MASSEY UNIVERSITY

SCHOOL OF ENGINEERING & ADVANCED TECHNOLOGY

Constraint Models for XML: Document Type Definitions (DTDs)

Kuda Dube

E-mail: k.dube@massey.ac.nz

Office: AHA3.74, **Telephone:** 06 951 7145, **Mobile:** 021 237 6613

Topic Outline

1. Topic Aim
2. Learning Objectives
3. Constraint Models for XML Documents
4. Syntax for Document Type Definitions (DTDs)
5. Exercise 2.1:
   - *Defining a DTD for an E-mail Message (from last lecture)*

Lecture Aim

To present a method for defining a custom markup language in XML by using the ***Document Type Definitions*** as one instance of a constraint model for XML vocabularies

Learning Objectives

At the end of this lecture, you must be able to:

1. Describe the syntactic elements of DTDs;
2. Design a markup language in XML;
3. Design a schema for a mark-up language;
4. Specify a schema for your markup language by using the DTD language;
5. Validate an XML document against the rules specified in the DTD-style schemas;
6. Discuss the pros and cons of DTDs;

Introduction: Working with DTDs

- A DTD or *Document Type Definition,* is a set of rules/constraints that defines a custom markup language in XML;
- Identifies elements and their attributes;
- An XML document that violates DTD rules is not valid for the DTD – the validation test;
- You will use an XML Editor or DTD processor to validate an XML document against a given DTD.

Introduction: Working with DTDs

XML Document

```
<?xml version="1.0"?>
<!DOCTYPE wonder SYSTEM "06-02.dtd">
<wonder>
      <name>Colossus of Rhodes</name>
      <location>Greece</location>
       <height>107</height>
</wonder>
```

This XML document has 4 elements:

- Root element: *wonder*
- Three children: *name, location and height*

DTD for the XML document

```
<!ELEMENT wonder (name,
   location, height)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT location (#PCDATA)>
<!ELEMENT height (#PCDATA)>
```

- This DTD defines the structure of the XML document;
- The DTD Reads: *The* **wonder** *element contains 3 child elements:* **name**, **location** *and* **height**; *the* **name** *element is of type PCDATA, the* **location** *element is of type PCDATA and the* **height** *element is of type PCDATA.*
- PCDATA = text

Defining an Element that Contains Text

XML Document

```
<?xml version="1.0"?>
<!DOCTYPE ancient_wonders SYSTEM "06-04.dtd">
<ancient_wonders>
<wonder>
<name language="English">Colossus of Rhodes</name>
<name language="Greek">Κολοσσός της Ρόδου</name>
<location>Rhodes, Greece</location>
<height units="feet">107</height>
<history>
<year_built era="BC">282</year_built>
<year_destroyed era="BC">226</year_destroyed>
<how_destroyed>earthquake</how_destroyed>
<story>In 294 BC, the people of the island of Rhodes ...</story>
</history>
</wonder>
</ancient_wonders>
```

DTD fragment

```
<!ELEMENT name (#PCDATA)>
<!ELEMENT location (#PCDATA)>
<!ELEMENT height (#PCDATA)>
<!ELEMENT year_built (#PCDATA)>
<!ELEMENT year_destroyed (#PCDATA)>
<!ELEMENT how_destroyed (#PCDATA)>
<!ELEMENT story (#PCDATA)>
```

- PCDATA – parsed character data. *Used to indicate that the element contains text.*
- This character data will be parsed or analysed by an XML processor.
- NB: everything is case sensitive

Defining an Empty Element

XML Document

```
<?xml version="1.0"?>
<!DOCTYPE ancient_wonders SYSTEM "06-06.dtd">
<ancient_wonders>
 <wonder>
  <name language="English">Colossus of Rhodes</name>
  <name language="Greek">Κολοσσός της Ρόδου</name>
  <location>Rhodes, Greece</location>
  <height units="feet">107</height>
  <history>
   <year_built era="BC">282</year_built>
   <year_destroyed era="BC">226</year_destroyed>
   <how_destroyed>earthquake</how_destroyed>
   <story>In 294 BC, the people of the island of Rhodes ...</story>
  </history>
<!-- ** XML2e ** -->
<!-- to follow the opposite DTD exerpt, see the section below -->
  <main_image file="lighthouse.jpg" w="528" h="349“ />
  <source sectionid="112" newspaperid="53“ />
 </wonder>
</ancient_wonders>
```

DTD Excerpt

```
<!-- ** XML2e ** --><!-- to follow the opposite XML
      document, see the section below -->
<!ELEMENT main_image EMPTY>
<!ELEMENT source EMPTY>
<!ELEMENT name (#PCDATA)>
<!ELEMENT location (#PCDATA)>
<!ELEMENT height (#PCDATA)>
<!ELEMENT year_built (#PCDATA)>
<!ELEMENT year_destroyed (#PCDATA)>
<!ELEMENT how_destroyed (#PCDATA)>
<!ELEMENT story (#PCDATA)>
```

The main_image and source elements are both empty elements.

It does’nt matter whether they use a single of separate opening and closing tags, they are both empty.

Defining an Element that Contains a Child

XML Document

```
<?xml version="1.0"?>
<!DOCTYPE ancient_wonders SYSTEM "06-08.dtd">
<ancient_wonders>
 <wonder>
  <name language="English">Colossus of Rhodes</name>
  <name language="Greek">Κολοσσός της Ρόδου</name>
  <location>Rhodes, Greece</location>
  <height units="feet">107</height>
  <history>
   <year_built era="BC">282</year_built>
   <year_destroyed era="BC">226</year_destroyed>
   <how_destroyed>earthquake</how_destroyed>
   <story>In 294 BC, the people of the island of Rhodes ...</story>
  </history>
  <main_image file="lighthouse.jpg" w="528" h="349"/>
  <source sectionid="112" newspaperid="53"/>
 </wonder>
</ancient_wonders>
```

DTD excerpt

```
<!ELEMENT ancient_wonders (wonder)>
```

This DTD defines that:

*The **ancient_wonders** element can contain a single element named **wonder**.*

The **wonder** element's contents depends on its definition only and are not affected by the **ancient_wonders** definition in the least.

Defining an Element that Contains a *Children*

XML Document

```
<?xml version="1.0"?>
<!DOCTYPE ancient_wonders SYSTEM "06-10.dtd">
<ancient_wonders>
 <wonder>
  <name language="English">Colossus of Rhodes</name>
<!-- ** XML2e ** -->
<!-- to follow this example, see the section below -->
<!-- I've commented out the second name element to validate the XML
       against the DTD -->
<!-- <name language="Greek">Κολοσσός της Ρόδου</name> -->
  <location>Rhodes, Greece</location>
  <height units="feet">107</height>
  <history>
   <year_built era="BC">282</year_built>
   <year_destroyed era="BC">226</year_destroyed>
   <how_destroyed>earthquake</how_destroyed>
   <story>In 294 BC, the people of the island of Rhodes ...</story>
  </history>
  <main_image file="lighthouse.jpg" w="528" h="349"/>
  <source sectionid="112" newspaperid="53"/>
 </wonder>
</ancient_wonders>
```

DTD Fragment

```
<!ELEMENT wonder (name, location,
     height, history, main_image, source)>
```

The DTD fragments validates the XML elements in bold in the opposite XML document.

It says that:

1. *The wonder element must contain each one of the listed elements, in the order they appear.*
2. *The wonder element may not contain anything else*

Defining how many occurrences

Use of Quantifiers

```
<!ELEMENT ancient_wonders (wonder+)>
<!ELEMENT wonder (name+, location, height, history,
      main_image, source*)>
```

- The special symbols used for occurrences are called quantifiers;
- Quantifiers make the definition much more flexible;
- The DTD fragment says:

1. The **ancient_wonders** element must contain at least one (and unlimited number of) **wonder** elements;
2. The **wonder** element must contain at least one **name** elements and there may be any number of **sourse** elements (including none);
3. The **location**, **height** and **main_image** must all appear exactly one (also the default)

The optional quantifier

```
<!ELEMENT history (year_built, year_destroyed?,
      how_destroyed?, story)>
```

This definition of the history element says that:

1. The **history** element must contain exactly one each of the **year_built** and **story** elements;
2. The **year_destroyed** and **how_destroyed** elements may be omitted or may appear at most one time.

Summary on the DTD Quantifiers

DTD cardinality operators are as follows:

- **?**: appears zero times or once
- ***:** appears zero or more times
- **+**: appears one or more times
- No cardinality operator: means exactly once (this is the default)

Defining Choices

XML

```
<?xml version="1.0"?><!DOCTYPE
   ancient_wonders SYSTEM "06-
   14.dtd">
<ancient_wonders>
<wonder>    <name>Colossus of
   Rhodes</name>  <location>Rhodes,
   Greece</location>
</wonder>
 <wonder>    Great Pyramid of Giza,
   Giza, Egypt
 </wonder>
<wonder>    Temple of Artemis at
   Ephesus    <city>Ephesus</city>
   <country>Turkey</country>
</wonder>
</ancient_wonders>
```

DTD

```
<!ELEMENT ancient_wonders (wonder+)>
<!-- ** XML2e ** --><!-- to follow this
example, see the bold section below -->
<!ELEMENT wonder (#PCDATA | name | location
| city | country)*>
<!ELEMENT name (#PCDATA)>
<!ELEMENT location (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT country (#PCDATA)>
```

This DTD uses choices to support the different structures of the wonder element.
It declares that:

> *The* **wonder** *element can contain zero or more occurrences of* **PCDATA**, **name**, **location**, **city**, *or* **country** *elements.*

Element that Contain Anything

XML

```
<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE
    ancient_wonders SYSTEM "06-17.dtd">
<ancient_wonders>
 <wonder>    <name>Colossus of Rhodes</name>
    <location>Rhodes, Greece</location> </wonder>
    <wonder>    Great Pyramid of Giza, Giza,
    Egypt
</wonder>  <
wonder>    Temple of Artemis at Ephesus
    <city>Ephesus</city>
    <country>Turkey</country>
</wonder>
<wonder>
<name>Mausoleum at Halicarnassus</name>
<location>
    <city>Bodrum</city>
    <country>Turkey</country>
</location>
</wonder>
</ancient_wonders>
```

DTD

```
<!ELEMENT ancient_wonders
   (wonder+)>
<!ELEMENT wonder (#PCDATA |
   name | location | city |
   country)*>
<!ELEMENT name (#PCDATA)>
<!ELEMENT location ANY>
<!ELEMENT city (#PCDATA)>
<!ELEMENT country
   (#PCDATA)>
```

DTD: Attribute Types

Similar to predefined data types, but limited selection;

The most important types are:

- **CDATA**, a string (sequence of characters)
- **ID**, a name that is unique across the entire XML document
- **IDREF**, a reference to another element with an ID attribute carrying the same value as the IDREF attribute
- **IDREFS**, a series of IDREFs
- **(v1| . . . |vn)**, an enumeration of all possible values

Limitations: no dates, number ranges etc.

DTD: Attribute Value Types

#REQUIRED

- Attribute must appear in every occurrence of the element type in the XML document

#IMPLIED

- The appearance of the attribute is optional

#FIXED "value"

- Every element must have this attribute

"value"

- This specifies the default value for the attribute

Defining Attributes

XML

```
<?xml
version="1.0"?><!DOCTYPE
wonder SYSTEM "06-19.dtd">
<wonder>
<!-- to follow this example,
see the section below -->
<height>39</height>
<height
units="feet">39</height>
<height
units="39">feet</height>

</wonder>
```

DTD

```
<!ELEMENT wonder (height*)>
<!-- to follow this
   example, see the section
   below -->
<!ELEMENT height (#PCDATA)>
<!ATTLIST height
          units CDATA
   #IMPLIED>
```

All these are valid since the **units** attribute is optional (#IMPLIED) and its contents may be any combination of characters

Declaring an External DTD

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE wonder SYSTEM "08-01.dtd">
<wonder>
    <name>Colossus of Rhodes</name>
    <name language="Greek">Κολοσσός της Ρόδου</name>
    <location>Greece</location>
    <height>107</height>
</wonder>
```

Exercise 2: The E-mail message DTD

From the design of the structure of an e-mail outlined in the last lecture, write a DTD for an e-mail message.

The End: *Nest Topic …*

Constraint Models for XML Documents:

- *XML Schema*