Chapter 4: Document Type Definitions

Chapter 4 Objectives

- Learn to create DTDs
- Validate an XML document against a DTD
- Use DTDs to create XML documents from multiple files

What's in a Name?

- DTD can be
  - Internal: embedded in the document
  - External

- Example: name2.xml

```
<?xml version="1.0"?>
<!DOCTYPE name [
  <!ELEMENT name (first, middle, last)>
  <!ELEMENT first (#PCDATA)>
  <!ELEMENT middle (#PCDATA)>
  <!ELEMENT last (#PCDATA)>
]>
<name>
  <first>John</first>
  <middle>Fitzgerald Johansen</middle>
  <last>Doe</last>
</name>
```

What's in a Name?

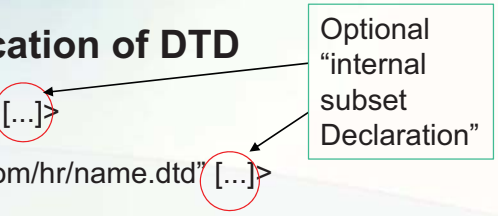
- DTD: Document Type Definition
- DOCTYPE: Document Type Declaration
  - informs parser that a DTD associated w/ this XML
  - must appear at the beginning
    - – preceded only by XML declaration

- Another example: name3.xml

```
<?xml version="1.0"?>
<!DOCTYPE name [
  <!ELEMENT name (first, middle, last)>
  <!ELEMENT first (#PCDATA)>
  <!ELEMENT middle (#PCDATA)>
  <!ELEMENT last (#PCDATA)>
]>
<name>
  <given>John</given>
  <middle>Fitzgerald Johansen</middle>
  <last>Doe</last>
</name>
```

External Document Type Declaration

System Identifiers – specifying location of DTD



<!DOCTYPE name SYSTEM “file:///c:/name.dtd” [...]>

<!DOCTYPE name SYSTEM “http://www.wiley.com/hr/name.dtd” [...]>

<!DOCTYPE name SYSTEM “name.dtd”>

Public Identifiers – identifying entry in catalog

<!DOCTYPE name PUBLIC “-//Beginning XML//DTD Name Example//EN”>

<!DOCTYPE name PUBLIC “-//Beginning XML//DTD Name Example//EN” “name.dtd”>

Formal Public identifier (FPI) format:
registered with ISO?(+ or -)//owner//class_description//language

E.g.: -//W3C/DTD XHTML 1.0 Strict//EN

“Plan B”

External DTD

name4.xml

```
<?xml version="1.0"?>
<!DOCTYPE name PUBLIC "-//Beginning XML//DTD Name Example//EN"
"name4.dtd">
<name>
  <first>John</first>
  <middle>Fitzgerald Johansen</middle>
  <last>Doe</last>
</name>
```

name4.dtd

```
<!ELEMENT name (first, middle, last)>
<!ELEMENT first (#PCDATA)>
<!ELEMENT middle (#PCDATA)>
<!ELEMENT last (#PCDATA)>
```

Sharing Vocabularies

- Use your favorite search engine
- Cover Pages
  - http://xml.coverpages.org
- Dublin Core Metadata Initiative
  - http://www.dublincore.org

Anatomy of a DTD

Element declarations

Attribute declarations

Entity declarations

Element Declarations

<!ELEMENT name (first, middle, last)>

Element declarations consist of three basic parts:

- The ELEMENT declaration
- The element name
- The element content model

Element Content

<!ELEMENT contact (name, location, phone, knows, description)>

Or

<!ELEMENT contact (name)>

Sequences

<!ELEMENT name (first, middle, last)>

```
<name>
  <first>John</first>
  <middle>Fitzgerald Johansen</middle>
  <last>Doe</last>
</name>
```

Order is important

Choices

<!ELEMENT location (address | GPS)>

Now you can choose address or GPS as content

Combining Sequences and Choice Using Groups

<!ELEMENT location (address | (latitude, longitude))>

Combining sequences and choices using groups

How about empty content?

<middle></middle> **<!ELEMENT middle EMPTY>**
or
<middle/>

Mixed Content

- Example:
 <description>XML is fun

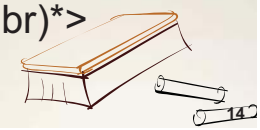
to learn!</description>
- They must use the choice mechanism (the vertical bar | character) to separate elements.
- The #PCDATA keyword must appear first in the list of elements.
- There must be no inner content models.
- If there are child elements the `*' cardinality indicator must appear at the end of the model.

In its simplest form

<!ELEMENT first (#PCDATA)>

For our example

<!ELEMENT description (#PCDATA | em | strong | br)*>



Cardinality

Indicator	Description
[none]	When no cardinality indicator is used, it indicates that the element must appear ***once and only once***. This is the default behavior for elements used in content models.
?	Indicates that the element may appear ***either once or not at all***.
+	Indicates that the element may appear ***one or more times***.
*	Indicates that the element may appear ***zero or more times***.

Any Contents

<!ELEMENT description ANY>

Any element declared within the DTD can be used within the content of the description element in any order

Try It Out

"Making Contact" Part 1
"Making Contact" Part 2

Attribute Declarations

```
<!ELEMENT contacts (contact*)>
<!ATTLIST    contacts source CDATA #IMPLIED>
```

An ATTLIST declaration consists of three basic parts:

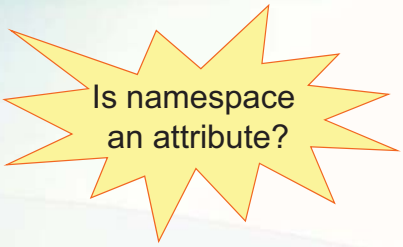
* The ATTLIST keyword
* The associated element's name
* The list of declared attributes

* The attribute name
* The attribute type
* The attribute value declaration

Example: source CDATA #IMPLIED

Attribute Declarations

- Passage in the book.



As far as DTDs are concerned, namespace declarations, such as xmlns:contacts= "http://wiley.com/contacts", are also treated as attributes. Although the Namespace Recommendation insists that xmlns statements are declarations and not attributes, DTDs must declare them in an ATTLIST declaration if they are used. Again, this is because the W3C finalized the syntax for DTDs before the Namespace Recommendation was completed.

Attribute Types

Type	Description
CDATA	Indicates the attribute value is character data
ID	uniquely identifies the containing element
IDREF	a reference, by ID, to a uniquely identifiable element
IDREFS	a whitespace-separated list of IDREF values
ENTITY	a reference to an external unparsed entity (we will learn more about entities later). The unparsed entity might be an image file or some other external resource such as an MP3 or some other binary file
ENTITIES	a whitespace-separated list of ENTITY values
NMTOKEN	a name token. An NMTOKEN is a string of character data consisting of standard name characters
NMTOKENS	a whitespace-separated list of NMTOKEN values
Enumerated List	declare an enumerated list of possible values for the attribute

Attribute Value Declarations

Has a default value

<!ATTLIST phone kind (Home | Work | Cell | Fax) "Home">

Has a fixed value

<!ATTLIST contacts version CDATA #FIXED "1.0">

Is required

<!ATTLIST phone kind (Home | Work | Cell | Fax) #REQUIRED>

Is implied – none of above, optional

<!ATTLIST knows contacts IDREFS #IMPLIED>

Specifying Multiple Attributes

<!ATTLIST contacts version CDATA #FIXED “1.0”
source CDATA #IMPLIED>

Either style is legal

<!ATTLIST contacts version CDATA #FIXED “1.0”>
<!ATTLIST contacts source CDATA #IMPLIED>

Entities

* Built-in entities
* Character entities
* General entities
* Parameter entities

Built-in Entities

```
<contacts source="Beginning XML 4E&apos;s Contact List" version="1.0">
<description>Jeff is a developer &amp; author for Beginning XML <em>4th
    edition</em> &#169; 2006 Wiley Publishing.<br/>Jeff
    <strong>loves</strong> XML!
</description>
```

&amp;	—the & character
&lt;	—the < character
&gt;	—the > character
&apos;	—the ‘ character
&quot;	—the “ character

Character Entities

© the © character

General Entities

Internal (declared in DTD)

<!ENTITY source-text "Beginning XML 4E's Contact List">

External (declared in DTD)

<!ENTITY jeff-description SYSTEM "jeff.txt">

or

<!ENTITY jeff-description PUBLIC
"-//Beginning XML//Jeff Description//EN" "jeff.txt">

Must be declared within the DTD

Referencing (in XML document)

&jeff-description;

"String substitution" performed when referenced

Parameter Entities

- To create reusable sections of replacement text
- Build DTD's from multiple files
- The space between % and NameDeclaration in the example is intentional when defining.
- The % next to the NameDeclaration, like %NameDeclaration is to reference it.
- Referenced in DTD, not in XML

Example:

- **Definition**
  <!ENTITY % NameDeclarations SYSTEM "name5.dtd">
- **Referencing**
  %NameDeclarations;

Parameter Entities

Internal

<!ENTITY % DefaultPhoneKind "Home">

External: SYSTEM or PUBLIC

<!ENTITY % NameDeclarations SYSTEM "name5.dtd">
or
<!ENTITY % NameDeclarations
PUBLIC "-//Beginning XML 4E//DTD External module//EN" "name5.dtd">

Referencing Examples

%NameDeclarations;
<!ENTITY % DefaultPhoneKind ""Home"">
(or <!ENTITY % DefaultPhoneKind '"Home"'>)
<!ATTLIST phone kind (Home | Work | Cell | Fax) %DefaultPhoneKind;>

Try It Out

"Making Contact" Part 3
"Making Contact" Part 4
"Making Contact" Part 5

Developing DTDs

- Use example XML document
- Modularization
- Comments are important
  - The following is valid:

```
<!-- source : allows you to describe the source of the contacts list -->
<!ATTLIST contacts source CDATA #IMPLIED>
```

  - The following is not valid:

```
<!ATTLIST contacts
 <!-- source : allows you to describe the source of the contacts list -->
source CDATA #IMPLIED>
```

DTD Limitations

Some limitations of DTDs include:

- Differences between DTD syntax and XML syntax
- Poor support for XML namespaces
- Poor data typing
- Limited content model descriptions

Solution?