Validating XML Documents with DTDs

DTDs For Data Interchange

4.1: A Very Simple Internal DTD

```
<!DOCTYPE name [

<!ELEMENT name (#PCDATA)>

]>

<name>Xavier Ampul III</name>
```

About Example 4.1

- DTDs are a kind of **schema** for XML documents.
- Schemas define which elements and attributes can occur within an XML document and in what order.
- XML **instance documents** are bound to their **DTD** via the `DOCTYPE declaration`.
- DOCTYPE declarations must identify the root element.
- DOCTYPE declarations must either contain (**internal**) or point to (**external**) the text of the DTD.
- Example 4.1 has in internal DTD.
- The DTD language is not an XML application.

About Example 4.1 (last)

- The single **element declaration** specifies a **content model** of #PCDATA for parsed character data.
- Instance documents that conform to their schema are said to be **valid**.
- XML documents can be checked for validity using a **validating parser**.

Checking Validity with XMLSpy

- Click on the green checkmark icon, or
- Press F8.

4.2.xml: Adding Children

```
<!DOCTYPE name SYSTEM "ex_4.2.dtd">
<name>
  <firstname>Xavier</firstname>
  <lastname>Ampul</lastname>
  <suffix>III</suffix>
</name>
```

4.2.dtd: Adding Children

```
<!ELEMENT name (firstname, lastname, suffix)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT lastname (#PCDATA)>
<!ELEMENT suffix (#PCDATA)>
```

About Example 4.2

- Example 4.2 uses an external DTD.
- External DTDs exist in a separate file pointed to by the DOCTYPE declaration.
- The content model for an element with children is: `(child1, child2, . . . , childn)`
- These children must always appear.
- These children must always be in this order.
- In addition to listing each child in the content model, the child must also have its own element declaration.

4.3.xml: Optional & Repeating Elements

```
<!DOCTYPE employee SYSTEM "ex_4.3.dtd">
<employee>
  <name>
    <firstname>Xavier</firstname>
    <lastname>Ampul</lastname>
    <suffix>III</suffix>
  </name>
  <address>
    <street>1234 Main Street</street>
    <street>Apartment 401</street>
    <city>Chicago</city>
    <state>IL</state>
    <zip>60610</zip>
  </address>
</employee>
```

4.3.dtd: Optional & Repeating Elements

```
<!ELEMENT employee (name, address)>

<!ELEMENT name (firstname, middleinit?, lastname, suffix?)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT middleinit (#PCDATA)>
<!ELEMENT lastname (#PCDATA)>
<!ELEMENT suffix (#PCDATA)>

<!ELEMENT address (street+, city, state, zip)>
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT zip (#PCDATA)>
```

About Example 4.3

- DTDs do not specify a fixed root element.
- You need to specify the correct root element on the DOCTYPE declaration.
- The content model for an optional element is:
  `(elementName?)`
- The content model for an element that is optional and may repeat is:
  `(elementName*)`
- The content model for an element that is required and may repeat is:
  `(elementName+)`

Other Content Models

- The content model for a selection of elements is:
  `(element1 | element2 | . . . | elementn)`
- The content model for an empty element is:
  `EMPTY`
- A non-constraining content model sometimes used during DTD development is:
  `ANY`
- There is no way to constrain the value of element text using a DTD.

4.4.xml: Adding XML Declaration and <phone>

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE employee SYSTEM "ex_4.4.dtd">
<employee>
  <name>
    <firstname>Xavier</firstname>
    <lastname>Ampul</lastname>
    <suffix>III</suffix>
  </name>
  <address>
    <street>1234 Main Street</street>
    <street>Apartment 401</street>
    <city>Chicago</city>
    <state>IL</state>
    <zip>60610</zip>
  </address>
  <phone>555-555-5555</phone>
</employee>
```

4.4.dtd: Adding XML Declaration and <phone>

```
<!ELEMENT employee (name, address, phone*)>

<!ELEMENT name (firstname, middleinit?, lastname, suffix?)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT middleinit (#PCDATA)>
<!ELEMENT lastname (#PCDATA)>
<!ELEMENT suffix (#PCDATA)>

<!ELEMENT address (street+, city, state, zip)>
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT zip (#PCDATA)>

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

About Example 4.4

- The XML declaration has no counterpart in the DTD.
- You must always remember to give repeating elements like <phone>the proper content model even if they don't repeat in this particular instance document.

4.5.xml: Creating <employeelist>

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE employeelist SYSTEM "ex_4.5.dtd">
<employeelist>

  <employee>
    <name>
      <firstname>Xavier</firstname>
      <lastname>Ampul</lastname>
      <suffix>III</suffix>
    </name>
    <address>
      <street>1234 Main Street</street>
      <street>Apartment 401</street>
      <city>Chicago</city>
      <state>IL</state>
      <zip>60610</zip>
    </address>
    <phone>555-555-5555</phone>
  </employee>
```

4.5.xml: Creating <employeelist> (cont.)

```
<employee>
  <name>
    <firstname>Frances</firstname>
    <middleinit>R</middleinit>
    <lastname>Smith</lastname>
  </name>
  <address>
    <street>559 Primary Avenue</street>
    <city>Evanston</city>
    <state>IL</state>
    <zip>60201</zip>
  </address>
  <phone>555-555-5590</phone>
  <phone>555-555-5591</phone>
</employee>

<employee>
  <name>
    <firstname>Raymond</firstname>
    <lastname>Jones</lastname>
```

4.5.xml: Creating <employeelist> (last)

```
    </name>
    <address>
      <street>987 Center Street</street>
      <city>Chicago</city>
      <state>IL</state>
      <zip>60610</zip>
    </address>
    <phone>555-555-0987</phone>
  </employee>

</employeelist>
```

4.5.dtd: Creating <employeelist>

```
<!ELEMENT employeelist (employee*)>

<!ELEMENT employee (name, address, phone*)>

<!ELEMENT name (firstname, middleinit?, lastname, suffix?)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT middleinit (#PCDATA)>
<!ELEMENT lastname (#PCDATA)>
<!ELEMENT suffix (#PCDATA)>

<!ELEMENT address (street+, city, state, zip)>
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT zip (#PCDATA)>

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

About Example 4.5

- Always remember to put the proper root element name on the DOCTYPE.
- Make sure to include enough content in your instance document to fully test your DTD.

4.6.xml: Adding the type Attribute

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE employeelist SYSTEM "ex_4.6.dtd">
<employeelist>

  <employee>
    <name>
      <firstname>Xavier</firstname>
      <lastname>Ampul</lastname>
      <suffix>III</suffix>
    </name>
    <address>
      <street>1234 Main Street</street>
      <street>Apartment 401</street>
      <city>Chicago</city>
      <state>IL</state>
      <zip>60610</zip>
    </address>
    <phone type="home">555-555-5555</phone>
  </employee>
```

4.6.xml: Adding the `type` Attribute (cont.)

```
<employee>
  <name>
    <firstname>Frances</firstname>
    <middleinit>R</middleinit>
    <lastname>Smith</lastname>
  </name>
  <address>
    <street>559 Primary Avenue</street>
    <city>Evanston</city>
    <state>IL</state>
    <zip>60201</zip>
  </address>
  <phone type="home">555-555-5590</phone>
  <phone type="mobile">555-555-5591</phone>
</employee>

<employee>
  <name>
    <firstname>Raymond</firstname>
    <lastname>Jones</lastname>
```

4.6.xml: Adding the `type` Attribute (last)

```
    </name>
    <address>
      <street>987 Center Street</street>
      <city>Chicago</city>
      <state>IL</state>
      <zip>60610</zip>
    </address>
    <phone type="mobile">555-555-0987</phone>
  </employee>

</employeelist>
```

4.6.dtd: Adding the type Attribute

```
<!ELEMENT employeelist (employee*)>

<!ELEMENT employee (name, address, phone*)>

<!ELEMENT name (firstname, middleinit?, lastname, suffix?)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT middleinit (#PCDATA)>
<!ELEMENT lastname (#PCDATA)>
<!ELEMENT suffix (#PCDATA)>

<!ELEMENT address (street+, city, state, zip)>
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT zip (#PCDATA)>

<!ELEMENT phone (#PCDATA)>
<!ATTLIST phone
  type CDATA #IMPLIED>
```

About Example 4.6

- Attributes are added to the DTD using the `ATTLIST` declaration.
- Multiple attributes can be added to an element with a single ATTLIST declaration.
- The DTD language allows for liberal use of line continuations.
- The DTD language allows for declarations occurring in any order.
- Specifying the attribute **type** of `CDATA` allows any type of XML character content.
- Specifying `#IMPLIED` makes this attribute optional.

4.7.xml: Finishing Touches

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE employeelist SYSTEM "ex_4.7.dtd">
<employeelist>

  <!-- The contents of this document are
       confidential.  Distribution is restricted
       to Xampul Employees and other
       previously authorized persons.   -->

  <listdate>April 4, 2006</listdate>
  <listregion>Central</listregion>

  <employee>
    <name>
      <firstname>Xavier</firstname>
      <lastname>Ampul</lastname>
      <suffix>III</suffix>
    </name>
    <address>
      <street>1234 Main Street</street>
```

4.7.xml: Finishing Touches (cont.)

```
    <street>Apartment 401</street>
    <city>Chicago</city>
    <state>IL</state>
    <zip>60610</zip>
  </address>
  <phone type="home">555-555-5555</phone>
</employee>

<employee>
  <name>
    <firstname>Frances</firstname>
    <middleinit>R</middleinit>
    <lastname>Smith</lastname>
  </name>
  <address>
    <street>559 Primary Avenue</street>
    <city>Evanston</city>
    <state>IL</state>
    <zip>60201</zip>
  </address>
```

4.7.xml: Finishing Touches (last)

```
    <phone type="home">555-555-5590</phone>
    <phone type="mobile">555-555-5591</phone>
  </employee>

  <employee>
    <name>
      <firstname>Raymond</firstname>
      <lastname>Jones</lastname>
    </name>
    <address>
      <street>987 Center Street</street>
      <city>Chicago</city>
      <state>IL</state>
      <zip>60610</zip>
    </address>
    <phone type="mobile">555-555-0987</phone>
  </employee>

</employeelist>
```

4.7.dtd: Finishing Touches

```
<!ELEMENT employeelist (listdate, listregion, employee*)>

<!ELEMENT listdate (#PCDATA)>
<!ELEMENT listregion (#PCDATA)>

<!ELEMENT employee (name, address, phone*)>

<!ELEMENT name (firstname, middleinit?, lastname, suffix?)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT middleinit (#PCDATA)>
<!ELEMENT lastname (#PCDATA)>
<!ELEMENT suffix (#PCDATA)>

<!ELEMENT address (street+, city, state, zip)>
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT zip (#PCDATA)>

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

4.7.dtd: Finishing Touches (last)

```
<!ATTLIST phone
  type  (home | office | mobile | fax) #REQUIRED>
```

About Example 4.7

- When adding an element to the DTD, remember to both add it to the content specification and to declare the new element.
- An **enumeration** can be used to constrain the valid values of an attribute.
- The syntax for an enumeration is
  `(value1 | value2 | . . . | valuen)`
- Specifying `#REQUIRED` makes this attribute mandatory.
- Remember to create versions of the instance document that test for a missing attribute or invalid attribute values.