COMP 4021
Internet Computing

XML DTD / Schema

Document Type Definitions (DTD)

- DTD defines the structure of an XML document by imposing constraints
  - The set of legal elements
  - An attribute is required or not
  - A certain element can only exist within a specific element (e.g., <price> must be nested within <item>)
  - A certain element must exist or not (if must exist, exist once or more)

```
<!-- address.dtd -->
<!ELEMENT address (name, street, city, state, postal-code)>
<!ELEMENT name (title? first-name, last-name)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT first-name (#PCDATA)>
<!ELEMENT last-name (#PCDATA)>
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT postal-code (#PCDATA)>
```

Constraints in DTD (I)

```
<!ELEMENT address (name, city, state)>
```

- The <address> element must contain a <name>, a <city>, and a <state> element, in that order. All of the elements are required.

```
<!ELEMENT name (title?, first-name, last-name)>
```

- The <name> element contains an optional <title> element, followed by a mandatory <first-name> and a <last-name> element; the question mark means zero or one occurrence

```
<!ELEMENT addressbook (address+)>
```

- An <addressbook> element contains one or more <address> elements; plus sign means an item must appear at least once

Constraints in DTD (II)

```
<!ELEMENT private-addresses (address*)>
```

- A <private-addresses> element contains zero or more <address> elements; the asterisk indicates zero or more occurrences

```
<!ELEMENT name (title?, first-name, (middle-initial |
middle-name)?, last-name)>
```

- A <name> element contains an optional <title> element, followed by a <first-name> element, followed by zero or one of <middle-initial> or a <middle-name> element, followed by a <last-name> element; vertical bars indicate a list of choices

```
<!ELEMENT name ((title?, first-name, last-name) |
(surname, mothers-name, given-name))>
```

- The <name> element can contain one of two sequences:
  - An optional <title>, followed by a <first-name> and a <last-name>
  - A <surname>, a <mothers-name>, and a <given-name>.

Defining Attributes in DTD

- Define which attributes are required
- Define default values for attributes
- List all of the valid values for a given attribute

```
<!ELEMENT city (#PCDATA)>
<!ATTLIST city state CDATA #REQUIRED
               postal-code CDATA #REQUIRED>
```

```
<!ELEMENT city (#PCDATA)>
<!ATTLIST city state CDATA (AZ|CA|NV|OR|UT|WA) "CA">
```

XML Schema

- XML schemas are themselves XML documents
  - A schema can be processed just like any other document
  - You can converts an XML schema into a Web form complete with automatically generated JavaScript code to validate the input data
- XML schemas support more data types than DTDs
  - Most of the data types in a programming language are supported
- XML schemas are extensible
  - User-defined and derived data types are supported
- XML schemas have more expressive power
  - XML schemas can restrict a value to be no longer than 2 characters, or matching a regular expression, e.g., [0-9]{5}(-[0-9]{4})?

XML Schema Example (I)

- A new data type is defined with the <xsd:complexType> element

```
<xsd:element name="address">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="name"/>
      <xsd:element ref="street"/>
      <xsd:element ref="city"/>
      <xsd:element ref="state"/>
      <xsd:element ref="postal-code"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

```
<xsd:element name="title"       type="xsd:string"/>
<xsd:element name="first-Name"  type="xsd:string"/>
<xsd:element name="last-Name"   type="xsd:string"/>
<xsd:element name="street"      type="xsd:string"/>
<xsd:element name="city"        type="xsd:string"/>
```

XML Schema Example (II)

- Derived data type
- Strings restricted by regular expression

```
<xsd:element name="state">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:length value="2"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:element name="postal-code">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[0-9]{5}(-[0-9]{4})?"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

DTD vs XML Schema: Rich Typing

- XML Schema supports string, int, float, unsigned Long, byte, etc.

```
<!ELEMENT item
(prodName+,USPrice,shipDate?)
<!ATTLIST item partNum CDATA>
<!ELEMENT prodName (#PCDATA)>
<!ELEMENT USPrice (#PCDATA)>
<!ELEMENT shipDate (#PCDATA)>
```

```
<xsd:element name="item">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="prodName" type="xsd:string" maxOccurs="5"/>
      <xsd:element name="USPrice"  type="xsd:decimal"/>
      <xsd:element name="shipDate" type="xsd:date" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="partNum" type="SKU"/>
  </xsd:complexType>
</xsd:element>
<!-- Stock Keeping Unit, a code for identifying products -->
<xsd:simpleType name="SKU">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="\d{3}-[A-Z]{2}"/>
  </xsd:restriction>
</xsd:simpleType>
```

User-defined data types and sub-classing

DTD vs XML Schema: Constraints

- DTDs use ?, *, and +, to specify, respectively, "zero or one“, "zero or more“, and "one or more" occurrences
- XML Schema can specify min/max occurrence constraints

```
<!ELEMENT donut (#PCDATA)>
<!ELEMENT donutorder
```

(donut,donut,donut,donut,donut,donut,donut,donut?,donut?,donut?,donut?, donut?)

```
<xsd:element name="donutorder">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="donut" type="xsd:string"
                   minOccurs="7" maxOccurs="12" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

DTD vs XML Schema: Enumeration

- XML Schema allows enumeration in element contents

```
<xsd:simpleType name="shoe_color">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="red"/>
    <xsd:enumeration value="green"/>
    <xsd:enumeration value="blue"/>
    <xsd:enumeration value="yellow"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:element name="person" type="person_type">
  <xsd:attribute name="shoes" type="shoe_color"/>
</xsd:element>
```

```
<!ATTLIST person shoes (red | green | blue | yellow)>
```

DTD vs XML Schema: More …

- XML Schema allows:
  - Null Values
  - Primary and secondary keys
  - XML schema is in itself an XML document
  - DTDs primarily constraints the “structure” (nesting of elements) but XML schema constraints the contents as well
- XML schemas are usually bulky compared to DTDs

Take Home Message

- XML DTD is more document centric while XML Schema is more database centric
- They both define constraints on XML elements so as to make the data more exchangeable and understandable
- Although you can create any XML that is syntactically correct, you must define the DTD or Schema to make it understandable and sharable to other applications