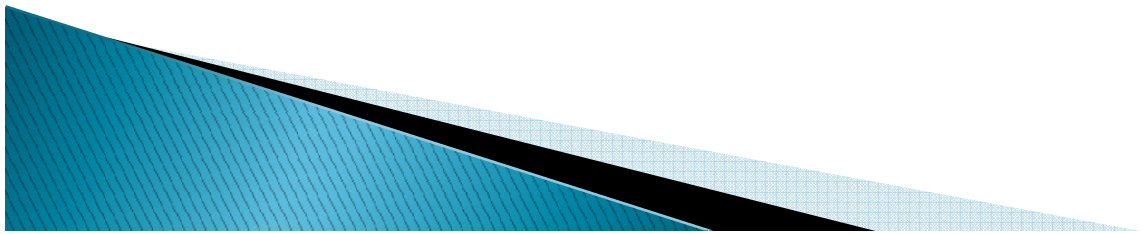
XML
Document Type Definitions
(DTD)

1.Introduction to DTD

- An XML document may have an *optional* DTD, which defines the document's grammar.

  Since the DTD defines the XML document's grammar, we can use an XML parser to check that if an XML document conforms to the grammar defined by the DTD.

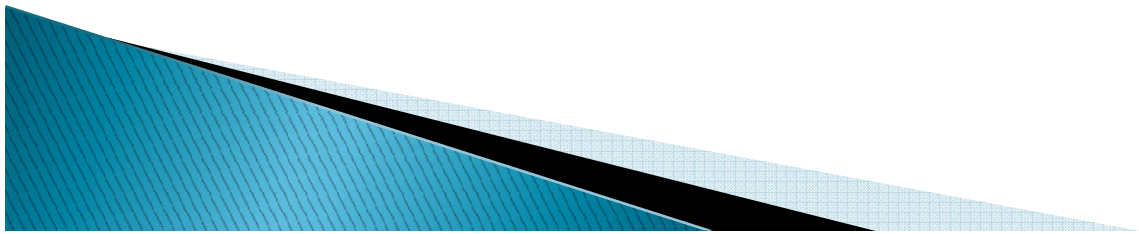


Well-Formed and Valid XML

- *Well-Formed XML* allows you to invent your own tags.
- *Valid XML* conforms to a certain DTD.

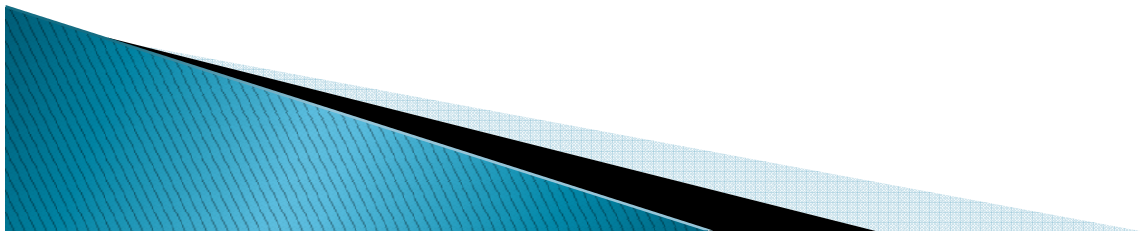
- The purpose of a DTD is to define the legal building blocks of an XML document.

Validation is useful in data exchange.



DTD Structure

```
<!DOCTYPE <root tag> [
  <!ELEMENT  <name>(<components>)>
  . . . more elements . . .
]>
```



Well-Formed XML

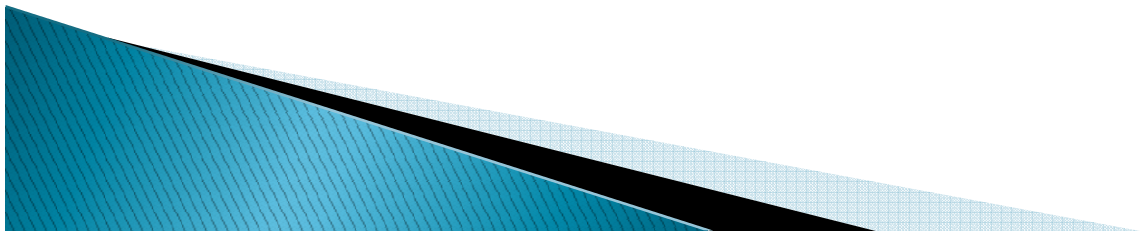
- Start the document with a *declaration*, surrounded by <?xml … ?> .
- Normal declaration is:

```
<?xml version = ”1.0” standalone =
  ”yes” ?>
```

  - “standalone” = “no DTD provided.”
- Balance of document is a *root tag* surrounding nested tags.

Use of DTD's

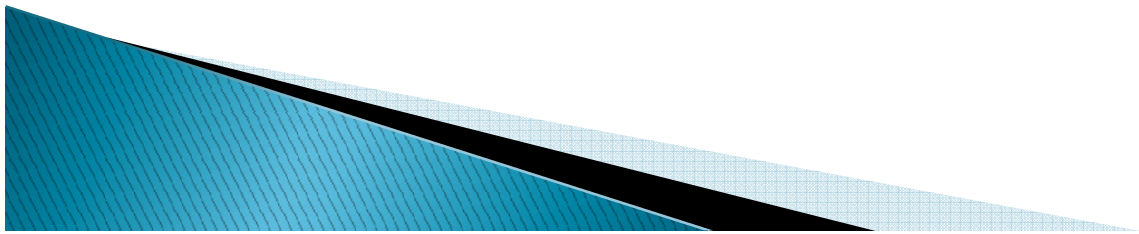
1. Set standalone = "no".
2. Either:
   a) Include the DTD as a preamble of the XML document, or
   b) Follow DOCTYPE and the <root tag> by SYSTEM and a path to the file where the DTD can be found.



- A DTD can be declared inside the XML document, or as an external reference.

1) *Internal DTD*

This is a example of a simple XML document with an internal DTD:



```
<!DOCTYPE company [
   < ! ELEMENT company    ( (person | product)*) >
   < ! ELEMENT person     ( ssn, name, office, phone?) >
   < ! ELEMENT ssn        ( # PCDATA) >
   < ! ELEMENT name       ( # PCDATA) >
   < ! ELEMENT office     ( # PCDATA) >
   < ! ELEMENT phone      ( # PCDATA) >
   < ! ELEMENT product    ( pid, name, description? ) >
   < ! ELEMENT pid        ( # PCDATA) >
   < ! ELEMENT description  ( # PCDATA) >
]>
<company>
<person> <ssn> 12345678 < /ssn>
            <name> John </name>
            <office> B432 </office>
            <phone> 1234 </phone>
</person>
<product> … </product>
…
</company>
```

2) *External DTD*

This is the same XML document with an external DTD:

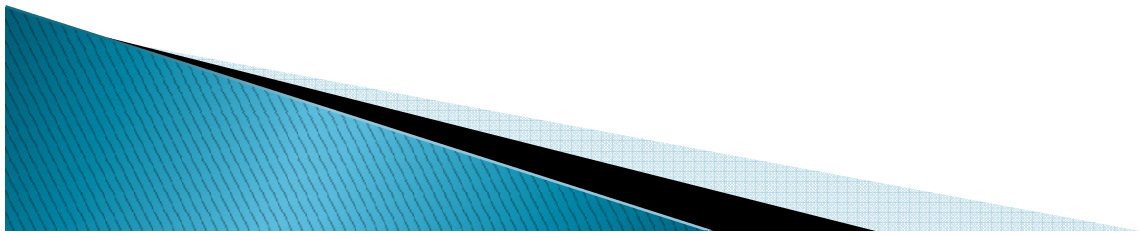
```
<!DOCTYPE company SYSTEM “company.dtd”>
<company>
<person> <ssn> 12345678 < /ssn>
            <name> John </name>
            <office> B432 </office>
            <phone> 1234 </phone>
</person>
<product> … </product>
…
</company>
```

The DTD can reside at a different URL, and then we refer to it as:

<!DOCTYPE company SYSTEM “http://www.mydtd.com/company.dtd”>

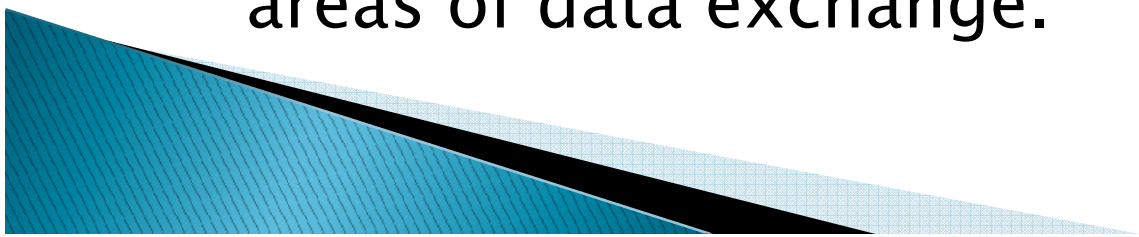
This is the file “company.dtd” containing the DTD:

```
< ! ELEMENT company    ( (person | product)*) >
< ! ELEMENT person      ( ssn, name, office,
  phone?)
< ! ELEMENT ssn          ( # PCDATA) >
< ! ELEMENT name        ( # PCDATA) >
< ! ELEMENT office       ( # PCDATA) >
< ! ELEMENT phone        ( # PCDATA) >
< ! ELEMENT product      ( pid, name, description? )
  >
< ! ELEMENT pid          ( # PCDATA) >
< ! ELEMENT description  ( # PCDATA) >
```



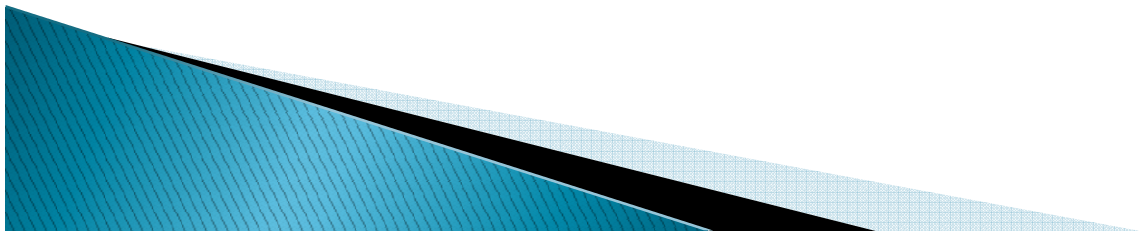
Why use a DTD?

- XML provides an application independent way of sharing data.
- With a DTD, independent groups of people can agree to use a common DTD for interchanging data.
- Your application can use a standard DTD to verify that data that you receive from the outside world is valid.
- You can also use a DTD to verify your own data.
- A lot of forums are emerging to define standard DTDs for almost everything in areas of data exchange.



2. DTD – XML building blocks

- The building blocks of XML documents:
  1) Elements: main building blocks.
  Example: “company”, “ person ” …
  2) Tags: are used to markup elements.
  3) Attributes: Attributes provide extra information about elements. Attributes are placed inside the starting tag of an element.
  Example:



4)PCDATA: parsed character data.

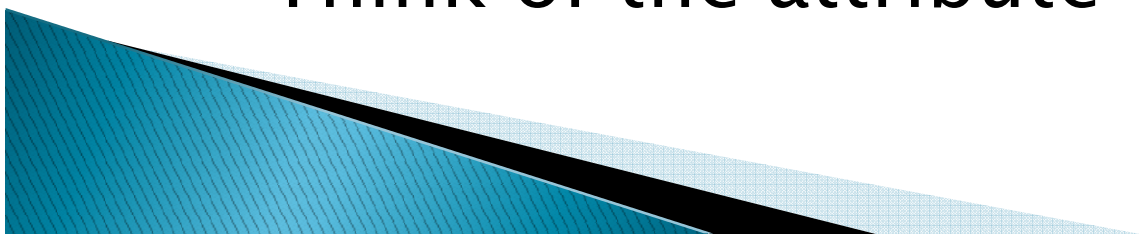
Think of character data as the *text* found between the starting tag and the ending tag of an XML element. <name> John</name>

PCDATA is text that will be parsed by a parser.

5)CDATA: character data.

CDATA is *text* that will NOT be parsed by a parser.

Think of the attribute value.

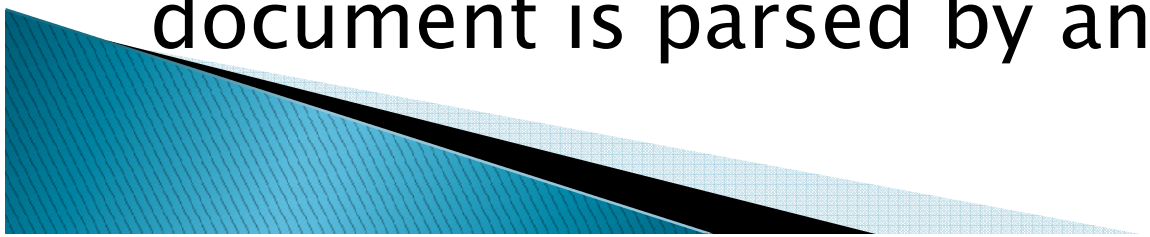


6) Entities: Entities as variables used to define common text. Entity references are references to entities.

The following entities are predefined in XML:

Entity References	Character
<	<
>	>
&	&
"	"
'	'

Entities are expanded when a XML document is parsed by an XML parser.



3.DTD – Elements

- In the DTD, XML elements are declared with an element declaration. An element declaration has the following syntax:

<!ELEMENT element-name (element-content)>

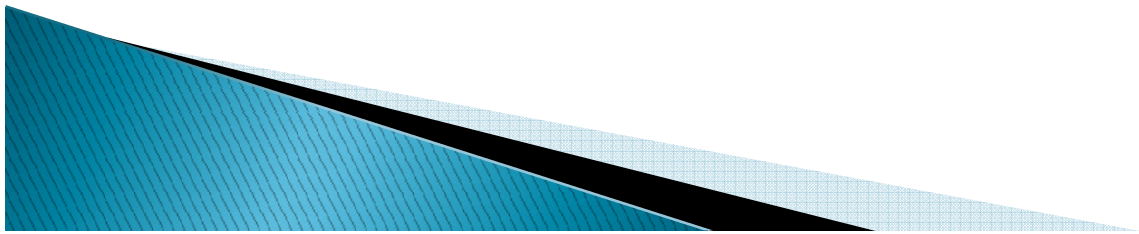
- Element-content model:

1) Empty Elements: keyword “EMPTY”.

<!ELEMENT element-name (EMPTY)>

example:

<!ELEMENT img (EMPTY)>



2) Text-only: Elements with text are declared:

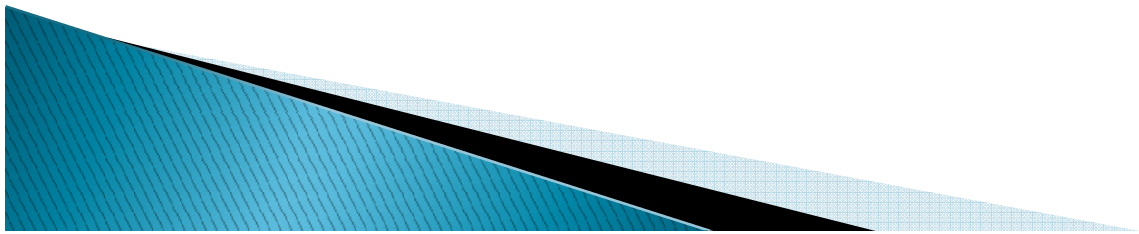
<!ELEMENT element-name (#PCDATA)>

example:

< ! ELEMENT name (# PCDATA) >

3) Any: keyword “ANY” declares an element with any content:

<!ELEMENT element-name (ANY)>

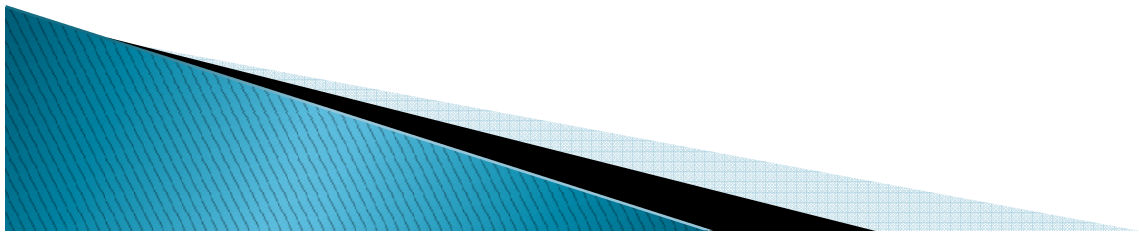


4) Complex: a regular expression over other elements.

Example:

< ! ELEMENT company ((person | product)*) >

Note: The elements in the regular expression must be defined in the DTD.

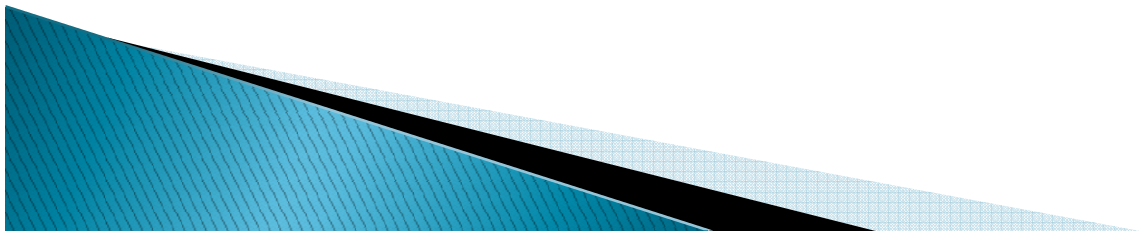


4) Mixed content:

Example:

<!ELEMENT note (to+,from,header,message*,#PCDATA)>

This example declares that the element **note** must contain at least one **to** child element, exactly one **from** child element, exactly one **header**, zero or more **message**, and some other **parsed character data** as well.



4.DTD – Attributes

In DTD, XML element attributes are declared with an ATTLIST declaration. Attribute declaration has the following syntax:

<!ATTLIST element-name attribute-name attribute-type default-value>

Example:

<! ELEMENT person (ssn, name, office, phone?) >

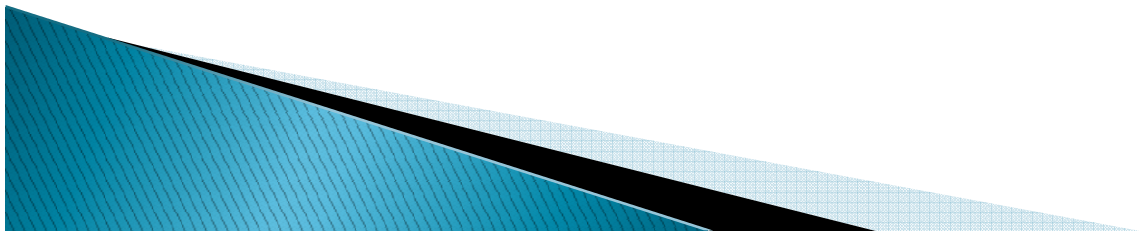
<! ATTLIST person age CDATA #REQUIRED) >

The attribute-type can have the following values:

- CDATA = string
- ID = key
- IDREF = foreign key
- IDREFS = foreign keys separated by space
- (val | val | …) = enumeration
- NMTOKEN = must be a valid XML name
- NMTOKENS = multiple valid XML names
- ENTITY = entity
- ENTITIES = a list of entities
- NOTATION = a name of a notation
- xml: = the value is predefined

The default-value can have the following values:

#DEFAULT value	The attribute has a default value
#REQUIRED	The attribute value must be included in the element
IMPLIED	The attribute is optional
FIXED value	The only value allowed



Attribute declaration examples

Example 1:

DTD example:

```
<!ELEMENT square EMPTY>
<!ATTLIST square width CDATA "0">
```

XML example:

```
<square width="100"></square>
```

or

```
<square width="100”/>
```

Example 2:

DTD example:

```
< !DOCTYPE family [
  < !ELEMENT family (person)* >
  < !ELEMENT person ( name ) >
  < !ELEMENT name (#PCDATA) >
  < !ATTLIST person id          ID
  #REQUIRED
                          mother   IDREF
  #IMPLIED
                          father    IDREF
  #IMPLIED
                          children IDREFS
  #IMPLIED
]>
```

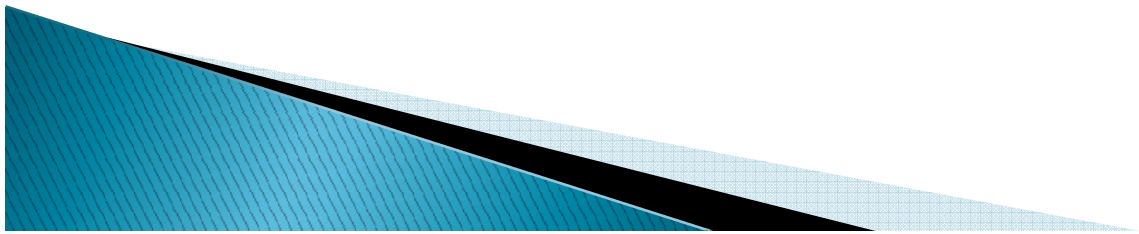
XML example:

```
<family>
 <person id = “jane”  mother = “mary”  father
  = “john”>
    <name> Jane Doe </name>
 </person>
 <person id = “john”  children = “jane jack”>
    <name> John Doe </name>
 </person>
 <person id = “mary”  children = “jane jack”>
   <name> Mary Smith </name>
 </person>
 <person id = “jack” mother = “smith”  father
  = “john”>
   <name> Jack Smith </name>
 </person>
</family>
```

5.DTD – Entities

Entities

- Entities as variables used to define shortcuts to common text.
- Entity references are references to entities.
- Entities can be declared internal.
- Entities can be declared external.



Internal Entity Declaration

Syntax:

```
<!ENTITY entity-name "entity-value">
```

DTD Example:

```
<!ENTITY writer "Jan Egil Refsnes.">
<!ENTITY copyright "Copyright
XML101.">
```

XML example:

```
<author>&writer;&copyright;</author>
```

External Entity Declaration

Syntax:

```
<!ENTITY entity-name SYSTEM "URL">
```

DTD Example:

```
<!ENTITY writer SYSTEM
 "http://www.xml101.com/entities/entit
 ies.xml">
<!ENTITY copyright SYSTEM
 "http://www.xml101.com/entities/entit
 ies.dtd">
```

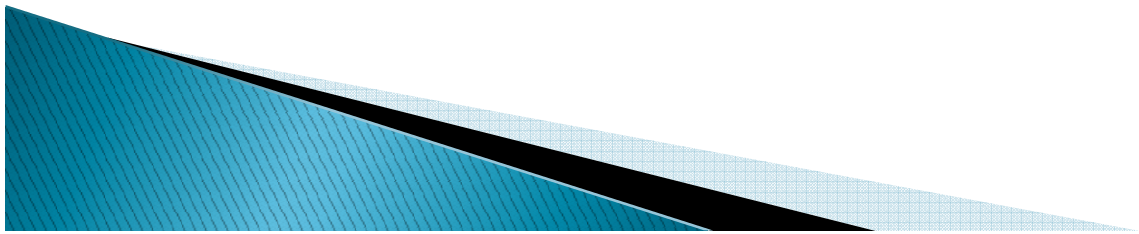
XML example:

```
<author>&writer;&copyright;</author>
```

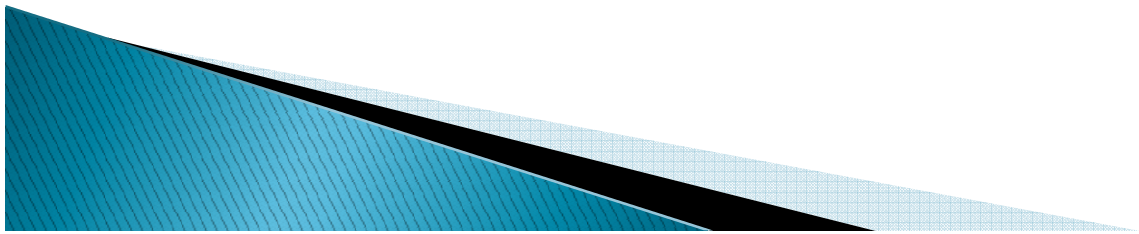
6.DTD Validation

How to test for DTD errors while loading XML document?

Since DTD is the grammar for XML, XML is a parse tree of its DTD. Then we can use a XML parser to check if the XML is valid.



7. DTD-examples



TV Scedule DTD

```
<!DOCTYPE TVSCHEDULE [

<!ELEMENT TVSCHEDULE (CHANNEL+)>
<!ELEMENT CHANNEL (BANNER, DAY+)>
<!ELEMENT BANNER (#PCDATA)>
<!ELEMENT DAY ((DATE, HOLIDAY) | (DATE, PROGRAMSLOT+))+>
<!ELEMENT HOLIDAY (#PCDATA)>
<!ELEMENT DATE (#PCDATA)>
<!ELEMENT PROGRAMSLOT (TIME, TITLE, DESCRIPTION?)>
<!ELEMENT TIME (#PCDATA)>
<!ELEMENT TITLE (#PCDATA)>
<!ELEMENT DESCRIPTION (#PCDATA)>

<!ATTLIST TVSCHEDULE NAME CDATA #REQUIRED>
<!ATTLIST CHANNEL CHAN CDATA #REQUIRED>
<!ATTLIST PROGRAMSLOT VTR CDATA #IMPLIED>
<!ATTLIST TITLE RATING CDATA #IMPLIED>
<!ATTLIST TITLE LANGUAGE CDATA #IMPLIED>
]>
```

Newspaper Article DTD

```
<!DOCTYPE NEWSPAPER [

<!ELEMENT NEWSPAPER (ARTICLE+)>
<!ELEMENT ARTICLE (HEADLINE, BYLINE, LEAD, BODY, NOTES)>
<!ELEMENT HEADLINE (#PCDATA)>
<!ELEMENT BYLINE (#PCDATA)>
<!ELEMENT LEAD (#PCDATA)>
<!ELEMENT BODY (#PCDATA)>
<!ELEMENT NOTES (#PCDATA)>

<!ATTLIST ARTICLE AUTHOR CDATA #REQUIRED>
<!ATTLIST ARTICLE EDITOR CDATA #IMPLIED>
<!ATTLIST ARTICLE DATE CDATA #IMPLIED>
<!ATTLIST ARTICLE EDITION CDATA #IMPLIED>

<!ENTITY NEWSPAPER "Vervet Logic Times">
<!ENTITY PUBLISHER "Vervet Logic Press">
<!ENTITY COPYRIGHT "Copyright 1998 Vervet Logic Press">

]>
```

Homework

Write XML documents that conforms to TV Schedule DTD OR Newspaper article DTD
Validate your document