XML DTD
Objectives

- To explain the main concepts of XML DTD (Data Type Definition)
Lecture outline

- Introduction
- Elements in DTD
- Names and namespaces
- An expanded DTD example
- Attributes and Entities in DTD
- Inline DTDs
- External DTDs
- Limitations of DTDs
- Validators
- Introduction

- XML and DTD
- Why DTD
- Parsers
- An XML example
- A DTD example
-- XML and DTDs

- A DTD (Document Type Definition) describes the structure of one or more XML documents.

- Specifically, a DTD describes:
  - Elements
  - Attributes, and
  - Entities

- An XML document is *well-structured* if it follows certain simple syntactic rules

- An XML document is *valid* if it also specifies and conforms to a DTD
-- Why DTDs?

- With DTD, each of your XML files can carry a description of its own format with it.

- 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 the data you receive from the outside world is valid.

- You can also use a DTD to verify your own data.
-- Parsers

- An *XML parser* is an API that reads the content of an XML document
  - Currently popular APIs are DOM (Document Object Model) and SAX (*S*imple *A*PI for *X*ML)
- A *validating parser* is an XML parser that compares the XML document to a DTD and reports any errors
-- An XML example

- `<novel>`
  - `<foreword>`
    - `<paragraph>` This is a great novel </paragraph>
  - `<chapter number="1">`
    - `<paragraph>` It was a dark and stormy night. </paragraph>
    - `<paragraph>` Suddenly, a shot rang out! </paragraph>
  - `</chapter>`
- `</novel>`

An XML document contains (and the DTD describes):
- Elements, such as novel and paragraph, consisting of tags and content
- Attributes, such as number="1", consisting of a name and a value
- Entities (not used in this example)
-- A DTD example

- `<!DOCTYPE novel [`
  - `<!ELEMENT novel (foreword, chapter+)`
  - `<!ELEMENT foreword (paragraph+)`
  - `<!ELEMENT chapter (paragraph+)`
  - `<!ELEMENT paragraph (#PCDATA)`
  - `<!ATTRIBUTE chapter number CDATA #REQUIRED>`
`]>

- A novel consists of a foreword and one or more chapters, in that order
  - Each chapter must have a number attribute

- A foreword consists of one or more paragraphs

- A chapter also consists of one or more paragraphs

- A paragraph consists of parsed character data (text that cannot contain any other elements)
- Elements in DTD

- Element description
- Elements without children
- Elements with unstructured child
- Elements with children
- Elements with mixed content
- ELEMENT descriptions

- Suffixes:
  - ?  optional foreword?
  - +  one or more chapter+
  - *  zero or more appendix*

- Separators:
  - ,  both, in order foreword?, chapter+
  - |  or section|chapter

- Grouping:
  - ()  grouping (section|chapter)+
- Elements without children

- The syntax is `<!ELEMENT name category>`
  - The `name` is the element name used in start and end tags
  - The `category` may be EMPTY:
    - In the DTD: `<!ELEMENT br EMPTY>`
    - In the XML: `<br></br>` or just `<br />`
    - In the XML, an empty element may not have any content between the start tag and the end tag
    - An empty element may (and usually does) have attributes
- Elements with unstructured children

- The syntax is <!ELEMENT name category>
  - The category may be ANY
    - This indicates that any content--character data, elements, even undeclared elements--may be used
    - Since the whole point of using a DTD is to define the structure of a document, ANY should be avoided wherever possible
  - The category may be (#PCDATA), indicating that only character data may be used
    - In the DTD: <!ELEMENT paragraph (#PCDATA)>
    - In the XML: <paragraph>A shot rang out!</paragraph>
    - The parentheses are required!
    - Note: In (#PCDATA), white space is kept exactly as entered
    - Elements may not be used within parsed character data
    - Entities are character data, and may be used
- Elements with children

- A category may describe one or more children:
  - `<!ELEMENT novel (foreword, chapter+)>`
  - Parentheses are required, even if there is only one child
  - A space must precede the opening parenthesis
  - Commas (, ) between elements mean that all children must appear, and must be in the order specified
  - “|” separators means any one child may be used
  - All child elements must themselves be declared
  - Children may have children
  - Parentheses can be used for grouping:
    - `<!ELEMENT novel (foreword, (chapter+|section+))>`
- Elements with mixed content

- #PCDATA describes elements with only character data

- #PCDATA can be used in an “or” grouping:
  - `<!ELEMENT note (#PCDATA|message)*>`

  This is called *mixed content*

- Certain (rather severe) restrictions apply:
  - #PCDATA must be first
  - The separators must be “|”
  - The group must be starred (meaning zero or more)
- Names and namespaces

- All names of elements, attributes, and entities, in both the DTD and the XML, are formed as follows:
  - The name must begin with a letter or underscore
  - The name may contain only letters, digits, dots, hyphens, underscores, and colons
  - The DTD doesn’t know about namespaces--as far as it knows, a colon is just part of a name
    - The following are different (and both legal):
      - `<!ELEMENT chapter (paragraph+)>
      - `<!ELEMENT myBook:chapter (myBook:paragraph+)>
    - Avoid colons in names, except to indicate namespaces
An expanded DTD example

```xml
<!DOCTYPE novel [
  <!ELEMENT novel (foreword, chapter+, biography?, criticalEssay*)>
  <!ELEMENT foreword (paragraph+)>
  <!ELEMENT chapter (section+|paragraph+)>
  <!ELEMENT section (paragraph+)>  
  <!ELEMENT biography(paragraph+)>
  <!ELEMENT criticalEssay (section+)>
  <!ELEMENT paragraph (#PCDATA)>
]
```
- Attributes and entities

- In addition to elements, a DTD may declare attributes and entities

- An attribute describes information that can be put within the start tag of an element
  - In XML: `<car name="Toyota" model="2001"></car>`
  - In DTD: `<!ATTLIST car
    name CDATA #REQUIRED
    model CDATA #IMPLIED >`

- An entity describes text to be substituted
  - In XML: `&copyright;`
  - In the DTD: `<!ENTITY copyright "Copyright KFUPM">`
-- Attributes

- The format of an attribute is:

```xml
<!ATTLIST element-name
    name type requirement
    name type requirement>
```

- where the name-type-requirement may be repeated as many times as desired

  - Note that only spaces separate the parts, so careful counting is essential
  - The element-name tells which element may have these attributes
  - The name is the name of the attribute
  - Each element has a type, such as CDATA (character data)
  - Each element may be required, optional, or “fixed”
  - In the XML, attributes may occur in any order
-- Important attribute types

- There are ten attribute types

- These are the most important ones:
  - CDATA The value is character data
  - (man|woman|child) The value is one from this list
  - ID The value is a unique identifier
    - ID values must be legal XML names and must be unique within the document
  - NM TOKEN The value is a legal XML name
    - This is sometimes used to disallow white space in the name
    - It also disallows numbers, since an XML name cannot begin with a digit
-- Less important attribute types

- **IDREF**  The ID of another element
- **IDREFS**  A list of other IDs
- **NM TOKENS**  A list of valid XML names
- **ENTITY**  An entity
- **ENTITIES**  A list of entities
- **NOTATION**  A notation
- **xml:**  A predefined XML value
-- Requirements

- Recall that an attribute has the form
  
  ```xml
  <!ATTLIST element-name name type requirement>
  ```

- The requirement is one of:
  - A default value, enclosed in quotes
    - Example: `<!ATTLIST degree CDATA "PhD">`
  - `#REQUIRED`
    - The attribute must be present
  - `#IMPLIED`
    - The attribute is optional
  - `#FIXED "value"`
    - The attribute always has the given value
    - If specified in the XML, the same value must be used
-- Entities

- There are exactly five predefined entities: <, >, &amp;, &quot;, and &apos;

- Additional entities can be defined in the DTD:
  - <!ENTITY copyright "Copyright KFUPM">

- Entities can be defined in another document:
  - <!ENTITY copyright SYSTEM "MyURI">

- Example of use in the XML:
  - This document is &copyright; 2002.

- Entities are a way to include fixed text (sometimes called “boilerplate”)

- Entities should not be confused with character references, which are numerical values between & and #
  - Example: &233; or &xE9; to indicate the character é
<?xml version="1.0"?>

<!DOCTYPE myXmlDoc SYSTEM "http://www.mysite.com/mydoc.dtd">
<weatherReport>
  <date>05/29/2002</date>
  <location>
    <city>Philadelphia</city>
    <state>PA</state>
    <country>USA</country>
  </location>
  <temperature-range>
    <high scale="F">84</high>
    <low scale="F">51</low>
  </temperature-range>
</weatherReport>
-- The DTD for this example

<!ELEMENT weatherReport (date, location, temperature-range)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT location (city, state, country)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT country (#PCDATA)>
<!ELEMENT temperature-range ((low, high)|(high, low))>
<!ELEMENT low (#PCDATA)>
<!ELEMENT high (#PCDATA)>
<!ATTLIST low scale (C|F) #REQUIRED>
<!ATTLIST high scale (C|F) #REQUIRED>
- Inline DTDs

- If a DTD is used only by a single XML document, it can be put directly in that document:

```xml
<?xml version="1.0">
<!DOCTYPE myRootElement [  <!-- DTD content goes here -->
  <!-- DTD content goes here -->
]>
<myRootElement>
  <!-- XML content goes here -->
</myRootElement>
```

- An inline DTD can be used only by the document in which it occurs
An external DTD (a DTD that is a separate document) is declared with a SYSTEM or a PUBLIC command:

```
<!DOCTYPE myRootElement SYSTEM "http://www.mysite.com/mydoc.dtd">
```

The name that appears after DOCTYPE (in this example, myRootElement) must match the name of the XML document’s root element.

Use SYSTEM for external DTDs that you define yourself, and use PUBLIC for official, published DTDs.

The file extension for an external DTD is .dtd.

External DTDs can only be referenced with a URL.

External DTDs are almost always preferable to inline DTDs, since they can be used by more than one document.
- Limitations of DTDs

- DTDs are a very weak specification language
  - You can’t put any restrictions on element contents
  - It’s difficult to specify:
    - All the children must occur, but may be in any order
    - This element must occur a certain number of times
  - There are only ten data types for attribute values

- But most of all: DTDs aren’t written in XML!
  - If you want to do any validation, you need one parser for the XML and another for the DTD
  - This makes XML parsing harder than it needs to be
  - There is a newer and more powerful technology: XML Schemas
  - However, DTDs are still very much in use
- Validators

- Opera 5 and Internet Explorer 5 can validate your XML against an internal DTD
  - IE provides (slightly) better error messages
  - Opera apparently just ignores external DTDs
  - IE considers an external DTD to be an error

- jEdit with the XML plugin will check for well structuredness and (if the DTD is inline) will validate your XML each time you do a Save
  - http://www.jedit.org/
- References

- W3School DTD Tutorial
  - http://www.w3schools.com/dtd/default.asp

- MSXML 4.0 SDK
  - http://www.topxml.com
  - http://www.xml.org
  - http://www.xml.com

- Several online presentations
- Reading List

- W3 Schools DTD Tutorial
  - http://www.w3schools.com/dtd/default.asp