



XML: Document Type Definitions

Qiang Yang

Thanks: Ethan Cerami
New York University

■ Road Map

- Introduction to DTDs
 - What's a DTD?
 - Why are they important?
 - What will we cover?
- Our First DTD
- Adding Attributes
- Multiple Elements
- Creating external DTDs



Introduction to DTDs

■ Definitions

- **Document Type Definition:** A set of rules for constructing an XML document.
- **Valid:** Documents that conform to a DTD are said to be valid.

■ Why are DTDs Important?

- In order to communicate, computers need to agree to specific rules.
- By agreeing to standards, lots of people/computers can share data.
- Once we can share data, we can create more powerful applications.

■ Case Study 1: Meekrat

- What is Meekrat?
 - "Open Wire Service" that collects news from multiple web sites.
 - Created by O'Reilly.com
- Uses an XML Document Type Definition, called RSS (Rich Site Summary)
- For Meekrat to work, all participating sites must adhere to the RSS DTD.
- <http://www.oreillynet.com/rss/>

■ Case Study: Meekrat

- RSS DTD was developed by Netscape for my.netscape.com
- Encapsulates specific article information, such as: Article Title, Article Description, Date, etc.
- By agreeing to follow the RSS DTD, hundreds of web sites can easily share data.

■ Example DTDs

- ICE (Information and Content Exchange): enables automatic transfer of data between two or more sites.
- XMLNews: standard for formatting news articles.
- cXML: Commerce XML; standard for E-Commerce transactions.

■ Example DTDs (Cont)

- Chemical Markup Language: used to represent chemical molecules.
- VoiceXML: used to encode text for voice recognition and synthesis.
- Open Trading Protocol (OTP)
- For a more complete list, go to:
http://www.xml.org/xmlorg_registry/index.shtml

■ Challenge in Creating DTDs

- In order to be useful, DTDs need to be widely used.
- Requires that companies/organizations collaborate on the creation of DTDs.
- Results can be slow and fractured.
- May result in competing standards.
- Nonetheless, many XML DTDs are likely to emerge within the very near future.

Our First DTD

■ Example 1: Internal DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DOCUMENT [
  <!ELEMENT DOCUMENT (#PCDATA)>
  <!ENTITY Description "This is entity content.">
]>
<DOCUMENT>
This is element text and an entity follows:&Description;
</DOCUMENT>
```

1. Prolog
2. Document Type Declaration
3. DTD Rules
4. XML Document Data

1. Prolog

- `<?xml version="1.0" encoding="UTF-8"?>`
- XML documents start with an XML prolog.
- Includes two major elements:
 - **version:** XML is currently in version 1.0
 - **encoding:** specifies the character encoding.

Character Encodings

- XML Parsers are required to support three types of encodings:
 - UTF-8: Unicode Transformation - 8 bit
 - UCS-2: Canonical Unicode Character site
 - UTF-16: Unicode format
- Why is this important?
 - Enables internationalization of software applications.

2. Document Type Declaration

`<!DOCTYPE DOCUMENT [`

Name of DTD
↙

- Indicates the start of the DTD.
- Specifies the name of the DTD. This name should match the name of the root element.

3. DTD Rules

- ```
<!ELEMENT DOCUMENT (#PCDATA)>
<!ENTITY Description "This is entity content.">
```
- **ELEMENT:** declares a specific element. Here, we declare a root element, called DOCUMENT.
  - **#PCDATA:** text data that does not contain any `&`, `<`, or `>` characters.
  - **ENTITY:** declares an abbreviation or a shortcut.
  - Whenever the parser sees `&Description`, it will replace it with "This is entity content."

## 4. XML Document Data

- ```
<DOCUMENT>
This is element text and an entity follows:&Description;
</DOCUMENT>
```
- Note that this document contains only one element: DOCUMENT, and it therefore adheres to the DTD.
 - The document is therefore valid.
 - When parsed by an XML Parser, the parser will replace `&Description` with, "This is entity content."

Validating your Document

- You can run your document through *an XML parser* to determine if the document is valid (in terms of the DTD)
- Simple web based validator:
- <http://www.stg.brown.edu/service/xmlvalid/>

Adding Attributes

Adding Attributes

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DOCUMENT [
<!ELEMENT DOCUMENT (#PCDATA)>
<!ATTLIST DOCUMENT
  trackNum CDATA #REQUIRED
  secLevel (unclassified|classified) "unclassified">
<!ENTITY Description "This is a very simple sample document.">
]>
<DOCUMENT trackNum="1234">This is element text and an entity
follows:&Description; </DOCUMENT>
```

Here, we have added two attributes to the Document element.

Attributes

- ```
<!ATTLIST ElementName
 AttributeName Type Default>
```
- When you create an attribute, you specify the associated element, the attribute name, the attribute type, and any attribute defaults.
  - Attribute Types:
    - CDATA: attribute may contain any text data.
    - (this one | that one): The value of the attribute must match one of the values listed.

## Attribute Defaults

- Attribute Defaults:
  - #REQUIRED: The attribute must be specified.
  - #IMPLIED: The attribute is optional.
  - #FIXED value: Attribute is preset to a specific value.
  - Defaultvalue: Provides a default value for the attribute.

## Attributes

- ```
<!ATTLIST DOCUMENT
  trackNum CDATA #REQUIRED
  secLevel (unclassified|classified) "unclassified">
<!ENTITY Description "This is a very simple sample document.">
]>
```
- Here, we specify that the trackNum attribute can contain any text and is required.
 - We also specify that secLevel can be set to one of two options: unclassified or classified. The default is unclassified.
 - Let's run it through our validator and see what happens...

Multiple Elements

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DOCUMENT [
<!ELEMENT DOCUMENT
  (TITLE,AUTHOR+,SUMMARY*,NOTE?)>
<!ATTLIST DOCUMENT
  trackNum CDATA #REQUIRED
  secLevel (unclassified|classified) "unclassified">
<!ELEMENT TITLE (#PCDATA)>
<!ELEMENT AUTHOR (#PCDATA)>
<!ELEMENT SUMMARY (#PCDATA)>
<!ELEMENT NOTE (#PCDATA)>
<ENTITY Description "This is a very simple sample document.">
]>
<DOCUMENT trackNum="1234">
<TITLE>Sample Document</TITLE>
<AUTHOR>Simon St. Laurent</AUTHOR>
<SUMMARY>This is element text and a
  follows:&Description; </SUMMARY>

```

Here, we have added three new elements to the DOCUMENT element. These elements must appear in this order.

Multiple Elements

```

<!ELEMENT DOCUMENT
  (TITLE,AUTHOR+,SUMMARY*,NOTE?)>

```

- When you declare an element, you specify:
 - the list of elements;
 - rules for which elements are required;
 - the sequence in which they appear;
 - how many times they may appear

Symbols for Element Structure

- | Any element may appear
- , Requires appearance in specified sequence
- ? Optional, but only one may appear
- * Allows any number to appear, even zero
- + Requires at least one to appear

Element Structure

```

<!ELEMENT DOCUMENT
  (TITLE,AUTHOR+,SUMMARY*,NOTE?)>

```

Here, we have:

- Rigid sequence: TITLE, AUTHOR, SUMMARY, NOTE
- TITLE (exactly one is required)
- AUTHOR (at least one must appear)
- SUMMARY (zero or more)
- NOTE (Zero or One)
- Let's try it on our validator...

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DOCUMENT [
<!ELEMENT DOCUMENT (TITLE,AUTHOR+,SUMMARY*,NOTE?)>
<!ATTLIST DOCUMENT
  trackNum CDATA #REQUIRED
  secLevel (unclassified|classified) "unclassified">
<!ELEMENT TITLE (#PCDATA)>
<!ELEMENT AUTHOR (FIRSTNAME, LASTNAME, (UNIVERSITY | COMPANY)?>
<!ELEMENT FIRSTNAME (#PCDATA)>
<!ELEMENT LASTNAME (#PCDATA)>
<!ELEMENT UNIVERSITY (#PCDATA)>
<!ELEMENT COMPANY (#PCDATA)>
<!ELEMENT SUMMARY (#PCDATA)>
<ENTITY Description "This is a very simple sample document.">
]>
<DOCUMENT trackNum="1234">
<TITLE>Sample Document</TITLE>
<AUTHOR>
  <FIRSTNAME>Simon</FIRSTNAME>
  <LASTNAME>St. Laurent</LASTNAME>
  <COMPANY>XML Mania</COMPANY>
</AUTHOR>
<SUMMARY>This is element text and an entity follows:
  &Description;
</SUMMARY></DOCUMENT>

```

Here, we have added four new elements to the AUTHOR element. These elements must appear in this order.

Creating External DTDs

Internal v. External DTDs

- Internal DTDs are really just for experimenting.
- You really want to separate your DTD into an external file.
- This enables multiple developers/companies to all use the same DTD.

The External DTD: simple.dtd

```
<!ELEMENT DOCUMENT (TITLE,AUTHOR+,SUMMARY*,NOTE?)>
<!ATTLIST DOCUMENT
    trackNum CDATA #REQUIRED
    secLevel (unclassified|classified) "unclassified">
<ELEMENT TITLE (#PCDATA)>
<ELEMENT AUTHOR (FIRSTNAME,LASTNAME, (UNIVERSITY |
COMPANY)?)>
<ELEMENT FIRSTNAME (#PCDATA)>
<ELEMENT LASTNAME (#PCDATA)>
<ELEMENT UNIVERSITY (#PCDATA)>
<ELEMENT COMPANY (#PCDATA)>
<ELEMENT SUMMARY (#PCDATA)>
<ENTITY Description "This is a very simple sample
document.">
```

XML Document

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DOCUMENT SYSTEM "simple.dtd">
<DOCUMENT trackNum="1234">
<TITLE>Sample Document</TITLE>
<AUTHOR><FIRSTNAME>Simon</FIRSTNAME>
<LASTNAME>St.Laurent</LASTNAME>
<COMPANY>XML Mania</COMPANY></AUTHOR>
<SUMMARY>This is element text and an entity
follows:&Description;
</SUMMARY></DOCUMENT>
```

↑
Here, we reference an
external DTD:
simple.dtd

More on External DTDs

- To reference a company wide DTD, use the keyword **SYSTEM**
 - To reference a Public DTD, use the keyword **PUBLIC**, and specify the URL of the DTD
- ```
<!DOCTYPE DOCUMENT SYSTEM "simple.dtd">
<!DOCTYPE rss PUBLIC "-//Netscape
Communications/DTD RSS 0.91/EN"
"http://my.netscape.com/publish/formats/rss-
0.91.dtd">
```