

XML DTDs

Section 6

April 16, 2009

Quick Review of XML

- [XML stands for eXtensible Markup Language

- [Similar syntax to HTML, but different purpose

- designed to carry data, not display it

- tags are user-defined

- data is stored as plain text

Well-Formed Documents

— [*A well-formed XML document has correct XML syntax*

— unique root element

— all elements must have closing tag

— tags are proper case

— elements must be properly nested

— attributes must be in quotes

Valid Documents

— [A “valid” XML document is one that is well-formed and conforms to the rules of a Document Type Definition (DTD)

— [Where in the XML document is this DTD referenced?

— Typically the DTD is in an external plain text file, and is imported like so:

— `<?xml version="1.0" encoding="ISO-8859-1"?>`

`<!DOCTYPE note SYSTEM "Note.dtd">`

`<note>`

`<to>Chris</to>`

`<from>Vlad</from>`

`<heading>Reminder</heading>`

`<body>Don't forget to bring the goods tonight.</body>`

`</note>`

— DOCTYPE definition has the following syntax: `<!DOCTYPE root-element SYSTEM "filename">`

The DTD

— [What might the DTD file for the note example look like?

```
— <!DOCTYPE note  
  [  
    <!ELEMENT note (to,from,heading,body)>  
    <!ELEMENT to (#PCDATA)>  
    <!ELEMENT from (#PCDATA)>  
    <!ELEMENT heading (#PCDATA)>  
    <!ELEMENT body (#PCDATA)>  
  ]>
```

— [What is the purpose of the DTD?

— with the DTD, each XML file carries a description of its own format

— allows standardization of data exchange

— facilitates data verification

DTD structure

— [In the context of DTDs, XML documents are made up of the following building blocks:

- Elements
- Attributes
- Entities
- PCDATA
- CDATA

Elements

— [We've seen these before. XHTML examples are `body`, `img`, `a`, `table`,...

— [main building blocks of XML documents

— [Can contain text, other elements, or be empty

Attributes

— [We've seen these before too. Examples are href (modifies a), src (modifies img), rowspan (modifies td),...

— [Give additional information about elements

— typically not directly related to element content

Entities

— [We've gone over these in the context of XHTML, but it is interesting to know that XML itself comes with 5 built in entities:

— < (<)

— > (>)

— & (&)

— " ("")

— ' ('')

CDATA and PCDATA

— [CDATA (*character data*) refers to data found between start and end tags of an XML element

— [PCDATA (*parsed character data*) is character data that will be parsed by the XML parser

— entities will be expanded and tags will be treated as markup

Declaring elements in DTD

Elements are declared with the following syntax:

`<!ELEMENT element-name category>` or
`<!ELEMENT element-name (element-content)>`

Empty elements are declared with the category keyword `EMPTY`

i.e. `<!ELEMENT br EMPTY>`

Elements containing only parsed character data are declared with `#PCDATA` inside parenthesis

i.e. `<!ELEMENT from (#PCDATA)>`

(continued)

— [Elements with the category **ANY** can contain any combination of parsable data

— [Elements with children are declared with the children names inside parenthesis

— `<!ELEMENT element-name (child1,child2,...)>`

— **IMPORTANT:** children must also be declared, and they must be declared in the sequence they are presented

(continued)

— [For exactly one occurrence of a child element:

— <!ELEMENT element-name (child-name)>

— [For at least one occurrence:

— <!ELEMENT element-name (child-name+)>

— [For zero or more:

— <!ELEMENT element-name (child-name*)>

— [0,1: <!ELEMENT element-name (child-name?)>

— [Either/or: <!ELEMENT note (to,from,header,(message | body))>

Attributes

— [Attribute declaration:

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

— default-value can be `#REQUIRED` (attribute required), `#IMPLIED` (attribute optional), or `#FIXED "value"` (has a value that author cannot change)

— Default attribute value:

— `<!ELEMENT square EMPTY>`

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

Entities

— [Internal entities are declared and used within the file

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

— [External entities are declared outside the DTD in a separate file:

— `<!ENTITY entity-name SYSTEM "URI/URL">`

Back to Note

```
— <!DOCTYPE note  
  [  
    <!ELEMENT note (to,from,heading,body)>  
    <!ELEMENT to (#PCDATA)>  
    <!ELEMENT from (#PCDATA)>  
    <!ELEMENT heading (#PCDATA)>  
    <!ELEMENT body (#PCDATA)>  
  ]>
```

Product Catalog DTD

```
<!DOCTYPE CATALOG [  
  
  <!ENTITY AUTHOR "John Doe">  
  <!ENTITY COMPANY "JD Power Tools, Inc.">  
  <!ENTITY EMAIL "jd@jd-tools.com">  
  
  <!ELEMENT CATALOG (PRODUCT+)>  
  <!ELEMENT PRODUCT (SPECIFICATIONS+,OPTIONS?,PRICE+,NOTES?)>  
  <!ATTLIST PRODUCT NAME CDATA #IMPLIED CATEGORY (HandTool | Table | Shop-Professional) "HandTool" PARTNUM CDATA #IMPLIED PLANT (Pittsburgh |  
  Milwaukee | Chicago) "Chicago" INVENTORY (InStock | Backordered | Discontinued) "InStock">  
  <!ELEMENT SPECIFICATIONS (#PCDATA)>  
  <!ATTLIST SPECIFICATIONS WEIGHT CDATA #IMPLIED POWER CDATA #IMPLIED>  
  
  <!ELEMENT OPTIONS (#PCDATA)>  
  <!ATTLIST OPTIONS FINISH (Metal | Polished | Matte) "Matte" ADAPTER (Included | Optional | NotApplicable) "Included" CASE (HardShell | Soft |  
  NotApplicable) "HardShell">  
  <!ELEMENT PRICE (#PCDATA)>  
  <!ATTLIST PRICE MSRP CDATA #IMPLIED WHOLESALE CDATA #IMPLIED STREET CDATA #IMPLIED SHIPPING CDATA #IMPLIED> <!ELEMENT NOTES  
  (#PCDATA)>  
  
>
```

Internal DTD

— [In some cases, you might want to define the DTD in the same XML document in which you use it

— [In this case, the DTD should be wrapped inside a DOCTYPE definition with the following syntax:

— `<!DOCTYPE root-element [element-declarations]>`

Internal DTD (Note)

```
<?xml version="1.0"?>  
<!DOCTYPE note [  
  <!ELEMENT note (to,from,heading,body)>  
  <!ELEMENT to (#PCDATA)>  
  <!ELEMENT from (#PCDATA)>  
  <!ELEMENT heading (#PCDATA)>  
  <!ELEMENT body (#PCDATA)>  
  
<note>  
  <to>Tove</to>  
  <from>Jani</from>  
  <heading>Reminder</heading>  
  <body>Don't forget me this weekend</body>  
</note>
```

Future of DTDs?

— [DTDs are quickly being replaced by XML Schemas

— [You'll learn about those tomorrow...

— and next week