Databases and Internet Applications

Part 1

Chapter 7.1-7.5

off the mark.com
by Mark Parisi
WOW, YOU'RE GOOD! I'M STILL ON MOON LEVEL...
MARK PARISI
offthemark.com
MarkParisi@aol.com
©2006 MARK PARISI DIST. BY UFS INC.

Uniform Resource Identifiers

- Uniform naming schema to identify *resources* on the Internet
- A resource can be anything:
  - Index.html
  - mysong.mp3
  - picture.jpg
- Example URIs:

  http://csbio.unc.edu/CCstatus
  mailto:webmaster@bookstore.com
  ftp://ftp.sanger.ac.uk/pub/

Structure of URIs

http://csbio.unc.edu/mcmillan/Comp521F16/Lecture14.pdf

- URI has three parts:
  - Name of the protocol used to access the resource (http)
  - Name of the host computer (csbio.unc.edu)
  - Name of the resource (mcmillan/Comp521F16/Lecture14.pdf)
- URLs are a subset of URIs
  - URL (*Universal Resource Locator*)
  - The distinction is not important for our purposes

Hypertext Transfer Protocol (HTTP)

- **What is a communication protocol?**
  - Set of standards that defines the structure of messages
  - Examples: TCP, IP, HTTP, FTP

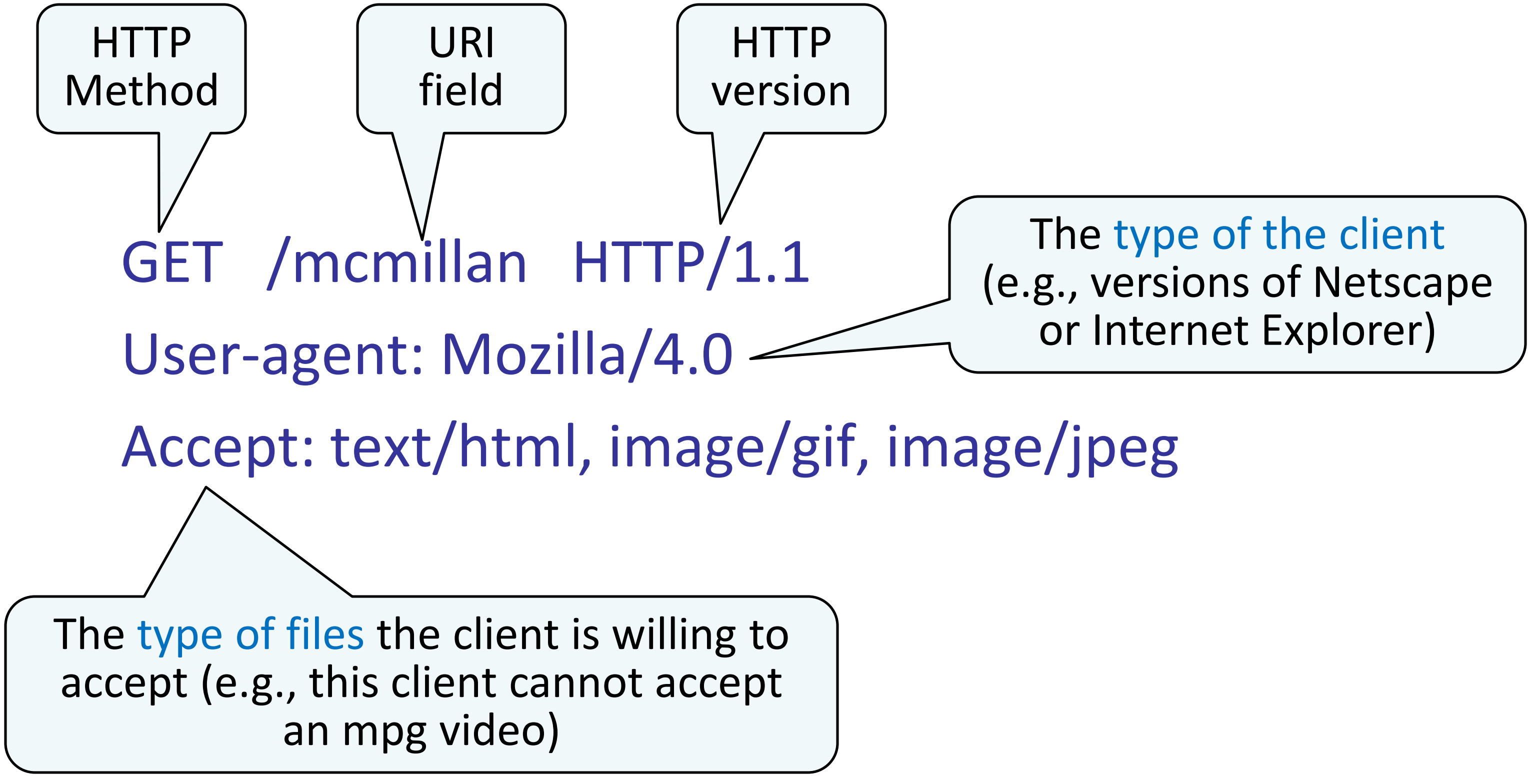
- **What happens if you click on**

  http://csbio.unc.edu/mcmillan ?

1. Client (web browser) sends an *HTTP request* to server (csbio.unc.edu)
2. Server replies with an *HTTP response*

HTTP Requests

HTTP Requests consists of several lines of ASCII text, with an empty line at the end.



HTTP Method

URI field

HTTP version

```
GET   /mcmillan   HTTP/1.1
User-agent: Mozilla/4.0
Accept: text/html, image/gif, image/jpeg
```

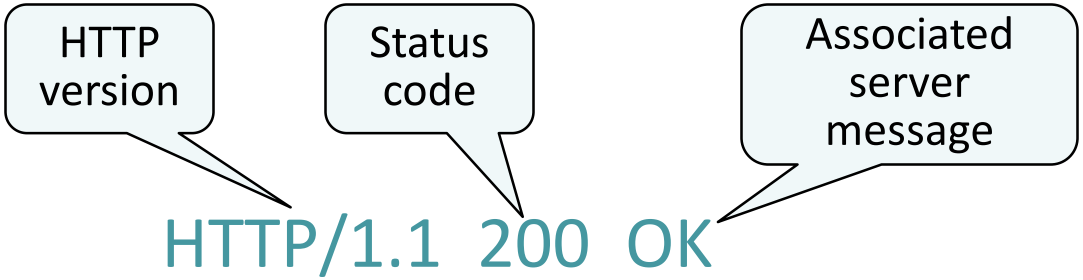
The type of the client (e.g., versions of Netscape or Internet Explorer)

The type of files the client is willing to accept (e.g., this client cannot accept an mpg video)

HTTP Responses

- The server retrieves the page "index.html" and uses it to assemble the HTTP response message
- The HTTP response message has three parts:
  - status line
  - several header lines
  - body of the message
    (which contains the requested object)

HTTP Response: Status Line



HTTP version

Status code

Associated server message

HTTP/1.1 200 OK

Common status codes and associated messages:

- 200 OK: The request succeeded and the object is in the body of the message
- 400 Bad Request: The request could not be fulfilled
- 404 Not Found: The requested object does not exist
- 505 HTTP Version Not Supported: The protocol version used by the client is not supported by the server

HTTP Responses: Header Lines

Server time

```
Date: Mon, 04 Mar 2002 12:00:00 GMT
Server: Apache/1.3.0 (Linux)
Last-Modified: Mon, 01 Mar 2002 09:23:24 GMT
Content-Length: 1024
Content-Type: text/html
```

Server type

Number of bytes being sent

Date when the object was created

Type of object being sent

HTTP Response: Body

```
<HTML> <HEAD></HEAD>
<BODY>
<h1>Barns and Nobble Internet Bookstore</h1>
Our inventory:
<h3>Science</h3>
<b>The Character of Physical Law</b>
…
```

HTTP is Stateless

- HTTP is stateless
  - No “sessions”
  - Every message is self-contained
  - No previous interaction are “remembered” by the protocol
  - Tradeoff between ease of implementation and ease of application development
  - Other functionality has to be built on top
- Implications for applications:
  - Any state information (shopping carts, user login-information) need to be encoded in every HTTP request and response!
  - Popular methods on how to maintain state:
    - Cookies (more on them next lecture)
    - Generate unique URL’s dynamically at the server level

Web Data Formats

- HTML: HyperText Markup Language
  - The presentation language for the Internet
- XML: eXtensible Markup Language
  - A self-describing, hierarchal data model
- DTD: Document Type Declarations
  - Standardizing rules/schemas for XML
- CSS: Cascading Style Sheets
  - Page layout and formatting hints
- XSL: eXtensible Style Language
  - not covered

HTML: Basic Contructs

<html>

An HTML document is enclosed by these two tags

</html>

Commands in HTML consist of a start tag and an end tag

HTML: Basic Constructs

```
<html>
<head>
…
</head>




</html>
```

The head section contains information about the page including the title, author, etc.

HTML: Basic Contructs

```
<html>
<head>
…
</head>
<body>


</body>
</html>
```

The body section contains the parts of the web page the browser will display: text, images, links, etc.

HTML: Basic Contructs

```
<html>
<head>
…
</head>
<body>
<h1>Section 1</h1>



</body>
</html>
```

There are six levels of section headers: h1 through h6

HTML: Basic Contructs

```
<html>
<head>
…
</head>
<body>
<h1>Section 1</h1>
  <ul>
    <li>This is the first item</li>
  </ul>
</body>
</html>
```

This is an unordered list

- Ordered List

```
<ol>
  <li>Coffee</li>
  <li>Tea</li>
</ol>
```

- Definition List

```
<dl>
  <dt>Coffee</dt>
  <dd> …  </dd>
  <dt>Tea</dt>
  <dd> … </dd>
</dl>
```

HTML: Basic Contructs

```
<html>
<head>
…
</head>
<body>
<h1>Section 1</h1>
  <ul>
    <li> This is the <b>first</b> item <\li>
  </ul>
</body>
</html>
```

Display “first” in boldface

HTML: An Example

```
<html>
  <head></head>
  <body>
  <h1>Barns and Nobble Internet Bookstore</h1>

  <h3>Science</h3>

  <b>The Character of Physical Law</b>
    <ul>
      <li>Author: Richard Feynman</li>
      <li>Published 1980</li>
      <li>Hardcover</li>
    </ul>

  <h3>Fiction</h3>

  <b>Oliver Twist</b>
    <ul>
      <li>Author: Charles Dickens</li>
      <li>Published 2002</li>
    </ul>

  <b>Pride and Prejudice</b>
    <ul>
      <li>Author:  Jane Austen</li>
      <li>Published 1983</li>
      <li>Paperback</li>
    </ul>
  </body>
</html>
```

HTML: Summary

- HTML is a markup language for describing content
- Commands are tag enclosures:
  - Start tag and end tag
  - Examples:
    - <HTML> … </HTML>
    - <UL> … </UL>
- Many editors automatically generate HTML directly from your document (e.g., Microsoft Word has an “Save as Web Page” facility)

HTML vs XML

- HTML
  - Supports a fixed set of predefined tags
  - Not enough tags to describe the structures of the content of specific applications (e.g., what part of the content are names?, etc.)
- XML
  - Allows users to define new tags to structure any type of data or document
  - It makes database systems more tightly integrated into Web applications

XML – The Extensible Markup Language

- Language
  - A way of communicating information
- Markup
  - Notes or meta-data that describe your data or language
- Extensible
  - Limitless ability to define new languages or data sets

XML Elements

- Elements are also called tags
- Elements are primary building blocks of an XML document
- Each element of a user-defined type ELM is enclosed by <ELM> and <\ELM>

  Example: <FIRSTNAME>Jessica</FIRSTNAME>
- Elements can be nested (forming a tree structure)

```
Example:  <BOOK>
                  <AUTHOR>
                     <FIRSTNAME>Charles</FIRSTNAME>
                     <LASTNAME>Dickens</LASTNAME>
                 <AUTHOR>
             </BOOK>
```
- EXML elements are case sensitive: BOOK ≠ Book

XML Elements /w Attributes

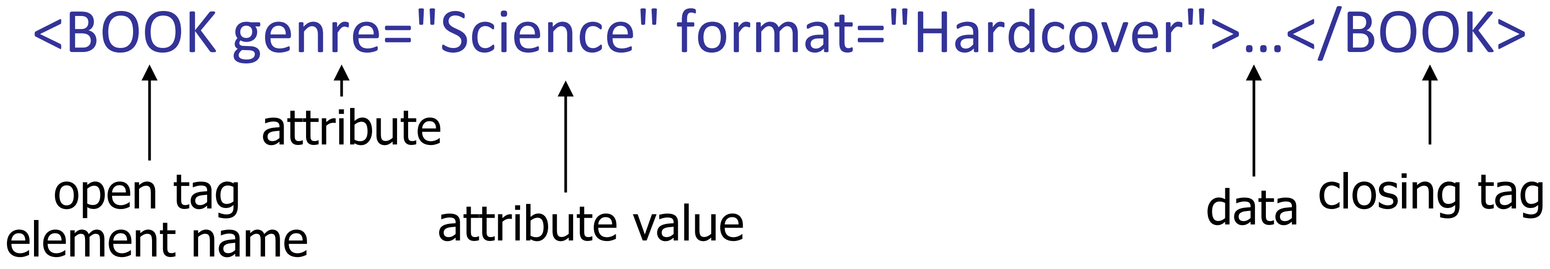
- An Element can have descriptive attributes
- The values of attributes are set inside the start tag of the element
- All attribute values must be enclosed in quotes

Example:

```
<BOOK GENRE=“Fiction” FORMAT=“Hardcover”>
    <AUTHOR>
        <FIRSTNAME>Charles</FIRSTNAME>
        <LASTNAME>Dickens</LASTNAME>
    <AUTHOR>
</BOOK>
```

XML – Structure

- XML looks like HTML
- XML is a hierarchy of user-defined tags called elements with attributes and data
- Data are described by elements, elements are described by attributes



```
<BOOK genre="Science" format="Hardcover">…</BOOK>
```

open tag
element name

attribute

attribute value

data

closing tag

XML Entity References

- XML data can’t contain the reserved characters
- Whenever an *entity reference* appears in the document, it is textually replaced by its content
- Format: &lt; “lt” is an entity reference for the character “<“

Reserved Characters	Entity References
<	lt
>	gt
&	amp
“	quot
‘	apos

&apos;1&lt;5&apos;

↓

‘1 < 5’

XML: Comments

- Comments start with <!-- and end with -->
- Comments can contain arbitrary text except the string --
- Example: <!- comment ->

XML: An Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<BOOKLIST>
    <BOOK genre="Science" format="Hardcover">
        <AUTHOR>
            <FIRSTNAME>Richard</FIRSTNAME><LASTNAME>Feynman</LASTNAME>
        </AUTHOR>
        <TITLE>The Character of Physical Law</TITLE>
        <PUBLISHED>1980</PUBLISHED>
    </BOOK>
    <BOOK genre="Fiction">
        <AUTHOR>
            <FIRSTNAME>Charles</FIRSTNAME><LASTNAME>Dickens</LASTNAME>
        </AUTHOR>
        <TITLE>Oliver Twist</TITLE>
        <PUBLISHED>2002</PUBLISHED>
    </BOOK>
    <BOOK genre="Fiction">
        <AUTHOR>
            <FIRSTNAME>Jane</FIRSTNAME><LASTNAME>Austen</LASTNAME>
        </AUTHOR>
        <TITLE>Pride and Prejudice</TITLE>
        <PUBLISHED>1983</PUBLISHED>
    </BOOK>
</BOOKLIST>
```

Should begin with an XML declaration

A root element contains all other elements

All elements must be properly nested

XML – What's The Point?

- You can include your data and a description of what the data represents
  - This is useful for defining your own language or protocol

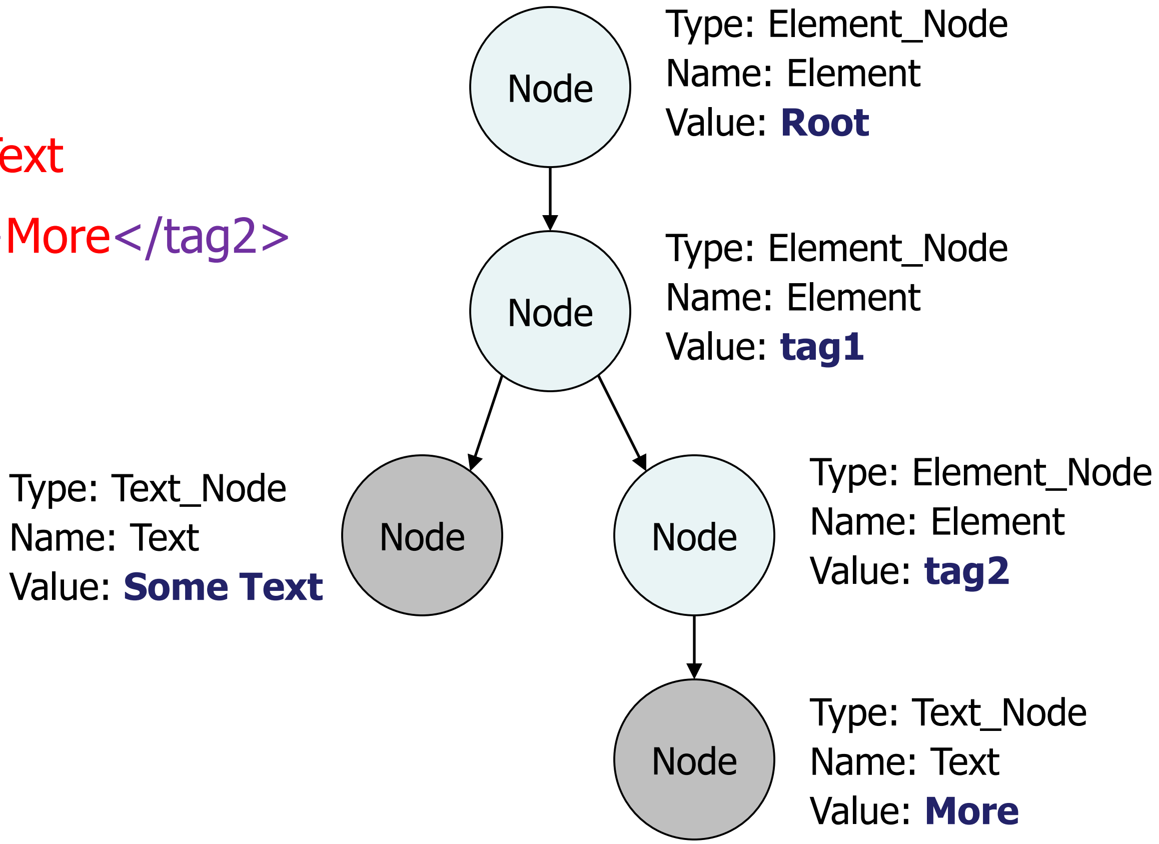
- Example: Chemical Markup Language

```
<molecule>
        <name>Methionine</name>
        <formula>C<sub>5</sub>H<sub>11
        </sub>NO<sub>2</sub>S</formula>
        <weight>149.2</weight>
        <spectra>…</spectra>
        <figures>…</figures>
</molecule>
```

XML – Storage

Storage is just an n-ary tree

```
<root>
   <tag1>
      Some Text
      <tag2>More</tag2>
   </tag1>
</root>
```



Type: Element_Node
Name: Element
Value: **Root**

Type: Element_Node
Name: Element
Value: **tag1**

Type: Text_Node
Name: Text
Value: **Some Text**

Type: Element_Node
Name: Element
Value: **tag2**

Type: Text_Node
Name: Text
Value: **More**

DTD – Document Type Definition

- Unlike HTML, XML has user-defined elements (tags)
  - → the user needs to describe these elements
- DTD is a set of rules that defines the user-defined elements for an XML document
  - → DTD is the schema for the XML data
  - → DTD says what elements and attributes are required or optional (the formal structure of the language)
- A document is valid if it is structured according to the rules set by the DTD

DTD Structure

```
<!DOCTYPE BOOKLIST [
```

A DTD is enclosed in:
<!DOCTYPE name [DTDdeclaration]>

```
]>
```

DTD Structure

```
<!DOCTYPE BOOKLIST [
<!ELEMENT BOOKLIST (BOOK)*>
```

- A DTD starts with the root element
- The root element BOOKLIST consists of zero or more BOOK elements

  * : zero or more occurrences
  + : one or more occurrences
  ? : zero or one occurrence

```
]>
```

DTD Structure

```
<!DOCTYPE BOOKLIST [
<!ELEMENT BOOKLIST (BOOK)*>
  <!ELEMENT BOOK (AUTHOR,TITLE,PUBLISHED?)>
```

- An element can have nested elements
- This rule says that a BOOK element contains an AUTHOR element, a TITLE element, and an optional PUBLISHED element

```
]>
```

DTD Structure

```
<!DOCTYPE BOOKLIST [
<!ELEMENT BOOKLIST (BOOK)*>
    <!ELEMENT BOOK (AUTHOR,TITLE,PUBLISHED?)>
        <!ELEMENT AUTHOR (FIRSTNAME,LASTNAME)>
              <!ELEMENT FIRSTNAME (#PCDATA)>
              <!ELEMENT LASTNAME (#PCDATA)>
]>
```

- Instead of containing other elements, an element can contain actual text
  - #PCDATA indicates character data
  - EMPTY indicates the element has no content
  - ANY indicates that any content is permitted. No checking inside this structure (avoided whenever possible)

DTD Structure

```
<!DOCTYPE BOOKLIST [
<!ELEMENT BOOKLIST (BOOK)*>
    <!ELEMENT BOOK (AUTHOR,TITLE,PUBLISHED?)>
            <!ELEMENT AUTHOR (FIRSTNAME,LASTNAME)>
                        <!ELEMENT FIRSTNAME (#PCDATA)>
                        <!ELEMENT LASTNAME (#PCDATA)>
            <!ELEMENT TITLE (#PCDATA)>
            <!ELEMENT PUBLISHED (#PCDATA)>
    <!ATTLIST BOOK GENRE (Science|Fiction) #REQUIRED>
    <!ATTLIST BOOK FORMAT (Paperback|Hardcover) “Paperback”>
]>
```

DTD Structure

```
<!DOCTYPE BOOKLIS
<!ELEMENT BOOKLIS
    <!ELEMENT BOO
        <!ELEMENT
              <!EL
              <!EL
        <!ELEMENT TI
        <!ELEMENT PUBLISHED (#PCDATA)>
    <!ATTLIST BOOK GENRE (Science|Fiction) #REQUIRED>
    <!ATTLIST BOOK FORMAT (Paperback|Hardcover) “Paperback”>
]>
```

- Attributes of elements are declared outside the element
- The BOOK element has two attributes
  - The GENRE attribute is required and can have the value ‘Science’ or ‘Fiction’
  - The FORMAT attribute can have the value ‘Paperback’ or ‘Hardcover’, and ‘Paperback’ is the default value
  - #REQUIRED is the default option

The whole DTD

```
<!DOCTYPE BOOKLIST [
<!ELEMENT BOOKLIST (BOOK)*>
  <!ELEMENT BOOK (AUTHOR,TITLE,PUBLISHED?)>
      <!ELEMENT AUTHOR (FIRSTNAME,LASTNAME)>
            <!ELEMENT FIRSTNAME (#PCDATA)>
            <!ELEMENT LASTNAME (#PCDATA)>
      <!ELEMENT TITLE (#PCDATA)>
      <!ELEMENT PUBLISHED (#PCDATA)>
  <!ATTLIST BOOK GENRE (Science|Fiction) #REQUIRED>
  <!ATTLIST BOOK FORMAT (Paperback|Hardcover) “Paperback”>
]>
```

Five Possible Content Types

<!ELEMENT (contentType)>

- Other elements
- Special symbol #PCDATA, EMPTY, or ANY
- A regular expression constructed from the preceding four choices
  - exp1, exp2, exp3: An ordered list of regular expressions
  - Exp*: An optional expression (zero or more occurrence)
  - Exp?: An optional expression (zero or one occurrences)
  - Exp+: A mandatory expression (one or more occurrences)
  - Exp1 | exp2: exp1 or exp2

DTD – An Example

```
<?xml version='1.0'?>
<!ELEMENT Basket (Cherry+, (Apple | Orange)*) >
    <!ELEMENT Cherry EMPTY>
        <!ATTLIST Cherry flavor CDATA #REQUIRED>
    <!ELEMENT Apple EMPTY>
        <!ATTLIST Apple color CDATA #REQUIRED>
    <!ELEMENT Orange EMPTY>
        <!ATTLIST Orange location ‘Florida’>
```

```
<Basket>
    <Cherry flavor=‘good’/>
    <Apple color=‘red’/>
    <Apple color=‘green’/>
</Basket>
```

Apple’s color is required. Cherry should go first.

```
<Basket>
    <Apple/>
    <Cherry flavor=‘good’/>
    <Orange/>
</Basket>
```

DTD – Well-Formed and Valid

```
<?xml version='1.0'?>
<!ELEMENT Basket (Cherry+)>
    <!ELEMENT Cherry EMPTY>
          <!ATTLIST Cherry flavor CDATA #REQUIRED>
```

Not Well-Formed

```
<basket>
   <Cherry flavor=good>
</Basket>
```

Well-Formed but Invalid

```
<Job>
   <Location>Home</Location>
</Job>
```

Well-Formed and Valid

```
<Basket>
   <Cherry flavor=‘good’>
</Basket>
```

XML and DTDs

- More and more standardized (domain-specific) DTDs will be developed
  - MathML (Mathematical Markup Language)
  - Chemical Markup Language
- Enable seamless data exchange among heterogeneous sources
- Sophisticated query languages for XML are available:
  - Xquery
  - XPath