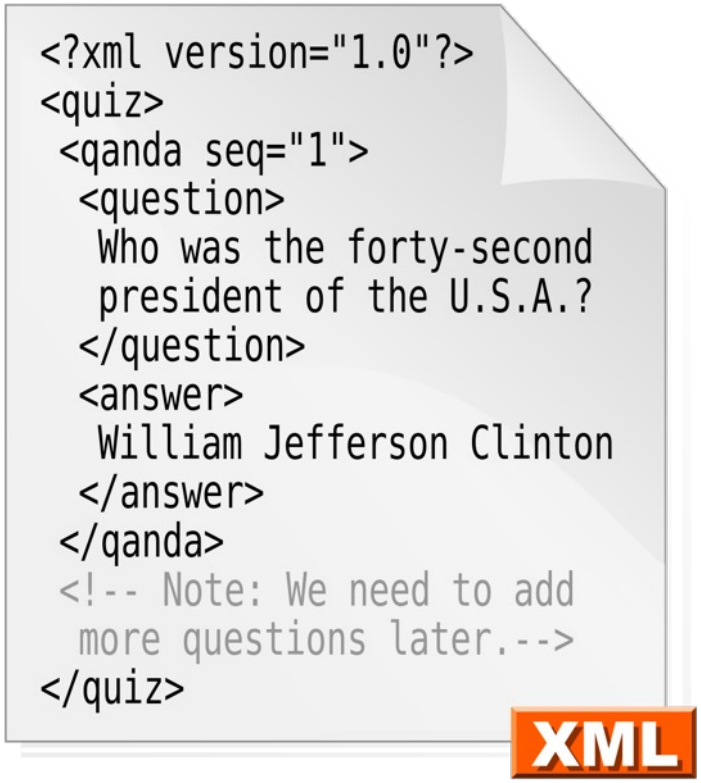


Structured Web Documents in XML

Adapted from slides from Grigoris Antoniou and Frank van Harmelen

XML Outline

(1) Introduction

(2) XML details

(3) XML Schema

(4) Namespaces

(5) Accessing, querying XML documents: XPath

(6) Transformations: XSLT

(7) Summary and comments

Role of XML in the Semantic Web

- The Semantic Web involves ideas and languages at a fairly abstract level, e.g.: for defining ontologies, publishing data using them
- XML is important for many reasons:
  - Source of many key RDF concepts & technology
  - Potential alternative for sharing data that newer schemes (like RDF) must improve on; and
  - A common serialization for SW data
  - How much useful online data is encoded, e.g. Wikidata represents equations using MathML

Standard disclaimer
(paraphrasing Jamie Zawinski)

Some people, when confronted with a problem, think, "I know, I'll use XML."

Now they have two problems.

“Some people, when confronted with a problem, think "I know, I'll use regular expressions." Now they have two problems.” -- Wikiquote

History

- XML's roots are in SGML
  - Standard Generalized Markup Language
  - A *metalanguage* for defining document markup languages
  - Extensible, but complicated, verbose, hard to parse, …
- HTML was defines using SGML, ~1990 by TBL
  - A markup language, not a markup *metalanguage*
- XML proposal to W3C in July 1996
  - Simplified SGML, expanding Web's power and flexibility
- Evolving series of W3C recommendations
  - Current recommendation: XML 5 (2008)

An HTML Example

```
<h2>Nonmonotonic Reasoning: Context-
   Dependent Reasoning</h2>
<i>by <b>V. Marek</b> and
   <b>M. Truszczynski</b></i><br>
Springer 1993<br>
ISBN 0387976892
```

The Same Example in XML

```
<book>
   <title>Nonmonotonic Reasoning: Context-Dependent
   Reasoning</title>
   <author>V. Marek</author>
   <author>M. Truszczynski</author>
   <publisher>Springer</publisher>
   <year>1993</year>
   <ISBN>0387976892</ISBN>
</book>
```

HTML versus XML: Similarities

- Both use **tags** (e.g. <h2> and </year>)
- Tags may be nested (tags within tags)
- Human users can read and interpret both HTML and XML representations “easily”

… **But how about machines?**

HTML vs XML: Structured information

- HTML documents generally require human-level knowledge and experience to understand

  Don't carry **structured information**: pieces document and their relations

- XML more easily accessible to machines since
  - Every piece of information is described
  - Relations defined through nesting structure, e.g., **<author>** tags appear within **<book>** tags and describe properties of that book
  - XML allows constraints on values, e.g., a year must be a four-digit integer

HTML vs. XML: Formatting

- **HTML:** goal is to **display** information in a human-readable way: it must define formatting
- **XML:** goal is to **separate** content from display
  - same information can be displayed in different ways
  - Presentation specified by documents using other XML standards (CSS, XSL)

HTML vs. XML: Different Use of Tags

- All HTML documents use the **same tags**
  - HTML tags come from a finite, pre-defined collection
  - Define properties for display: font, color, lists …
- XML documents can use different sets of tags
  - XML tags not fixed: **user definable tags**
  - XML is a *meta markup language*, i.e., a language for defining markup languages
- Used to define domain-specific markup languages like MusicXML, MathML, RSS, SVG, and XHTM

2: The XML Language

An XML document consists of

- A **prolog**
- A number of **elements**
- An optional **epilog** (not discussed, not used much)

XML documents are tree data structures

Prolog of an XML Document

The prolog consists of

- An XML declaration and
- An optional reference to external structuring documents

<?xml version="1.0" encoding="UTF-16” ?>

<!DOCTYPE book SYSTEM "book.dtd">

XML Elements

- Elements are the *things* the XML document talks about
  - E.g., books, authors, publishers, …
- An element consists of:
  - An opening tag
  - The content
  - A closing tag

<lecturer> David Billington </lecturer>

XML Elements

- Tag names can be chosen almost freely
- First character must be a letter, underscore, or colon
- No name may begin with the string “xml” in any combination of cases
  - E.g. “Xml”, “xML”

Content of XML Elements

- Content is what's between the tags
- It can be text, or other elements, or nothing

```
<lecturer>
  <name>David Billington</name>
  <phone> +61 – 7 – 3875 507 </phone>
</lecturer>
```

- If there is no content, then element is called empty; it can be abbreviated as follows:

<lecturer/> = <lecturer></lecturer>

XML Attributes

- An empty element isn't necessarily meaningless
  - It may have properties expressed as ***attributes***
- An **attribute** is a name-value pair inside the opening tag of an element

```
<lecturer
  name="David Billington"
  phone="+61 – 7 – 3875 507" />
```

XML Attributes: An Example

```
<order  orderNo="23456“
        customer="John Smith"
        date="October 15, 2017" >
  <item itemNo="a528" quantity="1" />
  <item itemNo="c817" quantity="3" />
</order>
```

The Same Example w/o Attributes

```
<order>
   <orderNo>23456</orderNo>
   <customer>John Smith</customer>
   <date>October 15, 2017</date>
   <item>
         <itemNo>a528</itemNo>
         <quantity>1</quantity>
   </item>
   <item>
         <itemNo>c817</itemNo>
         <quantity>3</quantity>
         </item>
</order>
```

XML Elements vs. Attributes

- Attributes can be replaced by elements
- When to use elements and when attributes is a mostly matter of taste
- **But attributes cannot be nested**

Well-Formed XML Documents

Constraints on syntactically correct documents:

- Only one outermost element (**root element**)
- Each element contains opening and corresponding closing tag (except self-closing tags like <foo/>)
- Tags may not overlap

  <author><name>Lee Hong</author></name>
- Attributes within an element have unique names
- Element and tag names must follow rules, e.g.: can’t use strings beginning with digit ("2ndbest”)

The Tree Model of XML Docs

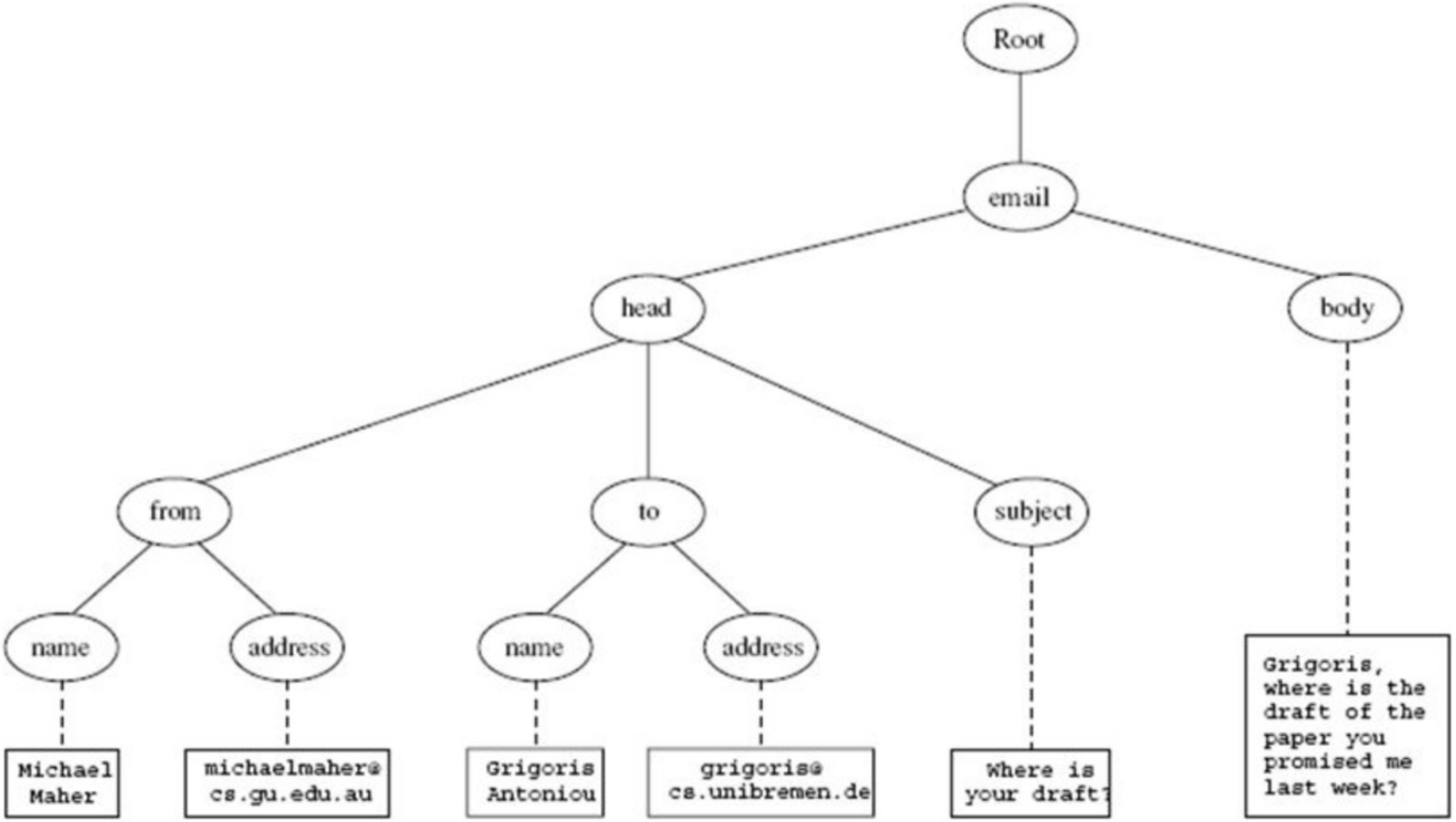
The tree representation of an XML document is an **ordered**, labeled tree:

- Exactly one root
- No cycles
- Each non-root node has exactly one parent
- Each node has a label
- Order of elements is important, but order of attributes is not

Tree Model of XML Documents

```
<email>
   <head>
      <from name="Michael Maher"
            address="michaelmaher@cs.gu.edu.au" />
      <to name="Grigoris Antoniou"
           address="grigoris@cs.unibremen.de" />
      <subject>Where is your draft?</subject>
   </head>
   <body>
      Grigoris, where is the draft of the paper you
       promised me last week?
   </body>
</email>
```

Tree Model of XML Documents



3: Structuring XML Documents

- Some XML documents must follow constraints defined in a “template” that can…
  - define the *element* and *attribute names* that may be used
  - define the *structure*
    - what values an attribute may take
    - which elements may or must occur within other elements, etc.
- If such structuring information exists, the document can be **validated**

Validating XML Documents

- An XML document is **valid** if
  - it is well-formed XML
  - respects the structuring information it uses
- Ways to define structure of XML documents:
  - **DTDs** (*Document Type Definition*) came first, was based on SGML's approach
  - **XML Schema** (aka *XML Schema Definition*, XSD) is more recent and expressive
  - RELAX NG and DSDs are two alternatives

XML Schema (XSD)

- XML Schema is a richer language for defining the structure of XML documents
- Syntax based on XML itself, so separate tools to handle them not needed
- Reuse and refinement of schemas => can expand or delete existing schemas
- Sophisticated set of **data types**, compared to DTDs, which only supports strings
- XML Schema recommendation published by W3C in 2001, version 1.1 in 2012

XML Schema

- An XML schema is an element with an opening tag like

  <schema

  "http://www.w3.org/2000/10/XMLSchema"

  version="1.0">

- Structure of schema elements
  - Element and attribute types using data types

Element Types & cardinality constraints

```
<element name="email"/>
<element name="head“
    minOccurs="1“
    maxOccurs="1"/>
<element name="to" minOccurs="1"/>
```

Cardinality constraints:

- **minOccurs="x"** (default value 1)
- **maxOccurs="x"** (default value 1)

Attribute Types

```
<attribute name="id" type="ID“ use="required"/>
<attribute name="speaks" type="Language"
    use="default" value="en"/>
```

- Existence: **use="x",** where **x** may be **optional** or **required**
- Default value: **use="x" value="..."**, where **x** may be **default** or **fixed**

Data Types

- Many **built-in data types**
  - Numerical data types: **integer**, **short,** etc.
  - String types: **string**, **ID**, **IDREF**, **CDATA,** etc.
  - Date and time data types: **time**, **month,** etc.
- Also **user-defined data types**
  - **simple data types**, which can't use elements or attributes
  - **complex data types**, which can use them

Complex Data Types

Complex data types are defined from existing data types by defining some attributes (if any) and using:

- **sequence**, a sequence of existing data type elements (order is important)
- **all**, a collection of elements that must appear (order is not important)
- **choice**, a collection of elements, of which one will be chosen

XML Schema: The Email Example

```
<element name="email" type="emailType"/>

<complexType name="emailType">
    <sequence>
        <element name="head" type="headType"/>
        <element name="body" type="bodyType"/>
    </sequence>
</complexType>
```

XML Schema: The Email Example

```
<complexType name="headType">
    <sequence>
        <element name="from" type="nameAddress"/>
        <element name="to" type="nameAddress"
                minOccurs="1" maxOccurs="unbounded"/>
        <element name="cc" type="nameAddress"
                minOccurs="0" maxOccurs="unbounded"/>
        <element name="subject" type="string"/>
    </sequence>
</complexType>
```

XML Schema: The Email Example

```
<complexType name="nameAddress">
        <attribute name="name" type="string"
          use="optional"/>
        <attribute name="address"
         type="string" use="required"/>
</complexType>
```

- Similar for bodyType

4: What's a Namespace?

- Wikipedia: "In computing, a **namespace** is a set of symbols that are used to organize objects of various kinds, so that these objects may be referred to by name"
- Let us combine/integrate multiple vocabularies and avoid name collisions
- Format: **prefix:name**, where prefix is a URI with which is an XML schema file
- **Namespace concept used important in RDF**

An Example

```
<vu:instructors
   xmlns:vu="http://www.vu.com/empDTD"
   xmlns:gu="http://www.gu.au/empDTD"
   xmlns:uky=http://www.uky.edu/empDTD
<uky:faculty uky:title="assistant professor"
   uky:name="John Smith"
   uky:department="Computer Science"/>
<gu:academicStaff  gu:title="lecturer"
   gu:name="Mate Jones"
   gu:school="Information Technology"/>
</vu:instructors>
```

Namespace Declarations

- Namespaces declared within elements for use in it and its children (elements and attributes)
- Namespace declaration has form:
  - **xmlns:prefix="location"**
  - **location** is a URI, often the DTD or schema file
- If no prefix specified: **xmlns="location"** then the **location** is used as the *default* prefix
- This same idea used in RDF

Addressing & Querying XML Documents

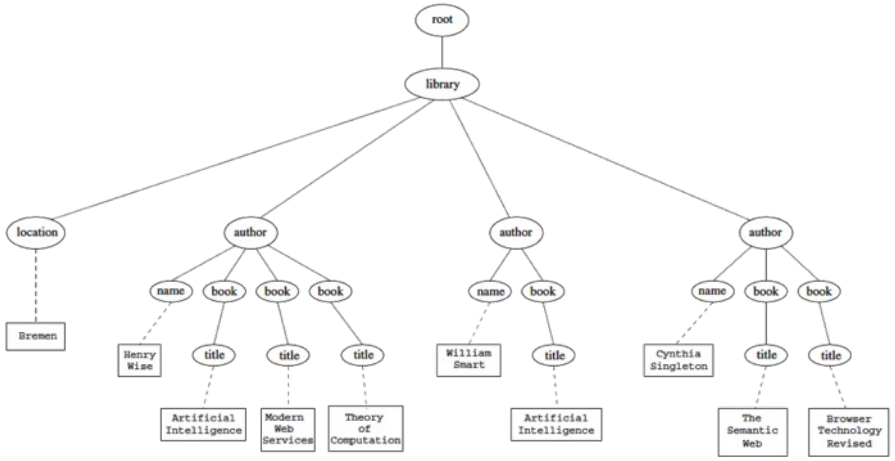
- In relational databases, parts of a database can be selected and retrieved using SQL
  - Also very useful for XML documents
  - **Query languages**: XQuery, XQL, XML-QL
- The central concept of XML query languages is a **path expression**
  - Specifies how a node or set of nodes, in the tree representation, can be reached
- Useful for extracting data from XML

5: Accessing data with Xpath queries

- XPath is core for XML query languages
- Select nodes in XML documents from the tree data model of XML using **path expressions**
  - **Absolute** (starting at the root of the tree)
    - Syntactically they begin with the symbol **/**
    - Refers to the root of the document (one level above document's root element)
  - **Relative** to a context node

An XML Example and its tree

```
<library location="Bremen">
  <author name="Henry Wise">
    <book title="Artificial Intelligence"/>
    <book title="Modern Web Services"/>
    <book title="Theory of Computation"/>
  </author>
  <author name="William Smart">
    <book title="Artificial Intelligence"/>
  </author>
  <author name="Cynthia Singleton">
    <book title="The Semantic Web"/>
    <book title="Browser Technology Revised"/>
  </author>
</library>
```



Examples of Path Expressions in XPath

- **Q1: /library/author**
  - Addresses **all** **author** elements that are children of the **library** element node immediately below root
  - **/t1/.../tn**, where each **ti+1** is a child node of **ti**, is a path through the tree representation
- **Q2: //author**
  - Consider all elements in document and check whether they are of type **author**
  - Path expression addresses all **author** elements **anywhere** in the document

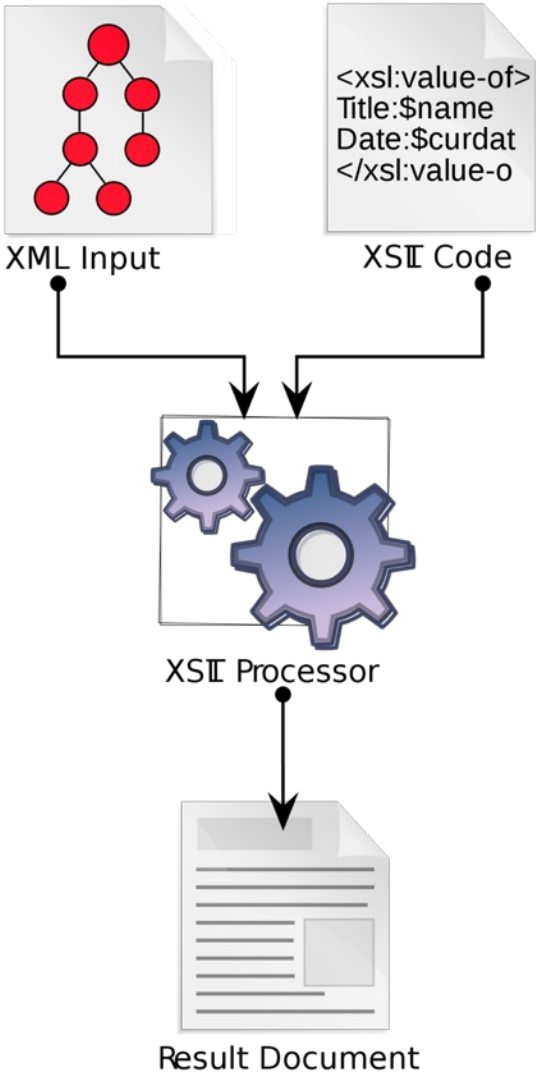
Examples of Path Expressions in XPath

- **Q3: /library/@location**
  - Addresses location attribute nodes within library element nodes
  - The symbol **@** is used to denote attribute nodes

- **Q4: //book/@title="Artificial Intelligence”**
  - Addresses all title attribute nodes within book elements anywhere in the document that have the value “Artificial Intelligence”

6: XSL Transformations (XSLT)

- **Idea:** use an external *style sheet* to transform an XML tree into an HTML or XML tree
- **XSLT** specifies rules to transform an XML document to another XML one, HTML, or plain text
- Output may use same schema, or completely different vocabulary
- Browsers do this automatically for an XML file with a linked XSLT file



Example: XML + XSLT => HTML

1
```
<author>
  <name>Grigoris Antoniou</name>
  <affiliation>University of Bremen</affiliation>
  <email>ga@tzi.de</email>
</author>
```

2
```
<xsl:template match="/author"> <html>
    <head><title>An author</title></head>
    <body bgcolor="white">
      <b><xsl:value-of select="name"/></b><br/>
      <xsl:value-of select="affiliation"/><br/>
      <i><xsl:value-of select="email"/></i>
    </body>
 </html></xsl:template>
```

3
```
<html>
   <head><title>An author</title></head>
   <body bgcolor="white">
         <b>Grigoris Antoniou</b><br/>
         University of Bremen<br/>
         <i>ga@tzi.de</i>
   </body>
</html>
```

CD Catalog example

```
<?xml-stylesheet type="text/xsl"
href="cdcatalog.xsl"?>
<catalog>
<cd>
  <title>Empire Burlesque</title>
  <artist>Bob Dylan</artist>
  <country>USA</country>
  <company>Columbia</company>
  <price>10.90</price>
  <year>1985</year>
</cd>
<cd>
  <title>Hide your heart</title>
  <artist>Bonnie Tyler</artist>
  <country>UK</country>
  <company>CBS Records</company>
…
</cd> …
```

```
<xsl:stylesheet …>
<xsl:template match="/">
 <html> <body>
 <h2>My CD Collection</h2>
  <table border="1">
   <tr bgcolor="#9acd32">
    <th align="left">Title</th>
    <th align="left">Artist</th>
   </tr>
   <xsl:for-each select="catalog/cd">
    <tr>
     <td><xsl:value-of select="title"/></td>
     <td><xsl:value-of select="artist"/></td>
    </tr>
   </xsl:for-each>
  </table>
 </body> </html>
</xsl:template>
</xsl:stylesheet>
```

See these files online

Viewing an XML file in a Browser

curl –L http://bit.ly/CdCat19

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
<catalog>
<cd>
 <title>Empire Burlesque</title>
 <artist>Bob Dylan</artist>
 <country>USA</country>
 <company>Columbia</company>
 <price>10.90</price>
 <year>1985</year>
</cd>
<cd>
 <title>Hide your heart</title>
 <artist>Bonnie Tyler</artist>
 <country>UK</country>
 <company>CBS Records</company>
 <price>9.90</price>
 <year>1988</year>
</cd>
...
```

The XML data with a linked style sheet

Browser uses style sheet to get HTML

7: XML Summary

- XML is a metalanguage that allows users to define markup
- XML separates content and structure from formatting
- XML is (one of the) the de facto standard to represent and exchange structured information on the Web
- XML is supported by query languages

Comments for Discussion

- Nesting of tags has *no standard meaning*
- *Semantics* of XML documents not accessible to machines and may or may not be for people
- Collaboration & exchange supported if there is underlying shared understanding of vocabulary
- XML well-suited for **close collaboration** where domain or community-based vocabularies are used; less so for global communication
- Databases went from tree structures (60s) to relations (80s) and graphs (10s)