Introduction to Semantic Web
Lecture 2: XML

Dr. Knarig Arabshian
Knarig.arabshian@hofstra.edu

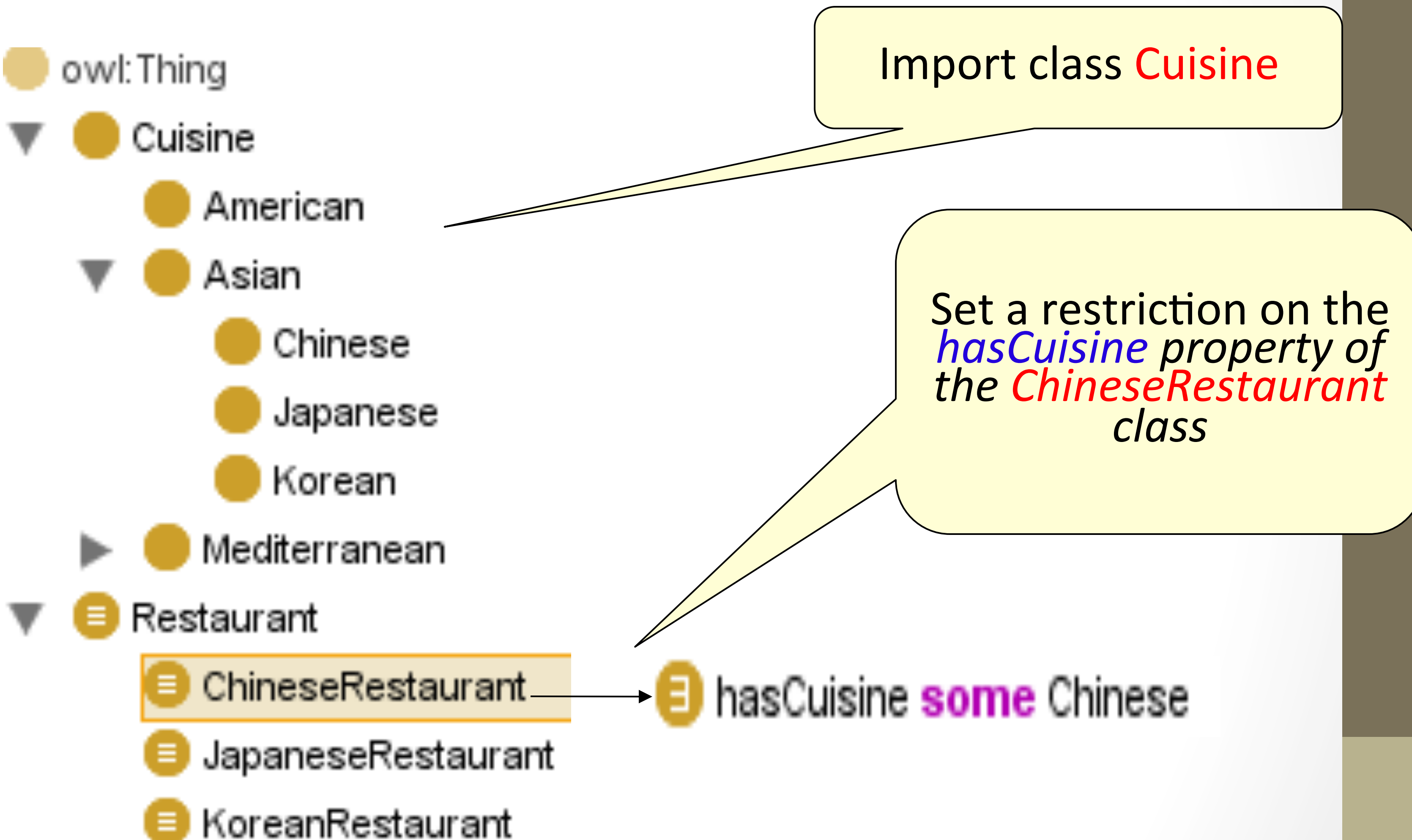
Review from Last class

Ontology

- Knowledge Representation using Ontologies
- Definition:
  - Wikipedia: is a formal representation of knowledge as a set of concepts within a domain, and the relationships between those concepts. It is used to reason about the entities within that domain, and may be used to describe the domain.
  - Webopedia: computer-based resources that represent agreed domain semantics...relatively generic knowledge that can be reused by different kinds of applications or tasks.
  - Gruber: A formal, explicit specification of a shared conceptualization

Ontology

Example Restaurant Ontology



Ontology

Classification

Since Chinese cuisine has non-disjoint siblings Japanese and Korean then also conclude that these are *similar* to Chinese

Conclude that QueryRestaurant is equivalent to ChineseRestaurant

Ontologies and Relational Databases

- Similarities
  - Both use a model to identify common classes and properties
  - ER model can be seen as a simple hierarchical ontology
- Differences
  - Ontologies are broader in scope (rules, incomplete knowledge)
  - Ontologies provide a way for automated reasoning to occur in order to discover new relationships between entities

Ramez Elmasri, Shamkant B. Navathe: Fundamentals of Database Systems, 3rd Edition. Benjamin/Cummings 2000

Discussion: How to build a model?

- Categorizing books
- Think about the different ways they can be classified
- Create a hierarchical classification for it
- Put as much information as needed

Quest for Semantics

Three main goals of the Semantic Web:

- *Building models*: quest for describing the world in abstract terms to allow for an easier understanding of complex reality
- *Computing with knowledge*: constructing reasoning machines that can draw meaningful conclusions from encoded knowledge
- *Exchanging Information*: the transmission of complex information resources among computers that allows us to distribute, interlink, and reconcile knowledge on a global scale

Hitzler, et al., Foundations of Semantic Web Technolgoies

Building Models

- Model: simplified description of certain aspects of reality, use for understanding, structuring, or predicting parts of the real world
- History of scientific modeling
  - Plato (429-347BC)
    - What is reality?
    - Which things can be said to exist?
    - What is the true nature of things?
    - First major contribution to philosophical field of *ontology*
- Ontology in computer science
  - Description of knowledge about a domain of interest, the core of which is a machine-processable specification with a formally defined meaning

Hitzler, et al., Foundations of Semantic Web Technolgoies

Building Models

- Taxonomy: hierarchical classification
  - Linnaean taxonomy: classifies all life forms
  - WHO's International Classification of Diseases
  - Dewey Decimal Classification: ordering books in a library
- Non-hierarchical classifications
  - Periodic table of chemical elements
  - Thesaurus

Calculating with Knowledge

Syllogism:

All A are B.

All B are C.

All A are C.

Domain-independent rules provide template-like ways for inferring knowledge

Calculating with Knowledge

- Goal of AI: build machines exhibiting human intelligence
- Amount of knowledge for basic AI applications is overwhelming. Transforming human knowledge to machine-processable form is difficult
- Inference techniques became too slow for medium or large-scale tasks
- Consequently: research focused on restricted domains
  - Expert systems, rule-based systems for highly structured areas

Exchanging Information

- Internet
  - Packet-switching developed by Baran, Davies and Kleinrock
  - Splitting transmission into small “packets” and transmitted individually
  - ARPANET first packet-switching network in 1969
- Applications
  - E-mail, Usenet
  - HTML, HTTP
  - Wikis, blogs, social networks, tagging

Syntax vs Semantics

- Communication
  - Different modes of communication (speech, writing, smoke signals)
- Sharing data can be broken down into two problems
  - Syntactic sharing problem
    - Finding a common medium for communication
  - Semantic sharing problem
    - Finding a mutual encoding of concepts within a common medium

Hitzler, et al., Foundations of Semantic Web Technolgoies

XML

XML

- A markup language that defines a set of rules for encoding documents in a format which is both human-readable and machine-readable.
- Defined by the W3C
- The design goals of XML
  - Emphasize simplicity, generality and usability across the Internet.
  - Focuses on documents, but is widely used for the representation of arbitrary data structures such as those used in web services.
  - You invent your own tags when to describe the data
- Complementary to HTML
  - HTML is for displaying data
  - XML is for describing data

http://en.wikipedia.org/wiki/XML

XML- eXtensible Markup Language

Basic idea: adding additional information or structure to (unstructured) text

- to *annotate* text means to add a note by way of comment or explanation
- usually done by way of *tags*:

<tag-name> ... Text ... </tag-name>

[opening tag]   [closing tag]

Markup Languages

In HTML

<h2>Relationship force-mass</h2>

<i> F = M × a </i> → display formula

In XML

<equation>

<meaning>Relationship force-mass</meaning>

<leftside> F </leftside>

<rightside> M × a </rightside> → describe formula

</equation>

HTML vs XML

- HTML tags are fixed and define how content is displayed (color, lists …)
- XML tags not fixed but are defined by users to describe *content*

HTML vs XML

- Most prominent example: HTML Annotations used for encoding display information

<i>This book</i> has the title FOST.

Browser shows: *This book* has the title **FOST**.

- Same idea can be used for content description:

<book>This book</book> has the title

<title>FOST</title>.

The XML Language

An XML document consists of

- a prolog
- a number of elements

Prolog of an XML Document

The prolog consists of an XML declaration

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

XML Elements

- 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
- The first character must be a letter, an underscore, or a colon
- No name may begin with the string "xml" in any combination of cases
    - E.g. "Xml", "xML"

Content of XML Elements

- Content may 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 the element is called empty; it is abbreviated as follows:

<lecturer/> or **<lecturer></lecturer>**

XML Attributes

- An empty element is not necessarily meaningless
  - It may have some properties in terms of 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, 2002">
    <item itemNo="a528" quantity="1"/>
    <item itemNo="c817" quantity="3"/>
</order>
```

The Same Example without Attributes

```
<order>
    <orderNo>23456</orderNo>
    <customer>John Smith</customer>
    <date>October 15, 2002</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 matter of taste
- But attributes **cannot** be nested

Well-Formed XML Documents

- Syntactically correct documents
- Some syntactic rules:
  - Only one outermost element (called root element)
  - Each element contains an opening and a corresponding closing tag
  - Tags may not overlap
    - **<author><name>Lee Hong</author></name>**
  - Attributes within an element have unique names
  - Element and tag names must be permissible

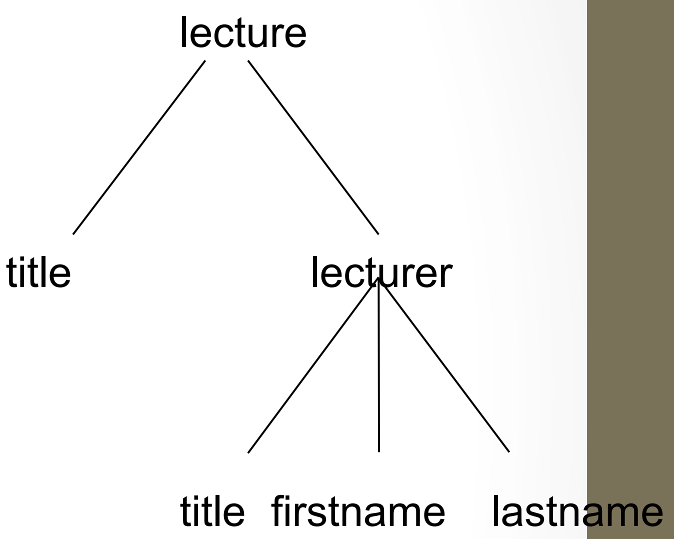
XML Tags

Can be nested

```
<lecture>
    <title>    CSC143   </title>
    <lecturer>
        <title> Dr.   </title>
        <firstname>Knarig     </firstName>
        <lastname> Arabshian    </lastName>
     </lecturer>
</lecture>
```

Tree Structure

```
<lecture>
  <title> Intro To SW   </title>
  <lecturer>
    <title> Dr. </title>
    <firstname>Knarig
     </firstName>
    <lastname> Arabshian </lastName>
  </lecturer>
</lecture>
```



Antoniu, et al., A Semantic Web Primer

The Tree Model of XML Docs

- The tree representation of an XML document is an ordered labeled tree:
  - There is exactly one root
  - There are no cycles
  - Each non-root node has exactly one parent
  - Each node has a label.
  - The order of elements is important
  - … but the order of attributes is not important

XML Schema

- Significantly richer language for defining the structure of XML documents
- Syntax is based on XML itself
  - not necessary to write separate tools
- Reuse and refinement of schemas
  - Expand or delete already existent schemas
- Sophisticated set of data types

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

<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

- There is a variety of 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.
- There are also user-defined data types
  - simple data types, which cannot use elements or attributes
  - complex data types, which can use these

Data Types

- Complex data types are defined from already 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

A Data Type Example

```
<complexType name="lecturerType">
    <sequence>
        <element name="firstname" type="string"
            minOccurs="0“   maxOccurs="unbounded"/>
        <element name="lastname" type="string"/>
    </sequence>
    <attribute name="title" type="string"  use="optional"/
    >
</complexType>
```

Meaning: an element in an XML document that is declared to be of type *lecturerType* may have a *title* attribute; it may also include any number of *firstname* elements and must include exactly one *lastname* element

Data Type Extension

Already existing data types can be extended by new elements or attributes. Example:

```
<complexType name="extendedLecturerType">
    <extension base="lecturerType">
        <sequence>
            <element name="email" type="string"
                minOccurs="0" maxOccurs="1"/>
        </sequence>
        <attribute name="rank" type="string" use="required"/>
    </extension>
</complexType>
```

Resulting Data Type

```
<complexType name="extendedLecturerType">
    <sequence>
        <element name="firstname" type="string"
            minOccurs="0" maxOccurs="unbounded"/>
        <element name="lastname" type="string"/>
        <element name="email" type="string"
            minOccurs="0" maxOccurs="1"/>
    </sequence>
    <attribute name="title" type="string" use="optional"/>
    <attribute name="rank" type="string" use="required"/>
</complexType>
```

Data Type Extension

- A hierarchical relationship exists between the original and the extended type
  - Instances of the extended type are also instances of the original type
  - They may contain additional information, but neither less information, nor information of the wrong type

Data Type Restriction

- An existing data type may be restricted by adding constraints on certain values
- Restriction is not the opposite from extension
  - Restriction is not achieved by deleting elements or attributes
- The following hierarchical relationship still holds:
  - Instances of the restricted type are also instances of the original type
  - They satisfy at least the constraints of the original type

Example of Data Type Restriction

```
<complexType name="restrictedLecturerType">
    <restriction base="lecturerType">
        <sequence>
                <element name="firstname" type="string"
                    minOccurs="1" maxOccurs="2"/>
            </sequence>
            <attribute name="title" type="string"
                use="required"/>
        </restriction>
</complexType>
```

Restriction of Simple Data Types

```
<simpleType name="dayOfMonth">
    <restriction base="integer">
        <minInclusive value="1"/>
        <maxInclusive value="31"/>
    </restriction>
</simpleType>
```

Data Type Restriction: Enumeration

```
<simpleType name="dayOfWeek">
	<restriction base="string">
		<enumeration value="Mon"/>
		<enumeration value="Tue"/>
		<enumeration value="Wed"/>
		<enumeration value="Thu"/>
		<enumeration value="Fri"/>
		<enumeration value="Sat"/>
		<enumeration value="Sun"/>
	</restriction>
</simpleType>
```

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**

Namespaces

- An XML document may use more than one schema
- Since each structuring document was developed independently, name clashes may appear
- The solution is to use a different prefix for each schema
  - **prefix:name**

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 are declared within an element and can be used in that element and any of its children (elements and attributes)
- A namespace declaration has the form:
  - **xmlns:prefix="location"**
  - **location** is the address of the schema
- If a prefix is not specified: **xmlns="location"** then the **location** is used by default

Reading Assignment

- Berners-Lee, Hendler, Lassila, The Semantic Web, Scientific American, May 2001 http://kill.devc.at/system/files/scientific-american_0.pdf
- FSWT: Chapter 1 and Appendix A (XML)