Resource Description Framework (RDF)

- By David Wilczynski, USC, dwilczyn@usc.edu

- Based on: http://www.w3.org/TR/rdf-primer/
  - which is edited by:
    - Frank Manola, The MITRE Corporation, fmanola@mitre.org
    - Eric Miller, W3C, em@w3.org

1. Introduction

- RDF is a language for:
  - (1) representing information about resources in the World Wide Web,
  - (2) presenting metadata about Web resources, such as the title and author of a Web page.
- Resources are things that can be *identified* on the Web, even when they can't be directly *retrieved* on the Web. Any person is an example of a resource.
- RDF is computer readable and “understandable.”

An RDF Graph Describing Eric Miller

The same RDF in XML

```
<?xml version="1.0"?>
   <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
   syntax-ns#"
   xmlns:contact="http://www.w3.org/2000/10/swap/pim/conta
   ct#"> <contact:Person
        rdf:about="http://www.w3.org/People/EM/contact#me">
   <contact:fullName>Eric Miller</contact:fullName>
   <contact:mailbox rdf:resource="mailto:em@w3.org"/>
   <contact:personalTitle>Dr.</contact:personalTitle>
   </contact:Person>
   </rdf:RDF>
```

- This says that there is a person whose name is Eric Miller, whose email address is em@w3.org, and whose personal title is "Dr.".

The Big Idea

- Besides Web pages, we can now convey information about cars, businesses, people, news events, etc.
- Further, RDF references themselves can be labeled, to indicate the kind of relationship that exists between the linked items.
- Maybe Web programs can be smarter!!

2. Making Statements About Resources

- RDF is intended to provide a simple way to make statements about Web resources, such as Web pages.
- This section describes how RDF does this.

2.1 Basic Concepts

"http://www.example.org/index.html has a creator whose value is John Smith"

- Every RDF statement is a triple of the form:
  (subject, property, object)
- In the example statement above:
  - the Web page's URL is the *subject* .
  - "creator" is a *property (*or *predicate)* of that page,
  - "John Smith" is the value or *object* of the property.

More Properties

- We can state other properties of this Web page:
  - http://www.example.org/index.html has a creation-date whose value is August 16, 1999
  - http://www.example.org/index.html has a language whose value is English
- In RDF, resources are described in terms of these triples, (subject, property, object).

Uniform Resource Identifiers (URIs)

- In real life we use names to refer to resources: "Bob", "The Moon", "373 Whitaker Ave.", "California", "VIN 2745534", "today's weather".
- But, names are ambiguous.
- To resolve this problem we use URIs to name things in the Web.

Uniform Resource Locators (URLs)

- The Web already provides one form of identifier, the *Uniform Resource Locator* (URL).
- We used a URL in our original example to identify the Web page that John Smith created.
- A URL is a character string that identifies a Web resource by its network location.
- But there are lots of resources besides retrievable ones. Hence, the URI is more general.

There are More than Just Pages on the Web

- We would like to be able to record information about many things in addition to Web pages:
  - For example, a human being has contact information (email, phone), medical information, hobbies, etc.
- But, certainly a human has no URL, though he may have a home page with a URL.
- We must try to formally identify various kinds of things that go by names such as "Social Security Number", or "Part Number" by using URIs.

URI to Name Anything

- We can create a URI to refer to anything we want to talk about, including:
  - network-accessible things, such as an HTML doc.
  - things that are not network-accessible, such as humans, corporations, and books in a library.
  - abstract concepts that don't physically exist, like that of a “unicorn".
- URIs constitute an infinite stock of names.

URIs and RDF

- RDF uses *URI references* to define its subjects, predicates, and objects.
- A URI reference (or *URIref*) is a URI, together with an optional *fragment identifier* at the end.
- E.g., the URI

  http://www.example.org/index.html#section2

  consists of:
  - the URI http://www.example.org/index.html
  - the fragment identifier: section2.
- A *resource* is identifiable by a URI reference

RDF and XML

- RDF is a graph—the object of one statement can be the subject of another.
- Extensible Markup Language (XML) provides us with:
  - a linear representation of this graph;
  - and, as such, a way for exchanging RDF statements between applications.

2.2 The RDF Model

- **In RDF, the English statement:**

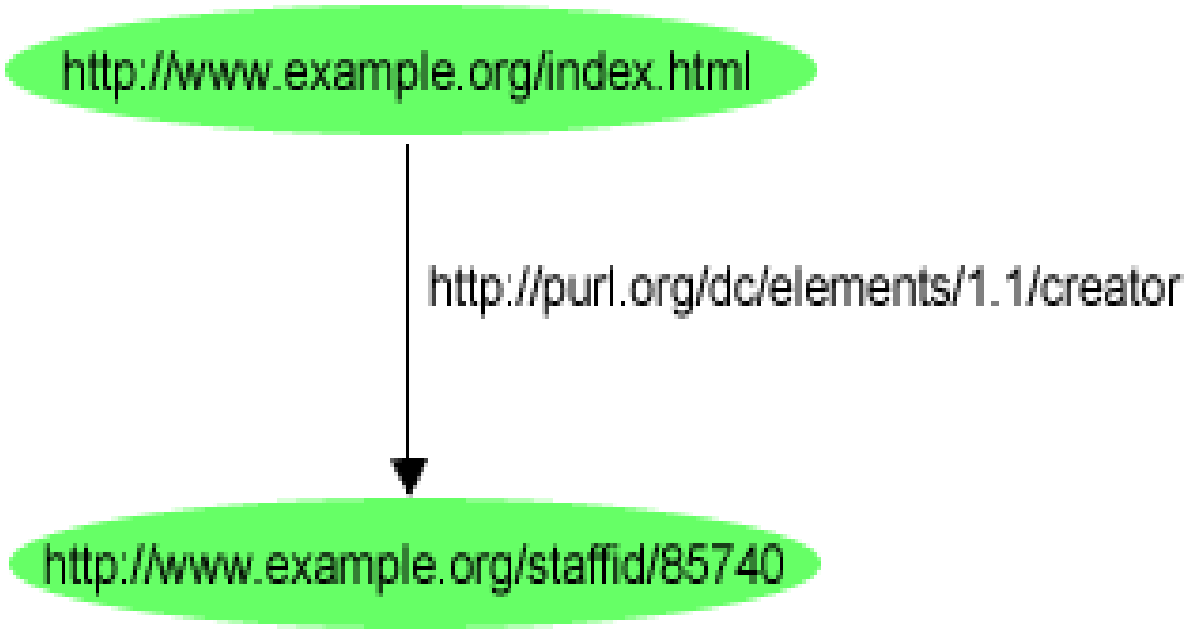
  "http://www.example.org/index.html has a creator
  whose value is John Smith."

  could be represented in RDF as a triple:

  - Subject: http://www.example.org/index.html
  - Predicate:
    http://purl.org/dc/elements/1.1/creator
  - Object: http://www.example.org/staffid/85740

- **Note the URIrefs instead of the words "creator" and "John Smith".**

RDF Nodes and Arcs in a Graph

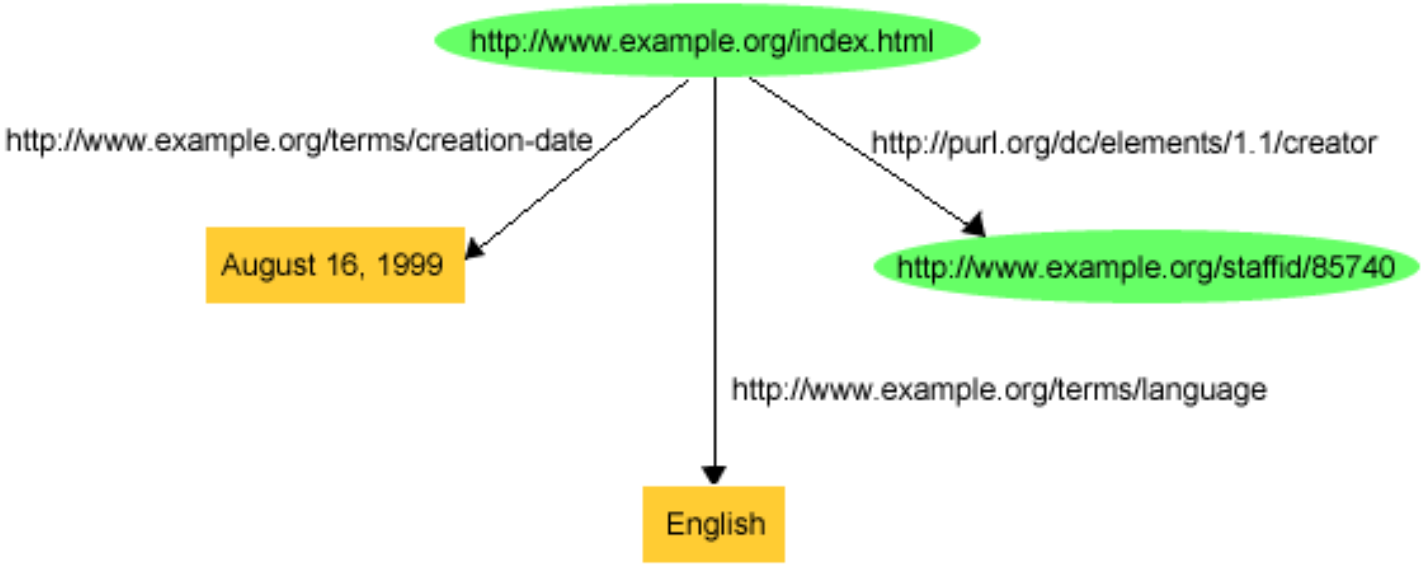


Groups of statements

- Adding new statements

http://www.example.org/index.html has a creation-date of August 16, 1999.

http://www.example.org/index.html has a language whose value is English



Groups of statements (cont.)

- Objects of RDF statements
  - may be resources identified by URIrefs,
  - or constant values (plain or typed literals),
- Literals can't be subjects of RDF statements

Triple Representation

<http://www.example.org/index.html>
<http://purl.org/dc/elements/1.1/creator>
http://www.example.org/staffid/85740 .

<http://www.example.org/index.html>
<http://www.example.org/terms/creation-date>
"August 16, 1999" .

<http://www.example.org/index.html>
<http://www.example.org/terms/language>
"English" .

- Each triple corresponds to a single arc in the graph.
- Triples have the same information as the graph.

Prefix As Namespace Identifier

- Prefix stands for a namespace URI.

ex:index.html dc:creator exstaff:85740 .

ex:index.html exterms:creation-date "August 16, 1999" .

ex:index.html exterms:language "English" .

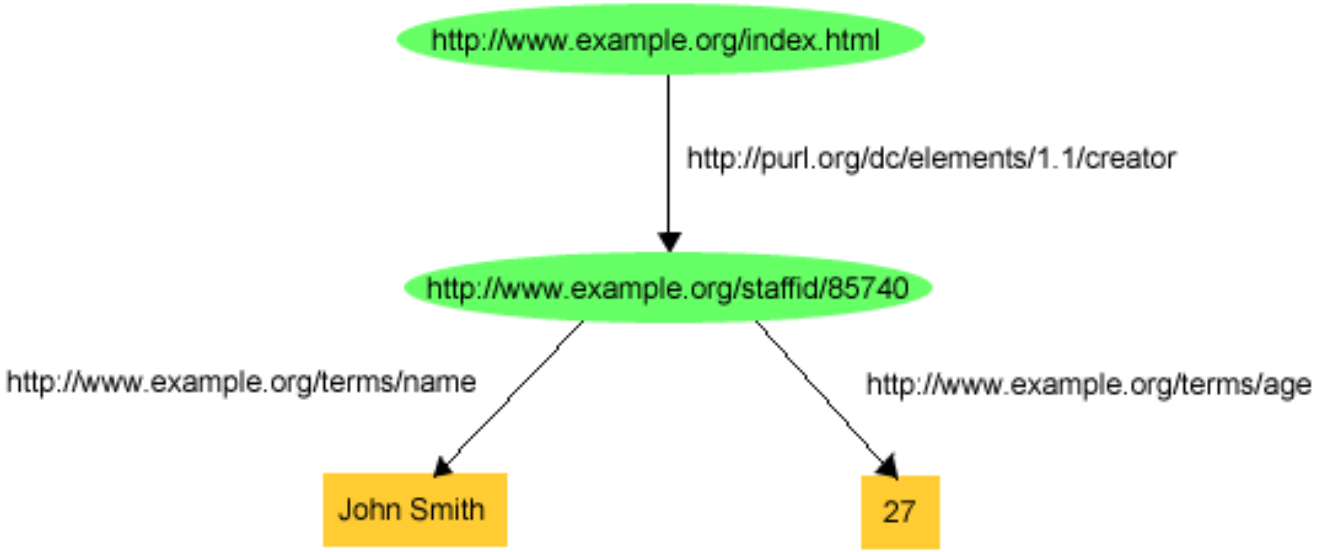
- In the above triples:

  ex:, dc:, exstaff:, and exterms: are URI prefixes.

- A name like **ex:index.html** is called a QName.

Review: URI's Identify Resources

- Creator of web page is identified by a URI.
- The URI has a name property with value "John Smith" and an age property of 27:



Review: RDF uses URIrefs as *predicates*

- The URIref http://www.example.org/terms/name is a predicate; NOT the string “name”.
- URIrefs are unique; strings aren’t.
- Predicates are resources themselves and can have descriptive properties, e.g. printstring

http://www.example.org/terms/name dc:printstring "Name:" .

Review: RDF as Shared Vocabulary

- For example, in the triple:
  ex:index.html dc:creator exstaff:85740 .

  the predicate dc:creator is an unambiguous reference in the Dublin Core metadata attribute set.
- People can still use different URIrefs to refer to the same thing.

Review: Simpler for Applications

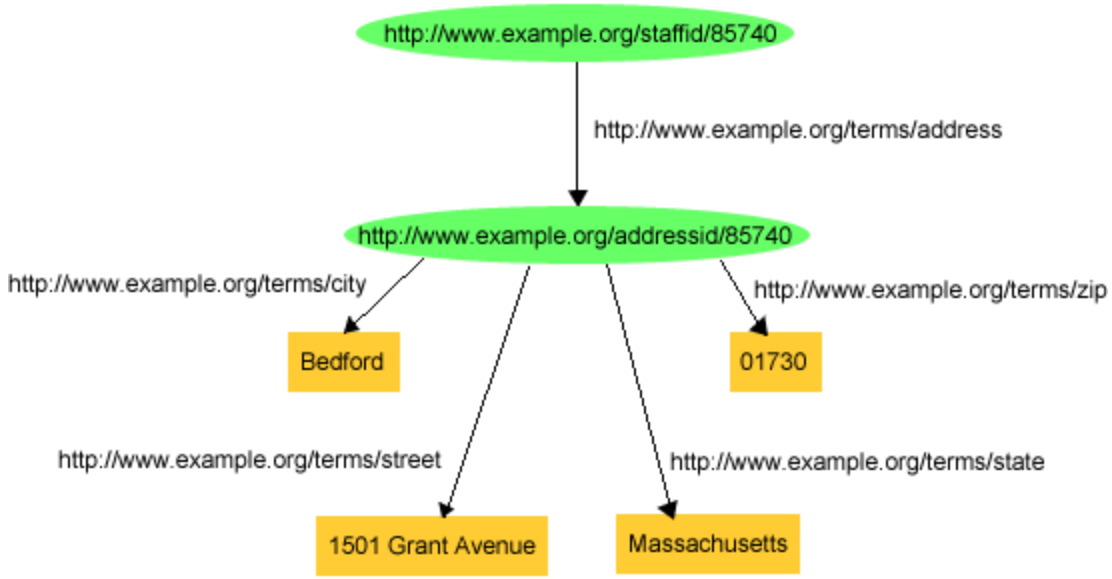
- RDF provides a way to make statements that applications can process more easily:
  - A program could search the Web for all book reviews and create an average rating for each book, and put that information back on the Web.
  - Another site could take that list of averages and create a "Top Ten Highest Rated Books" page.
- Key: a shared vocabulary about books and ratings.

2.3 Structured Property Values and Blank Nodes

- The exterms:address property can be filled by a literal like “1501 Grant Avenue, Bedford, Massachusetts 01730”.
- What about a *structure* consisting of separate street, city, state, and zip code values?
- We could model the address as a resource, give it a URIref say: http://www.example.org/addressid/85740

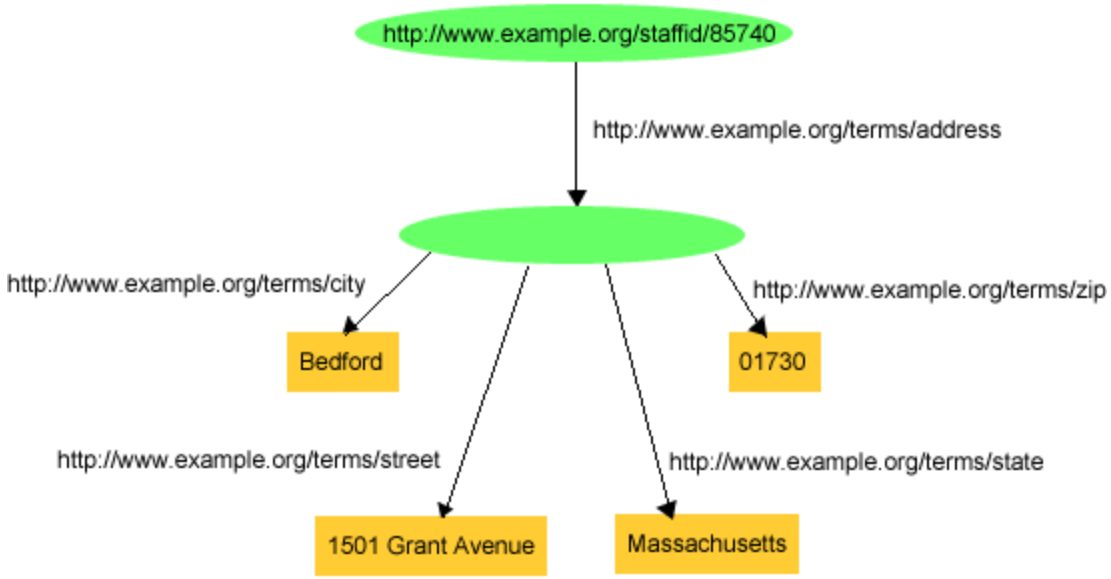
  and then make statements about it.

Something like:



- Nodes like John's address may not require "universal" identifiers.
- Nodes with only local meaning can be blank.

Using A Blank Node



- Here the blank node stands for the concept of "John Smith's address".

Blank Node Identifiers

- *Blank nodes* must have a name for triple usage.
- *Blank node identifiers* have the form `_:name`

```
exstaff:85740  exterms:address  _:johnaddress .
_:johnaddress  exterms:street"1501 Grant Avenue" .
_:johnaddress  exterms:city  "Bedford" .
_:johnaddress  exterms:state "Massachusetts" .
_:johnaddress  exterms:zip"01730" .
```

- If a node in a graph needs to be referenced from outside this context, a URIref is required.
- Blank nodes make binary relationships out of an *n-ary* one (between John and the street, city, etc.).

Blank Nodes For Correct Modeling

- Suppose Jane Smith has no URI but has email: **mailto:jane@example.org.**
- Should we use it as her URI?
- Putting age information about Jane on this URI is plain wrong!!!

Using A Blank Node

- A Blank Node To Represent Jane:

  _:jane exterms:mailbox mailto:jane@example.org .
  _:jane rdf:type exterms:Person .
  _:jane exterms:name "Jane Smith" .
  _:jane exterms:empID "23748" .
  _:jane exterms:age "26" .

- The resource named **_:jane has:**
  - type **exterms:Person**
  - email with value **mailto:jane@example.org**
  - name with value **Jane Smith**
  - etc.

2.4 Typed Literals

- "John is 27 years old." 27 is an integer, not a string.

  <http://www.example.org/staffid/85740>
  <http://www.example.org/terms/age>
  "27"^^<http://www.w3.org/2001/XMLSchema#integer> .

- Using our QName simplification:

  exstaff:85740 exterms:age "27"^^xsd:integer .

- Similarly a date triple might be

  ex:index.html exterms:creation-date "1999-08-16"^^xsd:date .

- Datatypes: http://www.w3.org/TR/xmlschema-2/
  RDFS: http://www.w3.org/TR/rdf-concepts/

Typed literals vs. Programming datatypes

- They are NOT the same.
- A typed literal must be interpreted by an RDF processor that "understands" it.
  - For example, you could write the triple:

    exstaff:85740 exterms:age "pumpkin"^^xsd:integer .

  - A datatype-aware processor would reject it.

2.5 Summary (so far)

- RDF is simple.
- We need to define the *vocabularies* used in those statements.
- RDF vocabularies (schemas) will be described later.

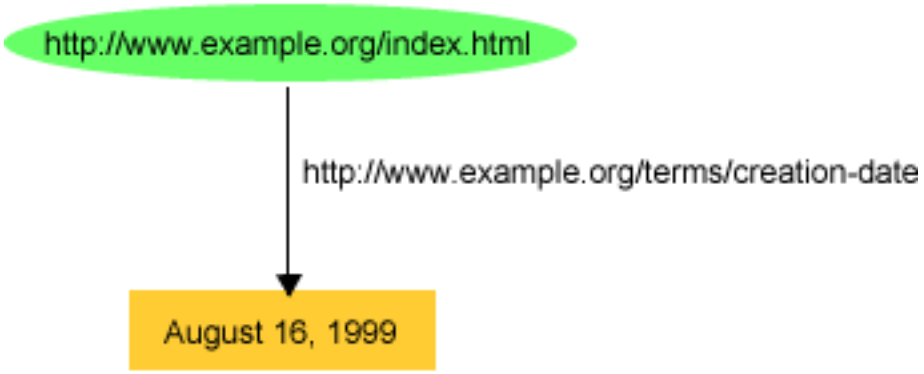
3. An XML Syntax for RDF: RDF/XML

- RDF's conceptual model is a graph of nodes and arcs.
- Triples are one textual, shorthand notation.
- RDF/XML is the normative way of writing down and exchanging RDF graphs.

3.1 Basic Principles

"http://www.example.org/index.html has a creation-date of August 16, 1999"

ex:index.html exterms:creation-date "August 16, 1999" .



In RDF/XML Syntax

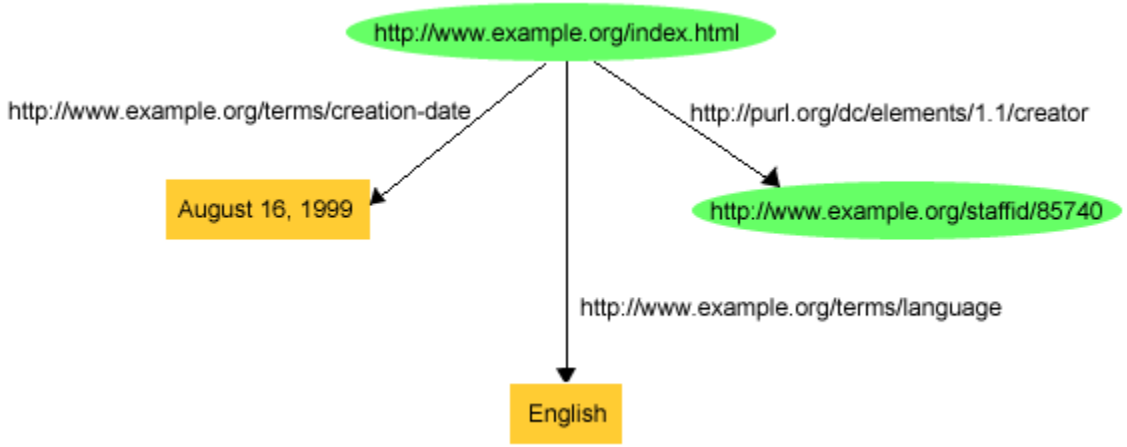
```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:exterms="http://www.example.org/terms/">
   <rdf:Description
        rdf:about="http://www.example.org/index.html">
       <exterms:creation-date>August 16,1999
                  </exterms:creation-date>
   </rdf:Description>
</rdf:RDF>
```

RDF for Multiple Statements

ex:index.html dc:creator exstaff:85740 .

ex:index.html exterms:creation-date "August 16, 1999" .

ex:index.html exterms:language "English" .



Here's the RDF/XML (an abbreviated form)

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
   syntax-ns#"
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:exterms="http://www.example.org/terms/">

 <rdf:Description
   rdf:about="http://www.example.org/index.html">
   <exterms:creation-date>August 16, 1999</exterms:creation-
   date>    <exterms:language>English</exterms:language>
    <dc:creator
   rdf:resource="http://www.example.org/staffid/85740"/>
  </rdf:Description>
</rdf:RDF>
```

- Could have done this with 3 rdf:Description blocks

Empty-Element Tag

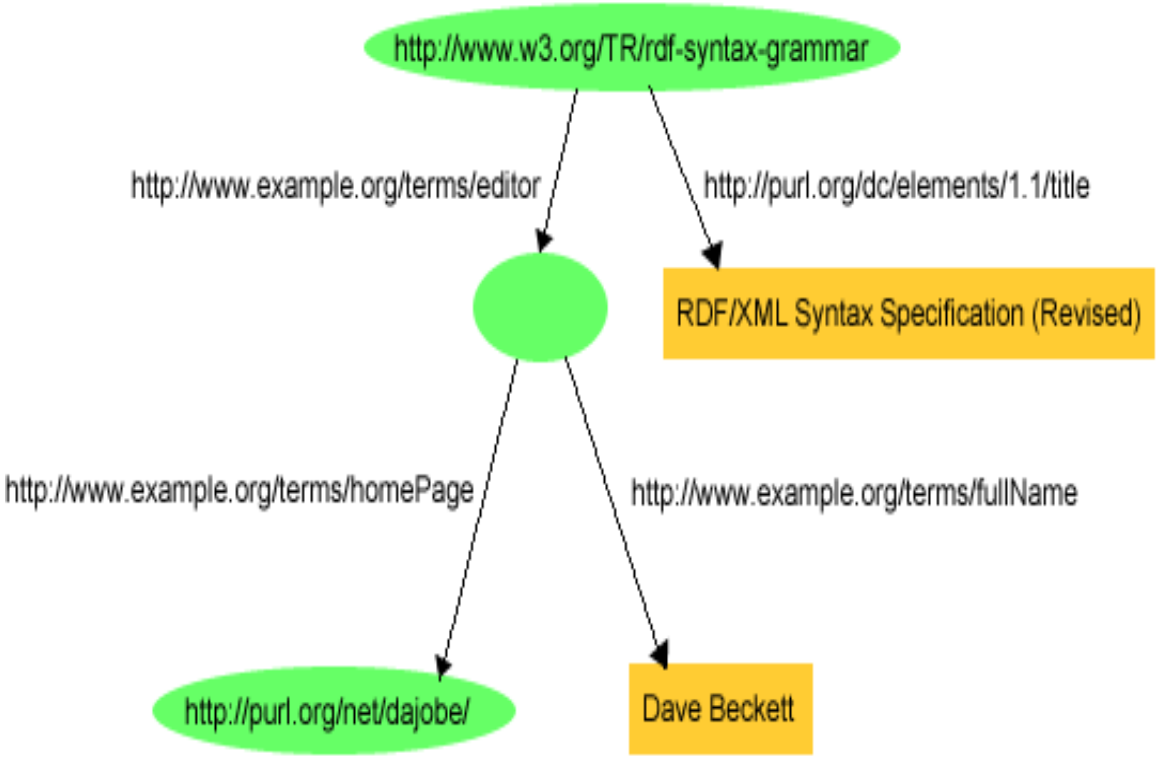
- Notice the dc:creator element with an attribute whose value is *another resource:*
  <dc:creator
  rdf:resource="http://www.example.org/staffid/85740"/>
- This is called an *empty-element tag.*
- If we had instead written:
  <dc:creator>http://www.example.org/staffid/85740</dc:creator>

  That would have defined the creator as a string literal (that looks like a URIref, but isn't).

More RDF/XML Abbreviations

- RDF/XML has many ways to say the same thing. Often, very confusing.
- Consult http://www.w3.org/TR/rdf-primer/#ref-rdf-syntax for more details.

Blank Node Abbreviations



Blank Nodes in RDF/XML

- Use a *blank node identifier* for the blank node when you don't have a URIref for the resource.
  - As a Subject:
    <rdf:Description rdf:nodeID=”someName”>
    instead of
    <rdf:Description rdf:about=“someUriRef">
  - As an object:
    <dc:creator rdf:nodeID="someName"/> instead of <dc:creator rdf:resource=“someUriRef”

RDF/XML For a Blank Node

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
   ns#"       xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:exterms="http://example.org/stuff/1.0/">
<rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-
   grammar">
     <dc:title>RDF/XML Syntax Specification (Revised)</dc:title>
     <exterms:editor rdf:nodeID="abc"/>
</rdf:Description>
 <rdf:Description rdf:nodeID="abc">
   <exterms:fullName>Dave Beckett</exterms:fullName>
   <exterms:homePage
   rdf:resource="http://purl.org/net/dajobe/"/>
 </rdf:Description>
</rdf:RDF>
```

RDF/XML Using a Typed Literal

- Can add a URIref for a datatype as an attribute to a triple as follows:

  ex:index.html exterms:creation-date "1999-08-16" becomes

  ex:index.html exterms:creation-date "1999-08-16"^^xsd:date .

- The RDF/XML part becomes:

```
<exterms:creation-date
        rdf:datatype= "http://www.w3.org/2001/XMLSchema#date">
    1999-08-16
</exterms:creation-date>
```

  where **1999-08-16** is the literal representation for August 16, 1999 in the XML Schema **#date** datatype.

- Can also use XML ENTITY to improve readability (see Primer for details).

Summary: A General Way To Serialize Graphs In RDF/XML.

- All blank nodes are assigned blank node identifiers.
- A subject of an un-nested **rdf:Description** element uses:
  - an rdf:about attribute if the node has a URIref,
  - or an rdf:nodeID attribute if the node is blank.
- Every object of a triple has either:
  - literal value (possibly empty),
  - an rdf:resource attribute if the object has a URIref,
  - or an rdf:nodeID attribute if the object is blank.

3.2 Abbreviating and Organizing RDF URIrefs

- Sometimes we want to achieve the *effect* of assigning URIrefs to resources that are part of an "organizing" resource, like a catalog.
- Imagine a sporting goods company, example.com, producing an RDF-based catalog of its products.
- Suppose the catalog is at:

  http://www.example.com/2002/04/products

- In that catalog resource, each product might be given a separate RDF description using rdf:ID.

Catalog Entries

- RDF/XML for catalog for “Overnighter” tent

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
    xmlns:exterms="http://www.example.com/terms/">
  <rdf:Description rdf:ID="item10245">
    <exterms:model>Overnighter</exterms:model>
    <exterms:sleeps>2</exterms:sleeps>
    <exterms:weight>2.4</exterms:weight>
    <exterms:packedSize>14x56</exterms:packedSize>
  </rdf:Description>
  ...other product descriptions...
</rdf:RDF>
```

Fragment Identifiers

- Notice use of rdf:ID attribute instead of an rdf:about attribute in:

  <rdf:Description rdf:ID="item10245">

- The attribute rdf:ID indicates a *fragment identifier*.
- Its absolute URIref is: http://www.example.com/2002/04/products#item10245.
- Similar to the ID usage attribute in XML and HTML.
- ID must be unique within the document.
- Other statements in this catalog could use an attribute with relative URIref rdf:about="#item10245"

Outsider Referring to the Catalog

- Outsiders could refer to this tent with the full URIref: **http://www.example.com/2002/04/products#item10245.**

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd
    "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
              xmlns:sportex="http://www.exampleRatings.com/terms/">
 <rdf:Description
    rdf:about="http://www.example.com/2002/04/products#item10245">
   <sportex:ratingBy>Richard Roe</sportex:ratingBy>
  <sportex:numberStars
    rdf:datatype="&xsd;integer">5</sportex:numberStars>
 </rdf:Description>
</rdf:RDF>
```

Outsider Referring to the Catalog

- Note that RDF does not assume any particular relationship exists between:

  http://www.example.com/2002/04/products#item10245 and
  http://www.example.com/2002/04/products
- Having the same base "means" nothing. They are just two resources.
- This further illustrates that the RDF describing a particular resource does not need to be located all in one place.

Base URI

- Fragment identifiers such as #item10245 will be interpreted relative to a *base URI*.
- By default, this base URI is the resource in which the fragment identifier is used.
- Not always desirable—Consider use of mirror sites.

Base URI (cont.)

- RDF/XML supports XML Base.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:exterms="http://www.example.com/terms/"
         xml:base="http://www.example.com/2002/04/products">
 <rdf:Description rdf:ID="item10245">
         …
 </rdf:Description>
  ...other product descriptions...
</rdf:RDF>
```

- Our tent, **#item10245**, will have the same URIref,

  http://www.example.com/2002/04/products#item10245

  no matter what the URI of the catalog is.

RDF types (or *classes)*

- RDF supports types using a property rdf:type
- The value of the rdf:type property is a resource.
- The subject of the property is an *instance* of the type:

```
<rdf:Description rdf:ID="item10245">
   <rdf:type rdf:resource="http://www.example.com/terms/Tent" />
   …
</rdf:Description>
```

- So, **item10245** is an *instance* of **http://www.example.com/terms/Tent**
- Types are normally defined in an *RDF Schema*.

An Abbreviation for rdf:type

- The **rdf:Description** element is replaced as follows:

```
<exterms:Tent rdf:ID="item10245">
    <exterms:model>Overnighter</exterms:model>
    <exterms:sleeps>2</exterms:sleeps>
    <exterms:weight>2.4</exterms:weight>
    <exterms:packedSize>14x56</exterms:packedSize>
</exterms:Tent>
```

- More like plain XML. More readable.
- If object has more that one type, add <rdf:type …> statements as needed.

4. Other RDF Capabilities

- Containers
- Collections
- Reification
- Structured Values

4.1 RDF Containers

- We need to describe *groups* of things:
  - a book created by several authors,
  - a list of students in a course.
- RDF's *container vocabulary* consists of bags, sequences, and alternative and some associated properties.

Bag (rdf:Bag)

- A *Bag* is a resource having type rdf:Bag.
- A Bag is a unordered group of resources or literals, possibly including duplicate members.
- For example, a Bag might model a group of part numbers used in assembling a motor. There might be duplicates (same part types used many times) and order doesn't matter.

Sequence (rdf:Seq)

- A *Sequence* is a resource having type rdf:Seq.
- A Sequence is a group of resources or literals, possibly including duplicate members, where the order of the members is significant.
- For example, a Sequence might be used to describe a group that must be maintained in alphabetical order.

Alternative (rdf:Alt)

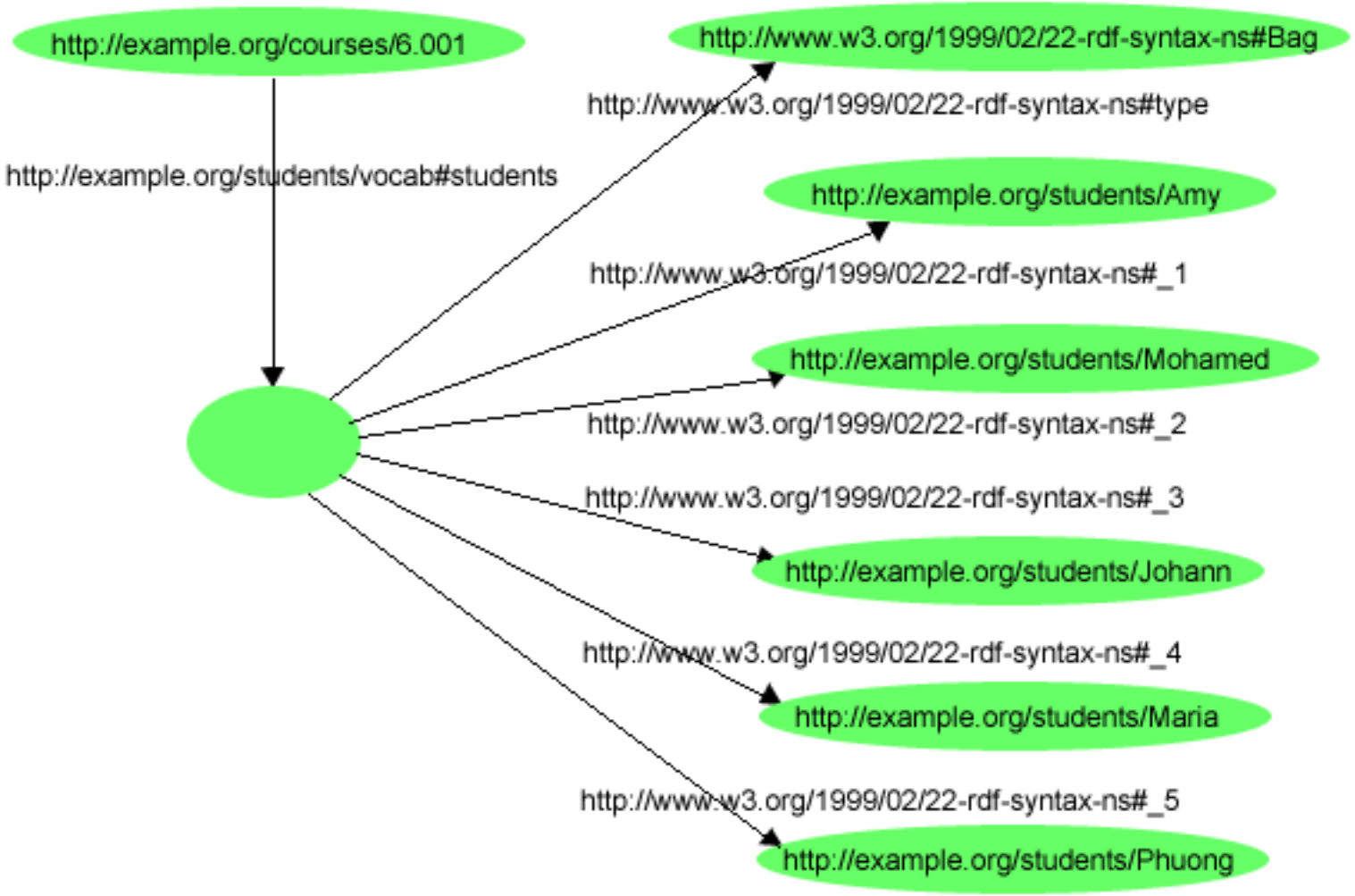
- An *Alternative* is a resource having type rdf:Alt.
- An Alternative is a group of resources or literals that are *alternatives* (typically for a single value of a property).
- For example, an Alt might be used to describe a list of alternative Internet sites at which a resource might be found.

Using Containers

- Give the resource an rdf:type property with value rdf:Bag, rdf:Seq, or rdf:Alt
- The container resource (which may either be a blank node or a resource with a URIref) denotes the group as a whole.
- The *members* of the container use a *container membership* with names of the form rdf:_*n*, where *n* > 0, e.g., rdf:_1, rdf_2, rdf_3

A Bag Example

- Let's represent the sentence: "Course 6.001 has the students Amy, Mohamed, Johann, Maria, and Phuong."



RDF/XML Syntax For this Graph

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://example.edu/students/vocab#">

  <rdf:Description rdf:about="http://example.edu/courses/6.001">
    <s:students>
      <rdf:Bag>
        <rdf:li rdf:resource="http://example.edu/students/Amy"/>
        <rdf:li rdf:resource="http://example.edu/students/Mohamed"/>
        <rdf:li rdf:resource="http://example.edu/students/Johann"/>
        <rdf:li rdf:resource="http://example.edu/students/Maria"/>
        <rdf:li rdf:resource="http://example.edu/students/Phuong"/>
      </rdf:Bag>
    </s:students>
  </rdf:Description>
</rdf:RDF>
```

Some Abbreviations

- RDF/XML provides li as a convenience to avoid having to number each membership property.
  - The numbered properties rdf:_1, rdf:_2, etc. are generated from the li elements in forming the corresponding graph.
- The use of a <rdf:Bag> element within the <s:students> property element.
  - The <rdf:Bag> element is abbreviation that lets us replace both an rdf:Description element and an rdf:type element with a single element.
  - The Bag is a blank node. Its nesting within the <s:students> property element is an abbreviated way of indicating it is the value of this property.

Sequences and Graph Structure

- The graph structure for an rdf:Seq container, and the corresponding RDF/XML, are similar to those for an rdf:Bag.
- The only difference is in the type, rdf:Seq.
- Remember, although an rdf:Seq container is intended to describe a sequence, it is up to applications creating and processing the graph to appropriately interpret the sequence of integer-valued property names.

Alternatives and Graph Structure

- The graph structure for an rdf:Alt container, and the corresponding RDF/XML, are similar to those for an rdf:Bag.
- An Alt container has at least one member, rdf:_1, which is the default value.
- Other than rdf:_1, the order of the remaining elements is not significant.

A Modeling Issue (example 1)

- Consider the sentence: “Sue has written Anthology of Time, Zoological Reasoning, and Gravitational Reflections.” It could be:

exstaff:Sue exterms:publication ex:AnthologyOfTime .

exstaff:Sue exterms:publication ex:ZoologicalReasoning .

exstaff:Sue exterms:publication ex:GravitationalReflections .

- Or, this model perhaps:

exstaff:Sue exterms:publication _:z

_:z rdf:type rdf:Bag .

_:z rdf:_1 ex:AnthologyOfTime .

_:z rdf:_2 ex:ZoologicalReasoning .

_:z rdf:_3 ex:GravitationalReflections .

A Modeling Issue (example 2)

- Now, consider: “The resolution was approved by the Rules Committee, having members Fred, Wilma, and Dino.” The following is wrong:

ex:resolution exterms:approvedBy ex:Fred .
ex:resolution exterms:approvedBy ex:Wilma .
ex:resolution exterms:approvedBy ex:Dino .

- since these statements say that each member individually approved the resolution. Correct is:

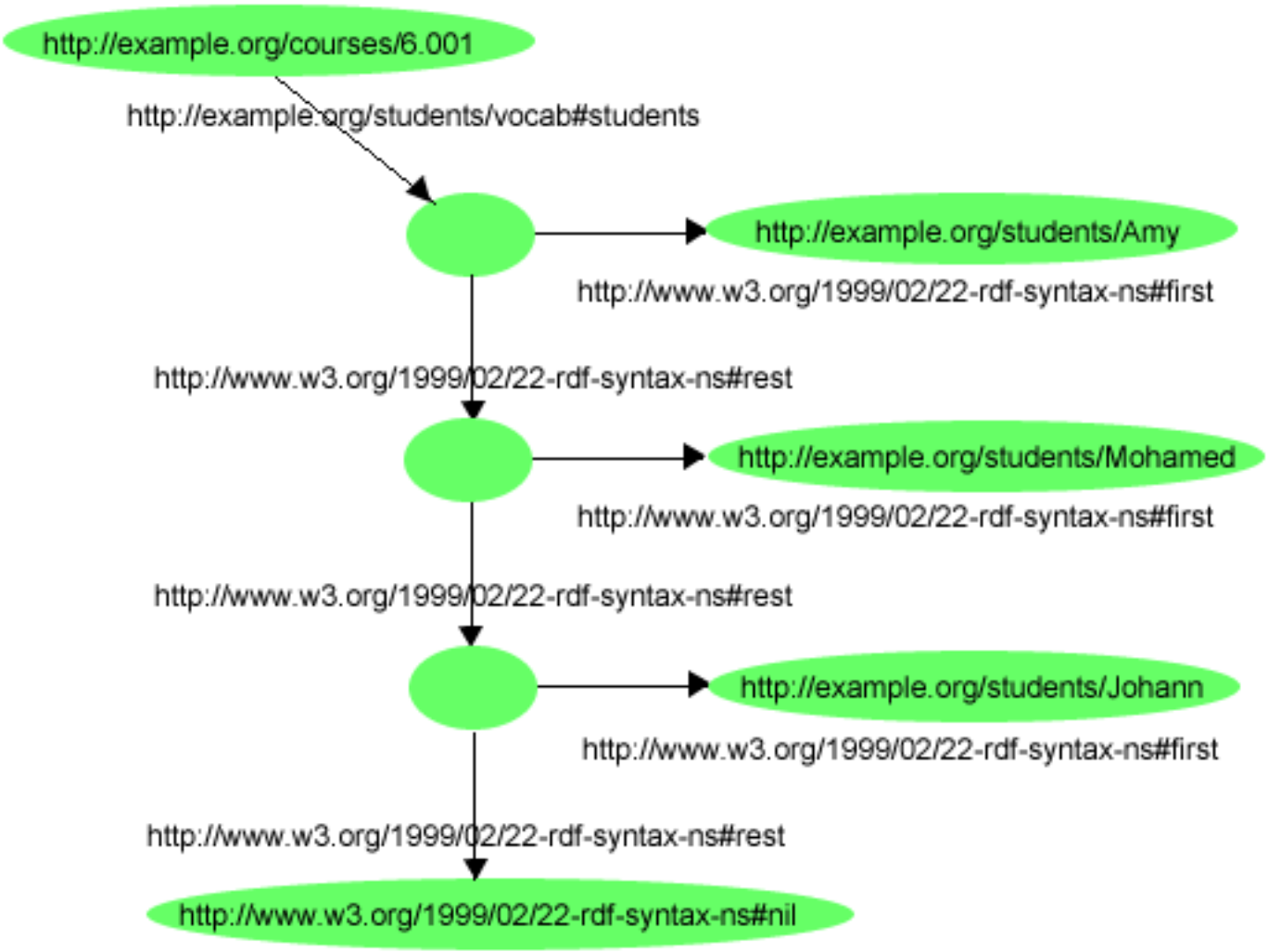
ex:resolution exterms:approvedBy ex:rulesCommittee
ex:rulesCommittee rdf:type rdf:Bag .
ex:rulesCommittee rdf:_1 ex:Fred .
ex:rulesCommittee rdf:_2 ex:Wilma .
ex:rulesCommittee rdf:_3 ex:Dino .

4.2 RDF Collections

- With containers there is no way to say that these are all the members of the container.
- The graph has no way to exclude the possibility that there is another graph somewhere that describes additional members.
- RDF *collections* can describe "closed" groups.
- An RDF collection is a LISP-like list of type rdf:List, with predefined properties rdf:first and rdf:rest, and the predefined resource rdf:nil.

A Collection Example

- Consider the sentence "The students in course 6.001 are Amy, Mohamed, and Johann":



RDF/XML for The Collection of Students (notice abbreviations)

```
<?xml version="1.0"?>
<rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:s="http://example.edu/students/vocab#">

  <rdf:Description rdf:about="http://example.edu/courses/6.001">
    <s:students rdf:parseType="Collection">
        <rdf:Description rdf:about="http://example.edu/students/Amy"/>
        <rdf:Description rdf:about="http://example.edu/students/Mohamed"/>
        <rdf:Description rdf:about="http://example.edu/students/Johann"/>
    </s:students>
  </rdf:Description>
</rdf:RDF>
```

4.3 Reification in RDF – Making Statements about Statements

- Suppose we have the triple:

  exproducts:item10245 exterms:weight "2.4"^^xsd:decimal .

- Now suppose we want to model that John Smith made this statement.
- We want something like:

  [**exproducts:item10245 exterms:weight "2.4"^^xsd:decimal .**]
  dc:creator exstaff:85740 .

- That is, to turn the original statement into a resource, i.e., reify it, so it can be a Subject.

RDF Reification Vocabulary

- RDF supplies:
  - a type: rdf:Statement,
  - and properties: rdf:subject, rdf:predicate, rdf:object.
- So, a *reification* of our original triple:

  exproducts:item10245 exterms:weight "2.4" .
- is given by the triples:

  exproducts:triple12345 rdf:type rdf:Statement .
  exproducts:triple12345 rdf:subject exproducts:item10245 . exproducts:triple12345 rdf:predicate exterms:weight .
  exproducts:triple12345 rdf:object "2.4"^^xsd:decimal .

- And we can add:

  exproducts:triple12345 dc:creator exstaff:85740 .

  to represent that 85740 made the statement.

Be Careful!!

- The above means that **triple12345** refers to is a *particular instance* of a triple in a particular RDF document, rather than some arbitrary triple having the same subject, predicate, and object.
- Suppose Jane Doe “also” said that item10245 weighed 2.4.
- How would you model it?
- With the same statement, **triple12345** , or with another statement, say **triple7890**, that has the same subject, predicate, and object?

The Reified Statement is not the Same as the Statement!

- When someone asserts that John said foo, they are not asserting foo themselves, just that John said it.
- Conversely, when someone asserts foo, they are not also asserting its reification.
- RDF can't "connect" an triple to its reification.
- **triple12345** has NO graph connection to the original triple:

  exproducts:item10245 exterms:weight "2.4" .

- And adding: **triple12345** dc:creator exstaff:85740 . does not allow you to say that John created the original triple.

Be Careful (cont.)

- We could attribute the statement to John simply by the statement:

  ex:triple12345 dc:creator exstaff:85740 .

- Now, if Jane were exstaff:900 and you asserted:

  ex:triple12345 dc:creator exstaff:900 .

- You would be saying that John and Jane made the SAME statement. Is that likely? For AI'ers to argue.

4.4 More on Structured Values: rdf:value

- We used blank nodes to turn n-ary properties into binary ones (like the address example).
- Often the blank node has one property which is its value. In our tent example, we said

exproduct:item10245 exterms:weight "2.4"^^xsd:decimal .

- A better description would include a “units” property, with 2.4 being the “value.” Perhaps:

exproduct:item10245 exterms:weight _:weight10245 .
_:weight10245 rdf:value "2.4"^^xsd:decimal .
_:weight10245 exterms:units exunits:kilograms .

Structured Values In RDF/XML

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd
    "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:exterms="http://www.example.org/terms/">

  <rdf:Description
    rdf:about="http://www.example.com/2002/04/products#item10245">
    <exterms:weight rdf:parseType="Resource">
      <rdf:value rdf:datatype="&xsd;decimal">2.4</rdf:value>
      <exterms:units rdf:resource="http://www.example.org/units/kilograms" />
    </exterms:weight>
  </rdf:Description>
</rdf:RDF>
```

- You could assign your own property name, such as **ex:amount**, instead of **rdf:value.**

4.5 XML Literals

- Sometimes the value of a property needs to be a fragment of XML, or text that might contain XML markup.
- Giving an element the attribute `rdf:parseType="Literal"` indicates that the contents of the element are to be interpreted as an XML fragment.

RDF/XML Fragment using an XML Literal

```
<rdf:Description
rdf:ID="book12345">
    <dc:title rdf:parseType="Literal">
      <span xml:lang="en">
        The <em>&lt;br /&gt;</em>
        Element Considered Harmful.
      </span>
    </dc:title>
</rdf:Description>
```

5. Defining RDF Vocabularies: RDF Schema

- RDF Schema provides a way to express:
  - simple statements defining classes of resources including subclass relationships,
  - statements defining properties including subclass relationships,
  - statements about domain and range of a property.

RDF Schema: A meta-language

- RDF Schema's *type system* is similar to those of object-oriented programming languages.
- RDF Schema allows resources to be defined as instances of one or more *classes.*
- Classes can be organized in a hierarchical fashion; for example a class ex:Dog might be defined as a subclass of ex:Mammal, meaning that any resource which is in class ex:Dog is also in class ex:Mammal.
- The RDF Schema (RDFS:) is defined in a namespace whose URI is: http://www.w3.org/2000/01/rdf-schema#".

5.1 Describing Classes: A MotorVehicle class

- To say that ex:MotorVehicle is a class, write:
  ex:MotorVehicle rdf:type rdfs:Class .
- To create an instance of ex:MotorVehicle, write:
  exthings:companyCar rdf:type ex:MotorVehicle .
- Convention:
  - class names start with an uppercase letter;
  - property and instance names are lowercase.
- A resource may be an instance of more than one class.

Defining Subclasses

- We can now define specialized kinds of motor vehicles, e.g., passenger vehicles, vans, minivans, and so on.

ex:Van rdf:type rdfs:Class .

ex:Van rdfs:subClassOf ex:MotorVehicle .

ex:Truck rdf:type rdfs:Class .

ex:Truck rdfs:subClassOf ex:MotorVehicle .

Meaning of Subclass

- subClassOf means if ex:myVan is an instance of ex:Van, then ex:myVan is also, by inference, an instance of ex:MotorVehicle.
- subClassOf is (obviously) *transitive*:
  - If **ex:Van rdfs:subClassOf ex:MotorVehicle .**
  - and **ex:MiniVan rdfs:subClassOf ex:Van .**
  - then **ex:MiniVan** is implicitly a subclass of **ex:MotorVehicle**.
- A class may be a subclass of more than one class. All classes are implicitly subclasses of class rdfs:Resource.

A Full Class Hierarchy

- The (ex:Truck rdf:type rdfs:Class) part of the graph is not shown.
- Notice Minivan is subClassOf two classes. (next slide as well)

Vehicle Hierarchy in RDF/XML

```
<?xml version="1.0"?>
<rdf:RDF    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

<rdf:Description rdf:ID="MotorVehicle">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>

<rdfs:Class rdf:ID="PassengerVehicle">
   <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdfs:Class>
...
<rdfs:Class rdf:ID="MiniVan">
 <rdfs:subClassOf rdf:resource="#Van"/>
 <rdfs:subClassOf rdf:resource="#PassengerVehicle"/>
</rdfs:Class >
</rdf:RDF>
```

Class Naming

- Fragment identifiers, like MotorVehicle, use rdf:ID give the effect of "assigning" URIrefs relative to the schema document.
- Relative URIrefs based on these names can then be used in other class definitions within the same schema, e.g., #MotorVehicle.
- The full URIref of this class would be: http://example.org/schemas/vehicles#MotorVehicle
- We could also include an explicit declaration: xml:base="http://example.org/schemas/vehicles"

Creating Instances of ex:MotorVehicle (notice both methods)

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:ex="http://example.org/schemas/vehicles">

  <rdf:Description rdf:ID="companyCar">
    <rdf:type rdf:resource="http://example.org/schemas/vehicles#MotorVehicle"/>
  </rdf:Description>
  <ex:MotorVehicle  rdf:ID="anotherCar">
        …
  </ex:MotorVehicle>
</rdf:RDF>
```

5.2 Describing Properties

- All properties in RDF are described as instances of class rdf:Property, e.g.

  exterms:weightInKg rdf:type rdf:Property .

- RDF Schema provides rdfs:range to define valid fillers for a triple's *Object*.
- RDF Schema provides rdfs:domain to define valid fillers for a triple's *Subject*.

The **rdfs:range** Property

- If the property ex:author has values that are instances of class ex:Person, we would write:

  ex:Person rdf:type rdfs:Class .
  ex:author rdf:type rdf:Property .
  ex:author rdfs:range ex:Person .

- If a property has more than one range, then its filler must be an instance of *all* of the classes specified as the ranges:

  ex:hasMother rdf:type rdf:Property .
  ex:hasMother rdfs:range ex:Person .
  ex:hasMother rdfs:range ex:Female .
  ex:Sally ex:HasMother exstaff:frances

- **exstaff:frances** must be both a Female and a Person.

Typed Literals As Ranges

- To say that the range of ex:age is an integer:

  ex:age rdf:type rdf:Property .
  ex:age rdfs:range xsd:integer .

- The datatype xsd:integer is identified by its URIref (http://www.w3.org/2001/XMLSchema#integer).
- It is optional, but “useful” to declare:

  xsd:integer rdf:type rdfs:Datatype .

- This statement documents the existence of the datatype, and indicates explicitly that it is being used in this schema.

The RDF **rdfs:domain** Property

- **rdfs:domain** indicates that a particular property applies to a class.
- Suppose books have authors. In RDF:

  ex:Book rdf:type rdfs:Class .

  ex:author rdf:type rdf:Property .

  ex:author rdfs:domain ex:Book .

- If a property has more than one domain, then any subject instance of that property must be an instance of each named domain.

Some of the RDF/XMLVehicle Schema

```
<rdf:Description rdf:ID="registeredTo">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:domain rdf:resource="#MotorVehicle"/>
  <rdfs:range rdf:resource="#Person"/>
</rdf:Description>

<rdf:Property rdf:ID="rearSeatLegRoom">
  <rdfs:domain rdf:resource="#PassengerVehicle"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/>
</rdf:Property>

<rdfs:Class rdf:ID="Person" />
```

Specializing Properties

- Like rdfs:subClassOf, we have rdfs:subPropertyOf to define a property hierarchy.
- For example, to say that the property ex:primaryDriver is a kind of ex:driver, write:

  ex:driver rdf:type rdf:Property .
  ex:primaryDriver rdf:type rdf:Property .
  ex:primaryDriver rdfs:subPropertyOf ex:driver .

- This means that if an instance ex:fred is a ex:primaryDriver of the instance ex:companyVan, then ex:fred is also a ex:driver of ex:companyVan.

More About Subproperties

- A property may be a subPropertyOf zero, one or more properties.
- All RDF rdfs:range and rdfs:domain properties that apply to an RDF property also apply to each of its subproperties.
- So, ex:primaryDriver, because of its subproperty relationship to ex:driver, implicitly also has an rdfs:domain of ex:MotorVehicle.

In RDF/XML

```
<rdf:Description rdf:ID="driver">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-
    rdf-syntax-ns#Property"/>
  <rdfs:domain rdf:resource="#MotorVehicle"/>
</rdf:Description>

<rdf:Property rdf:ID="primaryDriver">
    <rdfs:subPropertyOf rdf:resource="#driver"/>
</rdf:Property>
```

An Instance of **ex:PassengerVehicle**

```
<?xml version="1.0"?>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:ex="http://example.org/schemas/vehicles">

 <rdf:Description rdf:ID="johnSmithsCar">
    <rdf:type rdf:resource="http://example.org/schemas/vehicles#PassengerVehicle"/>
    <ex:registeredTo rdf:resource="http://www.example.org/staffid/85740"/>
    <ex:rearSeatLegRoom rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">
          127</ex:rearSeatLegRoom>
    <ex:primaryDriver rdf:resource="http://www.example.org/staffid/85740"/>
 </rdf:Description>
</rdf:RDF>
```

Some Details Worth Reviewing

- We assumed this instance was in a separate document from the schema http://example.org/schemas/vehicles.
- Using the namespace:

  xmlns:ex="http://example.org/schemas/vehicles"

  allows us abbreviations such as ex:registeredTo.
- However, in a rdf:type property, use the full URIref and not a QName using the ex: prefix.
- Remember this abbreviation?

```
<ex:PassengerVehicle rdf:ID="johnSmithsCar">
        ...
</ex:PassengerVehicle>
```

5.3 Interpreting RDF Schema Declarations

- Schemas are not *prescriptive* like programming language class definitions.
  - A Java class Book with an author attribute having values of type Person is usually interpreted as a group of *constraints*.
  - An instance of Book will have an author attribute that must be an object of class Person.
- Moreover, if author is the *only* attribute defined for class Book, the language will not allow an instance of Book to have other attributes.

Schema Usage is Application Dependent

- RDF Schema simply offers *descriptions* of resources, but not rules about how these descriptions should be used. Consider the property: (ex:author rdfs:range ex:Person)
- This property might be used in different ways:
  - as a constraint template for RDF data being created as it might be in a programming language;
  - as meta information to help decode untyped data it is receiving. (`ex:author` data must be a `ex:Person`.);
  - as meta information to validate some received data. If the object of an `ex:author` property is also an instance of `ex:Corporation` something is wrong;
  - and, finally, another application may not care that instance of `ex:Book` has no `ex:author` property.

5.4 Other Schema Information

- RDF Schema also provides documentation properties:
  - **rdfs:comment** for the obvious use.
  - **rdfs:label** to provide a more human-readable version of a resource's name.
  - **rdfs:seeAlso** to indicate a resource that might provide additional information about the subject resource.
  - **rdfs:isDefinedBy** property is a subproperty of rdfs:seeAlso

5.5 Richer Schema Languages

- RDF Schema is missing some capabilities:
  - *cardinality constraints* on properties, e.g., that a Person has *exactly one* biological father.
  - specifying that a given property (such as **hasAncestor**) is *transitive*, e.g., that if A **hasAncestor** B, and B **hasAncestor** C, then A **hasAncestor** C.
  - specifying that a given property is a unique identifier (or *key*) for instances of a particular class.
  - specifying that two different classes (having different URIrefs) actually represent the same concept.

Richer Schema Languages (cont.)

  - specifying that two different instances (having different URIrefs) actually represent the same individual.
  - to describe new classes in terms of combinations (e.g., unions and intersections) of other classes,
  - to say that two classes are disjoint (i.e., that no resource is an instance of both classes).
- These and more are the targets of *ontology* languages such as DAML+OIL and OWL.
- Both are based on RDF and RDF Schema.
- The development of such languages is a part of the Semantic Web effort.

6. Some RDF Applications

- Dublin Core Metadata Initiative
  - The Dublin Core is a set of "elements" (properties) for describing documents (and hence, for recording metadata).
- PRISM
  - Publishers want to (re)use existing content in many ways. Converting magazine articles to HTML for posting on the web is one example, reusing parts is another.
- XPackage
  - To maintain information about structured groupings of resources and their associations that are, or may be, used as a unit. The XML Package (XPackage) specification provides a framework for defining such groupings.

Some More RDF Applications

- RSS 1.0: RDF Site Summary
  - RSS 1.0 is a powerful and extensible way of describing, managing and making available to broad audiences relevant and timely news information.
- CIM/XML
  - A set of common definitions of power system entities. The Electric Power Research Institute (EPRI) developed a Common Information Model (CIM).
- Gene Ontology Consortium
  - The objective of the Gene Ontology (GO) Consortium is to provide controlled vocabularies to describe specific aspects of gene products.
- Describing Device Capabilities and User Preferences
  - The W3C's Composite Capabilities/Preferences Profile (CC/PP) specification defined a generic framework for describing a delivery context for mobile devices.

7. Other Parts of the RDF Specification

- RDF Semantics
  - RDF statements also have a *formal* meaning which determines the conclusions (or *entailments*) that machines can draw from an RDF graph.
  - The RDF Semantics defines this formal meaning, using a technique called *model theory*.

Other Parts of the RDF Specification

- Test Cases
  - Positive and Negative Parser Tests: These test whether RDF/XML parsers produce a correct N-triples output graph from legal RDF/XML input documents, or correctly report errors if the input documents are not legal RDF/XML.
  - Positive and Negative Entailment Tests: These test whether proper entailments (conclusions) are or are not drawn from sets of specified RDF statements.
  - Datatype-aware Entailment Tests: These are positive or negative entailment tests that involve the use of datatypes, and hence require additional support for the specific datatypes involved in the tests.
  - Miscellaneous Tests: These are tests that do not fall into one of the other categories.

Appendix A – More about URIs

URI Schemes

- Different *URI schemes* already exist:
  - http: (Hypertext Transfer Protocol for Web pages)
  - mailto: (email addresses), e.g., mailto:em@w3.org
  - ftp: (File Transfer Protocol)
- urn: Uniform Resource Names are persistent *location-independent* resource identifiers, e.g., urn:isbn:0-520-02356-0 (for a book)
- No one person or organization controls who makes URIs or how they can be used.
- URIs are defined in RFC 2396

Who controls URIs?

- No one person or organization controls who makes URIs or how they can be used.
- Some URI schemes, such as URL's http domain name, depend on centralized systems such as DNS, others, such as freenet:, are decentralized.
- This means that, as with any other kind of name, you don't need special authority to create a URI for something, even if you don’t own it.

Relative and Absolute URIrefs

- An *absolute* URIref refers to a resource independently of the context in which the URIref appears, e.g., the URIref http://www.example.org/index.html.
- A *relative* URIref has its prefix come from context
  - otherpage.html has the absolute URIref http://www.example.org/otherpage.html.
  - #section2 is equivalent to the absolute URIref http://www.example.org/index.html#section2.

URIs and Retrievability

- RDF use URIrefs to identify things.
- Web browsers may use URIrefs to *retrieve* things.
- However, a URIref may identify something, such as a person, that *cannot* be retrieved on the web.
- There a convention that a page containing descriptive information about a resource is retrievable "at" its URI.

Browsers and Fragment Identifiers

- Browsers handle fragment identifiers differently.
- Fragment identifiers in HTML documents identify a specific place within the document.
  - http://www.example.org/index.html
  - http://www.example.org/index.html#Section2
- As far as RDF is concerned, these two URI references are syntactically different, and hence may refer to unrelated things.