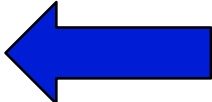




- Vittore Casarosa
 - casarosa@isti.cnr.it
 - Office: 050 621 3115
 - Mobile: 348 397 2168
 - Skype: vittore1201
- “Ricevimento” at the end of the lessons or by appointment
- Final assessment
 - 70% oral examination
 - 30% project (development of a small digital library))
- Reference material:
 - Ian Witten, David Bainbridge, David Nichols, How to build a Digital Library, Morgan Kaufmann, 2010, ISBN 978-0-12-374857-7 (Second edition)
 - Material provided by the teacher
- **<http://cloudone.isti.cnr.it/casarosa/BDG/>**

Modules



- Computer Fundamentals and Networking
- A conceptual model for Digital Libraries
- Bibliographic records and metadata
- Information Retrieval and Search Engines
- Knowledge representation 
- Digital Libraries and the Web
- Hands-on laboratory: the Greenstone system

Parallel evolution



Libraries

- Description (documents)
 - Bibliographic records
 - MARC
- Interoperability
 - Z39-50
- Conceptual model (classes)
 - FRBR – LRM for Works, Expr., Manif.
- Information Retrieval
 - Full text (catalogue and documents)

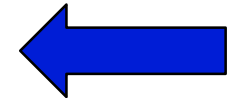
The Web

- Description (instances)
 - Metadata
 - Dublin Core
- Interoperability
 - OAI-PMH
- Conceptual model (classes)
 - **RDF and RDF Schema for all resources (ontologies)**
- Information Retrieval
 - Full text (web pages and resources)

Knowledge representation



- FRBR: Functional Requirements for Bibliographic Records
 - LRM: Library Reference Model
- RDF: Resource Description Framework
- RDF Schema
 - LOD: Linked Open Data



Representation of knowledge



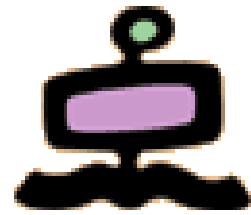
- Description of Information (resources) is an exercise in “knowledge representation”
- Description with metadata (such as Dublin Core) only captures the “syntactic” aspects of the resource
- It would be good to have also a “semantic” description (capturing our “knowledge” of the resource)
- Knowledge representation is the “Holy Graal” of Computer Science (Artificial Intelligence, Expert Systems, Ontologies, etc.)
- Many models/languages proposed in the last 50 years
 - Basically all of them are based on the “entity-relationship model”
- Most of the advances available today are due to “brute force” methods (computing power, today “neural networks”)
- Two conceptual models of interest to Digital Libraries:
 - FRBR
 - RDF Schema

RDF – Resource Description Framework



- Resource Description Framework (RDF) is a way to represent information about *resources* in the Web (in the World)
- A resource is anything that has identity. For example, a resource may be an electronic document, an image, a service (e.g., "today's weather report for Los Angeles"), and a collection of other resources. Not all resources are network "retrievable"; e.g., human beings, corporations, and bound books in a library can also be considered resources
- All resources are identified by a URI (Uniform Resource Identifier)
 - a string of characters that unambiguously identifies a particular resource
- *Resources* are described in terms of simple statements specifying properties and property values of resources

Resources and metadata



Metadata can be associated with any resource: physical, digital, abstract resource, etc.

- HTML documents
- digital images
- databases
- books
- museum objects
- archival records
- metadata records
- Web sites
- collections
- services
- physical places
- people
- institutions
- abstract “works”
- concepts
- events



RDF – Resource Description Framework

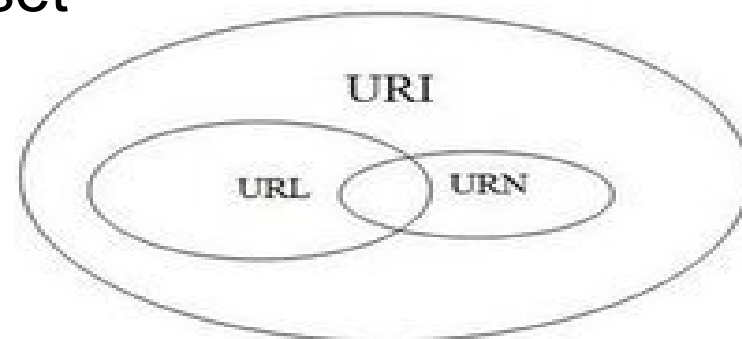


- Resource Description Framework (RDF) is a way to represent information about *resources* in the Web
- A resource is anything that has identity. For example, a resource may be an electronic document, an image, a service (e.g., "today's weather report for Los Angeles"), and a collection of other resources. Not all resources are network "retrievable"; e.g., human beings, corporations, and bound books in a library can also be considered resources
- **All resources are identified by a URI (Uniform Resource Identifier)**
 - a string of characters that unambiguously identifies a particular resource
- *Resources* are described in terms of simple statements specifying properties and property values of resources

URI (IRI), URL, URN



- URL - Uniform Resource Locator
is a subset of the URIs that include a network location
- URN - Uniform Resource Name
is a subset of URIs that include a name within a given space, but no location
- URI - Uniform Resource Identifier
identifies a resource (text document, image file, any other)
- IRI – International Resource Identifier
is a URI with Unicode character set



URLs



- URL (Uniform Resource Locator) is a reference (an address) to a resource on the Internet; it is a specific type of URI; it has two parts:
 - protocol identifier (e.g. http, ftp)
 - resource name (its structure depends on the protocol)
 - very common for the protocol http is “host name” followed by “file name”:
`http://www.example.com/index.html`
- URLs were the ones used at the beginning of the Web and are the actual links in Web pages

Several types of URIs

URI Syntax



<scheme name> : <hierarchical part> [? <query>] [# <fragment>]

any://example.com:8042/over/there?name=ferret#nose

_ / _ / _ / _ / _ /
scheme authority path query fragment

- ftp://ftp.is.co.za/rfc/rfc1808.txt
- <http://www.ietf.org/rfc/rfc2396.txt>
- ldap://[2001:db8::7]/c=GB?objectClass?one
- mailto:John.Doe@example.com
- news:comp.infosystems.www.servers.unix
- tel:+1-816-555-1212
- telnet://192.0.2.16:80/
- urn:oasis:names:specification:docbook:dtd:xml:4.1.2bb



RDF – Resource Description Framework



- Resource Description Framework (RDF) is a way to represent information about *resources* in the Web
- A resource is anything that has identity. For example, a resource may be an electronic document, an image, a service (e.g., "today's weather report for Los Angeles"), and a collection of other resources. Not all resources are network "retrievable"; e.g., human beings, corporations, and bound books in a library can also be considered resources
- All resources are identified by a URI (Uniform Resource Identifier)
 - a string of characters that unambiguously identifies a particular resource
- *Resources* are described in terms of simple statements specifying properties (attributes) and property values of resources

RDF statements



- Resources:
 - An object, an entity or anything we want to talk about (e.g. authors, books, publishers, places, people, facilities)
- Properties:
 - They codify **relations** (e.g. written-by, friend-of, located-in, ...) and **attributes** (e.g. age, date of birth, length ...)
- Statements:
 - Statements assert the properties of resources in form of **triples** subject-property-value (**subject-predicate-object**)
- Every resource and property has a URI (an URL or any other identifier)
- Values can be resources (for relations) or literals (for attributes)

Simple RDF statement



- A statements is composed of three parts: a subject, a predicate (about the subject), an object (the value of the predicate)
- Example
 - <http://www.example.org/index.html> has a creator whose value is John Smith
- the subject is the resource identified by this URI:
`http://www.example.org/index.html`
- the predicate is the phrase “has a creator”
- the object is the phrase "John Smith“
- To avoid “misunderstandings”, the three components of this statement should be indicated by URIs
 - Subject `http://www.example.org/index.html`
 - Predicate `http://purl.org/dc/elements/1.1/creator`
 - Object `http://www.example.org/staffid/85740`



Additional RDF statements (in natural language)

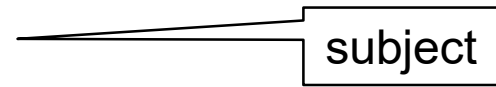
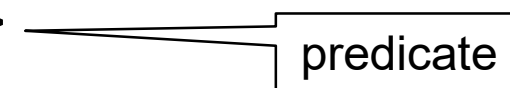
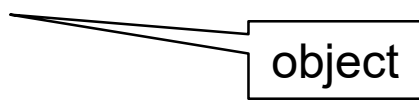


- <http://www.example.org/index.html>
has a creator
whose value is John Smith
- <http://www.example.org/index.html>
has a creation-date
whose value is August 16, 1999
- <http://www.example.org/index.html>
has a language
whose value is English

RDF statements as triples



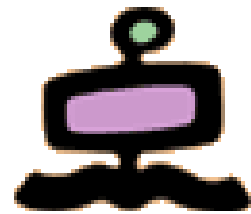
Each statement corresponds to a “triple”

- `<http://www.example.org/index.html>`  subject
- `<http://purl.org/dc/elements/1.1/creator>`  predicate
- `<http://www.example.org/staffid/85740>`  object

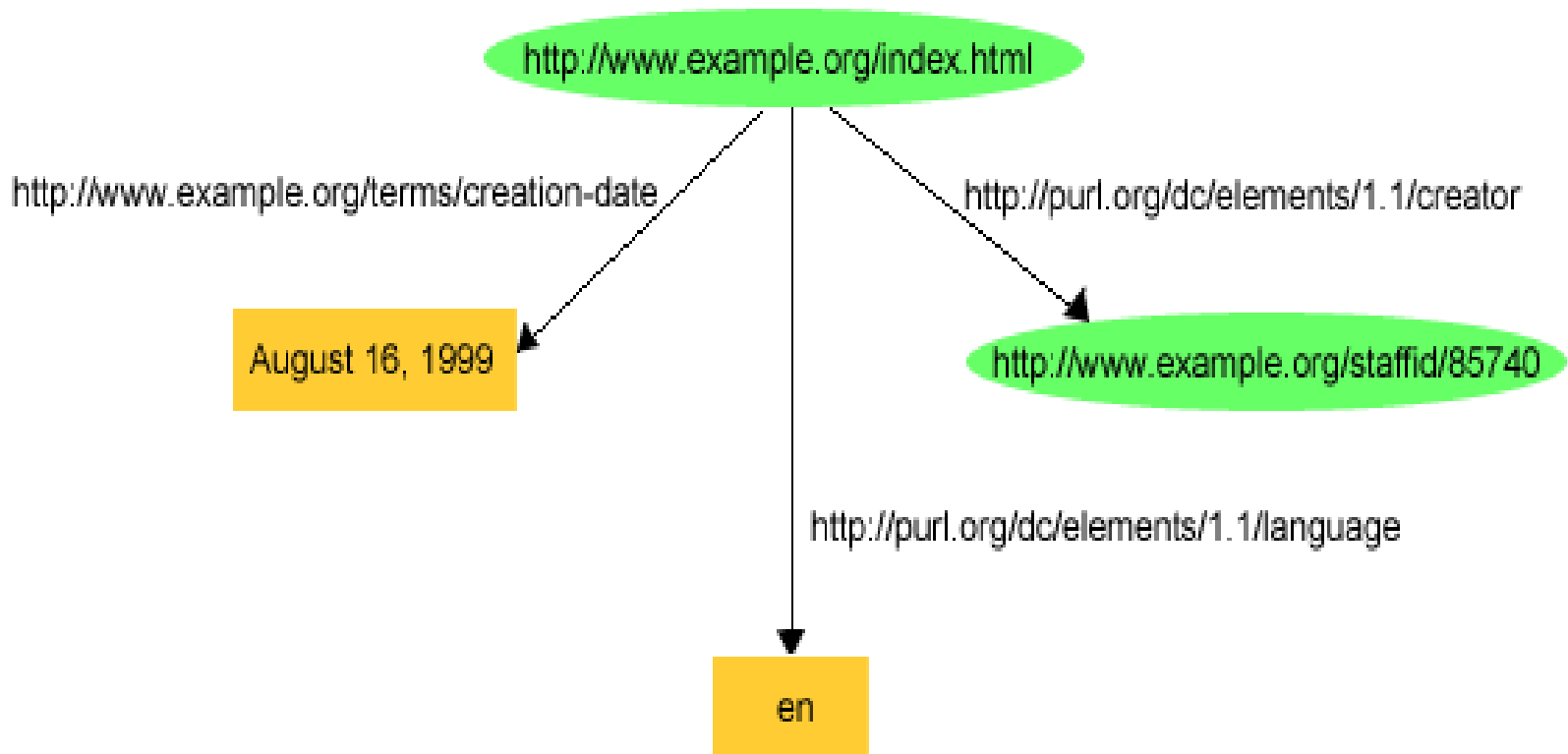
- `<http://www.example.org/index.html>`
- `<http://www.example.org/terms/creation-date>`
- "August 16, 1999" .

- `<http://www.example.org/index.html>`
- `<http://purl.org/dc/elements/1.1/language>`
- "en" .

RDF statements are graphs



Each triple corresponds to an arc in a graph



Another example of RDF triples



<Bob> <is a> <person>.

<Bob> <is a friend of> <Alice>.

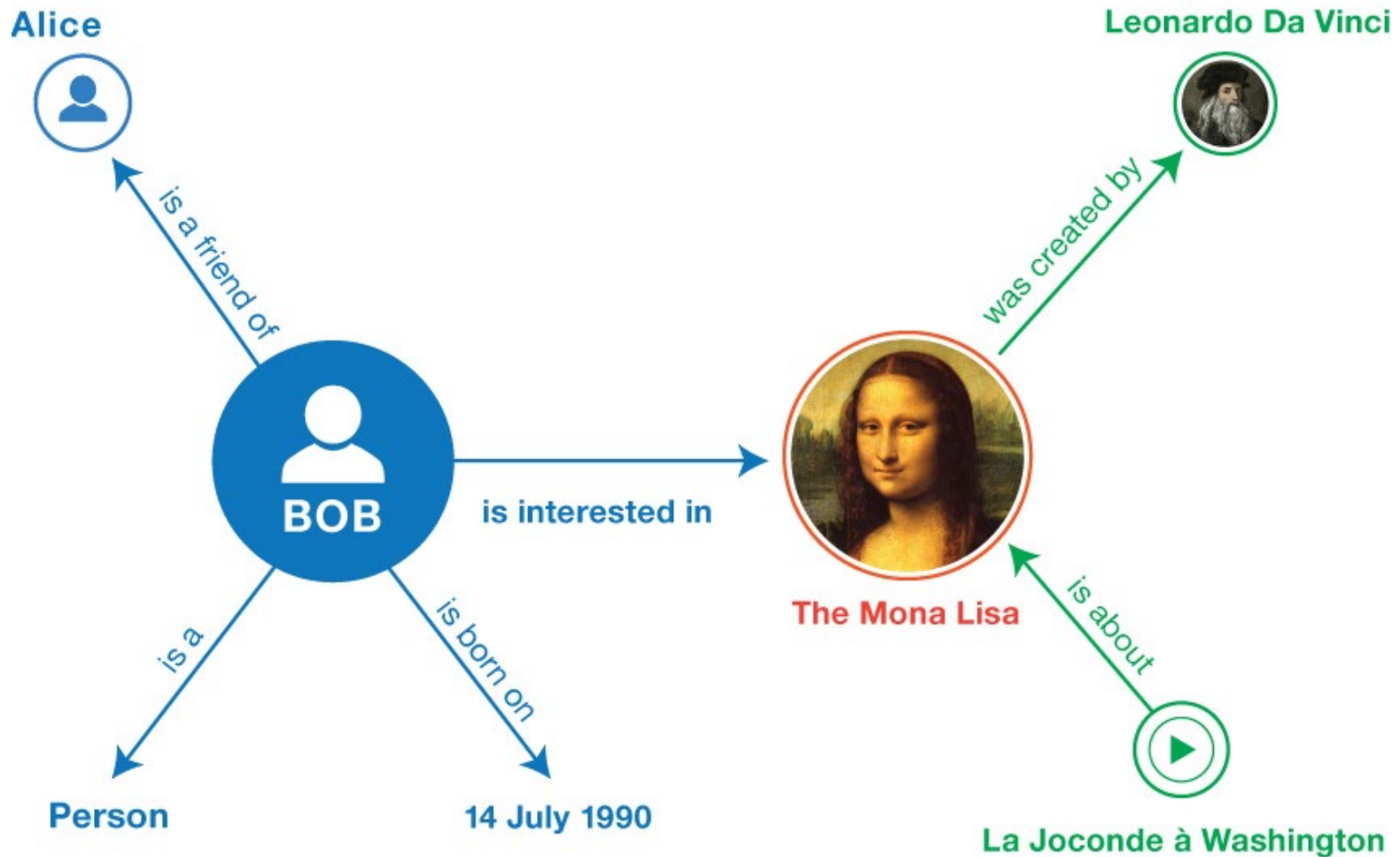
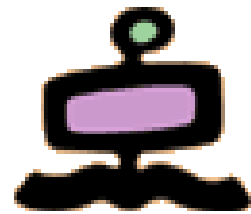
<Bob> <is born on> <the 4th of July 1990>.

<Bob> <is interested in> <the Mona Lisa>.

<the Mona Lisa> <was created by> <Leonardo da Vinci>.

**<the video 'La Joconde à Washington'> <is about>
<the Mona Lisa>**

An RDF graph



Abbreviation of URIs



URIs are usually long character strings. It is convenient to abbreviate them by defining **prefixes** of namespaces (the same idea as in XML)

- prefix `rdf:`, namespace URI:
`http://www.w3.org/1999/02/22-rdf-syntax-ns#`
- prefix `rdfs:`, namespace URI:
`http://www.w3.org/2000/01/rdf-schema#`
- prefix `dc:`, namespace URI:
`http://purl.org/dc/elements/1.1/`
- prefix `xsd:`, namespace URI:
`http://www.w3.org/2001/XMLSchema#`
- prefix `ex:`, namespace URI:
`http://www.example.org/` (or `http://www.example.com/`)
- prefix `exterms:`, namespace URI:
`http://www.example.org/terms/`
- prefix `exstaff:`, namespace URI:
`http://www.example.org/staffid/`

Target name space



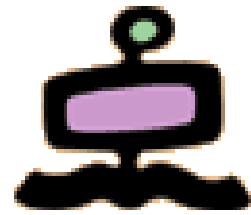
```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
            xs:targetNamespace="http://www.example.com/shiporderschema">
<xs:element name="shiporder">
  <xs:complexType>
    <xs:sequence>
      .....
    </xs:sequence>
    <xs:attribute name="orderid" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
</xs:schema>
```

An XML schema

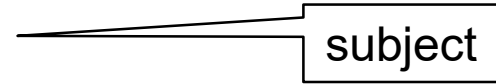
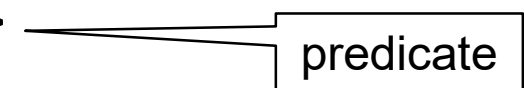
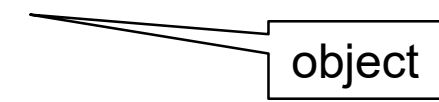


```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="shiporder">
  <xs:complexType>
    <xs:sequence>
      .....
      <xs:element name="item" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="title" type="xs:string"/>
            <xs:element name="note" type="xs:string" minOccurs="0"/>
            <xs:element name="quantity" type="xs:positiveInteger"/>
            <xs:element name="price" type="xs:decimal"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="orderid" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
</xs:schema>
```

RDF statements as triples



Each statement corresponds to a “triple”

- `<http://www.example.org/index.html>`  subject
- `<http://purl.org/dc/elements/1.1/creator>`  predicate
- `<http://www.example.org/staffid/85740>`  object

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

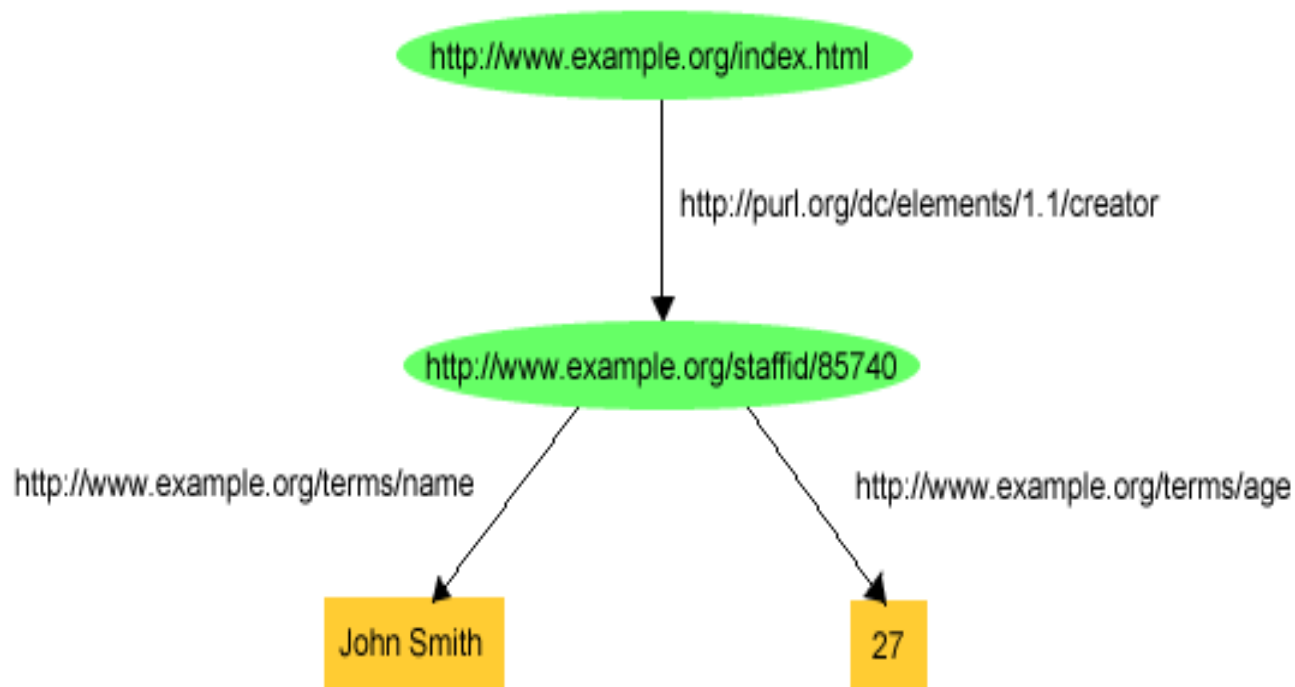
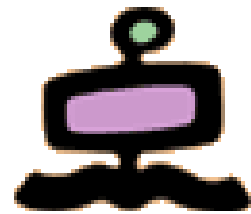
- `<http://www.example.org/index.html>`
- `<http://purl.org/dc/elements/1.1/language>`
- "en" .

Abbreviated triples

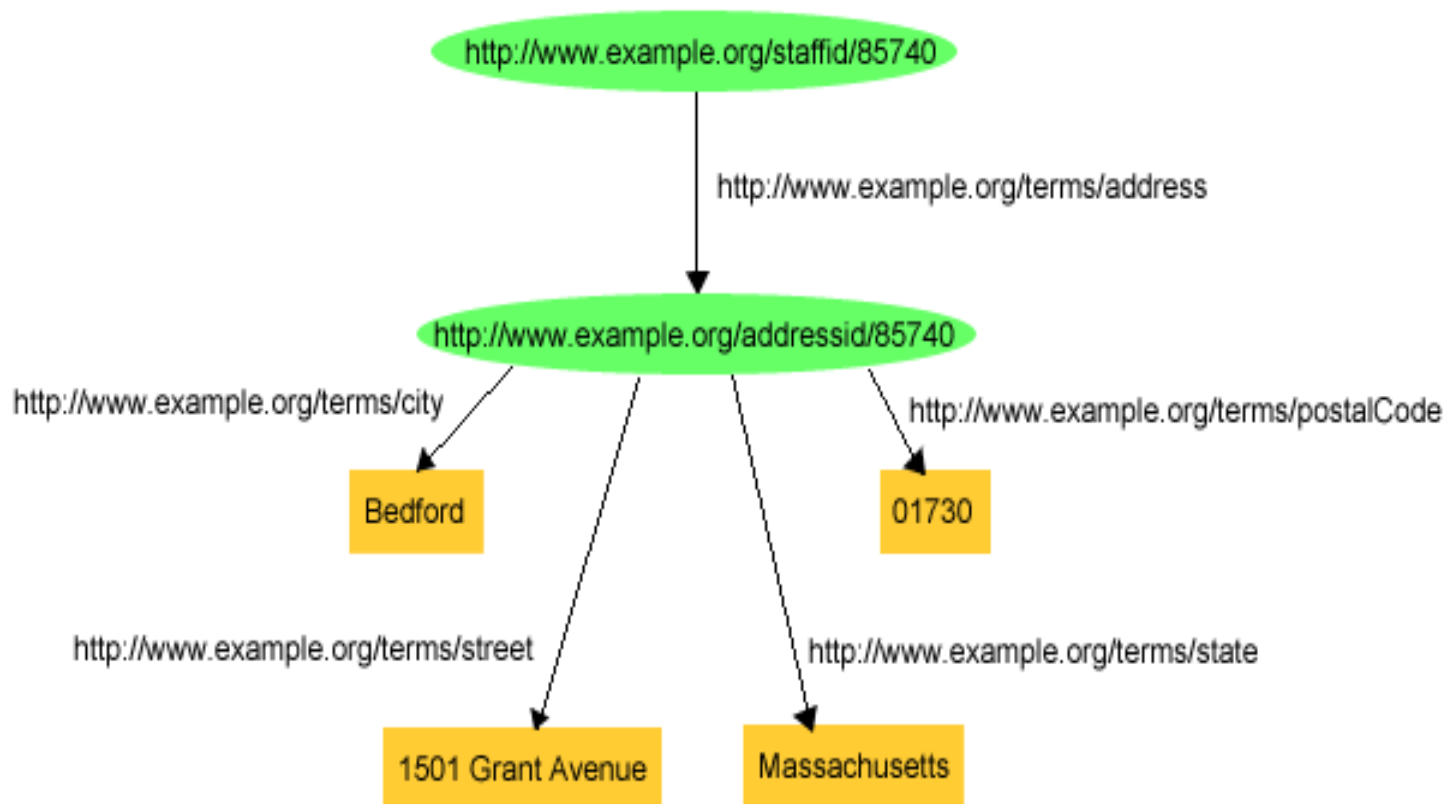
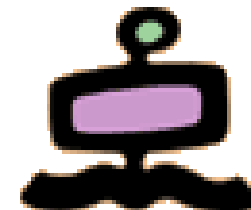


- `ex:index.html`
`dc:creator`
`exstaff:85740` .
- `ex:index.html`
`exterms:creation-date`
`"August 16, 1999"` .
- `ex:index.html`
`dc:language`
`"en"` .

URI values can be subjects ...



... to any depth ...

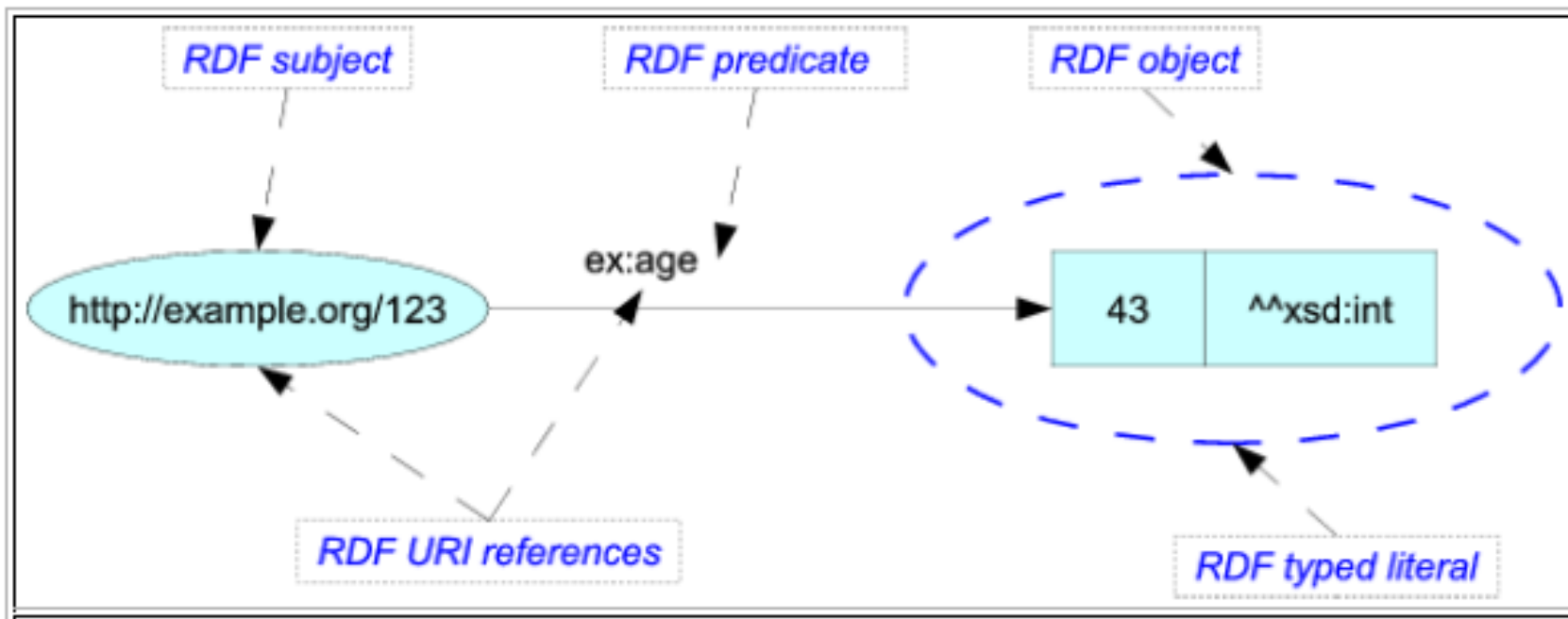
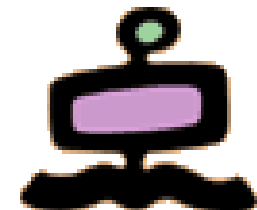


RDF summary

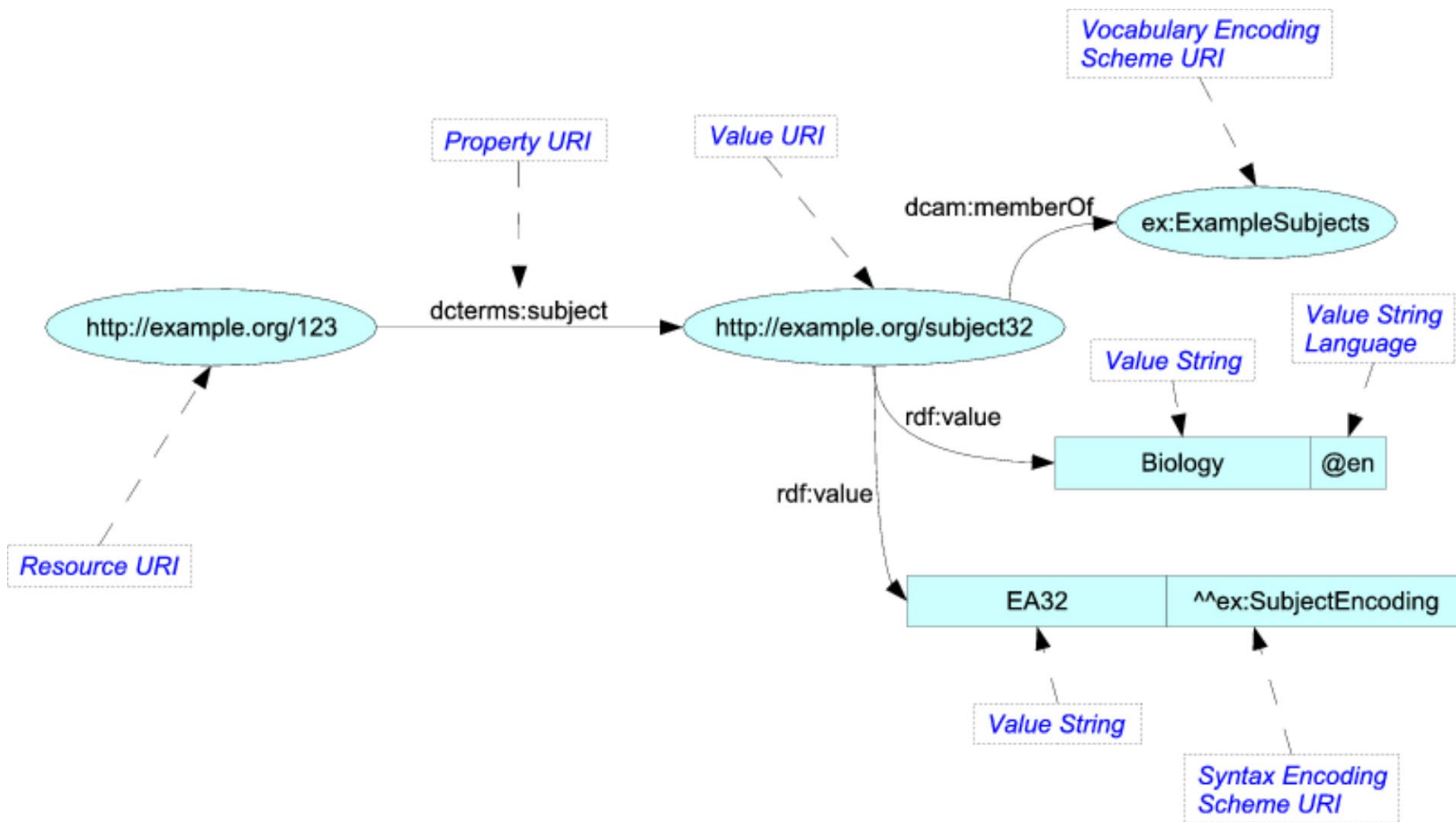
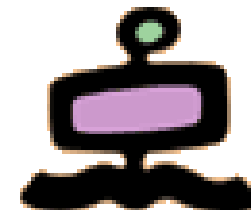


- A resource can be described by a set of RDF triples
- A set of RDF triples can be represented as a graph
- An RDF triple has three components
 - a **subject**, which is an RDF URI reference or a blank RDF node
 - a **predicate**, which is an RDF URI reference
 - an **object**, which is an RDF URI reference, a blank RDF node or an RDF literal
- An RDF literal can be of two kinds
 - an RDF **plain literal** is a character string with an optional associated language tag describing the language of the character string
 - an RDF **typed literal** is a character string with an associated RDF datatype URI. An RDF datatype defines the syntax and semantics of a set of character strings that represent data such as booleans, integers, dates, etc.

RDF example 1



RDF example 2

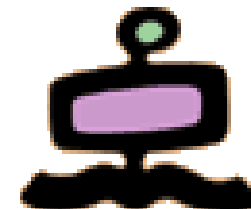


Serialization formats for RDF



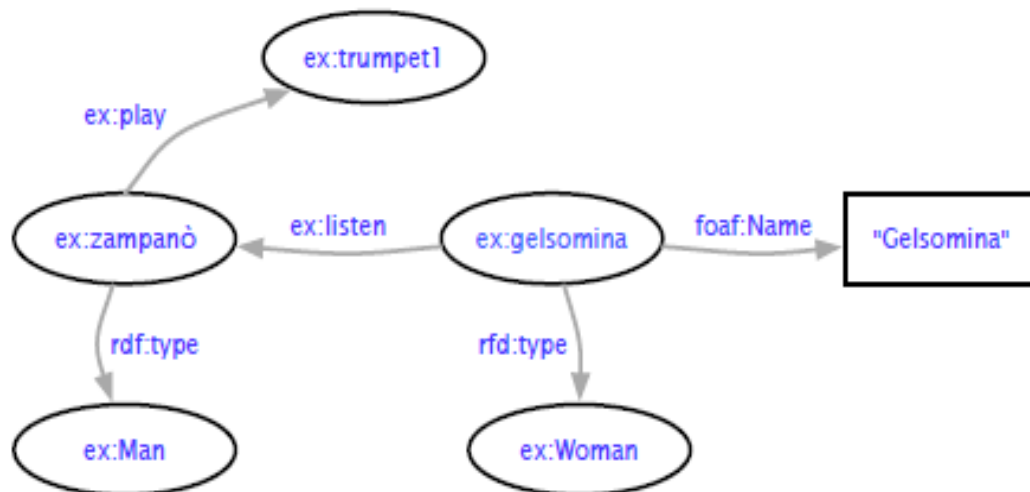
- Turtle and TriG
- JSON-LD (JSON based)
- RDFa (for HTML embedding)
- N-Triples and N-Quads
- RDF/XML

Turtle notation



```
@prefix ex: <http://www.example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf: <http://xmlns.com/foaf/spec/> .
```

```
ex:zampano    rdf:type      ex:Man .
ex:zampano    ex:play      ex:trumpet1 .
ex:gelsomina  rdf:type      ex:Woman .
ex:gelsomina  ex:listen    ex:zampano .
ex:gelsomina  foaf:name    "Gelsomina" .
```



RDFa: RDF info in HTML



```
<body>
  ...
  <h2>Title of Book1</h2>
  <p>Date: 2011-09-10</p>
  ...
</body>
```

```
<body>
  ...
  <div resource="http://example.com/alice/books/Book1">
    <h2 property="http://purl.org/dc/terms/title">
      Title of Book1</h2>
    <p>Date:
      <span property="http://purl.org/dc/terms/created">
        2011-09-10</span>
    </p>
  </div>
  ...
</body>
```


RDFa: browsers only “see” format

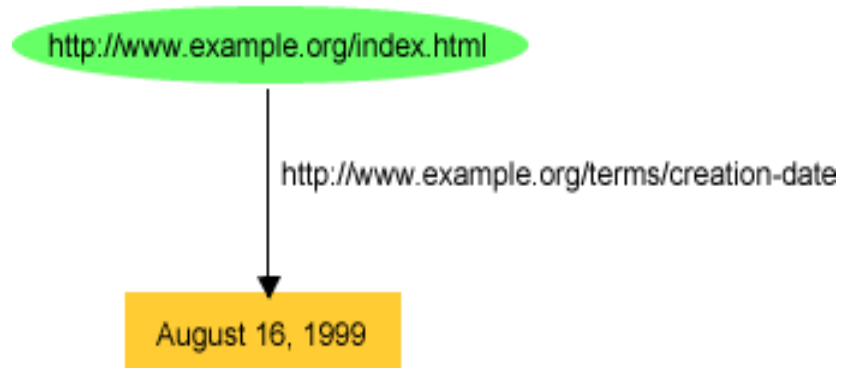


What browsers see



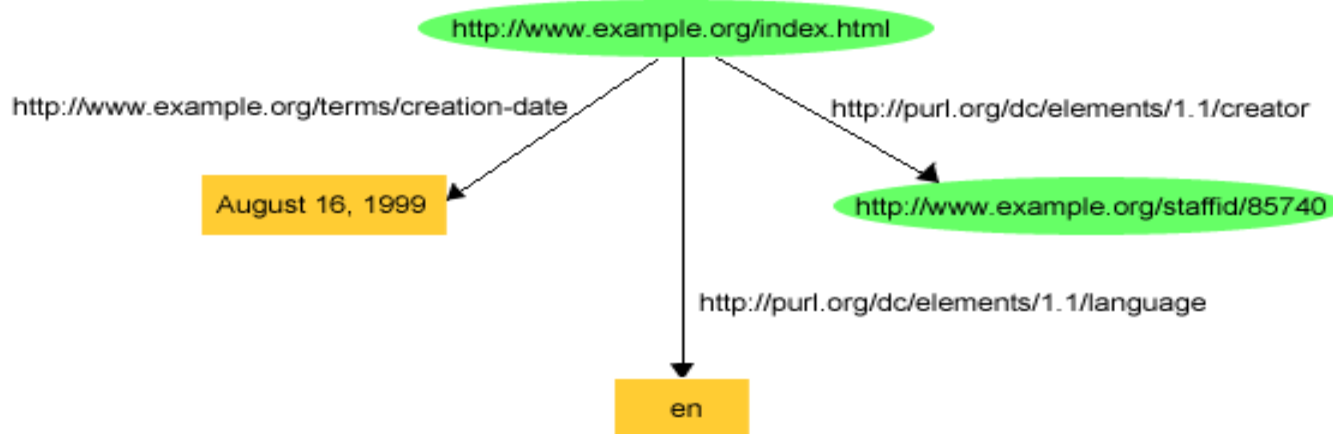
What humans see

RDF graphs represented in XML




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

RDF graph with abbreviated XML



1. `<?xml version="1.0"?>`
2. `<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:exterm="http://www.example.org/terms/">`
3. `<rdf:Description rdf:about="http://www.example.org/index.html">`
4. `<exterm:creation-date>August 16, 1999</exterm:creation-date>`
5. `<dc:language>en</dc:language>`
6. `<dc:creator rdf:resource="http://www.example.org/staffid/85740"/>`
7. `</rdf:Description>`
8. `</rdf:RDF>`



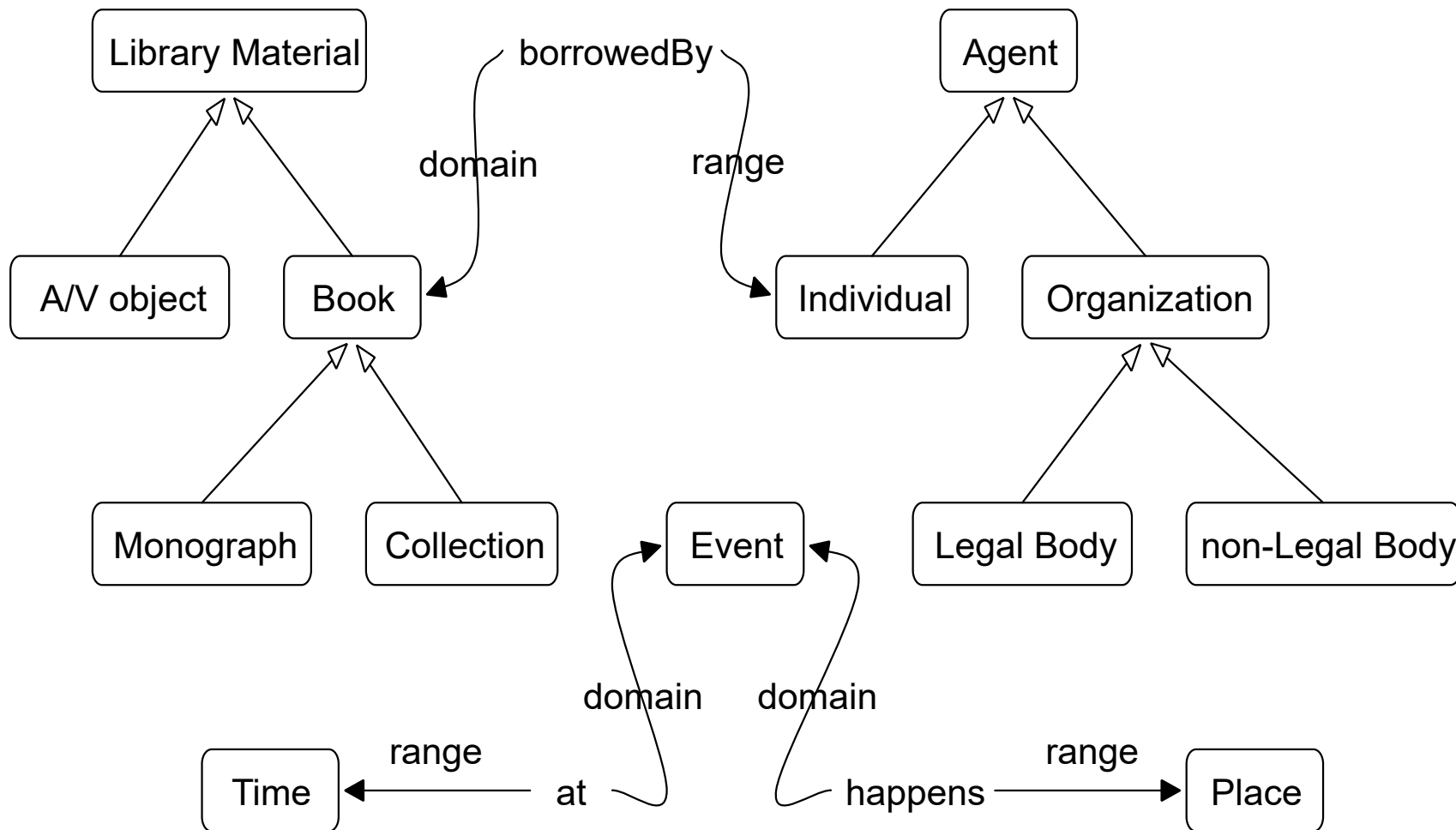
- FRBR: Functional Requirements for Bibliographic Records
 - LRM: Library Reference Model
- RDF: Resource Description Framework
- RDF Schema 
 - LOD: Linked Open Data

RDF Schema

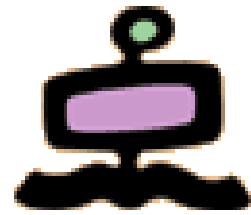


- RDF provides a way to express simple statements about resources, using “named” properties and values
- It is convenient to define the *vocabularies* (terms) that are going to be used in those statements, to indicate that they are describing specific kinds or **classes** of resources, and will use specific **properties** in describing those resources
- For example, to describe bibliographic resources we could define classes such as “Book” or “Journal Article”, and use properties such as “author”, “title”, “borrowedBy” to describe them
- RDF Schema defines the terms used in RDF descriptions by providing a **type system** to be used in the RDF descriptions
- In other words, it provides a way to represent a “conceptual model” of a (small) part of the world, by defining the main “concepts” (classes) in this part of the world, their properties and their relationships

Example of Classes and SubClasses



Basics of RDF Schema



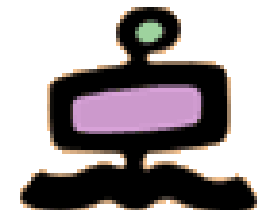
- In RDF Schema we have a way to express:
 - that something is a **class** or a **property**
 - that a class is a sub-class of another class
 - that a property is a sub-property of another property
 - that a class is the **domain** of a property
 - that a class is the **range** of a property

Main notions of RDF Schema



- The main notions of the RDF Schema are:
 - Classes, which can be organized in sub-classes, to any level (defining a taxonomy)
 - Properties, which also can be organized in sub-classes, to any level (defining another taxonomy)
- Vocabulary descriptions (schemas) written in the RDF Schema language are valid RDF graphs
- There is a close analogy with XML documents and XML schemas

Classes and subclasses



prefix ex:, <http://www.example.org/schemas/vehicles#>

Example of MotorVehicle



This is the "schema" (conceptual model)

<code>ex:MotorVehicle</code>	<code>rdf:type</code>	<code>rdfs:Class</code>
<code>ex:PassengerVehicle</code>	<code>rdf:type</code>	<code>rdfs:Class</code>
<code>ex:Van</code>	<code>rdf:type</code>	<code>rdfs:Class</code>
<code>ex:Truck</code>	<code>rdf:type</code>	<code>rdfs:Class</code>
<code>ex:MiniVan</code>	<code>rdf:type</code>	<code>rdfs:Class</code>
<code>ex:PassengerVehicle</code>	<code>rdfs:subClassOf</code>	<code>ex:MotorVehicle</code>
<code>ex:Van</code>	<code>rdfs:subClassOf</code>	<code>ex:MotorVehicle</code>
<code>ex:Truck</code>	<code>rdfs:subClassOf</code>	<code>ex:MotorVehicle</code>
<code>ex:MiniVan</code>	<code>rdfs:subClassOf</code>	<code>ex:Van</code>
<code>ex:MiniVan</code>	<code>rdfs:subClassOf</code>	<code>ex:PassengerVehicle</code>

This is an instance of the schema in the "real world"

Instance of a motor vehicle

<code>exthings:johnSmithsCar</code>	<code>rdf:type</code>	<code>ex:PassengerVehicle</code>
-------------------------------------	-----------------------	----------------------------------

Examples of properties



This is the "schema" (conceptual model)

<code>ex:registeredTo</code>	<code>rdf:type</code>	<code>rdfs:Property</code>
<code>ex:rearSeatLegRoom</code>	<code>rdf:type</code>	<code>rdfs:Property</code>
<code>ex:weight</code>	<code>rdf:type</code>	<code>rdfs:Property</code>
<code>ex:primaryDriver</code>	<code>rdf:type</code>	<code>rdfs:Property</code>

These are instances of the schema in the "real world"

<code>exthings:johnSmithsCar</code>	<code>rdf:type</code>	<code>ex:PassengerVehicle</code>
<code>exthings:johnSmithsCar</code>	<code>ex:registeredTo</code>	<code>exstaff:85740</code>
<code>exthings:johnSmithsCar</code>	<code>ex:rearSeatLegRoom</code>	<code>"127"^^xsd:integer</code>
<code>exthings:johnSmithsCar</code>	<code>ex:weight</code>	<code>"2500"^^xsd:integer</code>
<code>exthings:johnSmithsCar</code>	<code>ex:primaryDriver</code>	<code>exstaff:85740</code>

Domain and Range



- In the RDF schema properties are defined in order to describe the characteristics of the classes
- Properties can be arranged in a taxonomy by defining subProperties
- An important aspect of the RDF Schema is the possibility to define the **domain** and the **range** of a property
 - Domain is the class of entities that can be the subject of an RDF statement where the property is used as the predicate
 - Range is the class of entities that can be the object of an RDF statement where the property is used as the predicate

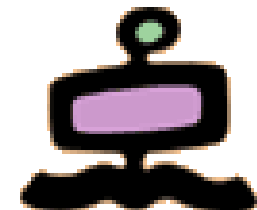
Examples of Domain and Range



<code>ex:Person</code>	<code>rdf:type</code>	<code>rdfs:Class</code>
<code>ex:Book</code>	<code>rdf:type</code>	<code>rdfs:Class</code>
<code>ex:hasAuthor</code>	<code>rdf:type</code>	<code>rdfs:Property</code>
<code>ex:hasAuthor</code>	<code>rdfs:domain</code>	<code>rdfs:Book</code>
<code>ex:hasAuthor</code>	<code>rdfs:range</code>	<code>rdfs:Person</code>
<code>ex:isAuthorOf</code>	<code>rdf:type</code>	<code>rdfs:Property</code>
<code>ex:isAuthorOf</code>	<code>rdfs:domain</code>	<code>rdfs:Person</code>
<code>ex:isAuthorOf</code>	<code>rdfs:range</code>	<code>rdfs:Book</code>

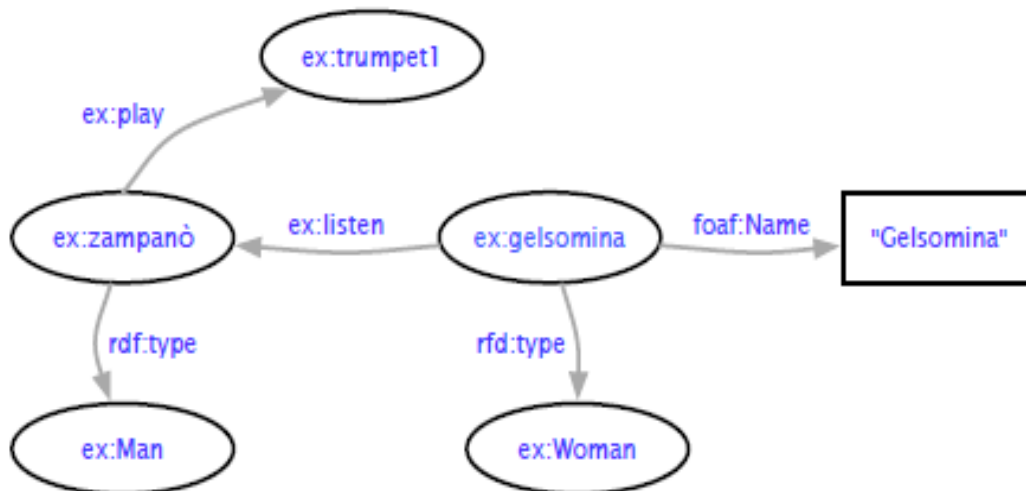
With these definitions of domain and range we are saying that in our (simple) “model of the world” books can only be written by a person, and a person can only write books (unless there are other triples in the schema defining other objects that can be written by a Person)

Turtle notation



```
@prefix ex: <http://www.example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf: <http://xmlns.com/foaf/spec/> .
```

```
ex:zampano    rdf:type      ex:Man .
ex:zampano    ex:play      ex:trumpet1 .
ex:gelsomina  rdf:type      ex:Woman .
ex:gelsomina  ex:listen    ex:zampano .
ex:gelsomina  foaf:name    "Gelsomina" .
```



Inference of Properties



RDF schema of "La Strada"

`ex:play` `rdf:type` `rdfs:Property`

`ex:play` `rdfs:domain` `ex:Person`

`ex:play` `rdfs:range` `ex:MusicalInstrument`

Instance of Zampanò'

`ex:zampanò` `ex:play` `ex:trumpet`

we can infer

`ex:zampanò` `rdf:type` `ex:Person`

`ex:trumpet` `rdf:type` `ex:MusicalInstrument`

Inference for answering queries



```
ex:hasMother rdf:type      rdfs:Property
ex:hasMother rdfs:range    ex:Female
ex:hasMother rdfs:range    ex:Person
```

```
exstaff:Frank ex:hasMother  exstaff:Mary
```

Mary (the mother of Frank) must be at the same time a person and a female

It is therefore possible to answer queries like:
"List the names of all the females"
and Mary will be in the list, without having ever provided the information that Mary is a female

Summarizing the RDF Schema



- An RDF Schema is a simple “meta” vocabulary used to describe **ontologies**
 - Class, subclassOf, type
 - e.g., Person, Team
 - Property, subPropertyOf
 - e.g., playsFor
 - Domain (the **class for the subjects** of a particular *property*)
 - **Person** *playsFor* Team
 - Range (the **class for the values** of a particular *property*)
 - Person *playsFor* **Team**