RDF and RDF Schema

Raúl García-Castro, Óscar Corcho

Ontology Engineering Group
Universidad Politécnica de Madrid, Spain

Speaker: Raúl García Castro
rgarcia@fi.upm.es

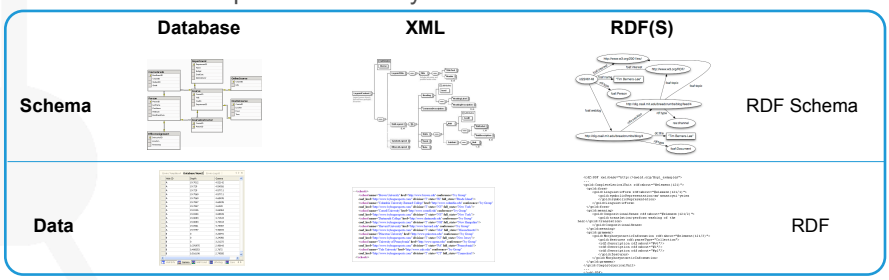
Curso Biblioteca Nacional
Madrid, Spain
21-25th November 2011

Index

- **Overview**
- RDF
  - Introduction
  - RDF components
  - Serializing RDF
  - Hands-on
- RDF Schema
  - Introduction
  - Main RDF-S components
  - Serializing RDF-S
  - Hands-on
  - RDF(S) Limitations
  - Hands-on
- Reasoning
  - Reasoning with RDF
  - Reasoning with RDF-S
- RDF(S) management APIs

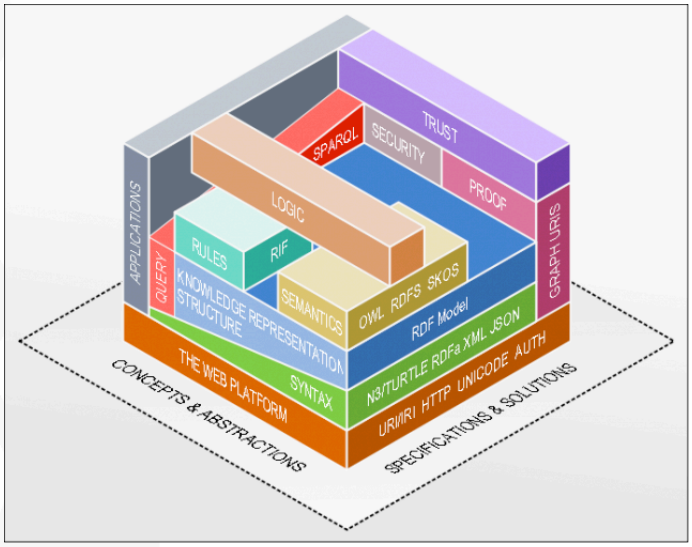
RDF and RDF Schema

- RDF: Resource Description Framework
- Goal
  - To describe the semantics of information in a machine- processable way



- W3C recommendations
  - Model
  - Syntax
  - Semantics

RDF(S) in the Semantic Web



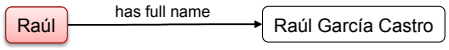
Index

- Overview
- **RDF**
  - **Introduction**
  - RDF components
  - Serializing RDF
  - Hands-on
- RDF Schema
  - Introduction
  - Main RDF-S components
  - Serializing RDF-S
  - Hands-on
  - RDF(S) Limitations
  - Hands-on
- Reasoning
  - Reasoning with RDF
  - Reasoning with RDF-S
- RDF(S) management APIs

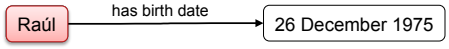
RDF statements

- Also known as "triples"
  - [Subject, Predicate, Object]
- "Raúl is a member of the Ontology Engineering Group"
  - [Raúl, is member of, Ontology Engineering Group]

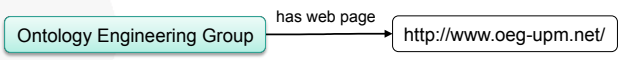
- "Raúl's full name is Raúl García Castro"
  - [Raúl, has full name, Raúl García Castro]



- "Raúl was born on 26 December 1975"
  - [Raúl, was born, 26 December 1975]

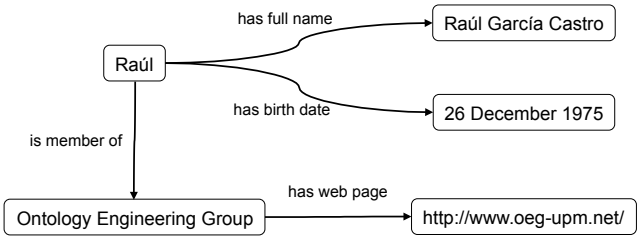


- "The Ontology Engineering Group web page is http://www.oeg-upm.net/"
  - [Ontology Engineering Group, has web page, http://www.oeg-upm.net/]



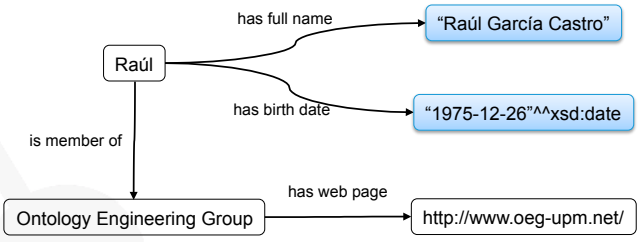
RDF graphs

- RDF graphs are sets of triples



RDF literals

- Triple objects can be literals (character strings)
  - Subject and predicates are always resources
- Literals can be typed
  - Usually using XML Schema datatypes
  - RDF provides the ***rdf:XMLLiteral*** datatype

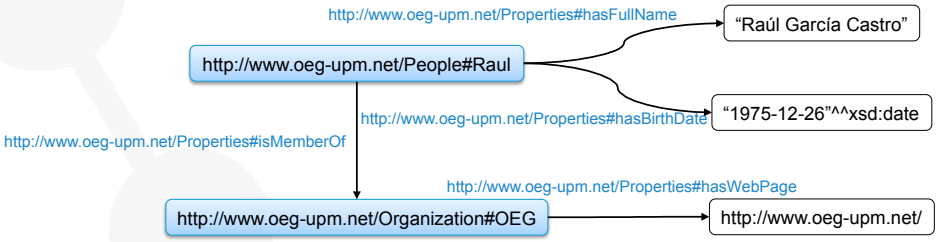


URIs in RDF

- URI component parts (RFC3986)
  - http://www.oeg-upm.net:8080/Info/People?position=current#Raul

Scheme | Authority | Path | Query | Fragment

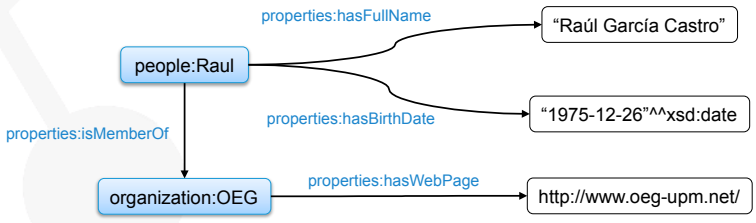
- RDF URIs:
  - Are URI references: URI + Fragment
  - Can contain Unicode characters
  - Identify resources and values (e.g., mailto:rgarcia@fi.upm.es)



Namespaces in RDF

- Namespaces defined using XML qualified names
- URIs under a namespace are called vocabularies

Prefix	URI
people	http://www.oeg-upm.net/People#
organization	http://www.oeg-upm.net/Organization#
properties	http://www.oeg-upm.net/Properties#
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	http://www.w3.org/2000/01/rdf-schema#
xsd	http://www.w3.org/2001/XMLSchema#

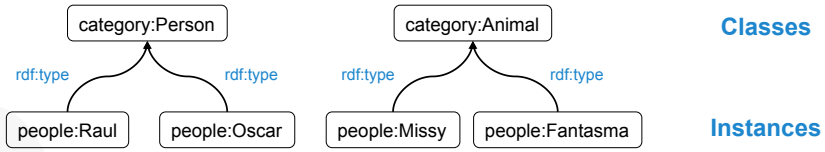


Index

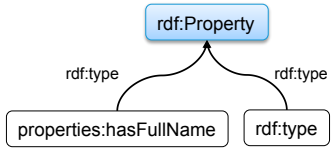
- Overview
- **RDF**
  - Introduction
  - **RDF components**
  - Serializing RDF
  - Hands-on
- RDF Schema
  - Introduction
  - Main RDF-S components
  - Serializing RDF-S
  - Hands-on
  - RDF(S) Limitations
  - Hands-on
- Reasoning
  - Reasoning with RDF
  - Reasoning with RDF-S
- RDF(S) management APIs

Classifying resources

- The ***rdf:type*** property is used to classify resources in categories/classes

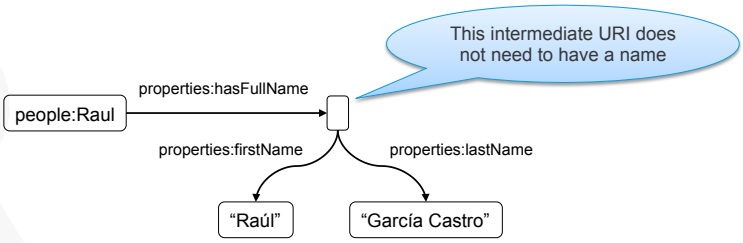


- The ***rdf:Property*** class is the class of all properties



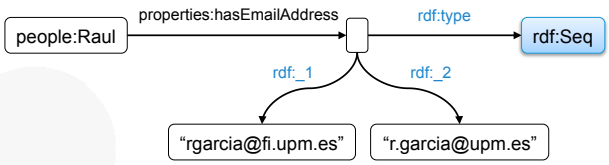
Blank nodes: structured property values

- Most real-world data involves structures that are more complicated than sets of RDF triple statements



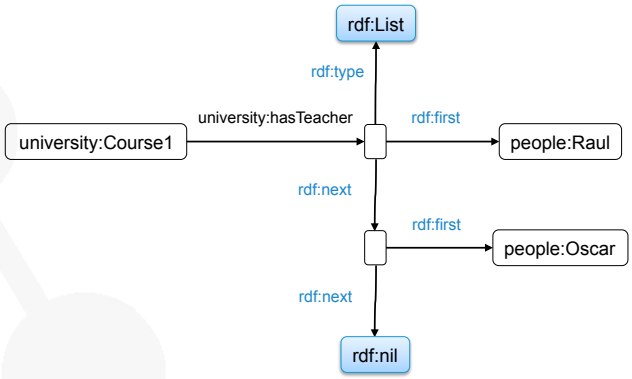
RDF Containers

- Describe groups of things
  - A book was created by several authors
  - A lesson is taught by several persons
  - etc.
- RDF provides a container vocabulary
  - ***rdf:Bag.*** Group of resources or literals, including duplicates, where order is not significant
  - ***rdf:Seq.*** Group of resources or literals, including duplicates, where order is significant
  - ***rdf:Alt.*** Group of resources or literals that are alternatives (typically for a single value of a property)



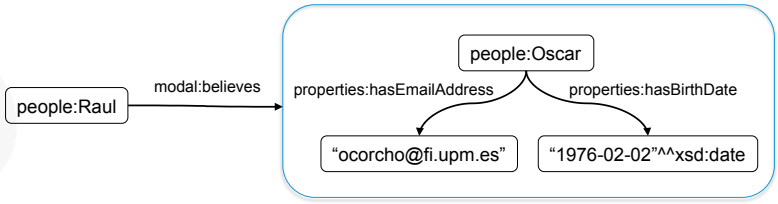
RDF Collections

- Groups of things represented as a list structure
  - "A container with limits"
- Constructed using ***rdf:List***, ***rdf:first***, ***rdf:rest***, and ***rdf:nil***



RDF Reification

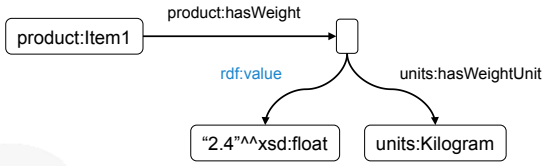
- RDF statements about other RDF statements
  - "Raúl believes that Oscar's birthdate is on Feb 2nd, 1976 and that his e-mail address is ocorcho@fi.upm.es"
- Expressed using ***rdf:Statement***, ***rdf:subject***, ***rdf:predicate***, and ***rdf:object***



- RDF Reification
  - Allows expressing beliefs (and other modalities)
  - Allows expressing trust models, digital signatures, etc.
  - Allows expressing metadata about metadata

Main value of a structured value

- Sometimes one of the values of a structured value is the main one
  - The weight of an item is 2.4 kilograms
  - The main value is 2.4, which is expressed with ***rdf:value***
- Scarcely used



RDF vocabulary summary

Classes	Properties	Individuals
Classification		
rdf:Property	*rdf:type*	
Containers		
rdf:Bag	*rdf:_1, rdf:_2, rdf:_3…*	
rdf:Seq		
rdf:Alt		
Collections		
rdf:List	*rdf:first*	*rdf:nil*
	rdf:rest	
Reification		
rdf:Statement	*rdf:subject*	
	rdf:predicate	
	rdf:object	
Values		
rdf:XMLLiteral	*rdf:value*	

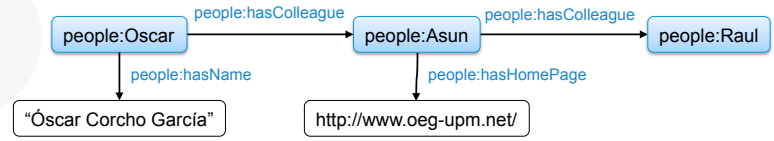
Index

- Overview
- **RDF**
  - Introduction
  - RDF components
  - **Serializing RDF**
  - Hands-on
- RDF Schema
  - Introduction
  - Main RDF-S components
  - Serializing RDF-S
  - Hands-on
  - RDF(S) Limitations
  - Hands-on
- Reasoning
  - Reasoning with RDF
  - Reasoning with RDF-S
- RDF(S) management APIs

RDF Serialisations

- Normative
  - **RDF/XML** (www.w3.org/TR/rdf-syntax-grammar/)
- Working Draft (9 August 2011)
  - **Turtle** (http://www.w3.org/TR/turtle/)
- Alternative (for human consumption)
  - **N3** (http://www.w3.org/DesignIssues/Notation3.html)
  - **TriX** (http://www.w3.org/2004/03/trix/)
  - …
- **Important:** the RDF serializations allow different syntactic variants.
  - E.g., the order of RDF statements has no meaning

RDF Serialisations. RDF/XML




```
<?xml version="1.0"?>
    <rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:person="http://www.ontologies.org/ontologies/people#"
        xmlns="http://www.oeg-upm.net/ontologies/people#"
        xml:base="http://www.oeg-upm.net/ontologies/people">

      <rdf:Property rdf:about="http://www.ontologies.org/ontologies/people#hasHomePage"/>
      <rdf:Property rdf:about="http://www.ontologies.org/ontologies/people#hasColleague"/>
      <rdf:Property rdf:about="http://www.ontologies.org/ontologies/people#hasName"/>

      <rdf:Description rdf:about="#Raul"/>
      <rdf:Description rdf:about="#Asun">
        <person:hasColleague rdf:resource="#Raul"/>
        <person:hasHomePage>http://www.fi.upm.es</person:hasHomePage>
      </rdf:Description>
      <rdf:Description rdf:about="#Oscar">
        <person:hasColleague rdf:resource="#Asun"/>
        <person:hasName>Oscar Corcho García</person:hasName>
      </rdf:Description>

    </rdf:RDF>
```

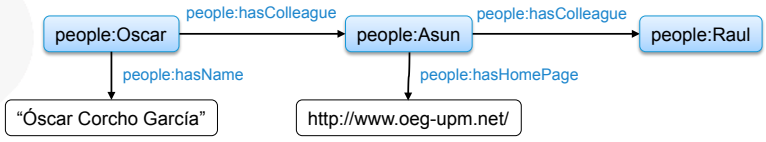

Turtle. URIs, blank nodes, literals

- URIs
  Enclosed in <>
  `<URI>`
  or
  `@prefix prefix <http://....>`
  `prefix:name`
- Blank Nodes
  `:name`
  or
  `[]` for a Blank Node used once
- Literals
  `"Literal"`
  `"Literal"@language`
  `"""Long literal with newlines"""`
- Datatyped Literals
  "lexical form"^^datatype URI
  `"10"^^xsd:integer`
  `"2006-09-04"^^xsd:date`

Turtle. Triples and abbreviations

- Triples separated by .
  `:a :b :c . :d :e :f .`
- Common triple predicate and subject:
  `:a :b :c, :d .`
  which is the same as `:a :b :c . :a :b :d .`
- Common triple subject:
  `:a :b :c; :d :e .`
  which is the same as: `:a :b :c . :a :d :e .`
- Blank node as a subject
  `:a :b [ :c :d ] .`
  which is the same as: `:a :b _:x . _:x :c :d .`
  for blank node `_:x`
- RDF collections
  - `:a :b ( :c :d :e :f ) .`
    which is short for many triples

RDF Serialisations. Turtle



@base <http://www.oeg-upm.net/ontologies/people/ >
@prefix vocab: <http://www.ontologies.org/ontologies/people#>
:Asun vocab:hasColleague :Raul ;
vocab:hasHomePage "http://www.fi.upm.es/".
:Oscar vocab:hasColleague :Asun ;
vocab:hasName "Óscar Corcho García".

Index

- Overview
- **RDF**
  - Introduction
  - RDF components
  - Serializing RDF
  - **Hands-on**
- RDF Schema
  - Introduction
  - Main RDF-S components
  - Serializing RDF-S
  - Hands-on
  - RDF(S) Limitations
  - Hands-on
- Reasoning
  - Reasoning with RDF
  - Reasoning with RDF-S
- RDF(S) management APIs

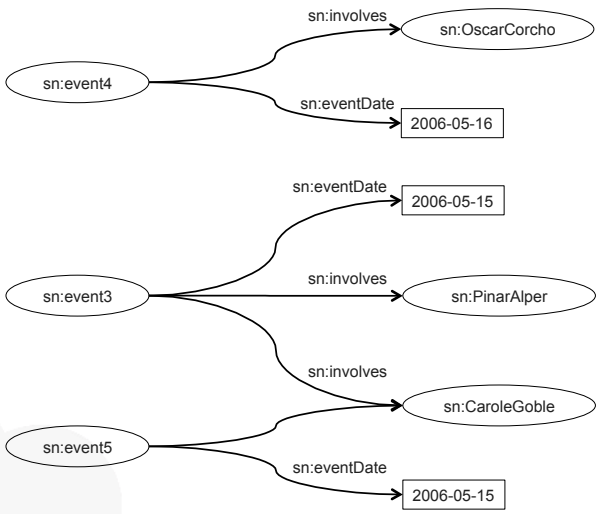
Exercise

- Objective
  - Get used to the different syntaxes of RDF
- Tasks
  - Take the text of an RDF file and create its corresponding graph
  - Take an RDF graph and create its corresponding RDF/XML and Turtle files

Exercise 1.a. Create a graph from a file

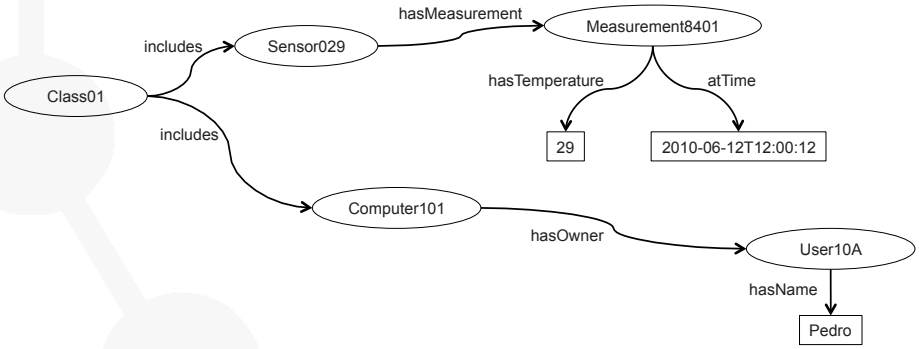
- Open the file StickyNote_PureRDF.rdf
- Create the corresponding graph from it
- Compare your graph with those of your colleagues

Exercise 1.a. StickyNote_PureRDF.rdf



Exercise 1.b. Create files from a graph

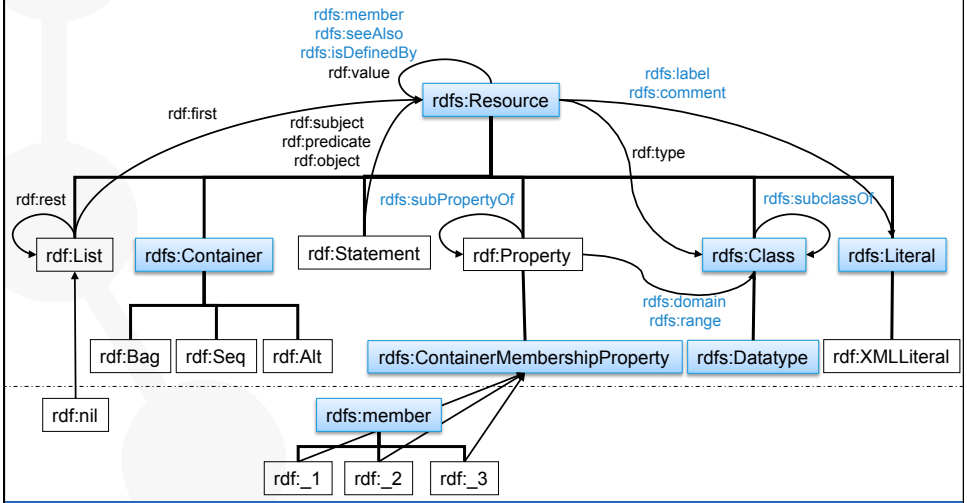
- Transform the following graph into the RDF/XML and Turtle syntaxes



Index

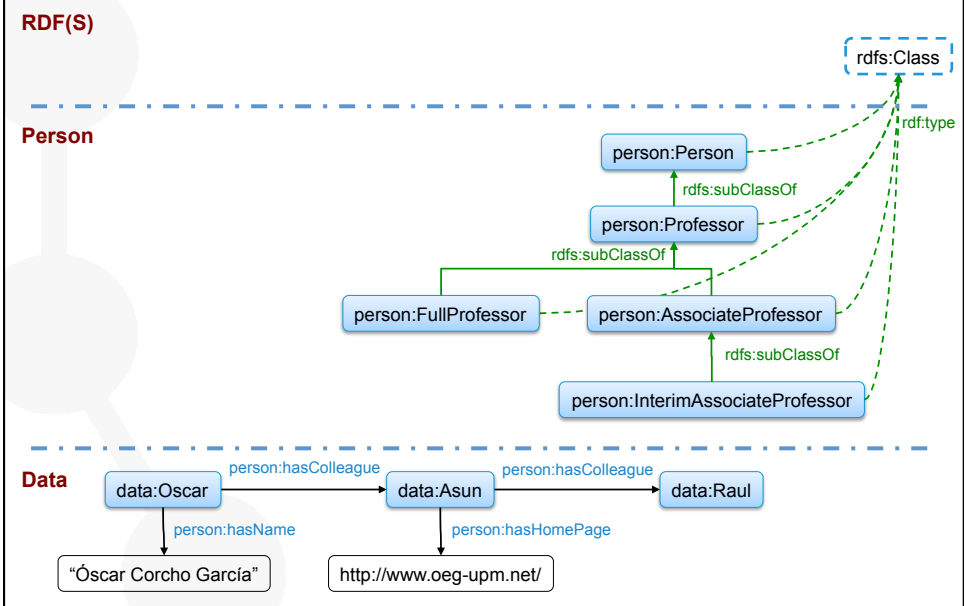
RDF Schema

- Extends RDF
- Allows describing classes of resources and their properties
- Adds constraints on models

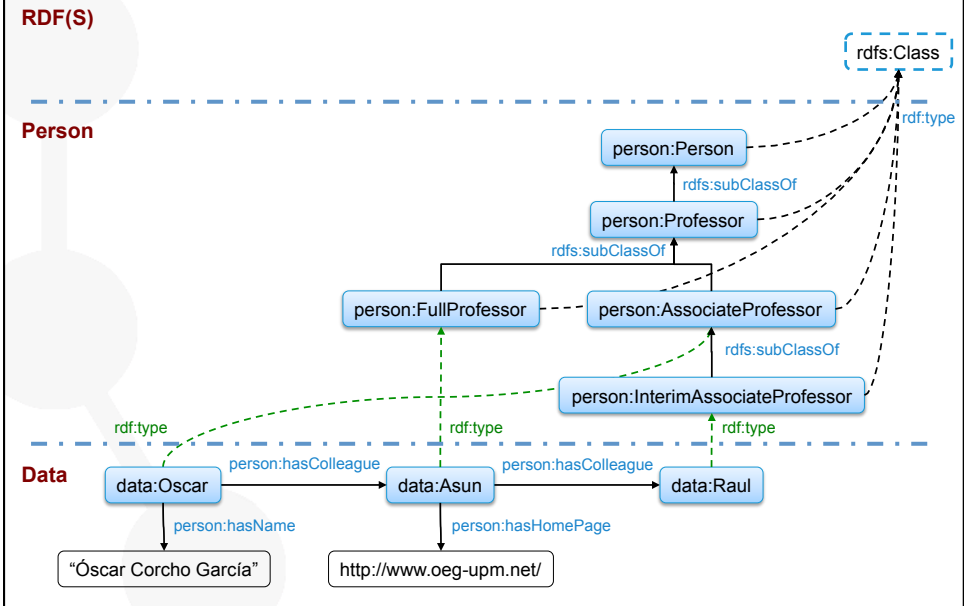


Index

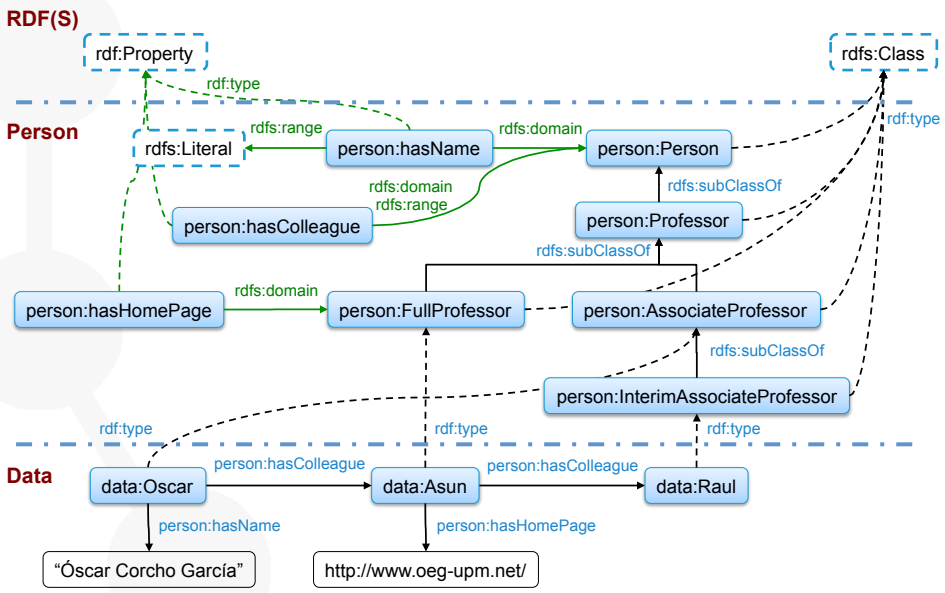
Describing classes



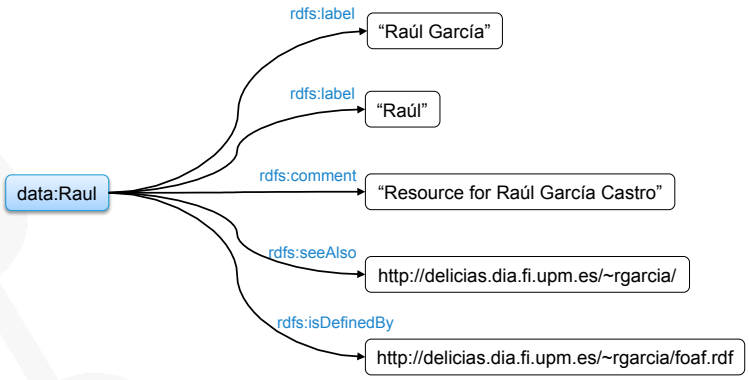
Describing individuals



Describing properties



Annotating resources



RDF-S vocabulary summary

Classes	Properties	Individuals
Classification		
rdfs:Resource	*rdfs:subClassOf*	
rdfs:Class		
Properties		
	rdfs:domain	
	rdfs:range	
	rdfs:subPropertyOf	
Datatypes		
rdfs:Literal		
rdfs:Datatype		
Containers		
rdfs:Container	*rdfs:member*	
rdfs:ContainerMembershipProperty		
Annotation		
	rdfs:label	
	rdfs:comment	
	rdfs:seeAlso	
	rdfs:isDefinedBy	

Index

RDF/XML syntax (1/2)

```
<?xml version="1.0"?>

    <!DOCTYPE rdf:RDF [
      <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
      <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
      <!ENTITY person "http://www.oeg-upm.net/ontologies/person#" >
    ]>

    <rdf:RDF xmlns="http://www.oeg-upm.net/ontologies/person#"
         xml:base="http://www.oeg-upm.net/ontologies/person"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:person="http://www.oeg-upm.net/ontologies/person#">

      <rdfs:Property rdf:about="&person;hasColleague">
        <rdfs:range rdf:resource="#Person"/>
        <rdfs:domain rdf:resource="#Person"/>
      </rdfs:Property>

      <rdfs:Property rdf:about="&person;hasHomePage">
        <rdfs:domain rdf:resource="#FullProfessor"/>
      </rdfs:Property>

      <rdfs:Property rdf:about="&person;hasName">
        <rdfs:domain rdf:resource="#Person"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
      </rdfs:Property>
...
```


RDF/XML syntax (2/2)

```
...
     <rdfs:Class rdf:about="#AssociateProfessor">
       <rdfs:subClassOf rdf:resource="#Professor"/>
     </rdfs:Class>

     <rdfs:Class rdf:about="#FullProfessor">
       <rdfs:subClassOf rdf:resource="#Professor"/>
     </rdfs:Class>

     <rdfs:Class rdf:about="#InterimAssociateProfessor">
       <rdfs:subClassOf rdf:resource="#AssociateProfessor"/>
     </rdfs:Class>

     <rdfs:Class rdf:about="#Person"/>

     <rdfs:Class rdf:about="#Professor">
       <rdfs:subClassOf rdf:resource="#Person"/>
     </rdfs:Class>

     <FullProfessor rdf:about="#Asun">
       <person:hasHomePage>http://www.fi.upm.es</person:hasHomePage>
       <person:hasColleague rdf:resource="#Raul"/>
     </FullProfessor>

     <AssociateProfessor rdf:about="#Oscar">
       <person:hasName>Oscar Corcho García</person:hasName>
       <person:hasColleague rdf:resource="#Asun"/>
     </AssociateProfessor>

     <InterimAssociateProfessor rdf:about="#Raul"/>
   </rdf:RDF>
```

Turtle syntax (1/2)

```
@base <http://www.oeg-upm.net/ontologies/person> .
@prefix : <http://www.oeg-upm.net/ontologies/person#> .
@prefix person: <http://www.oeg-upm.net/ontologies/person#> .

:hasColleague a rdfs:Property .
              rdfs:domain :Person .
              rdfs:range :Person ;

:hasHomePage a rdfs:Property .
             rdfs:domain :FullProfessor .

:hasName a rdfs:Property .
         rdfs:domain :Person ;
         rdfs:range rdfs:Literal .
…
```

a is equivalent to rdf:type

Turtle syntax (2/2)

```
…
:Person a rdfs:Class .

:Professor a rdfs:Class ;
           rdfs:subClassOf :Person .

:FullProfessor a rdfs:Class ;
               rdfs:subClassOf :Professor .

:AssociateProfessor a rdfs:Class ;
                    rdfs:subClassOf :Professor .

:InterimAssociateProfessor a rdfs:Class ;
                           rdfs:subClassOf :AssociateProfessor .

:Asun a :FullProfessor ;
      :hasHomePage "http://www.fi.upm.es" ;
      :hasColleague :Raul .

:Oscar a :AssociateProfessor ;
       :hasName "Oscar Corcho García" ;
       :hasColleague :Asun .

:Raul a :InterimAssociateProfessor .
```

a is equivalent to rdf:type

Index

- Overview
- RDF
  - Introduction
  - RDF components
  - Serializing RDF
  - Hands-on
- RDF Schema
  - Introduction
  - Main RDF-S components
  - Serializing RDF-S
  - **Hands-on**
  - RDF(S) Limitations
  - Hands-on
- Reasoning
  - Reasoning with RDF
  - Reasoning with RDF-S
- RDF(S) management APIs

Exercise

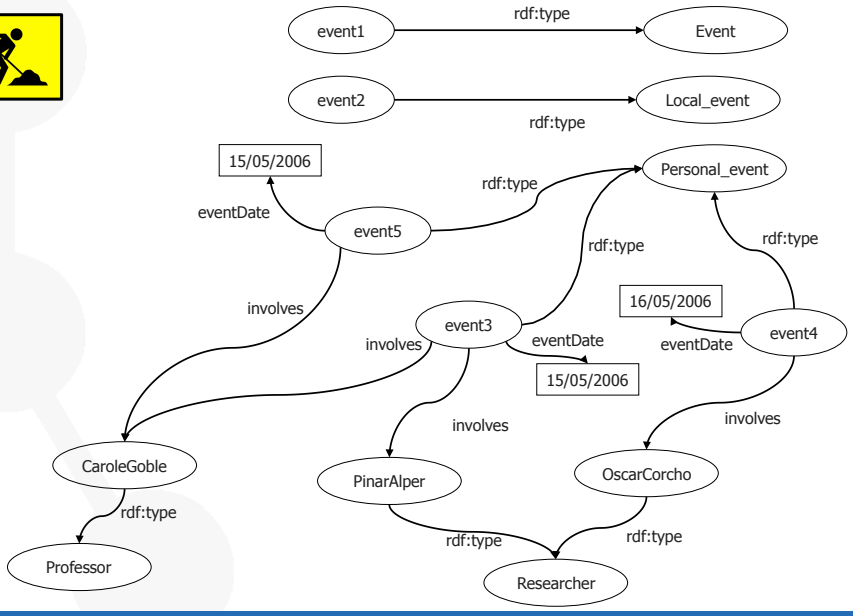
•Objective
- Get used to the different syntaxes of RDF(S)

•Tasks
- Take the text of an RDF(S) file and create its corresponding graph
- Take an RDF(S) graph and create its corresponding RDF/XML and N3 files

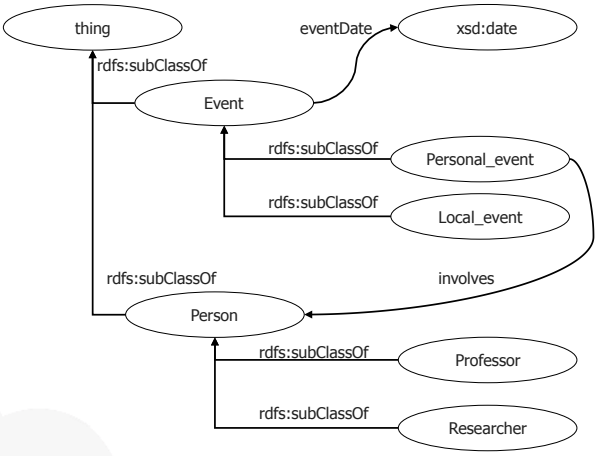
Exercise 2.a. Create a graph from a file

- Open the files StickyNote.rdf and StickyNote.rdfs
- Create the corresponding graph from them
- Compare your graph with those of your colleagues

Exercise 2.a. StickyNote.rdf

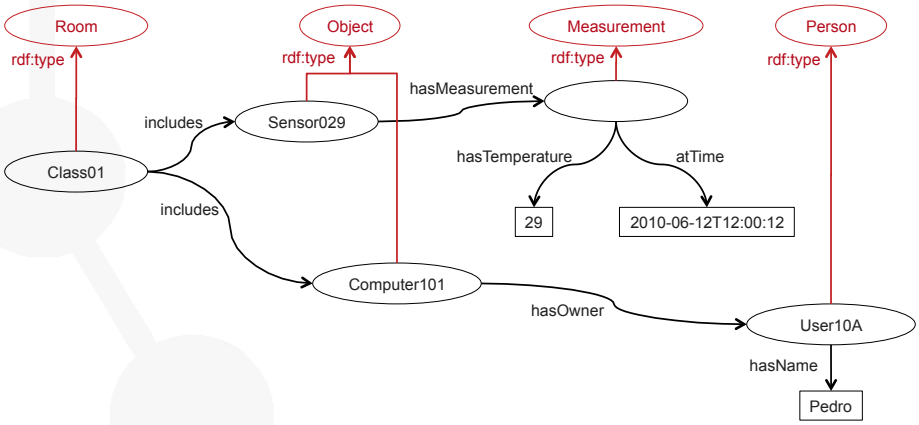


Exercise 2.a. StickyNote.rdfs



Exercise 2.b. Create files from a graph

- Transform the following graph into the RDF/XML and N3 syntaxes



Index

- Overview
- RDF
  - Introduction
  - RDF components
  - Serializing RDF
  - Hands-on
- **RDF Schema**
  - Introduction
  - Main RDF-S components
  - Serializing RDF-S
  - Hands-on
  - **RDF(S) Limitations**
  - Hands-on
- Reasoning
  - Reasoning with RDF
  - Reasoning with RDF-S
- RDF(S) management APIs

RDF(S) limitations

- RDFS too weak to describe resources in sufficient detail
  - No localised range and domain constraints
    - Can't say that the range of hasChild is person when applied to persons and elephant when applied to elephants
  - No existence/cardinality constraints
    - Can't say that all *instances* of person have a mother that is also a person, or that persons have exactly 2 parents
  - No boolean operators
    - Can't say or, not, etc.
  - No transitive, inverse or symmetrical properties
    - Can't say that isPartOf is a transitive property, that hasPart is the inverse of isPartOf or that touches is symmetrical
- Difficult to provide reasoning support
  - No "native" reasoners for non-standard semantics
  - May be possible to reason via FOL axiomatisation

Index

- Overview
- RDF
  - Introduction
  - RDF components
  - Serializing RDF
  - Hands-on
- **RDF Schema**
  - Introduction
  - Main RDF-S components
  - Serializing RDF-S
  - Hands-on
  - RDF(S) Limitations
  - **Hands-on**
- Reasoning
  - Reasoning with RDF
  - Reasoning with RDF-S
- RDF(S) management APIs

Exercise

•Objective
- Understand the features of RDF(S) for implementing ontologies, including its limitations

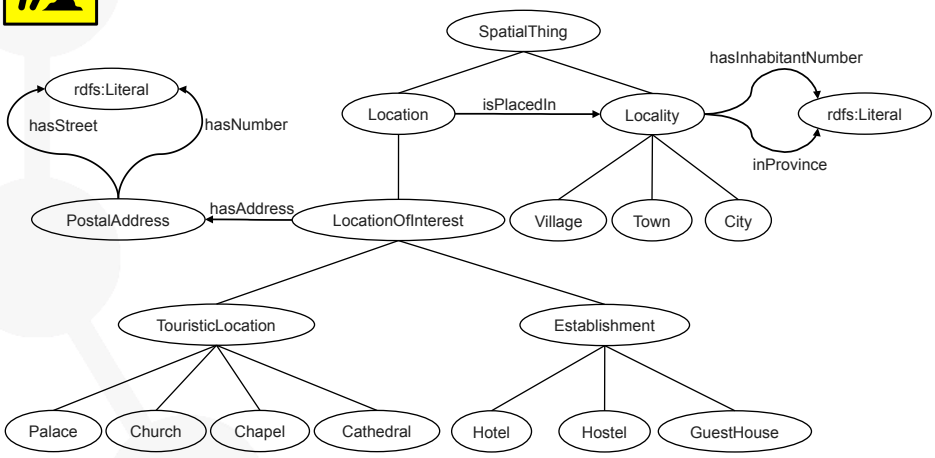
•Tasks
- From a domain description, create the RDF(S) graph
  - First only include the vocabulary from the domain
  - Then include references to the RDF and RDFS vocabularies

Exercise 3. Domain description

- A certain location can be a location of interest.
- Locations of interest can be touristic locations or establishments, but not both at the same time.
- Touristic locations can be palaces, churches, chapels, and cathedrals.
- Establishments can be hotels, hostels, or guest houses.
- A location is placed in a locality, which can be in turn a town, a village, or a city.
- A location of interest has some postal address that includes the street name and number.
- Localities have a certain number of inhabitants.
- Localities belong to a certain province.

- Covarrubias is a village with 634 inhabitants in the province of Burgos.
- The “El Galo” restaurant is placed in Covarrubias, in Mayor street, number 5.
- One of Covarrubias’ churches is at Santo Tomas street.

Exercise 3. Sample resulting ontology



Index

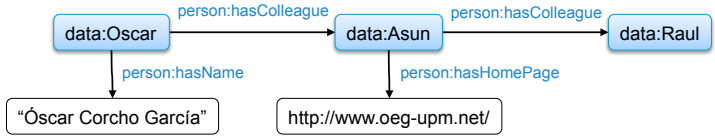
- Overview
- RDF
  - Introduction
  - RDF components
  - Serializing RDF
  - Hands-on
- RDF Schema
  - Introduction
  - Main RDF-S components
  - Serializing RDF-S
  - Hands-on
  - RDF(S) Limitations
  - Hands-on
- **Reasoning**
  - **Reasoning with RDF**
  - Reasoning with RDF-S
- RDF(S) management APIs

RDF inference. Graph matching techniques

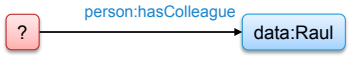
- RDF inference is based on graph matching techniques
- Basically, the RDF inference process consists of the following steps:
  - Transform an RDF query into a template graph that has to be matched against the RDF graph
    - It contains constant and variable nodes, and constant and variable edges between nodes
  - Match against the RDF graph, taking into account constant nodes and edges
  - Provide a solution for variable nodes and edges

RDF inference. Examples (I)

- Sample RDF graph



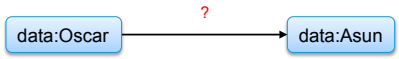
- **Query**: "Tell me who are the persons who have Asun as a colleague"



  - **Result**: data:Asun

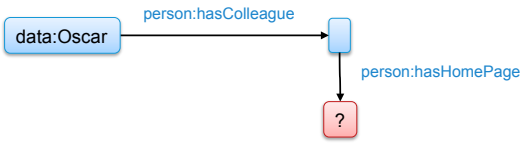
RDF inference. Examples (II)

- **Query**: "Tell me which are the relationships between Oscar and Asun"



  - **Result**: person:hasColleague

- **Query**: "Tell me the homepage of Oscar colleagues"



  - **Result**: "http://www.oeg-upm.net/"

RDF inference. Entailment rules

Rule Name	if E contains	then add
rdf1	uuu aaa yyy .	aaa `rdf:type rdf:Property .`
rdf2	uuu aaa lll . where lll is a well-typed XML literal .	_:nnn `rdf:type rdf:XMLLiteral .` where _:nnn identifies a blank node allocated to lll by rule lg.

Index

- Overview
- RDF
  - Introduction
  - RDF components
  - Serializing RDF
  - Hands-on
- RDF Schema
  - Introduction
  - Main RDF-S components
  - Serializing RDF-S
  - Hands-on
  - RDF(S) Limitations
  - Hands-on
- **Reasoning**
  - Reasoning with RDF
  - **Reasoning with RDF-S**
- RDF(S) management APIs

RDF(S) inference. Entailment rules

Rule Name	If E contains:	then add:
rdfs1	uuu aaa lll. where lll is a plain literal (with or without a language tag).	_:nnn rdf:type rdfs:Literal . where _:nnn identifies a blank node allocated to lll by rule rule lg.
rdfs2	aaa rdfs:domain xxx . uuu aaa yyy .	uuu rdf:type xxx .
rdfs3	aaa rdfs:range xxx . uuu aaa vvv .	vvv rdf:type xxx .
rdfs4a	uuu aaa xxx .	uuu rdf:type rdfs:Resource .
rdfs4b	uuu aaa vvv.	vvv rdf:type rdfs:Resource .
rdfs5	uuu rdfs:subPropertyOf vvv . vvv rdfs:subPropertyOf xxx .	uuu rdfs:subPropertyOf xxx .
rdfs6	uuu rdf:type rdf:Property .	uuu rdfs:subPropertyOf uuu .
rdfs7	aaa rdfs:subPropertyOf bbb . uuu aaa yyy .	uuu bbb yyy .
rdfs8	uuu rdf:type rdfs:Class .	uuu rdfs:subClassOf rdfs:Resource .
rdfs9	uuu rdfs:subClassOf xxx . vvv rdf:type uuu .	vvv rdf:type xxx .
rdfs10	uuu rdf:type rdfs:Class .	uuu rdfs:subClassOf uuu .
rdfs11	uuu rdfs:subClassOf vvv . vvv rdfs:subClassOf xxx .	uuu rdfs:subClassOf xxx .
rdfs12	uuu rdf:type rdfs:ContainerMembershipProperty .	uuu rdfs:subPropertyOf rdfs:member .
rdfs13	uuu rdf:type rdfs:Datatype .	uuu rdfs:subClassOf rdfs:Literal .

RDF(S) inference. Additional inferences

ext1	uuu rdfs:domain vvv . vvv rdfs:subClassOf zzz .	uuu rdfs:domain zzz .
ext2	uuu rdfs:range vvv . vvv rdfs:subClassOf zzz .	uuu rdfs:range zzz .
ext3	uuu rdfs:domain vvv . www rdfs:subPropertyOf uuu .	www rdfs:domain vvv .
ext4	uuu rdfs:range vvv . www rdfs:subPropertyOf uuu .	www rdfs:range vvv .
ext5	rdf:type rdfs:subPropertyOf www . www rdfs:domain vvv .	rdfs:Resource rdfs:subClassOf vvv .
ext6	rdfs:subClassOf rdfs:subPropertyOf www . www rdfs:domain vvv .	rdfs:Class rdfs:subClassOf vvv .
ext7	rdfs:subPropertyOf rdfs:subPropertyOf www . www rdfs:domain vvv .	rdf:Property rdfs:subClassOf vvv .
ext8	rdfs:subClassOf rdfs:subPropertyOf www . www rdfs:range vvv .	rdfs:Class rdfs:subClassOf vvv .
ext9	rdfs:subPropertyOf rdfs:subPropertyOf www . www rdfs:range vvv .	rdf:Property rdfs:subClassOf vvv .

Index

- Overview
- RDF
  - Introduction
  - RDF components
  - Serializing RDF
  - Hands-on
- RDF Schema
  - Introduction
  - Main RDF-S components
  - Serializing RDF-S
  - Hands-on
  - RDF(S) Limitations
  - Hands-on
- Reasoning
  - Reasoning with RDF
  - Reasoning with RDF-S
- **RDF(S) management APIs**

Sample RDF APIs

- RDF libraries for different languages:
  - Java, Python, C, C++, C#, .Net, Javascript, Tcl/Tk, PHP, Lisp, Obj-C, Prolog, Perl, Ruby, Haskell
  - List in http://esw.w3.org/topic/SemanticWebTools
- Usually related to a RDF repository
- Multilanguage:
  - Redland RDF Application Framework (C, Perl, PHP, Python and Ruby): http://www.redland.opensource.ac.uk/
- Java:
  - Jena: http://jena.sourceforge.net/
  - Sesame: http://www.openrdf.org/
- PHP:
  - RAP - RDF API for PHP: http://www4.wiwiss.fu-berlin.de/bizer/rdfapi/
- Python:
  - RDFLib: http://rdflib.net/
  - Pyrple: http://infomesh.net/pyrple/

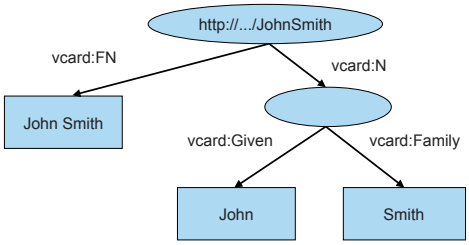
Jena

- Java framework for building Semantic Web applications
- Open source software from HP Labs
- The Jena framework includes:
  - A RDF API
  - An OWL API
  - Reading and writing RDF in RDF/XML, N3 and N-Triples
  - In-memory and persistent storage
  - A rule based inference engine
  - SPARQL query engine

Sesame

- A framework for storage, querying and inferencing of RDF and RDF Schema
- A Java Library for handling RDF
- A Database Server for (remote) access to repositories of RDF data
- Highly expressive query and transformation languages
  - SeRQL, SPARQL
- Various backends
  - Native Store
  - RDBMS (MySQL, Oracle 10, DB2, PostgreSQL)
  - Main memory
- Reasoning support
  - RDF Schema reasoner
  - OWL DLP (OWLIM)
  - Domain reasoning (custom rule engine)

Jena example. Graph creation




```
// some definitions
String personURI = "http://somewhere/JohnSmith";
String givenName = "John";
String familyName = "Smith";
String fullName = givenName + " " + familyName;
// create an empty
Model Model model = ModelFactory.createDefaultModel();
// create the resource
// and add the properties cascading style
Resource johnSmith = model.createResource(personURI)
     .addProperty(VCARD.FN, fullName)
     .addProperty(VCARD.N, model.createResource()
     .addProperty(VCARD.Given, givenName)
     .addProperty(VCARD.Family, familyName));
```

http://jena.sourceforge.net/tutorial/RDF_API/

Jena example. Read and write

```
// create an empty model
Model model = ModelFactory.createDefaultModel();

// use the FileManager to find the input file
InputStream in = FileManager.get().open( inputFileName );
if (in == null) {
    throw new IllegalArgumentException("File not found");
}

// read the RDF/XML file
model.read(in, "");

// write it to standard out
model.write(System.out);
```

```
<rdf:RDF
  xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
  xmlns:vcard='http://www.w3.org/2001/vcard-rdf/3.0#'
 >
  <rdf:Description rdf:nodeID="A0">
    <vcard:Family>Smith</vcard:Family>
    <vcard:Given>John</vcard:Given>
  </rdf:Description>
  <rdf:Description rdf:about='http://somewhere/JohnSmith/'>
    <vcard:FN>John Smith</vcard:FN>
    <vcard:N rdf:nodeID="A0"/>
  </rdf:Description>
...
</rdf:RDF>
```

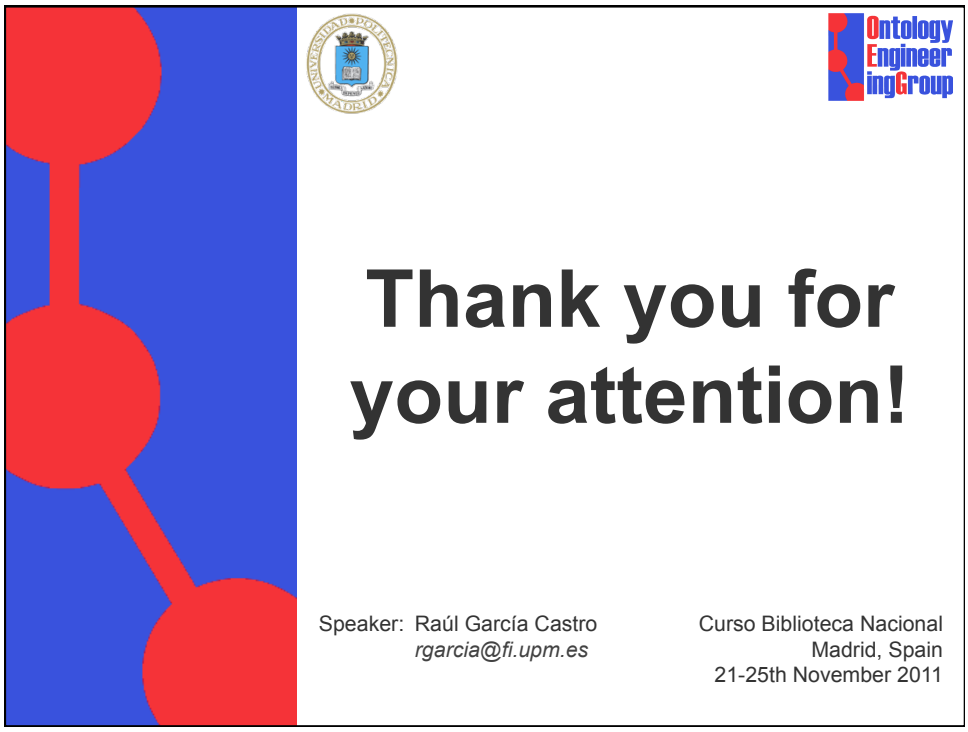
http://jena.sourceforge.net/tutorial/RDF_API/

Some RDF editors

- IsaViz
  - http://www.w3.org/2001/11/IsaViz/
- Morla
  - http://www.morlardf.net/
- RDFAuthor
  - http://rdfweb.org/people/damian/RDFAuthor/
- RdfGravity
  - http://semweb.salzburgresearch.at/apps/rdf-gravity/index.html
- Rhodonite
  - http://rhodonite.angelite.nl/

Main References

- Brickley D, Guha RV (2004) RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation
  http://www.w3.org/TR/PR-rdf-schema/
- Lassila O, Swick R (1999) Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation
  http://www.w3.org/TR/REC-rdf-syntax/
- RDF validator:
  http://www.w3.org/RDF/Validator/
- RDF resources:
  http://planetrdf.com/guide/


Ontology
Engineer
ingGroup
Thank you for your attention!
Speaker: Raúl García Castro
rgarcia@fi.upm.es
Curso Biblioteca Nacional
Madrid, Spain
21-25th November 2011