# Part 1 : RDF and RDFS Modeling

<?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#"

xmlns:persen="http://xmlns.com/foaf/0.1/"

xmlns:city="http://purl.org/dc/elements/1.1/"

xmlns:sy="http://purl.org/dc/elements/1.1/"

xmlns:ju="http://purl.org/dc/elements/1.1/">

<!-- City-->

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

<city:is\_capital\_of>Austria</city:is\_capital\_of>

</rdf:Description>

<!-- Jupiter -->

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

<ju:composed>Allegro Vivace(Jupiter)</ju:composed>

<ju:composed>Andante Cantabile(Jupiter)</ju:composed>

<ju:composed>Menuetto(Jupiter)</ju:composed>

<ju:composed>Molto Allegro(Jupiter)</ju:composed>

</rdf:Description>

<!-- Persen -->

<rdf:Description rdf:about="http://xmlns.com/foaf/0.1/Wolfgang Amadeus Mozart">

<persen:born\_at>Salzburg</persen:born\_at>

<rdfs:death\_at rdf:resource="#Vienna"/>

<persen:son\_of>Léopold Mozart</persen:son\_of>

<persen:husband\_of>Constance Weber</persen:husband\_of>

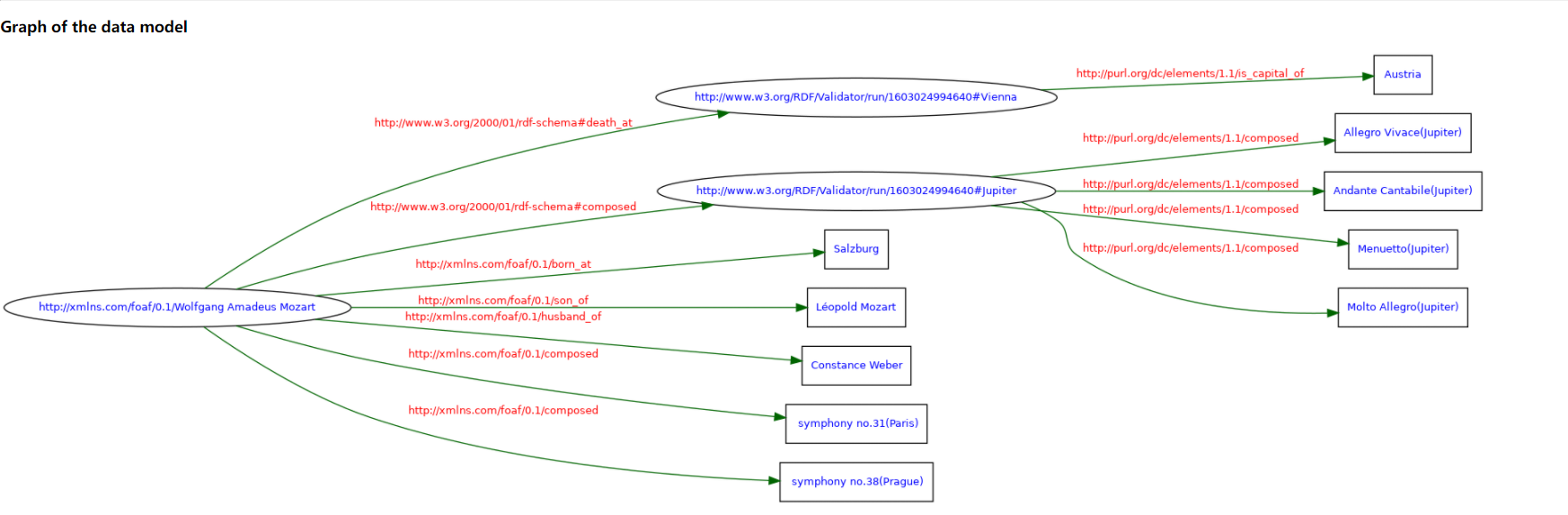
<persen:composed> symphony no.31(Paris)</persen:composed>

<persen:composed> symphony no.38(Prague)</persen:composed>

<rdfs:composed rdf:resource="#Jupiter"/>

</rdf:Description >

</rdf:RDF>



# Part 2 : Jena API

package Tutorial;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.OutputStream;

import org.apache.jena.riot.Lang;

import org.apache.jena.riot.RDFDataMgr;

import com.hp.hpl.jena.rdf.model.Model;

import com.hp.hpl.jena.rdf.model.ModelFactory;

import com.hp.hpl.jena.rdf.model.Property;

import com.hp.hpl.jena.rdf.model.RDFNode;

import com.hp.hpl.jena.rdf.model.Resource;

import com.hp.hpl.jena.rdf.model.Statement;

import com.hp.hpl.jena.rdf.model.StmtIterator;

import com.hp.hpl.jena.vocabulary.VCARD;

public class HelloRDFWorld

{

public static void main(String[] args)

{

// TODO Auto-generated method stub

//Person resource

String personURI = "http://xmlns.com/foaf/0.1/Wolfgang Amadeus Mozart ";

String husband = "Constance Weber";

String son = "Léopold Mozart";

String born = "Salzburg";

String composed1 = "http://xmlns.com/foaf/0.1/symphony no.41(Jupiter)";

String composed2 = "Symphony no.31(Paris)";

String composed3 = "Symphony no.38(Prague)";

//Part resource

String part1 = "Allegro Vivace(Jupiter)";

String part2 = "Andante Cantabile(Jupiter)";

String part3 = "Menuetto(Jupiter)";

String part4 = "Molto Allegro(Jupiter)";

//Capital resource

String CityURI = "http://xmlns.com/foaf/0.1/Vienna";

String City = "Vienna";

// create an empty Model

Model model = ModelFactory.createDefaultModel();

StmtIterator iter = model.listStatements();

//create the Property

//person property

Property hus = model.createProperty("husband of");

Property s = model.createProperty("son of");

Property b = model.createProperty("born at");

Property com1 = model.createProperty("composed");

Property com2 = model.createProperty("composed");

Property com3 = model.createProperty("composed");

//como = be composed of 4 parts

//part property

Property como1 = model.createProperty("be composed of");

Property como2 = model.createProperty("be composed of");

Property como3 = model.createProperty("be composed of");

Property como4 = model.createProperty("be composed of");

//capital property

Property dead = model.createProperty("death at");

Property capital = model.createProperty("is capital of");

// create the resource and add the properties

//Person

Resource Mozart

= model.createResource(personURI)

.addProperty(hus,husband)

.addProperty(s,son)

.addProperty(b,born)

.addProperty(com2,composed2)

.addProperty(com3,composed3)

//parts

.addProperty(com1,

model.createResource(composed1)

.addProperty(como1, part1)

.addProperty(como2, part2)

.addProperty(como3, part3)

.addProperty(como4, part4))

//capital

.addProperty(dead,

model.createResource(CityURI)

.addProperty(capital, City));

while(iter.hasNext())

{

Statement stmt = iter.nextStatement();

// get next statement

Resource subject = stmt.getSubject();

// get the subject

Property predicate = stmt.getPredicate();

// get the predicate

RDFNode object = stmt.getObject();

// get the object

System.out.print(subject.toString());

System.out.print(" " + predicate.toString() + " ");

if (object instanceof Resource) {

System.out.print(object.toString());

} else {

// object is a literal

System.out.print(" \"" + object.toString() + "\"");

}

System.out.println(" .");

}

try(OutputStream out = new FileOutputStream("filename.rdf")) {

RDFDataMgr.write(out, model, Lang.TURTLE);

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

}

# Part 3 : SPARQL Querying

Using SPARQL, write the queries to answer the following questions :

**import** com.hp.hpl.jena.query.Query;

**import** com.hp.hpl.jena.query.QueryExecution;

**import** com.hp.hpl.jena.query.QueryExecutionFactory;

**import** com.hp.hpl.jena.query.QueryFactory;

**import** com.hp.hpl.jena.query.QuerySolution;

**import** com.hp.hpl.jena.query.ResultSet;

**import** com.hp.hpl.jena.rdf.model.Literal;

**import** com.hp.hpl.jena.rdf.model.Model;

**import** com.hp.hpl.jena.util.FileManager;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

*sparqlTest*();

}

**static** **void** sparqlTest()

{

FileManager.*get*().addLocatorClassLoader(Main.**class**.getClassLoader());

Model model = FileManager.*get*().loadModel("f:/workspace/Jenapart3/SW-FAQ-feed.rdf");

String queryString =

"PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>" +

"PREFIX dc: <http://purl.org/dc/elements/1.1/>" +

"PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>" +

"PREFIX rss: <http://purl.org/rss/1.0/>" +

"PREFIX swq: <http://www.w3.org/2001/sw/SW-FAQ#>" +

"SELECT DISTINCT ?title\r\n" +

"FROM <http://www.w3.org/2001/sw/SW-FAQ-feed.rdf>" +

"WHERE { " +

"<http://www.w3.org/2001/sw/SW-FAQ> rss:title ?title" +

"}";

Query query = QueryFactory.*create*(queryString);

QueryExecution qexec = QueryExecutionFactory.*create*(query,model);

**try** {

ResultSet results = qexec.execSelect();

**while**(results.hasNext()) {

QuerySolution soln = results.nextSolution();

Literal name = soln.getLiteral("title");

System.***out***.println(name);

}

}**finally** {

qexec.close();

}

}

}

//All the query just need to change the blue part, so I just show the consequence and normall query.

1. What is the title (rss:title) of the RSS feed (rss:channel) ?

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

PREFIX dc: <http://purl.org/dc/elements/1.1/>

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX rss: <http://purl.org/rss/1.0/>

PREFIX swq: <http://www.w3.org/2001/sw/SW-FAQ#>

SELECT DISTINCT ?title

FROM <http://www.w3.org/2001/sw/SW-FAQ-feed.rdf>

WHERE {

<http://www.w3.org/2001/sw/SW-FAQ> rss:title ?title

}



1. What are the titles of the published items (rss:item) in the RSS feed?

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

PREFIX dc: <http://purl.org/dc/elements/1.1/>

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX rss: <http://purl.org/rss/1.0/>

PREFIX swq: <http://www.w3.org/2001/sw/SW-FAQ#>

SELECT DISTINCT ?title

FROM <http://www.w3.org/2001/sw/SW-FAQ-feed.rdf>

WHERE {

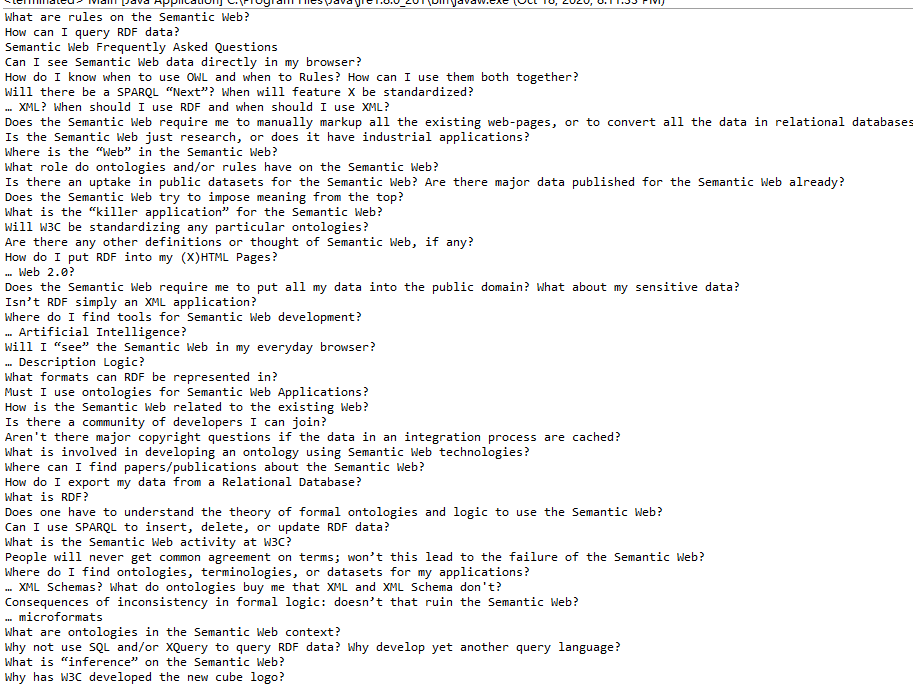
?x rss:title ?title

NOT EXISTS

{

<http://www.w3.org/2001/sw/SW-FAQ> rss:title ?title

}}



1. Give the first ten items of the RSS feed ?

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

PREFIX dc: <http://purl.org/dc/elements/1.1/>

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX rss: <http://purl.org/rss/1.0/>

PREFIX swq: <http://www.w3.org/2001/sw/SW-FAQ#>

SELECT ?items ?title

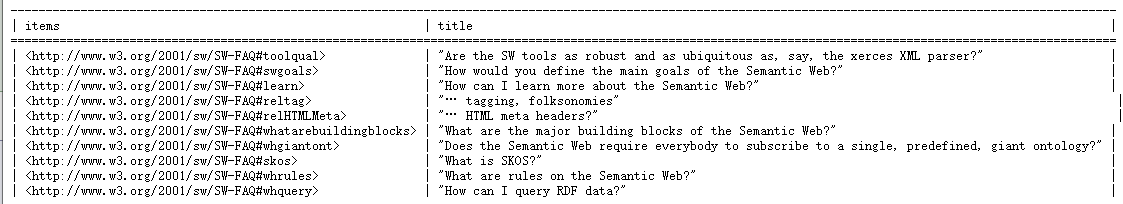
FROM <http://www.w3.org/2001/sw/SW-FAQ-feed.rdf>

WHERE {

?items rss:title ?title

}

LIMIT 10



1. What is the penultimate item of the RSS feed (chronological order) ?

// I use the first query to search how many items and then use the second query to offset the count

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

PREFIX dc: <http://purl.org/dc/elements/1.1/>

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX rss: <http://purl.org/rss/1.0/>

PREFIX swq: <http://www.w3.org/2001/sw/SW-FAQ#>

SELECT (count(?title) AS ?count)

FROM <http://www.w3.org/2001/sw/SW-FAQ-feed.rdf>

WHERE {

?items rss:title ?title

}

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

PREFIX dc: <http://purl.org/dc/elements/1.1/>

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX rss: <http://purl.org/rss/1.0/>

PREFIX swq: <http://www.w3.org/2001/sw/SW-FAQ#>

SELECT ?items ?date

FROM <http://www.w3.org/2001/sw/SW-FAQ-feed.rdf>

WHERE {

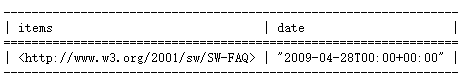
?items dc:date ?date

}

order by ?date

offset 51

limit 1



1. Is there any items published at the same date ?

// I find these items, but I can’t show them, I can just show the numbers of item which has same date

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

PREFIX dc: <http://purl.org/dc/elements/1.1/>

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX rss: <http://purl.org/rss/1.0/>

PREFIX swq: <http://www.w3.org/2001/sw/SW-FAQ#>

SELECT ?items ?date (COUNT(?date) AS ?count)

FROM <http://www.w3.org/2001/sw/SW-FAQ-feed.rdf>

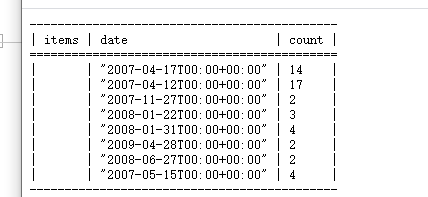
where {

?items dc:date ?date

}

group by ?date

having(COUNT(?date)>1)



1. What items have been published on 2007-04-12?? (Suggestion : date format “20070412T00:00+00:00”ˆˆxsd:string)

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

PREFIX dc: <http://purl.org/dc/elements/1.1/>

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX rss: <http://purl.org/rss/1.0/>

PREFIX swq: <http://www.w3.org/2001/sw/SW-FAQ#>

SELECT ?items ?date

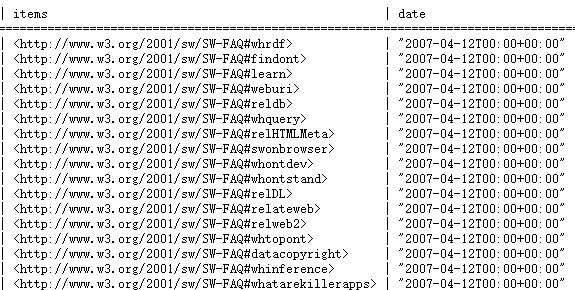
FROM <http://www.w3.org/2001/sw/SW-FAQ-feed.rdf>

where {

?items dc:date ?date

FILTER (?date = "2007-04-12T00:00+00:00"^^xsd:string)

}



1. Give the list of authors (without repetition) items in this RSS feed ?

// I m not understand why there are the repetition authors? Because every items’ author is Ivan

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

PREFIX dc: <http://purl.org/dc/elements/1.1/>

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX rss: <http://purl.org/rss/1.0/>

PREFIX swq: <http://www.w3.org/2001/sw/SW-FAQ#>

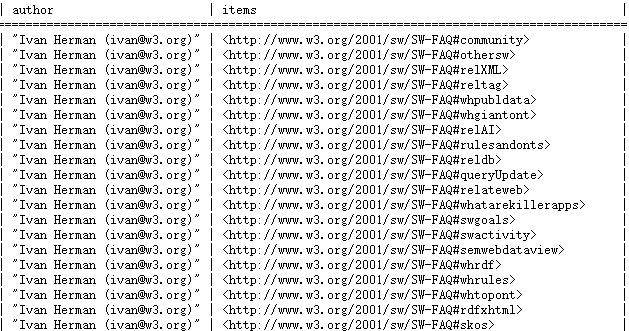
SELECT ?author ?items

FROM <http://www.w3.org/2001/sw/SW-FAQ-feed.rdf>

where {

?items dc:author ?author

}



1. Does the RSS feed use the "title" property defined by the Dublin Core namespace ? (dublincore.org/)

NO, “the” title property is defined by rss.

1. Does the RSS feed use the "subject" property defined by the Dublin Core namespace ?

Yes. The “subject” property is defined by DC.

1. List the publication dates of articles in the RSS feed.

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

PREFIX dc: <http://purl.org/dc/elements/1.1/>

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX rss: <http://purl.org/rss/1.0/>

PREFIX swq: <http://www.w3.org/2001/sw/SW-FAQ#>

SELECT ?date ?articles

FROM <http://www.w3.org/2001/sw/SW-FAQ-feed.rdf>

where {

?articles dc:date ?date

}

