UNIVERSITY OF
LEICESTER

Semantic Web

(CO7516)

Assignment 3

Programming in Jena and OWL API

Dr. Monika Solanki
Tutor: Dr Emma Tonkin

1 - Modelling a "lightweight" version of the OWL ontology

1. Simplified version of OWL ontology

To explore the capabilities of Apache Jena and Protégé OWL API to programmatically manipulate and query, the ontology of the construction materials has been simplified to a lightweight version as shown in Fig.1.

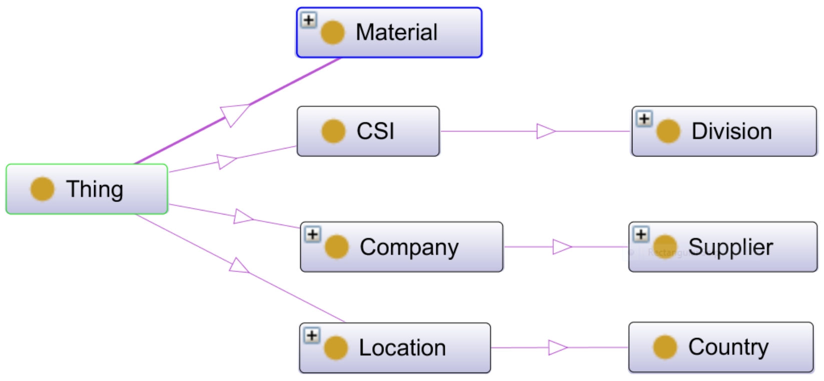


Figure 1: Simplified Version of Ontology

At this stage it is useful to eliminate the object and data properties to reduce the complexity of the source code generate by Protégé-OWL code generator. These properties are shown in Table 1 and 2.

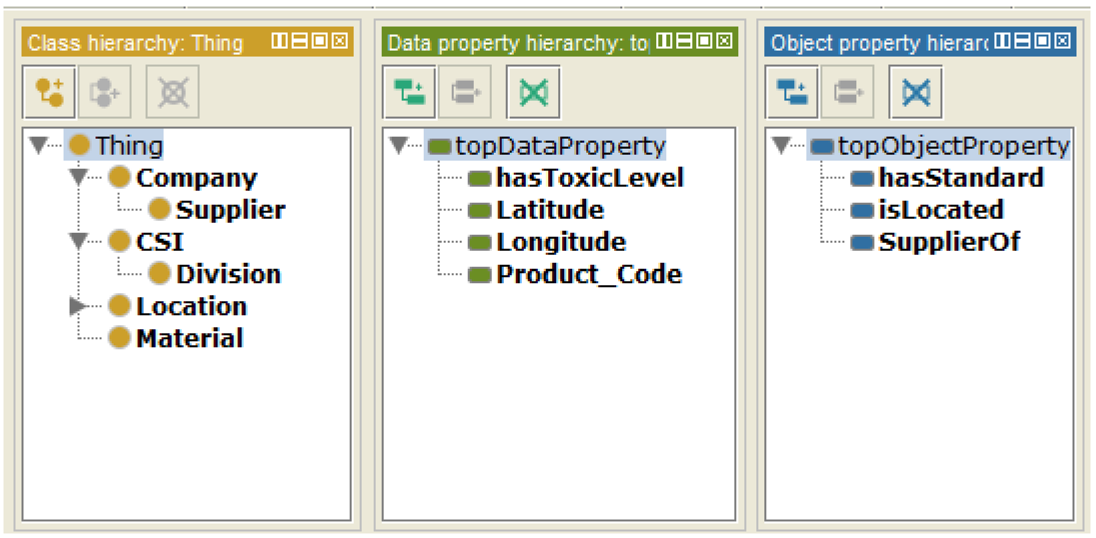
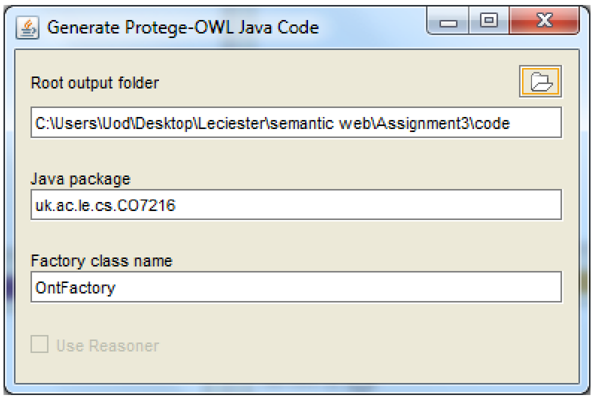


Figure 2: Object and Data Properties

2. Java template

This lightweight ontology used to generate the Java template using Protégé 4 **(Protégé->Tools->Generate Protégé- OWL API**) as shown in Fig.3. This template is used to modify the classes, instances and establish the relationships in the ontology.


Generate Protege-OWL Java Code

Root output folder
C:\Users\Uod\Desktop\Leciester\semantic web\Assignment3\code

Java package
uk.ac.le.cs.CO7216

Factory class name
OntFactory

Use Reasoner

Figure 3: Protégé-OWL Java Code

A java project "CO7216_CW3" is created in eclipse and the Jena and Protégé libraries are configured in the Java Built Path along with the Java template imported in the source folder. The project structure is shown in Fig.4.

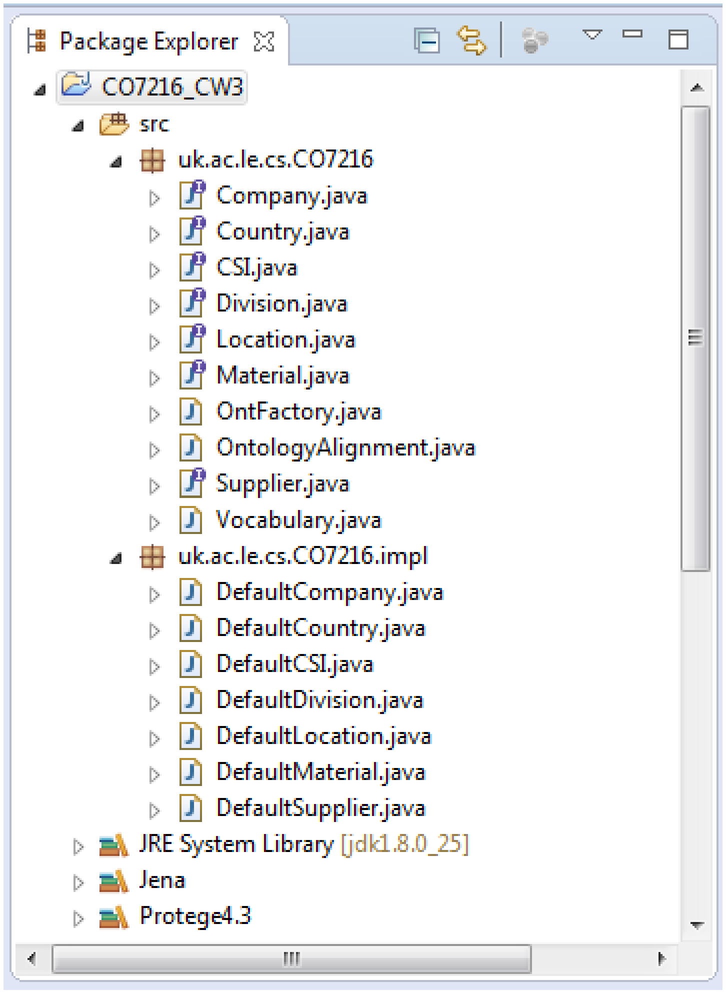

```
Package Explorer
CO7216_CW3
  src
    uk.ac.le.cs.CO7216
      Company.java
      Country.java
      CSI.java
      Division.java
      Location.java
      Material.java
      OntFactory.java
      OntologyAlignment.java
      Supplier.java
      Vocabulary.java
    uk.ac.le.cs.CO7216.impl
      DefaultCompany.java
      DefaultCountry.java
      DefaultCSI.java
      DefaultDivision.java
      DefaultLocation.java
      DefaultMaterial.java
      DefaultSupplier.java
  JRE System Library [jdk1.8.0_25]
  Jena
  Protege4.3
```

Figure 4: Project Structure

2 - Resources intended to use

The ontology of the Construction Material that covers Specifications Institute (CSI) to classifies the construction materials in CSI divisions. However, no significant research and resources are available in the semantic web. Ultimately, I had to configure the FUSEKI server to host the endpoint of my own set of tuples. For the construction Material, I found the following endpoints useful.
http://semantic.eurobau.com/

The data consists of ca. 1.5 million individual RDF/XML for European building and construction materials, especially for products, suppliers, and warehouses in N-Triples syntax.

http://www.freeclass.eu/freeclass_v1.html

http://purl.org/goodrelations/v1#

This ontology defines 5,676 classes and 174 properties, and 1,423 qualitative values for products and services from the construction and building materials domain.

2.1 - Identified Methods to populate ontology

Discovered three methods to populate and align the ontologies, i.e. Automated, Manual and Hybrid alignments defined in next sections.

3. Automated Alignment

This method retains the relationships in the target ontology and is the best reuse of existing resources. As shown in Fig. 5. Relation between Companies and Products “dbo:product” is retained but renamed to “isSupplier”. However, when requires new relationships, for instance, when defining Material hasDivision Division relation that does not exists in any of the source ontology this method has limited options.

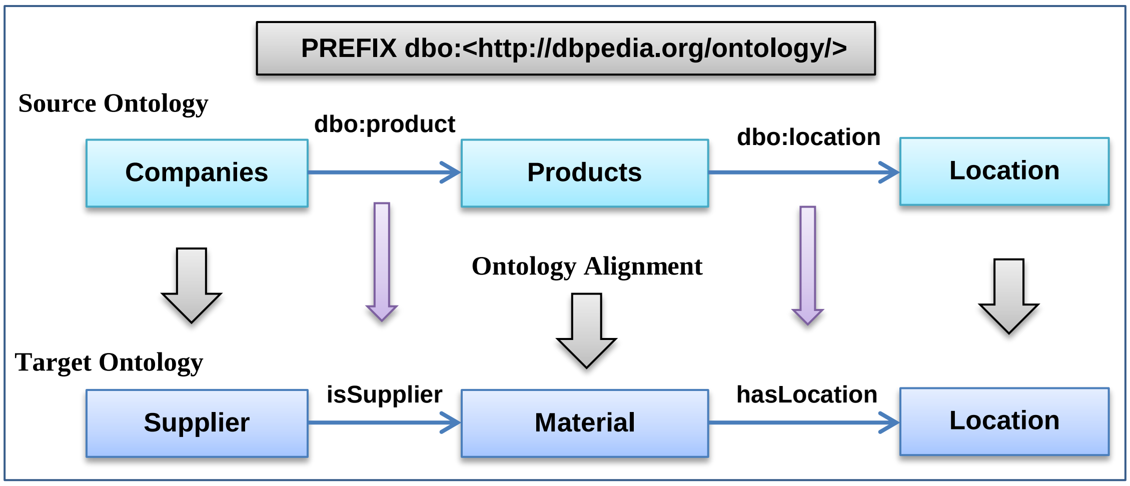


Figure 5: Automated Alignment

4. Manual Alignment

Another method is to populate individual entities in the target and define relationships manually. This is time consuming and ideally not good practice of ontology reuse. As shown in Fig. 6 two ontologies, "**goodrelation**" and "**freeclass**" are merged in the target and the relation "**isSupplier**" is defined manually for each of the instances.

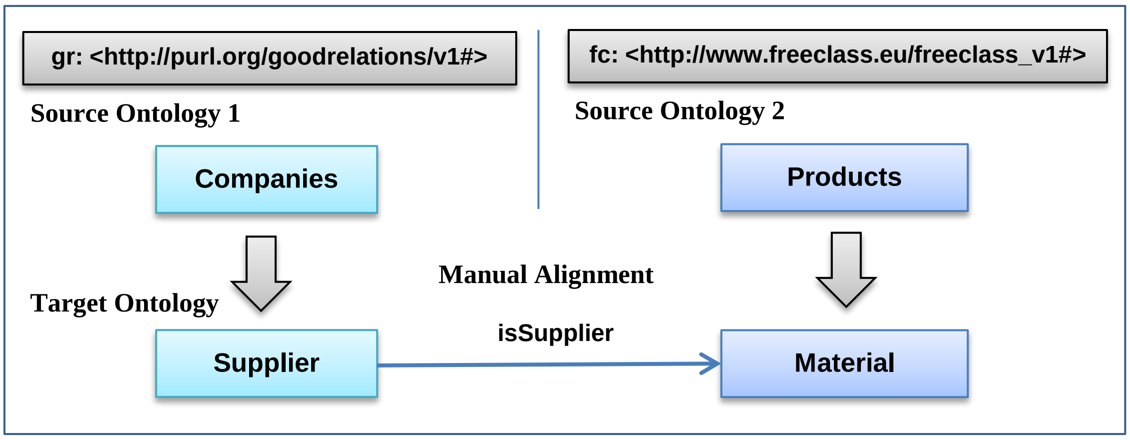


Figure 6: Manual Alignment

5. Hybrid Alignment:

This method (used in this coursework) is blending of the two methods. Partial data is fetched in the form of tuples. For instance the tuple "Company dbo:product Products" is directly merged and the "Material hasDivision Division" from the Fuseki Server merged and the relation is defined manually as shown in Fig.7.

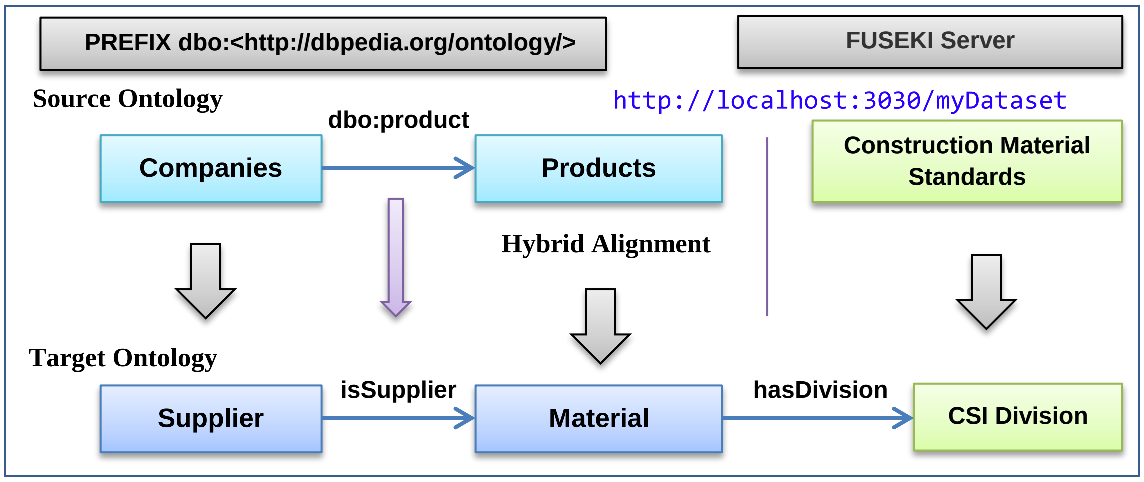


Figure 7: Hybrid Alignment using FUSEIKI server

3 - Implement Ontology Alignment

3.1 - Query repositories via SPARQL endpoints using Jena API

A list of endpoints and prefixes has been stored in a separate text file to query repositories using JENA API. The overall flow is shown in Fig 8.

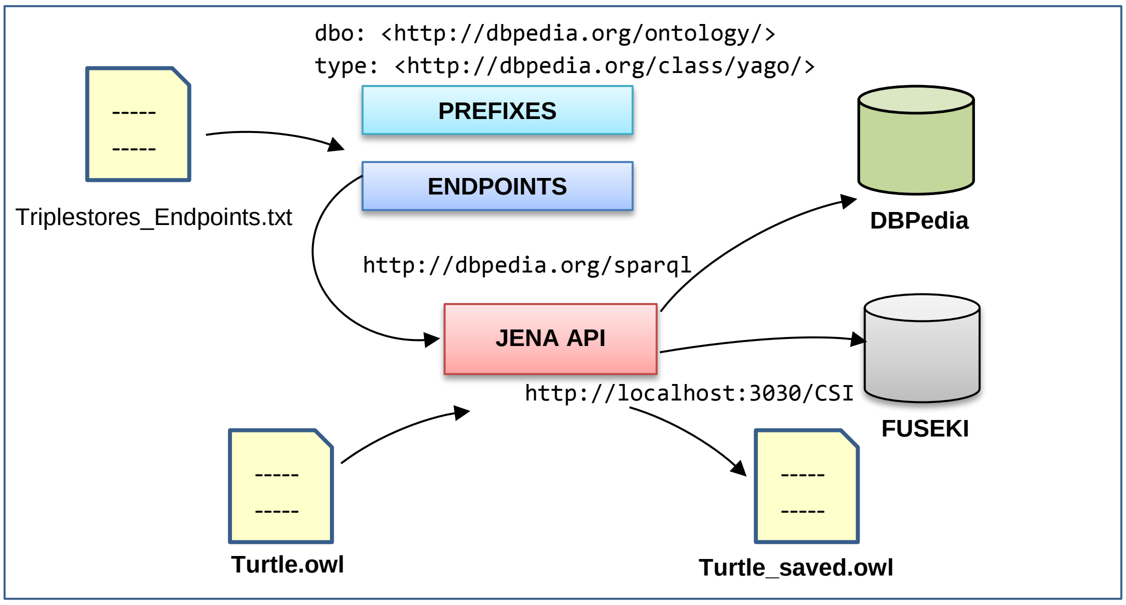

dbo: <http://dbpedia.org/ontology/>
type: <http://dbpedia.org/class/yago/>
PREFIXES
ENDPOINTS
Triplestores_Endpoints.txt
DBPedia
http://dbpedia.org/sparql
JENA API
http://localhost:3030/CSI
FUSEKI
Turtle.owl
Turtle_saved.owl

Figure 8: Query repositories via SPARQL endpoints

The file "Triplestores_Endpoints.txt" is located at the root of the java project. This technique helps users to query endpoints without modifying the source code of the program.

3.1.1 - Load TripleStores and Prefixes in ArrayList

The following code snippet load the triplestore endpoints in an array, download

```
FileInputStream fstream = new FileInputStream("Triplestores_Endpoints.txt");

  while ((strLine = buffer.readLine()) != null)   {
        String[] array=strLine.split(",");
        sparqlEndpoints.add(array[0]);
        prefix.add(array[1]);
  }
```

Code: Query online endpoints for Supplier Material Location

```
String sparql_Supplier ="PREFIX type:
<http://dbpedia.org/class/yago/>\n"+
  "PREFIX dbo: <http://dbpedia.org/ontology/>\n"+
  "SELECT ?supplier ?material ?location ?lat ?long WHERE {\n"+
  "?supplier  a type:Company108058098;" +
  " dbo:product      ?material;" +
  " dbo:location     ?location;" +
  " <http://www.w3.org/2003/01/geo/wgs84_pos#lat> ?lat;" +
  " <http://www.w3.org/2003/01/geo/wgs84_pos#long> ?long." +
  "} LIMIT 10";

executeSPARQL(sparql_Supplier,sparqlEndpoints.get(0));
```

Code: Query Local FUSEKI Server for CSI Standard divisions

```
String sparql_Local_endpoint ="PREFIX
owl:<http://www.w3.org/2002/07/owl#>\n"+
        "PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>\n"+
        "PREFIX csi:<"+ prefix.get(1) + ">"+
        "SELECT ?material ?division WHERE {\n"+
        " ?material a csi:Material;" +
         " csi:hasStandard ?division;" +
         "} LIMIT 10";

executeFUSEKI(sparql_Local_endpoint, sparqlEndpoints.get(1));
```

3.1.2 Configuration Steps

The binary distribution of the Fuseki Jena SPARQL server is downloaded from: apache-jena-fuseki-2.3.1.zip

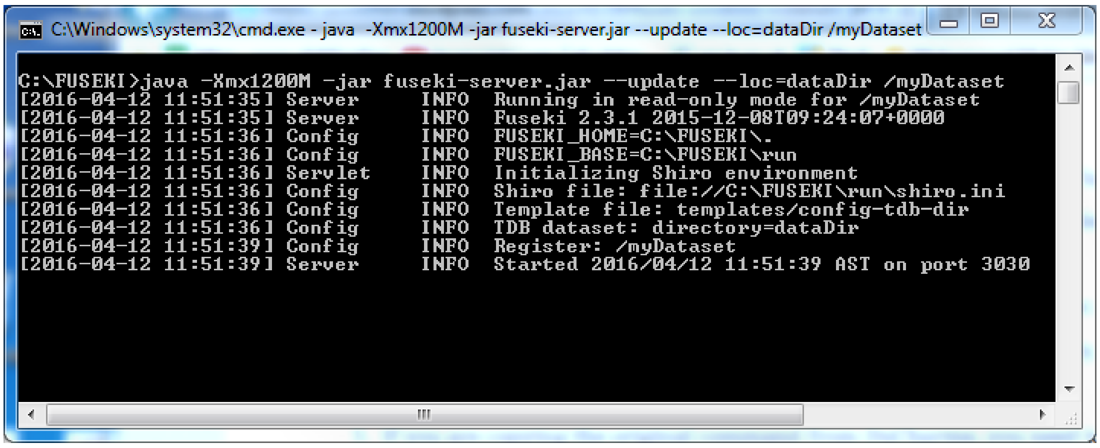
Extract the Fuseki zip file (http://jena.apache.org/download/index.cgi) to **"C:\FUSEKI"**

Create a folder /dataDir in the FUSEIKI folder

3.1.3 Running the Fuseki Server

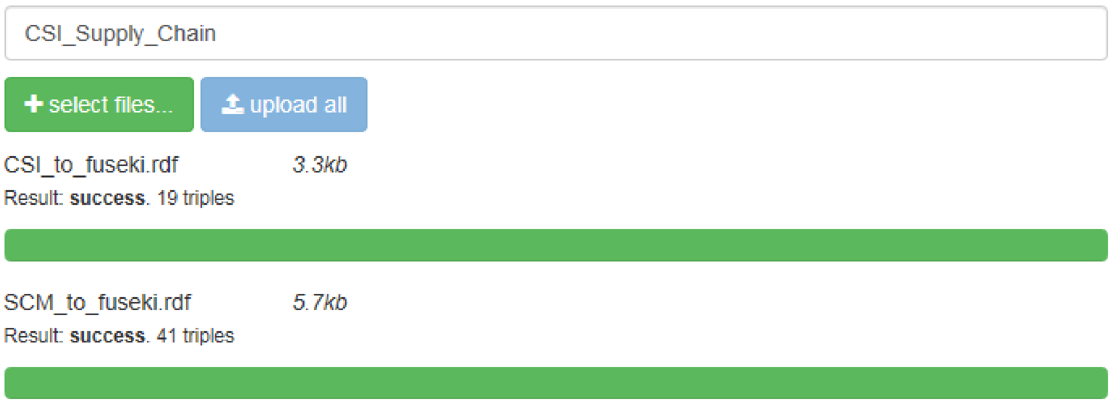
In the **Windows command prompt,** navigate to the **Fuseki folder**, and type the following command:

java -Xmx1200M -jar fuseki-server.jar --update --loc=dataDir /myDataset



```
C:\Windows\system32\cmd.exe - java  -Xmx1200M -jar fuseki-server.jar --update --loc=dataDir /myDataset

C:\FUSEKI>java -Xmx1200M -jar fuseki-server.jar --update --loc=dataDir /myDataset
[2016-04-12 11:51:35] Server     INFO  Running in read-only mode for /myDataset
[2016-04-12 11:51:35] Server     INFO  Fuseki 2.3.1 2015-12-08T09:24:07+0000
[2016-04-12 11:51:36] Config     INFO  FUSEKI_HOME=C:\FUSEKI\.
[2016-04-12 11:51:36] Config     INFO  FUSEKI_BASE=C:\FUSEKI\run
[2016-04-12 11:51:36] Servlet    INFO  Initializing Shiro environment
[2016-04-12 11:51:36] Config     INFO  Shiro file: file://C:\FUSEKI\run\shiro.ini
[2016-04-12 11:51:36] Config     INFO  Template file: templates/config-tdb-dir
[2016-04-12 11:51:36] Config     INFO  TDB dataset: directory=dataDir
[2016-04-12 11:51:39] Config     INFO  Register: /myDataset
[2016-04-12 11:51:39] Server     INFO  Started 2016/04/12 11:51:39 AST on port 3030
```

Figure 9: FUSEKI server running on Port 3030



CSI_to_fuseki.rdf 3.3kb
Result: **success**. 19 triples

SCM_to_fuseki.rdf 5.7kb
Result: **success**. 41 triples

Figure 10: Triplets uploaded to the endpoint

Download these RDFs : CSI_to_fuseki.rdf and SCM_to_Fuseki.rdf

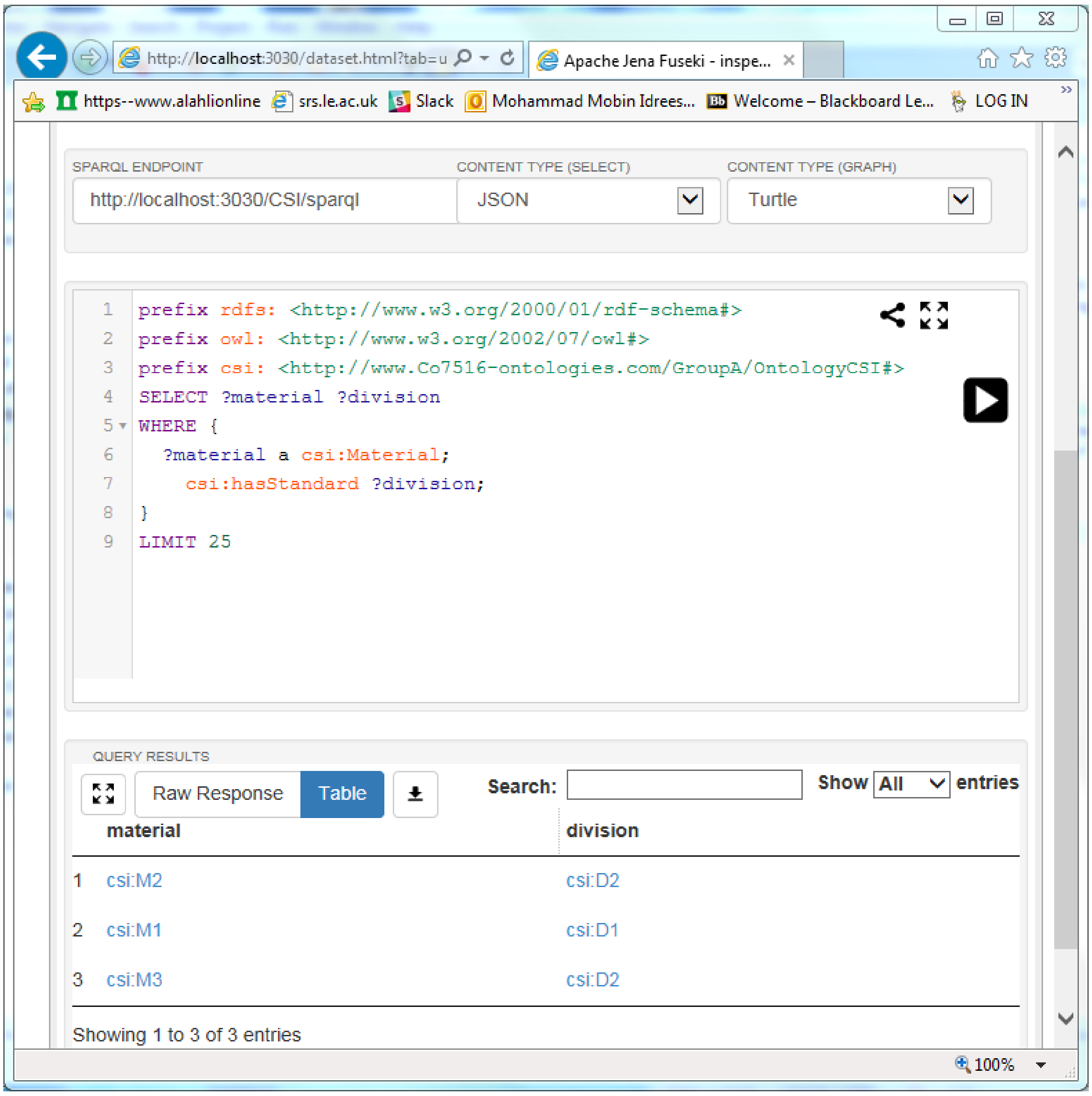
The local endpoint contains "M1" ,"M2" and "M3" materials and "D1" , "D2" CSI division and having following relations.

M1 hasDivision D1,

M2 hasDivision D2,

M3 hasDivision D2.

Figure 11, represents an online query to display this information.


http://localhost:3030/dataset.html?tab=u
Apache Jena Fuseki - inspe...
https--www.alahlionline
srs.le.ac.uk
Slack
Mohammad Mobin Idrees...
Welcome – Blackboard Le...
LOG IN

SPARQL ENDPOINT: http://localhost:3030/CSI/sparql

CONTENT TYPE (SELECT): JSON

CONTENT TYPE (GRAPH): Turtle

```
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix owl: <http://www.w3.org/2002/07/owl#>
prefix csi: <http://www.Co7516-ontologies.com/GroupA/OntologyCSI#>
SELECT ?material ?division
WHERE {
  ?material a csi:Material;
    csi:hasStandard ?division;
}
LIMIT 25
```

QUERY RESULTS

	material	division
1	csi:M2	csi:D2
2	csi:M1	csi:D1
3	csi:M3	csi:D2

Showing 1 to 3 of 3 entries

Figure 11: Query CSI - Apache Jena FUSEKI Server

3.2 - Populate the ontology

The following code snipped used to populate the ontology using DBpedia class "yago" that contains list of companies used to populate the target ontology. A point here is to convert Latitude and Longitute literals using lat.asLiteral().getFloat());

3.2.1 Populate using DBPedia endpoints

```
public void executeSPARQL(String sparql, String sparqlEndpoint){

//create instances
ResultSet results = qexec.execSelect();
while(results.hasNext()){
QuerySolution qs = results.nextSolution();

   //Retrieve variable from SPARQL Query
  RDFNode sup=qs.get("supplier");
  RDFNode material=qs.get("material");
  RDFNode location=qs.get("location");
  RDFNode lat=qs.get("lat");
  RDFNode lon=qs.get("lon");

  Supplier inst_supplier=factory.createSupplier(sup.toString());
  Material inst_material=factory.createMaterial(material.toString());
  Location inst_loc=factory.createLocation(location.toString());

  // Populate Default Relationships defined in online Triplestores
  inst_supplier.addSupplierOf(inst_material);
  inst_supplier.addIsLocated(inst_loc);
  inst_supplier.addLatitude(lat.asLiteral().getFloat());
  inst_supplier.addLongitude(lat.asLiteral().getFloat());
   supplier_count++;
   }

   System.out.print(supplier_count+" Instances Created"+"\n");

}
```

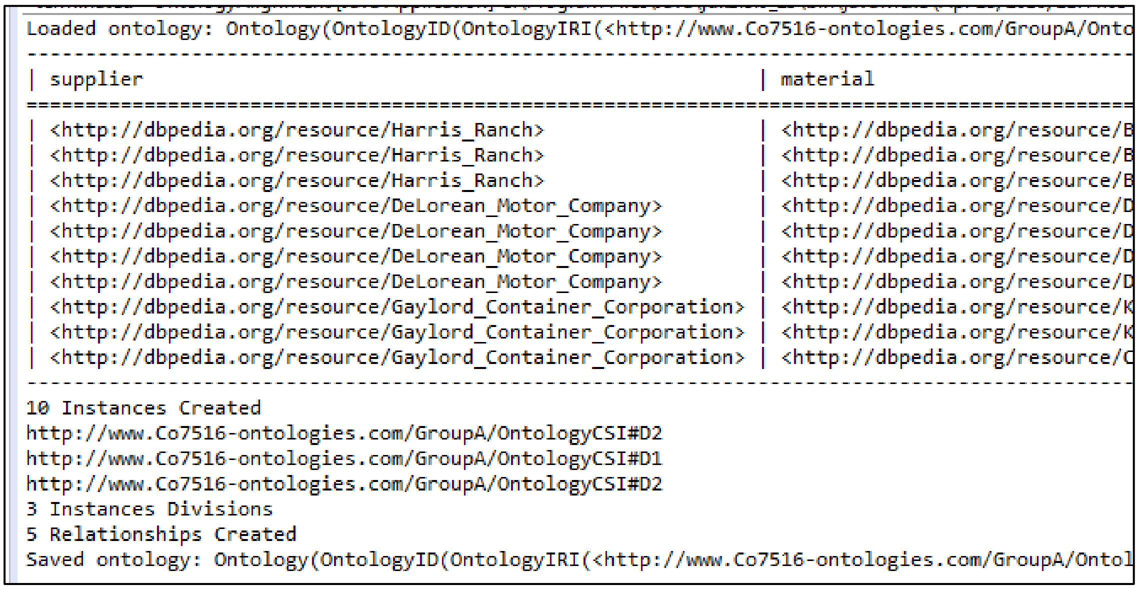
3.2.2 Populate using FUSEKI on localhost Server

By default this option has been disabled in the source code and the reason is that you might not have the CSI ontology and FUSEKI running on your system. Therefore, in the target ontology you would not see the instance D1. D2 etc.

```
public void executeFUSEKI(String sparql, String sparqlEndpoint){

int inst_Division = 0;

//create instances CSI
while(results.hasNext()){
        QuerySolution qs = results.nextSolution();
        RDFNode div=qs.get("division");
        Division inst_division=factory.createDivision(div.toString());
        System.out.print(div+"\t");
        System.out.print("\n");
        inst_Division++;
  }
  System.out.print(inst_Division+" Instances Divisions"+"\n");

}
```

3.3 Property values for these individuals and relationships

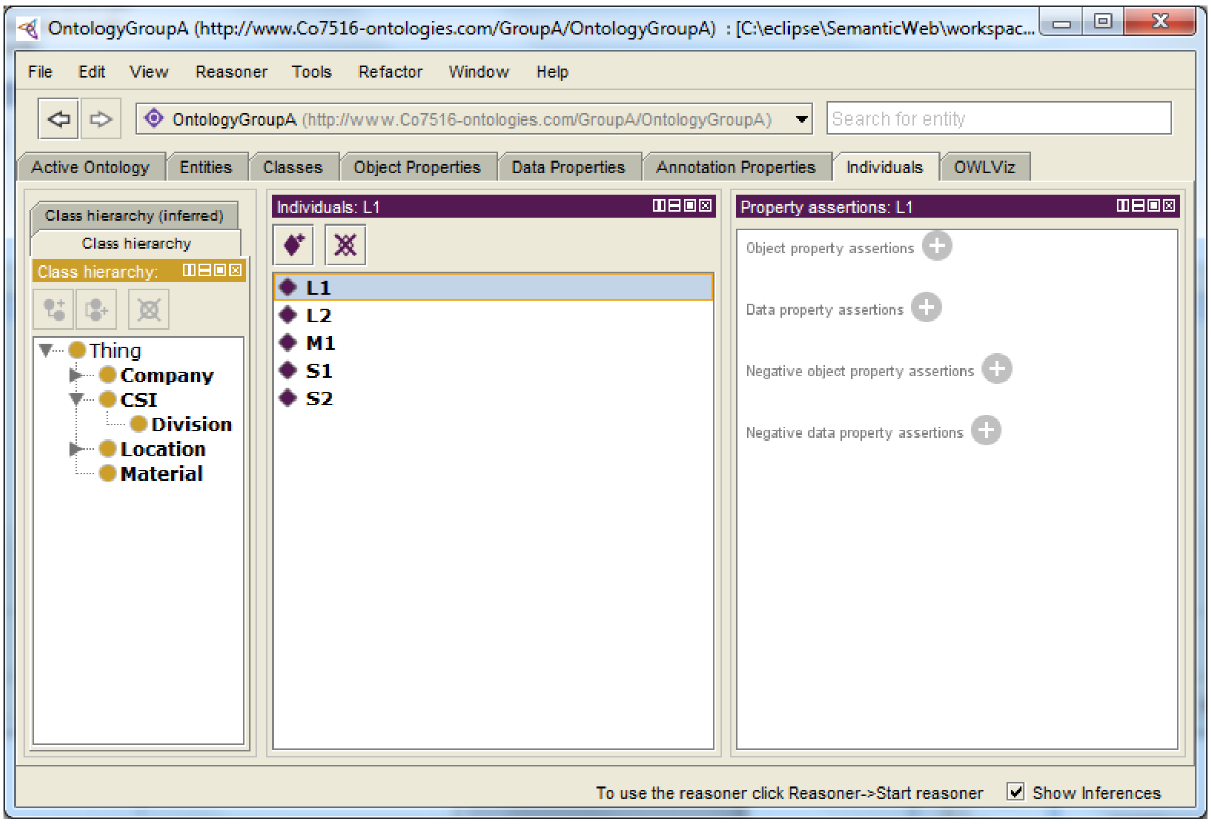
```
//Set Object Relationships for SupplierOf (Domain Supplier, Range
Material)

factory.getSupplier(prefix.get(0)
+"S1").addSupplierOf(factory.getMaterial(prefix.get(0) +"M1"));

//Set Object Relationships for Location (Domain Supplier, Range
Location)

factory.getSupplier(prefix.get(0)
+"S1").addIsLocated(factory.getLocation(prefix.get(0) +"L1"));

factory.getSupplier(prefix.get(0)
+"S2").addIsLocated(factory.getLocation(prefix.get(0) +"L2"));

//Set Data Properties for Material
factory.getMaterial(prefix.get(0) +"M1").addProduct_Code("AB-101");
factory.getMaterial(prefix.get(0) +"M1").addHasToxicLevel(2);
System.out.print(" Relationships Created"+"\n");
```



```
Loaded ontology: Ontology(OntologyID(OntologyIRI(<http://www.Co7516-ontologies.com/GroupA/Onto
----------------------------------------------------------------------------------------------
| supplier                                                | material
==============================================================================================
| <http://dbpedia.org/resource/Harris_Ranch>                | <http://dbpedia.org/resource/B
| <http://dbpedia.org/resource/Harris_Ranch>                | <http://dbpedia.org/resource/B
| <http://dbpedia.org/resource/Harris_Ranch>                | <http://dbpedia.org/resource/B
| <http://dbpedia.org/resource/DeLorean_Motor_Company>      | <http://dbpedia.org/resource/D
| <http://dbpedia.org/resource/DeLorean_Motor_Company>      | <http://dbpedia.org/resource/D
| <http://dbpedia.org/resource/DeLorean_Motor_Company>      | <http://dbpedia.org/resource/D
| <http://dbpedia.org/resource/DeLorean_Motor_Company>      | <http://dbpedia.org/resource/D
| <http://dbpedia.org/resource/Gaylord_Container_Corporation> | <http://dbpedia.org/resource/K
| <http://dbpedia.org/resource/Gaylord_Container_Corporation> | <http://dbpedia.org/resource/K
| <http://dbpedia.org/resource/Gaylord_Container_Corporation> | <http://dbpedia.org/resource/C
----------------------------------------------------------------------------------------------
10 Instances Created
http://www.Co7516-ontologies.com/GroupA/OntologyCSI#D2
http://www.Co7516-ontologies.com/GroupA/OntologyCSI#D1
http://www.Co7516-ontologies.com/GroupA/OntologyCSI#D2
3 Instances Divisions
5 Relationships Created
Saved ontology: Ontology(OntologyID(OntologyIRI(<http://www.Co7516-ontologies.com/GroupA/Ontol
```

Figure 12: Instances and Relations Created in Java Eclipse

In order to avoid runtime errors, I have included few instances in the source ontology as you might not have running FUSEKI on your system. Therefore, in the target ontology you might not see the instance D1. D2 etc. unless to configure the server and upload the CSI to fuseki.rdf. The source and target ontology is shown in figure 13 and 14.



OntologyGroupA (http://www.Co7516-ontologies.com/GroupA/OntologyGroupA) : [C:\eclipse\SemanticWeb\workspac...

OntologyGroupA (http://www.Co7516-ontologies.com/GroupA/OntologyGroupA)

Class hierarchy:

- Thing
  - Company
  - CSI
    - Division
  - Location
  - Material

Individuals: L1

- L1
- L2
- M1
- S1
- S2

Property assertions: L1

Object property assertions

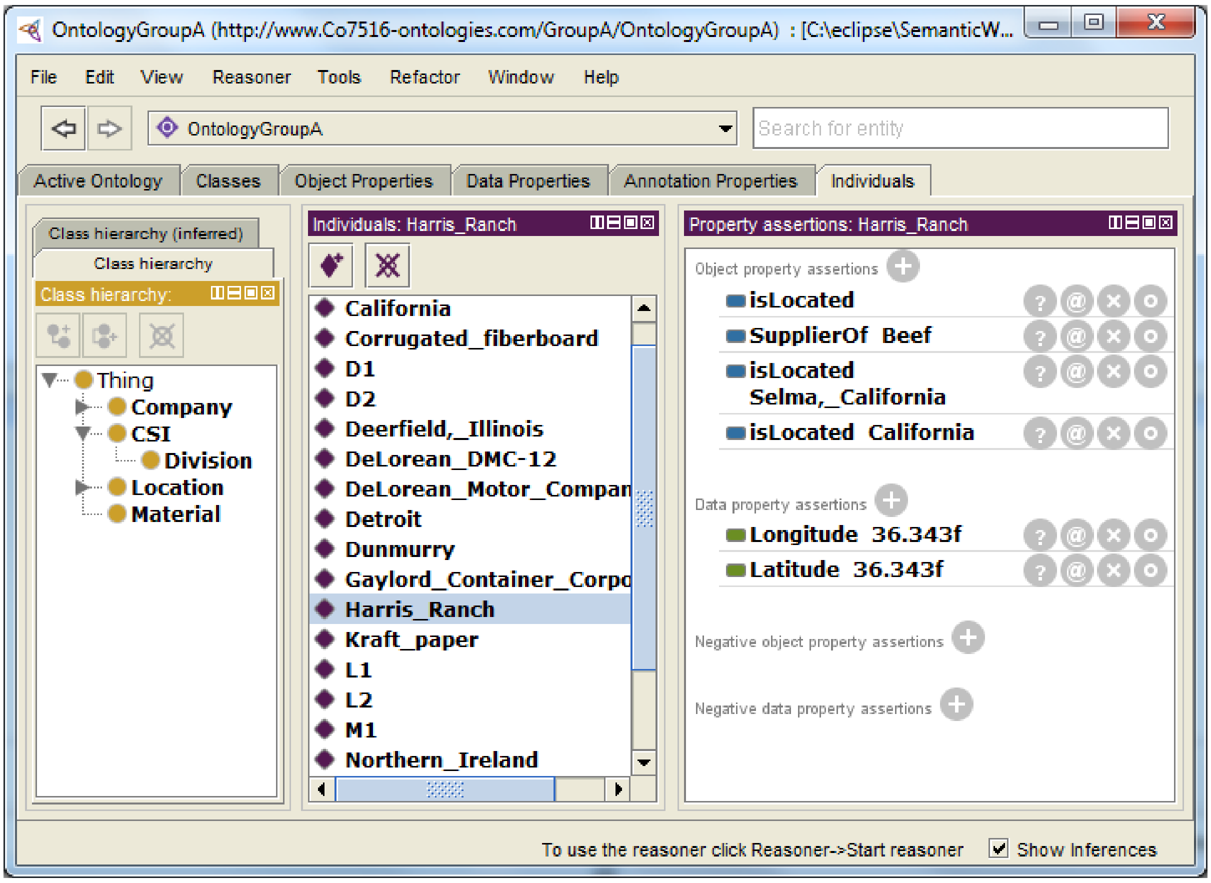
Data property assertions

Negative object property assertions

Negative data property assertions

To use the reasoner click Reasoner->Start reasoner

Figure 13: Input Ontology "turtle.owl"



OntologyGroupA (http://www.Co7516-ontologies.com/GroupA/OntologyGroupA) : [C:\eclipse\SemanticW...

OntologyGroupA

Class hierarchy:

- Thing
  - Company
  - CSI
    - Division
  - Location
  - Material

Individuals: Harris_Ranch

- California
- Corrugated_fiberboard
- D1
- D2
- Deerfield,_Illinois
- DeLorean_DMC-12
- DeLorean_Motor_Compan
- Detroit
- Dunmurry
- Gaylord_Container_Corpo
- Harris_Ranch
- Kraft_paper
- L1
- L2
- M1
- Northern_Ireland

Property assertions: Harris_Ranch

Object property assertions

- isLocated
- SupplierOf Beef
- isLocated Selma,_California
- isLocated California

Data property assertions

- Longitude 36.343f
- Latitude 36.343f

Negative object property assertions

Negative data property assertions

To use the reasoner click Reasoner->Start reasoner

Figure 14: Output Ontology "turtle_saved.owl"

4 Conclusion

A good piece of knowledge and hopefully wish to continue learning and research in this subject in the near future.

5 References

[1] Apache Jena : https://jena.apache.org/index.html [Accessed April 2016]
Protégé 4.3

[2] SPARQL Endpoints: https://www.w3.org/wiki/SparqlEndpoints. [Accessed April 2016]

[3] Tutorial: Programming the Semantic Web:
https://blackboard.le.ac.uk/bbcswebdav/pid-1348667-dt-content-rid-3496506_2/xid-3496506_2 [Accessed April 2016]

[4] Lecture Note: YiHong, Monika Solanki, Department of Computer Sciences, University of Leicester

[5] Lecture Notes: Khriyenko Oleksiy, Storing and querying RDF data, Practical Introduction to Semantic Technologies Autumn 2015University of Jyväskylä.

[6] Apache Jena - Fuseki: serving RDF data over HTTP:
https://jena.apache.org/documentation/serving_data/ [Accessed April 2016]