ISS SEMANTICS WEB

TP REPORT – 5 ISS INSA TOULOUSE

The purpose of this report is to show the skills that I have acquired and the concepts those I understood as a result of the courses and the couple of TP sessions.

The report is of 5 pages.

Lastname 1 : SELVARAJ

Firstname 1 : MOHAN PRABHU

A Kind Note- I did the TP's and the report as the only person.

Date
25-Dec-2019

1 Introduction

Using Protégé, I have attempted to create an ontology model for the 'Meteorological Application'. Followed by, using the sample dataset from Aarhus city, Denmark I attempted to understand

- how the 'Apache Jena wrapper classes' in Java can be used on the sample dataset
- how the ontology can be manipulated
- how the SPARQL queries can be executed using the Jena to perform the 'knowledge base' operations.

2 Creation of the ontology

As per RDF graph, the basic description unit is 'triple'. As per the 'triple', the 3 elements in RDF graph are,

- Subject – Place and type of place
- Property – Measurable parameters (Concepts)
- Object – Data value in a range

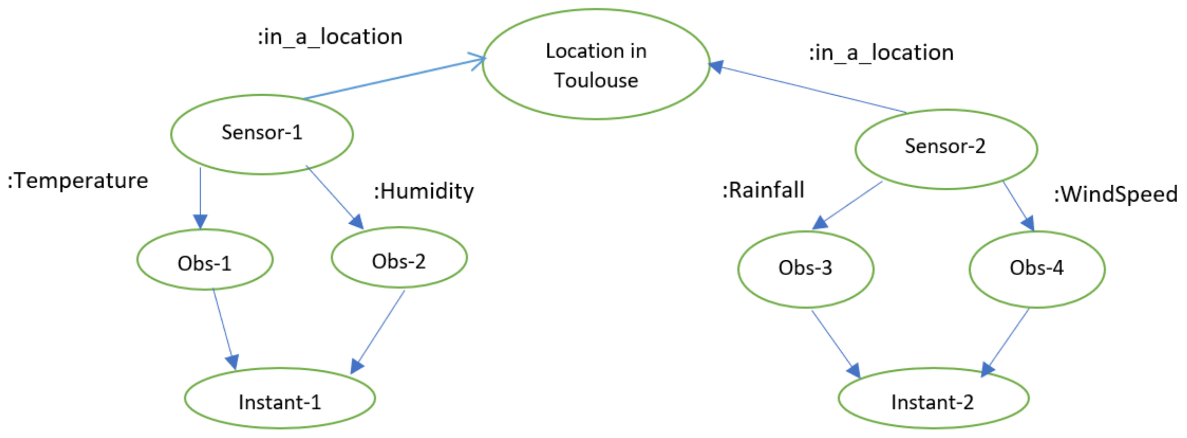
I attempted to build the ontology design based on the 'triple'.

Subject - Place

- City <- Toulouse
- Country <- France

Property - Measurable parameters as Concept class

- Temperature
- Humidity
- Rainfall
- Atmospheric pressure
- Wind Speed/Force



Object - Properties (relations) of Measurable parameters

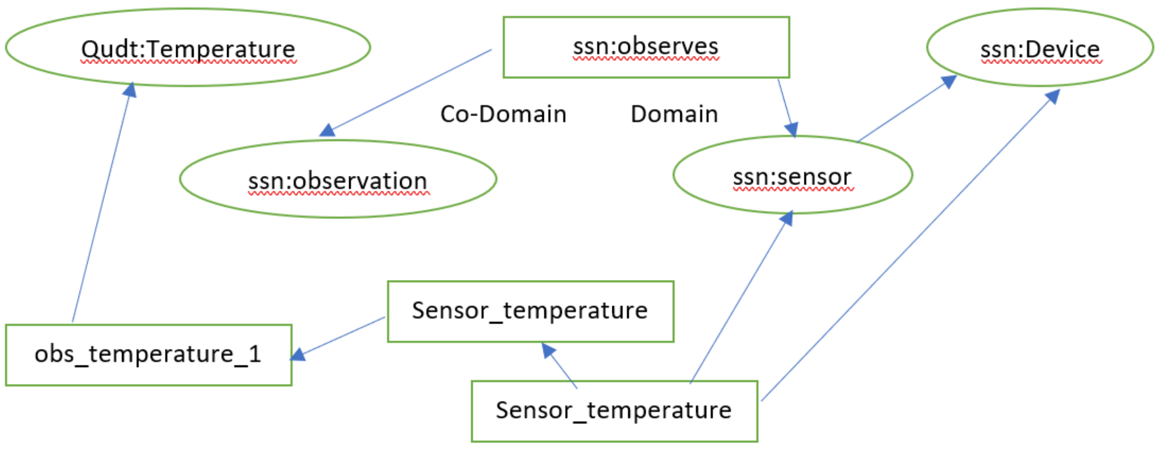
- has_duration_min
- has_moment
- has_dateTimeStamp
- has_temperature_value
- has_humidity_value
- has_rainfall_value

- has_atmospheric pressure_value
- has_wind speed_value

Class Design in Protégé

- Domain class
  - Place class
  - Measurable parameters class
- Value class
  - DataTimeStamp
  - Temperature
  - Humidity
  - Rainfall
  - Atmospheric pressure

Ontology - Hierarchical Inference



3 Conversion of 3-star data to 5-star data

In this TP-2 exercise, I attempted to understand how the temperature related 3-star 'data_set' of Aarhus city (Denmark) can be manipulated and converted to 5-star data.

3-star data

- Data is available in the web
- Data is structured
- Data is in free-format

4-star data

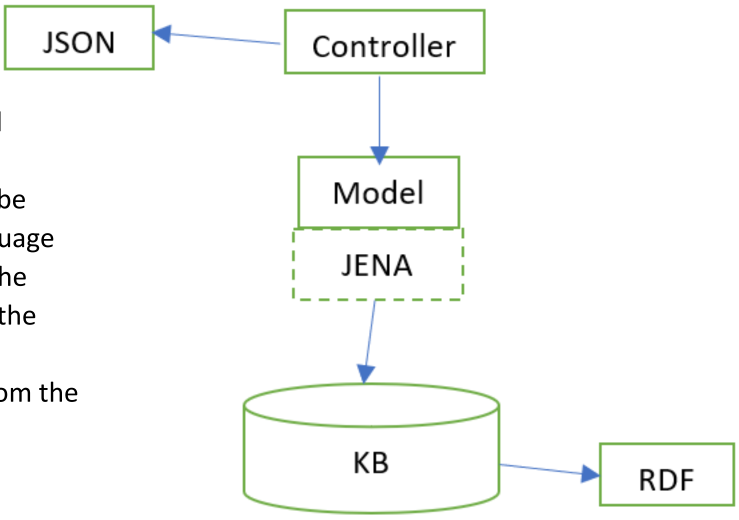
- 3-star data in addition to compliance to the W3C standards
- Means that, the 'subject', 'property' and 'object' are linked for example using RDF graph and manipulated using SPARQL language

5-star data

- 4-star data which can be linked to other data
- For example, the sensor data (4-star) linked to its observation value

In this task, a wrapper tool written in Java is used to manipulate the Ontology model and convert to the 5-star data.

- ICOntrolFunctions and IModelFunctions are the 2 Java interfaces files implemented as wrapper using Apache JENA.
- As mentioned earlier, a 3-star dataset can be Converted to 4-star using the SPARQL language
- The SPARQL queries which are defined in the IConvenientInterface are implemented by the IModelFunction
- Followed by, the IControlFunctions uses from the Model to enrich the dataset to 5-star



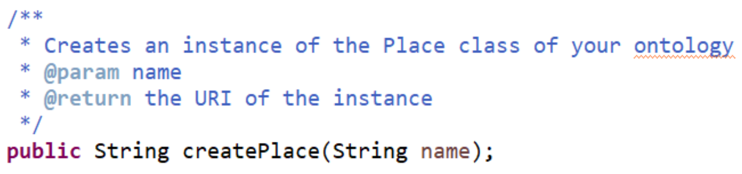
DoItYourselfModel Implementation

I studied the implementation of the functions in the DoItYourselfModel stubbed class and understood how they use the IConvenienceInterface and conduct the knowledge-based related operations. I present the understanding of each of the function,

Step-1

Creating the place – Toulouse city, France.

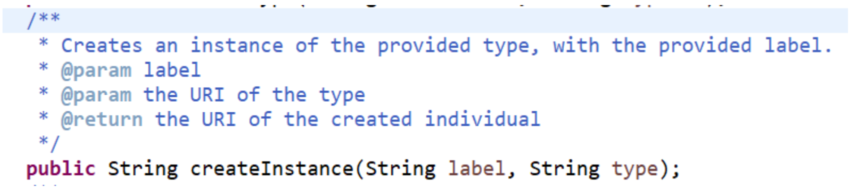
As per the 'Meteorology' application, the monitoring is made for 'Toulouse' city. Using this function, we call the createInstance to create the instance for 'Toulouse' city, France.



```
/**
 * Creates an instance of the Place class of your ontology
 * @param name
 * @return the URI of the instance
 */
public String createPlace(String name);
```

createInstance()

Here, the SPARQL query acts on the knowledge base to create an instance for the 'Toulouse' city and instance is returned.



```
/**
 * Creates an instance of the provided type, with the provided label.
 * @param label
 * @param the URI of the type
 * @return the URI of the created individual
 */
public String createInstance(String label, String type);
```

Step-2

In this step, as I understand, the 3-star data gets converted to 4-star data.

This is done, by linking the 'subject' (Toulouse city), 'Property' (tempertature sensor, humidity sensor etc.) and the 'Object' (timestamp).

createInstant(TimestampEntity instant)

This function is used to create an instance of the "Instant" class of the ontology & link the timestamp to it. This is to demonstrate how the 'data property' can be added to the 'Toulouse' city (Subject) and the property (Measurable parameters).

The property classes are,

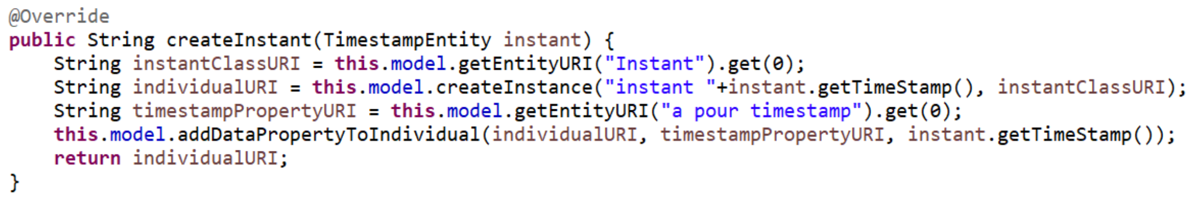
- Temperature

- Humidity
- Rainfall
- Atmospheric pressure
- Wind Speed/Force

Therefore, in this function, as seen below, the newly created instance of the 'property' class is linked to the 'data property' using the function ‘addDataPropertyToIndividual’.

* the data property represents the timestamp

* Only one instance is created for each actual timestamp.



```
@Override
public String createInstant(TimestampEntity instant) {
    String instantClassURI = this.model.getEntityURI("Instant").get(0);
    String individualURI = this.model.createInstance("instant "+instant.getTimeStamp(), instantClassURI);
    String timestampPropertyURI = this.model.getEntityURI("a pour timestamp").get(0);
    this.model.addDataPropertyToIndividual(individualURI, timestampPropertyURI, instant.getTimeStamp());
    return individualURI;
}
```

This way, the ‘timeStamp’ instances of each of the sensors can be manipulated in the knowledge base.

Step-3

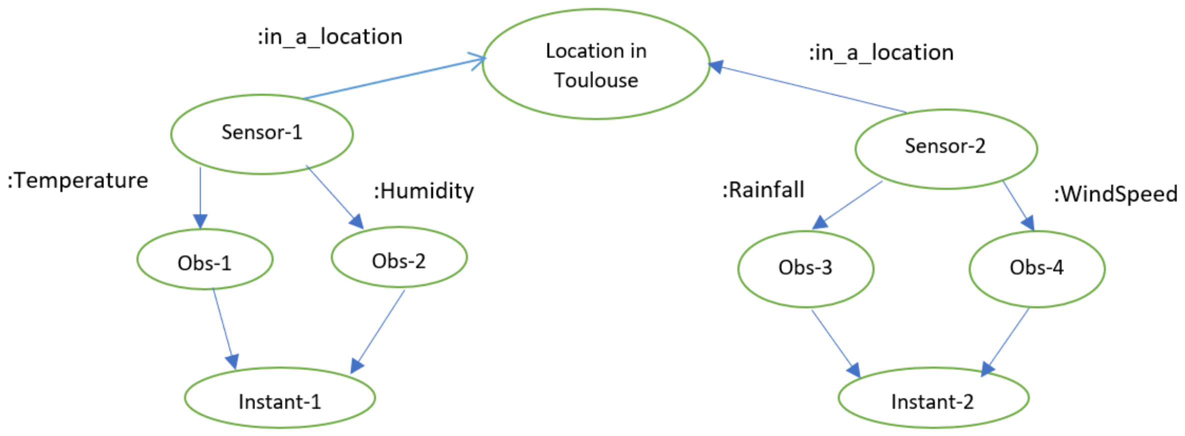
In this step, as I understand, the 4-star dataset gets converted to 5-star data.

This is done, by creating the ‘observation’ relation and link it with the 4-star dataset.

String createObs(String value, String paramURI, String instantURI);

The above function creates the observation/relation link between the given ‘property’ (measurable parameter) and the ‘object’/value (example- timestamp).

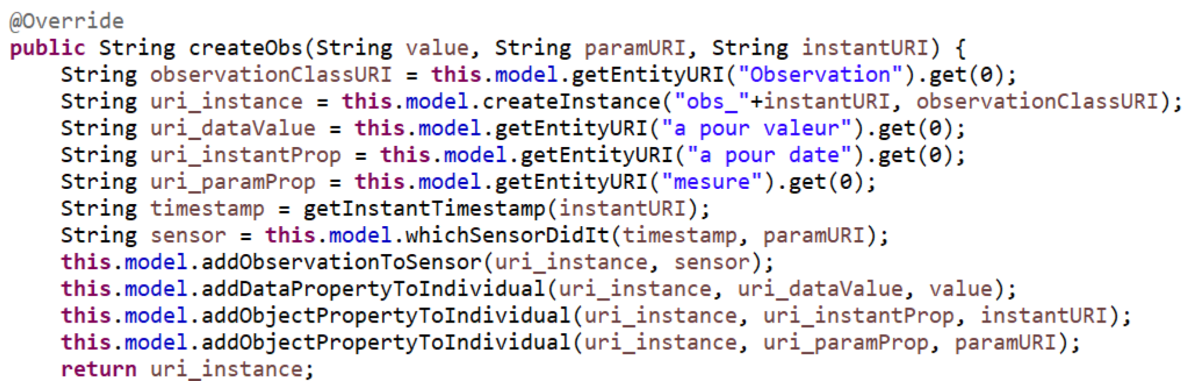
The function returns the URI for the newly created observation link which links the 2 different data – sensor and its value. Therefore, the new dataset can be called as 5-star data.



As per the above figure, in this step, using the createObs(), we create the Obs-1/Obs-2/Obs-3/Obs-4 relations/Links between the Instant-1/Instant-2 and the Sensor-1/Sensor-2.

As can be find from below function, using the ‘add’ functions, below 3 items are linked.

- Data Property – Example timestamp
- Object Property – example, the measurable parameters such as temperature, humidity
- Observation – the link between the data property and object property



```
@Override
public String createObs(String value, String paramURI, String instantURI) {
    String observationClassURI = this.model.getEntityURI("Observation").get(0);
    String uri_instance = this.model.createInstance("obs_"+instantURI, observationClassURI);
    String uri_dataValue = this.model.getEntityURI("a pour valeur").get(0);
    String uri_instantProp = this.model.getEntityURI("a pour date").get(0);
    String uri_paramProp = this.model.getEntityURI("mesure").get(0);
    String timestamp = getInstantTimestamp(instantURI);
    String sensor = this.model.whichSensorDidIt(timestamp, paramURI);
    this.model.addObservationToSensor(uri_instance, sensor);
    this.model.addDataPropertyToIndividual(uri_instance, uri_dataValue, value);
    this.model.addObjectPropertyToIndividual(uri_instance, uri_instantProp, instantURI);
    this.model.addObjectPropertyToIndividual(uri_instance, uri_paramProp, paramURI);
    return uri_instance;
```

Thus, we will be able to retrieve the timestamp instants/value data for temperature/humidity and rainfall/windspeed sensors. This makes the dataset a complete 5-star.