Partie II Programmation avec Jena

Télécharger Jena

1. Récupérer la librairie Jena à partir de Moodle
2. « Dezziper » Jena par la commande : **unzip jena-2.6.2.zip**

Ecrire un programme en Java permettant de faire de l'inférence et d'écrire des requêtes avec OWL

1. Lancer Eclipse
2. Créer un projet « Java Project »
3. Créer un dossier **data** dans le projet (Right click -> New -> Folder)
4. Créer le fichier **owlrules.txt** dans le répertoire **data** avec le contenu décrit ci-dessous
5. Créer le fichier **rules.txt** dans le répertoire **data** avec le contenu décrit ci-dessous
6. Créer le fichier **query.txt** dans le répertoire **data** avec le contenu décrit ci-dessous
7. Copier puis coller le fichier **family.owl** que vous avez crée dans le répertoire **data**
8. Créer un package **itsudparis.tools** dans le répertoire **src** dans le projet
9. Ouvrir le menu **Projet/Properties/Java Build Path.** Dans la partie **Libraries**, Cliquer sur **Add External JARS**. Ajouter *toutes les librairies de Jena* (les librairies de Jena se trouvent dans le lib de Jena).
10. Créer une classe **FileTool** avec le contenu décrit ci-dessous dans le package **itsudparis.tools**
11. Exécuter une classe **JenaEngine** avec le contenu décrit ci-dessous dans le package **itsudparis.tools**
12. Créer un package **itsudparis.application**
13. Créer une class **Main** avec le contenu décrit ci-dessous dans le package **itsudparis.application**
14. Exécuter la classe **Main**

Code

Fichier FileTool.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package itsudparis.tools;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;

/**
 *
 * @author DO.IT-SUDPARIS
```

```
 * Ce fichier sert à
 * + Lire le contenu d'un fichier: getContents
 * + Entrée: Objet File
 * + Sortie: une chaine de caractère
 */
public class FileTool {

    static public String getContents(File aFile) {
        //...checks on aFile are elided
        StringBuilder contents = new StringBuilder();

        try {
            //use buffering, reading one line at a time
            //FileReader always assumes default encoding is OK!
            BufferedReader input = new BufferedReader(new
FileReader(aFile));
            try {
                String line = null; //not declared within while loop
                        /*
                 * readLine is a bit quirky :
                 * it returns the content of a line MINUS the newline.
                 * it returns null only for the END of the stream.
                 * it returns an empty String if two newlines appear in a
row.
                 */
                while ((line = input.readLine()) != null) {
                    contents.append(line);
                    contents.append(System.getProperty("line.separator"));
                }
            } finally {
                input.close();
            }
        } catch (IOException ex) {
            ex.printStackTrace();
        }

        return contents.toString();
    }
}
```

Fichier JenaEngine.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package itsudparis.tools;

import java.io.File;
import java.io.InputStream;
import java.util.List;

import com.hp.hpl.jena.query.Query;
import com.hp.hpl.jena.query.QueryExecution;
```

```java
import com.hp.hpl.jena.query.QueryExecutionFactory;
import com.hp.hpl.jena.query.QueryFactory;
import com.hp.hpl.jena.query.ResultSet;
import com.hp.hpl.jena.query.ResultSetFormatter;
import com.hp.hpl.jena.rdf.model.*;
import com.hp.hpl.jena.reasoner.rulesys.GenericRuleReasoner;
import com.hp.hpl.jena.reasoner.rulesys.Rule;
import com.hp.hpl.jena.util.FileManager;
import java.io.IOException;
import java.io.OutputStream;

/**
 *
 * @author DO.ITSUDPARIS
 */
public class JenaEngine {
    static private String RDF = "http://www.w3.org/1999/02/22-rdf-
syntax-ns#";


    /**
     * Charger un modèle à partir d'un fichier owl
     * @param args
     * + Entree: le chemin vers le fichier owl
     * + Sortie: l'objet model jena
     */
    static public Model readModel(String inputDataFile) {
//          create an empty model
        Model model = ModelFactory.createDefaultModel();



        // use the FileManager to find the input file
        InputStream in = FileManager.get().open(inputDataFile);
        if (in == null) {
            System.out.println("Ontology file: " + inputDataFile + "
not found");
            return null;
        }

        // read the RDF/XML file
        model.read(in, "");
        try {
            in.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            return null;
```

```
        }
        return model;
    }
    /**
     * Faire l'inference
     * @param args
     * + Entree: l'objet model Jena avec le chemin du fichier de
regles
     * + Sortie: l'objet model infere Jena
     */
    static public Model readInferencedModelFromRuleFile(Model model,
String inputRuleFile) {
        InputStream in = FileManager.get().open(inputRuleFile);
        if (in == null) {
            System.out.println("Rule File: " + inputRuleFile + " not
found");
            return null;
        } else {
            try {
                in.close();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                return null;
            }
        }

        List rules = Rule.rulesFromURL(inputRuleFile);
        GenericRuleReasoner reasoner = new
GenericRuleReasoner(rules);
        reasoner.setDerivationLogging(true);
        reasoner.setOWLTranslation(true);               // not
needed in RDFS case
        reasoner.setTransitiveClosureCaching(true);
        InfModel inf = ModelFactory.createInfModel(reasoner, model);
        return inf;
    }


    /**
     * Executer une requete
     * @param args
     * + Entree: l'objet model Jena avec une chaine des caracteres
SparQL
     * + Sortie: le resultat de la requete en String
     */
    static public String executeQuery(Model model, String
queryString) {
```

```
        Query query = QueryFactory.create(queryString);
        // No reasoning
        // Execute the query and obtain results
        QueryExecution qe = QueryExecutionFactory.create(query,
model);
        ResultSet results = qe.execSelect();

        OutputStream output = new OutputStream() {

            private StringBuilder string = new StringBuilder();

            @Override
            public void write(int b) throws IOException {
                this.string.append((char) b);
            }

            //Netbeans IDE automatically overrides this toString()
            public String toString() {
                return this.string.toString();
            }
        };

        ResultSetFormatter.out(output, results, query);
        return output.toString();
    }
    /**
     * Executer un fichier d'une requete
     * @param args
     * + Entree: l'objet model Jena avec une chaine des caracteres
SparQL
     * + Sortie: le resultat de la requete en String
     */
    static public String executeQueryFile(Model model, String
filepath) {
        File queryFile = new File(filepath);
        // use the FileManager to find the input file
        InputStream in = FileManager.get().open(filepath);
        if (in == null) {
            System.out.println("Query file: " + filepath + " not
found");
            return null;
        } else {
            try {
                in.close();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                return null;
```

```
            }
        }
        String queryString = FileTool.getContents(queryFile);
        return executeQuery(model, queryString);
    }
    /**
     * Executer un fichier d'une requete avec le parametre
     * @param args
     * + Entree: l'objet model Jena avec une chaine des caracteres
SparQL
     * + Sortie: le resultat de la requete en String
     */
    static public String executeQueryFileWithParameter(Model model,
String filepath, String parameter) {
        File queryFile = new File(filepath);
        // use the FileManager to find the input file
        InputStream in = FileManager.get().open(filepath);
        if (in == null) {
            System.out.println("Query file: " + filepath + " not
found");
            return null;
        } else {
            try {
                in.close();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                return null;
            }
        }
        String queryString = FileTool.getContents(queryFile);
        queryString = queryString.replace("%PARAMETER%", parameter);
        return executeQuery(model, queryString);
    }
     /**
     * Creer une Instance
     * @param args
     * + Entree:
     *      - l'objet model Jena
     *      - Namespace de l'ontologie
     *      - Le nom de le classe
     *      - Le nom de l'instance
     * + Sortie: le resultat de la requete en String
     */
    static public boolean createInstanceOfClass(Model model, String
namespace, String className, String instanceName) {
        Resource rs = model.getResource(namespace + instanceName);
        if (rs == null)
```

```
            rs = model.createResource(namespace+instanceName);
        Property p = model.getProperty(RDF + "type");
        Resource rs2 = model.getResource(namespace + className);
        if ((rs2 != null)&&(rs != null) && (p != null)) {
            //add new value
            rs.addProperty(p,rs2);
            return true;
        }
        return false;
    }
    /**
     * Mettre a jour la valeur d'une propriete objet d'une instance
     * @param args
     * + Entree:
     *      - l'objet model Jena
     *      - Namespace de l'ontologie
     *      - Le nom de la première Instance
     *      - Le nom de la propriete
     *      - Le nom de la deuxieme Instance
     * + Sortie: le resultat de la requete en String
     */
    static public boolean updateValueOfObjectProperty(Model model,
String namespace, String object1Name, String propertyName, String
object2Name) {
        Resource rs1 = model.getResource(namespace + object1Name);
        Resource rs2 = model.getResource(namespace + object2Name);
        Property p = model.getProperty(namespace + propertyName);
        if ((rs1 != null) && (rs2 != null) && (p != null)) {
            //remove all old values of property p
            rs1.removeAll(p);
            //add new value
            rs1.addProperty(p,rs2);
            return true;
        }
        return false;
    }
    /**
     * Mettre a jour la valeur d'une propriete objet d'une Instance
     * @param args
     * + Entree:
     *      - l'objet model Jena
     *      - Namespace de l'ontologie
     *      - Le nom de la premiere Instance
     *      - Le nom de la propriete
     *      - Le nom de le deuxieme Instance
     * + Sortie: le resultat de la requete en String
     */
```

```
    static public boolean addValueOfObjectProperty(Model model,
String namespace, String instance1Name, String propertyName, String
instance2Name) {
        Resource rs1 = model.getResource(namespace + instance1Name);
        Resource rs2 = model.getResource(namespace + instance2Name);
        Property p = model.getProperty(namespace + propertyName);
        if ((rs1 != null) && (rs2 != null) && (p != null)) {
            //add new value
            rs1.addProperty(p,rs2);
            return true;
        }
        return false;
    }


    /**
     * Mettre a jour la valeur d'une propriete datatype d'une
Instance
     * @param args
     * + Entree:
     *      - l'objet model Jena
     *      - Namespace de l'ontologie
     *      - Le nom de l'Instance
     *      - Le nom de la propriete
     *      - La nouvelle valeur
     * + Sortie: le resultat de la requete en String
     */
    static public boolean updateValueOfDataTypeProperty(Model model,
String namespace, String instanceName, String propertyName, Object
value) {
        Resource rs = model.getResource(namespace + instanceName);
        Property p = model.getProperty(namespace + propertyName);
        if ((rs != null) && (p != null)) {
            //remove all old values of property p
            rs.removeAll(p);
            //add new value
            rs.addLiteral(p, value);
            return true;
        }
        return false;
    }
   /**
     * Ajouter la valeur d'une propriete datatype d'une Instance
     * @param args
     * + Entree:
     *      - l'objet model Jena
     *      - Namespace de l'ontologie
```

```
     *      - Le nom de l'Instance
     *      - Le nom de la propriete
     *      - La valeur
     * + Sortie: le resultat de la requete en String
     */
    static public boolean addValueOfDataTypeProperty(Model model,
String namespace, String instanceName, String propertyName, Object
value) {
        Resource rs = model.getResource(namespace + instanceName);
        Property p = model.getProperty(namespace + propertyName);
        if ((rs != null) && (p != null)) {
            //add new value
            rs.addLiteral(p, value);
            return true;
        }
        return false;
    }
    /**
     * Supprimer toutes les valeurs d'une propriete d'une Instance
     * @param args
     * + Entree:
     *      - l'objet model Jena
     *      - Namespace de l'ontologie
     *      - Le nom de l'Instance
     *      - Le nom de la propriete
     * + Sortie: le resultat de la requete en String
     */
    static public boolean removeAllValuesOfProperty(Model model,
String namespace, String objectName, String propertyName) {
        Resource rs = model.getResource(namespace + objectName);
        Property p = model.getProperty(namespace + propertyName);
        if ((rs != null) && (p != null)) {
            //remove all old values of property p
            rs.removeAll(p);
            //add new value
            return true;
        }
        return false;
    }


}
```

Fichier Main.java

```
/*
 * To change this template, choose Tools | Templates
```

```
 * and open the template in the editor.
 */

package itsudparis.application;

import com.hp.hpl.jena.rdf.model.Model;
import itsudparis.tools.JenaEngine;

/**
 * @author DO.ITSUDPARIS
 */
public class Main {
	/**
	 * @param args
	 *            the command line arguments
	 */
	public static void main(String[] args) {
		String NS = "";
		// lire le model a partir d'une ontologie
		Model model = JenaEngine.readModel("data/family.owl");
		if (model != null) {
			//lire le Namespace de l'ontologie
			NS = model.getNsPrefixURI("");
			// modifier le model
			// Ajouter une nouvelle femme dans le modele: Nora,
50, estFilleDe Peter
			JenaEngine.createInstanceOfClass(model, NS, "Femme",
"Nora");
			JenaEngine.updateValueOfDataTypeProperty(model, NS,
"Nora", "age", 50);
			JenaEngine.updateValueOfObjectProperty(model, NS,
"Nora", "estFilleDe", "Peter");

			// Ajouter un nouvel homme dans le modele: Rob, 51,
seMarierAvec Nora
			JenaEngine.createInstanceOfClass(model, NS, "Homme",
"Rob");
			JenaEngine.updateValueOfDataTypeProperty(model, NS,
"Rob", "age", 51);
			JenaEngine.updateValueOfDataTypeProperty(model, NS,
"Rob", "nom", "Rob Yeung");
			JenaEngine.updateValueOfObjectProperty(model, NS,
"Rob", "seMarierAvec", "Nora");

			//apply owl rules on the model
			Model owlInferencedModel =
JenaEngine.readInferencedModelFromRuleFile(model,
```

```
"data/owlrules.txt");
                // apply our rules on the owlInferencedModel
                Model inferedModel =
JenaEngine.readInferencedModelFromRuleFile(owlInferencedModel,
"data/rules.txt");
                // query on the model after inference

     System.out.println(JenaEngine.executeQueryFile(inferedModel,
                          "data/query.txt"));
           } else {
                System.out.println("Error when reading model from
ontology");
           }
     }
}
```

Fichier owlrules.txt

```
@include <OWLMicro>.
```

Fichier ourrules.txt

```
@prefix ns: <http://www.it-sudparis.eu/family#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.

[rule1: (?per rdf:type ns:Personne)  (?per ns:age ?age)
greaterThan(?age, 60)-> (?per rdf:type ns:PersonneAge)]
```

Fichier query.txt

```
PREFIX ns: <http://www.it-sudparis.eu/family#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?user ?age
WHERE {
        ?user rdf:type ns:PersonneAge.
        ?user ns:age ?age.
}
```

Exercice

1. Créer plusieurs fichiers requêtes pour les questions de la partie I. Exécuter les avec votre programme et afficher le résultat sur la console.
2. Construire une application dans le package itsudparis.application qui permet de :
    a. Recevoir un nom à partir de la console

b. Afficher tous les parents, tous les frères ou sœurs s'ils existent.
c. Si cette personne est mariée, afficher le nom, l'âge de son conjoint
d. Retourner à l'étape **a** pour saisir le nom d'une autre personne.