Домашно бр. 4
Consuming Linked Data
Константин Богданоски, 161048

А) Вовед

Покрај тоа што DBpedia (http://www.dbpedia.org) своите податоци ги прави достапни преку нивниот SPARQL Endpoint (http://dbpedia.org/sparql), тие се достапни и преку HTTP content negotiation.

Тоа претставува функционалност која врз база на деталите во корисничкото барање испратено до URI-то на DBpedia ресурсот (пр.: http://dbpedia.org/resource/Skopje), корисникот се пренасочува кон друго URL кое ги содржи деталите за ресурсот во бараниот формат (HTML, RDF/XML, Turtle, JSON-LD, …).

Токму како резултат на ваквиот content negotiation DBpedia знае да го пренасочи вашето барање од http://dbpedia.org/**resource**/Skopje кон http://dbpedia.org/**page**/Skopje, каде ви се прикажуваат деталите за ресурсот во HTML формат. Причина за ова е типот на барање кој го испраќа вашиот прелистувач, барајќи HTML содржина.

Доколку би го побарале истиот ресурс за Skopje со барање на содржината во некој друг формат, патеките кон кои ќе ве упати DBpedia се следниве:

- RDF/XML: http://dbpedia.org/**data**/Skopje**.xml**
- Turtle: http://dbpedia.org/**data**/Skopje**.ttl**
- N3: http://dbpedia.org/**data**/Skopje**.n3**
- JSON-LD: http://dbpedia.org/**data**/Skopje**.json**

Дури и без користење на content negotiation, можеме да ги добиеме овие URL-а за секој DBpedia ресурс кој нѝ е од интерес.

Б) Практична задача

Како практична задача, ќе треба да креирате Java + Jena проект кој ќе ги вчита RDF тројките за даден DBpedia ресурс (преку RDF фајлот на ресурсот, во RDF/XML или Turtle, на пример), потоа од нив ќе прочита одредени релации (по ваш избор) кои водат до други DBpedia ресурси и ќе ги вчита и нивните RDF тројки. Преку овој втор DBpedia ресурс, потоа ќе се лоцираат нови DBpedia ресурси, ќе се вчитаат нивните RDF тројки и ќе се испишат некои податоци на екран.

Со ова успеваме да конзумираме поврзани податоци (Linked Data) преку 3 чекори.

Пример: Доколку започнете од ресурсот за *градот Skopje*, можете да детектирате некој ресурс кој претставува личност која е *родена во* градот. Откако ќе ги вчитате податоците за таа личност, можете да најдете линкови кон ресурси како што се нивни *дела*, на пример (песна, албум, филм, итн.). Откако ќе ги вчитате RDF тројките за тоа дело, ќе можете дел од нив да отпечатите на конзола: *датум на издавање*, *наслов*, *жанр*, *слика*, итн.

I. Работа со DBpedia податоци без користење на SPARQL Endpoint

1. Одберете почетен DBpedia ресурс по ваш избор. Тоа може да биде некој град, држава, позната личност, филм, албум, музичка група, настан, итн.
   Red Hot Chilli Peppers (http://dbpedia.org/resource/Red_Hot_Chili_Peppers)

2. Креирајте нов Java проект во NetBeans. Вклучете ги во проектот сите .jar библиотеки од lib фолдерот од Jena. Jena преземете ја директно од Jena сајтот.

3. Во main() методот на главната класа од проектот, креирајте основен Jena model, кој ќе го содржи RDF графот за DBpedia ресурсот кој ќе го вчитате.

4. Со користење на read() методата на моделот, вчитајте ги RDF тројките за DBpedia ресурсот кој сте го одбрале. Внимавајте да го наведете форматот на датотеката која ја вчитувате.
   Напомена: За вчитување искористете некој од RDF фајловите кои DBpedia ги нуди за ресурсот: http://dbpedia.org/**data**/*ресурс.наставка*
   Алтернатива: Користејќи го пример кодот од аудиториските вежби, направете content negotiation со експлицитно дефинирање на Accept хедерот, за да DBpedia автоматски ви ги даде податоците во бараниот формат. Притоа, HTTP барањето ќе треба да го испратите до http://dbpedia.org/**resourse**/*ресурс*

```
model.read("http://dbpedia.org/data/Red_Hot_Chili_Peppers", "RDF/XML");
model.write(System.out, "TURTLE");
```

5. Разгледајте ги деталите за ресурсот на DBpedia. Одберете некоја релација која како вредност има друг DBpedia ресурс, а потоа селектирајте ја и отпечатете ја таа вредност од вчитаниот RDF граф за ресурсот со користење на Jena.

```
<http://dbpedia.org/resource/Flea_(musician)>
dbo:associatedBand     <http://dbpedia.org/resource/Red_Hot_Chili_Peppers> ;
dbo:associatedMusicalArtist<http://dbpedia.org/resource/Red_Hot_Chili_Peppers> .
```

6. Извршете ја програмата.

```
String s1 = "https://www.w3.org/2012/pyMicrodata/extract?format=turtle&uri=http%3A%2F
%2Fdbpedia.org%2Fresource%2FRed_Hot_Chili_Peppers";
String s2 = "https://www.w3.org/2012/pyRdfa/extract?format=turtle&uri=http%3A%2F
%2Fdbpedia.org%2Fresource%2FRed_Hot_Chili_Peppers";

Model m1 = ModelFactory.createDefaultModel();
m1.read(s1, "TURTLE");
Model m2 = ModelFactory.createDefaultModel();
m2.read(s2, "TURTLE");
Model m = m1.union(m2);
System.out.println("================================================");
System.out.println("Printing Red Hot Chili Peppers RDF Graph - Turtle");
System.out.println("================================================");
m.write(System.out, "TURTLE");

//Printing resource Dark Necessities
System.out.println(m.getResource("http://dbpedia.org/resource/Dark_Necessities"));
```

7. Заменете го печатењето од претходната точка со креирање на нов модел во кој ќе го вчитате целиот RDF граф за новиот DBpedia ресурс.
   Напомена: За да го добиете URL-то кое ги содржи RDF графот за ресурсот во одреден формат, ќе треба да го модифицирате URI-то кое го имате за ресурсот.

```
System.out.println("===============================================");
System.out.println("Printing Dark Necessities RDF Graph - Turtle");
System.out.println("===============================================");
Resource darkN = m.getResource("http://dbpedia.org/resource/Dark_Necessities");
Model m3 = ModelFactory.createDefaultModel();
m3.read(darkN.getURI());
m3.write(System.out, "TURTLE");
```

8. Разгледајте ги деталите за новиот ресурс на DBpedia. Повторно одберете некоја релација која како вредност има друг DBpedia ресурс, а потоа селектирајте ја и отпечатете ја таа вредност од вчитаниот RDF граф за ресурсот со користење на Jena.

```
Resource brendansDeathSong = m3.getResource("http://dbpedia.org/resource/Brendan\
u0027s_Death_Song");
System.out.println(brendansDeathSong);
```

9. Извршете ја програмата.

10. Заменете го печатењето од претходната точка со креирање на нов, трет модел, во кој ќе го вчитате целиот RDF граф за новиот, третиот DBpedia ресурс.

```
Model m4 = ModelFactory.createDefaultModel();
m4.read(brendansDeathSong.getURI());
m4.write(System.out, "TURTLE");
```

11. Откако ќе го вчитате RDF графот, отпечатете на конзола одреден број факти по ваш избор, кои ќе го опишат овој ресурс. Притоа, користете ја DBpedia страната за ресурсот за да одберете својства кои мислите дека би биле корисни да се отпечатат.

```
Property mainName = m4.getProperty("foaf:name");
//Loop name
m4.listResourcesWithProperty(mainName).forEachRemaining(System.out::println);

Property writer = m4.getProperty("dbo:writer");
//Loop that will print all the writers of the song
m4.listResourcesWithProperty(writer).forEachRemaining(System.out::println);

Property recorded = m4.getProperty("dbp:recorded");
//Loop that will print all values for the property: dbo:recorded
m4.listResourcesWithProperty(recorded).forEachRemaining(System.out::println);
```

12. Извршете ја програмата.

II. Извлекување поврзани податоци за лекови стартувајќи од локален RDF граф

Во овој дел од вежбата ќе работиме со податоците за лекови од првата лабораториска вежба, во која имавме и линкови кон лекови кои се наоѓаат во друго податочно множество. Ваша задача ќе биде да ги добиете податоците (RDF тројките) за овие „надворешни“ лекови со кои се поврзани лековите од нашето податочно множество, и да отпечатите дел од нивните својства.

---

13. Креирајте нова Java класа во проектот, во која ќе додадете и main() метод. Преземете ја датотеката “hifm-dataset.ttl” од Courses и снимете ја локално. Ова е датотеката со истите податоци за лекови како и на првата лабораториска вежба.

---

14. Во main() методот на новата класа напишете код со кој ќе ја прочитате содржината на оваа датотека. Внимавајте третиот параметар на model.read() да го поставите за вчитување на Turtle содржина.

```
BasicConfigurator.configure();
Model model = ModelFactory.createDefaultModel();
String file = "/home/konstantin/Desktop/FCSE/WBS/Labs/Homeworks/hifm-dataset.ttl";

model.read(file, "TURTLE");
model.write(System.out, "TURTLE");
```

15. Потсетете се на содржината на “hifm-dataset.ttl” датотеката. Станува збор за податочно множество кое содржи лекови од Фондот за здравство на РМ. За секој од лековите имаме повеќе различни релации (име, шифра, производувач, пакување, ...), но линк кон лек од DrugBank податочното множество (rdfs:seeAlso).

16. Одберете еден лек од графот (моделот) и за него излистајте ги и лековите кои ја имаат истата функција како и тој, а се наоѓаат во DrugBank податочното множество (rdfs:seeAlso). Извршете ја програмата.

```
public class JenaConsumeLinkedDataSparql {

    static String SPARQLEndpoint = "http://wifo5-04.informatik.uni-mannheim.de/drugbank/sparql";

    public static void main (String args[]) {

            String queryString = "SELECT * "
                    + "WHERE { "
                    + "?drug <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> "
                    + "<http://wifo5-04.informatik.uni-mannheim.de/drugbank/resource/drugbank/drugs> ; "
                    + "<http://www.w3.org/2000/01/rdf-schema#label> ?label "
                    + "}";

            System.out.println("Using SPARQL query: " + queryString);

            Query query = QueryFactory.create(queryString) ;
            try (QueryExecution qexec = QueryExecutionFactory.sparqlService(SPARQLEndpoint, query)) {
                ResultSet results = qexec.execSelect() ;
                for ( ; results.hasNext() ; )
                {
                  QuerySolution soln = results.nextSolution();
                  System.out.println("Drug: " + soln.get("drug").toString());
                  System.out.println("Label: " + soln.get("label").toString());
                }
            }

    }
}
```

Слика 1. Пример за користење SPARQL endpoint преку Jena.

17. Направете промена, со тоа што наместо да ги печатите, ќе ги вчитате RDF графовите за секој од DrugBank лековите со кои вашиот одбран лек е во релација. За секој од нив, ќе го отпечатите името (rdfs:label), името под кое се продава (drugbank:brandName), генеричкото име (drugbank:genericName) и фармаколошкото дејство на лекот (drugbank:pharmacology). Извршете ја програмата.

    Забелешка: Доколку DrugBank сервисот не е достапен во моментот, пробајте малку подоцна. Сервисот има активна скрипта која го рестартира и активира кога ќе „падне“.

    Напомена: За пристап до RDF граф на даден DrugBank лек имате две опции:

    (а) да направите промена на URI-то во URL кое ќе ви даде директен пристап до RDF тројките (како што правевме во првиот дел од вежбата со DBpedia), или

    (б) да поставите класично SPARQL прашање на SPARQL Endpoint-от на DrugBank: http://wifo5-04.informatik.uni-mannheim.de/drugbank/sparql.

    За опцијата под (а) потребно е од URI-то на лекот да направите промена на /page/ делот во /data/ и добивате пристап до Turtle верзија на податоците за лекот. На пример, за лекот http://wifo5-04.informatik.uni-mannheim.de/drugbank/**page**/drugs/DB00007, RDF податоците во Turtle формат се достапни на http://wifo5-04.informatik.uni-mannheim.de/drugbank/**data**/drugs/DB00007.

    За опцијата под (б) можете да употребите Java + Jena код како на *Слика 1*. Примерот од сликата илустрира како во Jena може да се креира SPARQL прашање кое потоа ќе се испрати до SPARQL endpoint. Во овој случај искористете го SPARQL прашањето за директно да ги добиете вредностите на својствата кои ве интересираат, наместо да го селектирате целиот граф за лекот.

```
String queryString =  "SELECT * " +
                "WHERE { " +
                "?drug rdfs:label \"Cefuroxime\"; " +
                "rdfs:label ?label. " +
                "} ";
Query query = QueryFactory.create(queryString);
try (QueryExecution queryExecution = new QueryExecutionFactory.sparqlService(SparqlEndpoint,
query)) {
    ResultSet resultSet = queryExecution.execSelect();
    while (resultSet.hasNext()){
        QuerySolution querySolution = resultSet.nextSolution();
        System.out.println("==========================================");
        System.out.println("Drug: " + querySolution.get("drug").toString());
        System.out.println("Label: " + querySolution.get("label").toString());
    }
}
```

Добивав грешка дека `sparqlService()` не постои како метод, иако постоеше и знаеше IntelliJ каде се наоѓа. Овој bug постои во IntelliJ и сеуште не е поправен, по долго дебагирање.