Домашно бр. 4

Consuming Linked Data

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.

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*
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.
6. Извршете ја програмата.

```java
public class Task1 {

    private static final String dbo = "http://dbpedia.org/ontology/";
    private static final String dbr = "http://dbpedia.org/resource/";
    private static final String dbData = "http://dbpedia.org/data/";

    public static void main(String[] args) throws IOException {

        Model startingPoint = ModelFactory.createDefaultModel();
        startingPoint.read(String.format("%s%s", dbData, "Skopje.ttl"), lang: "TURTLE");

        Resource skopje = startingPoint.getResource(String.format("%s%s", dbr, "Skopje"));
        Property coutry = startingPoint.getProperty(String.format("%s%s", dbo, "country"));
        Resource macedonia = (Resource) skopje.getProperty(coutry).getObject();
        System.out.println(macedonia);

    }

}
```

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.
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.
9. Извршете ја програмата.

```java
public class Task1 {

    private static final String dbo = "http://dbpedia.org/ontology/";
    private static final String dbr = "http://dbpedia.org/resource/";
    private static final String dbData = "http://dbpedia.org/data/";

    private static String getDataURL(String resourceURI){
        return resourceURI.replaceFirst( regex: "resource",  replacement: "data").concat(".ttl");
    }

    public static void main(String[] args) {

        Model startingPoint = ModelFactory.createDefaultModel();
        startingPoint.read(String.format("%s%s", dbData, "Skopje.ttl"),  lang: "TURTLE");

        Resource skopje = startingPoint.getResource(String.format("%s%s", dbr, "Skopje"));
        Property country = startingPoint.getProperty(String.format("%s%s", dbo, "country"));
        Resource macedonia = (Resource) skopje.getProperty(country).getObject();
        //System.out.println(macedonia);

        String macedoniaDataUrl = getDataURL(macedonia.getURI());
        Model macedoniaModel = ModelFactory.createDefaultModel();
        macedoniaModel.read(macedoniaDataUrl,  lang: "TURTLE");

        Resource gevgelija = macedoniaModel.getResource(String.format("%s%s", dbr, "Gevgelija"));
        System.out.println(gevgelija);

    }
```

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.
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.
12. Извршете ја програмата.

```java
public static void main(String[] args) {

    Model startingPoint = ModelFactory.createDefaultModel();
    startingPoint.read(String.format("%s%s", dbData, "Skopje.ttl"), lang: "TURTLE");

    Resource skopje = startingPoint.getResource(String.format("%s%s", dbr, "Skopje"));
    Property country = startingPoint.getProperty(String.format("%s%s", dbo, "country"));
    Resource macedonia = (Resource) skopje.getProperty(country).getObject();
    //System.out.println(macedonia);

    String macedoniaDataUrl = getDataURL(macedonia.getURI());
    Model macedoniaModel = ModelFactory.createDefaultModel();
    macedoniaModel.read(macedoniaDataUrl, lang: "TURTLE");
    macedonia = macedoniaModel.getProperty(String.format("%s%s", dbr, "Macedonia"));

    Resource gevgelija = macedoniaModel.getResource(String.format("%s%s", dbr, "Gevgelija"));
    //System.out.println(gevgelija);

    String gevgelijaDataUrl = getDataURL(gevgelija.getURI());
    Model gevgelijaModel = ModelFactory.createDefaultModel();
    gevgelijaModel.read(gevgelijaDataUrl, lang: "TURTLE");
    gevgelija = gevgelijaModel.getResource(String.format("%s%s", dbr, "Gevgelija"));

    Property label = gevgelijaModel.getProperty(String.format("%s%s", rdfs, "label"));
    Property population = gevgelijaModel.getProperty(String.format("%s%s", dbo, "populationTotal"));
    Property postalCode = gevgelijaModel.getProperty(String.format("%s%s", dbo, "postalCode"));
    Property longitude = gevgelijaModel.getProperty(String.format("%s%s", geo, "long"));
    Property latitude = gevgelijaModel.getProperty(String.format("%s%s", geo, "lat"));

    System.out.printf(
            "%s %s %s %s %s %s\n",
            gevgelija.getProperty(label, lang: "en").getString(),
            macedonia.getProperty(label, lang: "en").getString(),
            gevgelija.getProperty(population).getString(),
            gevgelija.getProperty(postalCode).getString(),
            gevgelija.getProperty(longitude).getString(),
            gevgelija.getProperty(latitude).getString()
    );
}
```

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. Но, овој пат, податочното множество покажува кон друг сервер со податоци за лекови, односно кој https://bio2rdf.org. 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-bio2rdf.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.
15. Потсетете се на содржината на “hifm-dataset-bio2rdf.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), хостирани на https://bio2rdf.org/ серверот.

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.

```java
public class Task2 {

    private static final String rdfs = "http://www.w3.org/2000/01/rdf-schema#";

    public static void main(String[] args) {

        Model drugs = ModelFactory.createDefaultModel();
        drugs.read( url: "src/main/java/lab4/data/hifm-dataset-bio2rdf.ttl",  lang: "TURTLE");
        Resource oneDrug = drugs.getResource( uri: "http://purl.org/net/hifm/data#83496");
        Property seeAlso = drugs.getProperty(String.format("%s%s", rdfs, "seeAlso"));

        StmtIterator iter = oneDrug.listProperties(seeAlso);
        while (iter.hasNext()){
            Resource similarDrug = iter.nextStatement().getResource();
            System.out.println(similarDrug);
        }

    }
}
```

```
http://bio2rdf.org/drugbank:DB00736
http://bio2rdf.org/drugbank:DB00338

Process finished with exit code 0
```

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 (dcterms:title) и описот на лекот (dcterms:description). Извршете ја програмата.

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

    (а) да се обратите до URL-то на лекот, но притоа да искористите HTTP content negotiation за да побарате Turtle одговор, или одговор во друг формат кој најмногу ви одговара и кој ќе ви даде директен пристап до RDF тројките за тој лек од графот (како во првиот дел од оваа вежба), или

    (б) да поставите класично SPARQL прашање на SPARQL Endpoint-от на Bio2RDF: https://bio2rdf.org/sparql.

    За опцијата под (а) потребно е HTTP барањето до URL-то на лекот да биде со HTTP header "Accept: text/turtle" или "Accept: application/rdf+xml" или "Accept/application/ld+json", итн., во зависност од форматот кој го преферирате.

    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.

```
private static final String rdfs = "http://www.w3.org/2000/01/rdf-schema#";
private static final String dcterms = "http://purl.org/dc/terms/";

public static void main(String[] args) {

    Model localDrugs = ModelFactory.createDefaultModel();
    localDrugs.read( url: "src/main/java/lab4/data/hifm-dataset-bio2rdf.ttl",  lang: "TURTLE");
    Resource oneDrug = localDrugs.getResource( uri: "http://purl.org/net/hifm/data#83496");
    Property seeAlso = localDrugs.getProperty(String.format("%s%s", rdfs, "seeAlso"));

    StmtIterator iter = oneDrug.listProperties(seeAlso);
    while (iter.hasNext()){
        Resource similarDrug = iter.nextStatement().getResource();

        Model remoteDrugs = ModelFactory.createDefaultModel();
        RDFParser.source(similarDrug.getURI())
                .httpAccept("text/turtle")
                .parse(remoteDrugs.getGraph());

        Resource currDrug = remoteDrugs.getResource(similarDrug.getURI());
        Property dcTitle = remoteDrugs.getProperty(String.format("%s%s", dcterms, "title"));
        Property dcDescription = remoteDrugs.getProperty(String.format("%s%s", dcterms, "description"));

        System.out.printf(
                "%s %s\n",
                currDrug.getProperty(dcTitle).getString(),
                currDrug.getProperty(dcDescription).getString()
        );
    }
```

```
Esomeprazole A highly effective inhibitor of gastric acid secretion used in the therapy of stomach ulcers and zollinger-ellison syndrome. The drug inhibits the H(+)-K(+)-ATPase (
Omeprazole A highly effective inhibitor of gastric acid secretion used in the therapy of stomach ulcers and Zollinger-Ellison syndrome. Omeprazole belongs to a class of antisecre
```