Semantic Knowledge Management and Ontology Development for Online Library

Nensi SKËNDERI
Faculty of Natural Sciences,
University of Tirana,
Tirana, Albania,
nensiskenderi@gmail.com

Areti BOJAXHIU
Faculty of Natural Sciences,
University of Tirana,
Tirana, Albania,
areti.bojaxhiu@fshn.edu.al

Abstract

Nowadays people with computer-related background and programming knowledge are focusing more and more towards the Semantic Web. The reason behind this trend is because many years ago, as the World Wide Web began to develop with the continued vision of Tim Berners-Lee; we saw how the computers could interact with humans by providing information on all kind of different topics people were interested in. Since its creation the Internet has evolved through time into the amazing World Wide Web that we see today. A negative aspect of it is that all we presently have is information that only humans can understand. We want this problem to be solved by a technology which would enable this information to be understandable by computers too. A technology that processes the data not only in a machine-readable way but also understands most of that information's meaning. The focus of this paper is to see how Semantic Web aims to do this by using ontology and most fundamentally, the Resource Description Framework (RDF).

1 Introduction

In recent years, Semantic Web has been increasingly useful because it provides an interesting vision of our online future and a more useful computer-human interaction. The Semantic Web vision had its roots in 2001 when it was first presented in an article by Tim Berners-Lee (the inventor of the World Wide Web). Despite the fact that through the year's critics found Berners-Lee's vision perhaps very distant, some find it appealing and encourage people to study this technology and bring the next-generation semantic data closer to us, even though it is still far away from its grand vision [Hep08].

Even though the Semantic Web concept remains very complex, we have solutions to create applications, process data and extract that data into meaningful results for the users [Dav09]. This is where the Resource Description Framework (RDF) [Swp] comes in handy. Being one the members of the World Wide Web Consortium (W3C) family specifications, RDF is designed as a metadata model and is used for conceptual description or modeling of information that is implemented in web resources, using a variety of syntax notations and data serialization formats [Rdf].

In this paper we are going to see how to process data in a specific RDF format by running some SPARQL queries and using Java as a programming language. The RDF format I am using is called Notation3. Notation3, or N3 as it is more commonly known, is a shorthand non-XML serialization of Resource Description Framework models, designed with human-readability in mind. N3 is much more compact and readable than XML RDF notation [Not].

2 Creating Ontologies using Protégé

By mentioning RDF formats like Notation3 we can create our ontologies based on this format and then by using the RDF query-language SPARQL [Spa13], we aim to get appropriate results against our queries.

Notation3 works in a way that allows us to store our ontology in an URL and add it as a prefix to retrieve data. We do this in order to be able to get the attributes of the ontology that we need and then by adding the prefix before the specific attribute, we can get the results we are searching for.

There is no doubt that we can create ontologies using simple editors, but to be able to understand the exact logic behind it or what are the basic requirements for building an ontology, we can use the help of a software called Protégé. This software goes from creating simple ontologies to managing the most complex ones because it has everything we need and offers a lot of built-in functionalities.

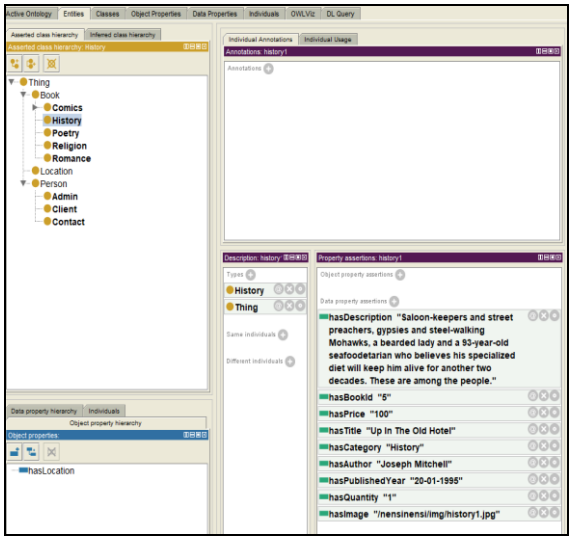

Figure 1: Protégé structure and data

Figure 2: Notation3 example syntax

3 Representing RDF Notation3 format

When using Protégé we first create our classes and then we give each class their specific Data Type Property which will have Data Property Assertions that will hold the values for those Data Type Properties [Ale08].

The way that RDF format Notation3 represents these interactions along with the use of prefixes is shown in the figure below in which:

- the *url* in the prefix is our stored ontology (created with Protégé)
- nensi:Admin is the Data Type Property
- nensi:hasAdminEmail "nensiskenderi20@gmail.com" is the data property assertion for the above data type property

4 Building the Application

After creating the ontology in Protégé and using Notation3 to represent the data there are some other steps we need to go through in order to create our semantic web application.

- Running Apache Jena Fuseki which is our SPARQL server at http://localhost:3030
- Programming with Java and SPARQL query
- Getting results from the applications

The first step is to run Apache Jena Fuseki so we prepare our environment by opening the command prompt and running the command fuseki-server.bat in the folder where we downloaded the server and that will set the server status to Online. Then we upload our ontology to Fuseki which will provide us some services to manipulate and retrieve our results such as:

- http://localhost:3030/nensi/sparql
- http://localhost:3030/nensi/update

Here, 'nensi' is the name of the dataset where our ontology is stored and with the services listed above, we are ready to go to the next step.

After uploading the data and getting the services by Fuseki, we can include them in our code and then be able to run queries. Now that we have all the information that we need we can start programming with Java and running SPARQL queries against the dataset.

```
public List<Book> getAllBooks(){
    List<Book> data = new ArrayList<Book>();
    String book_query = fuseki.PREFIX + "SELECT * \r\n" +
            "WHERE {\r\n" +
            "  ?book a owl:Thing. \r\n" +
            "  ?book nensi:hasTitle ?titulli .\r\n" +
            "  ?book nensi:hasAuthor ?author_name .\r\n" +
            "  ?book nensi:hasImage ?image_path  .\r\n" +
            "  ?book nensi:hasBookId ?book_id  .\r\n" +
            "  ?book nensi:hasPrice ?price  .\r\n" +
            "  ?book nensi:hasQuantity ?quantity  .\r\n" +
            "  ?book nensi:hasPublishedYear ?published_year  .\r\n" +
            "  ?book nensi:hasCategory ?category  .\r\n" +
                "}\r\n" +
            "";

    ResultSet res = fuseki.query_data(book_query);
    while(res.hasNext()) {
        QuerySolution solution = res.nextSolution();
        Book book = new Book();
        book.setTitulli(solution.getLiteral("titulli").getString());
        book.setAuthor_name(solution.getLiteral("author_name").getString());
        book.setImage_path(solution.getLiteral("image_path").getString());
        book.setBook_id(solution.getLiteral("book_id").getString());
        book.setPrice(solution.getLiteral("price").getString());
        book.setQuantity(solution.getLiteral("quantity").getString());
        book.setPublished_year(solution.getLiteral("published_year").getString());
        book.setCategory(solution.getLiteral("category").getString());
        data.add(book);

    }
    return data;
}
```

Figure 3: SPARQL query in Java

In the figure above, we show a SPARQL query that gives us results about all books in our dataset. After we get the data from the query we store it in a list that contains Book objects and then we use this list to show the books and their attributes like author name, book title, book price etc. in our application.

When we create our queries in the code we also can test the same queries in the Fuseki User Interface to make sure we are retrieving the results we want. An example of listing books and their attributes after we run the query in Figure 3 is shown below:

QUERY RESULTS

Table | Raw Response

Showing 1 to 22 of 22 entries

Search:

Show 50 ent

	book	titulli	author_name	image_path	book_id	price	quantity	published_year	category
1	nensi:romance1	"The Sometimes Sisters"	"Carolyn Brown"	"/nensinensi/img/romance1.jpg"	"1"	"60"	"222"	"20-01-1995"	"Romance"
2	nensi:romance2	"Never Let Me Go"	"Kazuo Ishiguro"	"/nensinensi/img/romance2.jpg"	"2"	"190"	"56"	"20-01-1995"	"Romance"
3	nensi:romance3	"Call Me By Your Name"	"Andre Aciman"	"/nensinensi/img/romance3.jpg"	"3"	"600"	"12"	"20-01-1995"	"Romance"
4	nensi:romance4	"Zippermouth"	"Carolyn Brown"	"/nensinensi/img/romance4.jpeg"	"4"	"160"	"0"	"20-01-1995"	"Romance"
5	nensi:history1	"Up In The Old Hotel"	"Joseph Mitchell"	"/nensinensi/img/history1.jpg"	"5"	"100"	"23"	"20-01-1995"	"History"
6	nensi:history2	"Marpingen"	"David BlackBourn"	"/nensinensi/img/history2.jpg"	"6"	"60"	"4"	"20-01-1995"	"History"
7	nensi:history3	"The Big Con"	"David W. Maurer"	"/nensinensi/img/history33.jpg"	"7"	"66"	"43"	"20-01-1995"	"History"
8	nensi:history4	"Martin Luther"	"Andre Aciman"	"/nensinensi/img/history44.jpg"	"8"	"130"	"9"	"20-01-1995"	"History"

Figure 4: Retrieving results about books after running SPARQL query

After researching, building semantic web library management system and after putting together all the resources and combining the technologies needed we now have a basic application. The result of this application is shown in the below figure:

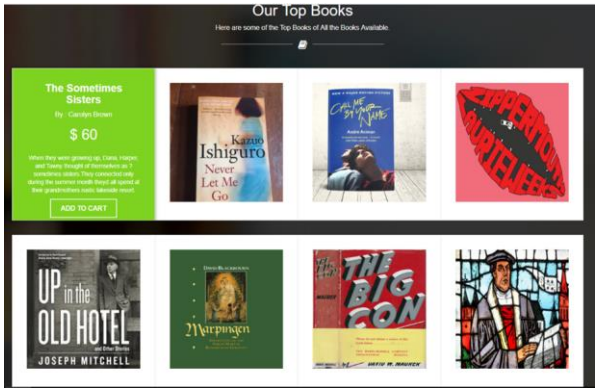

Figure 5: An overview of the library management system

4 Results

Based on the created application which uses the Semantic Web technology and all the researches made in this specific part of the World Wide Web, it can be asserted with certainty that a lot more can be done in order to make this technology move forward and protect its vision by giving the community what it needs: a Web that not only reads the data and shows it to you, but also understands its meaning and provides you with exact information on what you are looking for.

Our future work aims to facilitate not only how the data is being processed but also use new methods and technologies related to semantic web, by promising a future of reliable information and even more by helping people get the most accurate knowledge when they search for it.

References

[Hep08] M. Hepp, P. D. Leenheer, A. D. Moor and J. Sure. Ontology Management: Semantic Web, Semantic Web Services, and Business Applications (Semantic Web and Beyond). Springer, 2008.

[Dav09] H. J. Davies, D. Mladenic and M. Grobelnik. Semantic Knowledge Management. Springer, 2009.

[Ale08] D. Allemang and J. Hendler. Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL. Springer, 2008.

[Swp] https://en.wikipedia.org/wiki/Semantic_Web.

[Not] https://en.wikipedia.org/wiki/Notation3.

[Rdf] https://en.wikipedia.org/wiki/Resource_Description_Framework.

[Spa13] https://en.wikipedia.org/wiki/SPARQL.