Information Retrieval from Ontology through Jena

Mrs. Anita Gutta, Dr. Priti Srinivas Sajja
Department of Computer Science & Technology
Sardar Patel University
Vallabh Vidyanagar, Gujarat, India

Abstract— **Semantic Web is driving the evolution of the current web by enabling users to find, share and combine information more easily. Ontology is an important representation model for the semantic web. This paper talks about information extraction with the help of semantic technologies. Protégé is the tool used to build ontology model. The content is stored in Ontologies and this paper shows how Jena API can be used to extract the data.**

Keywords— Ontology Model, Jena, Protégé, Semantic Web.

I. INTRODUCTION

When Tim Berners-Lee invented the World-Wide Web, its aim was to facilitate communication. The initial goal was to get the information bridge between humans but now there is also a need to allow the participation of machines. He first coined the term 'Semantic Web' in a paper written in 2001. He stated: "A major obstacle to this goal is the fact that most information on the Web is designed solely for human consumption. Computers are better at handling carefully structured and well-designed data, yet even where information is derived from a database with well-defined meanings, the implications of those data are not evident to a robot browsing the web." [1]. Semantic Web can be thought of as an effective way of representing data in the World Wide Web world, or to regard it as a database, able to be understood by the machine some way, linked to the global Web document[2].

The Semantic web relies heavily on formal structure of data and Ontologies play a key role. Ontology consists of statements that define concepts, relationships, and constraints. It is analogous to a database schema or an object-oriented class diagram. Ontology forms an information domain model. The ontology in the application is developed using a free open source editor called Protégé [3]. Protégé is based on Java, is extensible by means of plugins, and provides a foundation for customized knowledge-based applications systems.

Jena [4] is a Semantic Web framework developed by Brian Bride at the HP Semantic Web Laboratory. Jena provides a programming environment and a basic RDF parser for RDF, RDF-S, and OWL, and contains an internal reasoning engine based on rules. Jena's SPARQL will be used to query OWL. Jena has been used in the development of wide range of applications [5]-[7] and Jeremy J. Carroll [8] gives detailed workings on how the framework is laid out and how to write the queries.

The rest of the paper is organized as follows: Section 2 briefs about the application being developed, Section 3 establishes Soil Testing ontology model with protégé. Section 4 goes over Jena queries.

II. APPLICATION

The Jena application covered in this paper is a tool to help farmers search for Soil Testing Locations. Soil Testing is an important first step in getting successful crop yields. Farmers don't have the knowledge that they have to get the soil testing done and based on the results have to select the right seeds and manure. Through this application they can search for testing locations nearby so they can get testing done in a timely manner. The model was developed using protégé and stored as an owl document. It can be stored in any database but for this application it is left as an owl document and data is directly accessed from it. This presents another way how data can be accessed from ontology's without the need of the databases. This also has the added advantage of being easily portable and any new outside Ontologies can be easily integrated into the application.

III. MODEL

The first step before building the ontology model in protégé is to layout the classes and properties that satisfy the model. For Soil Testing (ST) application, below are the classes and their properties.

Class	**State**
Property	Name

Class	**City**
Property	Name
SubOrganizationOf	State (Class)

Class	**Location**
Properties	Name
	Address
	Phone
part of	City (Class)
TestingType (Array)	TestingTypes (Class)

Class	**Testing Types**
Properties	Name
	Description

Once this is established the next step is to build the classes and their properties in protégé.

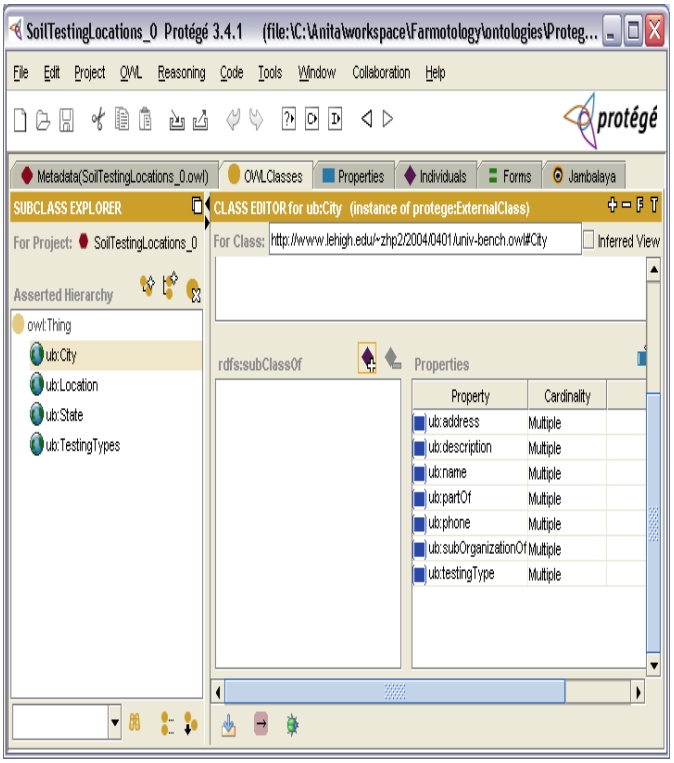

Fig. 1 Classes in Protégé

Now instances for each of these classes can be created as needed.

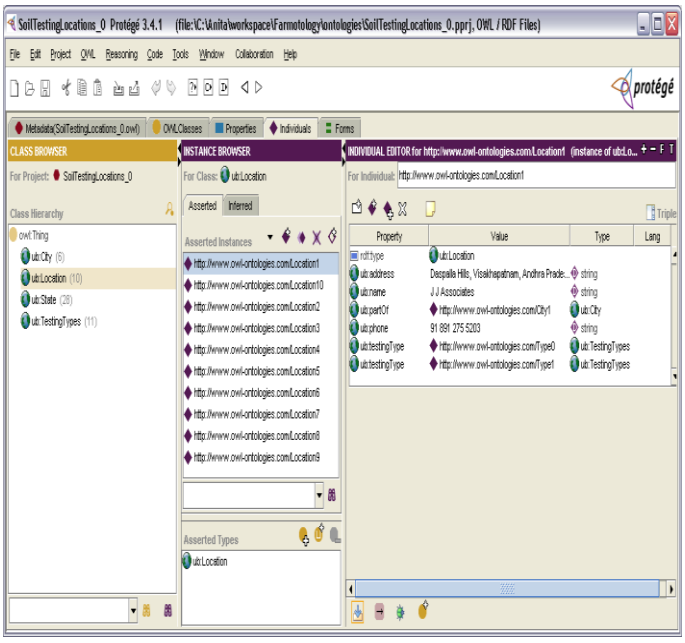

Fig. 2 Instances of classes in Protégé

When the entire model is built is protégé it can be saved as an owl document. The contents of the file will look like this.

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
   xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:swrl="http://www.w3.org/2003/11/swrl#"
   xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xml:base="SoilTestingLocations_0.owl">
 <owl:Ontology rdf:about="SoilTestingLocations_0.owl">
   <owl:imports rdf:resource="SoilTestingLocations_0.owl"/>
 </owl:Ontology>
 <ub:State rdf:about="http://www.owl-ontologies.com/State1">
   <ub:name>AndhraPradesh</ub:name>
 </ub:State>
 <ub:City rdf:about="http://www.owl-ontologies.com/City1">
       <ub:name>Vizag</ub:name>
      <ub:subOrganizationOf rdf:resource="http://www.owl-
ontologies.com/State1"/>
  </ub:City>
  <ub:TestingTypes rdf:about="http://www.owl-
ontologies.com/Type1">
       <ub:name>Grid Sampling</ub:name>
       <ub:description>Does a Grid Sampling soil
testing.</ub:description>
  </ub:TestingTypes>
  <ub:Location rdf:about="http://www.owl-
ontologies.com/Location1">
       <ub:partOf rdf:resource="http://www.owl-
ontologies.com/City1"/>
      <ub:address>Daspalla Hills, Visakhapatnam, Andhra Pradesh
530003, India</ub:address>
      <ub:name>J J Associates</ub:name>
      <ub:phone>91 891 275 5203</ub:phone>
      <ub:testingType rdf:resource="http://www.owl-
ontologies.com/Type0"/>
      <ub:testingType rdf:resource="http://www.owl-
ontologies.com/Type1"/>
  </ub:Location>
```

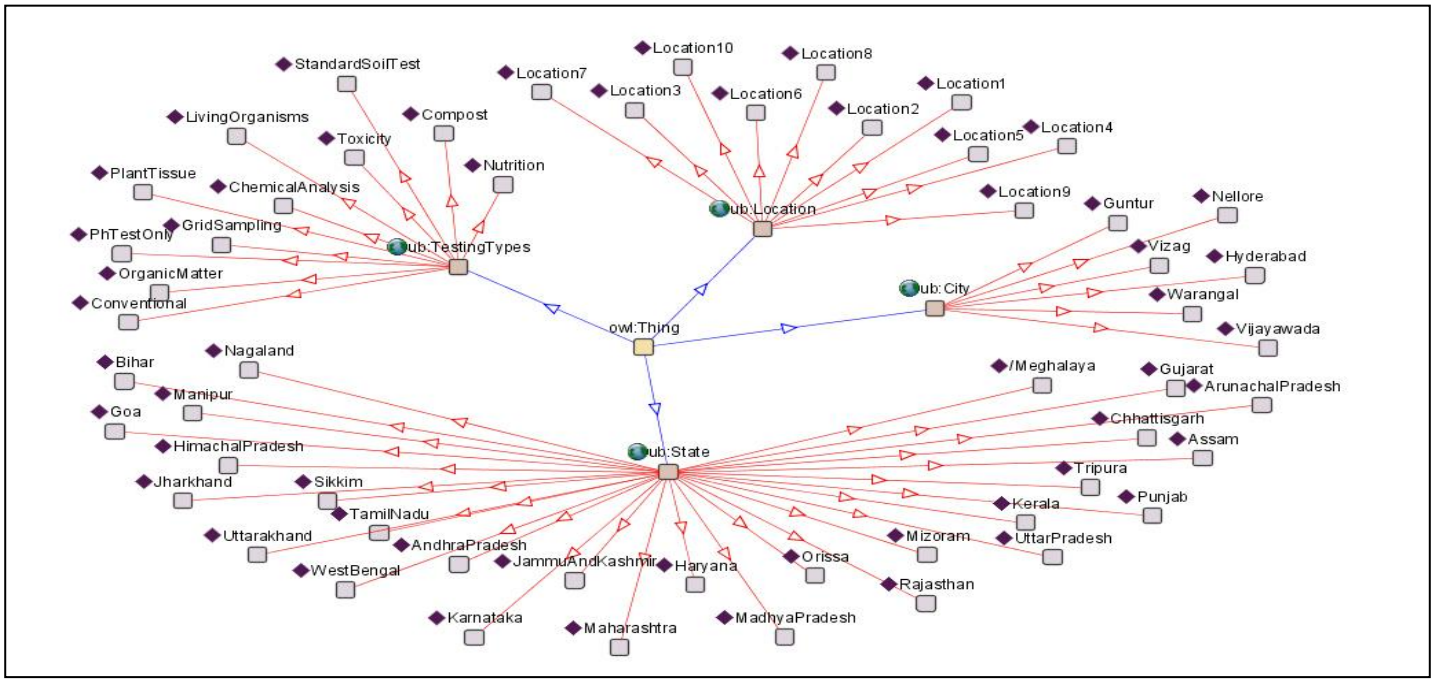

Fig 4: This is how it looks in Jambalaya tab in protégé

IV. RUNNING QUERIES

The application is based on Java and the front end is written in JSP's. Front end components will not be discussed here as they are beyond the scope of the paper. All Jena Queries are written in SPARQL language [9]. The following queries will be explained in detailed.

Q1: Gets all the states
Q2: Gets all the cities for a given state
Q3: Gets all the locations for a given city

A. Q1 – Gets all the states

The first query is very simple it does not take any input parameters. It does a retrieve on the all STATE instances and gets its name. The results are ordered alphabetically by stateName.

Query	SELECT ?name where { ?X rdf:type ub:State . ?X ub:name ?name} order by ?name

Results	[(?name = "AndhraPradesh"), (?name = "ArunachalPradesh"), (?name = "Assam"), (?name = "Bihar"), (?name = "Chhattisgarh"), (?name = "Goa"), ….. (?name = "Uttarakhand"), (?name = "WestBengal")]

On the application page this is how the results are displayed.

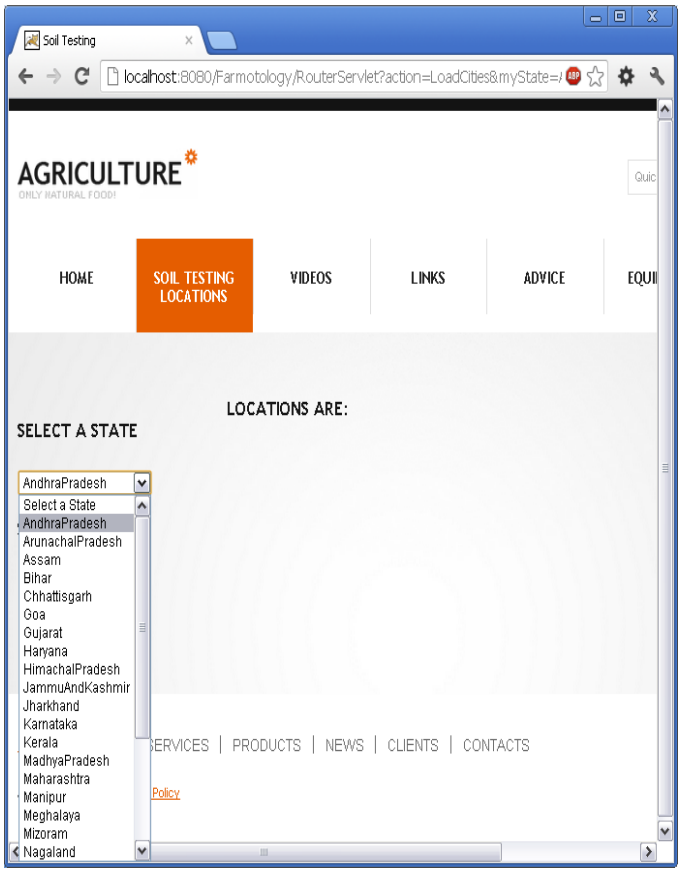

Fig 5: State's displayed on the screen

B. Q2: Gets all the cities for a given state

Query	SELECT ?name where { ?X rdf:type ub:City . ?X ub:name ?name . ?X ub:subOrganizationOf ?Y . ?Y ub:name ?stateName . FILTER (regex(?stateName, ?inputState, \"i\"))} order by ?name

In the second query we are getting all the city names when a state is selected. The first part of the query returns all the city instances. A particular city instance is associated with a state instance. So from this state name is inferred and it is matched with the help of regular expression with the input state name. Jena allows search parameters to be sent to queries at run time. So once the state is selected the search parameter is sent to the query. The following commands binds the state search parameter to the query variable "i" defined in the query.

```
RDFNode stateNode = m.createLiteral(inputState);
initialBindings.add("inputState", stateNode);
```

Results	[(?name = "Guntur"), (?name = "Hyderabad"), (?name = "Nellore"), (?name = "Vijayawada"), (?name = "Vizag"), (?name = "Warangal")]

The results are displayed as a drop down just like states.

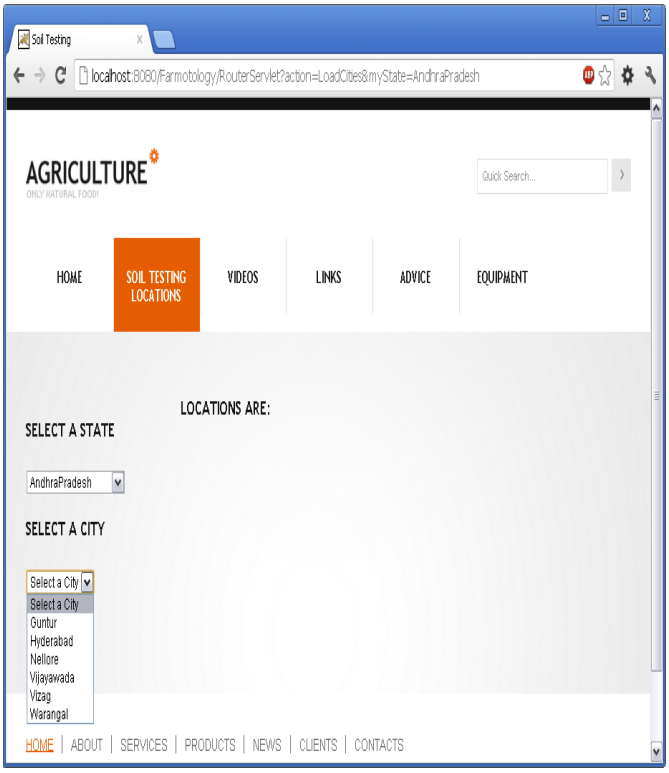

Fig 6: Cities displayed on the screen

C. Q3: Gets all the locations for a given city

Query	SELECT ?X ?name ?address ?phone ?testingName where { ?X rdf:type ub:Location . ?X ub:address ?address . ?X ub:name ?name . ?X ub:phone ?phone . ?X ub:testingType ?Z . ?Z ub:name ?testingName . ?X ub:partOf ?Y . ?Y ub:name ?cityName . FILTER (regex(?cityName, ?inputCity, \"i\"))}

In the owl document a location node has the address, reference to the city and testing types offered at this location. First all the locations nodes are pulled and then the city name is matched to the city the user selected. Regular expressions are used to do this kind of filtering. And also Jena provides with the capability to bind variables at run time. The above query returns the results as follows.

Results	[(?X = <http://www.owl-ontologies.com/Location9>) (?address = "34532234 SPTL Colony, Anand") (?name = "Santosh Soil Integration Services") (?phone = "91 891 250 6210") (?testingName = "Conventional"), (?X = <http://www.owl-ontologies.com/Location10>) (?address = "34532234 gachibowli, hyderabad") (?name = "Mahalakshmi Soil Integration Services") (?phone = "91 891 250 6210") (?testingName = "Grid Sampling"), (?X = <http://www.owl-ontologies.com/Location10>) (?address = "34532234 gachibowli, hyderabad") (?name = "Mahalakshmi Soil Integration Services") (?phone = "91 891 250 6210") (?testingName = "Conventional")]

Jena API is used to extract the needed fields from the output. This data is sent to JSP page and are displayed according to user needs.

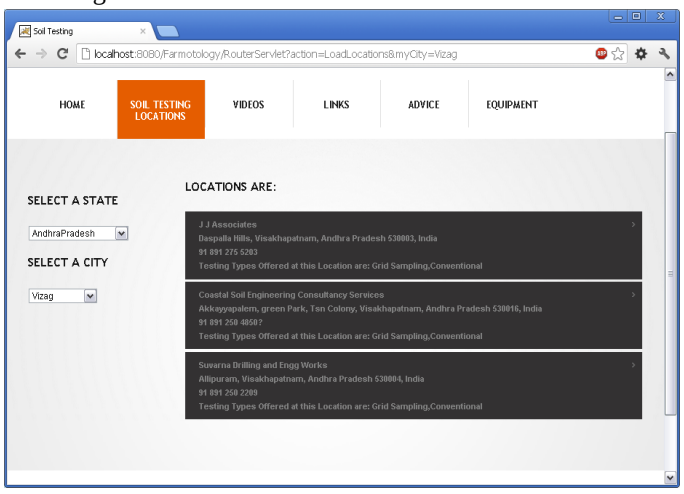

Fig 6: Locations displayed on the screen

V. CONCLUSIONS

There is a lot of research going on Semantic Web Technologies and Ontologies play a key role in them. We have taken a very small piece of it and shown how ontology owl documents can be used to build a real time application. We have left the ontology as an owl document instead of loading into database so that it is easily portable to other systems. And also it makes the application easy to add a new ontology in future.

The application presented in this paper uses only one ontology but multiple Ontologies can be used with the Jena Analyser. More complex applications can be developed with multiple JSP Pages and servlets to handle different requests. Each Jena Analyser can call one or more Ontologies to satisfy the requests. Also future work includes adding more complexity to the model to make it more versatile and comprehensive.

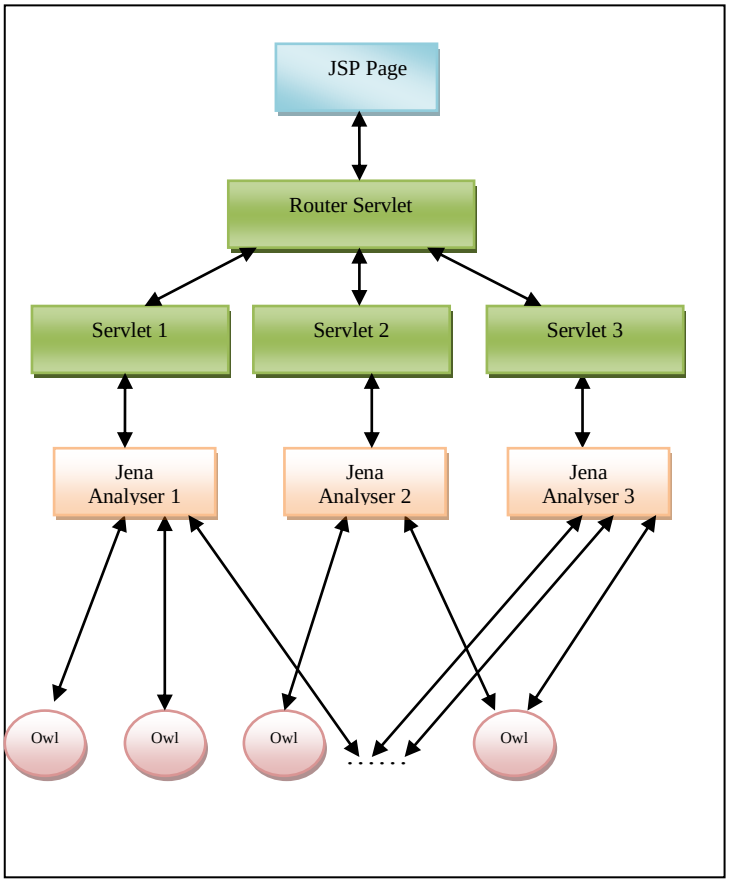

Fig 7: Design of a system with multiple Ontologies

REFERENCES

1. Berners-Lee, Tim. Semantic Web Road Map. W3C Design issues. [Online] 1998. http://www.w3.org/designissues/semantic.html.

2. Balani, N. The Future of the Web Is Semantic. [Online] 2005. http://www.ibm.com/developerworks/cn/java/wa-semweb .

3. *The Protege Ontology Editor and Knowledge Base Acquisition System.* [Online] http://protege.stanford.edu/.

4. *Jena—A Semantic Web Framework for Java.* [Online] http://jena.sourceforge.net/.

5. *A Study on the Improvement of Query Proc- essing Performance of OWL Data Based on Jena2.* Heo, S.-Y. International Conference on Convergence and Hybrid In-formation Technology, Cheonan : s.n., 2008.

6. *Reasoning and Realization Based on Ontology Model and Jena.* Duan, W. Zhang and L. G. Fifth International Conference on Theories and Applications (BIC-TA), Taiyuan : IEEE, 2010.

7. *The Research on the Jena- Based Web Page Ontology Extracting and Processing.* Ding, M. Wang and J. P. Shanghai : s.n., 2005, Vols. First International Conference on Semantics, Knowledge and Grid.

8. *Jena: Implementing the Semantic Web Recommendations.* Reynolds, J. J. Carroll and D. New York : s.n., 2004, Vol. Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters.

9. Eric Prud'hommeaux, Andy Seaborne. *SPARQL Query Language for RDF.* [Online] 2006. http://www.w3.org/TR/2006/CR-rdf-sparql-query-20060406/.