

Wonts - Web Ontology Searcher

A dissertation submitted in partial fulfilment of the requirements

For the MSc in Advanced Computing Technologies

by Salvatore Rapisarda

Department of Computer Science and Information Systems

Birkbeck College, University of London

September 2015

This report is substantially the result of my own work, expressed in my own words, except where explicitly indicated in the text. I give my permission for it to be submitted to the JISC Plagiarism Detection Service. *I have read and understood the sections on plagiarism in the Programme Handbook and the College website.*

The report may be freely copied and distributed provided the source is explicitly acknowledged.

Abstract

In today's fast moving world of accessible information at the touch of the finger, it is more and more prevalent that the speed of data accessibility and information retrieval is increasing. A global demand for data on the World Wide Web[1] (WWW) from users of mobile & tablet devices, laptops and desktop computers have created the need for intelligent forms of retrieving and linking together these data using a maturing technology known as Semantic Web[2–4].

Semantic Web contains a framework aimed at standardising common data, formats and exchange protocols on the Web. This standardisation is governed by World Wide Web Consortium [5](W3C), who aims to promote the sharing of data and exchange protocols on the Web by helping to build a technology architecture to support a web of data. Semantic Web refers to W3C's vision of the Web of linked data.

This project is based on the Semantic Web technologies for linking data and index web information. This report synthetically explains some of the basic concepts of semantic web and describes a system which is able to dynamically search, classify and add individuals to ontology[6–9] class models. The aim of this project is to create a system that is able to add individuals automatically to ontology class models.

Table of Contents

Wonts - Web Ontology Searcher.....	1
Abstract.....	3
1 Introduction	9
1.1 Structure	12
1.2 Development Approach.....	12
1.3 Process of classification	12
2 Background	13
2.1 Family ontology.....	13
2.2 Reasoning	14
2.3 Financial Ontology.....	14
2.4 Map Classes to Search Queries	15
2.4.1 Reason and Proof of Concept	16
3 Design	18
3.1 Requirements.....	18
3.2 Methodology and Technologies.....	18
3.3 Triples Crawler	19
3.4 OWL data manager and provider.....	21
3.5 Apache Solr as Document indexer	23
3.6 Service Indexer.....	23
3.7 Search Engine Service	23
4 Implementation	25
4.1 Triples Crawler	25
4.1.1 Crawling Status	26
4.2 OWL data manager and provider.....	28
4.3 Apache Solr as Document indexer	31
4.4 Service Indexer.....	31
4.5 Search Engine Service	32
4.6 Web Application.....	34
5 Evaluation	38
5.1 Tests.....	38
5.1.1 Financial Ontology Test	38
5.1.2 Pizza-Shop Ontology Test	40
5.1.3 Unit testing	46
5.2 Further Implementation	49
6 Conclusion.....	51
7 References	52
8 Appendix A – Program listing.....	56
8.1 data.services (src/main/java)	56
8.1.1 wonts.data.services.dao	56

8.1.2	wonts.data.services.model.....	63
8.1.3	wonts.data.services.provider	75
8.1.4	wonts.data.services.triples.....	85
8.1.5	wonts.data.services.triples.property.....	90
8.1.6	beans.xml.....	91
8.1.7	log4j.properties	92
8.2	data.services (src/test/java)	93
8.2.1	wonts.data.services	93
8.2.2	wonts.data.services.provider	106
8.2.3	wonts.data.services.repository.....	114
8.3	data.services (pom.xml)	124
8.4	solar.services (src/main/java).....	128
8.4.1	wonts.solr.service	128
8.4.2	wonts.solr.service.messages	159
8.4.3	wonts.solr.service.messages.responce.....	163
8.4.4	wonts.solr.service.properties	166
8.5	solar.services (src/test/java)	169
8.5.1	wonts.solr.service	169
8.5.2	resources	183
8.5.3	application.conf	183
8.6	solr.services (pom.xml).....	185
8.7	triples.crawler (src/main/java).....	188
8.7.1	wonts.triples.crawler	188
8.7.2	wonts.triples.crawler.parser.....	201
8.7.3	wonts.triples.crawler.robot.....	210
8.7.4	wonts.triples.crawler.services	214
8.7.5	wonts.triples.crawler.services.messages.....	219
8.7.6	wonts.triples.crawler.services.messages.response	219
8.7.7	wonts.triples.data.model.....	220
8.7.8	wonts.triples.data.persistence	225
8.7.9	wonts.triples.data.properties	230
8.7.10	beans.xml	231
8.7.11	data_context_beans.xml	232
8.7.12	log4j.properties	233
8.7.13	logback.xml.....	234
8.8	triples.crawler (src/main/resources)	234
8.8.1	application.conf	234
8.8.2	logback.xml.....	235
8.9	triples.crawler(src/test/java).....	235
8.9.1	wonts.triples.crawler	235
8.9.2	ActorContextTest.java	235
8.9.3	wonts.triples.crawler.base	251
8.9.4	wonts.triples.crawler.services	253
8.9.5	CrawlUrlProviderTest2.....	257

8.9.6	data_context_beans.xml	261
8.9.7	resources	262
8.10	triples.crawler (pom.xml)	264
8.11	webapp (src/main/java).....	268
8.11.1	wonts.webapp	268
8.11.2	wonts.webapp.controllers.....	269
8.11.3	wonts.webapp.dto	282
8.11.4	wonts.webapp.property.....	286
8.11.5	wonts.webapp.security	287
8.12	webapp (src/main/test)	293
8.12.1	wonts.webapp	293
8.12.2	WontsWebAppApplicationTests.java	293
8.13	webapp (resources).....	293
8.13.1	wontsapp.properties	293
8.14	webapp (src/main/resources)	294
8.14.1	application.properties	294
8.15	webapp (src/webapp/resources)	294
8.15.1	app	294
8.15.2	css	335
8.16	webapp (pom.xml)	336
9	Appendix B – Wont Database SQL Scripts	339
10	Appendix C - DVD Content.....	341

Wonts - Web Ontology Searcher - by Salvatore Rapisarda

FIGURE 1-1: ROMANCE BOOK ONTOLOGY.....	10
FIGURE 1-2: ROMANCE BOOK GOOGLE SEARCHING.....	10
FIGURE 1-3: ROMANCE BOOK ONTOLOGY WITH ANNOTATION PROPERTY "HASSEARCHQUERY"	11
FIGURE 2-1: EXAMPLE OF A TRIPLE	13
FIGURE 2-2: ONTOLOGY OF A "FAMILY"	13
FIGURE 2-3: FINANCIAL ONTOLOGY	15
FIGURE 2-4: SPARQL RESULTS USING HASSEARCHQUERY ANNOTATION PROPERTY ON PROTÉGÉ.....	17
FIGURE 3-1: PROJECT DESIGN ARCHITECTURE	18
FIGURE 3-3: ONTOLOGY CLASS NAVIGATION	21
FIGURE 4-1: REASONING CRAWLED ONTOLOGY RESULTS IN PROTÉGÉ	26
FIGURE 4-2: URL STATUS	26
FIGURE 4-3: UML DIAGRAM <i>TRIPLES.CRAWLER</i>	27
FIGURE 4-4: ENTITIES DATA MODEL	30
FIGURE 4-5: SERVICE INDEXER ULM DIAGRAM	32
FIGURE 4-6: SEARCH ENGINE SERVICE UML DIAGRAM	34
FIGURE 4-7: WEB APPLICATION STRUCTURE	35
FIGURE 4-8: HOME AND SIGN IN PAGES	35
FIGURE 4-9: LOGIN AND PROJECT MENU PAGES.....	36
FIGURE 4-10: PROJECT PAGES	36
FIGURE 4-11: INSERT URL AND OWL FILE IN A PROJECT	37
FIGURE 5-1: FINANCIAL ONTOLOGY CLASS VIEW	38
FIGURE 5-2: USING DL IN PROTÉGÉ	40
FIGURE 5-3: WONTS UNIT TEST CLASSES.....	46

Wonts - Web Ontology Searcher - by Salvatore Rapisarda

TABLE 2-1- SPARQL RESULTS USING HASSEARCHQUERY ANNOTATION PROPERTY	16
TABLE 3-1: MATRIX USED TO CALCULATE THE PAGE RANKING	19
TABLE 3-2 - WANTS DATABASE	22
TABLE 3-3: WONTAPP DATABASE.....	22
TABLE 5-1: PIZZA SHOP ONTOLOGY SCHEMA	41
TABLE 5-2: PIZZA SHOP ONTOLOGY RELATIONS.....	41
TABLE 5-3: URLs EXAMPLE USED IN PIZZA SHOP ONTOLOGY	43
TABLE 5-4: VIRTUAL MACHINE'S DETAILS	43
TABLE 5-5: GENERAL ONTOLOGY METRICS.....	44
TABLE 5-6: INDIVIDUALS GROUPED BY CLASS	44
TABLE 5-7- DIFFERENCES BETWEEN THREE DOMAINS SITES.....	45
TABLE 5-8: DATA.SERVICES UNIT TESTS.....	46
TABLE 5-9: SOLR.SERVICES UNIT TESTS	48
TABLE 5-10: TRIPLES.CRAWLER UNIT TESTS	49

1 Introduction

The aim of the semantic web is to give an opportunity for users to query the web. We can represent the web as a container of heterogeneous systems. The huge amount of information the web contains does not have any apparent order and has a decentralised structure. The semantic web allows sharing and reusing the web data. It gives to the data a linked structure organised in triples objects. By using it, it is possible to create the representation of the meaning of everything the web contains. Linking together data most of the time is not enough. It is necessary to have something that schematically gives an order, a nature, a classification and some rules to the web resources. This is one of the reasons why semantic web is an important support for the WWW. By using SPARQL[10] it is possible to execute complex queries on the huge amount of information the web contains. SPARQL is a query language used for semantic web ontologies[8]. DBpedia [11,12] is an example of data structured in the web. For instance, it would be possible to execute a query to search for a book written by Shakespeare and has the genre of English Renaissance theatre.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX ontology: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/property/>
select distinct ?book ?author
where {
  ?book rdf:type ontology:Book;
  ontology:author ?author;
  dbp:genre dbpedia:English_Renaissance_theatre .
  FILTER regex( ?author, "shakespeare", "i" ).
}
```

Executing the query above returns the URL[13] document/s associated to the requested book, genre and author (William Shakespeare). The vocabulary in the book example above is given by an ontology shared on DBpedia. In order to keep Dbpedia updated, somebody would need to search, collect and organise the data using the vocabulary in the ontology. This manual task is time consuming. The system **Wonts** described in this report, it is a very useful instrument for searching, collecting and organising the data for us. The system **Wonts** aims to make an automatic process for searching and adding individuals to the appropriate ontology classes. Therefore, it simplifies the task of adding data to existing ontologies.

The main idea of **Wonts** is to create a relation between an ontology class and a query string. This makes the system able to search the query string in a search engine and adds the results to an ontology class. I am now going to illustrate a very simple example, which explains the main idea behind **Wonts**. If for instance, we want to create an ontology that contains the concept of a "*romance book*", this can be graphically represented in the image below:

Wonts - Web Ontology Searcher - by Salvatore Rapisarda

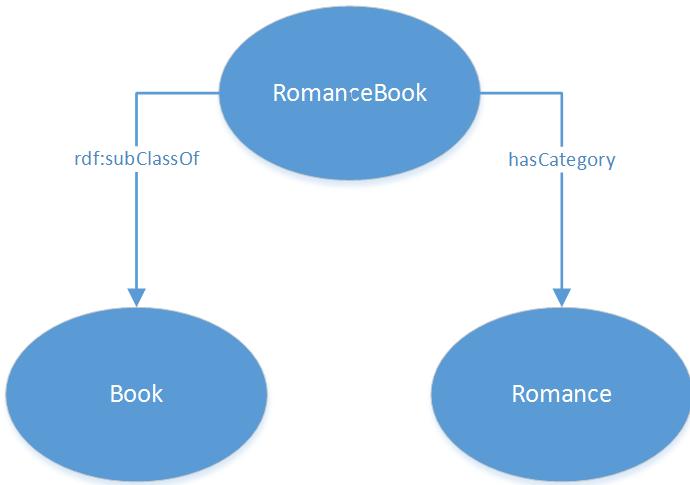


Figure 1-1: Romance Book Ontology

The image above shows that a “romance book” is a book (`rdf:type`) and has a category (`hasCategory`) *Romance*. Therefore, this ontology defines the concepts:

- Classes: *Book, Romance, RomanceBook*;
- Properties: `rdf:subclass`, `hasCategory`.

This ontology at the moment does not contain any individual books. Suppose now we want the ontology to include in the class *RomanceBook* the entire individual books that are categorised as romance. In order to this it would be necessary to search for all the individuals that match the criteria. For example, it is possible to use Google to search a query string like: Book and Romance. Google will returns all the web resource pages where the submitted query string is present.

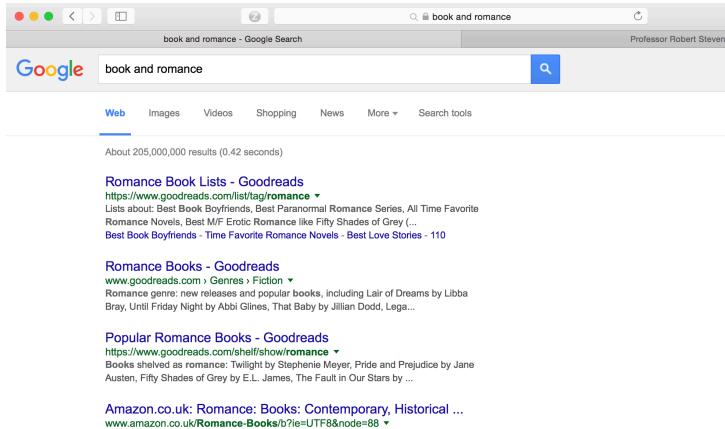


Figure 1-2: Romance Book Google searching

We can manually add parts or all the individuals present in the result that Google returns to the *RomanceBook* ontology class. This is a time consuming task and the process could be made quicker by adding automatically the individuals to *RomanceBook*, especially if the individuals (URL[13] resources) are hundreds or thousands or greater as shown in the figure above. In order to solve this problem, we need to create a relation between the query string and the ontology class that we are searching the individuals for. For instance, it is possible to add another property to the vocabulary ontology which is

used to annotate the query string assigned to the *RomanceBook* class. A name for this annotation property could be “**hasSearchQuery**”.

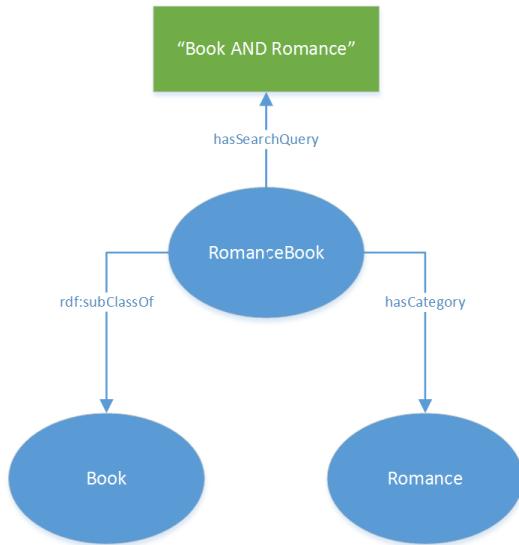


Figure 1-3: Romance Book Ontology with annotation property “hasSearchQuery”

The ontology in the drawing above is equal to the previous ontology except for this new triple: “*RomanceBook*” -> “**hasSearchQuery**” -> “*Book AND Romance*”. Therefore, to search and add all the individuals to the class *RomanceBook*, it is necessary to build a software application that does these three minimum tasks:

- Reads the search query assigned to the class *RomanceBook*
- Executes the query on the search engine
- Adds to the class *RomanceBook* all the web resources that the search engine returns

At the end of this process the class *RomanceBook* will be full of web resources about “Book and Romance”. The hypothetical software described in the example above has been very efficient and helpful because it has automatically searched and added the resources in the *RomanceBook* class for us.

Wonts is doing what is described for the hypothetical software. It searches, collects, classifies and adds possible individuals to ontology classes, where the annotation property **hasSearchQuery** (<http://titan.dcs.bbk.ac.uk/~srapi01/hasSearchQuery>) is present. In order to update the ontology, **Wonts** follows these main steps:

- Executes a SPARQL query on the ontology to get all the triples where the annotation property **hasSearchQuery** is present. The SPARQL query is executed to obtain every class that has a query string defined;
- Executes the query string/s in an internal search engine;
- Adds the URL results of each query string executed into the ontology class associated.

All the system architecture will be explained in more details further on in this report.

1.1 Structure

This project report is divided into five main chapters. In the first chapter (2 Background), I discussed the project background by introducing concepts of semantic web. In the second chapter (3 Design), I explained the design and all the requirements the project is based on. In the third chapter (4 Implementation), I described in detail how the application has been developed. I dedicated the forth and final chapter (5 Evaluation) to testing and analysing the project which includes some examples of how the software meets the specified requirements. Also included in this chapter are further implementations and a critical evaluation about the system.

The DVD attached to this report contains all the OWL files example discussed further in this report. Please see Appendix C.

1.2 Development Approach

To develop this project I will be using AGILE[14,15]/SCUM[16] methodology and Test-driven development[17] (TDD) approach with Unit Tests[18,19]. This will ensure that all the testing and evaluating will be completed during the development of the project.

1.3 Process of classification

This project is using a crawler[20,21], an indexer[21,22] and a search engine[21,23] for parsing, analysing and classifying web documents. A process of classification is semantic when it is possible to dynamically decide which is the most appropriate destination class for a web resource. A resource can be classified in many ways, therefore it can be an element belonging to different classes. The process that associates a resource to a class is based on the resource content. Each class in a model has a proper meaning and contains elements that have common natures and properties. For example, to use a search engine we need to specify a query of research. The search engine will return in order of importance all the resources that most probably fit the researched query criteria. However, the entire resources returned have common nature. Therefore a class can represent them. This makes us able to describe a class by using a query string. The more suitable the query string is defined for an ontology class, the more congruent and correct individual items will be bound to the same class. If each class has a well-defined associated query string, the search engine will return consistent results to add to the class. If we also want acceptable results obtained, it is necessary to restrict the domains of research that we are interested in. To do this manually is time consuming. **Wonts** is a system that helps in the process of searching and adding the resource it finds to ontology classes that contains a query string of research.

2 Background

One of the most important parts of the Semantic Web is the vocabulary defined by an ontology. The ontology is used to define the concepts and relationships.

Below is an example of a triple, which uses the basic representation of a concept: Subject – Predicate – Object



Figure 2-1: Example of a triple

The concept of Subject – Predicate – Object can be simplified in a sentence, for example “John is a man”. “John” is the subject, “is a” is the predicate (property or rule) and “man” is the object. In this case “man” is the class that “John” belongs to. Therefore “John” is an individual and each individual instances of this class is a “man”. The ontology and set of individual instances of classes is a knowledge base or TBox[24].

2.1 Family ontology

By using the triple we can easily represent a simple ontology of a “Family” that is shown in the drawing below.

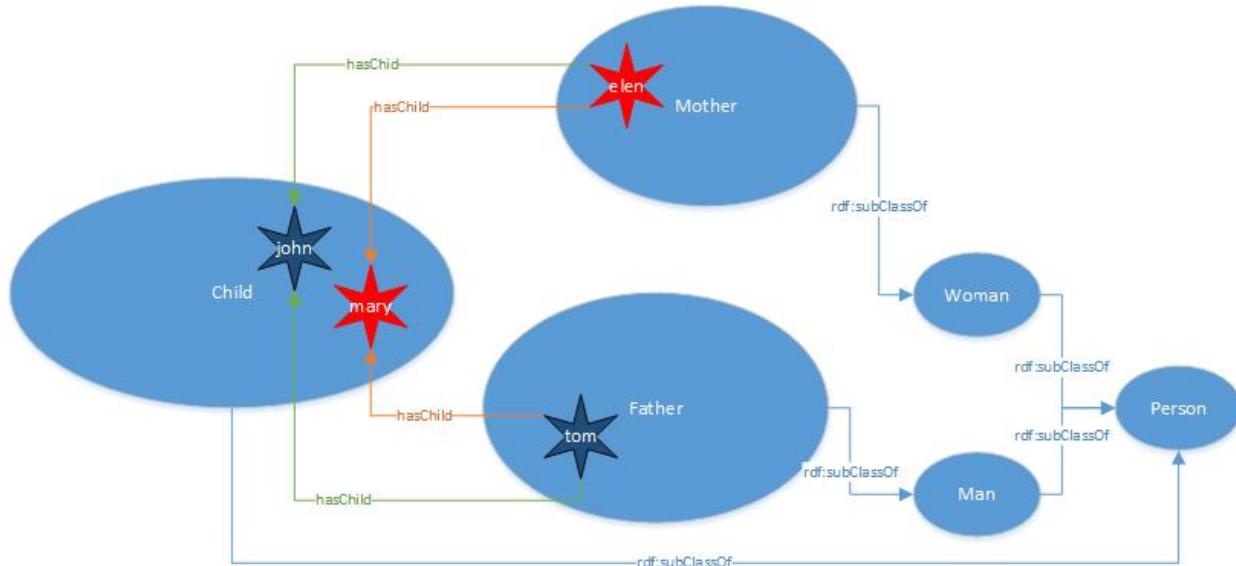


Figure 2-2: Ontology of a “Family”

From this drawing above we can distinguish

- Classes: Person, Woman, Man, Father, Mother and Child
- Properties: rdf:subClassOf and hasChild
- Individuals: *john*, *mary*, *tom* and *elen*

We have also defined the following concepts:

- A woman is a subclass of person
- A man is a subclass of person
- A mother is a subclass of woman and has a child
- A father is a subclass of Man and has a child
- A child is subclass of a person

2.2 Reasoning

We can easily extend the ontology by adding another concept, for example: a sibling has the same mother or father. Therefore we can logically conclude and infer that *john* and *mary* are siblings. It is possible for the ontology to use a “Semantic Reasoner”[25] for inferring all the logical consequences.

In the ontology “Family” example mentioned above we have added the individuals to specific classes that are part of the ontology. We have categorised and chosen the ontology class that best represents its individuals.

2.3 Financial Ontology

We can now define another ontology for the financial products by using the following concepts:

- Portfolio and Fund are sub classes of Financial
- Account is sub class of Portfolio
- ISA and SIPP are sub classes of Account

The graph above is a practical example of an ontology model. We can note that all the classes are interconnected to each other by property. For example Account is “rdf:subClassOf” of Portfolio and Portfolio is “rdf:subClassOf” Financial class.

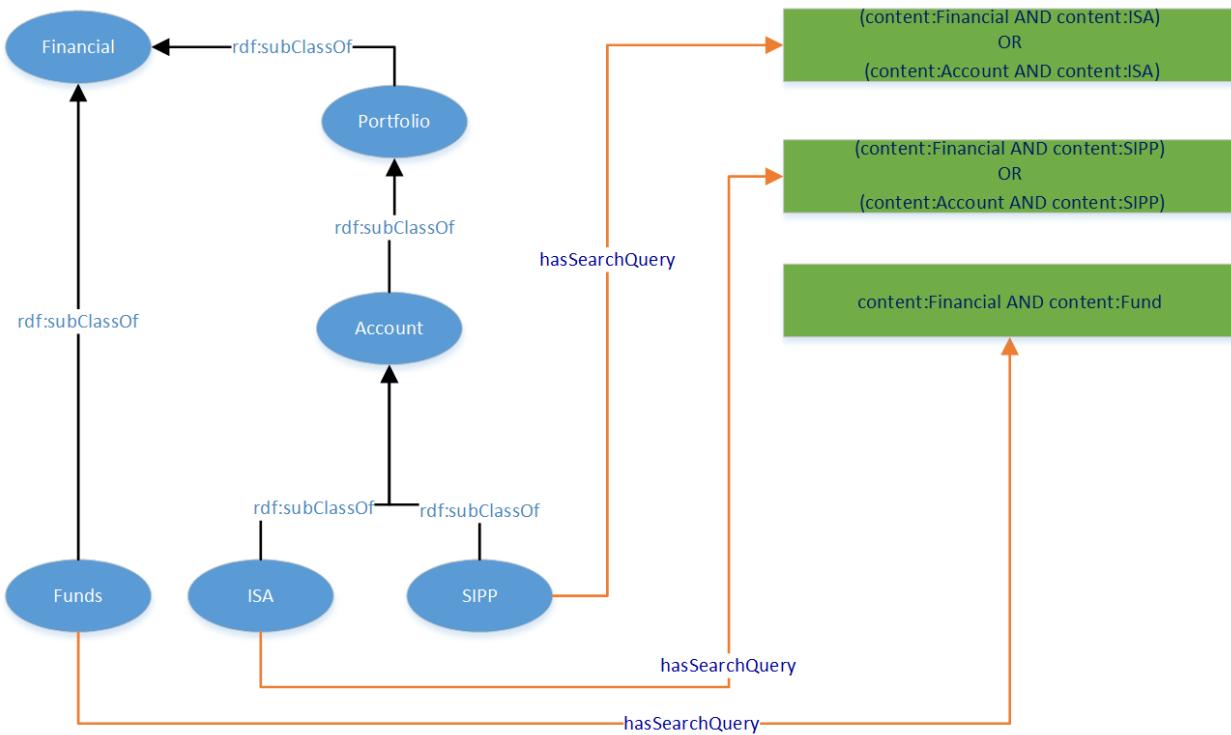


Figure 2-3: Financial Ontology

If we want to add to the ontology all the individuals that the classes Fund, ISA and SIPP contain, we need:

- All the web pages where the financial company offers to open an account that can be an ISA or SIPP;
- All the financial sites where it is possible to buy funds.

2.4 Map Classes to Search Queries

By using the annotation property **hasSearchQuery** described earlier in the introduction, it is easier to assign a search query to a class for finding all the possible individuals that belong to the class. The financial ontology mentioned before and shown in Figure 2-3 can be used for this purpose. This ontology can be written in **Turtle**[26,27] syntax as below:

```

@prefix : <http://titan.dcs.bbk.ac.uk/~srapis01/finance#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix srapis01: <http://titan.dcs.bbk.ac.uk/~srapis01#> .
@base <http://titan.dcs.bbk.ac.uk/~srapis01/finance> .

<http://titan.dcs.bbk.ac.uk/~srapis01/finance> rdf:type owl:Ontology .

srapis01:hasSearchQuery rdf:type owl:AnnotationProperty .

:Account rdf:type owl:Class ;
        rdfs:subClassOf :Portfolio .
  
```

```

:Financial rdf:type owl:Class .

:Funds rdf:type owl:Class ;
    rdfs:subClassOf :Financial ;
    srapis01:hasSearchQuery "content:financial AND content:fund" .

:ISA rdf:type owl:Class ;
    rdfs:subClassOf :Account ;
    srapis01:hasSearchQuery "content:financial AND content:isa OR ( content:account AND content:isa)" .

:Portfolio rdf:type owl:Class ;
    rdfs:subClassOf :Financial .

:SIPP rdf:type owl:Class ;
    rdfs:subClassOf :Account ;
    srapis01:hasSearchQuery "content:financial AND content:sipp OR (content:account AND content:sipp)" .

```

2.4.1 Reason and Proof of Concept

Now we need to instruct a search engine that automatically searches and adds the individual resources to the target ontology classes. However, it is necessary to read the ontology and search for all the triples that contains **hasSearchQuery** as annotation property. This task is not difficult to do and for this propose we can use SPARQL[10,28]. SPARQL is the SQL language for RDF and ontology. Therefore, to know exactly which are the classes and their relative search query associated, it is necessary just to execute a SPARQL query such as below:

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX wonts: <http://titan.dcs.bbk.ac.uk/~srapis01#>
SELECT ?subject ?object
WHERE { ?subject wonts:hasSearchQuery ?object }

```

The result of executing the query above will produce a list of subjects and objects where the annotation property **hasSearchQuery** is defined:

Table 2-1- SPARQL Results Using hasSearchQuery Annotation Property

subject	object
ISA	content:financial AND content:isa OR (content:account AND content:isa)
SIPP	content:financial AND content:sipp OR (content:account and content:sipp)
Funds	content:financial AND content:fund

Wonts - Web Ontology Searcher - by Salvatore Rapisarda

The screenshot shows the Wonts interface running on a Mac OS X system. The window title is "finance (http://titan.dcs.bbk.ac.uk/~srapis01/finance) : [/Users/salvo/Documents/git/wonts/ontology/finance.owl]". The tab bar includes "Active Ontology", "Entities", "Classes", "Object Properties", "Data Properties", "Annotation Properties", "Individuals", "OWLviz", "DL Query", "OntoGraf", "Ontology Differences", and "SPARQL Query". The "SPARQL Query" tab is selected. A search bar at the top right contains the placeholder "Search for entity".

The main area displays a SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX s: <http://titan.dcs.bbk.ac.uk/~srapis01#>
SELECT ?subject ?object
WHERE { ?subject s:hasSearchQuery ?object }
```

Below the query, the results are presented in a table:

subject	object
ISA	"(content:financial AND content:isa) OR (content:account AND content:isa)"@
Funds	"content:financial AND content:fund"@"
SIPP	"(content:financial AND content:sipp) OR (content:account AND content:sipp)"@

At the bottom of the interface, there is an "Execute" button and a status message: "To use the reasoner click Reasoner->Start reasoner" followed by a checked checkbox labeled "Show Inferences".

Figure 2-4: SPARQL Results Using hasSearchQuery Annotation Property On Protégé

3 Design

As already mentioned, to manually research and add the results to an existing ontology requires time, especially with large ontologies. For this purpose, **Wonts** has been created to search and execute the queries in Apache Solr[29](the content document indexer used in this project). The results returned from Apache Solr will be used as individual data to add to the ontology.

This project is like a search engine, which has as search query ontology and returns the same ontology fulfilled with individuals.

3.1 Requirements

These are the most important components that this project requires:

- Triples Crawler (***triples.services***)
- OWL [8] data manager and data service provider (***data.services***)
- Apache Solr document indexer (***Apache Solr***)
- Service indexer (***solr.services***)
- Search Engine Service (***solr.services***)
- Web Application
- Database **wonts**
- Database **wontsapp**

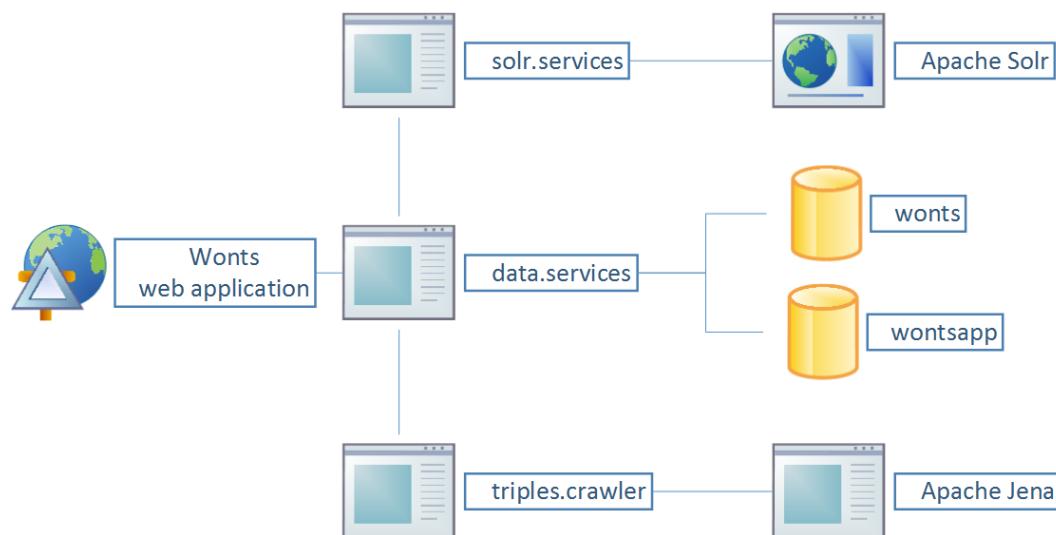


Figure 3-1: Project Design Architecture

The drawing above shows and explains the project design architecture overall.

3.2 Methodology and Technologies

All the project components should be developed using Java version 1.8. Each java project should use Apache Maven[30] to make it easier to manage and build a project using the xml file descriptor pom.xml that extends the idea and concept of project object model.

All the project components must be developed using TDD. A unit test scenario should be created before the code is implemented in each public software method. Therefore each software part of the project must be tested using JUnit[19]. The entire project should be developed using dependency injection[31](DI) technique which allows the class to be loosely coupled and to use inversion of control[32] (IoC). Spring[33] library framework can be chosen as the main DI container.

Akka[34] should be used to develop the services ***triples.crawler*** and ***solar.services*** because it is possible to create concurrent message-driven applications using the Scala[35] Actors[36,37] model implementation.

It is also necessary for the project to use Apache Jena which makes it easy to create and manage an ontology model. This will be mostly used during the process of crawling to save the crawled web resources and during the process used in the search engine service to populate the ontology with individuals.

Both Java Persistence (API)[38] (JPA) and Hibernate[39] as vendor adapter in Spring container entity manager[38] (EM) should be used in the ***data.services*** for allowing DI.

For the Web application it is comfortable to develop the necessary back-end application programming interfaces[40] (API) using Spring model-view-controller[41] MVC. The user interface should be developed with a single page application[42] and a good candidate is AngularJS [43]. Rather than implementing from scratch all the Cascading Style Sheet [44] (CSS) that the HTML[45] page requires, it would be possible to use Bootstrap[46] that has already implemented many HTML controls and styles ready to use. In the Web application for all the asynchronous communications between the WEB application and the HTML interface it would be better to use JSON[47,48] in combination with Ajax[49] and AngularJS Promises[50] asynchronous communication development techniques.

After this introduction about the architecture for Wonts system, I am going to describe more in detail, the main functions of each project components.

3.3 Triples Crawler

The first and most important system component to create is the crawler. Usually, a crawler is used to extract and represent as a matrix the interconnections between pages in the web. The matrix result is used to calculate the page ranking of each web resource.

Table 3-1: Matrix used to calculate the page ranking

	A	B	C
A	0	0	1
B	1	0	0
C	1	1	0

For example the above matrix can express a simple web schema links between pages:

- “A” links out “B” and “C”, so that “B” and “C” have a link in from “A”;
- “B” links out “C”, so that “C” has a link in from “B”

- “C” links out “A”, so that “A” has a link in from “C”

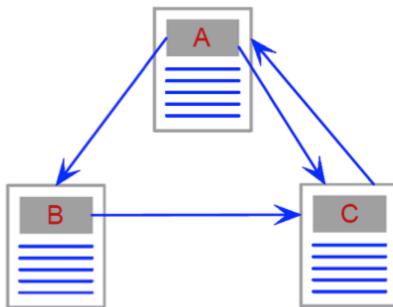


Figure 3-2: Simple Web Resources Network

The figure above is showing how the crawler normally can represent a simple web resources network.

The crawler that must be developed for this project should be different than a normal one. It has to read and add all the links contained in a web page to an OWL ontology model. Instead of using a matrix, it should use an ontology model to represent all the links that has been crawled. In an ontology model it is natural to link resources together and represent them as triples in a common RDF semantic linked model file.

The crawled resources can be represented by the following ontology concept, where the interpretation of the domain is:

$$I = (\Delta^I, \cdot^I), \quad KB = Tbox + ABox, \quad I \models KB$$

Tbox:

$\top \sqsubseteq \text{Document}$

$\top \sqsubseteq \text{Mediatype}$

$\text{Document} \sqcup \text{Mediatype} \sqsubseteq \text{WebResource}$

$\text{Html} \sqcup \text{Pdf} \sqcup \text{Doc} \sqcup \text{Docx} \sqcup \text{Ppd} \sqcup \text{zip} \sqsubseteq \text{WebResource}$

$\exists \text{linksOut}. \top \sqsubseteq \text{WebResource}$

$\top \sqsubseteq \forall \text{linksOut}. \text{WebResource}$

$\exists \text{hasLinkIn}. \top \sqsubseteq \text{WebResource}$

$\top \sqsubseteq \forall \text{hasLinkIn}. \text{WebResource}$

$\text{linksOut}^- \equiv \text{hasLinkIn}$

ABox: All the assertions and individual resources will be populated during the web crawling.

In the ontology described above all the individuals contained in the class *WebResource* are all the URLs that have been crawled.

In the Figure 3-2 above, “URL-A → linksOut → URL-B” therefore “URL-B → hasLinkIn → URL-A”. In this case “**linksOut**” and “**hasLinkIn**” are two properties where the domain and the range are equal and these are represented by the class **WebResource**. It is also possible to note that the **hasLinkIn** is the inverse function of **linksOut**.

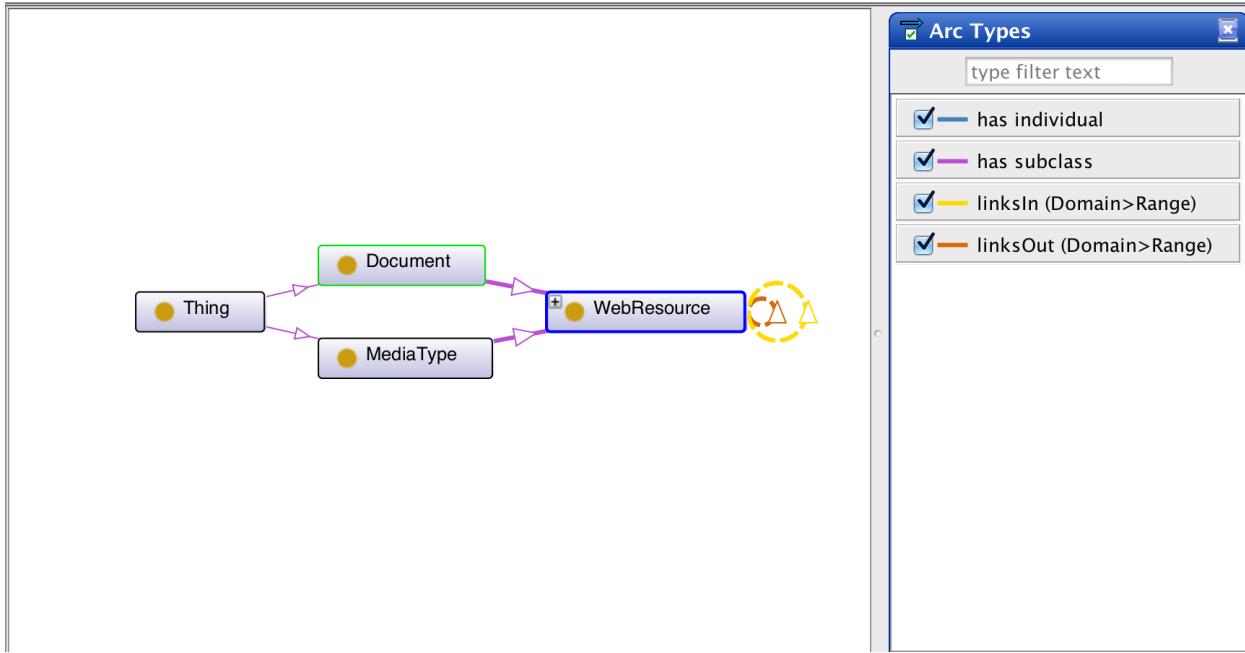


Figure 3-3: Ontology Class Navigation

The picture above is an example of how this ontology makes it possible to navigate and query, by using SPARQL[10], the individual web resources data that are represented by the entire URL page crawled.

The process of reasoning can easily infer all the relations between the entire resources crawled. The crawler must create a semantic model where is clearly defined the linking relations between all the resources in a supposed domain of research.

Once the crawler has finished processing, it will save in the database the ontology that contains all the relations between resources. This ontology will be used later on during the indexing process. Apache Jena library should be used to save and query the ontology model in **Wonts**

3.4 OWL data manager and provider

The **data.service** component should provide all the interfaces, models and methods necessary to manage the data information the system requires. The design system should have two databases:

- **wonts**
- **wontsapp**

All ontology's triples generated by the **triples.crawler** service will be stored in **wonts** database. Apache Jena library will be used to create and store the ontology models. This library makes it easier to create

classes, properties and individuals in the ontology models. Therefore, it will be widely used during the process of crawling and when it is necessary for:

- Writing the ontology model in a file,
- Saving the ontology model on **wonts** database or
- Executing a SPARQL query on any ontology model.

Apache Jena is also used to execute the SPARQL query in the ontology model and to get all the triples where the property **hasSearchQuery** is defined. As explained before this technique is used in the search engine to map each class with possible individuals.

By using the same database schema used in Apache Jena Fuseki[51] it is possible to store the triples that the crawler generates. This database schema is described in the table below:

Table 3-2 - wonts database

Tables	Description
nodes	This table contains the node resources
triples	This table contains all the triples represented subject (s), predicate (p), object (o)
prefix	This contains all the prefixes used in the ontologies
quads	This is not used in the project but apart from Apache Jena Fuseki DB

The database used for the web application and all the services will be **wontsapp**. This database will not contain a big amount of data but will have all the necessary information the system needs for working as a whole. Every time the application is started up, the tables and relative structures should be automatically created and updated by Hibernate entity manager provider. The JPA2 and DI are development software technologies that should be used in the **data.services** component.

This is a schema that the **wontsapp** should use:

Table 3-3: wontsapp database

Tables	Description
ontology	This table contains the information about all the ontology files that the user has uploaded
project	This table contains the user projects
url	This table contains the URLs in use
urlToCrawl	This table contains the URLs to crawl
user	This table contains user information
project_ontology	All the projects that are related to an ontologies in a one-to-many relationship
project_url	All projects that are related to the URLs in a many-to-many relationship
url_project	All URLs related to projects in a many-to-many relationship

3.5 Apache Solr as Document indexer

Apache Solr is a part of the requirements and will be used in this project for indexing all the resources, that have been previously crawled and stored as **triples** in the database **wonts**. Apache Solr is a mature and sophisticated document indexer based on Apache Lucene[52] project. This enables the user to create customised search engine applications and by using Apache Tika[53] the system will be able to read and analyse the content of any common document library. The combination of content analyser, indexer and query search engine makes Apache Solr a really useful web service extension to use in this project.

3.6 Service Indexer

The process of indexing each resource starts by reading each node previously processed by the crawler. This information is stored in **wonts** database therefore it is necessary to query the database using SPARQL. The query to use is:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX wonts: <http://titan.dcs.bbk.ac.uk/~srapis01#>
SELECT ?o ?s
WHERE { ?s wonts:linksOut ?o .
        FILTER regex(str(?s), "http://example.com", "i") }
```

Executing the query above will return a result set of “**?o**” (object) and “**?s**” (subject) where each “**?s**” starts with the string “<http://example.com>” and “linksOut” an “**?o**” which is the resource to index. Each resource that needs to be indexed is downloaded and the content is sent to Apache solar to be analysed by Apache Tika and indexed afterwards.

3.7 Search Engine Service

Until now the process of crawling and indexing has been described. Another fundamental part of this project is the search engine service. This search engine aims to populate ontology classes of individuals. As described above by using the annotation property “**hasSearchQuery**”, it is possible to associate a query string to a class. Therefore, the result obtained by searching this query string can be used as individuals to add to the class researched. If all the necessary web contents have already been indexed, the task of searching can be done by query Apache Solr. However, the actions needed to do this are:

- Search in the given ontology all the query string associated to a class;
- Execute each query string in Apache Solr;
- If a result is obtained from Apache Solr, this will be the list of web resources to add as individuals to the class of research.

As described above searching the query string in the ontology can be gained by executing a SPARQL query such as:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX wonts: <http://titan.dcs.bbk.ac.uk/~srapis01#>
SELECT ?subject ?object
WHERE { ?subject wonts:hasSearchQuery ?object }
```

This query will return, if any, a list of results where the **?object** is the query string and the **?subject** is the associated class. For each query string obtained as result of the query, it is possible to submit a request to Apache Solr. Then Apache Solar will return all the web resources that are matching the query requested. **SolrJ**[54] library will be used in this project to communicate to Apache Solr. This is an open source library that mostly fits the purpose of quickly interacting with Apache Solr. During the index process this library will also be used in the process of updating the content resource in Apache Solr. In order to query Apache Solr it will be necessary to send a **solrQuery** message. Apache Solr will then return a result set that will be used to populate the ontology class.

4 Implementation

As listed in the requirements section, Java version 1.8 has been used to develop and implement this project. The system Wonts is composed of four main projects. Each project is managed by using Maven pom.xml file descriptor. Listed below are the module components, which are contained in the java project solution:

- **Triples.crawler**, contains the crawler service;
- **Solar.services**, contains the service indexer and the search engine;
- **Data.services**, contains all the interfaces, models and methods to manage the databases: **wonts** and **wontsapp**;
- **Webapp** – contains Wonts Web Application.

4.1 Triples Crawler

The workflow used by the Triples crawler module is very simple. Each URL that should be crawled is inserted in the table *UrlToCrawl* where the crawler service is going to check repeatedly every 5 minutes (this time range can be changed in the configuration file). Every time an URL is processing its status is changed from “*isToCrawl*” to “*isCrawling*”. Once the process of crawling has finished, the status of the file is changed to “*IsCrawled*”. The status of the file in the table *UrlToCrawl* is important because it decides all the phases the URL links has done. When the status is set as “*IsCrawled*”, it means that the crawler has finished the process of crawling. At this point **solr.service** downloads and sends each crawled resource contents to Apache Solr web service. Apache Solr will then index the content that has been sent. At the end of this process the status is changed as “*isIndexed*”. The entire process of indexing will be described in details later on in this report.

The crawler parses the content of the web pages that are direct descendants of the URL of research. For instance, if we crawl Michael Zakharyashev web page (<http://www.dcs.bbk.ac.uk/~michael>) we will notice that all the pages that have been crawled are just the direct descendants. The crawler is using Apache Jena to create an ontology model and adds to the model all the links extracted from each parsed document page.

During the process of parsing, the crawler creates a relation for each link found in the page by using the “**linksOut**” annotation property. At the end of the process, this ontology will contain for each crawled page a “**linksOut**” property to each link extracted from that page.

While parsing the pages, the crawler has used only the property **linksOut**. However, as I described above the property **hasLinkIn** is the inverse property of **LinksOut** ($\text{linksOut}^- \equiv \text{hasLinkIn}$), therefore it is possible to infer all the **hasLinkIn** property by reasoning the ontology model.

For example if we are using Protégé[55–57] and we open an OWL file created during Michael Zakharyashev web site[58] crawling, we can easily start the reasoning process by using its plug in Hermit[59], navigate to the table DL Query and execute a query like: **j:0:hasLinkIn some j:0:WebResource**. As a result, 202 individuals will be obtained where the property **hasLinkIn** is present. This proves that the $\text{linksOut}^- \equiv \text{hasLinkIn}$ is consistent. Therefore this ontology can be used as the matrix which is used for page ranking as described above.

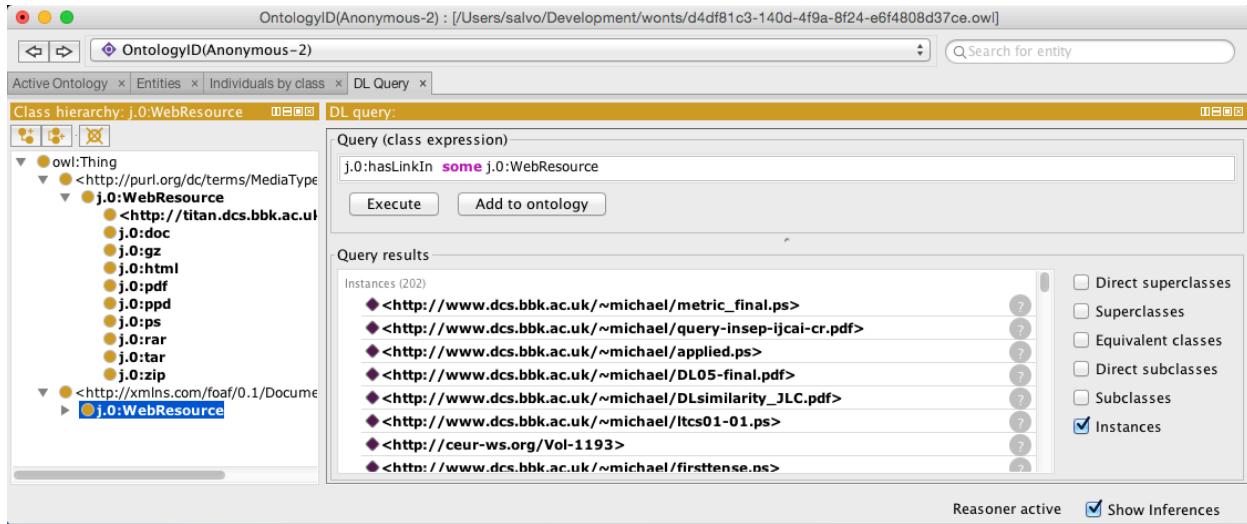


Figure 4-1: Reasoning Crawled Ontology Results in Protégé

4.1.1 Crawling Status

Continuing with the crawling process description. In the database **wontsapp** the table *UrlToCrawl* contains the field “**status**”. This field can assume the values on list describes below:

- (0, isToCrawl), the URL should be crawled;
- (1, isCrawling), the URL is crawling;
- (2, isCrawled), the URL has been crawled;
- (3, isPersisted), the Ontology OWL file has been persisted;
- (4, isIndexing), the URL resources are indexing;
- (5, isIndexed), the entire URL resources have been indexed.

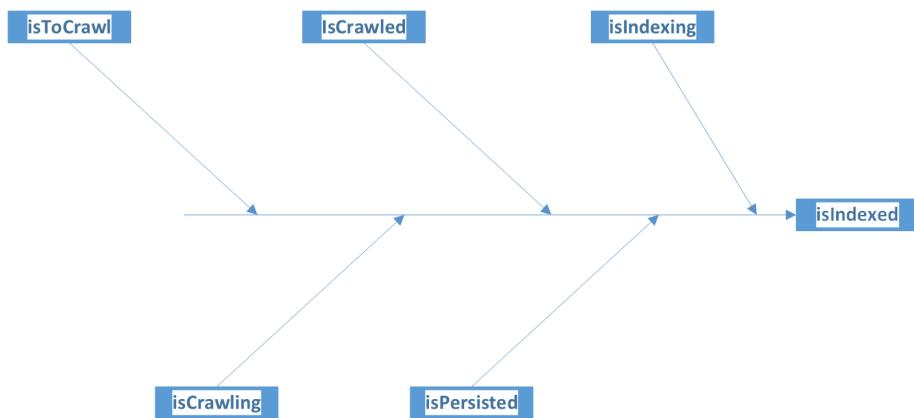


Figure 4-2: URL Status

Until now all the crawling process, the technologies involves, the high analysis and concepts have been described. Now I am going to describe in more detail how the ***triples.crawler*** works. The image below shows the UML sequence diagram that describes the component software architecture.

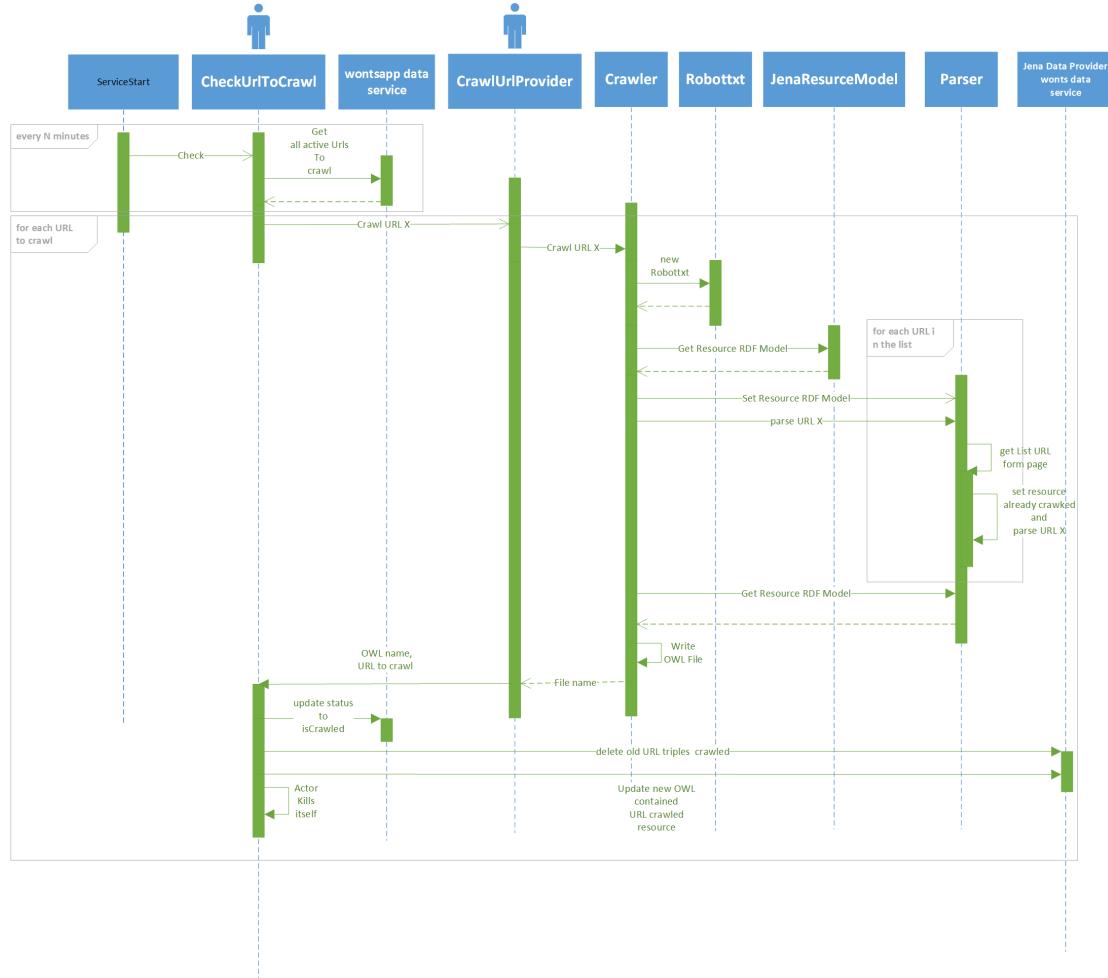


Figure 4-3: UML Diagram ***triples.crawler***

In the UML diagram above this component is using two Akka actors[34]: ***CheckUrlToCrawl*** and ***CrawlUrlProvider***. These two actors are referenced and used by the system context implemented in the ***ServicesStart*** object. The ***ServicesStart*** has scheduled a task that sends a “Check” message to the actor ***CheckUrlToCrawl*** every five minutes by default. Every time the ***CheckUrlToCrawl*** actor receives a “Check” message, it uses ***Wonts data services*** to check on the database ***wontsapp*** if there is some URL that needs to be crawled. In order for ***Wonts data services*** to do this task, it checks in the table ***UrlToCrawl*** if there is any record where the value of the field ***status*** is set to ***isToCrawl***. The service is searching for all the records that are active and with a ***status*** set to ***isToCrawl***. It then returns a list of URLs that should be crawled. The actor ***CheckUrlToCrawl*** sends a message to the actor ***CrawlUrlProvider*** for each URL that needs to be crawled. The ***CrawlUrlProvider*** actor is using an instance of ***Crawler*** class, given by the container Spring. The ***Crawler*** has the ***Robottxt*** dependency class that is used to check which area on the web side can be crawled and which should not[60] be take in consideration. The

Crawler is using the *JenaResourceModel* for creating the OWL file. The crawler also contains an object instance of the Parser class created by the *Spring* container as singleton. The Crawler sets in the Parser object an empty instance of the *JenaResourceModel* and the instance of *Robottxt*. During this process all the links that the *Parser* finds are added to the *JenaResourceModel*. After that, The Crawler calls the method *parse* of the Parse class with the URL to crawl as argument. During crawling and analysing of the URL, the Parse object adds in a list all the URLs that has been found on the page document and that need to be crawled. Each link on the list will be crawled in certain conditions:

- If the Robottxt validation is positive;
- If the URL has not been already crawled;
- If the URL is direct descendent of the starting URL (The Crawl does not crawl any resource that is external to the starting URL)

The Parser object also updates another list with all the URLs that the web resource contains. It uses the list to update the resource ontology model.

At this point for each URL the Parser recursively requests to the Spring container another instance of itself, the container returns a singleton instance of class. The Parser initialises again the object by setting on it all the resources already crawled, the same instance of Robottxt, the starting URL and the OWL resource model. The class has scope singleton to avoid stack overflow and maintains just one instance of Parser object. The Parser collects all the resources it has found in the document parsed. Once the document is completed for each links-out, it adds a triple to the OWL model. This triples contains the URL of the document that has been crawled, the property **LinksOut** and resource found. When there is no more documents to parse the final *JenaResourceModel* model will be used by the *Crawler* object to writes an OWL file. After that, the *Crawler* returns the full name of the OWL file to the *CrawlUrlProvider*.

The actor *CrawlUrlProvider*, then does the following actions:

- Update the URL status in the **wontapp** table as **isCrawled**
- Deletes form the **wonts** database any old triples previously crawled and that are descents of the crawled URL
- Updates the database with the new ontology.

4.2 OWL data manager and provider

Apache Jena functionalities are used in *data.service.jar*. The library *data.service.jar* is the service data provider used in the entire **Wonts** project. This data provider exposes many services methods used to access to database. It is using JPA2 and DI as development software technologies. The databases that it is managing are two:

- **wonts**
- **wontsapp**

The DB **wonts** has all the tables for all the entire triples generating by the crawler.

The structure of **wonts** database makes it possible to store the RDF triples. In terms of performance it is better to store the data in a database because it is quicker to execute a query in a database than execute the same in an xml file. At the moment the project is using MySQL DB but it can easily use

Wonts - Web Ontology Searcher - by Salvatore Rapisarda

another DB or if we have the necessity to store big amounts of information then it is possible to use a NoSQL repository as well. From the application configuration files it is possible to change all the connection strings and the drivers used by the application to access to the database.

The package `wonts.data.services.model` contains all the entities that the project is using such as:

- Ontology
- Project
- Url
- UrlToCraweler
- User

Wonts - Web Ontology Searcher - by Salvatore Rapisarda

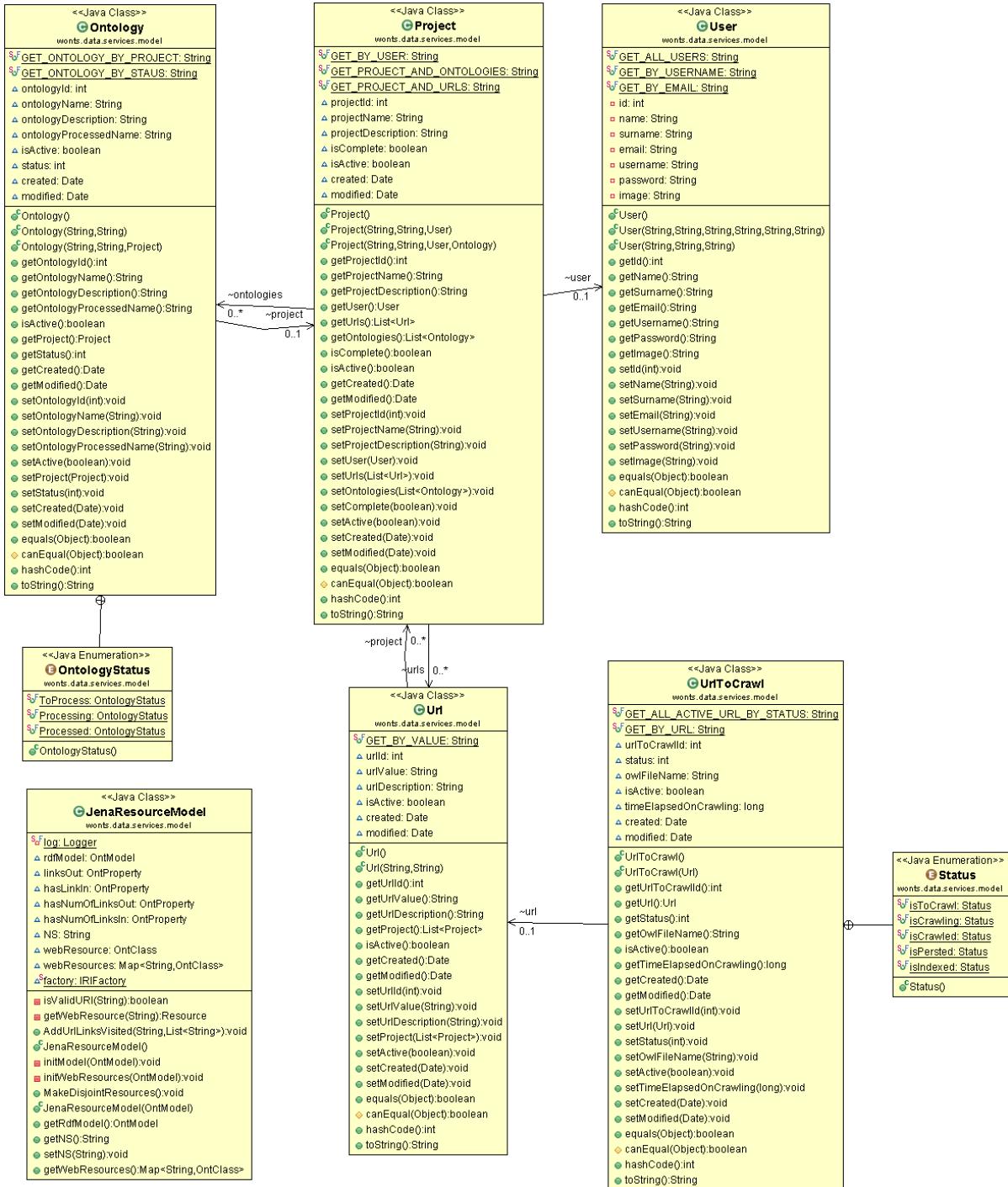


Figure 4-4: Entities Data Model

The figure above shows the entire entities data model that the system is using. It integrates all the POJO classes using java JPA. Using a JPA entity model makes it easier to design a data model without creating any SQL scripts. Therefore JPA has been used to develop most of the data model of this project. The combination between JPA and DI makes distributed applications robust and scalable. It is quite important that the instances of objects are dynamically created and injected by a container. By using DI a class that contains a dependency does not need any knowledge about which is the implementation in

use. It relies just on its interface or abstraction. This makes the project more scalable because any changes or extensions can be easily solved by creating another implementation class.

Spring is the library used in the project to create the Dependency Injection. Spring library during the start up of the application creates a container that is able to resolve all the dependencies that each class needs. If a dependency is missing the container it stops the application to run. The entire **Wonts** system application is using DI, especially in the data.service.jar that is a library shared between all the application modules.

4.3 Apache Solr as Document indexer

A core named **wonts** has been created in Apache Solr to be used by **Wonts** project. This core contains all the necessary configuration files and data structures Apache Solr needs for correctly indexing and searching the required information. When a new core is created a folder is added into "<solar-dir>/server/solar/<core-name>/". In this project the core is located inside the folder "<solar-dir>/server/solar/wonts/". **Core.properties** is an important configuration file and is present in this folder. Inside "**core.properties**" there are all the important property variables that Solr needs:

- **name=wonts** (it sets the name of the core)
- **config=solrconfig.xml** (it sets the name of the configuration file)
- **schema=schema.xml** (it sets the name of the configuration schema file)
- **dataDir=data** (it sets the name of the data directory)
-

One of the most important file descriptor is the schema.xml. The content of this file defines all the fields to index and the relative configuration. A part of this xml configuration file is reported here:

```
<schema name="wonts" version="1.5">
    <field name="_version_" type="long" indexed="true" stored="true"/>
    <field name="_root_" type="string" indexed="true" stored="false"/>
    <field name="id" type="string" indexed="true" stored="true" required="true"
        multiValued="false" />
    <field name="url" type="string" indexed="true" stored="true" required="true"
        multiValued="false" />
    <field name="content" type="text_general" indexed="true" stored="false" required="true"
        multiValued="false" />
    ...
    ...
    ...
    <uniqueKey>id</uniqueKey>
    ...
    ...
    ...
</schema>
```

An important field described in the file is the **content**. This field is indexed but not stored and is not key sensitive because it is a type **text_general**.

4.4 Service Indexer

All the processes described above are managed by using Akka actors. The following UML sequence diagram graphically explains in more detail how the actors interact to get, download, analyse, index and delete each one the crawled resources. All the processes are done in concurrency, when the actor receives a message checks and executes the request. An actors system makes it easier to create concurrent and distributed systems where each components receives and/or sends messages.

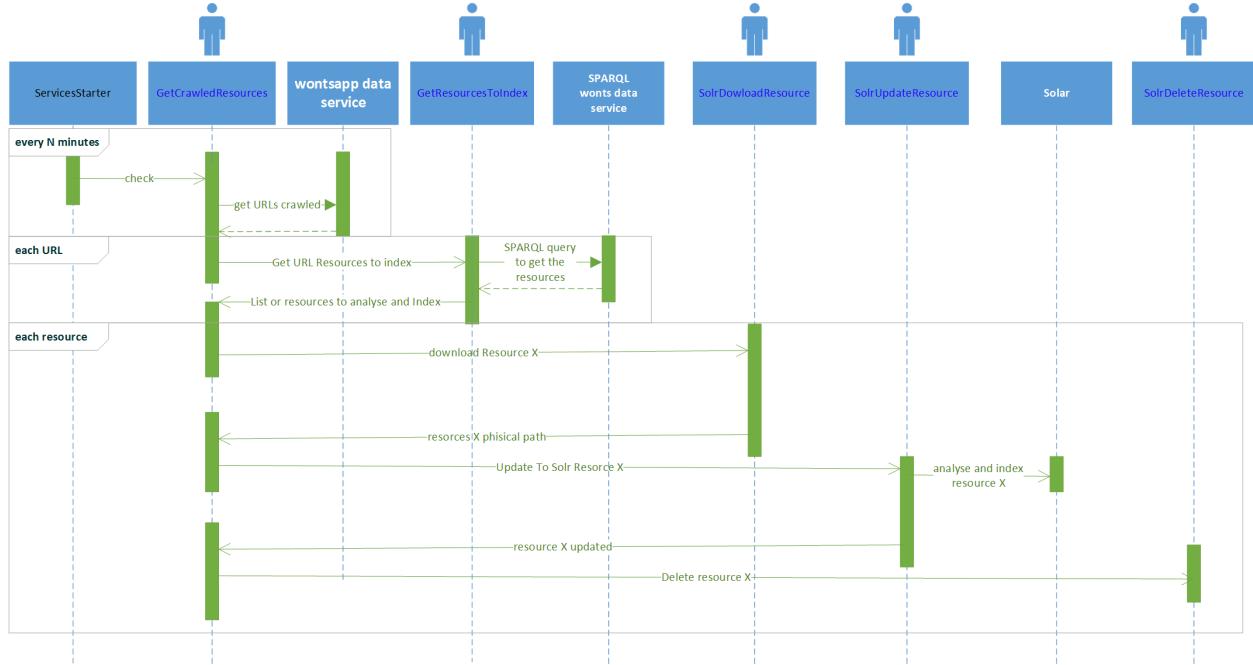


Figure 4-5: Service Indexer ULM Diagram

As described above in the sequence diagram there is a scheduled task that fires every five minutes by default. This scheduled process sends a message to the actor *GetCrawledResources* for checking if there are some URLs that have been crawled. It gets all this information from the service data provider. The service returns, if any, a list of strings, which represents all the crawled URLs. For each URL it sends a message to the actor *GetResourcesToIndex* to get all the resources that has been crawled using the URL. To do this task the actor uses the SPARQL data service. It queries the database **wonts** using SPARQL and Apache Jena library. Obtained the list of resource to index, the actor *GetResourcesToIndex* sends a message to the actor *GetCrawledResources* with a list of web resources to analyse and index. The actor *GetCrawledResources* for each web resource sends a message to the *SolrDowloadResource* that downloads the resource in local and sends a message to *GetCrawledResources* for updating the resource that has been downloaded. The actor *GetCrawledResources* sends a message to the actor *SolarUpdateResource*, for updating the resource by sending its content to Apache Solar. *SolarUpdateResource* sends the content to Apache Solr and after sends a message to *GetCrawledResources* for deleting the updated resource. The actor *GetCrawledResources* sends another message to *SolrDeleteResorces* for deleting the resource.

Above has been synthetically explained how the actors communicate each other during the process of indexing the web resources. The entire messages sent from actor to actor are asynchronous. The only process synchronous is made just when the actor needs information from the data service providers. This asynchronous messaging between actors made this service indexer a full concurrency system, therefore more efficient in time and resource allocated.

4.5 Search Engine Service

This process of searching and updating an ontology model is done by the actor ***processOntology*** that is fired every thirty seconds by default. This actor checks if there is any ontology to process. It is using the **wontsapp** data service (***data.services.jar***) to get the list of ontology to process. For each ontology it sends a message to the actor ***ReadOwlFile*** that reads and adds individuals to the given ontology model. ***ReadOwlFile*** executes the following steps:

- It queries the OWL ontology file in order to get a result-set that contains a search query and its relative class.
- It sends a message to the actor ***SolrQueryEngine***, for each pair of class and query string
- It waits for a response that contains the individuals to add in the class model.

SolrQueryEngine posts a request contained the searching query string to Apach Solr. When it receives back the response from Apache Solr, it sends a message as response back to ***ReadOwlFile*** that is waiting for the results. At this point ***ReadOwlFile*** actor adds the individuals received to the ontology class. This process terminates when the actor has checked the entire classes the result-set contains.

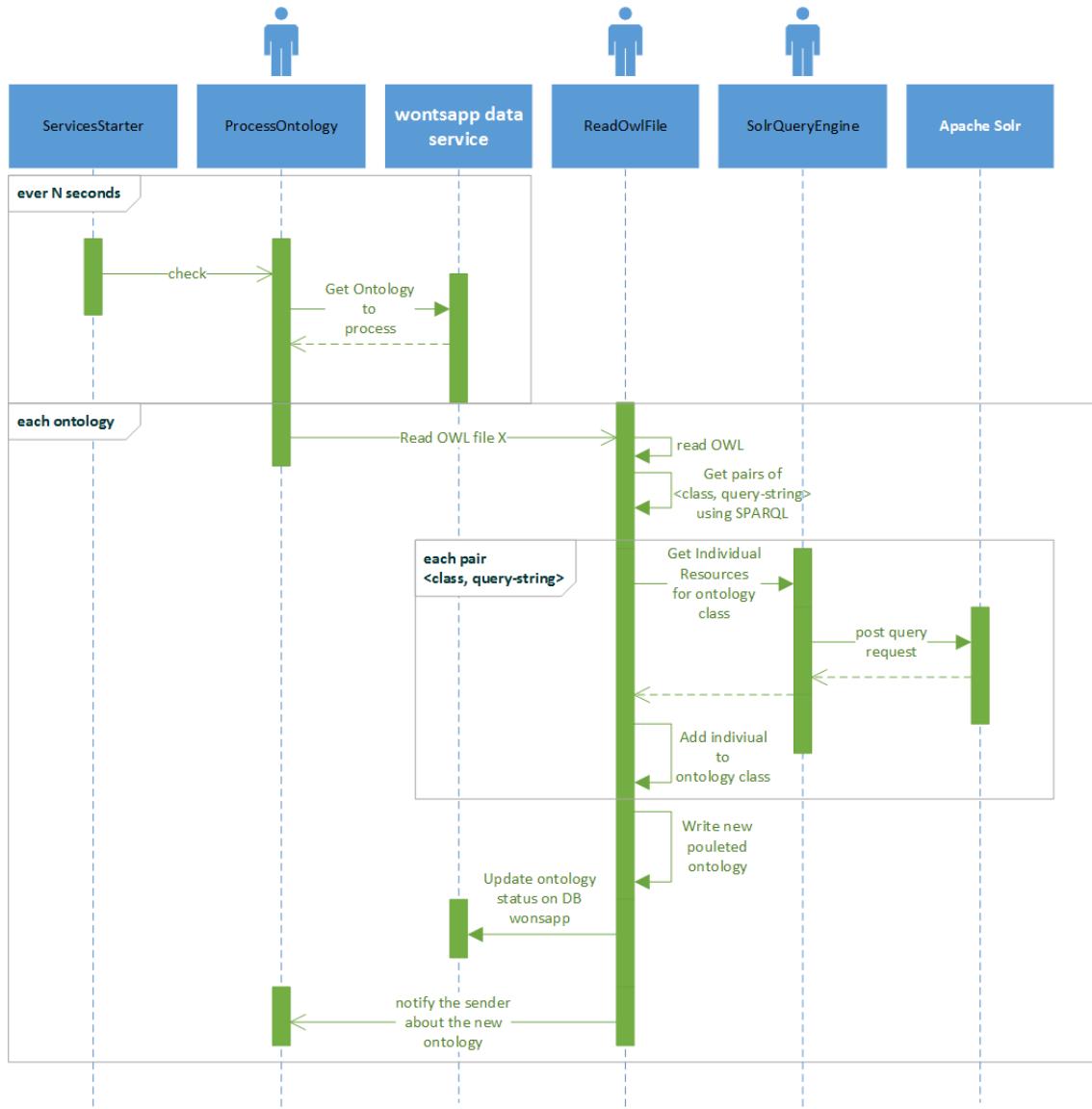


Figure 4-6: Search Engine Service UML Diagram

The sequence diagram above explains graphically all the actions used for populating and ontology model. This process makes it possible to automatically populate an empty ontology, with individuals by using the property: **$\exists \text{hasSearchQuery}.\{\text{content}:\text{"query to search"}\}$** .

It is evident that adopting an ontology model as a query of research is an advantage, because an ontology contains a knowledge base with an already defined Terminology (TBox). What this part of the application does is to populate the ontology Data Assertion (ABox).

4.6 Web Application

Until now has been described how Wonts crawls, indexes and populate our ontology with the web searched resources. Another important part of the project is the web application that is the interface between the user and the search engine.

Wonts - Web Ontology Searcher - by Salvatore Rapisarda

The picture below shows and graphically describes the high architectural design used in the **Wonts** web application.

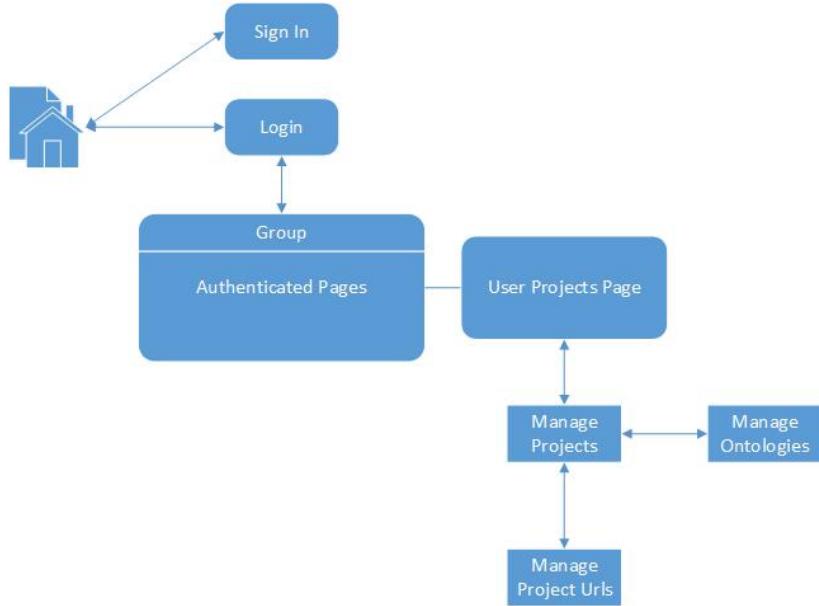


Figure 4-7: Web Application Structure

The picture above shows the basic structure used in the web application that has just has two public functionalities, which are the “login” and the “sign in”. The sign in is used to create a new account whereas the login is used to access to the authenticated web application part. The web application looks like the image below. It has a top menu where is possible to “Sign In” or “Log In” if we have already an account.

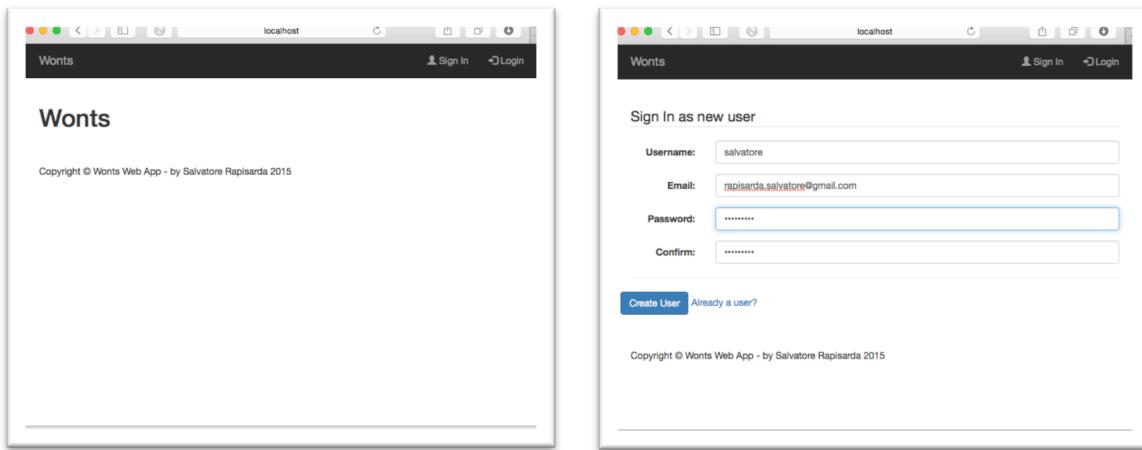


Figure 4-8: Home and Sign In Pages

To use the application it is necessary to create an account. By selecting in the label “Sign In” on the left side of the top menu, it is possible to create a new account. The form requires an username that is not less than six characters, a valid email and a password, that must have minimum 6 characters an upper-case letter and at least a number (es: Password1). The form will be asked to confirm the password as

Wonts - Web Ontology Searcher - by Salvatore Rapisarda

well. By clicking on the “Create User” button the application creates a new user on DB and redirects to the login page for the user to be authenticated.

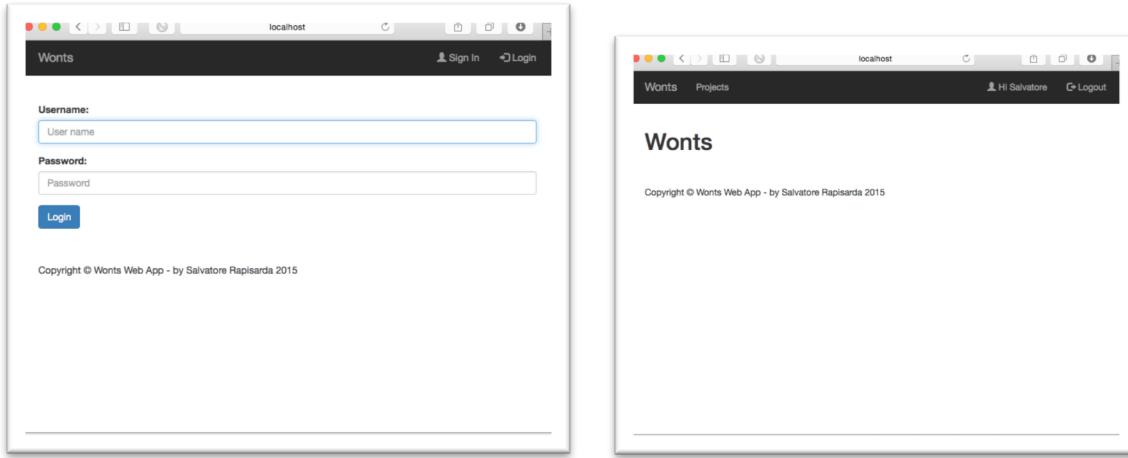


Figure 4-9: Login and Project Menu Pages

In the login page is necessary to add the user-name and the password to be logged in. After the log in on the top left will be visible the voice “Projects”, by click on it the application redirects in a projects page where it is possible to create a new project or select an existing one.

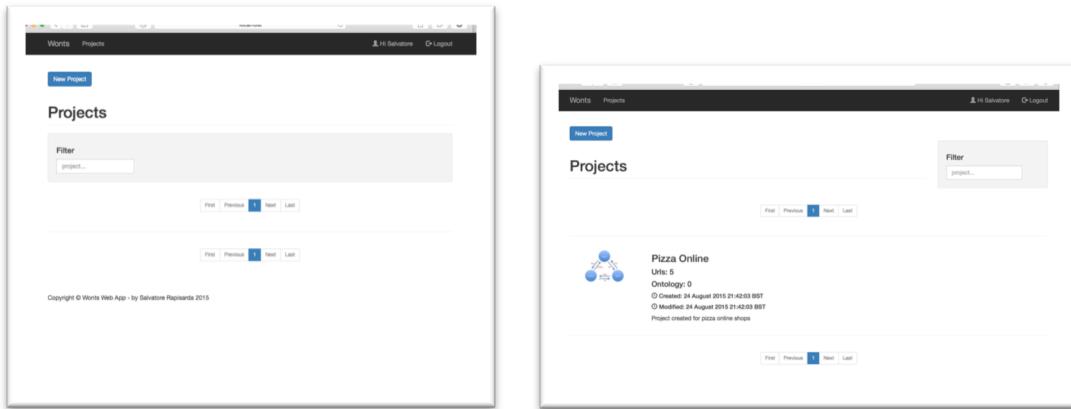


Figure 4-10: Project Pages

By clicking on the button “New Project” the application redirects to a page where it is possible to add a name and a description to the new project. The project name is a required field and should contain at least a character. The Description is mandatory as well and cannot be less than six characters.

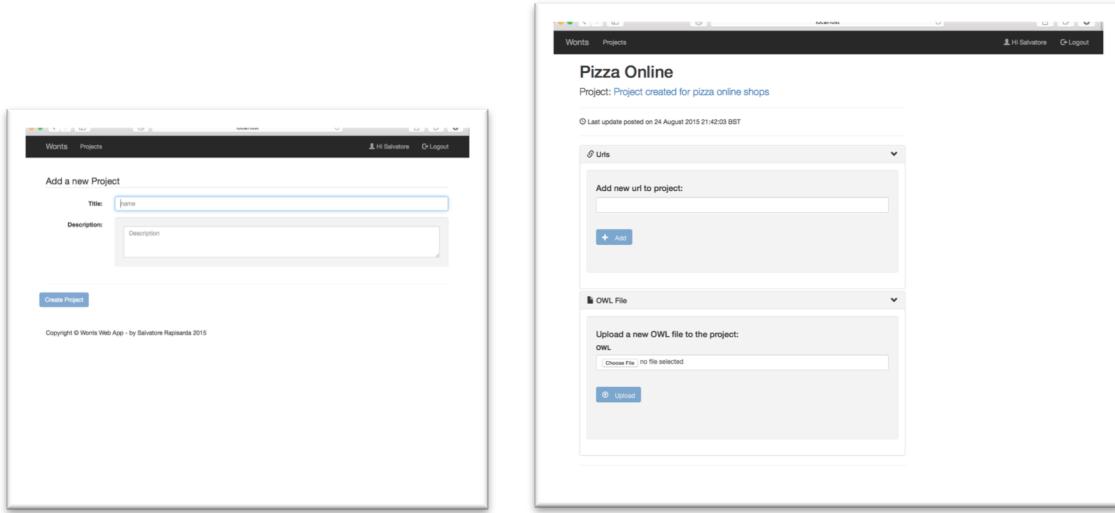


Figure 4-11: Insert URL and OWL File in a Project

After the user clicks on the button “Create Project”, the application creates a new project and redirects to another page where it is possible to manage the project URLs and ontology files. The project needs URLs in order to populate the ontology of individual resources. The users can have as many projects as they need. There is not restriction in the number of projects the users can add or the number of URL and ontology they can add to a single project.

If it is not already present and crawled, every time the user adds a new URL to the project it is crawled and successively indexed by the system. In a project created by the user it is possible to disable or enable a URL and to delete or add ontology. It is always possible to upload ontology as OWL file or delete and resubmit again. In the web application it is possible to check the status of the URLs. Therefore the user can decide to upload the ontology when the URLs are crawled and indexed. The ontology added to a project is the base model for a web research. The web application is the user interface and helps to manage the user’s projects and ontology.

5 Evaluation

In this chapter I will discuss the tests executed and a critical evaluation and restrictions about the system **Wonts**.

5.1 Tests

The test is divided into functional and unit tests. For testing the project functionalities two tests have been created and evaluated. These two tests validates the system functionalities in its overall. During each test a data evaluation has been dedicated and the second test shows some measures of the system performances as well. This functional tests are also verified the hypothesis asserted. The unit tests instead, are divided per solution project. They are asserting that all the Java code is working as expected. By using TDD approach these tests have been continually executed during the development.

5.1.1 Financial Ontology Test

A very easy example of a test can be the financial ontology that is mention at the beginning of this report. This ontology as explained before is very basic and can be represented graphically as the image below.

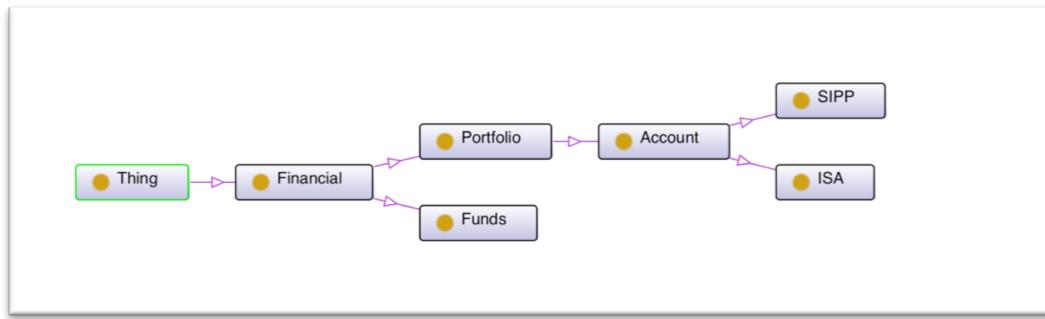


Figure 5-1: Financial Ontology Class View

5.1.1.1 Ontology Schema

The ontology has been defined as:

$$I = (\Delta^1, ^1), \quad KB = \mathcal{T}box + \mathcal{A}box, \quad I \models KB$$

$\mathcal{T}box$:

$$\top \sqsubseteq \text{Financial}$$

$$\text{Financial} \sqsubseteq \text{Portfolio} \sqcup \text{Funds}$$

$$\text{Portfolio} \sqsubseteq \text{Account}$$

$$\text{Account} \sqsubseteq \text{SIP} \sqcup \text{ISA}$$

$$\text{Funds} \sqsubseteq \forall \text{hasSearchQuery.} \{ \text{"content:financial AND content:fund"} \}$$

$$\text{ISA} \sqsubseteq \forall \text{hasSearchQuery.} \{ \text{"content: financial AND content: isa"} \}$$

$$\text{OR (content:account AND content: isa) }$$

$$\text{SIPP} \sqsubseteq \forall \text{hasSearchQuery.} \{ \text{"content: financial AND content: isa"} \}$$

$$\text{OR (content:account AND content: sipp) }$$

The ontology does not contain any individual (*ABox*), but we have defined a *Tbox*, therefore to have a complete knowledge base (KB = *Tbox* + *ABox*) it is possible to use Wonts.

5.1.1.2 Adding URLs To Crawl

First of all we need to create a project in the web application **Wonts** for adding all the URLs that will be the base for the individual resources of our research. In this example we have chosen the following URLs:

- <http://www.bestinvest.co.uk/our-services>
- <https://www.moneyadviceservice.org.uk/en>
- <http://www.tddirectinvesting.co.uk>

When the entire URLs have finished indexing by the system, this will be the moment to upload the ontology file. We need then to wait some seconds until it is populated with the desired individuals. At this point the new ontology contained the individuals will be available to download. It is possible to use Protégé and execute some query on it using SPARQL language or DL searching tools.

5.1.1.3 Data Evaluation

For example it is possible to check how many individual each ontology class contains. In this example shown below, we want count how many individuals are type ISA by using the following SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX fin: <http://titan.dcs.bbk.ac.uk/~srapis01/finance#>
SELECT (count(distinct ?individual) as ?numcount)
WHERE { ?individual rdf:type fin:ISA }
```

We will notice that after the execution the count number received is ?num=906. If we execute the same query for another financial product we will find that for SIPP the ?num=407 and for Funds is 883. If we also want to execute another query that evaluates where the class is the union of the three classes, than we will find that the ?num=278 common individual resources.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX fin: <http://titan.dcs.bbk.ac.uk/~srapis01/finance#>
SELECT ?num (count(distinct ?individual) as ?numcount)
WHERE { ?individual rdf:type fin:Funds ;
        rdf:type fin:ISA ;
        rdf:type fin:SIPP } group by ?num
```

If we want to find the total number of individual we already know that the individuals have been added to each class that has the annotation property **hasQueryString** (ISA, SIPP and Funds). Therefore, the query that should be executed can be:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX fin: <http://titan.dcs.bbk.ac.uk/~srapis01/finance#>
```

```
SELECT (count(distinct ?individual) as ?numcount)
WHERE {
    {?individual rdf:type fin:SIPP }
    UNION { ?individual rdf:type fin:ISA }
    UNION { ?individual rdf:type fin:Funds }
}
```

The result of this query is 1077 and we can do the same using DL query in Protégé. If we ask in DL query all the Instances of the class Financial we will find 1077. Therefore we can use DL query to get inferred results.

Using this kind of approach creates taxonomy on the ontology individuals. Therefore it can be used to classify and categorise the web resources by their nature, more quickly than before.

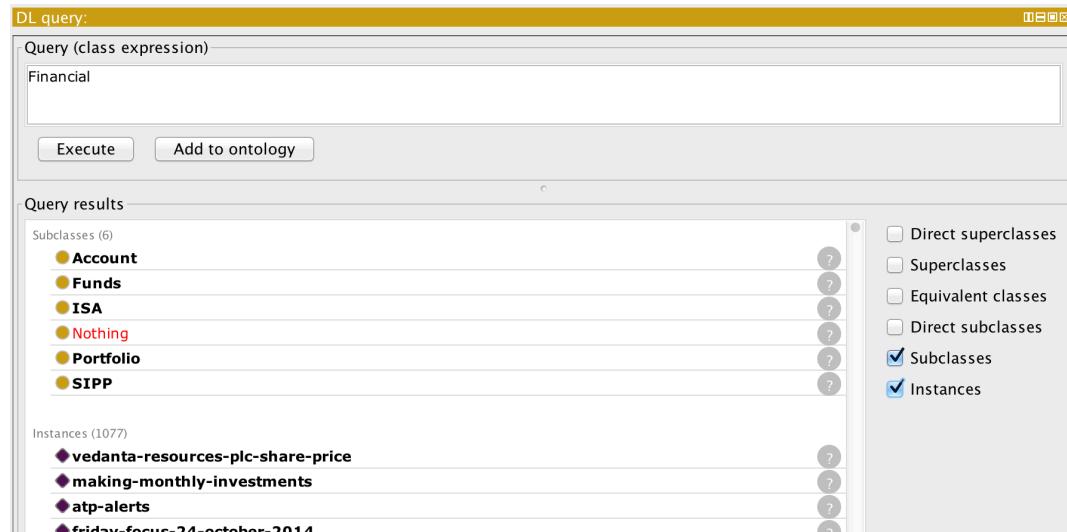


Figure 5-2: Using DL in Protégé

5.1.2 Pizza-Shop Ontology Test

In the case above has been created an ontology about financial products but we can create other kinds of ontology completely different then the one before.

We can easily create an ontology that describes a “pizza on line shops”. For example, because we want to search about some shops on the online market that sell pizza. For instance we are doing this research because, we want to open a web site that sells pizza on line or just because we would like to make a research on this field. However we want to improve our knowledge on the pizza online shops.

First of all it is necessary to create a specific ontology that describes a “pizza online shop” to use in our research. The ontology should have a class that is the concept of pizza, product, menu and price. It can include offers and deals. Where the shop sells drink, which category of pizza it sells: Vegetarian, biological gluten free. It is also necessary to use the property **hasSearchQuery** for associating all the desired classes with a query string. This ensures that the classes are populated with the desired individuals. In order for **Wonts** system engine to add individuals to the ontology classes the annotation property **hasSearchQuery** must be present. Therefore, it is necessary to add the annotation property **hasQueryString** in each class where is required. It is also important that the query string is created in base of the class nature and topic of research.

I am now going to define an example of an ontology that contains some of the classes and concepts we intend to describe. For instance the following can be a possible ontology to use for our test.

5.1.2.1 Ontology schema

Table 5-1: Pizza Shop Ontology Schema

Self-standing	Relations	Definable	Modifier
<ul style="list-style-type: none"> • Offer <ul style="list-style-type: none"> ◦ Latest ◦ Deal • Product <ul style="list-style-type: none"> ◦ Pizza ◦ Drink • Menu <ul style="list-style-type: none"> ◦ MenuItem • Location • Order <ul style="list-style-type: none"> ◦ Collected ◦ Delivered 	<ul style="list-style-type: none"> • hasSearchQuery • identifies • hasSlices • costs • pays 	<ul style="list-style-type: none"> • Price • Menulterm • PostCode • Money 	<ul style="list-style-type: none"> • PizzaCategory <ul style="list-style-type: none"> ◦ PizzaVegetarian ◦ PizzaBiological ◦ PizzaGlutenFree • PizzaSize <ul style="list-style-type: none"> ◦ PizzaSmallSize ◦ PizzaMediumSize ◦ PizzaLargeSize

5.1.2.2 Relations

Table 5-2: Pizza Shop Ontology Relations

Property Name	Domain	Range	Type
hasSearchQuery	Class	T	not functional not symmetric not transitive
identifies	PostCode	Location	not functional Symmetric not transitive
hasPrice	Product Order	Price	functional not symmetric not transitive
isValueOf	Price	Product Order	functional not symmetric not transitive
pays	Money	Order	not symmetric not functional not transitive
isPaidBy	Order	Money	inverse of pays

5.1.2.3 Definable

PizzaSize \sqsubseteq *PizzaSmallSize* \sqcup *PizzaMediumSize* \sqcup *PizzaLargeSize*

Price \equiv $\exists \text{isValueOf}.\text{Product}$

MenuItem \equiv *Product* \sqcap $\exists \text{hasPrice}.\text{Price}$

PostCode $\equiv \exists identify.Location$

Money $\sqsubseteq \exists pays.Order$

Order $\sqsubseteq \exists isPaidBy.Money \sqcap \exists hasPrice.Price$

Offer $\sqsubseteq \exists hasPrice.Price$

5.1.2.4 Annotation property *hasSearchQuery* definitions

Pizza $\sqsubseteq \exists hasSearchQuery.\{content:pizza\}$

Offer $\sqsubseteq Pizza \sqcap \exists hasSearchQuery.\{content:Pizza AND content:offer\}$

Latest $\sqsubseteq Offer \sqcap \exists hasSearchQuery.\{content:Pizza AND (content:latest OR content:new)\}$

Deal $\sqsubseteq Offer \sqcap \exists hasSearchQuery.\{content:Pizza AND content:deal\}$

Drink $\sqsubseteq \exists hasSearchQuery.\{content:drink\}$

Menu $\sqsubseteq \exists hasSearchQuery.\{content:menu\}$

Price $\sqsubseteq \exists hasSearchQuery.\{content:£\}$

Location $\sqsubseteq \exists hasSearchQuery.\{content:pizza AND (content:contact OR content:address)\}$

Order $\sqsubseteq \exists hasSearchQuery.\{content:pizza AND content:order\}$

Collected $\sqsubseteq \exists hasSearchQuery.\{content:pizza AND content:order AND (content:collect* OR content:"take away")\}$

Delivered $\sqsubseteq \exists hasSearchQuery.\{content:deliver*\}$

PizzaSmallSize $\sqsubseteq Pizza \sqcap \exists hasSearchQuery.\{content:"4 slice*"\}$

PizzaMediumSize $\sqsubseteq Pizza \sqcap \exists hasSearchQuery.\{content:"6 slice*"\}$

PizzaLargeSize $\sqsubseteq Pizza \sqcap \exists hasSearchQuery.\{content:"8 slice*"\}$

PizzaVegetarian $\sqsubseteq Pizza \sqcap \exists hasSearchQuery.\{content:Vegetar*\}$

PizzaBiological $\sqsubseteq Pizza \sqcap \exists hasSearchQuery.\{content:Bio*\}$

PizzaGlutenFree $\sqsubseteq Pizza \sqcap \exists hasSearchQuery.\{content:gluten* AND content:free\}$

(The symbol * is a wildcard that can be used to substitute for any other character in a query string)[61,62]

If we want to define the query string for *PizzaGlutenFree* we have to make the definition of the query string explicit for Apache Solr :

1. *PizzaGlutenFree* $\sqsubseteq Pizza \sqcap \exists hasSearchQuery.\{content:gluten* AND content:free\}$
 - 1.1. *Pizza* $\sqsubseteq \exists hasSearchQuery.\{content:pizza\}$
2. *PizzaGlutenFree* $\sqsubseteq \exists hasSearchQuery.\{content:pizza\} \sqcap \exists hasSearchQuery.\{content:gluten* AND content:free\}$
3. *PizzaGlutenFree* $\sqsubseteq \exists hasSearchQuery.\{content:pizza AND content:gluten* AND content:free\}$

The class *PizzaGlutenFree* can be associated to the query string: "content:pizza AND content:gluten* AND content:free". It is very important that the scope of research it is well defined this, for the search engine fetching appropriate individuals to add to our ontology classes. After a possible ontology has been defined, it is necessary to know which are the URLs we intend to crawl and index.

5.1.2.5 Create Wonts Project

At this point we need to use the web application Wonts. Firstly it is necessary to create a new project and add all the URLs and domains we intend to crawl. This step is necessary for defining the domain of research and it makes sure that our research is based on the URLs/domains inserted in the project. If the web contents is not already present this will be crawled and index by **Wonts** services.

5.1.2.6 Test

Finally, when the service engine has finished the process of crawling and indexing the entire content web documents it will be possible to upload the “pizza shop” ontology created for this reason. After we have waited some minutes it will be possible to download a new ontology file contained the individuals. The service is checking every 30 seconds if there is some ontology that needs to be processed.

At this point the downloaded ontology will be full of individuals (\mathcal{ABox} defined) so it will be possible to execute different SPARQL queries to search through the data and use a semantic reasoning to resolve and infer all the logical consequences[63] (\mathcal{L}) generated in the ontology model.

For the pizza example the URLs added to the project are:

- <http://www.pizzahut.co.uk>
- <http://www.pizzaexpress.com>
- <https://www.dominos.co.uk>

It is important to index URLs that are related with the ontology nature and scope of research. Crawling and indexing the URLs below in our laboratory environment has produced the data below, where the time is expressed in milliseconds:

Table 5-3: URLs Example Used in Pizza Shop Ontology

URL	Time Elapsed On Crawling	Time Elapsed On Indexing	Total Resource percentage	Total Resources
www.pizzahut.co.uk	394607ms	420397ms	68.8%	2456
www.pizzaexpress.com	70790ms	77493ms	5.0%	179
www.dominos.co.uk	296421ms	224021ms	26.2%	937

The time for crawling and indexing all the resources depends prevalently upon two factors, network speed and server machine hardware. For all the tests described in this report, this are the virtual machine’s details:

Table 5-4: Virtual Machine’s Details

Operating System	Linux srproject 3.13.0-32-generic #57~precise1-Ubuntu x86_64 x86_64 x86_64 GNU/Linux
CPU	AMD Opteron(tm) Processor 6376- 64 bits
Network Speed on download	3.19MB/s

Physical Memory	8GB
-----------------	-----

It is obviously evident that have a good download speed and a good hardware drastically improves the performance of the download and the index of the web resources. It is also necessary that Apache Solr has enough space for indexing all the document resources.

After we have uploaded the ontology model that represent the “pizza shop” it is possible to download a new ontology contained the following general ontology metrics shown in the table below:

Table 5-5: General Ontology Metrics

Axiom	15524
Logical axiom count	15475
Class count	24
Object property count	6
Data property count	0
Individual count	2780

All the individuals by class type obtained are summarised in the table below that categorise our research:

Table 5-6: Individuals Grouped by Class

Class	Individuals
Collected	857
Deal	123
Delivered	1238
Drink	842
Latest	38
Location	2679
Menu	1979
Offer	138
Order	1966
Pizza	2708
PizzaGlutenFree	39
PizzaLargeSize	17
PizzaMediumSize	19
PizzaSmallSize	6
PizzaVegetarian	70
Price	2708

By using Protégé to analyse the data it shows that many elements are in common to other classes of different nature. In this web research example it is difficult to create disjoined classes, especially when the resources to consider are huge amounts of individuals. Most of the individuals that the ontology

contains are from www.pizzahut.co.uk, but this is evident because the pages indexed are 68.8% of the total individuals. In this case www.pizzahut.co.uk contains more public pages or the web site it is bigger than the others two or the other two web sites have other subdomains that has not been considered.

For example it is possible to check how many offers there are in each web site. This data is easy to obtain by executing a SPARQL query such as:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX pizza: <http://titan.dcs.bbk.ac.uk/~rapis01/pizzashop#>
SELECT (count(distinct ?individual) as ?numOffer)
WHERE { ?individual rdf:type pizza:Offer
        FILTER regex(str(?individual), "www.pizzahut.co.uk", "i") }
```

We have found that for www.pizzahut.co.uk there are 9 occurrences and if we repeat for all the other websites and classes the following table will be obtained as results:

Table 5-7- Differences Between Three Domains Sites

Class	www.pizzahut.co.uk	www.pizzaexpress.com	www.dominos.co.uk	Individuals
Collected	7	4	846	857
Deal	3	19	101	123
Delivered	368	25	845	1238
Drink	9	3	830	842
Latest	8	27	3	38
Location	1793	20	886	2679
Menu	1007	117	885	1979
Offer	9	27	102	138
Order	1091	25	850	1966
Pizza	1796	46	866	2708
PizzaGlutenFree	25	14	0	39
PizzaLargeSize	17	0	0	17
PizzaMediumSize	19	0	0	19
PizzaSmallSize	6	0	0	6
PizzaVegetarian	63	6	1	70
Price	1796	46	866	2708

We can see from the table above the differences between the three domains sites. We can also see that for the classes PizzaLargeSize, PizzaMediumSize and PizzaSmallSize have been found just in one of the web site domain.

In protégé we can also use Hermit[59], that is a plugin for reasoning all the ontology logical consequences and verify that the ontology is consistent. We can verify by using DL in protégé that the class **MenuItem** defined above as **MenuItem** ≡ **Product** □ $\exists \text{hasPrice} . \text{Price}$ now contains 1979 individuals or instances. To ensure that the model remains consistent after it has been populated with individuals, it is necessary to do an accurate analysis on what the class should contains and which is our TBox. For example a relation where a class should contains an exact number of elements it is difficult to be satisfied, because during the web research the possible individuals that have been found are more than

we have previously defined. In this case we end up with and inconsistency in our ontology. Therefore, when we do not know the amount of individuals that a class contains, it is better to do not define a rule that can be incongruous. In our case the number of individuals that the research return is quite difficult to previously estimate. At least it is possible to make a statistical forecast based on the data already know.

In the web application it is also possible to upload to a project how many ontologies we want. However, the same URLs can be used in others ontology as well. In a project it is also possible to enable or disable the URLs in contains. This functionality is useful in case we want that the ontology contains just individuals that are the direct descendants of the enabled URLs.

5.1.3 Unit testing

All the system has been developed using Test Driven Development (TDD) therefore during the development all the tests have been continuously executed and integrated. TDD development methodology ensures the consistency and correctness of the written code. In order to test and assert the validity of each main code functionality there is a unit test. The unit tests are present in each project solution. The JUnit[19] class files are contained in the folder "src/main/java". This is a standard maven folder where to add the unit test classes the project is using. It is possible to execute the test in each project solution by using by the command: "mvn test". By running this command all the tests that the project contains will be executed.

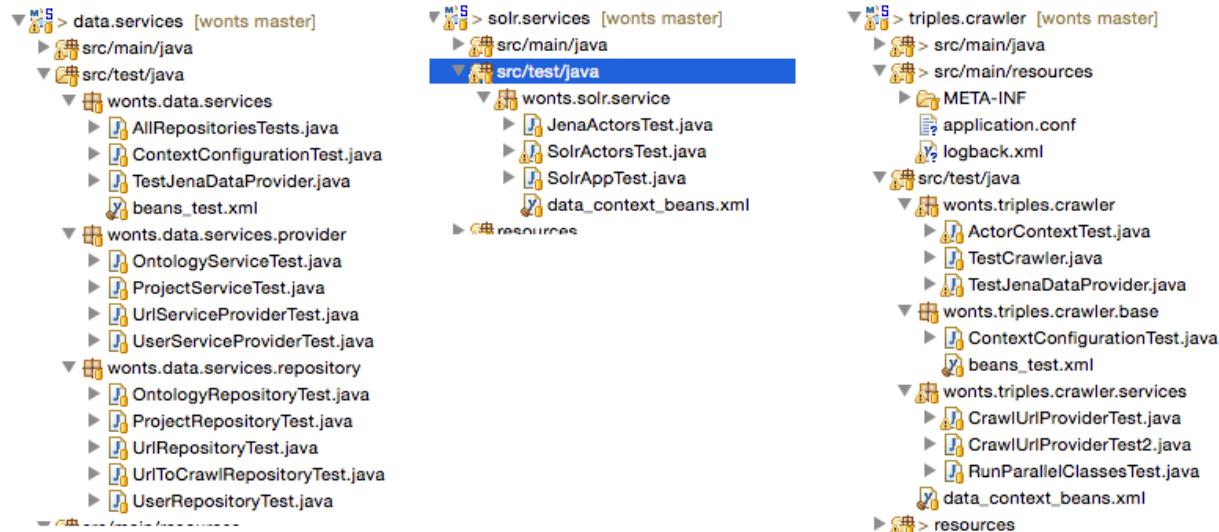


Figure 5-3: Wonts Unit Test Classes

The image above is showing three projects and their relative test classes. Each class contains many tests to assert the correctness of the code. The structure of the test packages is the same as we find in the folder "src/java/main". The file name contains a suffix or prefix "Test".

The following is a report of a table that contains the test executed in **Wonts**:

Table 5-8: data.services Unit Tests

data.services		
Class	Test Method	Pass/Fail

OntologyRepositoryTest	insertNewProject	PASS
OntologyRepositoryTest	getOntologyByIdTest	PASS
OntologyRepositoryTest	setActiveOntologyTest	PASS
OntologyRepositoryTest	ontologyIsActiveByDefaultTest	PASS
OntologyRepositoryTest	getOntologiesStatusTest	PASS
ProjectRepositoryTest	insertNewProject	PASS
ProjectRepositoryTest	getProjectById	PASS
ProjectRepositoryTest	deleteProject	PASS
ProjectRepositoryTest	addOntologyToProject	PASS
UrlRepositoryTest	insterNewUrlTest	PASS
UrlRepositoryTest	getUrlByIdTest	PASS
UrlRepositoryTest	getUrlByValueTest	PASS
UrlToCrawlRepositoryTest	insertUrlToCrawlTest	PASS
UrlToCrawlRepositoryTest	insertUrlToCheckIsActiveByDefaultTest	PASS
UrlToCrawlRepositoryTest	getUrlToCrawlTest	PASS
UrlToCrawlRepositoryTest	getAllActiveUrlToCrawl	PASS
UserRepositoryTest	testCreateNewUser	PASS
UserRepositoryTest	test GetUserByEmail	PASS
UserRepositoryTest	test GetUserByUserName	PASS
UserRepositoryTest	testIsUserNameNotAvailable	PASS
UserRepositoryTest	testIsUserNameAvailable	PASS
OntologyServiceTest	GetAllActiveOntologiesTest	PASS
OntologyServiceTest	CheckProjetInAllActiveOntologiesTest	PASS
ProjectServiceTest	addProjectTest	PASS
ProjectServiceTest	getProjectTest	PASS
ProjectServiceTest	getProjectByUserTest	PASS
UrlServiceProviderTest	updateUrlToCrawl	PASS
UrlServiceProviderTest	insertNewUrlToCrawl	PASS
UrlServiceProviderTest	getAllActiveUrlToCrawl	PASS
UrlServiceProviderTest	insertUrlToCrawlTestNoDuplicatesUlr	PASS
UserServiceTest	test UserServiceProviderBeNull	PASS
UserServiceTest	testCreateNewUser	PASS
UserServiceTest	testCreateNewFullUser	PASS
UserServiceTest	testCreateUserLessThan6chars	PASS
UserServiceTest	testPasswordUserLessShuldContainNumber	PASS
UserServiceTest	testUserNameShouldNotBeEmpty	PASS
UserServiceTest	testEmailShouldNotBeEmpty	PASS
UserServiceTest	testPasswordShouldNotBeEmpty	PASS
UserServiceTest	test GetUserByUsername	PASS
UserServiceTest	test GetUserByEmail	PASS
UserServiceTest	testSaveUpdate	PASS
UserServiceTest	testIsUsernameAvailable	PASS
UserServiceTest	testIsUsernameNotAvailable	PASS
UserServiceTest	test GetAllUsersInDB	PASS
TestJenaDataProvider	getLinkOutFromFile	PASS
TestJenaDataProvider	countDuplicates	PASS

TestJenaDataProvider	TestRdfDataProvider	PASS
TestJenaDataProvider	TestPersistencePropertyFile_DB	PASS
TestJenaDataProvider	TestPersistencePropertyFile_URL	PASS
TestJenaDataProvider	TestPersistencePropertyFile_DB_USER	PASS
TestJenaDataProvider	TestPersistencePropertyFile_CLASS_NAME	PASS
TestJenaDataProvider	TestPersistencePropertyFile_PROPERTY_FILE_NAME	PASS
TestJenaDataProvider	TestPersistenceWriteOnDB	PASS
TestJenaDataProvider	TestResourcesModelNotEmpty	PASS
TestJenaDataProvider	TestPersistenceQueryResultFormattedAsTripleLinkin	PASS
TestJenaDataProvider	TestPersistenceQueryResultFormattedAsTriple_Filte red_LIMIT	PASS
TestJenaDataProvider	TestPersistenceQueryResultFormattedAsTriple_mod el	PASS
TestJenaDataProvider	TestPersistenceQueryResultFormattedAsResulset_m odel	PASS
TestJenaDataProvider	TestPersistenceQueryResultFormattedAsTriple_Filte red_Property	PASS
TestJenaDataProvider	TestPersistenceQueryResultFormattedAsTriple_Filte red	PASS
TestJenaDataProvider	TestPersistenceQueryResultFormattedAsTriple	PASS
TestJenaDataProvider	TestPersistenceQuery	PASS
TestJenaDataProvider	TestDeleteUrl	PASS
TestJenaDataProvider	TestMakeDisjoinResourcesModel	PASS

Table 5-9: solr.services Unit Tests

solr.services		
Class	Test Method	Pass/Fail
JenaActorsTest	getResourcesFromCrawledUrl	PASS
JenaActorsTest	AddTwoURLToIndex	PASS
JenaActorsTest	getCrawledResource	PASS
JenaActorsTest	ReadOwlFileTest	PASS
JenaActorsTest	ReadOwlFileDomainsTest	PASS
SolrActorsTest	testDownloadActor	PASS
SolrActorsTest	testSendFileToSolarActor	PASS
SolrActorsTest	testSendFileDocToSolr	PASS
SolrActorsTest	testSendFilepdfToSolr	PASS
SolrActorsTest	testRemoveResourceActor	PASS
SolrActorsTest	testDownloadSendFileToSolarAndDeleteItActor	PASS
SolrActorsTest	testSolrQuery	PASS
SolrActorsTest	AddTwoURLToCrawl	PASS
SolrAppTest	testPropertyManager_SOLR_CORE_ENDPOINT	PASS
SolrAppTest	testPropertyManager_SOLR_RESOURCES_PATH_DO WNLOAD	PASS

Table 5-10: triples.crawler Unit Tests

triples.crawler		
Class	Test Method	Pass/Fail
ActorContextTest	<u>makeOntologyTest</u>	PASS
TestCrawler	<u>testCrawler</u>	PASS
TestCrawler	<u>testReplaceInternalLink</u>	PASS
TestCrawler	<u>testReplaceInternalLink2</u>	PASS
TestCrawler	<u>testReplaceInternalLink3</u>	PASS
TestCrawler	<u>testReplaceInternalLink4</u>	PASS
TestCrawler	<u>testReplaceInternalLink5</u>	PASS
TestCrawler	<u>testReplaceInternalLink6</u>	PASS
TestCrawler	<u>testReplaceInternalLink7</u>	PASS
CrawlUrlProviderTest	<u>AddTwoURLToCrawl</u>	PASS
CrawlUrlProviderTest	<u>getAllActiveUrlAndICrawlItTest</u>	PASS
CrawlUrlProviderTest	<u>getAllActiveUrlAndICrawlItTest2</u>	PASS
CrawlUrlProviderTest	<u>crawlUrlProviderActorTest2</u>	PASS
CrawlUrlProviderTest2	<u>getActorSystem</u>	PASS
CrawlUrlProviderTest2	<u>crawlUrlProviderActorTest</u>	PASS
CrawlUrlProviderTest2	<u>getAllActiveUrlToCrawlTest</u>	PASS
CrawlUrlProviderTest2	<u>getAllActiveUrlToCrawlCheckUrlValueTest</u>	PASS

5.2 Further Implementation

Wonts is a system that can be definitely improved to make a most consistent application then now. The system should be more configurable and user friendly. The project work plan in the proposal was underestimated especially for developing the web application. The SCRUM stories and the project tasks have been completed, but in some parts, the project can be improved by adding other features and improving existing features.

For example the web application has a lack of functionalities especially for managing the projects application. For instance, there is not any functionality designed to modify the title or to delete an existing project. At the moment, the modification can be done just using a database client. The same for the URLs added to the projects. It would be better to add a functionality to change the status of the URL on the DB therefore, to re-crawl and maintain updated the user project URLs. Also the system needs a feature that periodically update the user project URLs and the ontologies. For instance, some URLs in a newspaper are more often updated with new resource documents than some URLs of a recipe website.

The crawler in use in the system (***triples.crawler***), should be able to avoid or *escape* from the “spider traps”[21] that are scripts embedded in URL that generates huge number of URLs in the document page that the crawler is parsing. These *traps* generate loops where the crawler cannot exit. At the moment this functionality has not been developed in **Wonts**.

A possible future extension for this project is to make the system able to use existing ontologies as training set for other ontologies where it is not defined any property **hasSearchQuery**. This feature will be drastically automatizing the process of ontology *ABox* creation.

Wonts - Web Ontology Searcher - by Salvatore Rapisarda

A new extension to the project would be to expose all the crawled resources as web services REST for SPARQL query web requests by using Apache Fuseki[51], this is because the database **wonts** uses the same structure Apache Fuseky is using.

6 Conclusion

Creating this project has been a really challenging task for me. First of all, to design the architecture was not an easy task especially when the system is made of many components. The design should clearly identify which are the role and the task that each system component has in a distributed architecture. It has been a fantastic experience to create a message-orientated middleware[64,65] using Akka actor. Secondly, a big amount of Java code needed to be written in a short amount of time to develop each component. Finally, another and no less difficult phase was to create the configuration files and the *core* used in Apache Solr[29].

Overall, the results have been as expected and the outcomes match the project proposal. I really enjoyed creating a crawler that is using ontology models to store the results of its work. I was glad to see that OWL ontologies can be populated of data as expected. It was amazing to realise what was just an idea of automation became a reality.

I learnt how to use Akka actors, Apache Jena, Apache Solr and how to build a search engine which I previously studied[21]. I am really happy to have completed this project that definitely can be useful for linking, sharing and reusing ontology model data.

I created **Wonts** to be a helpful software application system that can be used to save time and avoid manual research and classification of web resource documents. I would like to see this application further developed by adding new features to it. The Semantic web it is just at the beginning and it is already a fantastic technology that allows sharing and reusing the web data resources available to everyone on the web. **Wonts** is helping to do this as well.

7 References

- [1] World Wide Web, Wikipedia Free Encycl. (2015).
https://en.wikipedia.org/w/index.php?title=World_Wide_Web&oldid=679227299 (accessed September 5, 2015).
- [2] P. Hitzler, M. Krötzsch, S. Rudolph, Foundations of Semantic Web Technologies, CRCPress, 2009.
<http://www.semantic-web-book.org>.
- [3] Semantic Web, Wikipedia Free Encycl. (2015).
https://en.wikipedia.org/w/index.php?title=Semantic_Web&oldid=678081340 (accessed September 5, 2015).
- [4] T. Berners-Lee, J. Hendler, O. Lassila, The Semantic Web, Sci. Am. Mag. (2001).
<http://www.sciam.com/article.cfm?id=the-semantic-web&print=true>.
- [5] World Wide Web Consortium, W.W.W. Consortium, Metadata Activity Statement, World Wide Web Consortium, 2002. <http://www.w3.org/Metadata/Activity.html>.
- [6] OWL Web Ontology Language Overview, World Wide Web Consortium (W3C), 2004.
<http://www.w3.org/TR/owl-features/>.
- [7] Submission Request to W3C: OWL 1.1 Web Ontology Language, W3C, 2006.
<http://www.w3.org/Submission/2006/10/>.
- [8] OWL 2 Web Ontology Language Document Overview, W3C, 2009. <http://www.w3.org/TR/owl2-overview/>.
- [9] Ontology components, Wikipedia Free Encycl. (2013).
http://en.wikipedia.org/w/index.php?title=Ontology_components&oldid=572458466 (accessed February 25, 2015).
- [10] SPARQL, Wikipedia Free Encycl. (2015).
<http://en.wikipedia.org/w/index.php?title=SPARQL&oldid=649015692> (accessed March 1, 2015).
- [11] wiki.dbpedia.org : About, Dbpedia.org, 2013. <http://dbpedia.org/About>.
- [12] DBpedia, Wikipedia Free Encycl. (2015).
<https://en.wikipedia.org/w/index.php?title=DBpedia&oldid=679321488> (accessed September 8, 2015).
- [13] URL Specification, n.d. <http://www.w3.org/Addressing/URL/url-spec.txt>.
- [14] Agile software development, Wikipedia Free Encycl. (2015).
http://en.wikipedia.org/w/index.php?title=Agile_software_development&oldid=648966603 (accessed February 26, 2015).
- [15] Agile management, Wikipedia Free Encycl. (2015).
https://en.wikipedia.org/w/index.php?title=Agile_management&oldid=679377210 (accessed September 5, 2015).
- [16] Scrum (software development), Wikipedia Free Encycl. (2015).
[http://en.wikipedia.org/w/index.php?title=Scrum_\(software_development\)&oldid=648940725](http://en.wikipedia.org/w/index.php?title=Scrum_(software_development)&oldid=648940725) (accessed February 26, 2015).
- [17] Test-driven development, Wikipedia Free Encycl. (2015).
http://en.wikipedia.org/w/index.php?title=Test-driven_development&oldid=647562226 (accessed February 26, 2015).
- [18] Unit testing, Wikipedia Free Encycl. (2015).
https://en.wikipedia.org/w/index.php?title=Unit_testing&oldid=676515051 (accessed September 5, 2015).
- [19] JUnit, Wikipedia Free Encycl. (2015).
<https://en.wikipedia.org/w/index.php?title=JUnit&oldid=672951038> (accessed September 5, 2015).

- [20] Web crawler, Wikipedia Free Encycl. (2015).
https://en.wikipedia.org/w/index.php?title=Web_crawler&oldid=677604117 (accessed September 11, 2015).
- [21] Wiley: An Introduction to Search Engines and Web Navigation, 2nd Edition - Mark Levene, (n.d.).
<http://eu.wiley.com/WileyCDA/WileyTitle/productCd-047052684X.html> (accessed September 11, 2015).
- [22] Web indexing, Wikipedia Free Encycl. (2015).
https://en.wikipedia.org/w/index.php?title=Web_indexing&oldid=668047355 (accessed September 11, 2015).
- [23] Web search engine, Wikipedia Free Encycl. (2015).
https://en.wikipedia.org/w/index.php?title=Web_search_engine&oldid=680382212 (accessed September 11, 2015).
- [24] Tbox, Wikipedia Free Encycl. (2014).
<https://en.wikipedia.org/w/index.php?title=Tbox&oldid=638316113> (accessed September 5, 2015).
- [25] Semantic reasoner, Wikipedia Free Encycl. (2015).
http://en.wikipedia.org/w/index.php?title=Semantic_reasoner&oldid=640859377 (accessed February 24, 2015).
- [26] Turtle (syntax), Wikipedia Free Encycl. (2015).
[https://en.wikipedia.org/w/index.php?title=Turtle_\(syntax\)&oldid=670822496](https://en.wikipedia.org/w/index.php?title=Turtle_(syntax)&oldid=670822496) (accessed September 7, 2015).
- [27] RDF 1.1 Turtle: Terse RDF Triple Language, W3C, 2014. http://www.w3.org/TR/turtle/#h2_sec-mediaReg.
- [28] B. DuCharme, Learning SPARQL, O'Reilly Media, Inc., 2013.
- [29] Apache Solr, Wikipedia Free Encycl. (2015).
https://en.wikipedia.org/w/index.php?title=Apache_Solr&oldid=679358425 (accessed September 5, 2015).
- [30] Apache Maven, Wikipedia Free Encycl. (2015).
https://en.wikipedia.org/w/index.php?title=Apache_Maven&oldid=676986940 (accessed September 5, 2015).
- [31] Dependency injection, Wikipedia Free Encycl. (2015).
https://en.wikipedia.org/w/index.php?title=Dependency_injection&oldid=679400152 (accessed September 5, 2015).
- [32] Inversion of control, Wikipedia Free Encycl. (2015).
https://en.wikipedia.org/w/index.php?title=Inversion_of_control&oldid=666044422 (accessed September 5, 2015).
- [33] Spring Framework, Wikipedia Free Encycl. (2015).
https://en.wikipedia.org/w/index.php?title=Spring_Framework&oldid=679156332 (accessed September 5, 2015).
- [34] Akka (toolkit), Wikipedia Free Encycl. (2015).
[https://en.wikipedia.org/w/index.php?title=Akka_\(toolkit\)&oldid=669187224](https://en.wikipedia.org/w/index.php?title=Akka_(toolkit)&oldid=669187224) (accessed September 5, 2015).
- [35] Scala (programming language), Wikipedia Free Encycl. (2015).
[https://en.wikipedia.org/w/index.php?title=Scala_\(programming_language\)&oldid=679536579](https://en.wikipedia.org/w/index.php?title=Scala_(programming_language)&oldid=679536579) (accessed September 5, 2015).
- [36] Actor model, Wikipedia Free Encycl. (2015).
https://en.wikipedia.org/w/index.php?title=Actor_model&oldid=679354856 (accessed September 5, 2015).
- [37] P. Haller, Actors in Scala /, Artima Press, 2011.

- [38] Java Persistence API, Wikipedia Free Encycl. (2015).
https://en.wikipedia.org/w/index.php?title=Java_Persistence_API&oldid=679578013 (accessed September 5, 2015).
- [39] Hibernate (Java), Wikipedia Free Encycl. (2015).
[https://en.wikipedia.org/w/index.php?title=Hibernate_\(Java\)&oldid=679393931](https://en.wikipedia.org/w/index.php?title=Hibernate_(Java)&oldid=679393931) (accessed September 5, 2015).
- [40] Application programming interface, Wikipedia Free Encycl. (2015).
https://en.wikipedia.org/w/index.php?title=Application_programming_interface&oldid=678965801 (accessed September 5, 2015).
- [41] Model–view–controller, Wikipedia Free Encycl. (2015).
<https://en.wikipedia.org/w/index.php?title=Model%E2%80%93view%E2%80%93controller&oldid=678089998> (accessed September 5, 2015).
- [42] Single-page application, Wikipedia Free Encycl. (2015).
https://en.wikipedia.org/w/index.php?title=Single-page_application&oldid=679510614 (accessed September 6, 2015).
- [43] AngularJS, Wikipedia Free Encycl. (2015).
<https://en.wikipedia.org/w/index.php?title=AngularJS&oldid=679143211> (accessed September 5, 2015).
- [44] Cascading Style Sheets, Wikipedia Free Encycl. (2015).
https://en.wikipedia.org/w/index.php?title=Cascading_Style_Sheets&oldid=679183479 (accessed September 6, 2015).
- [45] HTML, Wikipedia Free Encycl. (2015).
<http://en.wikipedia.org/w/index.php?title=HTML&oldid=649185224> (accessed March 1, 2015).
- [46] Bootstrap (front-end framework), Wikipedia Free Encycl. (2015).
[https://en.wikipedia.org/w/index.php?title=Bootstrap_\(front-end_framework\)&oldid=678237593](https://en.wikipedia.org/w/index.php?title=Bootstrap_(front-end_framework)&oldid=678237593) (accessed September 6, 2015).
- [47] JSON-LD 1.0: A JSON-based Serialization for Linked Data, W3C, n.d. <http://www.w3.org/TR/json-ld/>.
- [48] JSON, Wikipedia Free Encycl. (2015).
<https://en.wikipedia.org/w/index.php?title=JSON&oldid=679395103> (accessed September 6, 2015).
- [49] Ajax (programming), Wikipedia Free Encycl. (2015).
[https://en.wikipedia.org/w/index.php?title=Ajax_\(programming\)&oldid=679106809](https://en.wikipedia.org/w/index.php?title=Ajax_(programming)&oldid=679106809) (accessed September 6, 2015).
- [50] Futures and promises, Wikipedia Free Encycl. (2015).
https://en.wikipedia.org/w/index.php?title=Futures_and_promises&oldid=674376789 (accessed September 6, 2015).
- [51] Jena (framework), Wikipedia Free Encycl. (2015).
[https://en.wikipedia.org/w/index.php?title=Jena_\(framework\)&oldid=669271817](https://en.wikipedia.org/w/index.php?title=Jena_(framework)&oldid=669271817) (accessed September 6, 2015).
- [52] Lucene, Wikipedia Free Encycl. (2015).
<https://en.wikipedia.org/w/index.php?title=Lucene&oldid=680634266> (accessed September 13, 2015).
- [53] FrontPage - Tika Wiki, (n.d.). <http://wiki.apache.org/tika/> (accessed September 13, 2015).
- [54] Solrj - Solr Wiki, (n.d.). <https://wiki.apache.org/solr/Solrj> (accessed September 13, 2015).
- [55] protégé, (n.d.). <http://protege.stanford.edu/> (accessed September 13, 2015).
- [56] Protege Wiki, (n.d.). http://protegewiki.stanford.edu/wiki/Main_Page (accessed September 13, 2015).

- [57] Protégé (software), Wikipedia Free Encycl. (2015).
[https://en.wikipedia.org/w/index.php?title=Prot%C3%A9g%C3%A9_\(software\)&oldid=678323725](https://en.wikipedia.org/w/index.php?title=Prot%C3%A9g%C3%A9_(software)&oldid=678323725)
(accessed September 13, 2015).
- [58] Michael Zakharyashev, (n.d.). <http://www.dcs.bbk.ac.uk/~michael/> (accessed September 13, 2015).
- [59] Robots exclusion standard, Wikipedia Free Encycl. (2015).
https://en.wikipedia.org/w/index.php?title=Robots_exclusion_standard&oldid=678262900
(accessed September 5, 2015).
- [60] Wildcard queries, (n.d.). <http://nlp.stanford.edu/IR-book/html/htmledition/wildcard-queries-1.html>
(accessed September 13, 2015).
- [61] Apache Lucene - Query Parser Syntax, (n.d.).
https://lucene.apache.org/core/2_9_4/queryparsersyntax.html (accessed September 13, 2015).
- [62] Logical consequence, Wikipedia Free Encycl. (2015).
https://en.wikipedia.org/w/index.php?title=Logical_consequence&oldid=678405702 (accessed September 13, 2015).
- [63] Hermit Reasoner: Home, (n.d.). <http://hermit-reasoner.com/> (accessed September 13, 2015).
- [64] Middleware (distributed applications), Wikipedia Free Encycl. (2015).
[https://en.wikipedia.org/w/index.php?title=Middleware_\(distributed_applications\)&oldid=670662660](https://en.wikipedia.org/w/index.php?title=Middleware_(distributed_applications)&oldid=670662660) (accessed September 12, 2015).
- [65] Middleware, Wikipedia Free Encycl. (2015).
<https://en.wikipedia.org/w/index.php?title=Middleware&oldid=678239912> (accessed September 12, 2015).

8 Appendix A – Program listing

8.1 data.services (src/main/java)

8.1.1 wonts.data.services.dao

8.1.1.1 *OntologyRepository.java*

```
package wonts.data.services.dao;

import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

import lombok.Getter;

import org.springframework.stereotype.Repository;

import wonts.data.services.model.Ontology;
import wonts.data.services.model.Project;

@Repository
public class OntologyRepository {

    @PersistenceContext @Getter
    EntityManager em;

    public Ontology saveUpdate(Ontology ontology) {
        return em.merge(ontology);
    }

    public Ontology getOntology(int ontologyId) {
        return em.find(Ontology.class, ontologyId);
    }

    public List<Ontology> getOntologiesByProject(Project project) {
        return
em.createNamedQuery(Ontology.GET_ONTOLOGY_BY_PROJECT,
Ontology.class).
```

```
        setParameter("project", project)
.getResultList();
}

public int getOntologiesByProjectCount(Project project) {
    return
em.createNamedQuery(Ontology.GET_ONTOLOGY_BY_PROJECT,
Ontology.class).
        setParameter("project", project)
.getResultList().size();
}

public boolean remove(Ontology ontology) {
    if (ontology != null && ontology.getOntologyId()>0){
        em.remove(ontology);
        return true;
    }
    return false;
}

public List<Ontology> getOntologiesByStatus(
Ontology.OntologyStatus status ) {
    return
em.createNamedQuery(Ontology.GET_ONTOLOGY_BY_STAUS, Ontology.class)
        .setParameter("status",
status.ordinal()).getResultList() ;
}

}
```

8.1.1.2 ProjectRepository.java

```
package wonts.data.services.dao;

import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

import lombok.Getter;

import org.springframework.stereotype.Repository;
```

```
import wonts.data.services.model.Project;
import wonts.data.services.model.User;

@Repository
public class ProjectRepository {

    @Getter
    @PersistenceContext
    EntityManager em;

    public Project saveUpdate( Project project ) {
        return em.merge(project);
    }

    public Project getProject(int projectId) {
        return em.find(Project.class, projectId);
    }

    public List<Project> getProjects(User user) {
        return em.createNamedQuery( Project.GET_BY_USER,
Project.class).setParameter("user", user).getResultList();
    }

    public void deleteProject(Project p) {
        em.remove(p);
    }

    public Project getProjectAndOntology(int projectId) {
        return
em.createNamedQuery(Project.GET_PROJECT_AND_ONTOLOGIES,
Project.class).setParameter("projectId", projectId).getSingleResult();
    }

    public Project getProjectAndUrls(int projectId) {
        return em.createNamedQuery(Project.GET_PROJECT_AND_URLS,
Project.class).setParameter("projectId", projectId).getSingleResult();
    }

}
```

8.1.1.3 UrlRepository.java

```
package wonts.data.services.dao;

import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

import org.springframework.stereotype.Repository;

import wonts.data.services.model.Url;

@Repository
public class UrlRepository {

    @PersistenceContext
    EntityManager em;

    public Url saveUpdate(Url url) {
        return em.merge(url);
    }

    public Url getUrl(int urlId) {
        return em.find(Url.class, urlId);
    }

    public Url getUrl(String urlValue) {
        List<Url> urls = em.createNamedQuery(Url.GET_BY_VALUE,
Url.class).setParameter("urlValue", urlValue).getResultList();
        if (urls != null && urls.size() > 0)
            return urls.get(0);
        else
            return null;
    }

}
```

8.1.1.4 UrlToCrawlRepository.java

```
package wonts.data.services.dao;

import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
```

```
import org.springframework.stereotype.Repository;

import wonts.data.services.model.Url;
import wonts.data.services.model.UrlToCrawl;
import wonts.data.services.model.UrlToCrawl.Status;

@Repository
public class UrlToCrawlRepository {

    @PersistenceContext
    EntityManager em;

    public UrlToCrawl saveUpdate(UrlToCrawl urlToCrawl) {
        return em.merge(urlToCrawl);
    }

    public UrlToCrawl getUrlToCrawl(int urlToCrawlId) {
        return em.find(UrlToCrawl.class, urlToCrawlId);
    }

    public List<UrlToCrawl> getActiveUrl(Status status) {
        return
    em.createNamedQuery(UrlToCrawl.GET_ALL_ACTIVE_URL_BY_STATUS,
    UrlToCrawl.class).setParameter("status", status.ordinal()
    ).getResultList() ;
    }

    public UrlToCrawl getUrl(Url url){
        List<UrlToCrawl> ls =
    em.createNamedQuery(UrlToCrawl.GET_BY_URL,
    UrlToCrawl.class).setParameter("url", url).getResultList();
        if (ls.size() > 0 ){
            return ls.get(0);
        }else {
            return null;
        }
    }

    }

8.1.1.5 UserRepository.java
package wonts.data.services.dao;
```

```
import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

import lombok.Getter;

import org.springframework.stereotype.Repository;

import wonts.data.services.model.User;

/**
 * @author salvatore
 * Repository for the User entity
 *
 */
@Repository
public class UserRepository {

    @Getter
    @PersistenceContext
    private EntityManager entityManager;

    /**
     * get a user given its username
     *
     * @param username - the username of the searched user
     * @return a matching user, or null if no user found.
     */
    public User getUserByUsername(String username) {

        List<User> us =
getEntityManager().createNamedQuery(User.GET_BY_USERNAME,
User.class)
            .setParameter("username", username)
            .getResultList();

        return ! us.isEmpty() ? us.get(0) : null;
    }
}
```

```

}

/**
 * get a user given its email
 *
 * @param email - the email of the searched user
 * @return a matching user, or null otherwise
 */
public User getUserByEmail( String email ){
    List<User> us =
getEntityManager().createNamedQuery(User.GET_BY_EMAIL, User.class)
        .setParameter("email", email)
        .getResultList();

    return ! us.isEmpty() ? us.get(0) : null;
}

/**
 * update or insert a new user
 *
 * @param user - It is the an instance of User
 * @return User inserted or updated
 */
public User saveUpdate(User user) {
    return getEntityManager().merge(user);
}

/**
 * checks if a new username is available in the database
 *
 * @param username - the username to be checked for availability
 * @return true if the username is available
 */
public boolean isUsernameAvailable(String username) {

    List<User> users =
getEntityManager().createNamedQuery(User.GET_BY_USERNAME,
User.class)
        .setParameter("username", username)
        .getResultList();

    return users.isEmpty();
}

```

```
/**  
 * Return all the users  
 *  
 * @return List of user  
 */  
public List<User> getAllUsers() {  
    List<User> ret =  
getEntityManager().createNamedQuery(User.GET_ALL_USERS, User.class)  
        .getResultList();  
    return ret;  
}  
}
```

8.1.2 wonts.data.services.model

8.1.2.1 JenaResourceModel.java

```
package wonts.data.services.model;  
  
import java.nio.file.Files;  
import java.nio.file.Paths;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
  
import lombok.Getter;  
import lombok.Setter;  
  
import org.apache.commons.io.FilenameUtils;  
import org.apache.jena.iri.IRI;  
import org.apache.jena.iri.IRIFactory;  
import org.apache.log4j.LogManager;  
import org.apache.log4j.Logger;  
  
import com.hp.hpl.jena.ontology.OntClass;  
import com.hp.hpl.jena.ontology.OntModel;  
import com.hp.hpl.jena.ontology.OntModelSpec;  
import com.hp.hpl.jena.ontology.OntProperty;  
import com.hp.hpl.jena.rdf.model.ModelFactory;  
import com.hp.hpl.jena.rdf.model.Resource;  
import com.hp.hpl.jena.sparql.vocabulary.FOAF;  
import com.hp.hpl.jena.vocabulary.DCTerms;
```

```
import com.hp.hpl.jena.vocabulary.OWL2;
import com.hp.hpl.jena.vocabulary.RDF;
import com.hp.hpl.jena.vocabulary.XSD;

public class JenaResourceModel {
    private static final Logger log =
LogManager.getLogger(JenaResourceModel.class);

    @Getter
    OntModel rdfModel;
    OntProperty linksOut;
    OntProperty hasLinkIn;
    OntProperty hasNumOfLinksOut;
    OntProperty hasNumOfLinksIn;

    @Getter @Setter
    String NS;

    OntClass webResource;

    @Getter
    Map<String, OntClass> webResources;

    static IRIFactory factory = IRIFactory.jenaImplementation();

    private boolean isValidURI( String uri ) {
        IRI iri = factory.create( uri );
        return ! iri.hasViolation(false);
    }

    private Resource getWebResource( String url){
        // String key = null;
        //if ( url.matches("\\\\p{Graph}+\\\\.\\\\p{Alpha}{2,4}$") ){ //
url.replaceFirst("^.*/[^.]*\\\\.\\\\[^\\\\.]*$", "$1");
        String key = FilenameUtils.getExtension(url); //
url.substring(url.lastIndexOf(".") + 1 );
        if ( key != null && key != "" &&
this.webResources.containsKey( key ) ){
            return webResources.get(key);
        }
    }
}
```

```

        return webResource;
    }

    /**
     * Add to the model the links that have been visited.
     * @param url to add all the links out
     * @param linkOut list of links out
     */
    public void AddUrlLinksVisited ( String url , List<String>
linkOut ){
        if ( isValidURI(url)){
            Resource res = rdfModel.createResource(url,
getWebResource(url) );
            linkOut.stream().forEach( out ->
{
            if ( isValidURI(out)){
                Resource resOut =
rdfModel.createResource(out, getWebResource(out) );
                res.addProperty(linksOut, resOut);
            }
        });
        res.addProperty(hasNumOfLinksOut,
Integer.toString(linkOut.size()) );
    }
}

/**
 * Constructor
 */
public JenaResourceModel(){

    this ( ModelFactory.createOntologyModel(
OntModelSpec.OWL_MEM_MICRO_RULE_INF));
}

private void initModel(OntModel model){
    this.NS = "http://titan.dcs.bbk.ac.uk/~srapis01#";

    // TODO: it can be created by loading an OWL file.

    // Ontology

    webResource = model.createClass(NS + "WebResource");
    webResource.setSuperClass(FOAF.Document);
    webResource.addSuperClass(DCTerms.MediaType);
}

```

```

    // linksOut property
    linksOut = model.createOntProperty(NS + "linksOut");
    linksOut.setDomain(webResource);
    linksOut.setRange(webResource);
    model.add(linksOut, RDF.type, OWL2.IrreflexiveProperty);

    // linksIn property
    hasLinkIn = model.createOntProperty(NS + "hasLinkIn");
    hasLinkIn.setDomain(webResource);
    hasLinkIn.setRange(webResource);
    model.add(hasLinkIn, RDF.type,
    OWL2.IrreflexiveProperty);

    //linksIn property is the inverse of linksOut,
    model.add(linksOut, OWL2.inverseOf, hasLinkIn );
    model.add(hasLinkIn, OWL2.inverseOf, linksOut );

    // has numbers links in
    hasNumOfLinksIn = model.createOntProperty(NS +
"hasNumberOfLinksIn");
    hasNumOfLinksIn.setDomain(webResource);
    hasNumOfLinksIn.setRange(XSD.integer);
    // has numbers links out
    hasNumOfLinksOut = model.createOntProperty(NS +
"hasNumberOfLinksOut");
    hasNumOfLinksOut.setDomain(webResource);
    hasNumOfLinksOut.setRange(XSD.integer);

}

private void initWebResources(OntModel model){
    this.webResources = new HashMap<String, OntClass>();

    String [] resources = null;

    try {
        // it maps an add to the model the web resources as
        html, pdf, ps, doc, docx and so on
        String res = new
String(Files.readAllBytes(Paths.get("resources/web-resources"))); //
add to property file
        if ( res != null ){
            resources = res.split(",");
    
```

```

        }
    } catch (Exception e) {
        log.error(e);
    }
    if (resources != null) {
        for (String r : resources) {
            OntClass c = model.createClass(NS + r.trim());
            c.setSuperClass(webResource);
            webResources.put(r.trim(), c);
        }
    }
    MakeDisjointResources();
}

public void MakeDisjointResources(){
    this.webResources.forEach((k1, r1)-> {
        this.webResources.forEach((k2, r2)-> {
            if(!r2.equals(r1))
                r1.addDisjointWith(r2);
        });
    });
}

/**
 * Constructor that injects the model.
 * @param model this is the ontology model used by Apache Jena
 */
public JenaResourceModel(OntModel model){
    this.rdfModel = model;
    this.initModel(model);
    this.initWebResources(model);
}
}

```

8.1.2.2 Ontology.java

```

package wonts.data.services.model;

import java.util.Date;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;

```

```
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.NamedQueries;
import javax.persistenceNamedQuery;

import lombok.Data;

@NamedQueries({
    @NamedQuery( name=Ontology.GET_ONTOLOGY_BY_PROJECT ,
query="select o from Ontology o where o.project=:project" ),
    @NamedQuery( name=Ontology.GET_ONTOLOGY_BY_STAUS ,
query="select o from Ontology o left join fetch o.project where
o.status=:status")
})

@Data @Entity
public class Ontology {
    public static final String GET_ONTOLOGY_BY_PROJECT =
"getOntologyByProject";
    public static final String GET_ONTOLOGY_BY_STAUS =
"getOntologyByStatus";

    public static enum OntologyStatus {ToProcess, Processing,
Processed};

    @Id @GeneratedValue(strategy=GenerationType.AUTO)
    int ontologyId;
    @Column (unique=true)
    String ontologyName;
    String ontologyDescription;
    String ontologyProcessedName;
    boolean isActive;
    @ManyToOne
    Project project;
    int status;
    Date created;
    Date modified;

    public Ontology(){}
}

public Ontology( String ontologyName,      String
```

```
ontologyDescription ) {
    this(ontologyName, ontologyDescription, null);
}

public Ontology( String ontologyName, String
ontologyDescription, Project project ) {
    this.ontologyName = ontologyName;
    this.ontologyDescription = ontologyDescription;
    this.project = project;
    this.status = Ontology.OntologyStatus.ToProcess.ordinal();

    this.isActive = true;
    this.created = new Date();
    this.modified = new Date();
}
}
```

8.1.2.3 Project.java

```
package wonts.data.services.model;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToMany;
import javax.persistence.ManyToOne;
import javax.persistence.NamedQueries;
import javax.persistenceNamedQuery;
import javax.persistence.OneToMany;

import lombok.Data;

@NamedQueries({
    @NamedQuery(
        name = Project.GET_BY_USER,
        query = "select p from Project p join fetch p.user where
p.user = :user"
),
}
```

```
@NamedQuery(
    name= Project.GET_PROJECT_AND_ONTOLOGIES,
    query = "select p from Project p left join fetch
p.ontologies  where p.projectId = :projectId"
),
@NamedQuery(
    name= Project.GET_PROJECT_AND_URLS,
    query = "select p from Project p left join fetch p.urls
where p.projectId = :projectId"
)
}

@Data @Entity
public class Project {
    public static final String GET_BY_USER = "getProjectsUser";
    public static final String GET_PROJECT_AND_ONTOLOGIES=
"getProjectAndOntologies";
    public static final String GET_PROJECT_AND_URLS=
"getProjectAndUrl";

    @Id @GeneratedValue(strategy=GenerationType.AUTO)
    int projectId;
    String projectName;
    String projectDescription;
    @ManyToOne
    User user;
    @ManyToMany
    List<Url> urls;
    @OneToMany (cascade = CascadeType.ALL)
    List<Ontology> ontologies;
    boolean isComplete;
    boolean isActive;
    Date created;
    Date modified;

    public Project(){}
    public Project (String projectName, String projectDescription,
User user ){
        this(projectName, projectDescription, user, null);
    }
    public Project(String projectName, String projectDescription,
```

```
User user,
        Ontology ontology) {
    this.projectName = projectName;
    this.projectDescription = projectDescription;
    this.user = user;
    this.isActive=true;
    this.created = new Date();
    this.modified = new Date();
    if (ontology != null && ontology.getOntologyId()>0){
        this.ontologies = new ArrayList<Ontology>();
        this.ontologies.add(ontology);
    }
}
}
```

8.1.2.4 Url.java

```
package wonts.data.services.model;

import java.util.Date;
import java.util.List;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToMany;
import javax.persistence.NamedQueries;
import javax.persistenceNamedQuery;

import lombok.Data;

@NamedQueries({
    @NamedQuery(name=Url.GET_BY_VALUE, query="select u from Url u
LEFT JOIN FETCH u.project where u.urlValue = :urlValue" )
})

@Data @Entity
public class Url {
    public static final String GET_BY_VALUE = "getUrlByValue";
```

```
@Id @GeneratedValue(strategy=GenerationType.AUTO)
int urlId;
@Column(unique=true)
String urlValue;
String urlDescription;
@ManyToMany(cascade = CascadeType.ALL)
List<Project> project;
boolean isActive;
Date created;
Date modified;

public Url(){}
public Url(String urlValue, String urlDescription){
    this.urlValue = urlValue;
    this.urlDescription = urlDescription;
    this.isActive=true;
    this.created = new Date();
    this.modified = new Date();
}
}
```

8.1.2.5 UrlToCrawl.java

```
package wonts.data.services.model;

import java.util.Date;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.MapsId;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.OneToOne;

import lombok.Data;

@NamedQueries({
    @NamedQuery(
        name= UrlToCrawl.GET_ALL_ACTIVE_URL_BY_STATUS,
        query= "select u from UrlToCrawl u join fetch u.url
where u.status= :status and u.isActive=true order by u.modified"),
    ... other named queries ...
})
```

```
@NamedQuery(
    name= UrlToCrawl.GET_BY_URL,
    query= "select u from UrlToCrawl u join fetch u.url
where u.url = :url")
})

@Data @Entity
public class UrlToCrawl {

    public static final String GET_ALL_ACTIVE_URL_BY_STATUS =
"getAllActiveUrlByStatus";
    public static final String GET_BY_URL = "getUrlByName";
    public enum Status{isToCrawl, isCrawling, isCrawled,
isPersted, isIndexing, isIndexed };

@Id @GeneratedValue(strategy=GenerationType.AUTO)
int urlToCrawlId;

@OneToOne(cascade = {CascadeType.ALL})
@MapsId
Url url;
int status;
String owlFileName;
boolean isActive;
long timeElapsedOnCrawling;
int totalResources;
int resourcesToIndex;
long timeStartOnIndexing;
long timeElapsedOnIndexing;
Date created;
Date modified;

public UrlToCrawl(){}
public UrlToCrawl(Url url){
    this.url = url;
    this.isActive = true;
    this.modified = new Date();
    this.created = new Date();
}
}
```

8.1.2.6 User.java

```
package wonts.data.services.model;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;

import lombok.Data;

/**
 * @author salvatore
 *
 * This is the User class entity
 *
 */
@NamedQueries({
    @NamedQuery(
        name = User.GET_BY_USERNAME,
        query = "select u from User u where u.username =
:username"
    ),
    @NamedQuery(
        name = User.GET_ALL_USERS,
        query = "select u from User u"
    ),
    @NamedQuery(
        name = User.GET_BY_EMAIL,
        query = "select u from User u where u.email = :email"
    )
})

@Entity
@Data
public class User {
    public static final String GET_ALL_USERS = "user.getAllUser";
    public static final String GET_BY_USERNAME =
"user.getByUserName";
    public static final String GET_BY_EMAIL = "user.getByEmail";

    @Id @GeneratedValue(strategy=GenerationType.AUTO)
    private int id ;
```

```
private String name;
private String surname;
private String email;
private String username;
private String password;
private String image;

public User() {}

public User(String username, String passwordDigest, String email,
String name, String surname, String image) {
    this.username = username;
    this.password = passwordDigest;
    this.email = email;
    this.image = image;
    this.name = name;
    this.surname = surname;
}

public User(String username, String passwordDigest, String email)
{
    this( username, passwordDigest, email, null, null, null );
}

}
```

8.1.3 wonts.data.services.provider

8.1.3.1 *OntologyServiceProvider.java*

```
package wonts.data.services.provider;

import java.util.List;
import java.util.stream.Collectors;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import wonts.data.services.dao.OntologyRepository;
import wonts.data.services.model.Ontology;
import wonts.data.services.model.Ontology.OntologyStatus;
import wonts.data.services.model.Project;
```

```
@Transactional
@Service
public class OntologyServiceProvider {

    @Autowired
    OntologyRepository ontologyRepository;

    public Ontology getOntology(int ontologyId){
        return ontologyRepository.getOntology(ontologyId);
    }

    public Ontology saveUpdate(Ontology ontology){
        return ontologyRepository.saveUpdate(ontology);
    }

    public List<Ontology> getOntologiesByProject(Project project) {
        return ontologyRepository.getOntologiesByProject(project);
    }

    public int getOntologiesCountByProject(Project project) {
        return
    ontologyRepository.getOntologiesByProjectCount(project);
    }

    public boolean remove(Ontology ontology) {
        return ontologyRepository.remove(ontology);
    }

    public List<Ontology> getOntologiesByStatus( OntologyStatus
status ) {
        return ontologyRepository.getOntologiesByStatus(status);
    }

    public List<Ontology> getActiveOntologiesToProcess() {
        List <Ontology> ontologies
=ontologyRepository.getOntologiesByStatus(OntologyStatus.ToProcess);
        if ( ontologies != null ){
            return ontologies.stream().filter(p->
p.isActive()).collect(Collectors.toList());
        }
        return null;
    }
}
```

}

8.1.3.2 ProjectServiceProvider.java

```
package wonts.data.services.provider;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import wonts.data.services.dao.ProjectRepository;
import wonts.data.services.model.Project;
import wonts.data.services.model.User;

@Transactional
@Service
public class ProjectServiceProvider {

    @Autowired
    ProjectRepository projectRepository;

    public Project getProject(int id){
        return projectRepository.getProject(id);
    }

    public Project saveUpdate(Project project) {
        return projectRepository.saveUpdate(project);
    }

    public List<Project> getProjects(User user){
        return projectRepository.getProjects(user);
    }

    public Project getProjectAndUrls(int projectId ){
        return projectRepository.getProjectAndUrls(projectId);
    }
}
```

}

8.1.3.3 UrlServiceProvider.java

```
package wonts.data.services.provider;

import java.util.Date;
import java.util.List;

import javax.transaction.Transactional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import wonts.data.services.dao.UrlRepositoritory;
import wonts.data.services.dao.UrlToCrawlRepository;
import wonts.data.services.model.Url;
import wonts.data.services.model.UrlToCrawl;
import wonts.data.services.model.UrlToCrawl.Status;

@Transactional
@Service
public class UrlServiceProvider {
    @Autowired
    UrlRepositoritory urlRepository;

    @Autowired
    UrlToCrawlRepository urlToCrawlRepository;

    public UrlToCrawl saveUpdate(Url url) {
        Url alreadyUrl = urlRepository.getUrl(url.getUrlValue());

        if (url.getUrlId() <= 0 && alreadyUrl == null) {
            url = urlRepository.saveUpdate(url);
        } else {
            url = alreadyUrl;
        }

        UrlToCrawl urlToCrawl = urlToCrawlRepository.getUrl(url);
        if (urlToCrawl != null)
            return urlToCrawl;
        else
            return urlToCrawlRepository.saveUpdate(new
UrlToCrawl(url));
    }
}
```

```
public UrlToCrawl saveUpdate(UrlToCrawl urlToCrawl) {
    urlToCrawl.setModified(new Date());
    return urlToCrawlRepository.saveUpdate(urlToCrawl);
}

public List<UrlToCrawl> getAllActiveUrlToCrawl(){
    return
urlToCrawlRepository.getActiveUrl(UrlToCrawl.Status.isToCrawl);
}

public List<UrlToCrawl> getUrlThatAreCrawling(){
    return
urlToCrawlRepository.getActiveUrl(UrlToCrawl.Status.isCrawling);
}

public List<UrlToCrawl> getAllActiveUrlToIndex(){
    return
urlToCrawlRepository.getActiveUrl(UrlToCrawl.Status.isCrawled);
}

public UrlToCrawl getUrlToCrawl(int urlToCrawlId) {
    // TODO Auto-generated method stub
    return urlToCrawlRepository.getUrlToCrawl(urlToCrawlId);
}

public UrlToCrawl updateStatusUrlToCrawl(int urlToCrawlId,
Status status, long timeElapsedOnCrawling, String owlFileName) {
    UrlToCrawl url = this.getUrlToCrawl(urlToCrawlId);
    url.setStatus(status.ordinal());
    url.setModified(new Date());
    url.setOwlFileName(owlFileName);
    if (timeElapsedOnCrawling >= 0 ){
        url.setTimeElapsedOnCrawling(timeElapsedOnCrawling);
    }
    return this.saveUpdate(url);
}

public UrlToCrawl updateStatusUrlToCrawl(int urlToCrawlId,
Status status) {
```

```
        UrlToCrawl url = this.getUrlToCrawl(urlToCrawlId);
        url.setStatus(status.ordinal());
        url.setModified(new Date());

        return this.saveUpdate(url);
    }

    public Url getUrlByValue(String urlValue) {
        return urlRepository.getUrl(urlValue);
    }

    public Url getUrlById(int urlId) {
        return urlRepository.getUrl(urlId);
    }
}
```

8.1.3.4 UserServiceProvider.java

```
package wonts.data.services.provider;

import static
wonts.data.services.provider.ValidationUtils.assertMatches;
import static
wonts.data.services.provider.ValidationUtils.assertMinimumLength;
import static
wonts.data.services.provider.ValidationUtils.assertNotBlank;

import java.util.List;
import java.util.regex.Pattern;

import javax.inject.Named;

import org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import wonts.data.services.dao.UserRepository;
import wonts.data.services.model.User;

/**
 * @author salvatore

```

```
* this is the business service provider for User entity
*/
@Service @Named("UserService")
public class UserService {
    private static final Pattern REGEX_PWD =
Pattern.compile("(?=.*\\d)(?=.*[a-z])(?=.*[A-Z]).{6,}");  
  

    private static final Pattern REGEX_EMAIL =
Pattern.compile("^[_A-Za-z0-9-\\\\+]+(\\.[_A-Za-z0-9-\\\\+])*@"
+ "[A-Za-z0-9-\\\\+](\\.[_A-Za-z0-9-\\\\+])*([\\.][A-Za-z]{2,})$");  
  

    @Autowired
    private UserRepository userRepository;  
  

    /**
     *
     * creates a new user
     *
     * @param username - the username of the new user
     * @param email - the user email
     * @param password - the user plain text password
     */
    @Transactional
    public User createUser(String username, String email, String
password) {
        return this.createUser ( username, email, password, null,
null, null );
    }
  

    /**
     *
     * creates a new user
     *
     * @param username - the user name of the new user
     * @param email - the user email
     * @param password - the user plain text password
     * @param name - the user first name
     * @param surname - the user last name
     */
    @Transactional
    public User createUser(String username, String email, String
password, String name, String surname, String image) {
```

```

        assertNotBlank(username, "The username cannot be an empty
string.");
        assertMinimumLength(username, 6, "The username must have at
least 6 characters.");
        assertNotBlank(email, "The user email cannot be empty.");
        assertMatches(email, REGEX_EMAIL, "the user email is not
valid.");
        assertNotBlank(password, "The password cannot be empty an
empty string.");
        assertMatches(password, REGEX_PWD, "The password does not
match the creteria. It must have at least 6 characters, with 1 numeric
and 1 uppercase character.");

        if (!userRepository.isUsernameAvailable(username)) {
            throw new IllegalArgumentException("The username is
not available.");
        }

        User user = new User(username, new
BCryptPasswordEncoder().encode(password), email, name, surname,
image);

        return userRepository.saveUpdate(user);
    }

    /**
     * Get the user by user name
     *
     * @param username user name to search
     * @return an instance of User, otherwise null
     */
    @Transactional(readOnly = true)
    public User getUserByUsername(String username) {
        return userRepository.getUserByUsername(username);
    }

    /**
     * Get the user by user email
     *
     * @param email - user email to search
     * @return an instance of User, otherwise null
     */
    @Transactional(readOnly=true)

```

```
public User getUserByEmail(String email){
    return userRepository.getUserByEmail(email);
}

/**
 * Insert or Update the User
 *
 * @param user
 */
@Transactional
public void saveUpdate(User user) {
    userRepository.saveUpdate(user);
}

/**
 * Check if the user name can be available
 *
 * @param username
 * @return This returns true is the user name is not in use,
otherwise it returns false
 */
@Transactional(readOnly=true)
public boolean isUsernameAvailable(String username) {
    return this.getUserByUsername(username) == null;
}

/**
 * @return the entire user the db contains
 */
@Transactional(readOnly=true)
public List<User> getAllUsers() {
    return userRepository.getAllUsers();
}

}
```

8.1.3.5 ValidationUtils.java

```
package wonts.data.services.provider;

import org.apache.commons.lang3.NotImplementedException;
import org.apache.commons.lang3.StringUtils;
```

```
import java.util.regex.Pattern;

/**
 * This a class that contains data validation utility methods
 */
public final class ValidationUtils {

    private ValidationUtils() {
        throw new NotImplementedException("this class cannot be
instantiated");
    }

    /**
     * Assert that the username is not a blank field
     *
     * @param username
     * @param message
     */
    public static void assertNotBlank(String username, String
message) {
        if (StringUtils.isBlank(username)) {
            throw new IllegalArgumentException(message);
        }
    }

    /**
     * Assert the minimum length for a field
     * @param username
     * @param length
     * @param message
     */
    public static void assertMinimumLength(String username, int
length, String message) {
        if (username.length() < length) {
            throw new IllegalArgumentException(message);
        }
    }

    /**
     * Assert that we are using a proper email
     *
     * @param email
     * @param regex
     * @param message
     */
}
```

```
/*
public static void assertMatches(String email, Pattern regex,
String message) {
    if (!regex.matcher(email).matches()) {
        throw new IllegalArgumentException(message);
    }
}
```

8.1.4 wonts.data.services.triples

8.1.4.1 TriplesDataProvider

```
package wonts.data.services.triples;

import java.io.PrintStream;

import javax.inject.Named;

import lombok.Data;

import org.apache.log4j.LogManager;
import org.apache.log4j.Logger;

import wonts.data.services.triples.property.TriplesPropertiesManager;

import com.hp.hpl.jena.query.Dataset;
import com.hp.hpl.jena.query.Query;
import com.hp.hpl.jena.query.QueryExecution;
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.ResultSetFactory;
import com.hp.hpl.jena.query.ResultSetFormatter;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.reasoner.Reasoner;
import com.hp.hpl.jena.reasoner.ReasonerRegistry;
import com.hp.hpl.jena.sdb.SDBFactory;
import com.hp.hpl.jena.sdb.Store;
import com.hp.hpl.jena.sdb.StoreDesc;
import com.hp.hpl.jena.sdb.sql.JDBC;
import com.hp.hpl.jena.sdb.sql.SDBConnection;
import com.hp.hpl.jena.sdb.store.DatabaseType;
import com.hp.hpl.jena.sdb.store.DatasetStore;
```

```
import com.hp.hpl.jena.sdb.store.LayoutType;
import com.hp.hpl.jena.update.UpdateAction;
import com.hp.hpl.jena.util.FileManager;

@Data @Named("triplesDataProvider")
public class TriplesDataProvider {
    private static final Logger log =
LogManager.getLogger(TriplesDataProvider.class);

    private SDBConnection getSDBConnection(){

        JDBC.loadDriverMySQL();

        return new SDBConnection(
            TriplesPropertiesManager.getProperty(TriplesPropertiesManager.DB_URL),
            TriplesPropertiesManager.getProperty(TriplesPropertiesManager.DB_USER),
            TriplesPropertiesManager.getProperty(TriplesPropertiesManager.DB_PWD)
        );
    }

    private Store getStore(SDBConnection conn ){
        StoreDesc storeDesc = new
        StoreDesc(LayoutType.LayoutTripleNodesHash,
                  DatabaseType.MySQL) ;
        return SDBFactory.connectStore(conn, storeDesc) ;

    }

    private Dataset getDataset(SDBConnection conn){
        Store store = getStore(conn);
        return DatasetStore.create(store) ;
    }

    public boolean PutRdfInDataBase(String filename){
        boolean ret= false;
        try{
            SDBConnection conn = getSDBConnection();
```

```

try {
    Dataset ds = getDataset(conn);
    Model model = ds.getDefaultModel();
    model.begin();
    FileManager.get().readModel(model, filename);
    model.commit();
    model.close();
    ds.close();
    ret = true;
} catch ( Exception ex ){
    log.error(ex);
    ret=false;
}
finally {
    conn.close();
}
catch(Exception ex){
    log.error(ex);
    ret = false;
}
return ret;
}

```

```

public ResultSet execSelectQuery(String queryString) {
    ResultSet res = null;
    Query query = QueryFactory.create(queryString) ;
    SDBConnection conn = getSDBConnection();
    Dataset ds = getDataset(conn);
    Model model = ds.getDefaultModel();
    Reasoner r = ReasonerRegistry.getRDFSReasoner();
    r.bindSchema(model);
    try (QueryExecution qexec =
QueryExecutionFactory.create(query, model)){
        res =qexec.execSelect();
        res = ResultSetFactory.copyResults(res) ;
        qexec.close();
    } catch (Exception e) {
        res = null;
    }
    log.error(e);
} finally{
    ds.close();
    model.close();
}

```

```

        conn.close();
    }
    return res;
}

/**
 * This method is deleting all the URL pages that match all or
part of the urlString .
 * @param urlString
 * @return
 */
public boolean deleteLinkOutPages(String urlString ) {
    if ( urlString == null ){
        return false;
    }

    String action = "DELETE { ?s ?p ?o } "
        + " WHERE { ?s ?p ?o ."
        + " FILTER regex(str(?s), \\" + urlString + "\\",
"\\"i\\") "
        + " }";
    return this.updateQuery(action);
}

/**
 * This method is used to update the model.
 * @param action, is the query action. For Example: String
action = "DELETE WHERE { ?s ?p ?o . } it is going to clean the model.
 * @return a boolean value, that is true when the action has
been done correctly, otherwise it returns false.
 */
private boolean updateQuery(String action) {
    boolean res = false;
    SDBConnection conn = getSDBConnection();
    Dataset ds = getDataset(conn);
    Model model = ds.getDefaultModel();
    try {
        model.begin();
        UpdateAction.parseExecute(action, model);
        model.commit();
        res = true;
    }catch (Exception e) {
        res = false;
        log.error(e);
    }
}

```

```

}finally{
    ds.close();
    model.close();
    conn.close();
}
return res;
}

/**
 *
 * @param queryString
 * @param notation Predefined values are "RDF/XML",
 * "RDF/XML-ABBREV", "N-TRIPLE", "TURTLE", (and "TTL") and "N3".
The default value,
* represented by <code>null</code>, is "RDF/XML".</p>
* @return
*/
public String execSelectQuery(String queryString, String
notation) {
    String retval="";
    //JenaResourceModel jrm = new JenaResourceModel();
    //OntModel m = jrm.getRdfModel();
    Query q = QueryFactory.create(queryString);
    SDBConnection conn = getSDBConnection();
    Dataset ds = getDataset(conn);
    Model model = ds.getDefaultModel();
    Reasoner r = ReasonerRegistry.getRDFSReasoner();
    r.bindSchema(model);
    try (QueryExecution qe = QueryExecutionFactory.create(q,
model)){

        PrintStream baos = System.out;
        int queryType = q.getQueryType();
        switch (queryType) {
        case Query.QueryTypeAsk:
            boolean b = qe.execAsk();
            ResultSetFormatter.outputAsRDF(baos, notation, b);
            retval = baos.toString();
            break;
        case Query.QueryTypeConstruct:
            model = qe.execConstruct();
            model.write(baos, notation);
            retval = baos.toString();
            break;
        }
    }
}

```

```
        case Query.QueryTypeDescribe:
            model = qe.execDescribe();
            model.write(baos, notation);
            retval = baos.toString();
            break;
        case Query.QueryTypeSelect:
            ResultSet results = qe.execSelect();
            ResultSetFormatter.out(baos, results);
            retval = baos.toString();
            break;
    }

}catch (Exception e) {
    log.error(e);
}finally{
    ds.close();
    model.close();
    conn.close();
}
return retval;
}

}
```

8.1.5 wonts.data.services.triples.property

8.1.5.1 *TriplesPropertiesManager.java*

```
package wonts.data.services.triples.property;

import java.io.FileInputStream;
import java.util.Properties;

import org.apache.log4j.LogManager;
import org.apache.log4j.Logger;

public class TriplesPropertiesManager {
    private static final Logger log =
LogManager.getLogger(TriplesPropertiesManager.class);

    public static final String DB = "persistence.db";
```

```
public static final String DB_URL = "persistence.db.url";
public static final String     DB_USER = "persistence.db.urs";
public static final String     DB_PWD="persistence.db.pwd";
public static final String   CLASS_NAME=
"persistence.className";

public static final String PROPERTY_FILENAME=
"resources/persistence.properties";

static Properties properties;

static {
    properties = GetProviderProperty(PROPERTY_FILENAME);
}

public static Properties GetProviderProperty(String fileName){
    Properties ret= new Properties();
    try {
        ret.load(new FileInputStream(fileName));
    } catch (Exception e) {
        log.error(e);
    }
    return ret;
}

public static String getProperty(String key){
    return properties.getProperty(key);
}

}
```

8.1.6 beans.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">
```

```
<bean id="emf"
```

```

class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
    <property name="dataSource" ref="dataSource" />
    <property name="packagesToScan"
value="wonts.data.services.model" />
    <property name="jpaVendorAdapter">
        <bean
class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter"
/>
        </property>
    <property name="jpaProperties">
        <props>
            <prop key="hibernate.hbm2ddl.auto">update</prop>
            <prop
key="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</prop>
            </props>
        </property>
    </bean>

    <bean id="dataSource"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName" value="com.mysql.jdbc.Driver"
/>
        <property name="url"
value="jdbc:mysql://localhost:3306/wontsapp" />
        <property name="username" value="wontsuser" />
        <property name="password" value="user$wonts" />
    </bean>

    <bean id="transactionManager"
class="org.springframework.orm.jpa.JpaTransactionManager">
        <property name="entityManagerFactory" ref="emf" />
    </bean>

<context:component-scan base-package="wonts.data.services" />

    <!-- Root Context: defines shared resources visible to all other
web components -->
</beans>
```

8.1.7 log4j.properties

```

log4j.rootLogger=info, stdout
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
```

```
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern="%d %-5p [%t] %-17c{2}
(%13F:%L) %3x - %m%n
```

8.2 data.services (src/test/java)

8.2.1 wonts.data.services

8.2.1.1 AllRepositoriesTests.java

```
package wonts.data.services;

import org.junit.runner.RunWith;
import org.junit.runners.Suite;
import org.junit.runners.SuiteClasses;

import wonts.data.services.repository.OntologyRepositoryTest;
import wonts.data.services.repository.ProjectRepositoryTest;
import wonts.data.services.repository.UrlRepositoryTest;
import wonts.data.services.repository.UrlToCrawlRepositoryTest;
import wonts.data.services.repository.UserRepositoryTest;

@RunWith(Suite.class)
@SuiteClasses({ OntologyRepositoryTest.class,
ProjectRepositoryTest.class,
        UrlRepositoryTest.class, UrlToCrawlRepositoryTest.class,
        UserRepositoryTest.class, OntologyRepositoryTest.class })
public class AllRepositoriesTests {

}
```

8.2.1.2 ContextConfigurationTest.java

```
package wonts.data.services;

import javax.transaction.Transactional;

import org.junit.Assert;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.test.context.ActiveProfiles;
import org.springframework.test.context.ContextConfiguration;
import
org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
/***
 * This class is used to load the Context configuration files

```

```
* it will be extended by the tests classes.  
*  
* @author salvatore  
*/  
@ActiveProfiles("test")  
@ContextConfiguration( { "beans_test.xml"})  
@Transactional  
@RunWith(SpringJUnit4ClassRunner.class)  
public class ContextConfigurationTest {  
    public ContextConfigurationTest(){  
  
        @Test  
        public void PlaceholderTest(){  
            int a=1;  
            Assert.assertTrue(1==a);  
        }  
    }  
}
```

8.2.1.3 TestJenaDataProvider

```
package wonts.data.services;  
  
import static org.junit.Assert.assertEquals;  
import static org.junit.Assert.assertNotNull;  
import static org.junit.Assert.assertTrue;  
  
import java.io.IOException;  
import java.nio.file.Files;  
import java.nio.file.Path;  
import java.nio.file.Paths;  
import java.util.ArrayList;  
import java.util.HashSet;  
import java.util.List;  
import java.util.Set;  
import java.util.concurrent.atomic.AtomicInteger;  
import java.util.stream.Stream;  
  
import org.junit.Assert;  
import org.junit.BeforeClass;  
import org.junit.Test;  
  
import wonts.data.services.model.JenaResourceModel;
```

```
import wonts.data.services.triples.TriplesDataProvider;
import wonts.data.services.triples.property.TriplesPropertiesManager;

import com.hp.hpl.jena.query.QuerySolution;
import com.hp.hpl.jena.query.ResultSet;
import com.hp.hpl.jena.rdf.model.Property;
import com.hp.hpl.jena.rdf.model.RDFNode;
import com.hp.hpl.jena.rdf.model.Resource;
import com.hp.hpl.jena.rdf.model.Statement;
import com.hp.hpl.jena.rdf.model StmtIterator;

public class TestJenaDataProvider {

    @BeforeClass
    public static void setUpBeforeClass() throws Exception {
    }

    private List<String> getLinkOutFromFile(Path path) throws
IOException{
        List<String> linkOut = new ArrayList<>();
        try(Stream<String> lines = Files.lines(path)){;
            lines.forEach(linkOut::add);
            //Close the stream and it's underlying file as well
            lines.close();
        }
        return linkOut;
    }

    public int countDuplicates(List<String>
listContainingDuplicates) {
        AtomicInteger ret = new AtomicInteger();
        final Set<String> set1 = new HashSet<String>();

        listContainingDuplicates.forEach( p-> {
            if (!set1.add(p)) {
                ret.addAndGet(1);
            }
        });
        return ret.get();
    }
}
```

```

// Development mode test skipped on install
// @Test
public void TestRdfDataProvider() {

    String url = "http://www.dcs.bbk.ac.uk/~mark";
    List<String> linkOut = null;
    try {
        linkOut =
getLinkOutFromFile(Paths.get("tests/", "TestDataCrowler.txt"));
    } catch (IOException e) {
        Assert.fail();
    }

    // List<String> linkOut = new ArrayList<>();
    //
    linkOut.add("http://www.dcs.bbk.ac.uk/research/dbtech");
    linkOut.add("http://www.dcs.bbk.ac.uk");
    linkOut.add("http://www.bbk.ac.uk");

    assertNotNull(linkOut);
    assertTrue(linkOut.size()>0);

    JenaResourceModel j = new JenaResourceModel();
    j.AddUrlLinksVisited(url, linkOut);
    StmtIterator iter =
j.getRdfModel().listStatements();
    int actual = 0;

    int numDuplicates = countDuplicates(linkOut);
    int expected = linkOut.size() - numDuplicates;

    System.out.println(" num of duplicates in the
list: " + numDuplicates);

    List<String> actualLinks = new
ArrayList<String>();
    while (iter.hasNext()) {
        Statement stmt = iter.nextStatement();
        // get next statement
        Resource subject = stmt.getSubject();
        // get the subject
        Property predicate = stmt.getPredicate();

```

```

// get the predicate
RDFNode object = stmt.getObject();
// get the object

subject.toString());
System.out.print(" subject: " +
predicate.toString() + " ");
System.out.print(" predicate: " +
if (object instanceof Resource) {
    System.out.print(" object r:" +
object.toString());
} else {
    // object is a literal
    System.out.print(" object l:" +
object.toString() );
}
actualLinks.add(object.toString());
actual++;
System.out.println();
}

j.getRdfModel().write(System.out);

linkOut.forEach(p-> {
    if ( ! actualLinks.contains(p) ) {
        System.out.print(" THIS IS NOT
CONTAINED: " + p.toString() );
    }
} );
}

assertEquals(expected, actual);

}

@Test
public void TestPersistencePropertyFile_DB(){
    String actual =
TriplesPropertiesManager.getProperty(TriplesPropertiesManager.DB);
    assertTrue(actual != null && actual.length()>0);

}

```

```
@Test
public void TestPersistencePropertyFile_URL(){
    String actual =
TriplesPropertiesManager.getProperty(TriplesPropertiesManager.DB_URL);
    assertTrue(actual != null && actual.length()>0);

}

@Test
public void TestPersistencePropertyFile_DB_USER(){
    String actual =
TriplesPropertiesManager.getProperty(TriplesPropertiesManager.DB_USER);
    assertTrue(actual != null && actual.length()>0);

}

@Test
public void TestPersistencePropertyFile_CLASS_NAME(){
    String actual =
TriplesPropertiesManager.getProperty(TriplesPropertiesManager.CLASS_NAME);
    assertTrue(actual != null && actual.length()>0);

}

@Test
public void TestPersistencePropertyFile_PROPERTY_FILENAME(){
    String actual = TriplesPropertiesManager.
GetProviderProperty(TriplesPropertiesManager.PROPERTY_FILENAME)
    .
        getProperty(TriplesPropertiesManager.DB_URL);
    assertTrue(actual != null && actual.length()>0);
}

// @Test // uncomment on dev environment
public void TestPersistenceWriteOnDB(){

    TriplesDataProvider p = new TriplesDataProvider();
    String filename="d4d29503-159e-44e9-b135-35128a7836d4.owl";
    boolean actual = p.PutRdfInDataBase(filename);
    assertTrue(actual);

}
```

```
}
```

```
@Test
```

```
public void TestResourcesModelNotEmpty(){
    JenaResourceModel m = new JenaResourceModel();
    assertTrue( m.getWebResources() != null );
    assertTrue(m.getWebResources().size()>0);
}
```

```
@Test
```

```
public void TestPersistenceQueryResultFormattedAsTripleLinkin(){
    TriplesDataProvider p = new TriplesDataProvider();
```

```
String query= "PREFIX rdf: <http://www.w3.org/1999/02/22-
rdf-syntax-ns#>"  
        + " PREFIX owl: <http://www.w3.org/2002/07/owl#>"  
        + " PREFIX xsd:  
<http://www.w3.org/2001/XMLSchema#>"  
        + " PREFIX rdfs: <http://www.w3.org/2000/01/rdf-
schema#>"  
        + " PREFIX j.0:  
<http://titan.dcs.bbk.ac.uk/~srapis01#>"  
        + " SELECT ?s ?o"  
        + " WHERE { ?o j.0:linksOut ?s ."  
        + " }";
```

```
String results = p.execSelectQuery(query,"N-TRIPLE");
System.out.println(results);
assertNotNull( results );
```

```
}
```

```
@Test
```

```
public void
```

```
TestPersistenceQueryResultFormattedAsTriple_Filtered_LIMIT(){
    TriplesDataProvider p = new TriplesDataProvider();
```

```
String query= "PREFIX rdf: <http://www.w3.org/1999/02/22-
rdf-syntax-ns#>"  
        + " PREFIX owl: <http://www.w3.org/2002/07/owl#>"  
        + " PREFIX xsd:  
<http://www.w3.org/2001/XMLSchema#>"
```

```

+ " PREFIX rdfs: <http://www.w3.org/2000/01/rdf-
schema#>
+ " PREFIX j.0:
<http://titan.dcs.bbk.ac.uk/~srapis01#>
+ " SELECT ?o "
+ " WHERE { ?s ?p ?o ."
// + "           ?o rdf:type j.0:html "
+ " FILTER regex(str(?s),
\"http://www.dcs.bbk.ac.uk/~michael\", \"i\") "
+ "}" LIMIT 10 ";

String results = p.execSelectQuery(query,"N-TRIPLE");
System.out.println(results);
assertNotNull( results );
}

@Test
public void TestPersistenceQueryResultFormattedAsTriple_model(){
TriplesDataProvider p = new TriplesDataProvider();

String query= "PREFIX rdf: <http://www.w3.org/1999/02/22-
rdf-syntax-ns#>
+ " PREFIX owl: <http://www.w3.org/2002/07/owl#>
+ " PREFIX xsd:
<http://www.w3.org/2001/XMLSchema#>
+ " PREFIX rdfs: <http://www.w3.org/2000/01/rdf-
schema#>
+ " PREFIX wonts:
<http://titan.dcs.bbk.ac.uk/~srapis01#>
+ " SELECT ?o "
+ " WHERE { ?s wonts:linksOut ?o ."
+ " FILTER regex(str(?s),
\"http://www.dcs.bbk.ac.uk/~michael\", \"i\") }"
//+ " LIMIT 10 ";

String results = p.execSelectQuery(query,"N-TRIPLE");
System.out.println(results);
assertNotNull( results );
}

@Test
public void
TestPersistenceQueryResultFormattedAsResultSet_model(){
TriplesDataProvider p = new TriplesDataProvider();

```

```

String url = "http://www.dcs.bbk.ac.uk/~michael";

        String queryString= "PREFIX rdf:
<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
        + " PREFIX owl: <http://www.w3.org/2002/07/owl#>"
        + " PREFIX xsd:
<http://www.w3.org/2001/XMLSchema#>
        + " PREFIX rdfs: <http://www.w3.org/2000/01/rdf-
schema#>"
        + " PREFIX wonts:
<http://titan.dcs.bbk.ac.uk/~srapis01#>"
        + " SELECT ?o ?s"
        + " WHERE { ?s wonts:linksOut ?o ."
        + " FILTER regex(str(?s), \"%s\", \"i\") }" ;

queryString = String.format(queryString, url );

ResultSet results = p.execSelectQuery(queryString);

while (results.hasNext()) {
    QuerySolution type = results.next();
    System.out.println( type.get("o").toString());

    System.out.println( type.get("s").toString());
    assertNotNull( results );
}
}

@Test
public void
TestPersistenceQueryResultFormattedAsTriple_Filtered_Property(){
    TriplesDataProvider p = new TriplesDataProvider();

    String query= "PREFIX rdf: <http://www.w3.org/1999/02/22-
rdf-syntax-ns#>"
        + " PREFIX owl: <http://www.w3.org/2002/07/owl#>"
        + " PREFIX xsd:
<http://www.w3.org/2001/XMLSchema#>
        + " PREFIX rdfs: <http://www.w3.org/2000/01/rdf-
schema#>"
        + " PREFIX j.0:
<http://titan.dcs.bbk.ac.uk/~srapis01#>"
        + " SELECT ?s ?p ?o "
        + " WHERE { ?s ?p ?o ."

```

```
//+ "          ?p rdf:property j.0:linksOut ."
+ " FILTER regex(str(?s),
\"http://www.dcs.bbk.ac.uk/~michael\", \"i\") "
+ "}" LIMIT 10 ";

String results = p.execSelectQuery(query,"N-TRIPLE");
System.out.println(results);
assertNotNull( results );
}

@Test
public void
TestPersistenceQueryResultFormattedAsTriple_Filtered(){
    TriplesDataProvider p = new TriplesDataProvider();

    String query= "PREFIX rdf: <http://www.w3.org/1999/02/22-
rdf-syntax-ns#>"
        + " PREFIX owl: <http://www.w3.org/2002/07/owl#>"
        + " PREFIX xsd:
<http://www.w3.org/2001/XMLSchema#>"
        + " PREFIX rdfs: <http://www.w3.org/2000/01/rdf-
schema#>"
        + " PREFIX j.0:
<http://titan.dcs.bbk.ac.uk/~srapis01#>"
        + " PREFIX res:
<http://www.repubblica.it/economia/>"
        + " SELECT ?s ?p ?o "
        + " WHERE { ?s ?p ?o ."
//        + "      ?o rdf:type j.0:html "
        + " FILTER regex(str(?s),
\"http://www.repubblica.it/economia/rapporti\", \"i\") "
        + " }";

    String results = p.execSelectQuery(query,"N-TRIPLE");
    System.out.println(results);
    assertNotNull( results );
}

@Test
public void TestPersistenceQueryResultFormattedAsTriple(){
    TriplesDataProvider p = new TriplesDataProvider();

    String query= "PREFIX rdf: <http://www.w3.org/1999/02/22-
```

```

rdf-syntax-ns#>" +
+ " PREFIX owl: <http://www.w3.org/2002/07/owl#>" +
+ " PREFIX xsd:
<http://www.w3.org/2001/XMLSchema#>" +
+ " PREFIX rdfs: <http://www.w3.org/2000/01/rdf-
schema#>" +
+ " PREFIX j.0:
<http://titan.dcs.bbk.ac.uk/~srapis01#>" +
+ " SELECT ?s"
+ " WHERE { ?s rdfs:subClassOf j.0:WebResource
}";

String results = p.execSelectQuery(query, "N-TRIPLE");
System.out.println(results);
assertNotNull( results );
}

@Test
public void TestPersistenceQuery(){
    TriplesDataProvider p = new TriplesDataProvider();

    String query= "PREFIX rdf: <http://www.w3.org/1999/02/22-
rdf-syntax-ns#>" +
+ " PREFIX owl: <http://www.w3.org/2002/07/owl#>" +
+ " PREFIX xsd:
<http://www.w3.org/2001/XMLSchema#>" +
+ " PREFIX rdfs: <http://www.w3.org/2000/01/rdf-
schema#>" +
+ " PREFIX j.0:
<http://titan.dcs.bbk.ac.uk/~srapis01#>" +
+ " SELECT ?s"
+ " WHERE { ?s rdfs:subClassOf j.0:WebResource
}";

ResultSet results = p.execSelectQuery(query );
assertNotNull( results );
assertNotNull( results );

List<String> vars = results.getResultVars();
for ( ; results.hasNext() ; ){
    QuerySolution soln = results.nextSolution();
    System.out.println();
    vars.forEach( v-> { System.out.print( v + ":" +
soln.get( v ) + "; " ); });
}

```

```

        }

    }

    @Test
    public void TestDeleteUrl(){
        TriplesDataProvider p = new TriplesDataProvider();

        String urlString =
"http://www.repubblica.it/economia/rapporti/osserva-italia/eventi";
        assertTrue( p.deleteLinkOutPages(urlString) );

        String query= "PREFIX rdf: <http://www.w3.org/1999/02/22-
rdf-syntax-ns#>"
                + " PREFIX owl: <http://www.w3.org/2002/07/owl#>"
                + " PREFIX xsd:
<http://www.w3.org/2001/XMLSchema#>"
                + " PREFIX rdfs: <http://www.w3.org/2000/01/rdf-
schema#>"
                + " PREFIX j.0:
<http://titan.dcs.bbk.ac.uk/~srapis01#>"
                + " SELECT ?s ?o "
                + " WHERE { ?s j.0:linksOut ?o ."
                + " FILTER regex(str(?s), \"" + urlString + "\", "
"\\"i\\") "
                + " }";

        ResultSet results = p.execSelectQuery(query);
        assertTrue( results.hasNext() == false );
    }

    /*@Test
    public void TestReasonerInDataModel(){
        Model schema = null;
        Reasoner reasoner = ReasonerRegistry.getRDFSReasoner();
        reasoner.bindSchema(schema);

    } */

    @Test
    public void TestMakeDisjoinResourcesModel(){
        JenaResourceModel m = new JenaResourceModel();
        m.MakeDisjointResources();
    }
}

```

```

    m.getWebResources().forEach((k1,v1)-> {
        m.getWebResources().forEach((k2,v2)-> {
            if ( ! v1.equals( v2 ) ){
                Assert.assertTrue(v1.isDisjointWith(v2));
            }
        });
    });
}
}

```

8.2.1.4 beans_test.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">

    <bean id="emf"
class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
        <property name="dataSource" ref="dataSource" />
        <property name="packagesToScan"
value="wonts.data.services.model" />
        <property name="jpaVendorAdapter">
            <bean
class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter"
/>
            </property>
        <property name="jpaProperties">
            <props>
                <prop key="hibernate.hbm2ddl.auto">create-drop</prop>
                <prop
key="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</prop>
            </props>
        </property>
    </bean>

    <bean id="dataSource"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName" value="com.mysql.jdbc.Driver" />

```

```
/>
    <property name="url"
value="jdbc:mysql://localhost:3306/wontsapp_test" />
    <property name="username" value="wontsuser" />
    <property name="password" value="user$wonts" />
</bean>

<bean id="transactionManager"
class="org.springframework.orm.jpa.JpaTransactionManager">
    <property name="entityManagerFactory" ref="emf" />
</bean>

<context:component-scan base-package="wonts.data.services" />

<!-- Root Context: defines shared resources visible to all other
web components -->
</beans>
```

8.2.2 wonts.data.services.provider

8.2.2.1 OntologyServiceTest.java

```
package wonts.data.services.provider;

import static org.junit.Assert.assertTrue;

import java.util.List;
import java.util.stream.Collectors;

import org.junit.Test;
import org.springframework.beans.factory.annotation.Autowired;

import wonts.data.services.ContextConfigurationTest;
import wonts.data.services.model.Ontology;
import wonts.data.services.model.Project;
import wonts.data.services.model.User;

public class OntologyServiceTest extends ContextConfigurationTest{

    @Autowired
    OntologyServiceProvider ontologyServiceProvider;

    @Autowired
    ProjectServiceProvider projectServiceProvider;
```

```
@Autowired
UserServiceProvider userServiceProvider;

@Test
public void GetAllActiveOntologiesTest(){

    int maxontology=10;
    for(int x=0; x < maxontology; x++){
        Ontology o = new Ontology("Test " + x, "Test "+ x);
        ontologyServiceProvider.saveUpdate(o);
    }

    List<Ontology> ontologies =
ontologyServiceProvider.getActiveOntologiesToProcess();

    assertTrue( ontologies.size() == maxontology );
}

@Test
public void CheckProjectInAllActiveOntologiesTest(){

    User u = userServiceProvider.createUser("Salvatore",
"sa@sa.it", "Password1");

    Project project = projectServiceProvider.saveUpdate(new
Project("Test", "Test d", u));

    String name = "ontologytest.owl";
    Ontology ontology = new Ontology(name, name, project);
    ontology = ontologyServiceProvider.saveUpdate(ontology);
    final int ontologyid = ontology.getOntologyId();
    List<Ontology> ontologies =
ontologyServiceProvider.getActiveOntologiesToProcess();

    assertTrue( ontologies.stream().filter(o->
o.getOntologyId() == ontologyid)
        .collect(Collectors.toList()).get(0)
        .getProject() != null &&
ontologies.get(0).getProject().getProjectId() > 0  );

}
```

}

8.2.2.2 ProjectServiceTest.java

```
package wonts.data.services.provider;

import static org.junit.Assert.*;
import java.util.List;
import org.junit.Test;
import org.springframework.beans.factory.annotation.Autowired;
import wonts.data.services.ContextConfigurationTest;
import wonts.data.services.model.Project;
import wonts.data.services.model.User;

public class ProjectServiceTest extends ContextConfigurationTest{

    @Autowired
    ProjectServiceProvider projectServiceProvider;

    @Autowired
    UserServiceProvider userServiceProvider;

    @Test
    public void addProjectTest(){
        User user = userServiceProvider.createUser("salvoSalvo",
        "salvo@salvo.com", "Password1");
        Project project = new Project("name project", "description
        project", user);

        project = projectServiceProvider.saveUpdate(project);
        assertTrue(project.getProjectId()>0);
    }

    @Test
    public void getProjectTest(){
        User user = userServiceProvider.createUser("salvoSalvo",
        "salvo@salvo.com", "Password1");
        Project project = new Project("name project", "description
        project", user);
        Project expected =
    }
}
```

```
projectServiceProvider.saveUpdate(project);
    Project actual =
projectServiceProvider.getProject(expected.getProjectId());
        assertEquals(expected, actual);
}

@Test
public void getProjectByUserTest(){
    User user = userServiceProvider.createUser("salvoSalvo",
"salvo@salvo.com", "Password1");

    Project project1 = new Project("name project 1",
"description project 1", user);

    Project project2 = new Project("name project 2",
"description project 2", user);

    projectServiceProvider.saveUpdate(project1);
    projectServiceProvider.saveUpdate(project2);

    List<Project> projects =
projectServiceProvider.getProjects(user);

    assertTrue(projects.size() == 2);

}

}
```

8.2.2.3 UrlServiceProviderTest.java

```
package wonts.data.services.provider;

import static org.junit.Assert.*;

import java.util.List;

import org.junit.Test;
import org.springframework.beans.factory.annotation.Autowired;

import wonts.data.services.ContextConfigurationTest;
```

```
import wonts.data.services.model.Url;
import wonts.data.services.model.UrlToCrawl;

public class UrlServiceProviderTest extends ContextConfigurationTest
{

    @Autowired
    UrlServiceProvider urlServiceProvider;

    String urlValue =
"http://www.dcs.bbk.ac.uk/~srapis01/index.html";
    String urlDescription = "MSc in Advanced Computing Technologies";

    @Test
    public void insertNewUrlToCrawl(){

        Url url = new Url(urlValue, urlDescription);
        UrlToCrawl u = urlServiceProvider.saveUpdate( url );
        assertTrue(u.getUrlToCrawlId()>0);
    }

    @Test
    public void updateUrlToCrawl(){
        Url url = new Url(urlValue, urlDescription);
        UrlToCrawl urlToCrawl = urlServiceProvider.saveUpdate(
url );
        urlToCrawl.setActive(true);
        UrlToCrawl u = urlServiceProvider.saveUpdate(urlToCrawl);
        assertTrue(u.isActive() && u.getUrlToCrawlId() ==
urlToCrawl.getUrlToCrawlId() );
    }

    @Test
    public void getAllActiveUrlToCrawl(){
        Url url = new Url(urlValue, urlDescription);
        UrlToCrawl u = urlServiceProvider.saveUpdate( url );
        u.setActive(true);
        u.setStatus(UrlToCrawl.Status.isToCrawl.ordinal());

        UrlToCrawl expected = urlServiceProvider.saveUpdate( u);

        List<UrlToCrawl> actual =
urlServiceProvider.getAllActiveUrlToCrawl();
    }
}
```

```
        assertEquals(expected, actual.get(0));
    }

    @Test
    public void insertUrlToCrawlTestNoDuplicatesUrl(){
        Url url = new Url(urlValue, urlDescription);
        UrlToCrawl urlToCrawl1 = urlServiceProvider.saveUpdate(
url );
        UrlToCrawl urlToCrawl2 = urlServiceProvider.saveUpdate( url
);
        assertEquals(urlToCrawl1, urlToCrawl2);
    }

}
```

8.2.2.4 UserServiceProviderTest.java

```
package wonts.data.services.provider;

import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertTrue;

import java.util.List;

import org.junit.Test;
import org.springframework.beans.factory.annotation.Autowired;
import wonts.data.services.model.User;
import wonts.data.services.provider.UserServiceProvider;

import wonts.data.services.ContextConfigurationTest;

public class UserServiceProviderTest extends
ContextConfigurationTest {

    @Autowired
    private UserServiceProvider userProvider;

    @Test
    public void testUserServiceProviderBeNull(){
```

```
        assertTrue(userProvider != null);
    }

    @Test
    public void testCreateNewUser(){
        User u = userProvider.createUser("salvoSalvo",
"salvo@salvo.com", "Password1");
        assertTrue(u.getId() > 0 );
    }

    @Test
    public void testCreateNewFullUser(){
        String name = "Salvatore";
        String surname = "Rapisarda";
        String image = "face0.jpg";
        userProvider.createUser("salvoSalvo", "salvo@salvo.com",
>Password1", name, surname, image);
        User u= userProvider.getUserByUsername("salvoSalvo");
        assertTrue(u != null );
        assertTrue(name == u.getName());
        assertTrue(surname == u.getSurname());
        assertTrue(image==u.getImage());
    }

    @Test(expected= IllegalArgumentException.class)
    public void testCreateUserLessThan6chars() {
        userProvider.createUser("salvo", "salvo@salvo.com",
>Password1");
    }

    @Test(expected= IllegalArgumentException.class)
    public void testPasswordUserLessShuldContainNumber() {
        userProvider.createUser("salvosalvo", "salvo@salvo.com",
>Password");
    }

    @Test(expected= IllegalArgumentException.class)
    public void testUserNameShouldNotBeEmpty() {
        userProvider.createUser("", "salvo@salvo.com",
>Password1");
    }

    @Test(expected= IllegalArgumentException.class)
```

```
public void testEmailShouldNotBeEmpty() {
    userProvider.createUser("salvosalvo", "", "Password1");
}

@Test(expected= IllegalArgumentException.class)
public void testPasswordShouldNotBeEmpty() {
    userProvider.createUser("salvosalvo", "salvo@salvo.com",
    "");
}

@Test
public void test GetUserByUsername() {
    userProvider.createUser("salvoSalvo", "salvo@salvo.com",
    "Password1");
    User u= userProvider.getUserByUsername("salvoSalvo");
    assertTrue(u != null );
}

@Test
public void test GetUserByEmail() {
    userProvider.createUser("salvoSalvo", "salvo@salvo.com",
    "Password1");
    User u= userProvider.getUserByEmail("salvo@salvo.com");
    assertTrue(u != null );
}

@Test
public void testSaveUpdate() {
    userProvider.createUser("salvoSalvo", "salvo@salvo.com",
    "Password1");
    User u= userProvider.getUserByUsername("salvoSalvo");
    String expected_name = "Salvo2";
    String expected_email = "salvo@salvo2.com";

    u.setName(expected_name);
    u.setEmail(expected_email);

    userProvider.saveUpdate(u);

    u= userProvider.getUserByUsername("salvoSalvo");

    assertTrue(u.getEmail() == expected_email);
    assertTrue(u.getName() == expected_name);
}
```

```
@Test
public void testIsUsernameAvailable() {
    String userName = "salvosalvo";
    userProvider.createUser("salvatore", "salvo@salvo.com",
"Password1");
    assertTrue( userProvider.isUsernameAvailable(userName) );
}

@Test
public void testIsUsernameNotAvailable() {
    String userName = "salvatore";
    userProvider.createUser(userName, "salvo@salvo.com",
>Password1);
    assertFalse(userProvider.isUsernameAvailable(userName) );
}

@Test
public void testGetAllUsersInDB() {

    userProvider.createUser("userName", "salvo@salvo.com",
>Password1);

    List<User> users = userProvider.getAllUsers();
    assertTrue(users.size() > 0);
}

}
```

8.2.3 wonts.data.services.repository

8.2.3.1 OntologyRepositoryTest.java

```
package wonts.data.services.repository;

import static org.junit.Assert.*;
import java.util.List;

import org.junit.Test;
import org.springframework.beans.factory.annotation.Autowired;

import wonts.data.services.ContextConfigurationTest;
import wonts.data.services.dao.OntologyRepository;
import wonts.data.services.model.Ontology;
import wonts.data.services.model.OntologyStatus;
```

```
public class OntologyRepositoryTest extends ContextConfigurationTest
{
    @Autowired
    OntologyRepository ontologyRepository;

    String ontologyName = "ontology name test";
    String ontologyDescription = "ontology description test";

    @Test
    public void insertNewProject() {
        Ontology ontology = createNewOntology();

        ontology = ontologyRepository.saveUpdate(ontology);

        assertTrue( ontology.getOntologyId() > 0 );
    }

    private Ontology createNewOntology() {
        Ontology o = new Ontology(ontologyName,
        ontologyDescription);
        return o;
    }

    @Test
    public void getOntologyByIdTest(){
        Ontology o =
        ontologyRepository.saveUpdate(createNewOntology());

        Ontology actual =
        ontologyRepository.getOntology(o.getOntologyId());

        assertEquals(o.getOntologyId(), actual.getOntologyId());
    }

    @Test
    public void setActiveOntologyTest(){
        Ontology o =
        ontologyRepository.saveUpdate(createNewOntology());

        o.setActive(true);
    }
}
```

```
Ontology actual = ontologyRepository.saveUpdate(o);
assertTrue(actual.isActive());
}

@Test
public void ontologyIsActiveByDefaultTest(){
    Ontology actual =
ontologyRepository.saveUpdate(createNewOntology());
    assertTrue(actual.isActive());
}

@Test
public void getOntologiesStatusTest(){
    ontologyRepository.saveUpdate(createNewOntology());

    Ontology o = new Ontology("ontology processed", "ontology
processed");
    o.setStatus(OntologyStatus.Processed.ordinal());
    ontologyRepository.saveUpdate(o);

    List<Ontology> actual =
ontologyRepository.getOntologiesByStatus(OntologyStatus.Processed);
    assertTrue( actual.size() == 1 ) ;

    actual =
ontologyRepository.getOntologiesByStatus(OntologyStatus.ToProcess);
    assertTrue( actual.size() == 1 ) ;

    actual =
ontologyRepository.getOntologiesByStatus(OntologyStatus.Processing);
    assertTrue( actual.size() == 0 ) ;
}

}
```

8.2.3.2 ProjectRepositoryTest.java

```
package wonts.data.services.repository;

import static org.junit.Assert.assertNotNull;
import static org.junit.Assert.assertTrue;

import java.util.List;

import org.junit.Test;
import org.springframework.beans.factory.annotation.Autowired;

import wonts.data.services.ContextConfigurationTest;
import wonts.data.services.dao.OntologyRepository;
import wonts.data.services.dao.ProjectRepository;
import wonts.data.services.dao.UserRepository;
import wonts.data.services.model.Ontology;
import wonts.data.services.model.Project;
import wonts.data.services.model.User;

public class ProjectRepositoryTest extends ContextConfigurationTest
{

    @Autowired
    ProjectRepository projectRepository;

    @Autowired
    UserRepository userRepository;

    @Autowired
    OntologyRepository ontologyRepository;

    String ontologyName = "ontology name test";
    String ontologyDescription = "ontology description test";

    private Project createNewProject(){
        User userTest = new User("userName", "pwd",
        "srapis01@dcs.bbk.ac.uk");

        userTest = userRepository.saveUpdate(userTest);

        Project p = new Project("project test name", "Project
```

```
Description test", userTest);
        p = projectRepository.saveUpdate(p);
        return p;
    }

@Test
public void insertNewProject() {
    assertTrue( createNewProject().getProjectId() > 0 );
}

@Test
public void getProjectById() {
    Project p = createNewProject();

    Project actual =
projectRepository.getProject(p.getProjectId());
    assertNotNull( actual );
}

@Test
public void getProjectByUserTest() {

    Project p = createNewProject();

    List<Project> actual =
projectRepository.getProjects(p.getUser());
    assertNotNull( actual );
    assertTrue(actual.size() > 0 );
}

@Test
public void deleteProject(){
    Project p = createNewProject();

    projectRepository.deleteProject(p);

    p = projectRepository.getProject(p.getProjectId());

    assertTrue(p==null);
}

@Test
```

```
public void addOntologyToProject(){
    Ontology ontologyTest = createNewOntology();

    User userTest = new User("userName", "pwd",
    "srapis01@dcs.bbk.ac.uk");
    userTest = userRepository.saveUpdate(userTest);

    ontologyTest =
ontologyRepository.saveUpdate(ontologyTest);

    Project p = new Project("project test name", "Project
Description test", userTest, ontologyTest);
    p = projectRepository.saveUpdate(p);

    Project po =
projectRepository.getProjectAndOntology(p.getProjectId());

    assertNotNull(po.getOntologies());
    assertTrue(po.getOntologies().size()>0);

    ontologyTest.setProject(p);
    ontologyTest = ontologyRepository.saveUpdate(ontologyTest);

    List<Ontology> o =
ontologyRepository.getOntologiesByProject(p);

    assertTrue(o != null && o.size()>0 );
}

private Ontology createNewOntology() {
    Ontology o = new Ontology(ontologyName,
ontologyDescription);
    return o;
}

}

8.2.3.3 UrlRepositoryTest.java
package wonts.data.services.repository;
```

```
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;

import org.junit.Test;
import org.springframework.beans.factory.annotation.Autowired;

import wonts.data.services.ContextConfigurationTest;
import wonts.data.services.dao.UrlRepositoritory;
import wonts.data.services.model.Url;

public class UrlRepositoryTest extends ContextConfigurationTest {
    @Autowired
    UrlRepositoritory urlRepository;

    String urlValue= "http://something/";
    String urlDescription = "it is not really necessary";

    @Test
    public void insterNewUrlTest() {
        Url url = createNewSampleUrl();
        url = urlRepository.saveUpdate(url);
        assertTrue(url.getUrlId()>0 );
    }

    private Url createNewSampleUrl() {
        Url u = new Url( urlValue, urlDescription );
        return u;
    }

    @Test
    public void getUrlByIdTest(){
        Url u = urlRepository.saveUpdate(createNewSampleUrl());
        Url actual = urlRepository.getUrl(u.getUrlId());

        assertEquals(u.getUrlId(), actual.getUrlId());
    }

    @Test
    public void getUrlByValueTest(){
        Url u = urlRepository.saveUpdate(createNewSampleUrl());
        Url actual = urlRepository.getUrl(u.getUrlValue());
```

```
        assertEquals(u.getUrlId(), actual.getUrlId());
    }
}

8.2.3.4 UrlToCrawlRepositoryTest.java
package wonts.data.services.repository;

import static org.junit.Assert.*;

import java.util.List;

import org.junit.Test;
import org.springframework.beans.factory.annotation.Autowired;

import wonts.data.services.ContextConfigurationTest;
import wonts.data.services.dao.UrlToCrawlRepository;
import wonts.data.services.model.Url;
import wonts.data.services.model.UrlToCrawl;

public class UrlToCrawlRepositoryTest extends
ContextConfigurationTest {

    private String urlValue =
"http://www.dcs.bbk.ac.uk/~srapis01/index.html";
    private String urlDescription = "Description URL";

    @Autowired
    UrlToCrawlRepository urlToCrawlRepository;

    private UrlToCrawl getSampleUrlToCrawl(){
        UrlToCrawl u = new UrlToCrawl( new Url(urlValue,
urlDescription ) );
        u.setStatus(UrlToCrawl.Status.isTrue.ordinal());
        u.setActive(true);
        return u;
    }

    @Test
    public void insertUrlToCrawlTest(){
        UrlToCrawl urlToCrawl = urlToCrawlRepository.saveUpdate(
getSampleUrlToCrawl() );
        assertTrue(urlToCrawl.getUrlToCrawlId()>0);
    }
}
```

```
}

@Test
public void insertUrlToCheckIsActiveByDefaultTest(){
    UrlToCrawl urlToCrawl = urlToCrawlRepository.saveUpdate(
getSampleUrlToCrawl() );
    assertTrue(urlToCrawl.isActive());
}

@Test
public void getUrlToCrawlTest(){
    UrlToCrawl expected = urlToCrawlRepository.saveUpdate(
getSampleUrlToCrawl() );

    UrlToCrawl actual =
urlToCrawlRepository.getUrlToCrawl(expected.getUrlToCrawlId());
    assertEquals(expected, actual);
}

@Test
public void getAllActiveUrlToCrawl(){
    UrlToCrawl expected = urlToCrawlRepository.saveUpdate(
getSampleUrlToCrawl() );

    List<UrlToCrawl> actual =
urlToCrawlRepository.getActiveUrl(UrlToCrawl.Status.isToCrawl);

    assertEquals(expected, actual.get(0));
}

}
```

8.2.3.5 UserRepositoryTest.java

```
package wonts.data.services.repository;

import static org.junit.Assert.*;

import org.junit.Test;
import org.springframework.beans.factory.annotation.Autowired;

import wonts.data.services.ContextConfigurationTest;
```

```
import wonts.data.services.dao.UserRepository;
import wonts.data.services.model.User;

public class UserRepositoryTest extends ContextConfigurationTest {

    @Autowired
    private UserRepository userRepository;

    @Test
    public void testUserServiceProvidedBeNull(){
        assertTrue(userRepository != null);
    }

    String userEmail = "srapis01@dcs.bbk.ac.uk";
    String userPassword = "pwd";
    String userName = "salvatore";

    private User createNewSampleUser (){
        User user = new User(userName, userPassword, userEmail);
        return userRepository.saveUpdate(user);
    }

    @Test
    public void testCreateNewUser(){
        assertTrue(createNewSampleUser().getId() > 0 );
    }

    @Test
    public void test GetUserByEmail(){
        String expected = userEmail;

        createNewSampleUser();
        User u = userRepository.getUserByEmail(expected);
        assertNotNull(u);

        String actual = u.getEmail();
        assertEquals("the email should match.", expected, actual);
    }

    @Test
    public void test GetUserByUserName(){
        String expected = userName;
```

```
        createNewSampleUser();
        User u = userRepository.getUserByUsername(expected);
        assertNotNull(u);

        String actual = u.getUsername();
        assertEquals("the email should match.", expected, actual);
    }

    @Test
    public void testIsUserNameNotAvailable(){
        User u = createNewSampleUser();
        assertNotNull(u);

        assertFalse(userRepository.isUsernameAvailable(userName));
    }

    @Test
    public void testIsUserNameAvailable(){
        assertTrue(userRepository.isUsernameAvailable(userName));
    }

}
```

8.3 data.services (pom.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>wonts</groupId>
    <artifactId>data.services</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>data.services</name>
    <url>http://maven.apache.org</url>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <java-version>1.8</java-version>
    
```

```
<log4j-version>1.2.16</log4j-version>
<lombok-version>1.14.4</lombok-version>
<junit-version>4.12</junit-version>
<jena-jdbc-core-version>1.1.0</jena-jdbc-core-version>
<jena-sdb-version>1.5.0</jena-sdb-version>
<org.springframework-version>4.1.3.RELEASE</org.springframework-
version>
<org.spring-security-core>4.0.1.RELEASE</org.spring-security-
core>
<hibernate.version>4.3.7.Final</hibernate.version>
<org.aspectj-version>1.8.5</org.aspectj-version>
<org.slf4j-version>1.7.10</org.slf4j-version>
<jackson-version>2.4.4</jackson-version>
</properties>
<build>
<plugins>
<plugin>
<artifactId>maven-compiler-plugin</artifactId>
<version>3.1</version>
<configuration>
<source>${java-version}</source>
<target>${java-version}</target>
</configuration>
</plugin>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-javadoc-plugin</artifactId>
<version>2.10.3</version>
<configuration>
<show>private</show>
<nohelp>true</nohelp>
</configuration>
</plugin>
</plugins>
</build>
<dependencies>
<dependency>
<groupId>log4j</groupId>
<artifactId>log4j</artifactId>
<version>${log4j-version}</version>
</dependency>
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
```

```
<version>${junit-version}</version>
<scope>test</scope>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>${lombok-version}</version>
</dependency>

<!-- spring framework -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>${org.springframework-version}</version>
    <exclusions>
        <!-- Exclude Commons Logging in favor of SLF4j -->
        <exclusion>
            <groupId>commons-logging</groupId>
            <artifactId>commons-logging</artifactId>
        </exclusion>
    </exclusions>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
    <version>${org.springframework-version}</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-orm</artifactId>
    <version>${org.springframework-version}</version>
</dependency>

<!-- Spring security -->
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-core</artifactId>
    <version>${org.spring-security-core}</version>
</dependency>

<dependency>
    <groupId>javax.inject</groupId>
    <artifactId>javax.inject</artifactId>
    <version>1</version>
```

```
</dependency>

<!-- apache jena -->
<dependency>
    <groupId>org.apache.jena</groupId>
    <artifactId>jena-jdbc-core</artifactId>
    <version>${jena-jdbc-core-version}</version>
</dependency>
<dependency>
    <groupId>org.apache.jena</groupId>
    <artifactId>jena-sdb</artifactId>
    <version>1.5.0</version>
</dependency>
<dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>2.4</version>
</dependency>
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.31</version>
</dependency>
<!-- Hibernate -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>${hibernate.version}</version>
</dependency>
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-entitymanager</artifactId>
    <version>${hibernate.version}</version>
</dependency>

<dependency>
    <groupId>xml-apis</groupId>
    <artifactId>xml-apis</artifactId>
    <version>1.4.01</version>
</dependency>

<!-- Apache common -->
<dependency>
    <groupId>org.apache.commons</groupId>
```

```
<artifactId>commons-lang3</artifactId>
<version>3.3.2</version>
</dependency>
</dependencies>

</project>

8.4 solar.services (src/main/java)
8.4.1 wonts.solr.service

8.4.1.1 AppConfiguration.java
package wonts.solr.service;

import static wonts.solr.service.SpringExtension.SpringExtProvider;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.ImportResource;

import akka.actor.ActorSystem;

@ImportResource("classpath:data_context_beans.xml")
@Configuration
public class AppConfiguration {
    // the application context is needed to initialize the Akka
    Spring Extension
    @Autowired
    private ApplicationContext applicationContext;

    /**
     * Actor system singleton for this application.
     */
    @Bean
    public ActorSystem actorSystem() {
        ActorSystem system = ActorSystem.create("WontsSolr");
        SpringExtProvider.get(system).initialize(applicationContext);
        return system;
    }
}
```

}

8.4.1.2 GetCrawledResources.java

```
package wonts.solr.service;

import static wonts.solr.service.SpringExtension.SpringExtProvider;

import java.util.Date;
import java.util.Set;

import javax.inject.Inject;
import javax.inject.Named;

import org.springframework.context.annotation.Scope;

import wonts.data.services.model.UrlToCrawl;
import wonts.data.services.provider.UrlServiceProvider;
import wonts.solr.service.messages.GetResoucesToIndexMsg;
import wonts.solr.service.messages.ResourceIndexedMsg;
import wonts.solr.service.messages.SolrDeleteResourceMsg;
import wonts.solr.service.messages.SolrDownloadMsg;
import wonts.solr.service.messages.SolrUpdateMsg;
import wonts.solr.service.messages.responce.GetResourcesToIndexMsgRep;
import wonts.solr.service.messages.responce.SolrDownloadMsgResp;
import wonts.solr.service.messages.responce.SolrUpdateMsgResp;
import akka.actor.ActorRef;
import akka.actor.UntypedActor;
import akka.event.Logging;
import akka.event.LoggingAdapter;

@Named("getCrawledResources")
@Scope("prototype")
public class GetCrawledResources extends UntypedActor{
    final LoggingAdapter log =
Logging.getLogger(getContext().system(), this);

    @Inject
    UrlServiceProvider urlServiceProvider;

    ActorRef getResourcesToIndex;

    ActorRef solrDownloadResource;
```

```
ActorRef solrUpdateResource;

ActorRef solrDeleteResource;

public GetCrawledResources(){

    getResourcesToIndex = getContext().actorOf(
        SpringExtProvider.get(getContext().system()).props("getResourcesToIndex"),
        "getResourcesToIndex" );

    solrDownloadResource = getContext().actorOf(
        SpringExtProvider.get(getContext().system()).props("solrDownloadResource"),
        "solrDownloadResource" );

    solrUpdateResource = getContext().actorOf(
        SpringExtProvider.get(getContext().system()).props("solrUpdateResource"),
        "solrUpdateResource" );

    solrDeleteResource = getContext().actorOf(
        SpringExtProvider.get(getContext().system()).props("solrDeleteResource"),
        "solrDeleteResource" );
}

@Override
public void onReceive(Object message) throws Exception {
    if ( message == "getCrawledResources"){
        log.info("checking resources to crawl.");
        urlServiceProvider.getAllActiveUrlToIndex().foreach(
p->{
            GetResoucesToIndexMsg msg = new
GetResoucesToIndexMsg();
            msg.setUrlCrawled(p.getUrl().getUrlValue());
            msg.setUrlToCrawlId(p.getUrl().getUrlId());
            getResourcesToIndex.tell(msg, getSelf());
        urlServiceProvider.updateStatusUrlToCrawl(p.getUrlToCrawlId(),

```

```

UrlToCrawl.Status.isIndexing );
    });

}else if ( message instanceof GetResourcesToIndexMsgRep
){

    // message from Jena manager
    // now we can send a messages
    // to the actor for download the resource
    GetResourcesToIndexMsgRep msg =
(GetResourcesToIndexMsgRep ) message;
    UrlToCrawl urlToCrawl =
urlServiceProvider.getUrlToCrawl(msg.getUrlToCrawlId());
    if ( urlToCrawl != null &&
msg.getResouruces()!=null){
        urlToCrawl.setModified(new Date());

        urlToCrawl.setTotalResources(msg.getResouruces().size());

        urlToCrawl.setResourcesToIndex(msg.getResouruces().size());
        urlToCrawl.setTimeStartOnIndexing(new
Date().getTime());
        urlServiceProvider.saveUpdate(urlToCrawl);
        Set<String> res = (msg.getResouruces());
        for(String url : res ){
            //res.forEach(url -> {
            SolrDownloadMsg downloadMsg = new
SolrDownloadMsg(url, urlToCrawl.getUrlToCrawlId());
                log.info("sending message to download {}",

downloadMsg );
                solrDownloadResource.tell(downloadMsg,
getSelf() );

            //});
        }
    }
}else if (message instanceof SolrDownloadMsgResp){
    // The Download manager has download a resource,
    // therefore that is ready to be index in solar
    SolrDownloadMsgResp res = (SolrDownloadMsgResp)
message;
    SolrUpdateMsg msg = new SolrUpdateMsg(res.getUrl(),
res.getPath().toFile().getName(),
res.getPath().toFile().getAbsolutePath(), res.getUrlToCrawlId() );
    solrUpdateResource.tell(msg, getSelf());
}

```

```

        }else if ( message instanceof SolrUpdateMsgResp ) {
            SolrUpdateMsgResp res = (SolrUpdateMsgResp) message;
            SolrDeleteResourceMsg msg = new
SolrDeleteResourceMsg(res.getFilePath(), res.getUrlToCrawlId());
            solrDeleteResource.tell(msg, getSelf());
        }else if ( message instanceof ResourceIndexedMsg ){
            // This message is received when a resources has
finished to been indexed
            // it contains the status of the resources
            ResourceIndexedMsg msgIndexed =
(ResourceIndexedMsg)message;
            log.info("remove resource indexed id: " +
msgIndexed.getUrlToCrawlId() + " - res: " +
msgIndexed.getUrlResource() + " - status: " + msgIndexed.getStatus()
);
            UrlToCrawl urlToCrawl =
urlServiceProvider.getUrlToCrawl(msgIndexed.getUrlToCrawlId());
            if ( urlToCrawl != null ){
                urlToCrawl.setResourcesToIndex(
urlToCrawl.getResourcesToIndex() -1);
                long timeElapsed = new Date().getTime() -
urlToCrawl.getTimeStartOnIndexing();
                urlToCrawl.setTimeElapsedOnIndexing(timeElapsed);
                if( urlToCrawl.getResourcesToIndex() <= 0 ) {

urlToCrawl.setStatus(UrlToCrawl.Status.isIndexed.ordinal());
                }
                urlServiceProvider.saveUpdate(urlToCrawl);

            }
        }else if (message instanceof Exception) {
            throw (Exception) message;
        }
    }

}

```

8.4.1.3 GetResourcesToIndex.java

```

package wonts.solr.service;

import java.util.HashSet;

```

```
import java.util.Set;

import javax.inject.Named;

import org.springframework.context.annotation.Scope;

import wonts.data.services.triples.TriplesDataProvider;
import wonts.solr.service.messages.GetResoucesToIndexMsg;
import wonts.solr.service.messages.responce.GetResourcesToIndexMsgRep;
import akka.actor.UntypedActor;
import akka.event.Logging;
import akka.event.LoggingAdapter;

import com.hp.hpl.jena.query.QuerySolution;
import com.hp.hpl.jena.query.ResultSet;

@Named("getResourcesToIndex")
@Scope("prototype")
public class GetResourcesToIndex extends UntypedActor {
    final LoggingAdapter log =
Logging.getLogger(getContext().system(), this);

    @Override
    public void onReceive(Object message) throws Exception {
        if ( message instanceof GetResoucesToIndexMsg ) {
            log.debug("received message {}", message);
            Set<String> resources =
processMessage((GetResoucesToIndexMsg)message);
            GetResourcesToIndexMsgRep messageBack = new
GetResourcesToIndexMsgRep(resources,((GetResoucesToIndexMsg)message).g
etUrlToCrawlId());
            getSender().tell(messageBack, getSelf());
        }else if (message instanceof Exception) {
            throw (Exception) message;
    }
}

private Set<String> processMessage(GetResoucesToIndexMsg
message) {

    Set<String> ret = new HashSet<String>();
    TriplesDataProvider p = new TriplesDataProvider();
```

```
String queryString= "PREFIX rdf:  
<http://www.w3.org/1999/02/22-rdf-syntax-ns#>"  
        + " PREFIX owl: <http://www.w3.org/2002/07/owl#>"  
        + " PREFIX xsd:  
<http://www.w3.org/2001/XMLSchema#>"  
        + " PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>"  
        + " PREFIX wonts:  
<http://titan.dcs.bbk.ac.uk/~srapis01#>"  
        + " SELECT ?o ?s "  
        + " WHERE { ?s wonts:linksOut ?o ."  
        + " FILTER regex(str(?s), \"%s\", \"i\") }" ;  
  
queryString = String.format(queryString,  
message.getUrlCrawled());  
  
ResultSet results = p.execSelectQuery(queryString);  
  
while (results.hasNext()) {  
    QuerySolution type = results.next();  
    ret.add(type.get("o").toString());  
    ret.add(type.get("s").toString());  
}  
  
return ret;  
}  
}
```

8.4.1.4 ProcessOntology.java

```
package wonts.solr.service;  
  
import static wonts.solr.service.SpringExtension.SpringExtProvider;  
  
import java.io.File;  
import java.util.ArrayList;  
import java.util.List;  
  
import javax.inject.Inject;  
import javax.inject.Named;  
  
import org.springframework.context.annotation.Scope;
```

```
import wonts.data.services.model.Ontology;
import wonts.data.services.model.Project;
import wonts.data.services.provider.OntologyServiceProvider;
import wonts.data.services.provider.ProjectServiceProvider;
import wonts.data.services.provider.UrlServiceProvider;
import wonts.solr.service.messages.ProcessOntologyMsg;
import wonts.solr.service.messages.ReadOwlFileMsg;
import akka.actor.ActorRef;
import akka.actor.UntypedActor;
import akka.event.Logging;
import akka.event.LoggingAdapter;

@Named("processOntology")
@Scope("prototype")
public class ProcessOntology extends UntypedActor{
    final LoggingAdapter log =
Logging.getLogger(getContext().system(), this);

    @Inject
    OntologyServiceProvider ontologyServiceProvider;

    @Inject
    ProjectServiceProvider projectServiceProvider;

    @Inject
    UrlServiceProvider urlServiceProvider;

    ActorRef readOwlFile;

    public ProcessOntology(){
        readOwlFile = getContext().actorOf(
SpringExtProvider.get(getContext().system()).props("readOwlFile"),
"readOwlFile" );
    }

    @Override
    public void onReceive(Object message) throws Exception {
```

```

        if ( message instanceof ProcessOntologyMsg ){
            processOntology((ProcessOntologyMsg)message);

        }else if (message instanceof Exception) {
            throw (Exception) message;
        }
    }

    private void processOntology(ProcessOntologyMsg message) {
        List<Ontology> ontologies =
ontologyServiceProvider.getActiveOntologiesToProcess();
        if ( ontologies != null ) {
            ontologies.forEach( ont-> {
                Project project = getProjectFromOntology(ont);
                if ( project !=null ) {
                    String name;
                    if ( message.getOntologiesbasePath() ==
null || message.getOntologiesbasePath().trim().isEmpty() ) {
                        name = ont.getOntologyName();
                    }else{
                        name = message.getOntologiesbasePath()
+ File.separator + ont.getOntologyName();
                    }
                    List<String> domains =
getUrlDomainsFormProject(project);
                    ReadOwlFileMsg msg = new
ReadOwlFileMsg(ont.getOntologyId(), name, domains);
                    readOwlFile.tell(msg, getSelf());
                }
            }
        }
    }

    private List<String> getUrlDomainsFormProject(Project project){
        try {
            if (project.getUrls() == null ) return null;
            List<String> ret = new ArrayList<String>();
            project.getUrls().forEach(p -> ret.add(
p.getUrlValue()));
            return ret;
        }catch (Exception e ){
            log.error("error occured on ProcessOntology actor -
getUrlDomainsFormProject", e);
        }
    }
}

```

```
        log.error("project.getUrls(): "+ project.getUrls());
        return null;
    }

    private Project getProjectFromOntology(Ontology ontology){
        try{
            return
projectServiceProvider.getProjectAndUrls(ontology.getProject().getProj
ectId());
        }catch (Exception e ){
            log.error( "error occured on ProcessOntology actor - "
getProjectFromOntology");
            return null;
        }
    }
}
```

8.4.1.5 ReadOwlFile.java

```
package wonts.solr.service;

import static wonts.solr.service.SpringExtension.SpringExtProvider;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.util.List;
import java.util.UUID;

import javax.inject.Inject;
import javax.inject.Named;

import org.apache.jena.iri.IRI;
import org.apache.jena.iri.IRIFactory;
import org.springframework.context.annotation.Scope;

import scala.concurrent.Await;
import scala.concurrent.Future;
import scala.concurrent.duration.Duration;
import scala.reflect.ClassTag;
```

```
import scala.reflect.ClassTag;
import wonts.data.services.model.Ontology;
import wonts.data.services.model.OntologyStatus;
import wonts.data.services.provider.OntologyServiceProvider;
import wonts.solr.service.messages.ReadOwlFileMsg;
import wonts.solr.service.messages.SolrQueryMessage;
import wonts.solr.service.messages.responce.ReadOwlFileMsgResp;
import wonts.solr.service.messages.responce.SolrQueryMsgResp;
import akka.actor.ActorRef;
import akka.actor.UntypedActor;
import akka.event.Logging;
import akka.event.LoggingAdapter;
import akka.pattern.Patterns;
import akka.util.Timeout;

import com.hp.hpl.jena.ontology.OntClass;
import com.hp.hpl.jena.ontology.OntModel;
import com.hp.hpl.jena.ontology.OntModelSpec;
import com.hp.hpl.jena.query.QueryExecution;
import com.hp.hpl.jena.query.QueryExecutionFactory;
import com.hp.hpl.jena.query.QuerySolution;
import com.hp.hpl.jena.query.ResultSet;
import com.hp.hpl.jena.query.ResultSetFactory;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.rdf.model.Resource;
import com.hp.hpl.jena.reasoner.Reasoner;
import com.hp.hpl.jena.reasoner.ReasonerRegistry;

@Named("readOwlFile")
@Scope("prototype")
public class ReadOwlFile extends UntypedActor {
    final LoggingAdapter log =
Logging.getLogger(getContext().system(), this);
    final String NS = "http://titan.dcs.bbk.ac.uk/~srapis01#";
    final IRIFactory factory = IRIFactory.jenaImplementation();

    ActorRef solrQueryEngine;

    @Inject
    OntologyServiceProvider ontologyServiceProvider;

    public ReadOwlFile() {
        solrQueryEngine = getContext().actorOf(
```

```

SpringExtProvider.get(getContext().system()).props("solrQueryEngine"),
"solrQueryEngine" );
}

@Override
public void onReceive(Object message) throws Exception {
    if ( message instanceof ReadOwlFileMsg){
        Ontology ontology =
ontologyServiceProvider.getOntology(((ReadOwlFileMsg)
message).getId());

        ontology.setStatus(OntologyStatus.Processing.ordinal());
        ontology =
ontologyServiceProvider.saveUpdate(ontology);
        ReadOwlFileMsg msg = (ReadOwlFileMsg) message;
        InputStream in = new
FileInputStream(msg.getOwlFileName());
        OntModel model =
ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM);
        //OntModel model = ModelFactory.createDefaultModel();
        model.read(in, null);
        String queryString = getQuery(msg);
        // Property hasSearchQuery =
model.createOntProperty(NS + "links");
        ResultSet resultSet = executeQuery (model,
queryString);

        ClassTag<SolrQueryMsgResp> tag =
ClassTag$.MODULE$.apply(SolrQueryMsgResp.class);
        //List<Future<SolrQueryMsgResp> > futures = new
ArrayList<Future<SolrQueryMsgResp>>();
        List<String> domains = ((ReadOwlFileMsg)
message).getDomains();
        if ( resultSet != null ){
            while ( resultSet.hasNext() ) {
                QuerySolution query = resultSet.next();
                try{
                    Resource res =
query.get("s").asResource();
                    OntClass ontClass =
model.getOntClass(res.getURI());
                    String qu =

```

```

query.get("o").asLiteral().getValue().toString();

SolrQueryMessage solrMsg = new
SolrQueryMessage(qu);
Timeout timeout = new
Timeout(Duration.create(50, "seconds"));

Future<SolrQueryMsgResp> future =
Patterns.ask(solrQueryEngine, solrMsg, timeout).mapTo(tag);
SolrQueryMsgResp solrResponse =
(SolrQueryMsgResp) Await.result(future, timeout.duration());

if (solrResponse.getResources() != null){

solrResponse.getResources().forEach(uri ->{
    IRI iri = factory.create( uri );
    if (!iri.hasViolation(false))
&& ( domains == null || domains.isEmpty() || checkUrlDomain(domains,
uri) ) ) {

model.createIndividual(uri,ontClass);
}
});

}

//futures.add(future);

/*future.onComplete(
    new OnComplete<SolrQueryMsgResp>() {
        public void onComplete(Throwable
failure, SolrQueryMsgResp solrResponse ) {
            if (failure != null){
                log.info("failure to receive
solrResponse to a query");
            }else{

solrResponse.getResources().forEach(p ->{
    model.createIndividual(p,ontClass);
}
);

}

}, getContext().dispatcher() */
```

```

        }catch(Exception e){
            log.error("readOwlFile actor error ", e);
        }

    }

    /* TODO: Make sure that this is done once all all
the future are completed
    ExecutionContext ec =
ExecutionContext.fromExecutorService(getContext());
    Future<Iterable<SolrQueryMsgResp>> seq =
Futures.sequence(futures, getContext());
    */
    String ontologyProcessedName = ((ReadOwlFileMsg)
message).getOwlFileName() + "_" + UUID.randomUUID().toString() + ".owl";
    File f = new File(ontologyProcessedName);

ontology.setStatus(OntologyStatus.Processed.ordinal());
ontology.setOntologyProcessedName(f.getName());
ReadOwlFileMsgResp msgback = new
ReadOwlFileMsgResp(msg, model);
getSender().tell(msgback, getSelf());
model.write(new FileOutputStream(f));
ontologyServiceProvider.saveUpdate(ontology);

}

}

private boolean checkUrlDomain( List<String> domains,  String
uri){
    try {
        return domains.stream().anyMatch(dom->
uri.startsWith(dom) );
    } catch (Exception e) {
        log.error("ReadOWLfile error on checkUrlDomain ",e);
        return false;
    }
}

```

```
private String getQuery(ReadOwlFileMsg msg) {  
  
    String queryString= "PREFIX rdf:  
<http://www.w3.org/1999/02/22-rdf-syntax-ns#>"  
        + " PREFIX owl: <http://www.w3.org/2002/07/owl#>"  
        + " PREFIX xsd:  
<http://www.w3.org/2001/XMLSchema#>"  
        + " PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>"  
        + " PREFIX wonts:  
<http://titan.dcs.bbku.ac.uk/~srapi01#>"  
        + " SELECT ?s ?o "  
        + " WHERE { ?s wonts:hasSearchQuery ?o }";  
    return queryString;  
}  
  
private ResultSet executeQuery(Model model, String query ){  
    ResultSet res= null;  
  
    Reasoner r = ReasonerRegistry.getRDFSReasoner();  
    r.bindSchema(model);  
    try (QueryExecution qexec =  
QueryExecutionFactory.create(query, model)){  
        res =qexec.execSelect();  
        res = ResultSetFactory.copyResults(res) ;  
        qexec.close();  
    }catch (Exception e) {  
        res = null;  
        log.error("executeQuery in ReadOwlFile", e);  
    }  
    return res;  
}  
}
```

8.4.1.6 ServicesStarter.java

```
package wonts.solr.service;  
  
import static wonts.solr.service.SpringExtension.SpringExtProvider;  
  
import java.util.concurrent.TimeUnit;
```

```
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import scala.concurrent.duration.Duration;
import wonts.solr.service.messages.ProcessOntologyMsg;
import wonts.solr.service.messages.SolrUpdateMsg;
import wonts.solr.service.properties.PropertiesManager;
import akka.actor.ActorRef;
import akka.actor.ActorSystem;
import akka.actor.Cancellable;
import akka.actor.Props;
import akka.actor.ReceiveTimeout;
import akka.actor.UntypedActor;
import akka.event.Logging;
import akka.event.LoggingAdapter;
```

```
public class ServicesStarter {

    private final AnnotationConfigApplicationContext ctx;
    private final ActorSystem system;
    private final Cancellable cancellable;
    private final Cancellable cancellableOntology;
    private final LoggingAdapter log;

    public ServicesStarter() throws Exception{

        ctx = new AnnotationConfigApplicationContext();
        ctx.scan("wonts");
        ctx.refresh();

        system = ctx.getBean(ActorSystem.class);
        log = Logging.getLogger(system, this);

        final Props propsGetCrawledResources =
SpringExtProvider.get(system).props("getCrawledResources");
        final Props propsProcessOntology =
SpringExtProvider.get(system).props("processOntology");

        final ActorRef getCrawledResources = system.actorOf(
propsGetCrawledResources, "getCrawledResources");
        final ActorRef processOntology = system.actorOf(
```

```

propsProcessOntology, "processOntology");

        int timeGetResource =
getIntFromPropertyManager(PropertiesManager.CRAWLER_RESOURCES_SECO
NDS, 300);

        cancellable = system.scheduler().schedule(Duration.Zero(),
Duration.create(timeGetResource,
TimeUnit.SECONDS), getCrawledResources, "getCrawledResources",
system.dispatcher(), null);

        int timeProcessOntology =
getIntFromPropertyManager(PropertiesManager.PROCESS_ONTOLOGY_SECOND
S, 5);
        ProcessOntologyMsg message = new ProcessOntologyMsg();

        message.setOntologiesBasePath(PropertiesManager.getProperty(Prope
rtiesManager.WONTSAPP_ONTOLOGY_UPLOAD));
        cancellableOntology=
system.scheduler().schedule(Duration.Zero(),
Duration.create(timeProcessOntology,
TimeUnit.SECONDS), processOntology, message,
system.dispatcher(), null);

    }

private int getIntFromPropertyManager(String key, int defVal){
    int ret = defVal;
    try {
        String sval = PropertiesManager.getProperty(key);
        if (sval !=null && !sval.trim().isEmpty()){
            ret = Integer.parseInt(sval);
        }
    }catch (Exception e){
        log.error("error on ServicesStarter -
getIntFromPropertyManager", e);
    }
    return ret;
}

public static class Listener extends UntypedActor {
    final LoggingAdapter log =
Logging.getLogger(getContext().system(),

```

```
        this);

    @Override
    public void preStart() {
        // If we don't get any progress within 15 seconds then
the service
        // is unavailable
        // getContext().setReceiveTimeout(Duration.create("15
seconds"));
    }

    public void onReceive(Object msg) {
        log.debug("received message {}", msg);
        if (msg instanceof SolrUpdateMsg) {

            log.info("Current progress: {} %",
                    ((SolrUpdateMsg) msg).getUrl());
        } else if (msg == ReceiveTimeout.getInstance()) {
            getContext().system().shutdown();
        }else {
            unhandled(msg);
        }
    }
}

public static void main(String[] args) throws Exception{
    new ServicesStarter();
}

}
```

8.4.1.7 SolrDeleteResource.java

```
package wonts.solr.service;

import java.io.File;

import javax.inject.Named;

import org.springframework.context.annotation.Scope;

import wonts.solr.service.messages.ResourceIndexedMsg;
import wonts.solr.service.messages.ResourceIndexedMsg.ResourceStatus;
```

```
import wonts.solr.service.messages.SolrDeleteResourceMsg;
import akka.actor.UntypedActor;
import akka.event.Logging;
import akka.event.LoggingAdapter;

@Named("solrDeleteResource")
@Scope("prototype")
public class SolrDeleteResource extends UntypedActor {

    final LoggingAdapter log =
Logging.getLogger(getContext().system(), this);

    @Override
    public void onReceive(Object message) throws Exception {
        if (message instanceof SolrDeleteResourceMsg) {
            SolrDeleteResourceMsg msg =
(SolrDeleteResourceMsg)message;
            try{
                log.debug("received message {}", message);
                boolean res = processMessage(msg);
                getSender().tell(res, getSelf());
            }catch(Exception e){
                log.error("solrDeleteResource error:", e);
            }finally{
                ResourceIndexedMsg msgResourceIndexed = new
ResourceIndexedMsg(msg.getUrlToCrawlId(), msg.getPath(),
ResourceStatus.deleted );
                getSender().tell(msgResourceIndexed, getSelf());
            }
        }else if (message instanceof Exception) {
            throw (Exception) message;
        }
    }

    private boolean processMessage(SolrDeleteResourceMsg message) {
        File f = new File(message.getPath());
        return f.exists() && f.delete();
    }
}
```

8.4.1.8 SolrDowloadResource.java

```
package wonts.solr.service;
```

```
import java.io.File;
import java.io.IOException;
import java.net.MalformedURLException;
import java.net.SocketTimeoutException;
import java.net.URL;
import java.net.URLConnection;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.StandardCopyOption;
import java.util.UUID;

import javax.inject.Named;

import org.springframework.context.annotation.Scope;

import wonts.solr.service.messages.ResourceIndexedMsg;
import wonts.solr.service.messages.SolrDownloadMsg;
import wonts.solr.service.messages.responce.SolrDownloadMsgResp;
import wonts.solr.service.properties.PropertiesManager;
import akka.actor.UntypedActor;
import akka.event.Logging;
import akka.event.LoggingAdapter;

@Named("solrDowloadResource")
@Scope("prototype")
public class SolrDowloadResource extends UntypedActor {
    final LoggingAdapter log =
Logging.getLogger(getContext().system(), this);

    public SolrDowloadResource(){
    }

    @Override
    public void onReceive(Object message) throws Exception {
        if ( message instanceof SolrDownloadMsg ){
            SolrDownloadMsg downloadMsg = (SolrDownloadMsg)
message;
            try{
                log.info("received message {}", message);
                Path pfile =

```

```
processMessage((SolrDownloadMsg)message);
    SolrDownloadMsgResp msg = new
SolrDownloadMsgResp(((SolrDownloadMsg) message).getUrl(), pfile,
downloadMsg.getUrlToCrawlId());
    getSender().tell(msg, getSelf());
}catch(Exception e){
    log.error("error on SolrDowloadResource id: " +
downloadMsg.getUrlToCrawlId() + " - url:" + downloadMsg.getUrl()+
"\n"+ e);
    ResourceIndexedMsg msg = new
ResourceIndexedMsg(downloadMsg.getUrlToCrawlId(),
downloadMsg.getUrl(), ResourceIndexedMsg.ResourceStatus.downloaded );
    getSender().tell(msg, getSelf());
}
}else if (message instanceof Exception) {
    throw (Exception) message;
}

}
```

```
public Path download(String sourceUrl, String targetDirectory)
throws MalformedURLException, IOException, SocketTimeoutException{
```

```
    URL url = new URL(sourceUrl);
    URLConnection con = url.openConnection();
    con.setConnectTimeout(2*60*1000);
    con.setReadTimeout(2*60*1000);
    int indexExt = sourceUrl.lastIndexOf(".");
    String ext = indexExt > 0 ?
sourceUrl.substring(sourceUrl.lastIndexOf(".")) : "";
    String fileName = UUID.randomUUID().toString() + ext;
    fileName = fileName.replace('/', '_');
    Path targetPath = new File((targetDirectory.endsWith("/") ?
targetDirectory : targetDirectory + "/") + fileName).toPath();
    Files.copy(con.getInputStream(), targetPath,
StandardCopyOption.REPLACE_EXISTING);

    return targetPath;
}
```

```
/*
```

```
private String getContentType ( String url ) throws IOException{
    URLConnection connection = new URL (url).openConnection();
    connection.connect();
    return connection.getContentType();
}
*/
private Path processMessage(SolrDownloadMsg message) throws
MalformedURLException, IOException, SocketTimeoutException {
    return this.download(message.getUrl(),
PropertiesManager.getProperty(PropertiesManager.SOLR_RESOURCES_PATH_
DOWNLOAD));
}

}
```

8.4.1.9 SolrQueryEngine.java

```
package wonts.solr.service;

import java.io.IOException;

import javax.inject.Named;

import lombok.Getter;

import org.apache.solr.client.solrj.SolrClient;
import org.apache.solr.client.solrj.SolrQuery;
import org.apache.solr.client.solrj.SolrServerException;
import org.apache.solr.client.solrj.impl.HttpSolrClient;
import org.apache.solr.client.solrj.response.QueryResponse;
import org.springframework.context.annotation.Scope;

import com.hp.hpl.jena.query.Query;
import com.hp.hpl.jena.query.QueryExecution;
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.ResultSetFactory;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.reasoner.Reasoner;
import com.hp.hpl.jena.reasoner.ReasonerRegistry;

import wonts.solr.service.messages.SolrQueryMessage;
import wonts.solr.service.messages.responce.SolrQueryMsgResp;
```

```
import wonts.solr.service.properties.PropertiesManager;
import akka.actor.UntypedActor;
import akka.event.Logging;
import akka.event.LoggingAdapter;

@Named("solrQueryEngine")
@Scope("prototype")
public class SolrQueryEngine extends UntypedActor {

    final LoggingAdapter log =
Logging.getLogger(getContext().system(), this);

    @Getter
    SolrClient solr;

    public SolrQueryEngine() {
        String wonts =
PropertiesManager.getProperty(PropertiesManager.SOLR_CORE_ENDPOINT);
        solr = new HttpSolrClient(wonts);
    }

    @Override
    public void onReceive(Object message) throws Exception {
        if (message instanceof SolrQueryMessage) {
            // Execute query in solar "e.c.: content=pizza* AND
content=online"
            SolrQueryMsgResp resp =
executeQuery((SolrQueryMessage)message);
            getSender().tell(resp, getSelf());
        } else {
            unhandled(message);
        }
    }

    private SolrQueryMsgResp executeQuery(SolrQueryMessage message)
{
    SolrQueryMsgResp ret = new SolrQueryMsgResp();
    ret.setQuery(message.getQuery());

    SolrQuery parameters = new SolrQuery();
    parameters.set("q", message.getQuery());
}
```

```

parameters.set("qt", "/select");
parameters.set("rows", 10000);

try {
QueryResponse resp = solr.query(parameters);

if (resp.getResults() != null ){
    ret.setNumFound(resp.getResults().getNumFound());
    ret.setStart(resp.getResults().getStart());
    resp.getResults().forEach(p -> {
        String url = p.get("url").toString();
        ret.getResources().add(url);
    });
}
} catch (SolrServerException e) {
    log.error(e, "SolrServerException");
} catch (IOException e) {
    log.error(e, "IOException");
}
return ret;
}

public ResultSet execSelectQuery(String queryString, Model
model) {
ResultSet res = null;
Query query = QueryFactory.create(queryString) ;
Reasoner r = ReasonerRegistry.getRDFSReasoner();
r.bindSchema(model);
try (QueryExecution qexec =
QueryExecutionFactory.create(query, model)){
    res = qexec.execSelect();
    res = ResultSetFactory.copyResults(res) ;
    qexec.close();
}catch (Exception e) {
    res = null;
    log.error(e, "execSelectQuery");
}

return res;
}
}

```

8.4.1.10 *SolrServiceProvider.java*

```
package wonts.solr.service;

import java.io.IOException;

import org.apache.solr.client.solrj.SolrClient;
import org.apache.solr.client.solrj.impl.HttpSolrClient;
import org.apache.tika.exception.TikaException;
import org.xml.sax.SAXException;

public class SolrServiceProvider {
    private static SolrClient solr;
    //private static final Logger log =
    LogManager.getLogger(SolrServiceProvider.class);

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

        try {

            solr = new
HttpSolrClient("http://localhost:8983/solr/wonts");
            solr.deleteByQuery( "*:*" );

            /*String path = "d:/contents/";
            List<File> filesInFolder = Files.walk(Paths.get(path))
                .filter(Files::isRegularFile)
                .map(Path::toFile)
                .collect(Collectors.toList());*/

        }

        //filesInFolder.forEach(f -> processDocument(path +
f.getName() ));
        solr.commit();
    }
    catch (Exception ex) {
        System.out.println(ex.getMessage());
    }
}
/*
```

```
private static void processDocument(String pathfilename) {  
    InputStream stream=null;  
    AutoDetectParser parser = new AutoDetectParser();  
    BodyContentHandler handler = new  
    BodyContentHandler(10*1024*1024);  
    Metadata metadata = new Metadata();  
  
    try {  
        stream = new FileInputStream(new File(pathfilename));  
        parser.parse(stream, handler, metadata );  
        UUID guid = java.util.UUID.randomUUID();  
        String docid = guid.toString();  
        String doctitle = metadata.get(DublinCore.TITLE);  
        String doccreator = metadata.get(DublinCore.CREATOR);  
        String docurl = pathfilename;  
        String doccontent = handler.toString();  
  
        indexDocument(docid, doctitle, doccreator, docurl,  
        doccontent);  
    }  
    catch (Exception e) {  
        log.error(e);  
    } finally {  
        try {  
            stream.close();  
        } catch (IOException e) {  
            log.error(e);  
        }  
    }  
}  
  
private static void indexDocument(String docid, String doctitle,  
String doccreator, String docurl, String doccontent) {  
    try {  
        SolrInputDocument doc = new SolrInputDocument();  
  
        doc.addField("id", docid);  
        //doc.addField("title", doctitle);  
        //doc.addField("author", doccreator);  
        doc.addField("url", docurl);  
        doc.addField("content", doccontent);  
        solr.add(doc);  
    }  
}
```

```
        catch (Exception ex) {
            System.out.println(ex.getMessage());
        }
    } /*/
}

8.4.1.11      SolrUpdateResource.java
package wonts.solr.service;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.UUID;

import javax.inject.Named;

import lombok.Getter;

import org.apache.solr.client.solrj.SolrClient;
import org.apache.solr.client.solrj.impl.HttpSolrClient;
import org.apache.solr.client.solrj.response.UpdateResponse;
import org.apache.solr.common.SolrInputDocument;
import org.apache.tika.metadata.DublinCore;
import org.apache.tika.metadata.Metadata;
import org.apache.tika.parser.AutoDetectParser;
import org.apache.tika.sax.BodyContentHandler;
import org.springframework.context.annotation.Scope;

import wonts.solr.service.messages.ResourceIndexedMsg;
import wonts.solr.service.messages.SolrUpdateMsg;
import wonts.solr.service.messages.responce.SolrUpdateMsgResp;
import wonts.solr.service.properties.PropertiesManager;
import akka.actor.UntypedActor;
import akka.event.Logging;
import akka.event.LoggingAdapter;

@Named("solrUpdateResource")
@Scope("prototype")
public class SolrUpdateResource extends UntypedActor {
    //private static final Logger log =
    LogManager.getLogger(SolrUpdateResource .class);
    final LoggingAdapter log =
    Logging.getLogger(getContext().system(), this);
```

```
@Getter
SolrClient solr;

public SolrUpdateResource(){
    String wonts =
PropertiesManager.getProperty(PropertiesManager.SOLR_CORE_ENDPOINT);
    solr = new HttpSolrClient(wonts);
}

private void processDocument(SolrUpdateMsg message) {
    log.info("solr update message received for: " +
message.getFilePath());

    InputStream stream=null;
    AutoDetectParser parser = new AutoDetectParser();
    BodyContentHandler handler = new
BodyContentHandler(10*1024*1024);
    Metadata metadata = new Metadata();

    try {
        stream = new FileInputStream(new
File(message.getFilePath()));
        parser.parse(stream, handler, metadata );
        UUID guid = java.util.UUID.randomUUID();
        String docid = guid.toString();
        String doctitle = metadata.get(DublinCore.TITLE);
        String doccreator = metadata.get(DublinCore.CREATOR);
        String docurl = message.getUrl();
        String doccontent = handler.toString();

        indexDocument(docid, doctitle, doccreator, docurl,
doccontent);
    }
    catch (Exception e) {
        log.error("processDocument " , e);

    } finally {
        try {
            stream.close();
        }
    }
}
```

```

        } catch (IOException e) {
            log.error("processDocument - closing stream",e);
        }
    }

    private void indexDocument(String docid, String doctitle,
String doccreator, String docurl, String doccontent) {
    try {
        SolrInputDocument doc = new SolrInputDocument();

        doc.addField("id", docurl);
        //doc.addField("title", doctitle);
        //doc.addField("author", doccreator);
        doc.addField("url", docurl);
        doc.addField("content", doccontent);
        solr.add(doc);

    }
    catch (Exception ex) {
        System.out.println(ex.getMessage());
    }
}

@Override
public void onReceive(Object message) throws Exception {
    if ( message instanceof SolrUpdateMsg ){
        SolrUpdateMsg updateMessage = (SolrUpdateMsg)message;
        try{
            log.debug("received message {}", message);
            processDocument((SolrUpdateMsg)message);

            UpdateResponse responce = solr.commit();
            SolrUpdateMsgResp msgBack = new
SolrUpdateMsgResp(responce, (SolrUpdateMsg)message);

            getSender().tell(msgBack, getSelf());
        }catch(Exception e){
            log.error("solr update error", e);
            ResourceIndexedMsg msg = new ResourceIndexedMsg(
updateMessage.getUrlToCrawlId(),updateMessage.getUrl(),

```

```
ResourceIndexedMsg.ResourceStatus.updated);
        getSender().tell(msg, getSelf());
    }
} else if (message instanceof Exception) {
    throw (Exception) message;
}
}
```

8.4.1.12 SpringActorProducer.java

```
package wonts.solr.service;
```

```
import akka.actor.Actor;
import akka.actor.IndirectActorProducer;

import org.springframework.context.ApplicationContext;

public class SpringActorProducer implements IndirectActorProducer {
    final ApplicationContext applicationContext;
    final String actorBeanName;

    public SpringActorProducer(ApplicationContext
applicationContext,
                               String actorBeanName) {
        this.applicationContext = applicationContext;
        this.actorBeanName = actorBeanName;
    }

    @Override
    public Actor produce() {
        return (Actor) applicationContext.getBean(actorBeanName);
    }

    @SuppressWarnings("unchecked")
    @Override
    public Class<? extends Actor> actorClass() {
        return (Class<? extends Actor>)
applicationContext.getType(actorBeanName);
    }
}
```

8.4.1.13 *SpringExtension.java*

```
package wonts.solr.service;

import akka.actor.AbstractExtensionId;
import akka.actor.ExtendedActorSystem;
import akka.actor.Extension;
import akka.actor.Props;
import org.springframework.context.ApplicationContext;

public class SpringExtension extends
AbstractExtensionId<SpringExtension.SpringExt> {

    /**
     * The identifier used to access the SpringExtension.
     */
    public static SpringExtension SpringExtProvider = new
SpringExtension();

    /**
     * Is used by Akka to instantiate the Extension identified by this
     * ExtensionId, internal use only.
     */
    @Override
    public SpringExt createExtension(ExtendedActorSystem system) {
        return new SpringExt();
    }

    /**
     * The Extension implementation.
     */
    public static class SpringExt implements Extension {
        private volatile ApplicationContext applicationContext;

        /**
         * Used to initialize the Spring application context for the
         * extension.
         * @param applicationContext
         */
        public void initialize(ApplicationContext applicationContext) {
            this.applicationContext = applicationContext;
        }

        /**
         * Create a Props for the specified actorBeanName using the
         */
    }
}
```

```
* SpringActorProducer class.  
*  
* @param actorBeanName The name of the actor bean to create Props  
for  
* @return a Props that will create the named actor bean using  
Spring  
*/  
public Props props(String actorBeanName) {  
    return Props.create(SpringActorProducer.class,  
        applicationContext, actorBeanName);  
}  
}  
}
```

8.4.2 wonts.solr.service.messages

8.4.2.1 GetResoucesToIndexMsg.java

```
package wonts.solr.service.messages;  
  
import lombok.Data;  
  
@Data  
public class GetResoucesToIndexMsg {  
    int urlToCrawlId;  
    String urlCrawled;  
}
```

8.4.2.2 ProcessOntologyMsg.java

```
package wonts.solr.service.messages;  
  
import lombok.Data;  
  
@Data  
public class ProcessOntologyMsg {  
    String ontologiesbasePath;  
}
```

8.4.2.3 ReadOwlFileMsg.java

```
package wonts.solr.service.messages;  
  
import java.util.ArrayList;  
import java.util.List;
```

```
import lombok.Data;

@Data
public class ReadOwlFileMsg {
    public ReadOwlFileMsg(){}
    public ReadOwlFileMsg(int id, String owlFileName){
        this(id,owlFileName, null);
    }
    public ReadOwlFileMsg(int id, String owlFileName, List<String>
domains) {
        this.owlFileName = owlFileName;
        this.id = id;
        if ( domains == null)
            this.domains = new ArrayList<String>();
        else
            this.domains = domains;
    }
    String owlFileName;
    int id;
    List<String> domains;
}
```

8.4.2.4 ResourceIndexedMsg.java

```
package wonts.solr.service.messages;

import lombok.Data;

@Data
public class ResourceIndexedMsg {
    public enum ResourceStatus { downloaded, indexed, updated,
deleted }
    public ResourceIndexedMsg(){}
    public ResourceIndexedMsg(int urlToCrawlId, String urlResource,
ResourceStatus status ){
        this.urlToCrawlId = urlToCrawlId;
        this.urlResource = urlResource;
        this.status = status;
    }
    int urlToCrawlId;
    String urlResource;
```

```
    ResourceStatus status;  
}
```

8.4.2.5 SolrDeleteResourceMsg.java

```
package wonts.solr.service.messages;  
  
import lombok.Data;  
  
@Data  
public class SolrDeleteResourceMsg {  
    public SolrDeleteResourceMsg(String absolutePath) {  
        this(absolutePath, 0);  
    }  
  
    public SolrDeleteResourceMsg(String absolutePath,int urlToCrawlId ) {  
        this.path = absolutePath;  
        this.urlToCrawlId = urlToCrawlId;  
    }  
  
    String path;  
    int urlToCrawlId;  
}
```

8.4.2.6 SolrDownloadMsg.java

```
package wonts.solr.service.messages;  
  
import lombok.Data;  
  
@Data  
public class SolrDownloadMsg {  
    public SolrDownloadMsg(){}
    public SolrDownloadMsg(String url, int urlToCrawlId ){  
        this.url = url;  
        this.urlToCrawlId = urlToCrawlId;  
    }  
    String url;  
    int urlToCrawlId;  
}
```

8.4.2.7 SolrQueryMessage.java

```
package wonts.solr.service.messages;

import lombok.Data;

@Data
public class SolrQueryMessage {
    public SolrQueryMessage(){}
    public SolrQueryMessage(String query) {
        this.query = query;
    }

    String query;
}
```

8.4.2.8 SolrUpdateMsg.java

```
package wonts.solr.service.messages;

import java.io.Serializable;

import lombok.Data;

@Data
public class SolrUpdateMsg implements Serializable {

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    public SolrUpdateMsg( String url, String id, String filePath ){
        this(url, id, filePath,0);
    }

    public SolrUpdateMsg( String url, String id, String filePath,
int urlToCrawlId ){
        this.url=url;
        this.id=id;
        this.filePath=filePath;
        this.urlToCrawlId = urlToCrawlId;
    }
}
```

```
    private String url;
    private String id;
    private String filePath;
    private int urlToCrawlId;

}
```

8.4.3 wonts.solr.service.messages.responce

8.4.3.1 GetResourcesToIndexMsgRep.java

```
package wonts.solr.service.messages.responce;

import java.util.Set;

import lombok.Data;

@Data
public class GetResourcesToIndexMsgRep {
    Set<String> resources;
    int urlToCrawlId;
    public GetResourcesToIndexMsgRep(){}
    public GetResourcesToIndexMsgRep( Set<String> resources, int
urlToCrawlId ){
        this.resources = resources;
        this.urlToCrawlId = urlToCrawlId;
    }
}
```

8.4.3.2 ReadOwlFileMsgResp.java

```
package wonts.solr.service.messages.responce;

import wonts.solr.service.messages.ReadOwlFileMsg;

import com.hp.hpl.jena.ontology.OntModel;
import com.hp.hpl.jena.rdf.model.Model;

import lombok.Data;

@Data
public class ReadOwlFileMsgResp {
    public ReadOwlFileMsgResp(){}
}
```

```
public ReadOwlFileMsgResp(ReadOwlFileMsg msg, OntModel model) {
    this.owlFileName = msg.getOwlFileName();
    this.id = msg.getId();
    this.model = model;
}
String owlFileName;
int id;
Model model;
}
```

8.4.3.3 SolrDownloadMsgResp.java

```
package wonts.solr.service.messages.responce;

import java.nio.file.Path;

import lombok.Data;

@Data
public class SolrDownloadMsgResp {
    public SolrDownloadMsgResp(){}
    public SolrDownloadMsgResp(String url, Path pfile) {
        this(url, pfile, 0);
    }
    public SolrDownloadMsgResp(String url, Path pfile, int
urlToCrawlId) {
        this.url = url;
        this.path = pfile;
        this.urlToCrawlId = urlToCrawlId;
    }
    String url;
    Path path;
    int urlToCrawlId;
}
```

8.4.3.4 SolrQueryMsgResp.java

```
package wonts.solr.service.messages.responce;

import java.util.HashSet;
import java.util.Set;

import lombok.Data;
```

```
@Data
public class SolrQueryMsgResp {

    public SolrQueryMsgResp(){
        this.resources = new HashSet<String>();
    }

    String query;
    Set<String> resources;
    long numFound;
    long start;
}
```

8.4.3.5 SolrUpdateMsgResp.java

```
package wonts.solr.service.messages.responce;

import org.apache.solr.client.solrj.response.UpdateResponse;

import wonts.solr.service.messages.SolrUpdateMsg;
import lombok.Data;

@Data
public class SolrUpdateMsgResp {
    public SolrUpdateMsgResp(){}

    public SolrUpdateMsgResp(UpdateResponse updateResponse,
SolrUpdateMsg message) {
        this.updateResponse = updateResponse;
        this.id= message.getId();
        this.filePath= message.getFilePath();
        this.url = message.getUrl();
        this.urlToCrawlId = message.getUrlToCrawlId();
    }

    private UpdateResponse updateResponse;
    private String url;
    private String id;
    private String filePath;
    private int urlToCrawlId;
}
```

8.4.4 wonts.solr.service.properties

8.4.4.1 PropertiesManager.java

```
package wonts.solr.service.properties;

import java.io.FileInputStream;
import java.util.Properties;

import org.apache.log4j.LogManager;
import org.apache.log4j.Logger;

public class PropertiesManager {
    private static final Logger log =
LogManager.getLogger(PropertiesManager.class);

    public static final String SOLR_CORE_ENDPOINT =
"solr.core.endpoint";
    public static final String SOLR_RESOURCES_PATH_DOWNLOAD=
"solr.resources.pathdownload";
    public static final String WONTSAPP_ONTOLOGY_UPLOAD =
"wontsapp.ontology.uploaded";
    public static final String CRAWLWER_RESOURCES_SECONDS =
"crawled.resources.seconds";
    public static final String PROCESS_ONTOLOGY_SECONDS =
"process.ontology.seconds";
    public static final String PROPERTY_FILENAME=
"resources/solar.service.properties";

    static Properties properties;

    static {
        properties = GetProviderProperty(PROPERTY_FILENAME);
    }

    public static Properties GetProviderProperty(String fileName){
        Properties ret= new Properties();
        try {
            ret.load(new FileInputStream(fileName));
        } catch (Exception e) {
            log.error(e);
        }
    }
}
```

```
        }
        return ret;
    }

    public static synchronized String getProperty(String key){
        return properties.getProperty(key);
    }

}
```

8.4.4.2 *data_context_beans.xml*

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx.xsd">
```

```
    <bean id="emf"
          class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
        <property name="dataSource" ref="dataSource" />
        <property name="packagesToScan"
value="wonts.data.services.model" />
        <property name="jpaVendorAdapter">
            <bean
class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter"
/>
        </property>
        <property name="jpaProperties">
            <props>
                <prop key="hibernate.hbm2ddl.auto">update</prop>
                <prop
key="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</prop>
            </props>
        </property>
    </bean>
```

```

<bean id="dataSource"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="com.mysql.jdbc.Driver"
/>
    <property name="url"
value="jdbc:mysql://localhost:3306/wontsapp" />
    <property name="username" value="wontsuser" />
    <property name="password" value="user$wonts" />
</bean>

<bean id="transactionManager"
class="org.springframework.orm.jpa.JpaTransactionManager">
    <property name="entityManagerFactory" ref="emf" />
</bean>

<context:component-scan base-package="wonts.data.services" />
<tx:annotation-driven transaction-manager="transactionManager"/>

<!-- Root Context: defines shared resources visible to all other
web components -->
</beans>
```

8.4.4.3 log4j.properties

```

log4j.rootLogger=info, stdout, FILE
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern="%d %-5p [%t] %-17c{2}
(%13F:%L) %3x - %m%n

# Define the file appender
log4j.appender.FILE=org.apache.log4j.FileAppender
log4j.appender.FILE.File=logs/solar.services.out
log4j.appender.FILE.layout=org.apache.log4j.PatternLayout
log4j.appender.FILE.layout.ConversionPattern="%d %-5p [%t] %-17c{2}
(%13F:%L) %3x - %m%n
log4j.appender.FILE.MaxFileSize=1MB
log4j.appender.FILE.MaxBackupIndex=1
```

8.5 solar.services (src/test/java)

8.5.1 wonts.solr.service

8.5.1.1 JenaActorsTest.java

```
package wonts.solr.service;

import static org.junit.Assert.assertTrue;
import static wonts.solr.service.SpringExtension.SpringExtProvider;

import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.List;
import java.util.Set;
import java.util.UUID;
import java.util.concurrent.TimeUnit;

import org.junit.AfterClass;
import org.junit.BeforeClass;
import org.junit.Test;
import
org.springframework.context.annotation.AnnotationConfigApplicationContext
ext;

import scala.concurrent.Await;
import scala.concurrent.Future;
import scala.concurrent.duration.Duration;
import wonts.data.services.model.Url;
import wonts.data.services.model.UrlToCrawl;
import wonts.data.services.provider.UrlServiceProvider;
import wonts.solr.service.messages.GetResoucesToIndexMsg;
import wonts.solr.service.messages.ReadOwlFileMsg;
import wonts.solr.service.messages.responce.ReadOwlFileMsgResp;
import akka.actor.ActorRef;
import akka.actor.ActorSystem;
import akka.actor.Props;
import akka.testkit.JavaTestKit;
import akka.testkit.TestActorRef;
```

```
public class JenaActorsTest {

    static ActorSystem system;

    static UrlServiceProvider urlServiceProvider;

    @BeforeClass
    public static void setup() {
        ctx = new AnnotationConfigApplicationContext();
        ctx.scan("wonts");
        ctx.refresh();
        system = ctx.getBean(ActorSystem.class);
        urlServiceProvider =
ctx.getBean(UrlServiceProvider.class);
    }

    @AfterClass
    public static void teardown() {
        JavaTestKit.shutdownActorSystem(system);
        system = null;
    }

    static AnnotationConfigApplicationContext ctx;
    @Test
    public void getResourcesFromCrawledUrl() throws Exception{
        final Props props =
Props.create(GetResourcesToIndex.class);
        GetResoucesToIndexMsg msg = new GetResoucesToIndexMsg()
;
        msg.setUrlCrawled("http://www.dcs.bbk.ac.uk/~michael");
        final TestActorRef<GetResourcesToIndex> ref =
TestActorRef.create(system, props, "test Get Resources from jena " +
UUID.randomUUID().toString());
        final Future<Object> future =
akka.pattern.Patterns.ask(ref, msg, 1000 * 60 * 2 ); // I wait for 2
minutes
        assertTrue(future.isCompleted());
        @SuppressWarnings("unchecked")
        Set<String> urls = (Set<String>) Await.result(future,
Duration.Undefined());
        assertTrue( urls != null && urls.size() > 0 );
        System.out.println(urls.size());
    }
}
```

```
}

    private void AddTwoURLToIndex(UrlServiceProvider
urlServiceProvider){

    String urlValue = "http://www.dcs.bbk.ac.uk/~mark";
    String urlDescription = "description mark";

    String urlValue2 =
"http://www.dcs.bbk.ac.uk/~michael";
    String urlDescription2 = "description mark";
    Url url = new Url(urlValue, urlDescription);
    UrlToCrawl u = urlServiceProvider.saveUpdate( url );
    u.setActive(true);
    u.setStatus(UrlToCrawl.Status.isCrawled.ordinal());

    Url url2 = new Url(urlValue2, urlDescription2);
    UrlToCrawl u2 = urlServiceProvider.saveUpdate( url2
);
    u2.setActive(true);
    u2.setStatus(UrlToCrawl.Status.isCrawled.ordinal());

    urlServiceProvider.saveUpdate(u);
    urlServiceProvider.saveUpdate(u2);
}

@Test
public void getCrawledResource() throws Exception{

    AddTwoURLToIndex(urlServiceProvider);

    ActorRef ref = system.actorOf(
SpringExtProvider.get(system).props("getCrawledResources"),
"getCrawledResources");
    final Future<Object> future =
akka.pattern.Patterns.ask(ref, "getCrawledResources",
Duration.create(2, TimeUnit.MINUTES).toMillis()); // I wait for 2
minutes

    @SuppressWarnings("unchecked")
}
```

```
        Set<String> urls = (Set<String>) Await.result(future,
Duration.create(2, TimeUnit.MINUTES));
        assertTrue(urls != null && urls.size() > 0 );
        System.out.println(urls.size());
    }

    @Test
    public void ReadOwlFileTest() throws Exception{
        ActorRef ref = system.actorOf(
SpringExtProvider.get(system).props("readOwlFile"), "readOwlFile");

        ReadOwlFileMsg message = new ReadOwlFileMsg( 1,
"michael_m.owl" );
        final Future<Object> future =
akka.pattern.Patterns.ask(ref, message, Duration.create(5,
TimeUnit.MINUTES).toMillis()); // I wait for 5 minutes

        ReadOwlFileMsgResp response = (ReadOwlFileMsgResp)
Await.result(future, Duration.create( 5, TimeUnit.MINUTES));
        assertTrue(response.getModel() != null );

        PrintWriter out = new
PrintWriter("michael_m_filled.owl");
        response.getModel().write(out);

    }

    @Test
    public void ReadOwlFileDomainsTest() throws Exception{
        ActorRef ref = system.actorOf(
SpringExtProvider.get(system).props("readOwlFile"), "readOwlFile");

        List<String> domains = new ArrayList<>();
        domains.add("http://www.dcs.bbk.ac.uk/~michael");
        ReadOwlFileMsg message = new ReadOwlFileMsg( 1,
"michael_m.owl", domains );
        final Future<Object> future =
akka.pattern.Patterns.ask(ref, message, Duration.create(5,
TimeUnit.MINUTES).toMillis()); // I wait for 5 minutes
```

```
        ReadOwlFileMsgResp response = (ReadOwlFileMsgResp)
Await.result(future, Duration.create( 5, TimeUnit.MINUTES));
assertTrue(response.getModel() != null);

        PrintWriter out = new
PrintWriter("michael_m_filled_domains.owl");
response.getModel().write(out);

}
```

SolrActorsTest.java

```
package wonts.solr.service;

import static org.junit.Assert.assertTrue;
import static wonts.solr.service.SpringExtension.SpringExtProvider;

import java.nio.file.Files;
import java.nio.file.Path;
import java.util.Set;
import java.util.UUID;
import java.util.concurrent.TimeUnit;

import org.apache.jena.atlas.logging.Log;
import org.apache.solr.client.solrj.response.UpdateResponse;
import org.junit.AfterClass;
import org.junit.BeforeClass;
import org.junit.Test;
import
org.springframework.context.annotation.AnnotationConfigApplicationContext
ext;

import scala.concurrent.Await;
import scala.concurrent.Future;
import scala.concurrent.duration.Duration;
import wonts.data.services.model.Url;
import wonts.data.services.model.UrlToCrawl;
import wonts.data.services.provider.UrlServiceProvider;
import wonts.solr.service.messages.SolrDeleteResourceMsg;
```

```
import wonts.solr.service.messages.SolrDownloadMsg;
import wonts.solr.service.messages.SolrQueryMessage;
import wonts.solr.service.messages.SolrUpdateMsg;
import wonts.solr.service.messages.responce.SolrDownloadMsgResp;
import wonts.solr.service.messages.responce.SolrQueryMsgResp;
import wonts.solr.service.messages.responce.SolrUpdateMsgResp;
import akka.actor.ActorSystem;
import akka.actor.Props;
import akka.event.Logging;
import akka.event.LoggingAdapter;
import akka.testkit.JavaTestKit;
import akka.testkit.TestActorRef;

public class SolrActorsTest {

    // static LoggingAdapter log = new ;
    static ActorSystem system;

    @BeforeClass
    public static void setup() {
        ctx = new AnnotationConfigApplicationContext();
        ctx.scan("wonts");
        ctx.refresh();
        system = ctx.getBean(ActorSystem.class);
    }

    @AfterClass
    public static void teardown() {
        JavaTestKit.shutdownActorSystem(system);
        system = null;
    }

    static AnnotationConfigApplicationContext ctx;

    @Test
    public void testDownloadActor() throws Exception {
        final Props props =
Props.create(SolrDowloadResource.class);
        SolrDownloadMsg msg = new SolrDownloadMsg() ;
        msg.setUrl("http://www.dcs.bbk.ac.uk/courses/mscact/msc-
```

```

act-booklet.pdf");
    final TestActorRef<SolrDowloadResource> ref =
TestActorRef.create(system, props, "test download " +
UUID.randomUUID().toString());
    final Future<Object> future =
akka.pattern.Patterns.ask(ref, msg, 2000);
assertTrue(future.isCompleted());

        SolrDownloadMsgResp resp = (SolrDownloadMsgResp)
Await.result(future, Duration.Zero());
        assertTrue( Files.exists(resp.getPath()) );
        resp.getPath().toFile().delete();
}

@Test
public void testSendFileToSolarActor() throws Exception {
    final Props props =
Props.create(SolrDowloadResource.class);
    SolrDownloadMsg msg = new SolrDownloadMsg();
    String url ="http://www.dcs.bbk.ac.uk/courses/mscact/msc-
act-booklet.pdf";
    msg.setUrl(url);
    final TestActorRef<SolrDowloadResource> ref =
TestActorRef.create(system, props, "actor download " +
UUID.randomUUID().toString() );
    final Future<Object> future =
akka.pattern.Patterns.ask(ref, msg, 2000);
    assertTrue(future.isCompleted());
    SolrDownloadMsgResp res = (SolrDownloadMsgResp)
Await.result(future, Duration.Zero());
    assertTrue( Files.exists(res.getPath()) );

        // send resource to solr
        final Props sendingProps =
Props.create(SolrUpdateResource.class);
        SolrUpdateMsg sendingMessage = new SolrUpdateMsg(url,
res.getPath().toFile().getName(),
res.getPath().toFile().getAbsolutePath());
        final TestActorRef<SolrDowloadResource> sendingRef =
TestActorRef.create(system, sendingProps, "actor update " +
UUID.randomUUID().toString());
        final Future<Object> sendingFuture =

```

```
akka.pattern.Patterns.ask(sendingRef, sendingMessage, 5000);
    assertTrue(sendingFuture.isCompleted());
    SolrUpdateMsgResp resp = (SolrUpdateMsgResp)
Await.result(sendingFuture, Duration.Zero());
    assertTrue(resp.getUpdateResponse() != null);
    res.getPath().toFile().delete();
}

@Test
public void testSendFileDialogToSolr() throws Exception{
    // send resource to solr
    final Props sendingProps =
Props.create(SolrUpdateResource.class);
    String file =
"/Users/salvo/Development/wonts/resources_downloaded/p.docx";
    SolrUpdateMsg sendingMessage = new
SolrUpdateMsg("p.docx", "p.docx", file);
    final TestActorRef<SolrDowloadResource> sendingRef =
TestActorRef.create(system, sendingProps, "actor update " +
UUID.randomUUID().toString() );
    final Future<Object> sendingFuture =
akka.pattern.Patterns.ask(sendingRef, sendingMessage, 5000);
    assertTrue(sendingFuture.isCompleted());
    SolrUpdateMsgResp resp = (SolrUpdateMsgResp)
Await.result(sendingFuture, Duration.Zero());
    assertTrue(resp.getUpdateResponse() != null);

}

@Test
public void testSendFilepdfToSolr() throws Exception{
    // send resource to solr
    final Props sendingProps =
Props.create(SolrUpdateResource.class);
    String file =
"/Users/salvo/Development/wonts/resources_downloaded/p.pdf";
    SolrUpdateMsg sendingMessage = new
SolrUpdateMsg("p.pdf", "p.pdf", file);
    final TestActorRef<SolrDowloadResource> sendingRef =
TestActorRef.create(system, sendingProps, "actor update " +
UUID.randomUUID().toString() );
    final Future<Object> sendingFuture =
akka.pattern.Patterns.ask(sendingRef, sendingMessage, 5000);
    assertTrue(sendingFuture.isCompleted());
```

```

        SolrUpdateMsgResp resp = (SolrUpdateMsgResp)
Await.result(sendingFuture, Duration.Zero());
        assertTrue(resp.getUpdateResponse() != null);
    }

@Test
public void testRemoveResourceActor() throws Exception {

    final Props props =
Props.create(SolrDowloadResource.class);
    SolrDownloadMsg msg = new SolrDownloadMsg();
    String url ="http://www.dcs.bbk.ac.uk/courses/mscact/msc-
act-booklet.pdf";
    msg.setUrl(url);
    final TestActorRef<SolrDowloadResource> ref =
TestActorRef.create(system, props, "actor download " +
UUID.randomUUID().toString());
    final Future<Object> future =
akka.pattern.Patterns.ask(ref, msg, 2000);
    assertTrue(future.isCompleted());
    SolrDownloadMsgResp res = (SolrDownloadMsgResp)
Await.result(future, Duration.Zero());
    assertTrue(Files.exists(res.getPath()));

    // resource from disk
    final Props delProps =
Props.create(SolrDeleteResource.class);
    SolrDeleteResourceMsg removeMessage = new
SolrDeleteResourceMsg(res.getPath().toFile().getAbsolutePath());
    final TestActorRef<SolrDowloadResource> sendingRef =
TestActorRef.create(system, delProps, "actor delete resource " +
UUID.randomUUID().toString());
    final Future<Object> removeFuture =
akka.pattern.Patterns.ask(sendingRef, removeMessage, 5000);
    assertTrue(removeFuture.isCompleted());
    boolean resp = (boolean) Await.result(removeFuture,
Duration.Zero());
    assertTrue(resp);
    assertTrue(!Files.exists(res.getPath()));
}

@Test
public void testDownloadSendFileToSolarAndDeleteItActor()
throws Exception {

```

```

        final Props props =
Props.create(SolrDownloadResource.class);
        SolrDownloadMsg msg = new SolrDownloadMsg() ;
        String url ="http://www.dcs.bbk.ac.uk/courses/mscact/msc-
act-booklet.pdf";
        msg.setUrl(url);
        final TestActorRef<SolrDownloadResource> ref =
TestActorRef.create(system, props, "actor download " +
UUID.randomUUID().toString() );
        final Future<Object> future =
akka.pattern.Patterns.ask(ref, msg, 2000);
        assertTrue(future.isCompleted());
        SolrDownloadMsgResp res = (SolrDownloadMsgResp)
Await.result(future, Duration.Zero());
        assertTrue(Files.exists(res.getPath()) );

        // send resource to solr
        final Props sendingProps =
Props.create(SolrUpdateResource.class);
        SolrUpdateMsg sendingMessage = new SolrUpdateMsg(url,
res.getPath().toFile().getName(),
res.getPath().toFile().getAbsolutePath());
        final TestActorRef<SolrDownloadResource> sendingRef =
TestActorRef.create(system, sendingProps, "actor update " +
UUID.randomUUID().toString() );
        final Future<Object> sendingFuture =
akka.pattern.Patterns.ask(sendingRef, sendingMessage, 5000);
        assertTrue(sendingFuture.isCompleted());
        SolrUpdateMsgResp resp = (SolrUpdateMsgResp)
Await.result(sendingFuture, Duration.Zero());
        assertTrue(resp.getUpdateResponse()!=null);

        // resource from disk
        final Props delProps =
Props.create(SolrDeleteResource.class);
        SolrDeleteResourceMsg removeMessage = new
SolrDeleteResourceMsg(res.getPath().toFile().getAbsolutePath());
        final TestActorRef<SolrDownloadResource> delRef =
TestActorRef.create(system, delProps, "actor delete resource " +
UUID.randomUUID().toString() );
        final Future<Object> removeFuture =
akka.pattern.Patterns.ask(delRef, removeMessage, 5000);
        assertTrue(removeFuture.isCompleted());
        boolean respdel = (boolean) Await.result(removeFuture,

```

```
Duration.Zero());
        assertTrue(respdel);
        assertTrue( ! Files.exists(res.getPath()));
    }

/* @SuppressWarnings("unchecked")
@Test
public void testGetCrawledResources() throws Exception{

    UrlServiceProvider urlServiceProvider =
ctx.getBean(UrlServiceProvider.class);

    AddTwoURLToCrawl(urlServiceProvider);

    final Props props =
SpringExtProvider.get(system).props("getCrawledResources"); // +
Props.create(GetCrawledResources.class);

    final TestActorRef<SolrDowloadResource> ref =
TestActorRef.create(system, props, "actor getCrawledResources " +
UUID.randomUUID().toString() );
    final Future<Object> future =
akka.pattern.Patterns.ask(ref, "getCrawledResources",
Duration.create(1, TimeUnit.HOURS).toMillis());
    // assertTrue(future.isCompleted());
    Set<String> res = (Set<String>) Await.result(future,
Duration.create(1, TimeUnit.HOURS));
    assertTrue(res != null );
}

*/
@Test
public void testSolrQuery() throws Exception{
    final Props props =
SpringExtProvider.get(system).props("solrQueryEngine");
    final TestActorRef<SolrQueryEngine> ref =
TestActorRef.create(system, props, "solrQueryEngine");

    // TODO index Michael web content resources

    SolrQueryMessage message = new SolrQueryMessage();
    message.setQuery("content:web AND content:semantic");
}
```

```

        Duration d = Duration.create(2, TimeUnit.MINUTES);
        final Future<Object> future =
akka.pattern.Patterns.ask(ref, message, d.toMillis());
        assertTrue(future.isCompleted());
        SolrQueryMsgResp resp = (SolrQueryMsgResp)
Await.result(future, d );
        assertTrue(resp.getResources() != null &&
resp.getResources().size()>0);
        assertTrue(resp.getStart() == 0);
        assertTrue(resp.getNumFound() > 0 );

        System.out.println(resp.getResources().size());
    }

    private void AddTwoURLToCrawl(UrlServiceProvider
urlServiceProvider){

        String urlValue = "http://www.dcs.bbk.ac.uk/~mark";
        String urlDescription = "description mark";

        String urlValue2 =
"http://www.dcs.bbk.ac.uk/~michael";
        String urlDescription2 = "description mark";

        Url url = new Url(urlValue, urlDescription);
        UrlToCrawl u = urlServiceProvider.saveUpdate( url );
        u.setActive(true);
        u.setStatus(UrlToCrawl.Status.isCrawled.ordinal());

        Url url2 = new Url(urlValue2, urlDescription2);
        UrlToCrawl u2 = urlServiceProvider.saveUpdate( url2
);
        u2.setActive(true);
        u2.setStatus(UrlToCrawl.Status.isCrawled.ordinal());

        urlServiceProvider.saveUpdate(u);
        urlServiceProvider.saveUpdate(u2);
    }

}

```

8.5.1.2 SolrAppTest.java

```
package wonts.solr.service;

import wonts.solr.service.properties.PropertiesManager;
import junit.framework.Test;
import junit.framework.TestCase;
import junit.framework.TestSuite;

/**
 * Unit test for simple App.
 */
public class SolrAppTest
    extends TestCase
{
    /**
     * Create the test case
     *
     * @param testName name of the test case
     */
    public SolrAppTest( String testName )
    {
        super( testName );
    }

    /**
     *
     * @return the suite of tests being tested
     *
     */
    public static Test suite()
    {
        return new TestSuite( SolrAppTest.class );
    }

    /**
     * Tests for property manager.  START *****
     */
    public void testPropertyManager_SOLR_CORE_ENDPOINT()
    {
        String prop =
PropertiesManager.getProperty(PropertiesManager.SOLR_CORE_ENDPOINT);
        assertTrue( prop != null && prop.length()>0 );
    }
}
```

```
public void testPropertyManager_SOLR_RESOURCES_PATH_DOWNLOAD()
{
    String prop =
PropertiesManager.getProperty(PropertiesManager.SOLR_RESOURCES_PATH_
DOWNLOAD);
    assertTrue( prop != null && prop.length()>0 );
}
/***
 * Tests for property manager.  END ****
 */
```

}

8.5.1.3 data_context_beans.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx.xsd">
```



```
    <bean id="emf"
          class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
        <property name="dataSource" ref="dataSource" />
        <property name="packagesToScan"
value="wonts.data.services.model" />
        <property name="jpaVendorAdapter">
            <bean
class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter"
/>
        </property>
        <property name="jpaProperties">
            <props>
                <prop key="hibernate.hbm2ddl.auto">update</prop>
                <prop
```

```
key="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</prop>
    </props>
</property>
</bean>

<bean id="dataSource"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="com.mysql.jdbc.Driver" />
    <property name="url" value="jdbc:mysql://localhost:3306/wontsapp_test" />
    <property name="username" value="wontsuser" />
    <property name="password" value="user$wonts" />
</bean>

<bean id="transactionManager"
class="org.springframework.orm.jpa.JpaTransactionManager">
    <property name="entityManagerFactory" ref="emf" />
</bean>

<context:component-scan base-package="wonts.data.services" />
<tx:annotation-driven transaction-manager="transactionManager"/>

<!-- Root Context: defines shared resources visible to all other
web components -->
</beans>
```

8.5.2 resources

8.5.2.1 META-INF/beans.xml

```
<beans xmlns="http://xmlns.jcp.org/xml/ns/javaee"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                           http://xmlns.jcp.org/xml/ns/javaee/beans_1_2.xsd"
       bean-discovery-mode="all">
</beans>
```

8.5.3 application.conf

```
akka {
```

```
# loggers = ["akka.event.Logging$DefaultLogger", "akka.event.slf4j.Slf4jLogger"]  
loggers = ["akka.event.slf4j.Slf4jLogger"]  
loglevel = "DEBUG"  
logging-filter = "akka.event.slf4j.Slf4jLoggingFilter"  
  
}  
  
8.5.3.1 logback.xml  
<?xml version="1.0" encoding="UTF-8"?>  
<configuration>  
    <appender name="FILE"  
        class="ch.qos.logback.core.rolling.RollingFileAppender">  
        <encoder>  
            <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} -  
%msg%n</pattern>  
        </encoder>  
  
        <rollingPolicy  
        class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">  
            <fileNamePattern>logs\solar.service.%d{yyyy-MM-  
dd}.%i.log</fileNamePattern>  
            <timeBasedFileNamingAndTriggeringPolicy  
            class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">  
                <maxFileSize>5MB</maxFileSize>  
            </timeBasedFileNamingAndTriggeringPolicy>  
        </rollingPolicy>  
    </appender>  
  
    <root level="INFO">  
        <appender-ref ref="FILE" />  
    </root>  
</configuration>  
  
8.5.3.2 persistence.properties  
persistence.className=com.mysql.jdbc.Driver  
persistence.db=MySql  
persistence.db.url=jdbc:mysql://localhost/wonts  
persistence.db.urs=wontsuser  
persistence.db.pwd=user$wonts
```

8.5.3.3 *solar.service.properties*

```
solr.core.endpoint=http://localhost:8983/solr/wonts
solr.resources.pathdownload=/Users/salvo/Development/wonts/resources_d
ownloaded
wontsapp.ontology.uploaded=/Users/salvo/Development/wonts/ontology_upl
oaded
crawled.resources.seconds=300
process.ontology.seconds=5
```

8.6 solr.services (pom.xml)

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>wonts</groupId>
  <artifactId>solr.services</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>solr.services</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
  <build>
    <plugins>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.1</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
  <repositories>

    <repository>
      <id>org.ghost4j.repository.releases</id>
```

```
<name>Ghost4J releases</name>
<url>http://repo.ghost4j.org/maven2/releases</url>
</repository>
<repository>
    <id>org.ghost4j.repository.snapshots</id>
    <name>Ghost4J snapshots</name>
    <url>http://repo.ghost4j.org/maven2/snapshots</url>
</repository>
</repositories>

<dependencies>
    <dependency>
        <groupId>log4j</groupId>
        <artifactId>log4j</artifactId>
        <version>1.2.16</version>
    </dependency>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.7</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.apache.solr</groupId>
        <artifactId>solr-solrj</artifactId>
        <version>5.0.0</version>
    </dependency>
    <dependency>
        <groupId>org.apache.tika</groupId>
        <artifactId>tika-core</artifactId>
        <version>0.7</version>
    </dependency>
    <dependency>
        <groupId>org.apache.tika</groupId>
        <artifactId>tika-parsers</artifactId>
        <version>0.7</version>
    </dependency>
    <dependency>
        <groupId>commons-logging</groupId>
        <artifactId>commons-logging</artifactId>
        <version>1.1.2</version>
    </dependency>
    <!-- <dependency>
```

```
<groupId>org.slf4j</groupId>
<artifactId>slf4j-simple</artifactId>
<version>1.7.6</version>
</dependency>-->
<dependency>
    <groupId>org.ghost4j</groupId>
    <artifactId>ghost4j</artifactId>
    <version>0.5.2-SNAPSHOT</version>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.14.4</version>
</dependency>
<dependency>
    <groupId>commons-lang</groupId>
    <artifactId>commons-lang</artifactId>
    <version>2.6</version>
</dependency>
<dependency>
    <groupId>com.typesafe.akka</groupId>
    <artifactId>akka-actor_2.10</artifactId>
    <version>2.3.1</version>
</dependency>
<dependency>
    <groupId>com.typesafe.akka</groupId>
    <artifactId>akka-testkit_2.10</artifactId>
    <version>2.3.11</version>
</dependency>
<dependency>
    <groupId>com.typesafe.akka</groupId>
    <artifactId>akka-slf4j_2.11</artifactId>
    <version>2.3.12</version>
</dependency>

<!-- <dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>1.7.12</version>
</dependency> -->

<dependency>
    <groupId>ch.qos.logback</groupId>
    <artifactId>logback-classic</artifactId>
```

```
<version>1.1.3</version>
</dependency>

<dependency>
    <groupId>ch.qos.logback</groupId>
    <artifactId>logback-core</artifactId>
    <version>1.1.3</version>
</dependency>

<dependency>
    <groupId>javax.inject</groupId>
    <artifactId>javax.inject</artifactId>
    <version>1</version>
</dependency>
<!-- SolarJ -->
<dependency>
    <groupId>org.apache.solr</groupId>
    <artifactId>solr-solrj</artifactId>
    <version>5.2.1</version>
</dependency>
<dependency>
    <groupId>wonts</groupId>
    <artifactId>data.services</artifactId>
    <version>0.0.1-SNAPSHOT</version>
</dependency>
</dependencies>
</project>
```

8.7 triples.crawler (src/main/java)

8.7.1 wonts.triples.crawler

8.7.1.1 AppConfigration.java

```
package wonts.triples.crawler;

import static
wonts.triples.crawler.SpringExtension.SpringExtProvider;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
```

```
import org.springframework.context.annotation.ImportResource;

import akka.actor.ActorSystem;

@ImportResource("classpath:data_context_beans.xml")
@Configuration
public class AppConfigration {
    // the application context is needed to initialize the Akka
    Spring Extension
    @Autowired
    private ApplicationContext applicationContext;

    /**
     * Actor system singleton for this application.
     */
    @Bean
    public ActorSystem actorSystem() {
        ActorSystem system = ActorSystem.create("WontsCrawler");
        // initialize the application context in the Akka Spring
        Extension
        SpringExtProvider.get(system).initialize(applicationContext);
        return system;
    }
}
```

8.7.1.2 Crawler.java

```
package wonts.triples.crawler;

public interface Crawler {

    public abstract String crawl(String url);
}
```

8.7.1.3 HtmlLink.java

```
package wonts.triples.crawler;

import java.io.Serializable;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import lombok.Data;
/**
```

```
* This class is used to contains the URL link that has been parsed.
* @author salvo
*
*/
@Data
public class HtmlLink implements Serializable {

    private static final long serialVersionUID =
2834766438942320823L;

    private String url;
    private String parentUrl;
    private String domain;
    private String subDomain;
    private String path;
    private String href;
    private Boolean isSpecialLink;
    /**
     * This method is used to set the href of the links
     * @param href, hyper link
     */
    public void setHref(String href) {
        this.href = href;
        this.isSpecialLink = href.startsWith("mailto:")
                           || href.startsWith("javascript:");
    }

    /**
     * This method is used to set href and URL property of this class
     * @param link
     * @param startURL, this is the start URL.
     * @param urlToParse
     */
    public void createUrlFromLink (String link, String startURL,
String urlToParse ){
        this.setHref(link);
        if ( this.isSpecialLink ){ return;}
        String tempUrlToParse;
        if ( this.href.indexOf("../") >= 0 ){
            Pattern pattern = Pattern.compile("../");
            Matcher matcher = pattern.matcher(this.href);
            int endIndexLink = -1;
            tempUrlToParse = urlToParse;
            int step = 0;
```

```

        while (matcher.find()) {
            step++;
            endIndexLink = matcher.end();
        }
        tempUrlToParse = getParentUrl( tempUrlToParse, step);
        this.href = this.href.substring(endIndexLink-1);
        this.setUrl(tempUrlToParse +this.href );
    }else if ( this.href.startsWith("./") ){
        this.href = link.substring(1);
        tempUrlToParse = getParentUrl( urlToParse, 0);
        this.setUrl(tempUrlToParse +this.href );
    }else if (! link.startsWith("/") && link.indexOf("//") < 0
){
    this.href = "/" + link;
    this.setUrl(startURL + this.href );
}else if ( this.href.indexOf("//") < 0      ){
    this.setUrl(startURL +this.href );
}else {
    this.setUrl(this.href);
}
}

/**
 * This method is use to get the n parent from a string URL.
 * @param url,
 * @param step, is the number of parent to skip
 * @return string that represent the URL parent of the current
URL string.
 */
String getParentUrl( String url, int step ){
    for ( int i = step ; i>=0 ; i-- ){
        int endIdxUrl = url.lastIndexOf("/");
        if ( endIdxUrl >0 ){
            url = url.substring(0, endIdxUrl - i);
        }
    }
    return url;
}

@Override
public int hashCode() {
    return this.isSpecialLink ? href.hashCode():

```

```

url.hashCode();
}

@Override
public boolean equals(Object o) {
    if (this == o) {
        return true;
    }
    if (o == null || getClass() != o.getClass()) {
        return false;
    }

    HtmlLink otherUrl = (HtmlLink) o;
    return url != null && url.equals(otherUrl.getUrl());
}

/**
 * return the object to string. If it is a special link it
returns
 * its href property otherwise it returns its URL
 */
@Override
public String toString() {
    return this.isSpecialLink ? href: url;
}
/**
 * This method is used to set the URL.
 * @param url
 */
public void setUrl(String url) {
    this.url = url;
    this.syncDomain();
}

private void syncDomain(){
    subDomain = "";
    int domainStartIdx = url.indexOf("//") + 2;
    int domainEndIdx = url.indexOf('/', domainStartIdx);
    if (domainEndIdx <= domainStartIdx ){
        domain = url.substring(domainStartIdx);
    }else{
        domain = url.substring(domainStartIdx, domainEndIdx);
    }
}

```

```
    if ( domainEndIdx >= 0) {
        path = url.substring(domainEndIdx);
        int pathEndIdx = path.indexOf('?');
        if (pathEndIdx >= 0) {
            path = path.substring(0, pathEndIdx);
        }
    }
    syncSubDomain();
}

private void syncSubDomain(){
    String[] urlSegments = domain.split("\\\\.");
    if (urlSegments.length > 2) {
        domain = urlSegments[urlSegments.length - 2] + "." +
urlSegments[urlSegments.length - 1];
        int limit = 2;
        for (int i = 0; i < urlSegments.length - limit; i++)
{
            if (subDomain.length() > 0) {
                subDomain += ".";
            }
            subDomain += urlSegments[i];
        }
    }
}
}
```

8.7.1.4 JenaCrawler.java

```
package wonts.triples.crawler;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;
import java.util.List;
import java.util.UUID;

import javax.inject.Named;

import org.apache.log4j.LogManager;
import org.apache.log4j.Logger;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;

import wonts.triples.crawler.parser.HtmlParser;
import wonts.triples.crawler.robot.RobotTxt;
import wonts.triples.data.model.JenaResourceModel;
import wonts.triples.data.properties.PropertiesManager;

/**
 * Jena triples crawler
 *
 */
@Named("crawler")
@Scope("prototype")
public class JenaCrawler implements Crawler {

    String url;
    String name;
    List<String> listUrl;

    @Autowired
    JenaResourceModel resource;

    @Autowired
    HtmlParser parser;

    @Autowired
    RobotTxt robottxt;

    private static final Logger log =
LogManager.getLogger(HtmlParser.class);

    /* (non-Javadoc)
     * @see wonts.triples.crawler.Crawler#crawl(java.lang.String)
     */
    @Override
    public String crawl(String url) {
        String name = "";
        HtmlParser.startUrl= url;
        parser.setRobottxt(robottxt);
        parser.setResource(resource);
        parser.init();
        parser.parse(url);
```

```
    PrintWriter out1;

    //parser.getResource().getRdfModel().write(System.out);
    final String owlClawledPath =
PropertiesManager.getProperty(PropertiesManager.CRAWLER_OWL_PATH);
    name = UUID.randomUUID().toString() + ".owl";
    if (owlClawledPath != null && !
owlClawledPath.trim().isEmpty() ) {
        name = owlClawledPath + File.separator + name;
    }
    try {
        out1 = new PrintWriter(new File( name ), "UTF-8");
        parser.getResource().getRdfModel().write( out1 );
    } catch (FileNotFoundException e) {
        log.error(e);
    } catch (UnsupportedEncodingException e) {
        log.error(e);
    }
    return name;
}
}
```

8.7.1.5 ServicesStarter.java

```
package wonts.triples.crawler;

import static
wonts.triples.crawler.SpringExtension.SpringExtProvider;

import java.util.concurrent.TimeUnit;

import
org.springframework.context.annotation.AnnotationConfigApplicationContext;
ext;

import scala.concurrent.duration.Duration;
import wonts.triples.data.properties.PropertiesManager;
import akka.actor.ActorRef;
import akka.actor.ActorSystem;
import akka.actor.Cancellable;
import akka.actor.Props;
import akka.actor.UntypedActor;
import akka.event.Logging;
import akka.event.LoggingAdapter;
```

```
public class ServicesStarter {  
  
    private final AnnotationConfigApplicationContext ctx;  
    private final ActorSystem system;  
    private final Cancellable cancellable;  
  
    public ServicesStarter() throws Exception{  
  
        ctx = new AnnotationConfigApplicationContext();  
        ctx.scan("wonts");  
        ctx.refresh();  
  
        system = ctx.getBean(ActorSystem.class);  
  
        final Props propsCheckUrlToCrawl =  
SpringExtProvider.get(system).props("checkUrlToCrawl");  
  
        final ActorRef checkUrlToCrawl = system.actorOf(  
propsCheckUrlToCrawl, "checkUrlToCrawl");  
        int seconds= 30;  
        final String sec =  
PropertiesManager.getProperty(PropertiesManager.CRAWLER_CHECK_SEC);  
  
        if ( sec != null && !sec.isEmpty()){  
            seconds = Integer.parseInt(sec);  
        }  
        cancellable = system.scheduler().schedule(Duration.Zero(),  
                                         Duration.create(seconds, TimeUnit.SECONDS),  
                                         checkUrlToCrawl, "Check",  
                                         system.dispatcher(), null);  
    }  
  
    public static class Listener extends UntypedActor {  
        final LoggingAdapter log =  
Logging.getLogger(getContext().system(),  
                     this);  
  
        @Override  
        public void preStart() {  
            // If we don't get any progress within 15 seconds then  
            the service  
        }  
    }  
}
```

```
// is unavailable
// getContext().setReceiveTimeout(Duration.create("15
seconds"));
}

public void onReceive(Object msg) {
    log.debug("received message {}", msg);
    /*if (msg instanceof SolrUpdateMsg) {

        log.info("Current progress: {} %",
                ((SolrUpdateMsg) msg).getUrl());
    } else if (msg == ReceiveTimeout.getInstance()) {
        getContext().system().shutdown();
    }else {
        unhandled(msg);
    }*/
}
}

public static void main(String[] args) throws Exception{
    new ServicesStarter();
}

}
```

8.7.1.6 SpringActorProducer.java

```
package wonts.triples.crawler;

import akka.actor.Actor;
import akka.actor.IndirectActorProducer;
import org.springframework.context.ApplicationContext;

public class SpringActorProducer implements IndirectActorProducer {
    final ApplicationContext applicationContext;
    final String actorBeanName;

    public SpringActorProducer(ApplicationContext
applicationContext,
                                String actorBeanName) {
        this.applicationContext = applicationContext;
        this.actorBeanName = actorBeanName;
    }
}
```

```
}

@Override
public Actor produce() {
    return (Actor) applicationContext.getBean(actorBeanName);
}

@Override
public Class<? extends Actor> actorClass() {
    return (Class<? extends Actor>)
applicationContext.getType(actorBeanName);
}
}
```

8.7.1.7 SpringExtension.java

```
package wonts.triples.crawler;

import akka.actor.AbstractExtensionId;
import akka.actor.ExtendedActorSystem;
import akka.actor.Extension;
import akka.actor.Props;
import org.springframework.context.ApplicationContext;

public class SpringExtension extends
AbstractExtensionId<SpringExtension.SpringExt> {

/**
 * The identifier used to access the SpringExtension.
 */
public static SpringExtension SpringExtProvider = new
SpringExtension();

/**
 * Is used by Akka to instantiate the Extension identified by this
 * ExtensionId, internal use only.
 */
@Override
public SpringExt createExtension(ExtendedActorSystem system) {
    return new SpringExt();
}

/**
 * The Extension implementation.

```

```
/*
public static class SpringExt implements Extension {
    private volatile ApplicationContext applicationContext;

    /**
     * Used to initialize the Spring application context for the
     extension.
     * @param applicationContext
     */
    public void initialize(ApplicationContext applicationContext) {
        this.applicationContext = applicationContext;
    }

    /**
     * Create a Props for the specified actorBeanName using the
     * SpringActorProducer class.
     *
     * @param actorBeanName The name of the actor bean to create Props
     * for
     * @return a Props that will create the named actor bean using
     * Spring
     */
    public Props props(String actorBeanName) {
        return Props.create(SpringActorProducer.class,
                           applicationContext, actorBeanName);
    }
}
}
```

8.7.1.8 data_context_beans.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx.xsd">
```

```
<bean id="emf"
class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
    <property name="dataSource" ref="dataSource" />
    <property name="packagesToScan"
value="wonts.data.services.model" />
    <property name="jpaVendorAdapter">
        <bean
class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter"
/>
        </property>
    <property name="jpaProperties">
        <props>
            <prop key="hibernate.hbm2ddl.auto">create</prop>
            <prop
key="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</prop>
        </props>
    </property>
</bean>

    <bean id="dataSource"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName" value="com.mysql.jdbc.Driver"
/>
        <property name="url"
value="jdbc:mysql://localhost:3306/wontsapp_test" />
        <property name="username" value="wontsuser" />
        <property name="password" value="user$wonts" />
</bean>

    <bean id="transactionManager"
class="org.springframework.orm.jpa.JpaTransactionManager">
        <property name="entityManagerFactory" ref="emf" />
</bean>

<context:component-scan base-package="wonts.data.services" />
<tx:annotation-driven transaction-manager="transactionManager"/>

    <!-- Root Context: defines shared resources visible to all other
web components -->
</beans>
```

8.7.2 wonts.triples.crawler.parser

8.7.2.1 HtmlParser.java

```
package wonts.triples.crawler.parser;

// WRITTEN BY: Salvatore Rapisarda

import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.Reader;
import java.net.MalformedURLException;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.URLConnection;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashSet;
import java.util.LinkedList;
import java.util.List;
import java.util.Queue;
import java.util.Set;
import java.util.stream.Stream;

import javax.inject.Named;
import javax.swing.text.BadLocationException;
import javax.swing.text.EditorKit;
import javax.swing.text.SimpleAttributeSet;
import javax.swing.text.html.HTML;
import javax.swing.text.html.HTML.Tag;
import javax.swing.text.html.HTMLDocument;
import javax.swing.text.html.HTMLEditorKit;

import lombok.Getter;
import lombok.Setter;

import org.apache.log4j.LogManager;
import org.apache.log4j.Logger;
import org.springframework.beans.factory.BeanFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ApplicationContext;
```

```
import org.springframework.context.annotation.Scope;
import
org.springframework.context.support.ClassPathXmlApplicationContext;

import wonts.triples.crawler.HtmlLink;
import wonts.triples.crawler.robot.RobotTxt;
import wonts.triples.data.model.JenaResourceModel;

@Named("parser")
@Scope("prototype")
public class HtmlParser {

    private static final Logger log =
LogManager.getLogger(HtmlParser.class);

    @Autowired @Getter @Setter
    private RobotTxt robottxt;

    @Autowired @Getter @Setter
    JenaResourceModel resource;

    //static BeanFactory factory;

    @Getter @Setter
    List<String> pageParsed;

    @Autowired
    ApplicationContext factory;

    //@@Autowired
    //HtmlParser parser;

    static {
        //factory = new
ClassPathXmlApplicationContext("beans.xml");
    }
}
```

```
public static String startUrl;

public String checkURL (String url){
    if ( url.endsWith("/") ){
        url = url.substring(0, url.length() - 1 );
    }

    if ( ! url.contains("#") ){
        return url;
    }

    return url.replaceAll("[/#]+\w*$', ''");
}

public void init( ) {
    this.pageParsed = new ArrayList<String>();
    this.initLogFile();
    this.logVisitedLink(startUrl);
    //log.error("passato");
}

public HtmlParser(){
}

public HtmlParser( RobotTxt robotTxt){
    this();
    this.robottxt = robotTxt;
}

public void parse(String url){
    url = this.checkURL(url);
    pageParsed.add(url);
    List<String> listURL = this.parseHtmlLinks(url);
    //PageParsed.getInstance().add(url);
    listURL.forEach(s -> {
        s = this.checkURL(s);
        if ( s.startsWith(startUrl) && ! pageParsed.contains(s) ) {
            HtmlParser p = ( HtmlParser)
factory.getBean("parser");
            p.setResource(resource);
            p.setRobottxt(robottxt);
        }
    });
}
```

```

        p.setPageParsed(pageParsed);
        p.logVisitedLink(s);
        p.parse(s);
    }
}

/*
public void parse(String url){
    Queue<String> urls = new LinkedList<String>();
    urls.add(url);
    HashSet<String> pages = new HashSet<String>();
    HashSet<String> parsed = parseHelp(urls, pages );
    this.setPageParsed(Arrays.asList(parsed.toArray(new
String[0])));
}

private HashSet<String> parseHelp(Queue<String> urls,
HashSet<String> pageParsed2){
    if ( urls == null || urls.isEmpty() )
        return pageParsed2;
    else{
        String url = this.checkURL(urls.poll());
        if ( ! pageParsed2.contains(url) &&
url.startsWith(startUrl) ){
            pageParsed2.add(url);
            this.logVisitedLink(url);
            urls.addAll(this.parseHtmlLinks(url));
        }
        return parseHelp(urls, pageParsed2);
    }
}
*/
private void logVisitedLink( String link ){
    /*
    for ( int x =0 ; x<2 ; x++ ){
        this.logToFile("Visited: " + link + "", FILE_NAMES[x]);
    }
    */
    //log.info("Visited: " + link );
}

```

```

//  

/**  

 * It lists the links in the webpage.  

 * @param url  

 * @param links  

 * @return List of string  

 */  

public List<String> getListHtmlLinksToParse(String url,  

List<HtmlLink> links ) {  

    // this.logToFile("\t<No of links to Visited pages: " +  

links.size() + " >", FILE_RESULT);  

    ArrayList<String> ret = new ArrayList<String>();  

    List<String> linkOut = new ArrayList<>();  

    //for (int i = 0; i < this.getHtmlLinks().size(); i++) {  

        links.forEach( link -> {  

            //HtmlLink link = this.getHtmlLinks().get(i);  

            if ( link.getUrl() != null ) {  

                linkOut.add(link.getUrl());  

                if ( this.robottxt.isPathAllow( link.getPath() ) &&  

link.getUrl().contains(startUrl)){  

                    ret.add(link.getUrl());  

                }  

                // logToFile("\tLink: " + link.getUrl() + "",  

FILE_CRAWL );  

                //uncomment if necessary  

                // log.info ( "\tLink: " + link.getUrl() + "");  

            }  

        });  

        this.resource.AddUrlLinksVisited(url, linkOut);  

        return ret;  

    }  

private boolean isToParse(String urlToParse ){  

    boolean ret = true;  

    String [] exstensions = { ".pdf", ".docx", ".doc", ".ppt", ".ps"  

};  

    try {  

        for (String suffix : exstensions) {  

            if (urlToParse.endsWith(suffix) )  

                return false;

```

```

        }

    } catch (Exception e) {
        log.error(e);
        ret = false;
    }
    return ret;
}

private List<HtmlLink> getAllHtmlTags(HTMLDocument doc, String urlToParse, Tag tag ){
    List<HtmlLink> ret = new ArrayList<HtmlLink>();

    try {

        HTMLDocument.Iterator it = doc.getIterator(tag);

        while (it.isValid()) {
            try {
                String link = null;
                if (tag.equals(HTML.Tag.FRAME)){
                    link =
it.getAttributes().getAttribute(HTML.Attribute.SRC ).toString();
                }else if ( tag.equals(HTML.Tag.A) )  {
                    SimpleAttributeSet s=null;
                    s = (SimpleAttributeSet) it.getAttributes();
                    if ( s != null ){
                        link=(String)
s.getAttribute(HTML.Attribute.HREF);
                    }
                }
            }

            if ( link != null  ) {
                link = this.checkURL(link);
                HtmlLink href = new HtmlLink();
                href.createUrlFromLink(link, startUrl,
urlToParse );
                ret.add(href);
            }
        }
    }
}

```

```

        } catch (Exception e) {
            log.error(e);
        }
        it.next();
    }
} catch (Exception e) {
    log.error(e);
}

return ret;
}

// Parses HTML links in the webpage.
public List<String> parseHtmlLinks(String urlToParse) {
    List<String> ret= new ArrayList<>();
    List<HtmlLink> links = new ArrayList<HtmlLink>();
    try {
        URL url = new URI(urlToParse).toURL();
        URLConnection conn = url.openConnection();
        InputStream inputStream = conn.getInputStream();
        Reader rd = new InputStreamReader(inputStream);
        if (!isToParse(urlToParse)) {
            return ret;
        }
        EditorKit kit = new HTMLEditorKit();
        HTMLDocument doc = (HTMLDocument)
    kit.createDefaultDocument();
        doc.putProperty("IgnoreCharsetDirective", Boolean.TRUE);
        kit.read(rd, doc, 0);

        /*
        HTMLDocument.Iterator it = doc.getIterator(HTML.Tag.A);

        while (it.isValid()) {
            SimpleAttributeSet s=null;
            s = (SimpleAttributeSet) it.getAttributes();
            String link = null;
            if ( s != null ) {
                link=(String) s.getAttribute(HTML.Attribute.HREF);
            }
            if ( link != null ) {

```

```

        link = this.checkURL(link);
        HtmlLink href = new HtmlLink();
        href.createUrlFromLink(link, startUrl, urlToParse
);
        links.add(href);
    }
    it.next();
}

*/
links.addAll(get GetAllHtmlTags(doc, urlToParse,
HTML.Tag.FRAME) );
links.addAll(get GetAllHtmlTags(doc, urlToParse, HTML.Tag.A
) );
ret= this.getListHtmlLinksToParse(urlToParse,links );
}
catch(MalformedURLException e) {
log.error(e);
}
catch(URISyntaxException e) {
log.error(e);
}
catch(BadLocationException e) {
log.error(e);
}
catch(IOException e) {
log.error(e);
}
return ret;
}

private void initLogFiles(){
/*
for ( String filename : FILE_NAMES ){
File myFile = new File(filename);
if(myFile.exists())
myFile.delete();
}
*/
}

/*private void logToFile( String str, String filename ){
```

```
//log.info(str);

BufferedWriter writer;
try {
    writer = new BufferedWriter(new FileWriter(filename,
true));
    writer.append(str+"\n");
    writer.close();
    if ( filename.compareTo(FILE_CRAWL) == 0){
        log.info(str);
    }
} catch (FileNotFoundException e) {
    log.error(e);
} catch (UnsupportedEncodingException e) {
    log.error(e);
} catch (IOException e) {
    log.error(e);
}

}*/
```

}

8.7.2.2 PageParsed.java

```
package wonts.triples.crawler.parser;

import java.util.ArrayList;
import java.util.List;

public class PageParsed {

    private static PageParsed singleton;

    private List<String> parsed;

    private PageParsed(){
        parsed = new ArrayList<String>();
    }

    public void add (String link ){
```

```
        this.parsed.add(link);
    }

    public boolean contains ( String link ){
        return this.parsed.contains(link);
    }

    public static synchronized PageParsed getInstance(){
        if ( singleton == null ){
            singleton = new PageParsed();
        }
        return singleton;
    }
}
```

8.7.3 wonts.triples.crawler.robot

8.7.3.1 RobotList.java

```
package wonts.triples.crawler.robot;

import java.util.ArrayList;
/**
 * This Class extends a ArrayList of Strings.
 *
 * @author salvo
 */
public class RobotList extends ArrayList<String> {
    /**
     *
     */
    private static final long serialVersionUID = -
1646059267111191709L;

    @Override
    public boolean add(String resource){
        boolean retVal = super.add(resource);
        return retVal;
    }
}
```

8.7.3.2 RobotManager.java

```
package wonts.triples.crawler.robot;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.URLConnection;
import java.util.Scanner;

import org.apache.log4j.LogManager;
import org.apache.log4j.Logger;

public class RobotManager {
    private static final String FILE_ROBOTS_NAME= "/robots.txt";

    private static final String ROBOT_REGEX_USERAGENT_KEY =
"(?i)^User-agent:.*";
    private static final String ROBOT_REGEX_DISALLOW_KEY =
"(?i)Disallow:.*";
    private static final String ROBOT_REGEX_ALLOW_KEY =
"(?i)Allow:.*";

    private static final int ROBOT_LEN_USERAGENT_KEY = "User-
agent:".length();
    private static final int ROBOT_LEN_DISALLOW_KEY =
"Disallow:".length();
    private static final int ROBOT_LEN_ALLOW_KEY =
"Allow:".length();

    private static final Logger log =
LogManager.getLogger(RobotManager.class);
    /**
     * This method returns a RobotTxt through the urlToParse
     parameter.
     *
     *
     * @param urlToParse
     * @param agentToParse
     * @return
     */
}
```

```
public RobotTxt getRobotTxt( String urlToParse, String agentToParse ) {
    RobotTxt ret = new RobotTxt();
    ret.setAgentToParse(agentToParse);
    URL url;

    String pathRobots = urlToParse + FILE_ROBOTS_NAME;

    try {
        url = new URI(pathRobots).toURL();
        URLConnection conn = url.openConnection();
        try (Scanner sc = new Scanner(conn.getInputStream())) {
            Boolean isUserAgentFound = false;
            while (sc.hasNextLine()) {
                String line = sc.nextLine().trim();
                if (line.length() == 0 ){
                }else if (
line.matches(ROBOT_REGEX_USERAGENT_KEY) ){
                    String agent =
line.substring(ROBOT_LEN_USERAGENT_KEY).trim().toLowerCase();
                    isUserAgentFound =
agent.compareTo("*") == 0 || agent.compareTo(agentToParse) == 0;
                }else if ( isUserAgentFound &&
line.matches(ROBOT_REGEX_ALLOW_KEY) ){
                    String allow =
line.substring(ROBOT_LEN_ALLOW_KEY).trim();
                    URL u = new URL(urlToParse + allow);
                    ret.pushAllowItem( u.getPath() );
                }else if ( isUserAgentFound &&
line.matches(ROBOT_REGEX_DISALLOW_KEY) ){
                    String disallow =
line.substring(ROBOT_LEN_DISALLOW_KEY).trim();
                    try {
                        URL u = new URL(urlToParse +
disallow);
                        ret.pushDisallowItem( u.getPath()
);
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                }
            }
        }catch (FileNotFoundException e) {

```

```
        log.error("The file " + pathRobots + " doesn't
exist!!" );
    }
} catch (MalformedURLException e) {
    log.error(e);
} catch (URISyntaxException e) {
    log.error(e);
}catch ( IOException e) {
    log.error(e);
}

return ret;
}

}
```

8.7.3.3 RobotTxt.java

```
package wonts.triples.crawler.robot;

import javax.inject.Named;

import org.springframework.context.annotation.Scope;

import lombok.Getter;
import lombok.Setter;

@Named("robottxt")
@Scope("prototype")
public class RobotTxt {

    private RobotList disallowsItems=null;
    private RobotList allowsItems=null;

    @Getter @Setter
    private String agentToParse=null;
    /**
     * Constructor
     */
    public RobotTxt (){
```

```
        this.disallowsItems = new RobotList();
        this.allowsItems = new RobotList();
    }

    /**
     * Add a disallow item page.
     * @param item
     */
    public void pushDisallowItem( String item ){
        this.disallowsItems.add(item);
    }

    public boolean isPathAllow(String path) {

        if (path== null || path.contains("mailto:")){
            return false;
        }

        for (int i =0 ; i<disallowsItems.size(); i ++ ){
            if (path.contains( disallowsItems.get(i) )){
                return false;
            }
        }

        return !disallowsItems.contains(path) || allowsItems.contains(path);
    }

    /**
     * Add an allow item page.
     * @param item
     */
    public void pushAllowItem( String item ){
        this.allowsItems.add(item);
    }

}
```

8.7.4 wonts.triples.crawler.services

8.7.4.1 CheckUrlToCrawl.java

```
package wonts.triples.crawler.services;
```

```
import static
wonts.triples.crawler.SpringExtension.SpringExtProvider;  
  
import java.util.List;  
  
import javax.inject.Named;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.context.annotation.Scope;  
  
import scala.concurrent.duration.Duration;  
import wonts.data.services.model.UrlToCrawl;  
import wonts.data.services.provider.UrlServiceProvider;  
import wonts.triples.crawler.services.messages.CrawlUrlMessage;  
import akka.actor.ActorRef;  
import akka.actor.OneForOneStrategy;  
import akka.actor.SupervisorStrategy;  
import akka.actor.SupervisorStrategy.Directive;  
import akka.actor.UntypedActor;  
import akka.event.Logging;  
import akka.event.LoggingAdapter;  
import akka.japi.Function;  
  
@Named("checkUrlToCrawl")
@Scope("prototype")
public class CheckUrlToCrawl extends UntypedActor {  
  
    final LoggingAdapter log =
Logging.getLogger(getContext().system(),
    this);  
  
    @Autowired
    UrlServiceProvider urlServiceProvider;  
  
    @Override
    public SupervisorStrategy supervisorStrategy() {
        return new OneForOneStrategy(3, Duration.Inf(), new
Function<Throwable, SupervisorStrategy.Directive>() {
```

```

        @Override
        public Directive apply(final Throwable t) {
            // TODO add and log the exception
            // CrawlUrlMessage msg = (CrawlUrlMessage) t.
            getMessage();
            //
            urlServiceProvider.getUrlToCrawl(msg.getUrlToCrawlId());
            //    getSender().tell("error", getSelf());
            return SupervisorStrategy.stop();
        }
    } );

}

@Override
public void onReceive(Object message) throws Exception {
    if (message.equals("Check")) {
        log.info( "checking url to crawl.");
        // UrlServiceProvider urlServiceProvider =
        ctx.getBean(UrlServiceProvider.class);
        List<UrlToCrawl> urlCrawling =
        urlServiceProvider.getUrlThatAreCrawling();
        if ( urlCrawling == null || urlCrawling.size() <= 0
    ){
        List<UrlToCrawl> urlsToCrawl =
        urlServiceProvider.getAllActiveUrlToCrawl();
        if (urlsToCrawl != null && urlsToCrawl.size()
>0 ){
            UrlToCrawl p = urlsToCrawl.get(0);
            // urlsToCrawl.forEach(p ->{
            ActorRef crawlerRef =
            getContext().actorOf(
                SpringExtProvider.get(getContext().system()).props("crawlUrlProvider")
                , "crawlUrlProvider" + p.getUrlToCrawlId() );
                log.info("crawling : " +
                p.getUrlToCrawlId() );
                CrawlUrlMessage msg = new
                CrawlUrlMessage();
                msg.setUrlToCrawlId(p.getUrlToCrawlId());
                msg.setUrl(
                p.getUrl().getUrlValue());
                crawlerRef.tell(msg,

```

```
getSelf());  
  
        log.info("a message has been  
sent: " +msg );  
  
Patterns.ask(crawlerRef, msg);  
Duration.Inf);  
        // } );  
    }else{  
        log.info("There aren't any urls to  
crawl.");  
    }  
    }else {  
        UrlToCrawl u = urlCrawling.get(0);  
        log.info( "id: " + u.getUrlToCrawlId() + " url:  
" + u.getUrl().getUrlValue() + " is already crawling.");  
    }  
} else {  
    unhandled(message);  
}  
}  
}
```

8.7.4.2 CrawlUrlProvider.java

```
package wonts.triples.crawler.services;

import java.util.Date;

import javax.inject.Inject;
import javax.inject.Named;

import org.springframework.context.annotation.Scope;

import wonts.data.services.model.UrlToCrawl;
import wonts.data.services.model.UrlToCrawl.Status;
import wonts.data.services.provider.UrlServiceProvider;
import wonts.data.services.triples.TriplesDataProvider;
import wonts.triples.crawler.Crawler;
import wonts.triples.crawler.services.messages.CrawlUrlMessage;
import

wonts.triples.crawler.services.messages.response.CrawlUrlMessageResp;
import akka.actor.UntypedActor;
```

```
import akka.event.Logging;
import akka.event.LoggingAdapter;

@Named("crawlUrlProvider")
@Scope("prototype")
public class CrawlUrlProvider extends UntypedActor {
    final LoggingAdapter log =
Logging.getLogger(getContext().system(), this);

//    @Inject @Named("crawler")
//    Crawler crawler;

//public CrawlUrlProvider(){}

@Inject
TriplesDataProvider jenaDataProvider;

@Inject
public CrawlUrlProvider(@Named("crawler") Crawler crawler) {
    this.crawler = crawler;
}

@Inject
UrlServiceProvider urlServiceProvider;

@Override
public void onReceive(Object message) throws Exception {
    if ( message instanceof CrawlUrlMessage ){
        log.debug("received message {}", message);
        long timeElapsed=0;
        Date start= new Date();

        CrawlUrlMessage msg = ( CrawlUrlMessage ) message;
        updateStatusUrlToCrawl(msg.getUrlToCrawlId() ,
Status.isCrawling, timeElapsed, "" );

        String owl = processMessage(msg);
        Date stop = new Date();
        timeElapsed = stop.getTime() - start.getTime();
        updateStatusUrlToCrawl(msg.getUrlToCrawlId() ,
Status.isCrawled, timeElapsed, owl );
        jenaDataProvider.deleteLinkOutPages(msg.getUrl() );
        jenaDataProvider.PutRdfInDataBase(owl);
        getSender().tell( new CrawlUrlMessageResp(
```

```
msg.getUrl(), msg.getUrlToCrawlId(), owl), getSelf());  
        // kill itself after have sent the message back  
        context().stop(getSelf());  
    }else if (message instanceof Exception) {  
        throw (Exception) message;  
    }  
}  
  
private String processMessage(CrawlUrlMessage message) {  
    // crawler = message.getCrawler();  
    return crawler.crawl(message.getUrl());  
}  
  
  
  
private void updateStatusUrlToCrawl(int urlToCrawlId,  
UrlToCrawl.Status status, long timeElapsed, String owl ){  
    if (urlToCrawlId>0 )  
  
        urlServiceProvider.updateStatusUrlToCrawl(urlToCrawlId, status,  
timeElapsed, owl );  
    }  
}
```

8.7.5 wonts.triples.crawler.services.messages

8.7.5.1 CrawlUrlMessage.java

```
package wonts.triples.crawler.services.messages;  
  
import lombok.Data;  
  
@Data  
public class CrawlUrlMessage {  
    String url;  
    int urlToCrawlId;  
}
```

8.7.6 wonts.triples.crawler.services.messages.response

8.7.6.1 CrawlUrlMessageResp.java

```
package wonts.triples.crawler.services.messages.response;
```

```
import lombok.Data;

@Data
public class CrawlUrlMessageResp {
    CrawlUrlMessageResp(){}
    public CrawlUrlMessageResp(String url, int urlToCrawlId, String owl){
        this.url=url;
        this.urlToCrawlId = urlToCrawlId;
        this.owl = owl;
    }
    String url;
    int urlToCrawlId;
    String owl;
}
```

8.7.7 wonts.triples.data.model

8.7.7.1 JenaResourceModel.java

```
package wonts.triples.data.model;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import javax.inject.Named;

import lombok.Getter;
import lombok.Setter;

import org.apache.commons.io.FilenameUtils;
import org.apache.jena.iri.IRI;
import org.apache.jena.iri.IRIFactory;
import org.apache.log4j.LogManager;
import org.apache.log4j.Logger;
import org.springframework.context.annotation.Scope;

import com.hp.hpl.jena.ontology.OntClass;
import com.hp.hpl.jena.ontology.OntModel;
import com.hp.hpl.jena.ontology.OntModelSpec;
import com.hp.hpl.jena.ontology.OntProperty;
import com.hp.hpl.jena.rdf.model.ModelFactory;
```

```
import com.hp.hpl.jena.rdf.model.Resource;
import com.hp.hpl.jena.sparql.vocabulary.FOAF;
import com.hp.hpl.jena.vocabulary.DCTerms;
import com.hp.hpl.jena.vocabulary.OWL;
import com.hp.hpl.jena.vocabulary.OWL2;
import com.hp.hpl.jena.vocabulary.RDF;
import com.hp.hpl.jena.vocabulary.RDFS;
import com.hp.hpl.jena.vocabulary.XSD;

@Named("resource")
@Scope("prototype")
public class JenaResourceModel {
    private static final Logger log =
LogManager.getLogger(JenaResourceModel.class);

    @Getter
    OntModel rdfModel;
    OntProperty linksOut;
    OntProperty hasLinkIn;
    OntProperty hasNumberOfLinksOut;
    OntProperty hasNumberOfLinksIn;

    @Getter @Setter
    String NS;

    OntClass webResource;

    @Getter
    Map<String, OntClass> webResources;

    static IRIFactory factory = IRIFactory.jenaImplementation();

    private boolean isValidURI( String uri ) {
        IRI iri = factory.create( uri );
        return ! iri.hasViolation(false);

    }

    private Resource getWebResource( String url){
        // String key = null;
        //if ( url.matches("\\p{Graph}+\\.\\p{Alpha}{2,4}$") ){ //
url.replaceFirst("^.*?[^/]*\\.[^\\.]*$", "$1");
        
```

```

        String key = FilenameUtils.getExtension(url); //  

url.substring(url.lastIndexOf(".") + 1 );  

        if ( key != null && key != "" &&  

this.webResources.containsKey( key ) ){  

            return webResources.get(key);  

        }  

    }  

    return webResource;  

}  
  

/**  

 * Add to the model the links that have been visited.  

 * @param url  

 * @param linkOut  

 */  

public void AddUrlLinksVisited ( String url , List<String>  

linkOut ){  

    if ( isValidURI(url)){  

        Resource res = rdfModel.createResource(url,  

getWebResource(url) );  

        linkOut.stream().forEach( out ->  

    {  

        if ( isValidURI(out)){  

            Resource resOut =  

rdfModel.createResource(out, getWebResource(out) );  

            res.addProperty(linksOut, resOut);  

        }  

    });  

        res.addProperty(hasNumOfLinksOut,  

Integer.toString(linkOut.size()) ) ;  

    }
}  

/**  

 * Constructor  

 */  

public JenaResourceModel(){  
  

    this ( ModelFactory.createOntologyModel(  

OntModelSpec.OWL_MEM_MICRO_RULE_INF));  

}  
  

private void initModel(OntModel model){  

    this.NS = "http://titan.dcs.bbk.ac.uk/~srapis01#";
}

```

```

// TODO: it can be created by loading an OWL file.

// Ontology

webResource = model.createClass(NS + "WebResource");
webResource.setSuperClass(FOAF.Document);
webResource.addSuperClass(DCTerms.MediaType);
// linksOut property
linksOut = model.createOntProperty(NS + "linksOut");
linksOut.setDomain(webResource);
linksOut.setRange(webResource);
model.add(linksOut, RDF.type, OWL2.IrreflexiveProperty);

// linksIn property
hasLinkIn = model.createOntProperty(NS + "hasLinkIn");
hasLinkIn.setDomain(webResource);
hasLinkIn.setRange(webResource);
model.add(hasLinkIn, RDF.type,
OWL2.IrreflexiveProperty);

//linksIn property is the inverse of linksOut,
model.add(linksOut, OWL2.inverseOf, hasLinkIn );
model.add(hasLinkIn, OWL2.inverseOf, linksOut );



// has numbers links in
hasNumOfLinksIn = model.createOntProperty(NS +
"hasNumberOfLinksIn");
hasNumOfLinksIn.setDomain(webResource);
hasNumOfLinksIn.setRange(XSD.integer);
// has numbers links out
hasNumOfLinksOut = model.createOntProperty(NS +
"hasNumberOfLinksOut");
hasNumOfLinksOut.setDomain(webResource);
hasNumOfLinksOut.setRange(XSD.integer);

}

private void initWebResources(OntModel model){
    this.webResources = new HashMap<String, OntClass>();

    String [] resources = null;

    try {

```

```

        // it maps an add to the model the web resources as
        html, pdf, ps, doc, docx and so on
        String res = new
String(Files.readAllBytes(Paths.get("resources/web-resources"))); // 
add to property file
        if ( res != null ){
            resources = res.split(",");
        }
        } catch (Exception e) {
            log.error(e);
        }
        if ( resources != null ){
            for ( String r : resources) {
                OntClass c = model.createClass(NS + r.trim());
                c.setSuperClass(webResource);
                webResources.put( r.trim(), c );
            }
        }
        MakeDisjointResources();
    }

public void MakeDisjointResources(){
    this.webResources.forEach( (k1, r1)-> {
        this.webResources.forEach( (k2, r2)-> {
            if(!r2.equals(r1))
                r1.addDisjointWith(r2);
        });
    });
}

< /**
 * Constructor that injects the model.
 * @param model
 */
public JenaResourceModel(OntModel model){
    this.rdfModel = model;
    this.initModel(model);
    this.initWebResources(model);
}

}

```

8.7.8 wonts.triples.data.persistence

8.7.8.1 DataProvider.java

```
package wonts.triples.data.persistence;

import java.io.PrintStream;

import lombok.Data;

import org.apache.log4j.LogManager;
import org.apache.log4j.Logger;

import wonts.triples.data.properties.PropertiesManager;

import com.hp.hpl.jena.query.Dataset;
import com.hp.hpl.jena.query.Query;
import com.hp.hpl.jena.query.QueryExecution;
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.ResultSetFactory;
import com.hp.hpl.jena.query.ResultSetFormatter;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.reasoner.Reasoner;
import com.hp.hpl.jena.reasoner.ReasonerRegistry;
import com.hp.hpl.jena.sdb.SDBFactory;
import com.hp.hpl.jena.sdb.Store;
import com.hp.hpl.jena.sdb.StoreDesc;
import com.hp.hpl.jena.sdb.sql.JDBC;
import com.hp.hpl.jena.sdb.sql.SDBConnection;
import com.hp.hpl.jena.sdb.store.DatabaseType;
import com.hp.hpl.jena.sdb.store.DatasetStore;
import com.hp.hpl.jena.sdb.store.LayoutType;
import com.hp.hpl.jena.update.UpdateAction;
import com.hp.hpl.jena.util.FileManager;

@Data
public class DataProvider {
    private static final Logger log =
LogManager.getLogger(DataProvider.class);

    private SDBConnection getSDBConnection(){
```

```
JDBC.loadDriverMySQL();

return new SDBConnection(
PropertiesManager.getProperty(PropertiesManager.DB_URL),
PropertiesManager.getProperty(PropertiesManager.DB_USER),
PropertiesManager.getProperty(PropertiesManager.DB_PWD)
);
}

private Store getStore(SDBConnection conn ){
    StoreDesc storeDesc = new
StoreDesc(LayoutType.LayoutTripleNodesHash,
    DatabaseType.MySQL) ;
    return SDBFactory.connectStore(conn, storeDesc) ;

}

private Dataset getDataset(SDBConnection conn){
    Store store = getStore(conn);
    return DatasetStore.create(store) ;
}

public boolean PutRdfInDataBase(String filename){
    boolean ret= false;
    try{
        SDBConnection conn = getSDBConnection();
        try {
            Dataset ds = getDataset(conn);
            Model model = ds.getDefaultModel();
            model.begin();
            FileManager.get().readModel(model, filename);
            model.commit();
            model.close();
            ds.close();
            ret = true;
        }catch ( Exception ex ){
            log.error(ex);
            ret=false;
        }
        finally {
            conn.close();
        }
    }
}
```

```

        }
    }catch(Exception ex){
        log.error(ex);
        ret = false;
    }
    return ret;
}

public ResultSet execSelectQuery(String queryString) {
    ResultSet res = null;
    Query query = QueryFactory.create(queryString) ;
    SDBConnection conn = getSDBConnection();
    Dataset ds = getDataset(conn);
    Model model = ds.getDefaultModel();
    Reasoner r = ReasonerRegistry.getRDFSReasoner();
    r.bindSchema(model);
    try (QueryExecution qexec =
QueryExecutionFactory.create(query, model)){
        res = qexec.execSelect();
        res = ResultSetFactory.copyResults(res) ;
        qexec.close();
    }catch (Exception e) {
        res = null;
        log.error(e);
    }finally{
        ds.close();
        model.close();
        conn.close();
    }
    return res;
}

/**
 * This method is deleting all the URL pages that match all or
part of the urlString .
 * @param urlString
 * @return
 */
public boolean deleteLinkOutPages(String urlString ){
    if ( urlString == null ){
        return false;
    }
}

```

```

        String action = "DELETE { ?s ?p ?o } "
            + " WHERE { ?s ?p ?o ."
            + " FILTER regex(str(?s), \\" + urlString + "\", "
        +"i\") "
            + " }";
    return this.updateQuery(action);
}

/**
 * This method is used to update the model.
 * @param action, is the query action. For Example: String
action = "DELETE WHERE { ?s ?p ?o . } it is going to clean the model.
 * @return a boolean value, that is true when the action has
been done correctly, otherwise it returns false.
 */
private boolean updateQuery(String action) {
    boolean res = false;
    SDBConnection conn = getSDBConnection();
    Dataset ds = getDataset(conn);
    Model model = ds.getDefaultModel();
    try {
        model.begin();
        UpdateAction.parseExecute(action, model);
        model.commit();
        res = true;
    }catch (Exception e) {
        res = false;
        log.error(e);
    }finally{
        ds.close();
        model.close();
        conn.close();
    }
    return res;
}

/**
 *
 * @param queryString
 * @param notation Predefined values are "RDF/XML",
 * "RDF/XML-ABBREV", "N-TRIPLE", "TURTLE", (and "TTL") and "N3".
The default value,
 * represented by <code>null</code>, is "RDF/XML".</p>

```

```

* @return
*/
public String execSelectQuery(String queryString, String
notation ) {
    String retval="";
    //JenaResourceModel jrm = new JenaResourceModel();
    //OntModel m = jrm.getRdfModel();
    Query q = QueryFactory.create(queryString);
    SDBConnection conn = getSDBConnection();
    Dataset ds = getDataset(conn);
    Model model = ds.getDefaultModel();
    Reasoner r = ReasonerRegistry.getRDFSReasoner();
    r.bindSchema(model);
    try (QueryExecution qe = QueryExecutionFactory.create(q,
model)){

        PrintStream baos = System.out;
        int queryType = q.getQueryType();
        switch (queryType) {
        case Query.QueryTypeAsk:
            boolean b = qe.execAsk();
            ResultSetFormatter.outputAsRDF(baos, notation, b);
            retval = baos.toString();
            break;
        case Query.QueryTypeConstruct:
            model = qe.execConstruct();
            model.write(baos, notation);
            retval = baos.toString();
            break;
        case Query.QueryTypeDescribe:
            model = qe.execDescribe();
            model.write(baos, notation);
            retval = baos.toString();
            break;
        case Query.QueryTypeSelect:
            ResultSet results = qe.execSelect();
            ResultSetFormatter.out(baos, results);
            retval = baos.toString();
            break;
        }

    }catch (Exception e) {
        log.error(e);
    }finally{

```

```
        ds.close();
        model.close();
        conn.close();
    }
    return retval;
}

}
```

8.7.9 wonts.triples.data.properties

8.7.9.1 PropertiesManager.java

```
package wonts.triples.data.properties;

import java.io.FileInputStream;
import java.util.Properties;

import org.apache.log4j.LogManager;
import org.apache.log4j.Logger;

public class PropertiesManager {
    private static final Logger log =
LogManager.getLogger(PropertiesManager.class);

    public static final String DB = "persistence.db";
    public static final String DB_URL = "persistence.db.url";
    public static final String DB_USER = "persistence.db.urs";
    public static final String DB_PWD="persistence.db.pwd";
    public static final String CLASS_NAME=
"persistence.className";

    public static final String CRAWLER_OWL_PATH =
"crawler.owl.path";
    public static final String CRAWLER_CHECK_SEC =
"crawler.check.sec";

    public static final String PROPERTY_FILENAME=
"resources/persistence.properties";
```

```
static Properties properties;

/*
public PropertiesManager(){
    this(PROPERTY_FILENAME);
}
*/

static {
    properties = GetProviderProperty(PROPERTY_FILENAME);
}

public static Properties GetProviderProperty(String fileName){
    Properties ret= new Properties();
    try {
        ret.load(new FileInputStream(fileName));
    } catch (Exception e) {
        log.error(e);
    }
    return ret;
}

public static String getProperty(String key){
    return properties.getProperty(key);
}

}
```

8.7.10 beans.xml

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:util="http://www.springframework.org/schema/util"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans-
2.5.xsd
                           http://www.springframework.org/schema/aop
                           http://www.springframework.org/schema/aop/spring-aop-
2.5.xsd
                           http://www.springframework.org/schema/context
                           http://www.springframework.org/schema/context/spring-
context-2.5.xsd">
```

```
<!-- <context:component-scan base-
package="wonts.triples.crawler.parser" /> -->

    <bean id="parser" class="wonts.triples.crawler.parser.HtmlParser"
/>
    <bean id="resource"
class="wonts.triples.data.model.JenaResourceModel" />
    <bean id="crawler" class="wonts.triples.crawler.JenaCrawler"
scope="prototype" />
        <bean id="robottxt" class="wonts.triples.crawler.robot.RobotTxt"
scope="prototype" />
            <bean id="crawlUrlProvider"
class="wonts.triples.crawler.services.CrawlUrlProvider"
scope="prototype" />

</beans>
```

8.7.11 data_context_beans.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx.xsd">
```

```
    <bean id="emf"
class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
        <property name="dataSource" ref="dataSource" />
        <property name="packagesToScan"
value="wonts.data.services.model" />
        <property name="jpaVendorAdapter">
            <bean
class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter"
/>
            </property>
        <property name="jpaProperties">
```

```

<props>
    <prop key="hibernate.hbm2ddl.auto">update</prop>
    <prop
key="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</prop>
    </props>
</property>
</bean>

<bean id="dataSource"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="com.mysql.jdbc.Driver"
/>
    <property name="url"
value="jdbc:mysql://localhost:3306/wontsapp" />
    <property name="username" value="wontsuser" />
    <property name="password" value="user$wonts" />
</bean>

<bean id="transactionManager"
class="org.springframework.orm.jpa.JpaTransactionManager">
    <property name="entityManagerFactory" ref="emf" />
</bean>

<context:component-scan base-package="wonts.data.services" />
<tx:annotation-driven transaction-manager="transactionManager"/>

<!-- Root Context: defines shared resources visible to all other
web components -->
</beans>
```

8.7.12 log4j.properties

```

log4j.rootLogger=info, stdout, FILE
log4j.level=error
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern="%d %-5p [%t] %-17c{2}
(%13F:%L) %3x - %m%n

# Define the file appender
```

```
log4j.appender.FILE=org.apache.log4j.FileAppender
log4j.appender.FILE.File=logs/triples.crawler.out
log4j.appender.FILE.layout=org.apache.log4j.PatternLayout
log4j.appender.FILE.layout.ConversionPattern="%d %5p [%t] %-17c{2}
(%13F:%L) %3x - %m%n
log4j.appender.FILE.MaxFileSize=1MB
log4j.appender.FILE.MaxBackupIndex=1
```

8.7.13 logback.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <appender name="FILE"
        class="ch.qos.logback.core.rolling.RollingFileAppender">
        <encoder>
            <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} -
%msg%n</pattern>
        </encoder>

        <rollingPolicy
class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <fileNamePattern>logs\triples.crawler.%d{yyyy-MM-
dd}.%i.log</fileNamePattern>
            <timeBasedFileNamingAndTriggeringPolicy
class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">
                <maxFileSize>5MB</maxFileSize>
            </timeBasedFileNamingAndTriggeringPolicy>
        </rollingPolicy>
    </appender>

    <root level="DEBUG">
        <appender-ref ref="FILE" />
    </root>
</configuration>
```

8.8 triples.crawler (src/main/resources)

8.8.1 application.conf

```
akka {
```

```
# loggers = ["akka.event.Logging$DefaultLogger", "akka.event.slf4j.Slf4jLogger"]

loggers = ["akka.event.slf4j.Slf4jLogger"]
```

```
loglevel = "DEBUG"

logging-filter = "akka.event.slf4j.Slf4jLoggingFilter"

}

8.8.2 logback.xml
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <appender name="FILE"
        class="ch.qos.logback.core.rolling.RollingFileAppender">
        <encoder>
            <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} -
%msg%n</pattern>
        </encoder>

        <rollingPolicy
class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <fileNamePattern>logs\triples.crawler.%d{yyyy-MM-
dd}.%i.log</fileNamePattern>
            <timeBasedFileNamingAndTriggeringPolicy
class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">
                <maxFileSize>5MB</maxFileSize>
            </timeBasedFileNamingAndTriggeringPolicy>
        </rollingPolicy>
    </appender>

    <root level="DEBUG">
        <appender-ref ref="FILE" />
    </root>
</configuration>
```

8.9 triples.crawler(src/test/java)

8.9.1 wonts.triples.crawler

8.9.2 ActorContextTest.java

```
package wonts.triples.crawler;
```

```
import static org.junit.Assert.assertTrue;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
```

```
import java.io.UnsupportedEncodingException;
import java.util.Map;

import lombok.Getter;

import org.apache.jena.atlas.lib.PropertiesSorted;
import org.hibernate.tuple.PropertyFactory;
import org.junit.AfterClass;
import org.junit.BeforeClass;
import org.junit.Test;
import
org.springframework.context.annotation.AnnotationConfigApplicationContext;

import com.hp.hpl.jena.ontology.OntClass;
import com.hp.hpl.jena.ontology.OntModel;
import com.hp.hpl.jena.ontology.OntModelSpec;
import com.hp.hpl.jena.ontology.OntProperty;
import com.hp.hpl.jena.rdf.model.Literal;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.rdf.model.Property;
import com.hp.hpl.jena.rdf.model.Resource;
import com.hp.hpl.jena.sparql.vocabulary.FOAF;
import com.hp.hpl.jena.vocabulary.DCTerms;
import com.hp.hpl.jena.vocabulary.OWL2;
import com.hp.hpl.jena.vocabulary.RDF;

import akka.actor.ActorSystem;
import akka.testkit.JavaTestKit;

public class ActorContextTest {

    static ActorSystem system;

    @BeforeClass
    public static void setup() {
        ctx = new AnnotationConfigApplicationContext();
        ctx.scan("wonts");
        ctx.refresh();
        system = ctx.getBean(ActorSystem.class);
    }

    @AfterClass
```

```
public static void teardown() {
    JavaTestKit.shutdownActorSystem(system);
    system = null;
}

static AnnotationConfigApplicationContext ctx;

/*@Test
public void getActorSystem(){
    AnnotationConfigApplicationContext ctx1 =
        new AnnotationConfigApplicationContext();
    ctx1.scan("wonts.crawler");
    ctx1.refresh();

    // get hold of the actor system
    system = ctx1.getBean(ActorSystem.class);
    assertTrue(system!=null);
    ctx1.close();
}
*/
@Test
public void makeOntologyTest(){
    String NS = "http://titan.dcs.bbk.ac.uk/~srapis01#";

    OntClass Michael;

    OntModel model = ModelFactory.createOntologyModel(
OntModelSpec.OWL_MEM_MICRO_RULE_INF);

    OntProperty hasSearchQuery;

    // TODO: it can be created by loading an OWL file.

    // Ontology
//model =

    Michael = model.createClass(NS + "Michael");

    // linksOut property
    hasSearchQuery = model.createOntProperty(NS +
"hasSearchQuery");
```

```
        model.add(hasSearchQuery, RDF.type,
OWL2.IrreflexiveProperty);

        model.add(Michael, hasSearchQuery, "content:michael
AND content:semantic");

        try {
            PrintWriter out1 = new PrintWriter(new File(
"michael_m.owl" ), "UTF-8");
            model.write( out1 );
        } catch (FileNotFoundException e) {
            System.out.println(e);
        } catch (UnsupportedEncodingException e) {
            System.out.println(e);
        }

    }

}
```

8.9.2.1 TestCrawler

```
package wonts.triples.crawler;

import static org.junit.Assert.*;
import org.junit.Before;
import org.junit.Test;
import org.springframework.beans.factory.annotation.Autowired;

import wonts.triples.crawler.base.ContextConfigurationTest;
import wonts.triples.crawler.parser.HtmlParser;

public class TestCrawler extends ContextConfigurationTest {

    @Before
    public void setUp() throws Exception {
    }

    @Autowired
    Crawler crawler;
```

```
/*
 * this test method at the moment it is used just to run the
crawler
*/
@Test
public void testCrawler() {

    // c.crawl("http://www.dcs.bbk.ac.uk/~martin");
    //c.crawl("http://www.dcs.bbk.ac.uk/~mark");
    //c.crawl("http://www.dcs.bbk.ac.uk/staff");
    //c.crawl("http://www.dcs.bbk.ac.uk");

    //c.crawl("http://www.repubblica.it/economia/rapporti/osserva-
italia");
    String ret =
crawler.crawl("http://www.dcs.bbk.ac.uk/~michael");
    assertTrue ( ret!= null && ret.length() > 0 );
    //c.crawl("http://www.dcs.bbk.ac.uk/horde");
    // c.crawl("http://titan.dcs.bbk.ac.uk/~srapis01");
    //c.crawl("http://www.bestinvest.co.uk");
}

@Test
public void testReplaceInternalIperlink(){
    HtmlParser p = new HtmlParser(null);

    String expected = "http://www.repubblica.it/spettacoli";
    String actual =
p.checkURL("http://www.repubblica.it/spettacoli/#1");

    assertEquals(expected, actual);

}

@Test
public void testReplaceInternalIperlink2(){
    HtmlParser p = new HtmlParser(null);

    String expected = "http://www.repubblica.it/spettacoli";
    String actual =
```

```
p.checkURL("http://www.repubblica.it/spettacoli/####122");
    assertEquals(expected, actual);

}

@Test
public void testReplaceInternalIperlink3(){
    HtmlParser p = new HtmlParser(null);

    String expected = "http://www.repubblica.it/spettacoli";
    String actual =
p.checkURL("http://www.repubblica.it/spettacoli///####1222");

    assertEquals(expected, actual);

}

@Test
public void testReplaceInternalIperlink4(){
    HtmlParser p = new HtmlParser(null);

    String expected = "http://www.repubblica.it/spettacoli";
    String actual =
p.checkURL("http://www.repubblica.it/spettacoli///####AAAAAAA");

    assertEquals(expected, actual);

}

@Test
public void testReplaceInternalIperlink5(){
    HtmlParser p = new HtmlParser(null);

    String expected = "http://www.repubblica.it/spettacoli";
    String actual =
p.checkURL("http://www.repubblica.it/spettacoli///####AA222AAAAA3");
```

```
        assertEquals(expected, actual);

    }

    @Test
    public void testReplaceInternalIperlink6(){
        HtmlParser p = new HtmlParser(null);

        String expected = "http://www.repubblica.it/spettacoli";
        String actual =
p.checkURL("http://www.repubblica.it/spettacoli///#/####AA222AAAAA3");

        assertEquals(expected, actual);

    }

    @Test
    public void testReplaceInternalIperlink7(){
        HtmlParser p = new HtmlParser(null);

        String expected = "http://www.repubblica.it/spettacoli";
        String actual =
p.checkURL("http://www.repubblica.it/spettacoli/");

        assertEquals(expected, actual);

    }

}
```

8.9.2.2 TestJenaDataProvider.java

```
package wonts.triples.crawler;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertNotNull;
import static org.junit.Assert.assertTrue;

import java.io.IOException;
```

```
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.stream.Stream;

import org.junit.Assert;
import org.junit.BeforeClass;
import org.junit.Test;

import wonts.triples.data.model.JenaResourceModel;
import wonts.triples.data.persistence.DataProvider;
import wonts.triples.data.properties.PropertiesManager;

import com.hp.hpl.jena.query.QuerySolution;
import com.hp.hpl.jena.query.ResultSet;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.Property;
import com.hp.hpl.jena.rdf.model.RDFNode;
import com.hp.hpl.jena.rdf.model.Resource;
import com.hp.hpl.jena.rdf.model.Statement;
import com.hp.hpl.jena.rdf.model StmtIterator;
import com.hp.hpl.jena.reasoner.Reasoner;
import com.hp.hpl.jena.reasoner.ReasonerRegistry;
import com.hp.hpl.jena.reasoner.rulesys.RDFSRuleReasonerFactory;

public class TestJenaDataProvider {

    @BeforeClass
    public static void setUpBeforeClass() throws Exception {
    }

    private List<String> getLinkOutFromFile(Path path) throws
IOException{
        List<String> linkOut = new ArrayList<>();
        try(Stream<String> lines = Files.lines(path)){;
```

```
        lines.forEach(linkOut::add);
        //Close the stream and it's underlying file as well
        lines.close();
    }
    return linkOut;
}

public int countDuplicates(List<String>
listContainingDuplicates) {
    AtomicInteger ret = new AtomicInteger();
    final Set<String> set1 = new HashSet<String>();

    listContainingDuplicates.forEach( p-> {
        if (!set1.add(p)) {
            ret.addAndGet(1);
        }
    });
    return ret.get();
}

@Test
public void TestRdfDataProvider() {

    String url = "http://www.dcs.bbk.ac.uk/~mark";
    List<String> linkOut = null;
    try {
        linkOut =
getLinkOutFromFile(Paths.get("tests/", "TestDataCrowler.txt"));
    } catch (IOException e) {
        Assert.fail();
    }

    //          List<String> linkOut = new ArrayList<>();
    //
    linkOut.add("http://www.dcs.bbk.ac.uk/research/dbtech" );
    //
    linkOut.add("http://www.dcs.bbk.ac.uk");
    //
    linkOut.add("http://www.bbk.ac.uk");

    assertNotNull(linkOut);
    assertTrue(linkOut.size()>0);

    JenaResourceModel j = new JenaResourceModel();
```

```

        j.AddUrlLinksVisited(url, linkOut);
        StmtIterator iter =
j.getRdfModel().listStatements();
        int actual = 0;

        int numDuplicates = countDuplicates(linkOut);
        int expected = linkOut.size() - numDuplicates;

        System.out.println(" num of duplicates in the
list: " + numDuplicates);

        List<String> actualLinks = new
ArrayList<String>();
        while (iter.hasNext()) {
            Statement stmt      = iter.nextStatement();
// get next statement
            Resource subject    = stmt.getSubject();
// get the subject
            Property predicate  = stmt.getPredicate();
// get the predicate
            RDFNode object      = stmt.getObject();
// get the object

            System.out.print(" subject: " +
subject.toString());
            System.out.print(" predicate: " +
predicate.toString() + " ");
            if (object instanceof Resource) {
                System.out.print(" object r:" +
object.toString());
            } else {
// object is a literal
                System.out.print(" object l:" +
object.toString() );
            }
            actualLinks.add(object.toString());
            actual++;
            System.out.println();
        }

        j.getRdfModel().write(System.out);

        linkOut.forEach(p-> {

```

```
        if ( ! actualLinks.contains(p) ) {
            System.out.print(" THIS IS NOT
CONTAINED: " + p.toString() );
        }
    } );  
  
        assertEquals(expected, actual);
    }  
  
    @Test
    public void TestPersistencePropertyFile_DB(){
        String actual =
PropertiesManager.getProperty(PropertiesManager.DB);
        assertTrue(actual != null && actual.length()>0);  
  
    }  
  
    @Test
    public void TestPersistencePropertyFile_URL(){
        String actual =
PropertiesManager.getProperty(PropertiesManager.DB_URL);
        assertTrue(actual != null && actual.length()>0);  
  
    }  
  
    @Test
    public void TestPersistencePropertyFile_DB_USER(){
        String actual =
PropertiesManager.getProperty(PropertiesManager.DB_USER);
        assertTrue(actual != null && actual.length()>0);  
  
    }  
  
    @Test
    public void TestPersistencePropertyFile_CLASS_NAME(){
        String actual =
PropertiesManager.getProperty(PropertiesManager.CLASS_NAME);
        assertTrue(actual != null && actual.length()>0);  
  
    }  
  
    @Test
```

```
public void TestPersistencePropertyFile_PROPERTY_FILE_NAME(){
    String actual = PropertiesManager.

        GetProviderProperty(PropertiesManager.PROPERTY_FILENAME).
            getProperty(PropertiesManager.DB_URL);
        assertTrue(actual != null && actual.length()>0);
}

@Test
public void TestPersistenceWriteOnDB(){

    DataProvider p = new DataProvider();
    String filename="tests/60c40c45-40d6-4a7d-9c8c-
2001e62ce914.owl";
    boolean actual = p.PutRdfInDataBase(filename);
    assertTrue(actual);
}

@Test
public void TestResourcesModelNotEmpty(){
    JenaResourceModel m = new JenaResourceModel();
    assertTrue( m.getWebResources() != null );
    assertTrue(m.getWebResources().size()>0);
}

@Test
public void TestPersistenceQueryResultFormattedAsTripleLinkin(){
    DataProvider p = new DataProvider();

    String query= "PREFIX rdf: <http://www.w3.org/1999/02/22-
rdf-syntax-ns#>"
                + " PREFIX owl: <http://www.w3.org/2002/07/owl#>"
                + " PREFIX xsd:
<http://www.w3.org/2001/XMLSchema#>"
                + " PREFIX rdfs: <http://www.w3.org/2000/01/rdf-
schema#>"
                + " PREFIX j.0:
<http://titan.dcs.bbk.ac.uk/~srapis01#>
                + " SELECT ?s ?o"
                + " WHERE { ?o j.0:linksOut ?s ."
                + " }";

    String results = p.execSelectQuery(query,"N-TRIPLE");
```

```
        System.out.println(results);
        assertNotNull( results );
    }

    @Test
    public void
TestPersistenceQueryResultFormattedAsTriple_Filtered_LIMIT(){
    DataProvider p = new DataProvider();

        String query= "PREFIX rdf: <http://www.w3.org/1999/02/22-
rdf-syntax-ns#>" +
                    + " PREFIX owl: <http://www.w3.org/2002/07/owl#>" +
                    + " PREFIX xsd:
<http://www.w3.org/2001/XMLSchema#>" +
                    + " PREFIX rdfs: <http://www.w3.org/2000/01/rdf-
schema#>" +
                    + " PREFIX j.0:
<http://titan.dcs.bbk.ac.uk/~srapis01#>" +
                    + " SELECT ?o " +
                    + " WHERE { ?s ?p ?o ." +
// + "           ?o rdf:type j.0:html " +
                    + " FILTER regex(str(?s),
\"http://www.dcs.bbk.ac.uk/~michael\", \"i\") "
                    + " } LIMIT 10 ";

        String results = p.execSelectQuery(query,"N-TRIPLE");
        System.out.println(results);
        assertNotNull( results );
    }

    @Test
    public void TestPersistenceQueryResultFormattedAsTriple_model(){
    DataProvider p = new DataProvider();

        String query= "PREFIX rdf: <http://www.w3.org/1999/02/22-
rdf-syntax-ns#>" +
                    + " PREFIX owl: <http://www.w3.org/2002/07/owl#>" +
                    + " PREFIX xsd:
<http://www.w3.org/2001/XMLSchema#>" +
                    + " PREFIX rdfs: <http://www.w3.org/2000/01/rdf-
schema#>" +
                    + " PREFIX wonts:
<http://titan.dcs.bbk.ac.uk/~srapis01#>"
```

```
+ " SELECT ?o "
+ " WHERE { ?s wonts:linksOut ?o ."
// + " FILTER regex(str(?s),
\"http://www.dcs.bbk.ac.uk/~michael\", \"i\") "
+ "} LIMIT 10 ";

String results = p.execSelectQuery(query, "N-TRIPLE");
System.out.println(results);
assertNotNull( results );
}

@Test
public void
TestPersistenceQueryResultFormattedAsTriple_Filtered_Property(){
DataProvider p = new DataProvider();

String query= "PREFIX rdf: <http://www.w3.org/1999/02/22-
rdf-syntax-ns#>"
+ " PREFIX owl: <http://www.w3.org/2002/07/owl#>"
+ " PREFIX xsd:
<http://www.w3.org/2001/XMLSchema#>"
+ " PREFIX rdfs: <http://www.w3.org/2000/01/rdf-
schema#>"
+ " PREFIX j.0:
<http://titan.dcs.bbk.ac.uk/~srapis01#>"
+ " SELECT ?s ?p ?o "
+ " WHERE { ?s ?p ?o ."
+ "         ?p rdf:property j.0:linksOut ."
+ " FILTER regex(str(?s),
\"http://www.dcs.bbk.ac.uk/~michael\", \"i\") "
+ "} LIMIT 10 ";

String results = p.execSelectQuery(query, "N-TRIPLE");
System.out.println(results);
assertNotNull( results );
}

@Test
public void
TestPersistenceQueryResultFormattedAsTriple_Filtered(){
DataProvider p = new DataProvider();
```

```
String query= "PREFIX rdf: <http://www.w3.org/1999/02/22-
rdf-syntax-ns#>"  
        + " PREFIX owl: <http://www.w3.org/2002/07/owl#>"  
        + " PREFIX xsd:  
<http://www.w3.org/2001/XMLSchema#>"  
        + " PREFIX rdfs: <http://www.w3.org/2000/01/rdf-
schema#>"  
        + " PREFIX j.0:  
<http://titan.dcs.bbk.ac.uk/~srapis01#>"  
        + " PREFIX res:  
<http://www.repubblica.it/economia/>"  
        + " SELECT ?s ?p ?o  "  
        + " WHERE { ?s ?p ?o ."  
    // + "      ?o  rdf:type j.0:html "  
    + " FILTER regex(str(?s),  
\"http://www.repubblica.it/economia/rapporti\", \"i\") "  
    + " }";  
  
String results = p.execSelectQuery(query,"N-TRIPLE");  
System.out.println(results);  
assertNotNull( results );  
}  
  
  


```
@Test
public void TestPersistenceQueryResultFormattedAsTriple(){
 DataProvider p = new DataProvider();

 String query= "PREFIX rdf: <http://www.w3.org/1999/02/22-
rdf-syntax-ns#>"
 + " PREFIX owl: <http://www.w3.org/2002/07/owl#>"
 + " PREFIX xsd:
<http://www.w3.org/2001/XMLSchema#>"
 + " PREFIX rdfs: <http://www.w3.org/2000/01/rdf-
schema#>"
 + " PREFIX j.0:
<http://titan.dcs.bbk.ac.uk/~srapis01#>"
 + " SELECT ?s"
 + " WHERE { ?s rdfs:subClassOf j.0:WebResource
}";

String results = p.execSelectQuery(query,"N-TRIPLE");
System.out.println(results);
assertNotNull(results);
```


```

```

}

@Test
public void TestPersistenceQuery(){
    DataProvider p = new DataProvider();

    String query= "PREFIX rdf: <http://www.w3.org/1999/02/22-
rdf-syntax-ns#>"
        + " PREFIX owl: <http://www.w3.org/2002/07/owl#>"
        + " PREFIX xsd:
<http://www.w3.org/2001/XMLSchema#>"
        + " PREFIX rdfs: <http://www.w3.org/2000/01/rdf-
schema#>"
        + " PREFIX j.0:
<http://titan.dcs.bbk.ac.uk/~srapis01#>"
        + " SELECT ?s"
        + " WHERE { ?s rdfs:subClassOf j.0:WebResource
}"; 

    ResultSet results = p.execSelectQuery(query );
    assertNotNull( results );
    assertNotNull( results );

    List<String> vars = results.getResultVars();
    for ( ; results.hasNext() ; ){
        QuerySolution soln = results.nextSolution();
        System.out.println();
        vars.forEach( v-> { System.out.print( v + ":" +
soln.get( v ) + "; " ); } );
    }
}

@Test
public void TestDeleteUrl(){
    DataProvider p = new DataProvider();

    String urlString =
"http://www.repubblica.it/economia/rapporti/ossevra-italia/eventi";
    assertTrue( p.deleteLinkOutPages(urlString ) );

    String query= "PREFIX rdf: <http://www.w3.org/1999/02/22-
rdf-syntax-ns#>"
        + " PREFIX owl: <http://www.w3.org/2002/07/owl#>"
```

```

+ " PREFIX xsd:
<http://www.w3.org/2001/XMLSchema#>
+ " PREFIX rdfs: <http://www.w3.org/2000/01/rdf-
schema#>
+ " PREFIX j.0:
<http://titan.dcs.bbk.ac.uk/~srapis01#>
+ " SELECT ?s ?o "
+ " WHERE { ?s j.0:linksOut ?o ."
+ " FILTER regex(str(?s), \"" + urlString + "\", 
\"i\")
+ " }";

ResultSet results = p.execSelectQuery(query);
assertTrue( results.hasNext() == false );

}

@Test
public void TestReasonerInDataModel(){
    Model schema = null;
    Reasoner reasoner = ReasonerRegistry.getRDFSReasoner();
    reasoner.bindSchema(schema);

}

@Test
public void TestMakeDisjoinResourcesModel(){
    JenaResourceModel m = new JenaResourceModel();
    m.MakeDisjointResources();
    m.getWebResources().forEach((k1,v1)-> {
        m.getWebResources().forEach((k2,v2)-> {
            if ( ! v1.equals( v2 ) ){
                Assert.assertTrue(v1.isDisjointWith(v2));
            }
        });
    });
}
}

```

8.9.3 wonts.triples.crawler.base

8.9.3.1 ContextConfigurationTest.java

```
package wonts.triples.crawler.base;
```

```
import org.junit.runner.RunWith;
import org.springframework.test.context.ActiveProfiles;
import org.springframework.test.context.ContextConfiguration;
import
org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

@ActiveProfiles("test")
@ContextConfiguration( { "beans_test.xml"})
@RunWith(SpringJUnit4ClassRunner.class)
public class ContextConfigurationTest {
    public ContextConfigurationTest(){}
}
```

8.9.3.2 beans_test.xml

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:util="http://www.springframework.org/schema/util"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans-
2.5.xsd
                           http://www.springframework.org/schema/aop
                           http://www.springframework.org/schema/aop/spring-aop-
2.5.xsd
                           http://www.springframework.org/schema/context
                           http://www.springframework.org/schema/context/spring-
context-2.5.xsd">

<!-- <context:component-scan base-
package="wonts.triples.crawler.parser" /> -->

    <bean id="parser" class="wonts.triples.crawler.parser.HtmlParser"
scope="prototype" />
        <bean id="resource"
class="wonts.triples.data.model.JenaResourceModel" scope="prototype"
/>
            <bean id="crawler" class="wonts.triples.crawler.JenaCrawler"
scope="prototype" />
```

```
<bean id="robotTxt" class="wonts.triples.crawler.robot.RobotTxt"
scope="prototype" />
<bean id="crawlUrlProvider"
class="wonts.triples.crawler.services.CrawlUrlProvider"
scope="prototype" />

</beans>
```

8.9.4 wonts.triples.crawler.services

8.9.4.1 CrawlUrlProviderTest.java

```
package wonts.triples.crawler.services;

import static org.junit.Assert.assertTrue;
import static
wonts.triples.crawler.SpringExtension.SpringExtProvider;

import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.TimeUnit;

import javax.inject.Inject;

import org.junit.AfterClass;
import org.junit.BeforeClass;
import org.junit.Test;
import
org.springframework.context.annotation.AnnotationConfigApplicationContext
ext;

import scala.concurrent.Await;
import scala.concurrent.Future;
import scala.concurrent.duration.Duration;
import wonts.data.services.model.Url;
import wonts.data.services.model.UrlToCrawl;
import wonts.data.services.provider.UrlServiceProvider;
import wonts.triples.crawler.Crawler;
import wonts.triples.crawler.services.messages.CrawlUrlMessage;
import
wonts.triples.crawler.services.messages.response.CrawlUrlMessageResp;
import akka.actor.ActorRef;
import akka.actor.ActorSystem;
import akka.actor.PoisonPill;
import akka.actor.UntypedActor;
```

```
import akka.japi.Creator;
import akka.testkit.JavaTestKit;

public class CrawlUrlProviderTest { //extends
ContextConfigurationTest {

    static ActorSystem system;

    @BeforeClass
    public static void setup() {
        ctx = new AnnotationConfigApplicationContext();
        ctx.scan("wonts");
        ctx.refresh();

        system = ctx.getBean(ActorSystem.class);
    }

    @AfterClass
    public static void teardown() {
        JavaTestKit.shutdownActorSystem(system);
        system = null;
    }

    // @Inject
    static AnnotationConfigApplicationContext ctx;

    @Inject
    Crawler crawler;

    private void AddTwoURLToCrawl(UrlServiceProvider
urlServiceProvider){

        String urlValue = "http://www.dcs.bbk.ac.uk/~mark";
        String urlDescription = "description mark";

        String urlValue2 =
"http://www.dcs.bbk.ac.uk/~michael";
        String urlDescription2 = "description mark";
        Url url = new Url(urlValue, urlDescription);
        UrlToCrawl u = urlServiceProvider.saveUpdate( url );
        u.setActive(true);
    }
}
```



```
"crawlUrlProvider" + p.getUrlToCrawlId() );
    CrawlUrlMessage msg = new CrawlUrlMessage();
    msg.setUrlToCrawlId(p.getUrlToCrawlId());
    msg.setUrl( p.getUrl().getUrlValue());
    crawlerRef.tell(msg, probe.getRef());
    probe.expectMsgClass(Duration.create(1,
TimeUnit.HOURS), CrawlUrlMessageResp.class);
}

// assertEquals(expected, actual.size());
}

@Test
public void getAllActiveUrlAndICrawlItTest2(){
    UrlServiceProvider urlServiceProvider =
ctx.getBean(UrlServiceProvider.class);
    AddTwoURLToCrawl(urlServiceProvider);

}

@Test
public void crawlUrlProviderActorTest2() throws Exception{
    UrlServiceProvider urlServiceProvider =
ctx.getBean(UrlServiceProvider.class);
    AddTwoURLToCrawl(urlServiceProvider);

    ActorRef checkUrlToCrawlRef = system.actorOf(
SpringExtProvider.get(system).props("checkUrlToCrawl"),
"checkUrlToCrawl");

    final Future<Object> future =
akka.pattern.Patterns.ask(checkUrlToCrawlRef, "Check",
Duration.create(1, TimeUnit.HOURS).toMillis());

    String path = (String) Await.result(future,
Duration.create(1, TimeUnit.HOURS));
    System.out.println(path);
    assertTrue( path != null );
}
```

```
//path.toFile().delete();
}

}

8.9.5 CrawlUrlProviderTest2
package wonts.triples.crawler.services;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;
import static
wonts.triples.crawler.SpringExtension.SpringExtProvider;

import java.util.List;
import java.util.concurrent.TimeUnit;

import org.junit.AfterClass;
import org.junit.BeforeClass;
import org.junit.Test;
import
org.springframework.context.annotation.AnnotationConfigApplicationContext
ext;

import scala.concurrent.Await;
import scala.concurrent.Future;
import scala.concurrent.duration.Duration;
import wonts.data.services.model.Url;
import wonts.data.services.model.UrlToCrawl;
import wonts.data.services.provider.UrlServiceProvider;
import wonts.triples.crawler.services.messages.CrawlUrlMessage;
import
wonts.triples.crawler.services.messages.response.CrawlUrlMessageResp;
import akka.actor.ActorRef;
import akka.actor.ActorSystem;
import akka.testkit.JavaTestKit;

public class CrawlUrlProviderTest2 { //extends
ContextConfigurationTest {

    static ActorSystem system;
```

```
@BeforeClass
public static void setup() {
    ctx = new AnnotationConfigApplicationContext();
    ctx.scan("wonts");
    ctx.refresh();
    system = ctx.getBean(ActorSystem.class);
}

@BeforeClass
public static void teardown() {
    JavaTestKit.shutdownActorSystem(system);
    system = null;
}

static AnnotationConfigApplicationContext ctx;

public void getActorSystem(){
    AnnotationConfigApplicationContext ctx1 =
        new AnnotationConfigApplicationContext();
    ctx1.scan("wonts.triples");
    ctx1.refresh();

    // get hold of the actor system
    ActorSystem system1 = ctx1.getBean(ActorSystem.class);
    assertTrue(system1 != null);
    ctx1.close();
}

@Test
public void crawlUrlProviderActorTest() throws Exception{
    CrawlUrlMessage message = new CrawlUrlMessage();
    message.setUrl("http://www.dcs.bbk.ac.uk/~srapis01");
    //message.setUrl("http://roma.repubblica.it/sport");
    ActorRef crawlerRef = system.actorOf(
        SpringExtProvider.get(system).props("crawlUrlProvider"),
        "crawlUrlProvider");
    final Future<Object> future =
        akka.pattern.Patterns.ask(crawlerRef, message, Duration.create(1,
        TimeUnit.HOURS).toMillis());
}
```

```
    CrawlUrlMessageResp path = (CrawlUrlMessageResp)
Await.result(future, Duration.create(1, TimeUnit.HOURS));
    System.out.println(path);
    assertTrue( (path!=null) );
}

@Test
public void getAllActiveUrlToCrawlTest(){

    UrlServiceProvider urlServiceProvider =
ctx.getBean(UrlServiceProvider.class);

    String urlValue = "http://www.dcs.bbk.ac.uk/~mark";
    String urlDescription = "description mark";

    Url url = new Url(urlValue, urlDescription);
    UrlToCrawl u = urlServiceProvider.saveUpdate( url );
    u.setActive(true);
    u.setStatus(UrlToCrawl.Status.isToCrawl.ordinal());

    UrlToCrawl expected = urlServiceProvider.saveUpdate(u);

    List<UrlToCrawl> actual =
urlServiceProvider.getAllActiveUrlToCrawl();

    assertEquals(expected.getUrlToCrawlId(),
actual.get(0).getUrlToCrawlId());

    u.setActive(false);
    expected = urlServiceProvider.saveUpdate(u);
    assertTrue(!expected.isActive()));

}

@Test
public void getAllActiveUrlToCrawlCheckUrlValueTest(){

    UrlServiceProvider urlServiceProvider =
ctx.getBean(UrlServiceProvider.class);
```

```
String urlValue = "http://www.dcs.bbk.ac.uk/~mark";
String urlDescription = "description mark";

Url url = new Url(urlValue, urlDescription);
UrlToCrawl u = urlServiceProvider.saveUpdate( url );
u.setActive(true);
u.setStatus(UrlToCrawl.Status.isToCrawl.ordinal());

UrlToCrawl expected = urlServiceProvider.saveUpdate(u);

List<UrlToCrawl> actual =
urlServiceProvider.getAllActiveUrlToCrawl();

assertEquals(urlValue,
actual.get(0).getUrl().getUrlValue());

u.setActive(false);
expected = urlServiceProvider.saveUpdate(u);
assertTrue(!expected.isActive());

}

@Test
public void getAllActive100UrlToCrawlTest(){

UrlServiceProvider urlServiceProvider =
ctx.getBean(UrlServiceProvider.class);

String urlValue = "http://www.dcs.bbk.ac.uk/~mark";
String urlDescription = "description mark";
int expected = 100;
for(int x=0; x<expected; x++){
    Url url = new Url(urlValue+x, urlDescription);
    UrlToCrawl u = urlServiceProvider.saveUpdate( url );
    u.setActive(true);

    u.setStatus(UrlToCrawl.Status.isToCrawl.ordinal());
}
```

```
        urlServiceProvider.saveUpdate(u);
    }

    List<UrlToCrawl> actual =
urlServiceProvider.getAllActiveUrlToCrawl();

assertEquals(expected, actual.size());

}

8.9.6 data_context_beans.xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx.xsd">

    <bean id="emf"
          class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
        <property name="dataSource" ref="dataSource" />
        <property name="packagesToScan"
value="wonts.data.services.model" />
        <property name="jpaVendorAdapter">
            <bean
class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter"
/>
        </property>
        <property name="jpaProperties">
```

```

<props>
    <prop key="hibernate.hbm2ddl.auto">create-drop</prop>
    <prop
key="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</prop>
    </props>
</property>
</bean>

<bean id="dataSource"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="com.mysql.jdbc.Driver"
/>
    <property name="url"
value="jdbc:mysql://localhost:3306/wontsapp_test" />
    <property name="username" value="wontsuser" />
    <property name="password" value="user$wonts" />
</bean>

<bean id="transactionManager"
class="org.springframework.orm.jpa.JpaTransactionManager">
    <property name="entityManagerFactory" ref="emf" />
</bean>

<context:component-scan base-package="wonts.data.services" />
<tx:annotation-driven transaction-manager="transactionManager"/>

<!-- Root Context: defines shared resources visible to all other
web components -->
</beans>
```

8.9.7 resources

```

8.9.7.1 data_context_beans_test.xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">
```

```

<bean id="emf"
class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
    <property name="dataSource" ref="dataSource" />
    <property name="packagesToScan"
value="wonts.data.services.model" />
    <property name="jpaVendorAdapter">
        <bean
class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter"
/>
        </property>
    <property name="jpaProperties">
        <props>
            <prop key="hibernate.hbm2ddl.auto">create-drop</prop>
            <prop
key="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</prop>
            </props>
        </property>
    </bean>

    <bean id="dataSource"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName" value="com.mysql.jdbc.Driver"
/>
        <property name="url"
value="jdbc:mysql://localhost:3306/wontsapp_test" />
        <property name="username" value="wontsuser" />
        <property name="password" value="user$wonts" />
    </bean>

    <bean id="transactionManager"
class="org.springframework.orm.jpa.JpaTransactionManager">
        <property name="entityManagerFactory" ref="emf" />
    </bean>

    <context:component-scan base-package="wonts.data.services" />

    <!-- Root Context: defines shared resources visible to all other
web components -->
</beans>

8.9.7.2 logback.xml
<?xml version="1.0" encoding="UTF-8"?>

```

```
<configuration>
    <appender name="FILE"
        class="ch.qos.logback.core.rolling.RollingFileAppender">
        <encoder>
            <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} -
%msg%n</pattern>
        </encoder>

        <rollingPolicy
            class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <fileNamePattern>logs\triples.crawler.%d{yyyy-MM-
dd}.%i.log</fileNamePattern>
            <timeBasedFileNamingAndTriggeringPolicy
            class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">
                <maxFileSize>5MB</maxFileSize>
            </timeBasedFileNamingAndTriggeringPolicy>
        </rollingPolicy>
    </appender>

    <root level="DEBUG">
        <appender-ref ref="FILE" />
    </root>
</configuration>
```

8.9.7.3 persistence.properties

```
persistence.className=com.mysql.jdbc.Driver
persistence.db=MySql
persistence.db.url=jdbc:mysql://localhost/wonts
persistence.db.urs=wontsuser
persistence.db.pwd=user$wonts
crawler.owl.path=/Users/salvo/Development/wonts/owl_crawled
crawler.check.sec=30
```

8.9.7.4 web-resources

html, pdf, doc, ps, ppd, gz, tar, zip, rar, 7z

8.10 triples.crawler (pom.xml)

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
```

```
<groupId>wonts</groupId>
<artifactId>triples.crawler</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>jar</packaging>

<name>triples.crawler</name>
<url>http://maven.apache.org</url>

<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <java-version>1.8</java-version>
        <org.springframework-version>4.1.3.RELEASE</org.springframework-
version>
</properties>
    <build>
        <plugins>
            <plugin>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.1</version>
                <configuration>
                    <source>${java-version}</source>
                    <target>${java-version}</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
<dependencies>
    <dependency>
        <groupId>commons-logging</groupId>
        <artifactId>commons-logging</artifactId>
        <version>1.1.2</version>
    </dependency>
    <dependency>
        <groupId>log4j</groupId>
        <artifactId>log4j</artifactId>
        <version>1.2.16</version>
    </dependency>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.9</version>
        <scope>test</scope>
    </dependency>
    <dependency>
```

```
<groupId>org.apache.jena</groupId>
<artifactId>apache-jena-libs</artifactId>
<type>pom</type>
<version>2.11.2</version>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.14.4</version>
</dependency>
<!-- Spring -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>${org.springframework-version}</version>
<!-- <exclusions> -->
    <!-- Exclude Commons Logging in favor of SLF4j -->
<!-- <exclusion> -->
<!--     <groupId>commons-logging</groupId> -->
<!--     <artifactId>commons-logging</artifactId> -->
<!--     </exclusion> -->
<!-- </exclusions> -->
</dependency>
<!-- <dependency> -->
<!--     <groupId>edu.stanford.protege</groupId> -->
<!--     <artifactId>org.protege.editor.owl</artifactId> -->
<!--     <version>4.3.0</version> -->
<!-- </dependency> -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.31</version>
</dependency>
<dependency>
    <groupId>org.apache.jena</groupId>
    <artifactId>jena-jdbc-core</artifactId>
    <version>1.1.0</version>
</dependency>
<dependency>
    <groupId>org.apache.jena</groupId>
    <artifactId>jena-sdb</artifactId>
    <version>1.5.0</version>
</dependency>
<dependency>
```

```
<groupId>commons-io</groupId>
<artifactId>commons-io</artifactId>
<version>2.4</version>
</dependency>
<dependency>
    <groupId>com.typesafe.akka</groupId>
    <artifactId>akka-actor_2.10</artifactId>
    <version>2.3.1</version>
</dependency>
<dependency>
    <groupId>com.typesafe.akka</groupId>
    <artifactId>akka-testkit_2.10</artifactId>
    <version>2.3.11</version>
</dependency>
<dependency>
    <groupId>com.typesafe.akka</groupId>
    <artifactId>akka-slf4j_2.11</artifactId>
    <version>2.3.12</version>
</dependency>
<dependency>
    <groupId>ch.qos.logback</groupId>
    <artifactId>logback-classic</artifactId>
    <version>1.1.3</version>
</dependency>

<dependency>
    <groupId>ch.qos.logback</groupId>
    <artifactId>logback-core</artifactId>
    <version>1.1.3</version>
</dependency>
<dependency>
    <groupId>javax.inject</groupId>
    <artifactId>javax.inject</artifactId>
    <version>1</version>
</dependency>
<dependency>
    <groupId>wonts</groupId>
    <artifactId>data.services</artifactId>
    <version>0.0.1-SNAPSHOT</version>
</dependency>
</dependencies>

</project>
```

8.11 webapp (src/main/java)

8.11.1 wonts.webapp

8.11.1.1 *ServletInitializer.java*

```
package wonts.webapp;

import org.springframework.boot.builder.SpringApplicationBuilder;
import
org.springframework.boot.context.web.SpringBootServletInitializer;

public class ServletInitializer extends SpringBootServletInitializer
{

    @Override
    protected SpringApplicationBuilder
configure(SpringApplicationBuilder application) {
        return application.sources(WontsWebAppApplication.class);
    }

}
```

8.11.1.2 *WontsWebAppApplication.java*

```
package wonts.webapp;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.orm.jpa.EntityScan;
import org.springframework.context.annotation.ComponentScan;
import
org.springframework.data.jpa.repository.config.EnableJpaRepositories;

@ComponentScan("wonts")
@EntityScan("wonts.data.services.model")
@EnableJpaRepositories
@SpringBootApplication
public class WontsWebAppApplication {

    public static void main(String[] args) {
        SpringApplication.run(WontsWebAppApplication.class, args);
    }
}
```

8.11.2 wonts.webapp.controllers

8.11.2.1 *ProjectController.java*

```
package wonts.webapp.controllers;

import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.security.Principal;
import java.util.ArrayList;
import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.validation.groups.Default;

import org.apache.commons.io.IOUtils;
import org.apache.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotationResponseStatus;
import org.springframework.web.multipart.MultipartFile;

import wonts.data.services.model.Ontology;
import wonts.data.services.model.OntologyStatus;
import wonts.data.services.model.Project;
import wonts.data.services.model.Url;
import wonts.data.services.model.User;
import wonts.data.services.provider.OntologyServiceProvider;
import wonts.data.services.provider.ProjectServiceProvider;
import wonts.data.services.provider.UrlServiceProvider;
import wonts.data.services.provider.UserServiceProvider;
import wonts.webapp.dto.ProjectDTO;
import wonts.webapp.dto.UrlDTO;
```

```
import wonts.webapp.property.PropertiesManager;

/**
 *
 * @author salvatore
 *
 *      this controller handles all the project requests
 *
 */
@Controller
@RequestMapping("/api/projects")
public class ProjectController {

    private final Logger log =
Logger.getLogger(ProjectController.class);

    @Autowired
    ProjectServiceProvider projectServiceProvider;

    @Autowired
    UserServiceProvider userServiceProvider;

    @Autowired
    OntologyServiceProvider ontologyServiceProvider;

    @Autowired
    UrlServiceProvider urlServiceProvider;

    @ResponseBody
    @ResponseStatus(HttpStatus.OK)
    @RequestMapping(value= "/{projectId}",
produces="application/json", method = RequestMethod.GET)
    public ProjectDTO getProjectById(@PathVariable int projectId,
Principal principal){
        if (principal==null) return null;
        User user =
userServiceProvider.getUserByUsername(principal.getName());
        if ( user != null ){
            Project project =
projectServiceProvider.getProject(projectId);
            List<Ontology> ontologies =
ontologyServiceProvider.getOntologiesByProject(project);
            project.setOntologies(ontologies);
            ProjectDTO projectDTO = new ProjectDTO( project );
        }
    }
}
```

```

        return projectDTO;
    }
    return null;
}

@ResponseBody
@ResponseStatus(HttpStatus.OK)
@RequestMapping(value = "/new", produces="application/json", method
= RequestMethod.POST)
public ProjectDTO newProject( @RequestBody ProjectDTO
newProject, Principal principal){
    if (principal==null) return null;
    log.info("new project for user: " + principal.getName() + " for
new project " + newProject );
    ProjectDTO ret = null;
    User user =
userServiceProvider.getUserByUsername(principal.getName());
    Project project = new Project(newProject.getProjectName(),
newProject.getProjectDescription(), user);
    project = projectServiceProvider.saveUpdate(project);
    ret = new ProjectDTO(project);
    log.info("user: " + principal.getName() + " added new project:
" + ret );
    return ret;
}

@ResponseBody
@ResponseStatus(HttpStatus.OK)
@RequestMapping(value = "/urls/{urlId}/set-
status", produces="application/json", method = RequestMethod.POST)
public boolean setUrlStatus(@RequestBody int isActive,
@PathVariable int urlId, Principal principal){
    if (principal == null) return false;
    Url url = urlServiceProvider.getUrlById(urlId);
    if (url == null ) return false;
    url.setActive(isActive==1);
    urlServiceProvider.saveUpdate(url);
    return true;
}

@ResponseBody
@ResponseStatus(HttpStatus.OK)

```

```
    @RequestMapping(value =
"/{projectId}/urls/new", produces="application/json", method =
RequestMethod.POST)
    public boolean newUrl(@RequestBody UrlDTO newUrl, @PathVariable
int projectId, Principal principal){
        if (principal == null) return false;
        log.info("new project for user: " + principal.getName() + " for
new url " + newUrl );
        ProjectDTO ret = null;

        User user =
userServiceProvider.getUserByUsername(principal.getName());
        if ( user == null ) return false;

        Project project = projectServiceProvider.getProject(projectId);
        if (project == null ) return false;

        // check if is already present the project
        Url alreadyUrl =
urlServiceProvider.getUrlByValue(newUrl.getUrlValue());
        if ( alreadyUrl != null && alreadyUrl.getProject() != null
&& alreadyUrl.getProject().stream().anyMatch(p>
p.getProjectId() == projectId) ){
            return false;
        }

        Url url =
urlServiceProvider.getUrlByValue(newUrl.getUrlValue());

        if ( url == null){
            url = new Url( newUrl.getUrlValue(),
newUrl.getUrlDescription());
        }

        if (url.getProject() == null ){
            url.setProject(new ArrayList<Project>());
        }
        url.getProject().add(project);

        project.getUrls().add(url);

        urlServiceProvider.saveUpdate(url);
```

```
projectServiceProvider.saveUpdate(project);

    ret = new ProjectDTO(project);
    log.info("user: " + principal.getName() + " added new project:
" + ret );
        return true;
    }

@ResponseBody
@ResponseStatus(HttpStatus.OK)
@RequestMapping( produces="application/json", method =
RequestMethod.GET)
public List<ProjectDTO> projects(Principal principal){
if (principal==null) return null;

List<ProjectDTO> ret = new ArrayList<ProjectDTO>();

User user =
userServiceProvider.getUserByUsername(principal.getName());
List<Project> projects =
projectServiceProvider.getProjects(user);

projects.forEach(p-> {
    List<Ontology> ontologies =
ontologyServiceProvider.getOntologiesByProject(p);
    if (ontologies!= null){
        //p.getOntologies().addAll(ontologies);
    }
    ret.add( new ProjectDTO (p));
});

return ret;
}

@ResponseBody
@ResponseStatus(HttpStatus.OK)
@RequestMapping( value="/ontology-delete/{ontologyId}",
produces="application/json", method = RequestMethod.POST)
public boolean ontologyDelete( @PathVariable int ontologyId,
Principal principal){
```

```

Ontology ontology =
ontologyServiceProvider.getOntology(ontologyId);

if ( ontology != null ){
    String name = ontology.getOntologyName();
    String fullName = getOntologyPath(name);
    if ( new File( fullName ).delete() ) {
        log.debug("The file does not exist: " + fullName ) ;
    }
    return ontologyServiceProvider.remove(ontology);
}
return false;
}

@ResponseBody
@ResponseStatus(HttpStatus.OK)
@RequestMapping( value="/ontology-upload/{projectId}",
produces="application/json", method = RequestMethod.POST)
public boolean upload(@PathVariable int projectId, String
name, MultipartFile file, Principal principal){
    log.info("file: " + file);
    log.info("name: " + name);
    // I ensure that the project exist
    Project project = projectServiceProvider.getProject(projectId);
    if ( project == null ){ return false; }

    if (!file.isEmpty()) {
        try {
            byte[] bytes = file.getBytes();
            name = file.getOriginalFilename();
            BufferedOutputStream stream =
                new BufferedOutputStream(new
FileOutputStream(
                    new File( getOntologyPath(name) ))));
            stream.write(bytes);
            stream.close();

            // add new ontology
            Ontology ontology = new Ontology(name, name,
project);
            ontology= ontologyServiceProvider.saveUpdate(ontology
);
            if (project.getOntologies() == null ) {
                project.setOntologies(new ArrayList<Ontology>());

```

```

        }
        project.getOntologies().add(ontology);
        projectServiceProvider.saveUpdate(project);
        log.info( "Successfully uploaded " + name + "!" );
        return true;
    } catch (Exception e) {
        log.error( "You failed to upload " + name + " => ", e
);
        return false;
    }
} else {
    return false;
}
}

@ResponseBody
@ResponseStatus(HttpStatus.OK)
@RequestMapping( value="/ontologies/{ontologyId}/delete/" ,
produces="application/json", method = RequestMethod.DELETE)
public boolean ontologyDelete( @PathVariable int ontologyId,
Principal principal){
    if (principal == null || ontologyId<=0 ) return false;

    Ontology ontology =
ontologyServiceProvider.getOntology(ontologyId);
    try{
        String name = ontology.getOntologyName();
        log.info("Deleting ontology: id: " +
ontology.getOntologyId() + " name: " + name + " by user: " +
principal.getName() );
        deledeOntologyFile(getOntologyPath(
ontology.getOntologyName()) );

        deledeOntologyFile(getOntologyPath(ontology.getOntologyProcessedN
ame()) );

        Project project =
projectServiceProvider.getProject(ontology.getProject().getProjectId()
);

        project.getOntologies().remove(ontology);
        projectServiceProvider.saveUpdate(project);

        ontologyServiceProvider.remove(ontology);
    }
}

```

```

}catch(Exception e){
    log.error("An error occurred on deleting ontology.", e);
    return false;
}

return true;
}

private void deleteOntologyFile (String name){
try{
    if ( name != null && !name.isEmpty() ){
        File f = new File(name);
        if (f.exists()){
            f.delete();
        }
    }
}catch(Exception ex){
    log.error("Error occurred during deleteOntologyFile", ex);
}
}

@RequestMapping(value = "/ontologies/{ontologyId}/download",
method = RequestMethod.GET)
public void downloadOwl(@PathVariable int ontologyId, final
HttpServletRequest request, final HttpServletResponse response){
    log.info("owl id : {" + ontologyId);
    Ontology ontology =
ontologyServiceProvider.getOntology(ontologyId);
    try{
        String name = ontology.getOntologyName();
        String fullName= getOntologyPath(name);
        log.info( "downloading ontology : " + fullName);
        File file = new File(fullName);

        log.trace("Write OWL file in response");
        try (InputStream fileInputStream = new
FileInputStream(file);
              OutputStream output =
response.getOutputStream();) {

            response.reset();

            response.setContentType("application/octet-stream");
        }
    }
}
}

```

```
        response.setContentLength((int) (file.length()));

        response.setHeader("Content-Disposition",
"attachment; filename=\"" + file.getName() + "\"");

        IOUtils.copyLarge(fileInputStream, output);
        output.flush();
    } catch (IOException e) {
        log.error(e.getMessage(), e);
    }
} catch(Exception e){
    log.error(e.getMessage(), e);
}

}

@ResponseStatus(HttpStatus.OK)
@RequestMapping(value = "/ontologies/{ontologyId}/download-
processed", method = RequestMethod.GET)
public void downloadProcessedOwl(@PathVariable int ontologyId,
final HttpServletRequest request, final HttpServletResponse
response){
    log.info("download owl id : " + ontologyId);
    Ontology ontology =
ontologyServiceProvider.getOntology(ontologyId);
    try{

        if ( ontology.getStatus() ==
OntologyStatus.Processed.ordinal() ){
            String name = ontology.getOntologyProcessedName();
            File file = new File(getOntologyPath(name));
            log.trace("Write OWL file in response");
            try (InputStream fileInputStream = new
FileInputStream(file);
                  OutputStream output =
response.getOutputStream();) {

                response.reset();

                response.setContentType("application/octet-
stream");
                response.setContentLength((int)
(file.length()));
            }
        }
    }
}
```

```

        response.setHeader("Content-Disposition",
"attachment; filename=\"" + file.getName() + "\"");

        IOUtils.copyLarge(fileInputStream, output);
        output.flush();
    } catch (IOException e) {
        log.error(e.getMessage(), e);
    }
}

} catch(Exception e){
    log.error(e.getMessage(), e);
}

}

@ResponseStatus(HttpStatus.OK)
@RequestMapping(value = "/ontologies/{ontologyId}/download-
processed-check", method = RequestMethod.GET)
public boolean downloadProcessedOwlcheck(@PathVariable int
ontologyId, final HttpServletRequest request, final
HttpServletResponse response){
    Ontology ontology =
ontologyServiceProvider.getOntology(ontologyId);
    return ontology != null && ontology.getStatus() ==
OntologyStatus.Processed.ordinal();
}

private String getOntologyPath(String name){
    String path="/home/salvatore/wonts/ontology_uploaded";
    String configpath =
PropertiesManager.getProperty(PropertiesManager.ONTOLOGY_UPLOADED );
    return (configpath==null ? path: configpath ) + File.separator
+name;
}

}

```

8.11.2.2 Routes.java

```

package wonts.webapp.controllers;

import org.springframework.stereotype.Controller;

```

```
import org.springframework.web.bind.annotation.RequestMapping;

/**
 * Maps all AngularJS routes to index so that they work with direct
linking.
 */
@Controller
public class Routes {
    @RequestMapping({
        "/",
        "/home",
        "/login",
        "/signin",
        "/projects",
        "/projects/{id:\w+}",
        "/projects/new"
    })
    public String index() {
        return "forward:/resources/app/views/index.html";
    }
}
```

8.11.2.3 UserController.java

```
package wonts.webapp.controllers;

import java.security.Principal;

import org.apache.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotationResponseStatus;

import wonts.data.services.model.User;
import wonts.data.services.provider.UserServiceProvider;
import wonts.webapp.dto.AddNewUserDTO;
import wonts.webapp.dto.UserDTO;
```

```

/**
 * @author salvatore
 *
 * this class contains all the user API
 */
@Controller
@RequestMapping("/api/user")
public class UserController {
    private static final Logger logger =
Logger.getLogger(UserController.class);

    @Autowired
    UserServiceProvider userService;

    @ResponseBody
    @ResponseStatus(HttpStatus.OK)
    @RequestMapping(value = "new", produces="application/json",
method = RequestMethod.POST)
    public AddNewUserDTO addNewUser( @RequestBody AddNewUserDTO
user ){
        logger.info("add a new user! " + user.getUsername() );
        try {
            if ( userService.isUsernameAvailable(user.getUsername() ) )
{
                User u= userService.createUser(user.getUsername(),
user.getEmail(), user.getPassword(), null, null, user.getImage());
                if ( u != null && u.getId()> 0 ){
                    logger.info("user successfully added.");
                    user.setPassword("");
                    return user;
                }else{
                    logger.warn("add a new user fail to insert! ");
                    user.setPassword("");
                    user.setError(true);
                    user.setErrorMessage("Cannot be possible add a
new user.");
                    return user;
                }
            }else {
                user.setPassword("");

```

```
        user.setError(true);
        user.setErrorMessage("The user name already exist.");
        return user;

    }
} catch (Exception e) {
    logger.error("add a new user fail with exception! ", e);
    user.setPassword("");
    user.setError(true);
    user.setErrorMessage("An error occurred during the creation
of the new user.");
    return user;
}
}

@ResponseBody
@ResponseStatus(HttpStatus.OK)
@RequestMapping( value =
"/authenticated",produces="application/json", method =
RequestMethod.GET)
public boolean isUserAuthenticated(Principal principal){
    return principal != null;
}

@ResponseBody
@ResponseStatus(HttpStatus.OK)
@RequestMapping( value = "/user-
authenticated",produces="application/json", method =
RequestMethod.GET)
public UserDTO getUserNameAuthenticated(Principal principal){
    UserDTO ret = new UserDTO();
    if ( principal != null ){
        User user=
userService.getUserByUsername(principal.getName());
        ret = new UserDTO(user);
        ret.setAuthenticated(true);
    }
    return ret;
}
```

}

8.11.3 wonts.webapp.dto

8.11.3.1 AddNewUserDTO.java

```
package wonts.webapp.dto;

import lombok.Getter;
import lombok.Setter;
/**
 * Visual Model for adding a new user
 * @author salvatore
 *
 */
public class AddNewUserDTO extends UserDTO{
    @Getter @Setter
    String password;
    @Getter @Setter
    String email;
    @Getter @Setter
    boolean error;
    @Getter @Setter
    String errorMessage;
    @Getter @Setter
    String image;
}
```

8.11.3.2 OntologyDTO.java

```
package wonts.webapp.dto;

import java.util.Date;

import wonts.data.services.model.Ontology;
import lombok.Data;

@Data
public class OntologyDTO {
    int ontologyId;
    String name;
    String description;
    String ontologyProcessedName;
    boolean isActive;
    int status;
    Date created;
```

```
Date modified;  
  
public OntologyDTO(){  
  
public OntologyDTO(Ontology ontology){  
    this.ontologyId = ontology.getOntologyId();  
    this.name = ontology.getOntologyName();  
    this.description = ontology.getOntologyDescription();  
    this.created = ontology.getModified();  
    this.modified = ontology.getModified();  
    this.isActive = ontology.isActive();  
    this.status = ontology.getStatus();  
    this.ontologyProcessedName =  
ontology.getOntologyProcessedName();  
}  
  
}  

```

8.11.3.3 ProjectDTO.java

```
package wonts.webapp.dto;  
  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
  
import wonts.data.services.model.Project;  
import lombok.Data;
```

```
@Data  
public class ProjectDTO {  
    int projectId;  
    String projectName;  
    String projectDescription;  
    List<UrlDTO> urls;  
    List<OntologyDTO> ontologies;  
    boolean isComplete;  
    boolean isActive;  
    Date created;  
    Date modified;  
  
public ProjectDTO(){}
```

```
public ProjectDTO(Project project) {
    projectId = project.getProjectId();
    projectName = project.getProjectName();
    projectDescription = project.getProjectDescription();
    isComplete = project.isComplete();
    isActive = project.isActive();
    created = project.getCreated();
    modified = project.getModified();

    if (project.getUrls() != null) {
        urls = new ArrayList<UrlDTO>();
        project.getUrls().forEach(url -> { urls.add(new
UrlDTO(url)); });
    }

    if (project.getOntologies() != null) {
        ontologies = new ArrayList<OntologyDTO>();
        project.getOntologies().forEach(ontology -> {
ontologies.add(new OntologyDTO( ontology ));});
    }
}
```

8.11.3.4 UrlDTO.java

```
package wonts.webapp.dto;
```

```
import java.util.Date;

import wonts.data.services.model.Url;
import lombok.Data;

@Data
public class UrlDTO {
    int urlId;
    String urlValue;
    String urlDescription;
    boolean isActive;
    Date created;
```

```
Date modified;  
  
public UrlDTO(){  
  
public UrlDTO(Url url) {  
    urlId = url.getUrlId();  
    urlValue = url.getUrlValue();  
    urlDescription = url.getUrlDescription();  
    isActive = url.isActive();  
    created = url.getCreated();  
    modified = url.getModified();  
}  
  
}  
  
}
```

8.11.3.5 UserDTO.java

```
package wonts.webapp.dto;  
  
import lombok.Data;  
import wonts.data.services.model.User;  
  
/**  
 * Visual Model for User entity  
 * @author salvatore  
 *  
 */  
@Data  
public class UserDTO {  
    public UserDTO(){  
  
        public UserDTO(String username) {  
            this.username = username;  
        }  
        public UserDTO(User user) {  
            this.username = user.getUsername();  
            this.image = user.getImage();  
        }  
  
        private boolean isAuthenticated;  
        private String username;  
        private String image;  
    }  
}
```

8.11.4 wonts.webapp.property

```
8.11.4.1      PropertiesManager.java
package wonts.webapp.property;

import java.io.FileInputStream;
import java.util.Properties;

import org.apache.log4j.LogManager;
import org.apache.log4j.Logger;

public class PropertiesManager {
    private static final Logger log =
LogManager.getLogger(PropertiesManager.class);

    public static final String DB = "persistence.db";
    public static final String DB_URL = "persistence.db.url";
    public static final String DB_USER = "persistence.db.urs";
    public static final String DB_PWD="persistence.db.pwd";
    public static final String CLASS_NAME=
"persistence.className";
    public static final String ONTOLOGY_UPLOADED =
"wontsapp.ontology.uploaded";

    public static final String PROPERTY_FILENAME=
"resources/wontsapp.properties";

    static Properties properties;

    /*
    public PropertiesManager(){
        this(PROPERTY_FILENAME);
    }
    */

    static {
        properties = GetProviderProperty(PROPERTY_FILENAME);
    }
}
```

```
public static Properties GetProviderProperty(String fileName){
    Properties ret= new Properties();
    try {
        ret.load(new FileInputStream(fileName));
    } catch (Exception e) {
        log.error(e);
    }
    return ret;
}

public static String getProperty(String key){
    return properties.getProperty(key);
}

}
```

8.11.5 wonts.webapp.security

```
8.11.5.1      AjaxAuthenticationFailureHandler.java
package wonts.webapp.security;

import java.io.IOException;

import javax.inject.Named;

import org.springframework.security.core.AuthenticationException;
import
org.springframework.security.web.authentication.SimpleUrlAuthenticatio
nFailureHandler;

/**
 * Authentication failure handler for Single Page Applications that
 need to login using Ajax instead
 */
@Named("authenticationFailureHandler")
public class AjaxAuthenticationFailureHandler extends
SimpleUrlAuthenticationFailureHandler {

    /* (non-Javadoc)
     * @see
org.springframework.security.web.authentication.SimpleUrlAuthenticatio
nFailureHandler#onAuthenticationFailure(javax.servlet.http.HttpServlet
```

```
Request, javax.servlet.http.HttpServletResponse,
org.springframework.security.core.AuthenticationException)
*/
public void
onAuthenticationFailure(javax.servlet.http.HttpServletRequest request,
                      javax.servlet.http.HttpServletResponse response,
AuthenticationException exception)
{

    try {
        response.getWriter().print("ko");
        response.getWriter().flush();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}
```

8.11.5.2 AjaxAuthenticationSuccessHandler.java

```
package wonts.webapp.security;

import java.io.IOException;

import javax.inject.Named;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.security.core.Authentication;
import
org.springframework.security.web.authentication.SimpleUrlAuthenticatio
nSuccessHandler;

/**
 * Authentication success handler for Single Page Applications that
need to login using Ajax
*/
@Named("authenticationSuccessHandler")
public class AjaxAuthenticationSuccessHandler extends
SimpleUrlAuthenticationSuccessHandler {
```

```
/* (non-Javadoc)
 * @see
org.springframework.security.web.authentication.SimpleUrlAuthenticationSuccessHandler#onAuthenticationSuccess(javax.servlet.http.HttpServletRequest, javax.servlet.http.HttpServletResponse,
org.springframework.security.core.Authentication)
 */
public void onAuthenticationSuccess(HttpServletRequest request,
HttpServletResponse response, Authentication auth)
throws IOException, ServletException {

    response.getWriter().print("ok");
    response.getWriter().flush();

}
}
```

8.11.5.3 SecurityUserDetailsService.java

```
package wonts.webapp.security;
```

```
import java.util.ArrayList;
import java.util.List;

import org.apache.log4j.Logger;
//import org.apache.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.data.jpa.repository.config.EnableJpaRepositories;
import org.springframework.security.core.GrantedAuthority;
import
org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
import
org.springframework.security.core.userdetails.UserDetailsService;
import
org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Repository;

import wonts.data.services.model.User;
import wonts.data.services.provider.UserServiceProvider;
```

```

/**
 * UserDetails service that reads the user credentials from the
database, using a JPA repository.
 *
 */
@Repository
@EnableJpaRepositories
public class SecurityUserDetailsService implements
UserDetailsService {
    /* (non-Javadoc)
     */
    private static final Logger log =
Logger.getLogger(SecurityUserDetailsService.class);

    @Autowired
    UserServiceProvider userService;

    /* (non-Javadoc)
     * @see
org.springframework.security.core.userdetails.UserDetailsService#loadU
serByUsername(java.lang.String)
     */
    @Override
    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        log.info( "Username: " + username);

        User user = null;
        try{
            user = userService.getUserByUsername(username);
        }catch(Exception ex){
            log.error(ex);
        }

        if (user == null) {
            String message = "Username not found: " + username;
            log.info(message);
            throw new UsernameNotFoundException(message);
        }
        List<GrantedAuthority> authorities = new ArrayList<>();
        authorities.add(new SimpleGrantedAuthority("ROLE_USER"));

        log.info("Found user in database: " + user);
    }
}

```

```
org.springframework.security.core.userdetails.User u =
    new
org.springframework.security.core.userdetails.User(username,
user.getPassword(), authorities);
    return u;
}
}
```

8.11.5.4 *WebSecurityConfig.java*

```
package wonts.webapp.security;

import javax.inject.Inject;
import javax.inject.Named;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import
org.springframework.data.jpa.repository.config.EnableJpaRepositories;
import
org.springframework.security.config.annotation.authentication.builders
.AuthenticationManagerBuilder;
import
org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSe
curityConfigurerAdapter;
import
org.springframework.security.config.annotation.web.servlet.configurati
on.EnableWebMvcSecurity;
import
org.springframework.security.core.userdetails.UserDetailsService;
import
org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import
org.springframework.security.web.authentication.AuthenticationFailureH
andler;
import
org.springframework.security.web.authentication.AuthenticationSuccessH
andler;
```

```
@Configuration
@EnableWebMvcSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
    @Autowired
    UserDetailsService userDetailsService;

    @Inject @Named("authenticationSuccessHandler")
    AuthenticationSuccessHandler successHandler;

    @Inject @Named("authenticationFailureHandler")
    AuthenticationFailureHandler authenticationFailureHandler;

    @Override
    protected void configure(HttpSecurity http) throws
Exception {
        http
            .authorizeRequests()
                .antMatchers("/", "/home", "/api/**",
"/resources/**", "/authenticate").permitAll()
                .anyRequest().authenticated()
                .and()
            .formLogin()
                .passwordParameter("password")
                .usernameParameter("username")
                .loginPage("/#login")
                .loginProcessingUrl("/authenticate")
                .successHandler(successHandler)
                .failureHandler(authenticationFailureHandler)
                .permitAll()
                .and()
            .logout()
                .logoutUrl("/logout")
                .permitAll()
                .and()
            .csrf().disable();
    }

    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder
auth) throws Exception {

```

```
    PasswordEncoder passwordEncoder = new
BCryptPasswordEncoder();
```

```
        auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder);
    }
}
```

8.12 webapp (src/main/test)

8.12.1 wonts.webapp

8.12.2 WontsWebAppApplicationTests.java

```
package wonts.webapp;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.test.context.web.WebAppConfiguration;
import org.springframework.boot.test.SpringApplicationConfiguration;
import
org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

import wonts.webapp.WontsWebAppApplication;

@RunWith(SpringJUnit4ClassRunner.class)
@SpringApplicationConfiguration(classes =
WontsWebAppApplication.class)
@WebAppConfiguration
public class WontsWebAppApplicationTests {

    @Test
    public void contextLoads() {

    }
}
```

8.13 webapp (resources)

8.13.1 wontsapp.properties

```
wontsapp.ontology.uploaded=/Users/salvo/Development/wonts/ontology_uploaded
```

8.14 webapp (src/main/resources)

8.14.1 application.properties

```
spring.jpa.database=MYSQL
spring.datasource.url=jdbc:mysql://localhost:3306/wontsapp
spring.datasource.username=wontsuser
spring.datasource.password=user$wonts
spring.datasource.testOnBorrow=true
spring.datasource.validationQuery=SELECT 1

spring.jpa.hibernate.ddl-auto=update
spring.datasource.driverClassName=com.mysql.jdbc.Driver
spring.jpa.hibernate.naming-
strategy=org.hibernate.cfg.EJB3NamingStrategy

logging.level.org.springframework.web=INFO
logging.level.org.hibernate=INFO
logging.file=wontswebapp.out
```

8.15 webapp (src/webapp/resources)

8.15.1 app

8.15.1.1 common

```
8.15.1.1 common.js
(function () {
    'use strict';

    // Define the common module
    // Contains services:
    // - common
    // - logger
    // - spinner
    var commonModule = angular.module('common', []);

    // Must configure the common service and set its
    // events via the commonConfigProvider
    commonModule.provider('commonConfig', function () {
        this.config = {
            // These are the properties we need to set
            //controllerActivateSuccessEvent: '',
            //spinnerToggleEvent: ''
        };
    });
});
```

```

        this.$get = function () {
            return {
                config: this.config
            };
        };
    });

commonModule.factory('common',
    ['$q', '$rootScope', '$timeout', 'commonConfig', 'logger',
'$http', common]);

function common($q, $rootScope, $timeout, commonConfig, logger,
$http) {
    var throttles = {};

    var service = {
        // common angular dependencies
        $broadcast: $broadcast,
        $q: $q,
        $timeout: $timeout,
        // generic
        activateController: activateController,
        createSearchThrottle: createSearchThrottle,
        debouncedThrottle: debouncedThrottle,
        isNumber: isNumber,
        logger: logger, // for accessibility
        $http: $http,
        textContains: textContains,
        getDateAsString: getDateAsString
    };

    return service;

    function activateController(promises, controllerId) {
        return $q.all(promises).then(function (eventArgs) {
            var data = { controllerId: controllerId };

            $broadcast(commonConfig.config.controllerActivateSuccessEvent, data);
        });
    }

    function $broadcast() {
        return $rootScope.$broadcast.apply($rootScope, arguments);
    }
}

```

```

function createSearchThrottle(viewmodel, list, filteredList,
filter, delay) {
    // After a delay, search a viewmodel's list using
    // a filter function, and return a filteredList.

    // custom delay or use default
    delay = +delay || 300;
    // if only vm and list parameters were passed, set others
    by naming convention
    if (!filteredList) {
        // assuming list is named sessions, filteredList is
        filteredSessions
        filteredList = 'filtered' + list[0].toUpperCase() +
list.substr(1).toLowerCase(); // string
        // filter function is named sessionFilter
        filter = list + 'Filter'; // function in string form
    }

    // create the filtering function we will call from here
    var filterFn = function () {
        // translates to ...
        // vm.filteredSessions
        //      = vm.sessions.filter(function(item{ returns
        vm.sessionFilter (item} );
        viewmodel[filteredList] =
viewmodel[list].filter(function(item) {
            return viewmodel[filter](item);
        });
    };

    return (function () {
        // Wrapped in outer IFFE so we can use closure
        // over filterInputTimeout which references the
        timeout
        var filterInputTimeout;

        // return what becomes the 'applyFilter' function in
        the controller
        return function(searchNow) {
            if (filterInputTimeout) {
                $timeout.cancel(filterInputTimeout);
                filterInputTimeout = null;
            }
        }
    });
}

```

```

        if (searchNow || !delay) {
            filterFn();
        } else {
            filterInputTimeout = $timeout(filterFn,
delay);
        }
    };
})();
}

function debouncedThrottle(key, callback, delay, immediate) {
    // Perform some action (callback) after a delay.
    // Track the callback by key, so if the same callback
    // is issued again, restart the delay.

    var defaultDelay = 1000;
    delay = delay || defaultDelay;
    if (throttles[key]) {
        $timeout.cancel(throttles[key]);
        throttles[key] = undefined;
    }
    if (immediate) {
        callback();
    } else {
        throttles[key] = $timeout(callback, delay);
    }
}

function isNumber(val) {
    // negative or positive
    return /^[ -]?[0-9]+(\.[0-9]+)?$/i.test(val);
}

function textContains(text, searchText) {
    return text && -1 !==
text.toLowerCase().indexOf(searchText.toLowerCase());
}

function getDateAsString(value){
    if ( value != null){
        var d = new Date(value);
        return d.toLocaleString();
    }
}

```

```
        return "n.n.";
    }

}

})();

8.15.1.1.2 logger.js
(function () {
    'use strict';

angular.module('common').factory('logger', ['$log', logger]);

function logger($log) {
    var service = {
        getLogFn: getLogFn,
        log: log,
        logError: logError,
        logSuccess: logSuccess,
        logWarning: logWarning
    };

    return service;

    function getLogFn(moduleId, fnName) {
        fnName = fnName || 'log';
        switch (fnName.toLowerCase()) { // convert aliases
            case 'success':
                fnName = 'logSuccess'; break;
            case 'error':
                fnName = 'logError'; break;
            case 'warn':
                fnName = 'logWarning'; break;
            case 'warning':
                fnName = 'logWarning'; break;
        }

        var logFn = service[fnName] || service.log;
        return function (msg, data, showToast) {
            logFn(msg, data, moduleId, (showToast === undefined)
? true : showToast);
        };
    }
}

function log(message, data, source, showToast) {
```

```
        logIt(message, data, source, showToast, 'info');
    }

    function logWarning(message, data, source, showToast) {
        logIt(message, data, source, showToast, 'warning');
    }

    function logSuccess(message, data, source, showToast) {
        logIt(message, data, source, showToast, 'success');
    }

    function logError(message, data, source, showToast) {
        logIt(message, data, source, showToast, 'error');
    }

    function logIt(message, data, source, showToast, toastType) {
        var write = (toastType === 'error') ? $log.error :
$log.log;
        source = source ? '[' + source + ']' : '';
        write(source, message, data);
        if (showToast) {
            if (toastType === 'error') {
                toastr.error(message);
            } else if (toastType === 'warning') {
                toastr.warning(message);
            } else if (toastType === 'success') {
                toastr.success(message);
            } else {
                toastr.info(message);
            }
        }
    }
}
})();
}
```

8.15.1.1.3 spinner.js

```
(function () {
    'use strict';

angular.module('common')
    .factory('spinner', ['common', 'commonConfig', spinner]);
```

```
function spinner(common, commonConfig) {
    var service = {
        spinnerHide: spinnerHide,
        spinnerShow: spinnerShow
    };

    return service;

    function spinnerHide() { spinnerToggle(false); }

    function spinnerShow() { spinnerToggle(true); }

    function spinnerToggle(show) {
        common.$broadcast(commonConfig.config.spinnerToggleEvent,
{ show: show });
    }
}
});
```

8.15.1.2 controllers

8.15.1.2.1 homeCtrl.js

```
angular.module('wonts')
    .controller('homeCtrl',['$rootScope','$scope', '$http',
'$location', '$log', 'datacontext','common',
    function($rootScope,$scope, $http, $location, $log,
datacontext, common) {
        $scope.title = "Wonts";
        $scope.isUserAuthenticated=$rootScope.authenticated;
        var options={};
    }
]);
```

8.15.1.2.2 loginCtrl.js

```
angular.module('wonts')
    .controller('loginCtrl',['$rootScope','$scope', '$http',
'$routeParams', '$location', '$window',
    function($rootScope, $scope, $http, $routeParams, $location,
>window) {
        $scope.bookId = $routeParams.bookId;
        $scope.title="Login";
        $scope.username="";
        $scope.password="";
    }
});
```

```
$scope.errorMessage = 'Problems during the
authentication occurred.';
$scope.alerts = [];

$scope.addAlert = function(message) {
    $scope.alerts.push({msg: message});
};

$scope.closeAlert = function(index) {
    $scope.alerts.splice(index, 1);
};

$scope.preparepostData = function () {
    return 'username=' + $scope.username+
'&password=' + $scope.password;
}

$scope.login = function (username, password) {
    var postData = $scope.preparepostData();

    $http({
        method: 'POST',
        url: 'authenticate',
        data: postData,
        headers: {
            "Content-Type": "application/x-www-
form-urlencoded",
            "X-Login-Ajax-call": 'true'
        }
    })
    .then(function(response) {
        console.log(response.data) ;
        console.log($scope.username);
        if (response.data && response.data=="ok"){
//response.data.username == $scope.username) {
            $rootScope.authenticated = true;
            $rootScope.userAuthenticated =
$scope.username;
            $location.path("home")
            console.log("OK");
        }
        else {
            $rootScope.authenticated = false;
            $rootScope.userAuthenticated = "";
        }
    })
}
```

```
        $scope.errorMessage = "The user or password  
is incorrect";  
        console.log("Fail");  
    }  
    $scope.error = ! $rootScope.authenticated;  
});  
}  
]);
```

8.15.1.2.3 navigationCtrl.js

```
angular.module('wonts')  
    .controller('loginCtrl', ['$rootScope', '$scope', '$http',  
    '$routeParams', '$location', '$window',  
    function($rootScope, $scope, $http, $routeParams, $location,  
$window) {  
        $scope.bookId = $routeParams.bookId;  
        $scope.title="Login";  
        $scope.username='';  
        $scope.password='';  
        $scope.errorMessage = 'Problems during the  
authentication occurred.';  
        $scope.alerts = [];  
  
        $scope.addAlert = function(message) {  
            $scope.alerts.push({msg: message});  
        };  
  
        $scope.closeAlert = function(index) {  
            $scope.alerts.splice(index, 1);  
        };  
  
        $scope.preparepostData = function () {  
            return 'username=' + $scope.username+  
'&password=' + $scope.password;  
        }  
  
        $scope.login = function (username, password) {  
            var postData = $scope.preparepostData();
```

```

$http({
    method: 'POST',
    url: 'authenticate',
    data: postData,
    headers: {
        "Content-Type": "application/x-www-
form-urlencoded",
        "X-Login-Ajax-call": 'true'
    }
})
.then(function(response) {
    console.log(response.data) ;
    console.log($scope.username);
    if (response.data && response.data=="ok"){
//response.data.username == $scope.username) {
        $rootScope.authenticated = true;
        $rootScope.userAuthenticated =
$scope.username;
        $location.path("home")
        console.log("OK");
    }
    else {
        $rootScope.authenticated = false;
        $rootScope.userAuthenticated = "";
        $scope.errorMessage = "The user or password
is incorrect";
        console.log("Fail");
    }
    $scope.error = ! $rootScope.authenticated;
});
}
]);

```

```

8.15.1.2.4 navigationCtrl.js
angular.module('wonts')
.controller('navigationCtrl',['$rootScope','$scope', '$http',
'$routeParams', '$location', '$window','datacontext',
function($rootScope, $scope, $http, $routeParams, $location,
$window, datacontext) {

    $rootScope.authenticated=false;

```

```

$scope.logout = function() {
    $http.post('logout', {}).success(function() {
        $rootScope.authenticated = false;
        $location.path("/#books");
    }).error(function(data) {
        $rootScope.authenticated = false;
    });
}

$scope.checkUserAuthenticated = function (){
    datacontext.getUserAuthenticated().then(function (
authPromise){
    var user = authPromise.data;
    $rootScope.authenticated = user.authenticated;
    $rootScope.userAuthenticated = user.username;
    console.log( "User Authenticated: " +
$rootScope.authenticated);
    /* if ( $rootScope.authenticated ) {

        datacontext.getUserNameAuthenticated().then(function (
authPromise){
            $rootScope.userAuthenticated =
authPromise.data;
            console.log( "User
Authenticated: " + $rootScope.userAuthenticated);
        });
    }*/
});
}

/*
$scope.checkUserAuthenticated = function (){
    datacontext.getUserNameAuthenticated().then(function (
authPromise){
        $rootScope.authenticated = authPromise.data !=
null && authPromise.data != "";
        if ($rootScope.authenticated ){
            $rootScope.userAuthenticated =
authPromise.data;

        }
    console.log( "User Authenticated: " +

```

```
$rootScope.userAuthenticated);
    });
}

}*/



$scope.checkUserAuthenticated();
}]);
```

8.15.1.2.5 projectDetailCtrl.js

```
angular.module('wonts')
    .controller('projectDetailCtrl', ['$rootScope', '$scope', '$http',
    '$routeParams', '$location', '$log', '$window',
    'datacontext', 'common', 'Upload', 'downloadOwlService',
        function($rootScope, $scope, $http, $routeParams, $location,
    $log, $window, datacontext, common, Upload, downloadOwlService) {
        $scope.title = "Projects";
        $scope.project = {};
        $scope.projectId = $routeParams.projectId;
        $scope.isUserAuthenticated=$rootScope.authenticated;
        $scope.oneAtATime=false;
        $scope.progressPercentage=0;
        $scope.errorMessage ="";
        $scope.urlValue = { text: ""};
        $scope.alerts = [];
        $scope.urlAlerts = [];

        $scope.getProject = function(){
            datacontext.getProject($scope.projectId).then(
function ( promise){
            $scope.project = promise.data;
            $log.info($scope.project);
        });
    }
    $scope.getProject();

    $scope.getDateAsString = function (val){
        return common.getDateAsString(val);
    };

    // redirect in case the user is not authenticate
    if (! $rootScope.authenticated ){
        $location.path("login");
    }
}
```

```

$scope.status = {
    isFirstOpen: true,
    isFirstDisabled: false,
    isOpen: true,
    openOwl:true
};

$scope.uploadOwl = function (files) {
    $log.info(files);
    $scope.formUpload = true;
    if (files != null) {
        $scope.upload(files)
    }
};

$scope.addNewUrl = function () {
    var newUrl = {
        urlValue: $scope.urlValue.text,
        urlDescription: $scope.urlValue.text
    };
    $log.info(newUrl);
    datacontext.addNewUrlToProject( $scope.projectId,
newUrl ).then(
        function(promise) {
            if ( promise.data ){
                // reload
                $scope.urlValue.text="";
                $scope.getProject();
                $scope.errorMessage="";
                $scope.urlAlerts = [];
                $scope.form.$setPristine();
                $scope.getProject();
            }else {
                var message ="Some problem
occured during inserting the url!";
                // message error
                $scope.urlAlerts.push({msg:
message});
            }
        }
);
}

```

```

        });
    }

$scope.getDateAsString = function (val){
    return common.getDateAsString(val);
};

$scope.addAlert = function(message) {
    $scope.alerts.push({msg: message});
};

$scope.closeUrlAlert = function(index) {
    $scope.urlAlerts.splice(index, 1);
};

$scope.closeAlert = function(index) {
    $scope.alerts.splice(index, 1);
};

$scope.clearAlerts= function(){
    $scope.alerts = [];
}

$scope.changeStatusUrl = function (urlId, isActive){
    $log.info("urlId:" + urlId + " isActive: " +
isActive );
    datacontext.setUrlStatus(urlId, isActive).then(
        function(promise) {
            if ( promise.data){
                $scope.getProject();
            }
        });
}

$scope.upload = function (files) {
    $log.info($scope.owlName);
    if (files && files.length) {
        for (var i = 0; i < files.length; i++) {
            var file = files[i];
            Upload.upload({
                url: 'api/projects/ontology-upload/' +
$scope.projectId,
                fields: {'name': $scope.owlName},
            })
        }
    }
}

```

```

        file: file
    }).progress(function (evt) {
        $scope.progressPercentage =
parseInt(100.0 * evt.loaded / evt.total);
        $log.info('progress: ' +
$scope.progressPercentage + '%' + evt.config.file.name);
    }).success(function (data, status,
headers, config) {
        $log.info('file ' + config.file.name +
'uploaded. Response: ' + data);
        $scope.owlName="";
        angular.forEach(
            angular.element("input[type='file']"),
            function(inputElem) {
                angular.element(inputElem).val(null);
            });
        $scope.progressPercentage = 0;
        $scope.getProject();
    }).error(function (data, status, headers,
config) {
        $log.error('error status: ' + status);
    })
});
}
};

$scope.download = function(owlId ) {
    downloadOwlService.download(owlId,
'api/projects/ontologies/'+owlId+'/download')
    .then(function(success) {
        console.log('success : ' + success);
    }, function(error) {
        console.log('error : ' + error);
    });
};

$scope.downloadProcessed = function(owlId ) {
    downloadOwlService.download(owlId,
'api/projects/ontologies/'+owlId+'/download-processed')
    .then(function(success) {
        console.log('success : ' + success);
    }, function(error) {

```

```
        console.log('error : ' + error);
    });
};

$scope.deleteOwl = function( owlId){
    if ( ! $window.confirm('Are you sure you want to delete
the ontology?' ) ) return;
    datacontext.deleteOwl(owlId).then(function(promise){
        if ( promise.data){
            $scope.getProject();
        }
    });
}

}]);
```

```
8.15.1.2.6 projectListCtrl.js
angular.module('wonts')
    .controller('projectListCtrl',[ '$rootScope', '$scope', '$http',
'$location', '$log', 'datacontext','common',
    function($rootScope, $scope, $http, $location, $log,
datacontext, common) {
        $scope.title = "Projects";
        $scope.isUserAuthenticated=$rootScope.authenticated;
        $scope.projects;
        $scope.filteredProjects = [];
        $scope.currentPage = 1;
        $scope.numPerPage = 10;
        $scope.maxSize = 5;
        var options={};

        // redirect in case the user is not authenticate
        if (! $rootScope.authenticated ){
            $location.path("login");
        }

        datacontext.getProjects().then(
            function (promise){
                $scope.projects = promise.data;
                $log.info($scope.projects);
```

```
$scope.$watch('currentPage + numPerPage',
function() {
    var begin = (($scope.currentPage - 1) *
$scope.numPerPage)
        , end = begin + $scope.numPerPage;

    $scope.filteredProjects =
$scope.projects.slice(begin, end);
}
);

$scope.addNewProject = function (){
    $location.path("new-project");
}

$scope.numPages = function () {
    return Math.ceil($scope.projects.length /
$scope.numPerPage);
};

$scope.getDateAsString = function (val){
    return common.getDateAsString(val);
};

$scope.setPage = function (pageNo) {
    $scope.currentPage = pageNo;
};

$scope.pageChanged = function() {
    console.log('Page changed to: ' +
$scope.currentPage);
};

});

8.15.1.2.7 projectNewCtrl.js
angular.module('wonts')
```

```

.controller('projectNewCtrl',['$scope', '$http', '$log',
'$location', 'datacontext',
  function($scope, $http, $log, $location, datacontext) {
    $scope.name = "";
    $scope.description="";

    $scope.addNewProject = function (){
      var newProject = {
        projectName: $scope.name,
        projectDescription:
$scope.description,
      };

      datacontext.addNewProject(newProject).then(
        function(bookPromise){
          var project = bookPromise.data;
          if (project.projectId > 0){
            $log.info("the project has
been added.");
          }

          $location.path('projects/' + project.projectId);
          $scope.error=false;
        }else {
          $log.info("problems has
occured to add a new project.");
          $scope.error=true;
        }
      },
      function(error){
        $scope.error=true;
      }
    );
  };
}]);

```

8.15.1.2.8 signinCtrl.js

```

angular.module('wonts')
  .controller('signinCtrl',['$scope', '$http', '$routeParams',
'$location', '$timeout', 'datacontext',
  function($scope, $http, $routeParams, $location, $timeout,
datacontext) {
    $scope.bookId = $routeParams.bookId;

```

```

$scope.title="Sign In";

$scope.confirm_password="";
$scope.password="";
$scope.username="";
$scope.email="";
$scope.userCreated = false;

$scope.pwmatch = new function (){
    return ! $scope.password ==
$scope.confirm_password;
}

$scope.addNewUser = function (){
    var newUser = {
        username: $scope.username,
        email: $scope.email,
        password: $scope.password,
        image: 'face0.jpg'
    }

    datacontext.addNewUser( newUser ).then(
        function (newUserPromise){
            newUser = newUserPromise.data;
            if ( newUser.error ){
                $scope.error = true;
                $scope.errorMessage =
$scope.errorMessage;
            }else {
                $scope.userCreated = true;
                $location.path("login");
            }
        },
        function (reason){
            $scope.errorMessage = "An error
occured during the creation of the user. Please try later.";
        }
    );
}
}

```

]);

8.15.1.3 services

8.15.1.3.1 datacontext.js

```
(function () {
    'use strict';

    var serviceId = 'datacontext';
    angular.module('wonts').factory(serviceId, ['$http', '$q', '$log',
datacontext]);

    function datacontext($http, $q, $log) {

        //      var getLogFn = config.logger.getLogFn;
        //      var log = getLogFn(serviceId);
        //      var logError = getLogFn(serviceId, 'error');
        //      var logSuccess =getLogFn(serviceId, 'success');

        var service = {
            isUserAuthenticated: isUserAuthenticated,
            getUserAuthenticated: getUserAuthenticated,
            addNewUser: addNewUser,
            getProjects: getProjects,
            addNewProject: addNewProject,
            getProject: getProject,
            addNewUrlToProject: addNewUrlToProject,
            setUrlStatus: setUrlStatus,
            deleteOwl: deleteOwl
        };

        return service;

        function isUserAuthenticated(){
            return $http.get("api/user/authenticated");
        }

        function getUserAuthenticated(){
            return $http.get("api/user/user-authenticated");
        }
    }
})();
```

```
function addNewUser(user ){
    $log.info('adding a new user:');
    $log.info(user);
    return $http.post( "api/user/new", user);
}

function getProjects(){
    return $http.get( "api/projects");
}

function addNewProject(newProject){
    return $http.post( "api/projects/new", newProject);
}

function getProject(projectId){
    return $http.get( "api/projects/" + projectId);
}

function addNewUrlToProject(projectId, newUrl){
    $log.info("adding new url: " + newUrl);
    var loc = "api/projects/" + projectId + "/urls/new";
    return $http.post(loc, newUrl);
}

function setUrlStatus(urlId, isActive){
    return $http.post("api/projects/urls/"+urlId+"/set-status",
isActive?1:2);
}

function deleteOwl(owlId){
    return
$http['delete']("api/projects/ontologies/" +owlId+ "/delete/");
}

}

})();
```

8.15.1.3.2 download.service.js

```
(function() {
    'use strict';
```

```
var downloadOwlModule = angular.module('owl.download', []);

downloadOwlModule.factory('downloadOwlService', ['$q', '$timeout',
'$window',
function($q, $timeout, $window) {
    return {
        download: function(owlId, location) {
            var defer = $q.defer();

            $timeout(function() {
                $window.location = location;
            }, 1000)
            .then(function() {
                defer.resolve('success');
            }, function() {
                defer.reject('error');
            });
        }
    }
}]);
})();
});
```

8.15.1.4 views

8.15.1.4.1 account

8.15.1.4.1.1 login.html

```
<div class="alert alert-danger" ng-show="error">
    {{errorMessage}}
</div>
<form name="form" role="form" ng-submit="login()">
    <div class="form-group">
        <label for="username">Username:</label><input type="text"
            class="form-control" id="username" name="username"
            placeholder="User name" ng-model="username" required ng-
            minlength="6"/>
    </div>
    <div class="form-group">
        <label for="password">Password:</label> <input
            type="password"
            class="form-control" id="password" name="password" ng-
            model="password">
    </div>
```

```

        placeholder="Password" required ng-minlength="6"
        pattern="(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{6,}"
        ng-focus="focus('password')" ng-
blur="blur('password')"/>

        <!-- <div class="error-messages" ng-
messages="form.password.$error" ng-cloak>
            <div ng-message="required">The password is
mandatory</div>
            <div ng-message="minlength">must have minimum 6
characters</div>
            <div ng-message="pattern">At least one number and
uppercase</div>
        </div> -->
    </div>
    <button type="submit" class="btn btn-primary">Login</button>
</form>
```

8.15.1.4.1.2 signin.html

```

<div class="alert alert-danger" ng-show="error">
    There was a problem logging in. Please try again.
</div>
<alert ng-show="userCreated" type="success"
close="closeAlert($index)">The user has been created.</alert>
<form ng-show="!userCreated" name="formSignIn" role="form"
class="form-horizontal" ng-submit="createUser()" class="pure-form
login-form" novalidate ng-controller="signinCtrl">
    <fieldset class="pure-group">
        <legend>Sign In as new user</legend>

        <div class="form-group">
            <label class="col-xs-2 control-label"
for="username">Username:</label>
            <div class="col-xs-10">
                <input ng-focus="" type="text" class="form-
control" id="username" name="username" placeholder="User name" ng-
model="username" required ng-minlength="6"/>
                <span class="error" ng-
show="formSignIn.username.$error.required &&
formSignIn.username.$dirty">The username is mandatory</span>
                <span class="error" ng-
show="formSignIn.username.$error.minlength &&
```

```

formSignIn.username.$dirty">must have minimum 6 characters</span>
    </div>
    </div>

        <div class="form-group">
            <label class="col-xs-2 control-label"
for="email">Email:</label>
            <div class="col-xs-10">
                <input type="email" class="form-control"
id="email" name="email" placeholder="Email" ng-model="email"
pattern="[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,4}$" required ng-
minlength="6"/>
                <span class="error" ng-
show="formSignIn.email.$error.required && formSignIn.email.$dirty">The
email is mandatory</span>
                <span class="error" ng-
show="formSignIn.email.$error.email && formSignIn.email.$dirty">must
be a valid email</span>
                <span class="error" ng-
show="formSignIn.email.$error.pattern && formSignIn.email.$dirty">
(es: rapisarda.salvatore@gmail.com)</span>
            </div>

        </div>

        <div class="form-group">
            <label class="col-xs-2 control-label"
for="password">Password:</label>
            <div class="col-xs-10">
                <input type="password"
                    class="form-control" id="password"
name="password" ng-model="password"
                    placeholder="Password" required ng-
minlength="6"
                    pattern="(?=.*\d)(?=.*[a-z])(?=.*[A-
Z]).{6,}">
                <span class="error" ng-
show="formSignIn.password.$error.required &&
formSignIn.password.$dirty">The password is mandatory</span>
                <span class="error" ng-
show="formSignIn.password.$error.minLength &&
formSignIn.password.$dirty">must have minimum 6 characters</span>
                <span class="error" ng-
show="formSignIn.password.$error.pattern &&

```

```

formSignIn.password.$dirty">an upper-case char and a number at least
(es: Password1)</span>
    </div>

    </div>
    <div class="form-group">
        <label class="col-xs-2 control-label"
for="confirm_password">Confirm:</label>
        <div class="col-xs-10">
            <input type="password"
                class="form-control" id="confirm_password"
name="confirm_password" ng-model="confirm_password"
                placeholder="Confirm Password" required ng-
minlength="6"
                pattern="(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{6,}" />
            <span class="error" ng-
show="formSignIn.confirm_password.$error.required &&
formSignIn.confirm_password.$dirty">The password is mandatory</span>
            <span class="error" ng-
show="formSignIn.confirm_password.$error.minLength &&
formSignIn.confirm_password.$dirty">must have minimum 6
characters</span>
            <span class="error" ng-
show="formSignIn.confirm_password.$error.pattern &&
formSignIn.confirm_password.$dirty">an upper-case char and a number at
least (es: Password1)</span>
            <span class="error" ng-show="password !=
confirm_password && formSignIn.confirm_password.$dirty">the password
should match</span>

        </div>
    </div>
</fieldset>
    <hr>
    <div class="form-group">
        <button type="submit" class="btn btn-primary" ng-
disabled="password != confirm_password || formSignIn.$invalid" ng-
click="addNewUser()">Create User</button>
        <a class="new-user-link" href="#/login">Already a
user?</a>
    </div>

</form>

```

8.15.1.4.2 projects

8.15.1.4.2.1 project-detail.html

```
<script type="text/ng-template" id="authorDetail.html"
src="/bface/resources/app/templates/authorDetail.html" > </script>

<div class="container-fluid">

    <!-- Blog Post Content Column -->
    <div class="col-lg-8">

        <!-- Blog Post -->

        <!-- Title -->
        <h1>{{project.projectName}}</h1>

        <!-- Author -->
        <p class="lead">
            Project: <a
href="#/projects/{{project.projectId}}">{{project.projectDescription}}</a>
        </p>

        <hr>

        <!-- Date/Time -->
        <p><span class="glyphicon glyphicon-time"></span> Last
update posted on {{getDateAsString(project.modified)}}</p>

        <hr>
        <accordion close-others="oneAtATime">
            <accordion-group ng-init="status.open = true"
is-open="status.open">
                <accordion-heading>
                    <span class="glyphicon glyphicon-
link"></span> Urls <i class="pull-right glyphicon" ng-
class="{'glyphicon-chevron-down': status.open, 'glyphicon-chevron-
right': !status.open}"></i>
                </accordion-heading>
                <div class="row" ng-repeat="a in
project.urls" >
```

```

<div class="col-md-2" >
    <span class="glyphicon " ng-
class="{'glyphicon-ok': a.active, 'glyphicon-remove' : !a.active}"
tooltip="click to {{!a.active? 'enable': 'disable'}}" ng-
click="changeStatusUrl(a.urlId,!a.active)" >{{a.active? ' enabled': 'disabled' }}</span>
</div>
<div class="col-md-3" >
    <span class="glyphicon glyphicon-
glyphicon-time" aria-hidden="true" tooltip="created">
{{getDateAsString(a.created )}}</span>
</div>
<div class="col-md-7"> {{a.urlValue}}</div>
<hr />
</div>
<div class="well" ng-
show="isUserAuthenticated">
    <h4>Add new url to project:</h4>
    <form name="formComment" role="form">
        <div class="form-group">
            <input class="form-control"
type="url" name="url" ng-model="urlValue.text" rows="3"
required></input>
        </div>
        <div role="alert">
            <span class="error" ng-
show="formComment.url.$error.required && formComment.url.$dirty">
                Required!</span>
            <span class="error" ng-
show="formComment.url.$error.url && formComment.url.$dirty">
                Not valid url!</span>
        </div>
        <alert ng-repeat="alert in
urlAlerts" type="{{alert.type}}"
close="closeUrlAlert($index)">{{alert.msg}}</alert>
        <div >
            <br />
            <button type="submit" class="btn-
btn-primary glyphicon glyphicon-plus" ng-click="addNewUrl()" ng-
disabled="formComment.$invalid" > Add</button>
        </div>
    </form>
    <hr>
</div>

```

```

        </accordion-group>
        <accordion-group ng-init="status.open = true"
is-open="status.openOwl">
            <accordion-heading>
                <span class="glyphicon glyphicon-
file"></span> OWL File <i class="pull-right glyphicon" ng-
class="{ 'glyphicon-chevron-down': status.open, 'glyphicon-chevron-
right': !status.open }"></i>
            </accordion-heading>
            <div class="row" ng-repeat="o in
project.ontologies" >
                <div class="col-md-5" >
                    <span class="glyphicon glyphicon-
remove centred" aria-hidden="true" tooltip="delete ontology" ng-
click="deleteOwl(o.ontologyId)"> </span>
                    <span class="glyphicon glyphicon-
download centred" aria-hidden="true" tooltip="download ontology" ng-
click="downloadOwl(o.ontologyId)"></span>
                    <span class="glyphicon glyphicon-
download centred" aria-hidden="true" tooltip="download processed
ontology" ng-click="downloadProcessed(o.ontologyId)"></span>
                    <span class="glyphicon glyphicon-
glyphicon-time" aria-hidden="true" tooltip="created">
{{getDateString(o.created )}}</span>
                </div>
                <div class="col-md-7"><span
tooltip="Ontology Name"><b> {{o.name}}</b></span></div>
                <hr />
            </div>

            <div class="well" ng-
show="isUserAuthenticated">
                <h4>Upload a new OWL file to the
project:</h4>
                <form name="formOWL" role="form">
                    <div class="form-group">
                        <label for="file" >OWL</label>
                        <input class="form-control"
type="file" ngf-select ng-model="owlFile" name="file" accept=".owl"
required></input>
                        <span class="error" ng-
show="formOWL.file.$error.required && formOWL.file.$dirty">
                            The OWL file is required!</span>
                </form>
            </div>
        </div>
    </div>
</div>

```

```

        </div>

        <br />
        <div>
            <button type="submit" class="btn
btn-primary glyphicon glyphicon-upload" ng-click="uploadOwl(owlFile)"
ng-disabled="!formOWL.$valid" > Upload</button>
        </div>
        <br />
        <alert ng-repeat="alert in
alerts" type="{{alert.type}}"
close="closeAlert($index)">{{alert.msg}}</alert>

        <div class="row progress" ng-
show="progressPercentage > 0" >

            <div class="progress-
bar" role="progressbar" aria-valuenow="{{progressPercentage}}" aria-
valuemin="0" aria-valuemax="100" style="width:100%">
{{progressPercentage}}%
            </div>
        </div>
        <span ng-
show="owlFile[0].result">Upload Successful</span>
        </form>
        <hr>
    </div>
</accordion-group>
</accordion>
<hr />

        </div>
    </div>
<!-- /.row -->

```

8.15.1.4.2.2 project-list.html

```

<!-- Page Heading -->
    <button ng-show="isUserAuthenticated" type="submit"
class="btn btn-primary" ng-click="addNewProject()">New
Project</button>
    <div class="row">

```



```

        <option value="50">50</option>
        <option value="100">100</option>
    </select>
</div>
</div>      -->
<hr>
<div class="" ng-repeat="b in filteredProjects | filter:{q
as results}">
    <!-- Project One -->
    <div class="row">
        <div class="col-md-2">
            <a href="#/projects/{{b.projectId}}">
                
            </a>
        </div>
        <div class="col-md-7">
            <h3>{{b.projectName}}</h3>
            <h4>Urls: {{b.urls.length}}<br/>
                <!-- <ul> <li ng-repeat="a in b.urls">
{{a.fullName}}</li>
                </ul> -->
            </h4>
            <h4>Ontology: {{b.ontologies.length}}<br/>
                <!-- <ul> <li ng-repeat="a in b.urls">
{{a.fullName}}</li>
                </ul> -->
            </h4>
            <h5><span class="glyphicon glyphicon-
time"></span> Created: {{getDateAsString(b.created)}}</h5>
            <h5><span class="glyphicon glyphicon-
time"></span> Modified: {{getDateAsString(b.modified)}}</h5>
            <p>{{b.projectDescription}}</p>
        </div>
    </div>

    <!-- /.row -->

    <hr>
</div>
<div class="" ng-if="books.length == 0">
    <strong>No results found...</strong>
</div>

```

```
<!-- Pagination -->
<div class="row text-center">
    <div class="col-lg-12">
        <pagination total-items="books.length" ng-
model="currentPage" max-size="maxSize" on-select-page="setPage(page)">
            ng-change="pageChanged()" class="pagination-sm" boundary-
links="true"></pagination>
    </div>
</div>

<!-- /.row -->

8.15.1.4.2.3 project-new.html
<style>

.error{
    color: red;
}
</style>

<div class="alert alert-danger" ng-show="error">
    There was a problem adding a new book. Please try again.
</div>

<alert ng-show="bookCreated" type="success"
close="closeAlert($index)">The book has been created.</alert>

<form ng-show="authenticated" name="formNewProject" role="form"
class="form-horizontal" ng-submit="createProject()" class="pure-form
login-form" novalidate >
    <fieldset class="pure-group">
        <legend>Add a new Project</legend>

        <div class="form-group">
            <label class="col-xs-2 control-label"
for="title">Title:</label>
            <div class="col-xs-10">
                <input ng-focus="" type="text" class="form-
control" id="name" name="name" placeholder="name" ng-model="name"
required ng-maxlength="255" ng-minlength="1"/>
                <span class="error" ng-
```

```

show= "formNewProject.name.$error.required &&
formNewBook.title.$dirty">The title is mandatory</span>
    <span class="error" ng-
show= "formNewProject.name.$error.minLength &&
formNewBook.title.$dirty">must have minimum 1 characters</span>
    </div>
    </div>

    <!-- <div class="form-group">
        <label class="col-xs-2 control-label"
for="title">Image:</label>
        <div class="col-xs-8">
            <select class="form-control" id="image"
name="image" ng-model="image" required >
                <option ng-repeat="n in [1, 2, 3, 4, 5, 6, 7,
8, 9, 10] track by $index" value="face{{n}}.jpg">project picture
{{n}}</option>
            </select>
        </div>

        <div class="col-xs-2">
            
        </div>
    </div> -->

    <div class="form-group">
        <label class="col-xs-2 control-label"
for="description">Description:</label>
        <div class="col-xs-10">
<!--           &lt;input ng-focus="" type="text" class="form-
control" id="description" name="description" placeholder="Description"
ng-model="description" required ng-minlength="6"/&gt; --&gt;
            &lt;div class="well" &gt;
                &lt;textarea class="form-control" id="description"
name="description" placeholder="Description" ng-model="description"
required ng-maxlength="1500" ng-minlength="6" rows="3"
required&gt;&lt;/textarea&gt;
            &lt;/div&gt;
        &lt;/div&gt;

            &lt;span class="error" ng-
<u>show= "formNewProject.description.$error.required &&
formNewProject.description.$dirty">The description is mandatory</span>

```

```
<span class="error" ng-
show="formNewProject.description.$error.minLength &&
fformNewProject.description.$dirty">must have description 6
characters</span>
</div>
</div>

</fieldset>

<hr>

<div class="form-group">
    <button type="submit" class="btn btn-primary" ng-
disabled="formNewProject.$invalid" ng-click="addNewProject()">Create
Project</button>
</div>
</form>
```

8.15.1.4.3 index.html

```
<!DOCTYPE html>
<html lang="en">

<head>

    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1">
    <meta name="description" content="">
    <meta name="author" content="">

    <title>Wonts Web App</title>

    <!-- Bootstrap Core CSS -->
    <link href="resources/css/bootstrap.min.css" rel="stylesheet">

    <!-- Custom CSS -->
    <link href="resources/css/wonts.css" rel="stylesheet">

    <!-- HTML5 Shim and Respond.js IE8 support of HTML5 elements and
media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the page via
file:// -->
```

```
<!--[if lt IE 9]>
<script
src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
<script
src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></script>
<!endif]-->

<!-- jQuery -->
<script src="resources/js/jquery-2.1.3.min.js"></script>

<!-- Bootstrap Core JavaScript -->
<script src="resources/js/bootstrap.min.js"></script>
<!-- angular js library /> -->
<script src="resources/js/angular/angular.js"></script>
<script src="resources/js/angular/angular-animate.js"></script>
<script src="resources/js/angular/angular-sanitize.js"></script>
<script src="resources/js/angular/angular-route.js"></script>
<script src="resources/js/angular/angular-messages.js"></script>
<script src="resources/js/ui-bootstrap-tpls-0.12.1.min.js"></script>
<script src="resources/js/file-upload/ng-file-upload-shim.js"></script>
<script src="resources/js/file-upload/ng-file-upload.js"></script>
<!-- angularjs app -->
<script src="resources/app/app.js"></script>
<script src="resources/app/config.route.js"></script>
<script src="resources/app/common/common.js"></script>
<script src="resources/app/common/logger.js"></script>
<script src="resources/app/common/spinner.js"></script>
<script
src="resources/app/common/bootstrap/bootstrap.dialog.js"></script>
<script src="resources/app/services/datacontext.js"></script>
<script
src="resources/app/services/download.service.js"></script>
<script src="resources/app/controllers/homeCtrl.js"></script>
<script src="resources/app/controllers/loginCtrl.js"></script>
<script src="resources/app/controllers/signinCtrl.js"></script>
<script
src="resources/app/controllers/projectListCtrl.js"></script>
<script
src="resources/app/controllers/projectNewCtrl.js"></script>
```

```

<script
src="resources/app/controllers/projectDetailCtrl.js"></script>
<script
src="resources/app/controllers/navigationCtrl.js"></script>

</head>
<body ng-app="wonts">

    <!-- Navigation -->
    <nav class="navbar navbar-inverse navbar-fixed-top"
role="navigation" ng-controller="navigationCtrl" >
        <div class="container">
            <!-- Brand and toggle get grouped for better mobile
display -->
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-
toggle="collapse" data-target="#bs-example-navbar-collapse-1">
                    <span class="sr-only">Toggle navigation</span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                <a class="navbar-brand" href="#">Wonts</a>
            </div>
            <!-- Collect the nav links, forms, and other content for
toggling -->
            <div class="collapse navbar-collapse" id="bs-example-
navbar-collapse-1">
                <ul class="nav navbar-nav">
                    <!-- <li>
                        <a href="#">About</a>
                    </li> -->
                    <li>
                        <a ng-show="authenticated"
href="#">Projects</a>
                    </li>
                </ul>
                <ul class="nav navbar-nav navbar-right">
                    <li ng-show="!authenticated"><a
href="#">Sign
In</a></li>
                    <li ng-show="!authenticated" ><a
href="#">Login</a></li>
                </ul>
            </div>
        </div>
    </nav>

```

```
        <li ng-show="authenticated" ><a href=""><span
class="glyphicon glyphicon-user"></span> Hi
{{userAuthenticated}}</a></li>
        <li ng-show="authenticated"><a href="" ng-
click="logout()"><span class="glyphicon glyphicon-log-out"></span>
Logout</a></li>
    </ul>
</div>
<!-- /.navbar-collapse -->
</div>
<!-- /.container -->
</nav>
<!-- Page Content -->
<div class="container">
    <h1>{{title}}</h1>

    <div ng-view></div>

    <!-- Footer -->
<footer>
    <div class="row">
        <div class="col-lg-12">
            <p>Copyright ©; Wonts Web App - by Salvatore
Rapisarda 2015</p>
        </div>
    </div>
    <!-- /.row -->
</footer>
</div>

</body>

</html>
```

```
8.15.1.5 app.js
(function () {
    'use strict';

    var app = angular.module('wonts', [
        // Angular modules
        'ngAnimate',          // animations
        'ngRoute',             // routing
```

```
'ngSanitize',           // sanitizes html bindings (ex:  
sidebar.js)  
'ngMessages',          //  
// Custom modules  
'common',               // common functions, logger, spinner  
'common.bootstrap',    // bootstrap dialog wrapper functions  
  
// 3rd Party Modules  
'ui.bootstrap',         // ui-bootstrap (ex: carousel,  
pagination, dialog)  
// to upload file  
'ngFileUpload',  
'owl.download'  
]);  
});  
});
```

8.15.1.6 config.exceptionHandler.js

```
(function () {  
  'use strict';  
  
  var app = angular.module('wonts');  
  
  app.config(['$provide', function ($provide) {  
    $provide.decorator('$exceptionHandler',  
      ['$delegate', 'config', 'logger',  
       extendExceptionHandler]);  
  }]);  
  
  function extendExceptionHandler($delegate, config, logger) {  
    var appErrorPrefix = config.appErrorPrefix;  
    var logError = logger.getLogFn('app', 'error');  
    return function (exception, cause) {  
      $delegate(exception, cause);  
      if (appErrorPrefix &&  
        exception.message.indexOf(appErrorPrefix) === 0) { return; }  
  
      var errorData = { exception: exception, cause: cause };  
      var msg = appErrorPrefix + exception.message;  
      logError(msg, errorData, true);  
    };  
  }  
});
```

```
        };
    }
})();
8.15.1.7 config.js
(function () {
    'use strict';

    var app = angular.module('wonts');

    // For use with the
    var remoteServiceName = 'api';

    var events = {
        controllerActivateSuccess: 'controller.activateSuccess',
        spinnerToggle: 'spinner.toggle'
    };

    var config = {
        appErrorPrefix: '[HT Error] ', //Configure the
exceptionHandler decorator
        docTitle: wonts,
        events: events,
        remoteServiceName: remoteServiceName,
        version: '1.0.0'
    };

    app.value('config', config);

    app.config(['$logProvider', function ($logProvider) {
        // turn debugging off/on (no info or warn)
        if ($logProvider.debugEnabled) {
            $logProvider.debugEnabled(true);
        }
    }]);
    //#region Configure the common services via commonConfig
    app.config(['commonConfigProvider', function (cfg) {
        cfg.config.controllerActivateSuccessEvent =
config.events.controllerActivateSuccess;
        cfg.config.spinnerToggleEvent = config.events.spinnerToggle;
    }]);
    //#endregion
});
```

```
}());
```

8.15.1.8 config.route.js

```
(function () {
    'use strict';

    var app = angular.module('wonts');

    // Collect the routes
    app.constant('routes', getRoutes());

    // Configure the routes and route resolvers
    app.config(['$routeProvider', '$httpProvider', 'routes',
        routeConfigurator]);
    function routeConfigurator($routeProvider, $httpProvider, routes)
    {
        routes.forEach(function (r) {
            $routeProvider.when(r.url, r.config);
        });
        $routeProvider.otherwise({ redirectTo: '/home' });

        $httpProvider.defaults.headers.common['X-Requested-With'] =
        'XMLHttpRequest';
    }

    // Define the routes
    function getRoutes() {
        return [
            {
                url: '/home',
                config: {
                    templateUrl: 'resources/app/views/home.html',
                    controller: 'homeCtrl',
                    title: 'Home',
                    settings: {
                        content: '<i class="fa fa-dashboard"></i>
home'
                    }
                }
            },
            {
                url: '/login',
                config: {
                    title: 'login',

```

```
        templateUrl:  
'resources/app/views/account/login.html',  
        controller: 'loginCtrl',  
        settings: {  
            content: '<i class="fa fa-lock"></i> Login'  
        }  
    },  
    {  
        url: '/signin',  
        config: {  
            title: 'signin',  
            templateUrl:  
'resources/app/views/account/signin.html',  
            controller: 'signinCtrl',  
            settings: {  
                content: '<i class="fa fa-lock"></i> Sing In'  
            }  
        }  
    },  
    {  
        url: '/projects',  
        config: {  
            title: 'signin',  
            templateUrl:  
'resources/app/views/projects/project-list.html',  
            controller: 'projectListCtrl',  
            settings: {  
                content: '<i class="fa fa-lock"></i> Books'  
            }  
        }  
    },  
    {  
        url: '/projects/:projectId',  
        config: {  
            title: 'signin',  
            templateUrl:  
'resources/app/views/projects/project-detail.html',  
            controller: 'projectDetailCtrl',  
        }  
    },  
    {  
        url: '/new-project',  
    }
```

```
        config: {
            title: 'Add a new Project',
            templateUrl:
'resources/app/views/projects/project-new.html',
            controller: 'projectNewCtrl',
        }
    ]
);
})();

```

8.15.1.9 metadata.js

```
window.app = window.app || {};
```

8.15.2 css

8.15.2.1 wonts.css

```
/*
 * Start Bootstrap - Blog Post HTML Template
 (http://startbootstrap.com)
 * Code licensed under the Apache License v2.0.
 * For details, see http://www.apache.org/licenses/LICENSE-2.0.
 */

body {
    padding-top: 70px; /* Required padding for .navbar-fixed-top.
Remove if using .navbar-static-top. Change if height of navigation
changes. */
}

footer {
    margin: 50px 0;
}

.animate-repeat.ng-move,
.animate-repeat.ng-enter,
.animate-repeat.ng-leave {
    -webkit-transition: all linear 0.5s;
    transition: all linear 0.5s;
}

.animate-repeat.ng-leave.ng-leave-active,
.animate-repeat.ng-move,
```

```
.animate-repeat.ng-enter {  
    opacity: 0;  
    max-height: 0;  
}  
  
.animate-repeat.ng-leave,  
.animate-repeat.ng-move.ng-move-active,  
.animate-repeat.ng-enter.ng-enter-active {  
    opacity: 1;  
    max-height: 40px;  
}  
  
.error{  
    color: red;  
    font-size: 16px;  
}  
  
.alert{  
}
```

8.16 webapp (pom.xml)

```
<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0  
    http://maven.apache.org/xsd/maven-4.0.0.xsd">  
    <modelVersion>4.0.0</modelVersion>  
  
    <groupId>wonts</groupId>  
    <artifactId>webapp</artifactId>  
    <version>0.0.1-SNAPSHOT</version>  
    <packaging>war</packaging>  
  
    <name>WontsWebApp</name>  
    <description>Wonts Web Application</description>  
  
    <parent>  
        <groupId>org.springframework.boot</groupId>  
        <artifactId>spring-boot-starter-parent</artifactId>  
        <version>1.2.5.RELEASE</version>  
  
        <relativePath/> <!-- lookup parent from repository -->  
    </parent>
```

```
<properties>
    <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
    <java.version>1.8</java.version>
</properties>

<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-jdbc</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-tomcat</artifactId>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>wonts</groupId>
        <artifactId>data.services</artifactId>
        <version>0.0.1-SNAPSHOT</version>
    </dependency>

```

```
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter</artifactId>
<exclusions>
<exclusion>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-logging</artifactId>
</exclusion>
</exclusions>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-log4j</artifactId>
</dependency>

</dependencies>

<build>
<plugins>
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
<configuration>
<jvmArguments>
-Xdebug -
</jvmArguments>
</configuration>
</plugin>
</plugins>
<resources>
<resource>
<directory>resources</directory>
<targetPath>resources</targetPath>
</resource>
</resources>
</build>
```

Xrunjdwp:transport=dt_socket,server=y,suspend=y,address=5555

</project>

9 Appendix B – Wont Database SQL Scripts

This SQL scripts below are used to create **wonts** database:

```
CREATE database wonts DEFAULT CHARACTER SET utf8
DEFAULT COLLATE utf8_general_ci;

CREATE USER 'wontsuser'@'localhost' IDENTIFIED BY 'user$wonts';
GRANT ALL ON wonts.* TO 'wontsuser'@'localhost';

USE wonts;

CREATE TABLE `nodes` (
  `hash` bigint(20) NOT NULL DEFAULT '0',
  `lex` longtext CHARACTER SET utf8 COLLATE utf8_bin,
  `lang` varchar(10) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL DEFAULT '',
  `datatype` varchar(200) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL DEFAULT '',
  `type` int(10) unsigned NOT NULL DEFAULT '0',
  PRIMARY KEY (`hash`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `prefixes` (
  `prefix` varchar(50) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
  `uri` varchar(500) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
  PRIMARY KEY (`prefix`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `quads` (
  `g` bigint(20) NOT NULL,
  `s` bigint(20) NOT NULL,
  `p` bigint(20) NOT NULL,
  `o` bigint(20) NOT NULL,
  PRIMARY KEY (`g`, `s`, `p`, `o`),
  KEY `SubjPredObj` (`s`, `p`, `o`),
  KEY `PredObjSubj` (`p`, `o`, `s`),
  KEY `ObjSubjPred` (`o`, `s`, `p`),
  KEY `GraPredObj` (`g`, `p`, `o`),
```

```
KEY `GraObjSubj` (`g`, `o`, `s`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `triples` (
`s` bigint(20) NOT NULL,
`p` bigint(20) NOT NULL,
`o` bigint(20) NOT NULL,
PRIMARY KEY (`s`, `p`, `o`),
KEY `ObjSubj` (`o`, `s`),
KEY `PredObj` (`p`, `o`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

10 Appendix C - DVD Content

The DVD structure and contents are described below:

- wonts
 - triples.services.jar, this file contains the triple crawler service
 - solr.services.jar, this file contains the Solr Indexer Service and the Search Engine Service
 - solr.services_lib, this folder contains all the solr.services.jar dependency libraries
 - triples.crawler_lib, this folder contains all triples.services.jar dependency libraries
 - owl_crawled, this folder contains all the OWL files that have been crawled
 - resources, this folder contains resources and application properties
 - ontology_uploaded this folder contains all the ontology files uploaded and processed
 - logs, this folder contains all the log files
 - start-solrservice.sh, this file is used for running solr.services
 - stop-solrservice.sh, this file is used for stopping solr.services
 - start-triplescrawler.sh, this file is used for running triples.crawler services
 - stop-triplescrawler.sh, this file is used for stopping triples.crawler service
- ontologies-example, this folder contains all the OWL file examples mentioned in this report
 - finance.owl, this OWL file does not contain any individual. This example has been used during the project evaluation (5 Evaluation) in the Financial Ontology Test (5.1.1)
 - finance.owl_individuals.owl, this OWL file contains individuals added by **Wonts**. This example has been used during the project evaluation (5 Evaluation) in the Financial Ontology Test (5.1.1)
 - pizzashop.owl, this OWL file does not contain any individual. This example has been used during the project evaluation (5 Evaluation) as example in the Pizza Shop Test (5.1.2)
 - pizzashop.owl_individuals.owl, this OWL file contains individuals added by **Wonts**. This example has been used during the project evaluation (5 Evaluation) as example in the Pizza Shop Test (5.1.2)
 - MichaelZakharyaschevWebSite.owl, this OWL file is an example used in the project implementation section (4-1 Triples Crawler)
- Dev, this folder contains all the codes and scripts for each part of the system discussed in the report
 - data.services project
 - solr.services project
 - triples.crawler project
 - webapp project
- solr-5.0.0, this folder contains Apache Solr and **wonts** core used in this report.