ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

(ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΠΟΛΥΜΕΣΩΝ)

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΜΕ ΘΕΜΑ:

Δημιουργία δυναμικού καταλόγου επιλογών (wizard) για τη δημιουργία σημασιολογικών ερωτημάτων από απλούς χρήστες.

της σπουδάστριας: ΣΤΕΙΑΚΑΚΗ ΜΑΛΒΙΝΑΣ (Α.Μ. 2658)

Εισηγητές:
ΜΑΛΑΜΟΣ ΑΘΑΝΑΣΙΟΣ
ΚΑΠΕΤΑΝΑΚΗΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

Ηράκλειο, 2013

ΕΥΧΑΡΙΣΤΙΕΣ

Ένα τεράστιο ευχαριστώ στον ***Κωνσταντίνο Κοντάκη***

απόφοιτο του τμήματος Εφαρμοσμένης Πληροφορικής

& Πολυμέσων για την βοήθεια που μου προσέφερε σε

όλη τη διάρκεια αυτής της πτυχιακής. Επίσης θα ήθελα

να ευχαριστήσω τους εισηγητές της πτυχιακής κύριους

Αθανάσιο Μαλάμο και ***Κωνσταντίνο Καπετανάκη***

καθώς και την ***οικογένεια μου*** για την στήριξή τους .

Abstract

In this thesis we created semantic web application which uses the technologies of HTML,CSS and Javascript. The application was developed in Java language and the ontology interaction operations are based on the Jena framework API. The ontology type is OWL-DL, which was developed by protégé. This application supports two kinds of users with distinct but interdependent capabilities. This application enables the administrator to create an interior space which is saved in the database with SDB and in separate file. The administrator through these has the ability to modify or to enrich his interior space for the second time. Also the administrator is responsible for the definition of the form fields and the visitor fills them out in accordance to his interior space and submits them in a routine. The form adds dynamically new items according to the user's choices. These choices translate into semantic web language which is SPARQL.

Keywords: ***SEMANTIC WEB, JENA, SDB, SPARQL, OWL***

Περίληψη

Σε αυτή την πτυχιακή αναπτύχτηκε εφαρμογή σημασιολογικού ιστού η οποία χρησιμοποιεί τις τεχνολογίες HTML,CSS και Javascript.Η εφαρμογή αναπτύχθηκε στη γλώσσα της Java και οι λειτουργίες αλληλεπίδρασης της οντολογίας με βάση το framework της Jena. Η οντολογία είναι τύπου OWL-DL και αναπτύχθηκε μέσω του protégé. Αυτή η εφαρμογή υποστηρίζει δύο επίπεδα χρηστών με ξεχωριστές αλλά αλληλένδετες ικανότητες. Ο διαχειριστής έχει τη δυνατότητα δημιουργίας ενός χώρου εσωτερικής διακόσμησης ο οποίος αποθηκεύεται σε βάση δεδομένων μέσω της SDB αλλά και σε ξεχωριστό αρχείο. Ο διαχειριστής μέσω αυτών έχει τη δυνατότητα να τροποποιήσει ή να εμπλουτίσει το δικό του εσωτερικό χώρο διακόσμησης σε δεύτερο χρόνο. Επίσης είναι υπεύθυνος για τον ορισμό των πεδίων της φόρμας και ο επισκέπτης τα συμπληρώνει και τα στέλνει σε μία ρουτίνα. Η φόρμα προσθέτει δυναμικά νέα στοιχεία σύμφωνα με τις επιλογές του χρήστη. Αυτές οι επιλογές μεταφράζονται σε γλώσσα σημασιολογικού χαρακτήρα η οποία είναι η SPARQL.

Λέξεις κλειδιά: ***ΣΗΜΑΣΙΟΛΟΓΙΚΟΣ ΙΣΤΟΣ, JENA, SDB, SPARQL, OWL***

Πίνακας περιεχομένων

Πίνακας εικόνων

Λίστα Πινάκων

1 ΣΗΜΑΣΙΟΛΟΓΙΚΟΣ ΙΣΤΟΣ (SEMANTIC WEB)

Πηγή εικόνας: http://www.auth.gr/sites/default/files/wordle_SWUinline2.png

Ο σημασιολογικός Ιστός αποτελεί το επόμενο βήμα εξέλιξης στο διαδίκτυο. Η ανάπτυξη του Παγκόσμιου Ιστού (World Wide Web) κατέστησε το διαδίκτυο προσβάσιμο σε εκατομμύρια χρήστες, επιτρέποντας την απρόσκοπτη δημοσιοποίηση και πρόσβαση σε έγγραφα. Η εκρηκτική ανάπτυξη του Παγκόσμιου Ιστού δημιούργησε προβλήματα "πληροφοριακής υπερφόρτισης". Η παγκόσμια ερευνητική κοινότητα έχει στραφεί εδώ και λίγα χρόνια σε μία νέα κατεύθυνση εξέλιξης του ιστού, η οποία ονομάζεται "Σημασιολογικός Ιστός" (Semantic Web) και περιλαμβάνει τη σαφή αναπαράσταση του νοήματος των πληροφοριών και των εγγράφων, επιτρέποντας την αυτόματη επεξεργασία και ενοποίηση διαδικτυακών πόρων από "έξυπνα" προγράμματα-πράκτορες.

Ο όρος *Semantic Web*, καθώς και η αρχιτεκτονική για την υλοποίησή του, προτάθηκαν από τον Tim Berners-Lee, τον εφευρέτη του σημερινού Παγκόσμιου Ιστού. Ο Σημασιολογικός Ιστός υιοθετήθηκε από το World Wide Web Consortium (W3C), έναν Οργανισμό που στοχεύει στην προώθηση, ανάπτυξη και εξέλιξη του Web και των πρωτοκόλλων που το υποστηρίζουν. Ο Σημασιολογικός Ιστός[1] είναι μία επέκταση και βελτίωση του σημερινού Web στην κατεύθυνση, κυρίως της δόμησης της πληροφορίας, έτσι ώστε αυτή να είναι προσπελάσιμη από εφαρμογές υπολογιστών, με τελικό στόχο την αυτοματοποίηση πολλών λειτουργιών στο Διαδίκτυο. Η σημερινή αναπαράσταση της πληροφορίας που προορίζεται για χρήση από ανθρώπους θα αντικατασταθεί από μία αναπαράσταση κατανοητή από υπολογιστές. Ο Σημασιολογικός Ιστός προσπαθεί να επιλύσει το πρόβλημα της αναπαράστασης της γνώσης από τους υπολογιστές[2].Βασικό συστατικό του Σημασιολογικού Ιστού είναι ο μηχανισμός επεξεργασίας της γνώσης που διαχειρίζεται λογικά τις

[1] Berners-Lee, Hendler & Lassila, 2001
[2] Fensel, 2002

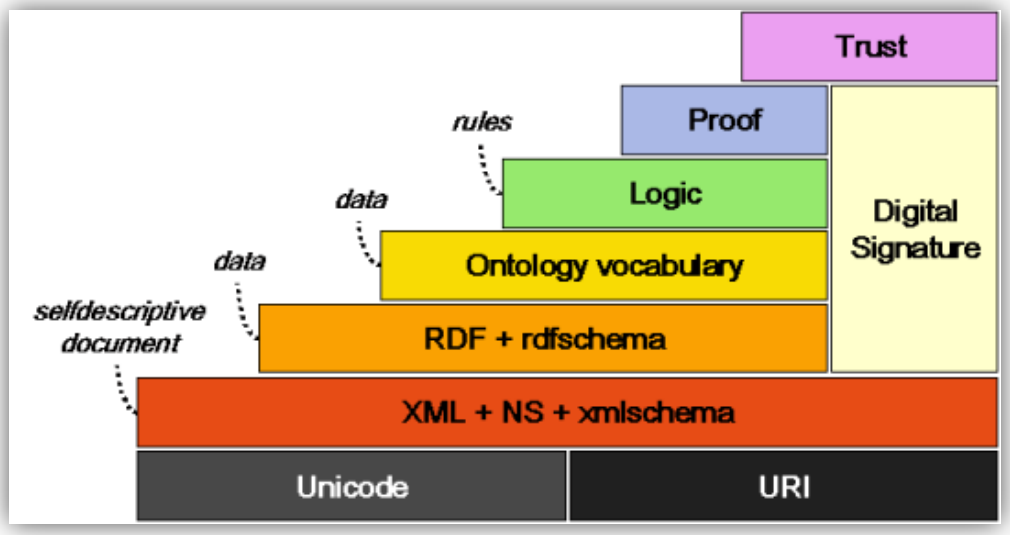
πληροφορίες με σκοπό την εξαγωγή συμπερασμάτων, τη δημιουργία νέας γνώσης, την υποστήριξη στη λήψη αποφάσεων και τέλος την αυτόματη εκτέλεση ενεργειών.

Οι βασικές αρχές του Σημασιολογικού Ιστού είναι:

- *Η διατήρηση του κατανεμημένου περιεχομένου του Διαδικτύου.*
- *Η αναπαράσταση και ανάκτηση της πληροφορίας, καθώς οι εφαρμογές των υπολογιστών προσπελαύνουν δομημένες πηγές πληροφορίας και κανόνες, οι οποίοι χρησιμοποιούνται για να αιτιολογούν τις σχέσεις μεταξύ των πληροφοριών.*
- *Η αναπαράσταση των εννοιών μιας θεματικής περιοχής επιτυγχάνεται με τη χρήση των οντολογιών.*
- *Η ύπαρξη πρακτόρων λογισμικού (software agents), δηλαδή προγραμμάτων που θα αναλαμβάνουν για λογαριασμό του χρήστη να κινούνται στο Διαδίκτυο και να συλλέγουν την πληροφορία από διάφορες πηγές που διαθέτουν σημασιολογικό περιεχόμενο.*

1.1 ΒΑΣΙΚΕΣ ΤΕΧΝΟΛΟΓΙΕΣ ΣΗΜΑΣΙΟΛΟΓΙΚΟΥ ΙΣΤΟΥ

Συχνά, ο όρος «Σημασιολογικός Ιστός» αναφέρεται απλά σε ένα σύνολο από τεχνολογίες και πρότυπα που αποτελούν τα βασικά συστατικά ενός συστήματος το οποίο θα μπορούσε να υποστηρίξει το όραμα ενός Ιστού βασισμένου στη γνώση. Η παρακάτω εικόνα παρουσιάζει την αρχιτεκτονική του συστήματος αυτού και οι παρακάτω ενότητες αναλύουν το κάθε κομμάτι της αρχιτεκτονικής του σημασιολογικού ιστού.



Εικόνα 1-Αρχιτεκτονική Σημασιολογικού Ιστού

Πηγή εικόνας: http://www.w3.org/2001/12/semweb-fin/swlevels.png

1.1.1 Unicode και URI

Στο χαμηλότερο επίπεδο βρίσκουμε το πρότυπο unicode, ένα σύστημα κωδικοποίησης χαρακτήρων που σχεδιάστηκε για να υποστηρίζει την ανταλλαγή, επεξεργασία και εμφάνιση κειμένων σε οποιαδήποτε γλώσσα του κόσμου (αντιστοιχώντας ένα μοναδικό αριθμό σε κάθε χαρακτήρα). Το unicode αποτελεί βασικό συστατικό όλων των σύγχρονων πρωτοκόλλων στις τεχνολογίες πληροφορικής και τηλεπικοινωνιών, καθώς παρέχει μία κοινή και ενιαία αρχιτεκτονική για την κωδικοποίηση περισσότερων από 96.000 χαρακτήρων και αποτελεί τη βάση για την επεξεργασία, αποθήκευση και ανταλλαγή δεδομένων κειμένου παγκοσμίως. Στο ίδιο επίπεδο με το unicode, βρίσκεται επίσης και το URI (Uniform Resource Identifier), το πρότυπο για την αναγνώριση και τον εντοπισμό ενός ονόματος ή μιας πηγής (π.χ. μιας ιστοσελίδας) σε ένα δίκτυο, με τη χρήση μιας προκαθορισμένης μορφής απόδοσης ονομάτων. Συσχετίζοντας μία πηγή με ένα URI, ο καθένας μπορεί να συνδεθεί και αναφερθεί σε αυτή ή να ανακτήσει μία αναπαράστασή της. Τα δύο αυτά πρότυπα αποτελούν τα θεμέλια της αρχιτεκτονικής του Σημασιολογικού Ιστού, μιας και όπως θα δούμε στη συνέχεια είναι απαραίτητα για τον καθορισμό άλλων προτύπων που βρίσκονται στα επόμενα επίπεδα, όπως π.χ. της XML η οποία βασίζεται στο unicode και του RDF το οποίο βασίζεται στα URIs.

1.1.2 XML

Ένα βασικό βήμα για την υλοποίηση του Σημασιολογικού Ιστού είναι η χρήση των λεγόμενων μεταδεδομένων. Ο πιο απλός ορισμός που μπορεί να δοθεί για τα μεταδεδομένα, είναι ότι πρόκειται για δεδομένα σχετικά με δεδομένα. Τα μεταδεδομένα μας διευκολύνουν σημαντικά στην κατανόηση, χρήση και διαχείριση των δεδομένων, ενώ η XML - που χρησιμοποιείται ήδη κατά κόρον στον κόσμο του διαδικτύου - αποτελεί ένα σημαντικό μηχανισμό για τη δημιουργία μεταδεδομένων.

Η XML (eXtensible Markup Language - επεκτάσιμη γλώσσα σήμανσης) είναι η επικρατέστερη γλώσσα για την περιγραφή και ανταλλαγή δεδομένων και κειμένων στο Διαδίκτυο. Η XML παρέχει τη δυνατότητα δημιουργίας κειμένων με απεριόριστα πολύπλοκη δομή και συντακτικό. Έτσι, μπορούν να δομηθούν οι πληροφορίες που περιέχονται στα κείμενα για να επεξεργάζονται πιο εύκολα από τους υπολογιστές. Στις ιστοσελίδες, τα υπάρχοντα κείμενα θα αντικατασταθούν με δομημένα κείμενα σε μορφή XML και RDF.

Η XML όπως και η HTML διέπεται από ένα σύνολο κανόνων (ή διαφορετικά ένα πακέτο κατευθυντήριων γραμμών και συμβάσεων) για το σχεδιασμό μορφών κειμένου που διευκολύνουν τη δόμηση ενός εγγράφου (document). Η XML έχει σχεδιαστεί για τη σήμανση εγγράφων που περιέχουν «αυθαίρετες» δομές και στοιχεία, σε αντίθεση με την HTML, η οποία σχεδιάστηκε για έγγραφα του διαδικτύου που περιέχουν προκαθορισμένα στοιχεία. Για να γίνει καλύτερα κατανοητή η σημασία της XML και η διαφορά της σε σχέση

με την HTML, παρακάτω παραθέτουμε ένα παράδειγμα περιγραφής ενός βιβλίου, πρώτα χρησιμοποιώντας HTML:

```
<h2>A Semantic Web Primer</h2>

<i>by <b>Grigoris Antoniou</b> and <b>Frank van

Harmelen</b></i><br>

MIT Press 2004<br>

ISBN 0-262-01210-3
```

Και στη συνέχεια χρησιμοποιώντας XML:

```
<book>
    <title>A Semantic Web Primer</title>
    <author>Grigoris Antoniou</author>
    <author>Frank van Harmelen</author>
    <publisher>MIT Press</publisher>
    <year>2004</year>
    <isbn>0-262-01210-3</isbn>
</book>
```

Όπως παρατηρούμε, τόσο η HTML όσο και η XML, όντας γλώσσες σήμανσης και οι δύο βασίζονται στα λεγόμενα tags Το περιεχόμενο, μαζί με τα tags από τα οποία περιβάλλεται, ονομάζεται στοιχείο (element). Το XML έγγραφο είναι πολύ πιο εύκολα προσβάσιμο από υπολογιστές, μιας και κάθε κομμάτι πληροφορίας που περιέχει περιγράφεται κατάλληλα από μεταδεδομένα. Επιπλέον, οι σχέσεις ανάμεσα σε αυτά τα κομμάτια πληροφοριών ορίζονται από τη φωλιασμένη δομή της XML. Για παράδειγμα, τα <author> tags βρίσκονται μέσα στα <book> tags, που σημαίνει ότι περιγράφουν τις ιδιότητες του συγκεκριμένου βιβλίου. Όταν ένας υπολογιστής επεξεργαστεί αυτό το έγγραφο, θα είναι σε θέση να συμπεράνει ότι το στοιχείο author αναφέρεται στο στοιχείο book.

Το πρόβλημα ωστόσο που προκύπτει, οφείλεται στο γεγονός ότι, όπως αναφέραμε παραπάνω, τα έγγραφα XML δεν περιλαμβάνουν συγκεκριμένη δομή και προκαθορισμένα tags. Συνεπώς, όταν π.χ. δύο επιχειρήσεις θελήσουν να ανταλλάξουν δεδομένα με βάση το πρότυπο XML, θα πρέπει να συμφωνήσουν και να ορίσουν με κάποιο τρόπο τη δομή και τα tags που θα περιλαμβάνουν τα έγγραφα XML που θα ανταλλάσσουν. Για το σκοπό αυτό, στην πρώτη έκδοση της XML (XML 1.0), χρησιμοποιούνται τα Document Type Definitions (DTD), ένα είδος γραμματικής για τον καθορισμό περιορισμών, όσον αφορά τη χρήση των tags και τις επιτρεπόμενες σχέσεις μεταξύ τους σε ένα έγγραφο XML. Τον τελευταίο καιρό όμως, γίνονται προσπάθειες για την αντικατάσταση του DTD από ένα νεότερο πρότυπο με την ονομασία XML Schema το οποίο αν και παρέχει πολλά πλεονεκτήματα σε σχέση με το DTD, έχει ουσιαστικά τον ίδιο στόχο, δηλαδή αποτελεί ένα είδος γραμματικής για έγγραφα XML.

1.1.3 XML Schema

Το πρότυπο XML συμπληρώνεται από το πρότυπο XML Schema, μια γλώσσα με την οποία γράφουμε "λεξικά" και "γραμματικές" για XML κείμενα. Το XML Schema ορίζει τα επιτρεπόμενα στοιχεία, τις ιδιότητές τους, καθώς και τον τρόπο με τον οποίο συνδυάζονται μεταξύ τους μέσα στο XML κείμενο. Με απλά λόγια, το XML Schema αποτελεί το "συντακτικό" του XML κειμένου. Μία τέτοια σχεδιαστική απόφαση παρέχει δύο σημαντικά πλεονεκτήματα: αυξάνεται κατά πολύ η ευκολία ανάγνωσης ενός XML Schema, ενώ επιτυγχάνεται σημαντική επαναχρησιμοποίηση της τεχνολογίας.

1.1.4 RDF

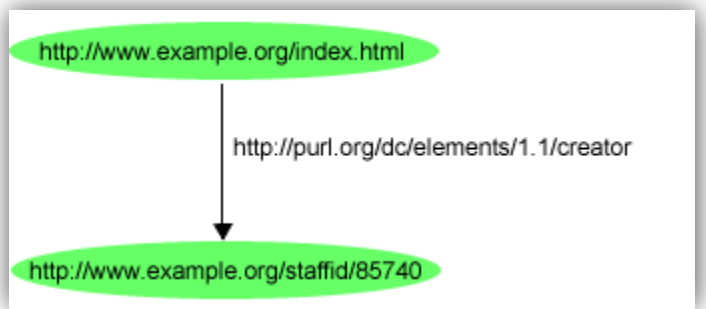
Η γλώσσα RDF (Resource Description Framework - Περιβάλλον Περιγραφής Πόρων) είναι το πρότυπο που υιοθετήθηκε από το W3C για την περιγραφή πληροφοριακών πόρων και γενικότερα για την αναπαράσταση της γνώσης στο περιβάλλον του Διαδικτύου. Μέσω του RDF είναι δυνατή η μετατροπή της πληροφορίας σε σημασιολογική. Το RDF βασίζεται στην ιδέα του προσδιορισμού πραγμάτων με τη χρήση των Uniform Resource Identifiers (URIs) καθώς και στην απλή περιγραφή των ιδιοτήτων αυτών των πραγμάτων. Το RDF χρησιμοποιεί ένα μοντέλο γράφων (graph model), για την αναπαράσταση προτάσεων με τη χρήση κόμβων και βελών. Σύμφωνα με αυτό το μοντέλο, μία πρόταση αναπαρίσταται με:

- Έναν κόμβο για το subject,
- Έναν κόμβο για το object και
- Ένα βέλος για το predicate, με φορά από τον κόμβο του subject στον κόμβο του object.

Με βάση τα παραπάνω, η πρόταση:

http://www.example.org/index.html has a **creator** whose value is **John Smith**

μπορεί να αναπαρασταθεί με το γράφο της Εικόνας 2



Εικόνα 2-Γράφος RDF

Πηγή εικόνας: RDF Primer ,2004 (http://www.w3.org/TR/rdf-primer/)

Πολλές φορές, δεν είναι βολική η απεικόνιση γράφων για την αναπαράσταση RDF προτάσεων και γι' αυτό το λόγο υπάρχει ένας εναλλακτικός τρόπος αναπαράστασης αυτών, με τη μορφή triplets (τριπλέτες). Σε αυτή την περίπτωση, κάθε πρόταση του γράφου, γράφεται ως μια απλή τριπλέτα που αποτελείται από ένα subject, predicate και object, μεαυτή τη σειρά. Για παράδειγμα, η πρόταση της Εικόνας 2, θα γραφόταν ως εξής:

```
<http://www.example.org/index.html>
<http://purl.org/dc/elements/1.1/creator>
<http://www.example.org/staffid/85740> .
```

Η συντριπτική πλειοψηφία της γνώσης που θέλουμε να αναπαραστήσουμε στους υπολογιστές μπορεί να αναπαρασταθεί με αυτή την μορφή. Στο περιβάλλον RDF, το αντικείμενο και η ιδιότητα δηλώνονται με ένα URI. Η τιμή μπορεί να δηλώνεται με ένα URI ή μπορεί να είναι ένα αλφαριθμητικό ή μια λέξη. Τέλος, το πρότυπο RDF ορίζει το συντακτικό XML, δηλαδή τον τρόπο με τον οποίο οι προτάσεις RDF εκφράζονται ως XML κείμενα.

1.1.5 RDF Schema

Το RDF Schema είναι η επέκταση του RDF. Είναι μια γλώσσα με την οποία το μοντέλο δεδομένων του RDF εμπλουτίζεται με χαρακτηριστικά αντικειμενοστραφούς αναπαράστασης, όπου ο πόρος αντιστοιχεί σε αντικείμενο. το RDF Schema δεν παρέχει ένα συγκεκριμένο λεξιλόγιο που να αναφέρεται σε συγκεκριμένες εφαρμογές (όπως για παράδειγμα οι κλάσεις και οι ιδιότητες που περιγράψαμε παραπάνω). Αντί αυτού, το RDFS παρέχει τις προϋποθέσεις που απαιτούνται για να περιγράψει κανείς τέτοιες κλάσεις και ιδιότητες και για να υποδηλώσει ποιες κλάσεις και ιδιότητες αναμένεται να χρησιμοποιηθούν μαζί (για παράδειγμα να καθορίσει ότι η ιδιότητα ex:jobTitle θα χρησιμοποιηθεί για την περιγραφή του ex:Person). Με άλλα λόγια, το RDF Schema παρέχει ένα σύστημα τύπων για το RDF. Το σύστημα αυτό του RDFS είναι παρόμοιο, έως ένα βαθμό, με τα συστήματα τύπων των αντικειμενοστραφών γλωσσών προγραμματισμού όπως η Java. Για παράδειγμα, το RDF Schema επιτρέπει τον ορισμό πόρων ως στιγμιότυπα (instances) μίας ή περισσότερων κλάσεων. Επιπλέον, επιτρέπει την οργάνωση των κλάσεων σε μία ιεραρχική δομή. Παραδείγματος χάριν, μία κλάση ex:Dog μπορεί να οριστεί ως υποκλάση του ex:Mammal, το οποίο με τη σειρά του αποτελεί υποκλάση του ex:Animal, εννοώντας ότι οποιοσδήποτε πόρος που είναι στιγμιότυπο της κλάσης ex:Dog είναι επίσης έμμεσα στιγμιότυπο της κλάσης ex:Animal. Οι ιδιότητες του RDF Schema, παρέχονται και οι ίδιες υπό την μορφή ενός RDF λεξιλογίου, δηλαδή ως ένα εξειδικευμένο σετ από προκαθορισμένες πηγές RDF με το δικό τους ειδικό νόημα. Οι πηγές του λεξιλογίου του RDF Schema προσδιορίζονται με το πρόθεμα rdfs: το οποίο παραπέμπει στο URI http://www.w3.org/2000/01/rdf-schema#. Οι περιγραφές του λεξιλογίου που είναι γραμμένες στη γλώσσα του RDF Schema, αποτελούν έγκυρους RDF γράφους.

1.1.6 Ontology (Οντολογία)

Οι γλώσσες οντολογιών, δίνουν τη δυνατότητα στους χρήστες να αναπαριστούν με ξεκάθαρο τρόπο domain models. Οι κύριες απαιτήσεις τους είναι:

- Καλά-ορισμένη σύνταξη (well-defined syntax),
- Τυπική σημασιολογία (formal semantics).
- Αποτελεσματική υποστήριξη συλλογισμών (efficient reasoning support).
- Επαρκή εκφραστική δύναμη (efficient expressive power).
- Ευκολία στην έκφραση (convenience of expression).

Η τυπική σημασιολογία (formal semantics), περιγράφει το νόημα της γνώσης με ακρίβεια. δεν αναφέρεται σε υποκειμενικές κρίσεις ούτε είναι ανοιχτή σε διαφορετικές ερμηνείες από διαφορετικούς ανθρώπους Μία χρήση της τυπικής σημασιολογίας είναι για τη διευκόλυνση της διαδικασίας συλλογισμού γύρω από τη γνώση (reasoning about knowledge). Όσον αφορά την οντολογική γνώση, η διαδικασία συλλογισμού μπορεί να αναφέρεται στα εξής:

- Συμμετοχή σε κλάσεις (class membership).
- Ισοδυναμία κλάσεων.
- Συνάφεια
- Κατηγοριοποίηση.

Η σημασιολογία είναι προαπαιτούμενο για την υποστήριξη της διαδικασίας συλλογισμού. Συμπεράσματα όπως τα προηγούμενα μπορούν να βγουν αυτόματα αντί να υπολογίζονται με το χέρι. Η υποστήριξη συλλογισμών (reasoning support) είναι σημαντική γιατί επιτρέπει:

- Τον έλεγχο της συνάφειας της οντολογίας και της γνώσης,
- Τον έλεγχο ανεπιθύμητων σχέσεων ανάμεσα σε κλάσεις,
- Την αυτόματη ταξινόμηση στιγμιοτύπων κλάσεων.

1.1.7 Logic (Λογική)

Η Λογική, παρέχει το υπόβαθρο για τη δυνατότητα αυτοματοποιημένου συλλογισμού και συμπερασμών βάσει των πληροφοριών που δομούνται σε μία οντολογία. Το επίπεδο αυτό επιπλέον καθιστά δυνατή και ισχυροποιείται από τη χρήση τυπικών κανόνων, βάσει των οποίων γίνεται εφικτή η ψευδονοήμων διαδικασία λήψης, αποφάσεων από τις υπολογιστικές μηχανές.

1.1.8 Proof (Απόδειξη)

Τα αποτελέσματα που συμπεραίνονται από δεδομένα στο Σημασιολογικό Ιστό να μπορούν να οδηγήσουν πίσω στις υποθέσεις που τα προκάλεσαν.

1.1.9 Trust (Εμπιστοσύνη)

Η εμπιστοσύνη σε συνδυασμό με την τεχνολογία των ψηφιακών υπογραφών (digital signatures), εξασφαλίζει το βαθμό στον οποίο οι πληροφορίες που διακινούνται, επεξεργάζονται και συμπεραίνονται στον Σημασιολογικό Ιστό, είναι αξιόπιστες, με αυτοματοποιημένο τρόπο (για παράδειγμα, στην επικοινωνία μεταξύ πρακτόρων).

2 WEB ONTOLOGY LANGUAGE - OWL

Πηγές εικόνων: http://dumontierlab.com/images/owl.jpg , http://www.cmswire.com/images/owl.jpg και http://www.w3.org/Icons/SW/sw-vert.png

Η ***OWL*** (Web Ontology Language-Γλώσσα Οντολογιών Ιστού) είναι μία οικογένεια γλωσσών που χρησιμοποιείται για την περιγραφή των οντολογιών που υπάρχουν στο Διαδίκτυο. Η οντολογία είναι μία αυστηρή περιγραφή των πόρων και των μεταξύ τους σχέσεων. Συγκεκριμένα, η οντολογία είναι η αποδεκτή σημασιολογικά κωδικοποίηση της πληροφορίας ενός θεματικού χώρου. Οι οντολογίες επιτρέπουν στους χρήστες να έχουν κοινή ονοματολογία και αντίληψη για τα αντικείμενα που δηλώνουν ή χρησιμοποιούν. Βοηθούν τον χρήστη να πλοηγηθεί στον θεματικό χώρο της πληροφορίας που βασίζεται σε σημασιολογικές και όχι σε λεξιλογικές έννοιες. Στις οντολογίες, η δυσκολία εντοπίζεται στο ότι οι κοινότητες χρηστών με κοινά ενδιαφέροντα θα πρέπει να συμφωνήσουν στην οντολογική περιγραφή του θεματικού χώρου ενδιαφέροντός τους.

Η OWL όπως προείπαμε είναι μια οικογένεια γλωσσών καθώς αποτελείται από τρείς υπογλώσσες καθεμία από τις οποίες σχεδιάστηκε έτσι ώστε να ικανοποιεί διαφορετικές ανάγκες σε συγκεκριμένες κοινότητες χρηστών. Οι γλώσσες αυτές χαρακτηρίζονται από την επίσημη σημασιολογία και βασίζονται στο RDF/XML serializations για τον σημασιολογικό ιστό.

Οι τρεις επιμέρους υπογλώσσες είναι:

- ***OWL Lite***
- ***OWL DL***
- ***OWL Full***

Η OWL επικυρώνεται από τον World Wide Web Consortium (W3C) και έχει προσελκύσει το ακαδημαϊκό, ιατρικό και εμπορικό ενδιαφέρον.

2.1 ΕΠΙΠΕΔΑ (ΥΠΟΓΛΩΣΣΕΣ) OWL

Κάθε μία από αυτές τις υπογλώσσες, αποτελεί μία επέκταση της απλούστερης προηγούμενης γλώσσας, τόσο σχετικά με το τι μπορεί να εκφραστεί, όσο και με το τι συμπέρασμα μπορεί να βγει. Ισχύουν οι παρακάτω σχέσεις (αλλά όχι οι αντίστροφες):

- ✓ Κάθε έγκυρη OWL Lite οντολογία αποτελεί και έγκυρη OWL DL οντολογία.
- ✓ Κάθε έγκυρη OWL DL οντολογία αποτελεί και έγκυρη OWL Full οντολογία.
- ✓ Κάθε έγκυρο OWL Lite συμπέρασμα αποτελεί και έγκυρο OWL DL συμπέρασμα.
- ✓ Κάθε έγκυρο OWL DL συμπέρασμα αποτελεί και έγκυρο OWL Full συμπέρασμα.

Κατά την ανάπτυξη οντολογιών, θα πρέπει να γίνεται επιλογή της κατάλληλης υπογλώσσας, ανάλογα με τις ανάγκες κάθε φορά. Η επιλογή ανάμεσα στην OWL Lite και στην OWL DL, εξαρτάται από το βαθμό στον οποίο οι χρήστες απαιτούν τα πιο εκφραστικά στοιχεία που παρέχονται από την OWL DL. Η επιλογή ανάμεσα στην OWL DL και στην OWL Full κυρίως εξαρτάται από το βαθμό στον οποίο οι χρήστες απαιτούν τις ευκολίες μετα-μοντελοποίησης που παρέχει το RDF Schema (π.χ. ο ορισμός κλάσεων και η απόδοση ιδιοτήτων σε κλάσεις). Όταν χρησιμοποιείται η OWL Full σε σχέση με την OWL DL, η υποστήριξη συλλογιστικής είναι λιγότερο αναμενόμενη, καθώς πλήρεις υλοποιήσεις της OWL Full δεν υπάρχουν προς το παρόν. Η OWL Full μπορεί να θεωρηθεί μία επέκταση του RDF, ενώ η OWL Lite και η OWL DL, μπορούν να θεωρηθούν επεκτάσεις μιας περιορισμένης όψης του RDF.

2.1.1 OWL Lite

Η OWL Lite υποστηρίζει τους χρήστες που κυρίως χρειάζονται μία απλή ιεραρχία κατηγοριών και κάποιους απλούς περιορισμούς. Για παράδειγμα, ενώ υποστηρίζει περιορισμούς πληθικότητας, επιτρέπει μόνο τιμές πληθικότητας 0 ή 1. Το βασικό πλεονέκτημα της OWL Lite, είναι ότι αποτελεί μία γλώσσα που είναι εύκολη τόσο στην κατανόηση (για τους χρήστες), όσο και στην εφαρμογή (για τους δημιουργούς εργαλείων λογισμικού). Το μειονέκτημα φυσικά είναι η περιορισμένη εκφραστικότητα που παρέχει.

2.1.2 OWL DL

Η OWL DL υποστηρίζει τους χρήστες που επιθυμούν μέγιστη εκφραστικότητα αλλά παράλληλα θέλουν να εξασφαλίζουν και υπολογιστική πληρότητα (ότι δηλαδή όλοι οι υπολογισμοί θα τελειώσουν σε πεπερασμένο χρόνο). Η OWL DL περιλαμβάνει όλα τα δομικά στοιχεία της OWL, αλλά αυτά μπορούν να χρησιμοποιηθούν μόνο υπό ορισμένους περιορισμούς (για παράδειγμα, ενώ μία κλάση μπορεί να αποτελεί υποκλάση πολλών κλάσεων, μία κλάση δε μπορεί να είναι στιγμιότυπο μίας άλλης κλάσης). Το DL στο όνομα της OWL DL, προέρχεται από το Description Logics, το πεδίο έρευνας που εξετάζει τη λογική πίσω από τη φορμαλιστική θεμελίωση της OWL. Το βασικό πλεονέκτημα της OWL DL είναι ότι επιτρέπει την αποτελεσματική υποστήριξη συλλογισμών (efficient reasoning support). Το μειονέκτημα είναι ότι χάνεται η πλήρης συμβατότητα με το RDF.

2.1.3 OWL Full

Η OWL Full προορίζεται για χρήστες που επιθυμούν μέγιστη εκφραστικότητα και τη συντακτική ελευθερία που παρέχει το RDF, χωρίς να παρέχονται εγγυήσεις σχετικά με την υπολογιστική πληρότητα. Η OWL Full, δίνει τη δυνατότητα σε μία οντολογία να ενισχύει το νόημα του προκαθορισμένου (RDF ή OWL) λεξιλογίου. Είναι σχετικά απίθανο, ότι θα υπάρξει λογισμικό που να μπορεί να υποστηρίζει πλήρη συλλογισμό για κάθε χαρακτηριστικό της OWL Full. Το πλεονέκτημά της είναι ότι είναι πλήρως συμβατή με το RDF. Το μειονέκτημα είναι ότι λόγω της εκφραστικότητας και της δύναμης που παρέχει, γίνεται πολύ δύσκολη η πλήρης (ή αποτελεσματική) υποστήριξη συλλογισμών.

2.2 ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΣΤΟΙΧΕΙΑ ΤΗΣ ΓΛΩΣΣΑΣ OWL

Η OWL βασίζεται στις γλώσσες RDF και RDF Schema και χρησιμοποιεί τη σύνταξη της RDF που βασίζεται στην XML. Τα περισσότερα στοιχεία μιας OWL οντολογίας αφορούν κλάσεις, ιδιότητες, στιγμιότυπα αυτών των κλάσεων και συσχετίσεις μεταξύ αυτών των στιγμιότυπων.

2.2.1 Στοιχεία Κλάσεων (Classes)

Οι κλάσεις (Classes) ορίζονται με τη χρήση ενός στοιχείου owl:Class. Μπορούμε να δηλώσουμε ότι η κλάση αυτή είναι ξένη ως προς άλλες κλάσεις κάνοντας χρήση στοιχείων owl:disjointWith ή μπορούμε να δηλώσουμε ισοδυναμία κλάσεων μέσω του στοιχείου owl:equivalentClass. Επίσης υπάρχουν δύο προκαθορισμένες κλάσεις οι owl:Thing η owl:Nothing. Η πρώτη είναι η πιο γενική κλάση που περιέχει τα πάντα και η τελευταία είναι η κενή κλάση. Επομένως κάθε κλάση είναι υποκλάση της owl:Thing και υπερκλάση της owl:Nothing.

Ο ορισμός της υποκλάσης σχετίζει μια συγκεκριμένη κλάση σε μια περισσότερο γενική κλάση. Αν X είναι μα υποκλάση του Y, τότε κάθε στιγμιότυπο της X είναι επίσης και στιγμιότυπο της Y. Επίσης, η σχέση της υποκλάσης είναι μεταβατική, που σημαίνει ότι αν X είναι μια υποκλάση της Y και Y μια υποκλάση της Z τότε η X είναι και υποκλάση της Z.Η δήλωση μιας κλάσης έχει δύο μέρη: ένα εισαγωγικό όνομα ή παραπομπή και μια λίστα με περιορισμούς. Κάθε μία από τις άμεσα περιεχόμενες εκφράσεις στην καθορισμένη κλάση περιορίζει παραπέρα τα στιγμιότυπα της δηλωθέντας κλάσης. Τα στιγμιότυπα της κλάσης ανήκουν στην τομή των περιορισμών, με τις περισσότερες οντολογίες να έχουν τουλάχιστον ένα περιορισμό, που αναγκάζει μια καινούργια κλάση να είναι υποκλάση κάποιας άλλης επονομαζόμενης κλάσης.Αν και οι παραπάνω κλάσεις είναι περιορισμένες και μπορούν να χαρακτηρίσουν μόνο μια ατελής οντολογία, οι πληροφορίες που παρέχονται είναι και πάλι αρκετές για τη δημιουργία και το συλλογισμό ατόμων.

2.2.2 Στοιχεία Ατόμων (Individuals)

Σε αντίθεση με τις κλάσεις, θέλουμε να είναι δυνατή και η περιγραφή των μελών αυτών των στοιχείων. Στο δικό μας κόσμο πραγμάτων θεωρούμε αυτά τα στοιχεία ως άτομα. Ένα άτομο αποκτάει ύπαρξη μέσω της δήλωσής του ως μέλος σε μια κλάση. Μια κλάση είναι απλά ένα όνομα και μια συλλογή από ιδιότητες που περιγράφουν συσχετίσεις ανάμεσα σε ένα σύνολο ατόμων. Τα άτομα είναι τα μέλη αυτού του συνόλου.

2.2.3 Στοιχεία Ιδιοτήτων (Properties)

Στην owl υπάρχουν δύο είδη ιδιοτήτων. Οι ιδιότητες αντικειμένου (object properties) και οι ιδιότητες τύπου δεδομένων (datatype properties). Οι datatype properties που αφορούν σχέσεις ανάμεσα σε στιγμιότυπα κλάσεων και RDF literals και XML Schema datatypes και οι object properties που αφορούν σχέσεις μεταξύ στιγμιότυπων δύο κλάσεων. Κατά τη δήλωση μια ιδιότητας χρησιμοποιείται μια μέθοδος για τον περιορισμό της σχέσης, η οποία καθορίζεται μέσω του domain και του range.

Μια ιδιότητα X που έχει ως domain την κλάση Y και ως range την κλάση Z, έχει ως αποτέλεσμα να συσχετίσει τα στιγμιότυπα της κλάσης Y με τα στιγμιότυπα της κλάσης Z. Πολλαπλά domains σημαίνουν ότι το domain της ιδιότητας αποτελεί την τομή των αναφερόμενων κλάσεων όπως επίσης ισχύει το ίδιο και για το range.

Παράδειγμα χρήσης object property:

```
<owl:ObjectProperty rdf:ID="vintageOf">
  <rdfs:domain rdf:resource="#Vintage" />
  <rdfs:range  rdf:resource="#Wine" />
</owl:ObjectProperty>
```

Παράδειγμα χρήσης datatype property:

```
<owl:DatatypeProperty rdf:ID="yearValue">
  <rdfs:domain rdf:resource="#VintageYear" />
  <rdfs:range  rdf:resource="&xsd;positiveInteger"/>
</owl:DatatypeProperty>
```

2.2.4 Τύποι δεδομένων

Παρόλο που η γλώσσα XML Schema παρέχει ένα μηχανισμό δημιουργίας τύπων δεδομένων που ορίζονται από το χρήστη (π.χ. ο τύπος δεδομένων adultAge που περιλαμβάνει όλους τους ακέραιους που είναι μεγαλύτεροι από 18 ή ο τύπος δεδομένων όλων των αλφαριθμητικών που αρχίζουν με αριθμό),τέτοιοι παράγωγοι τύποι δεδομένων δεν μπορούν να χρησιμοποιηθούν στην OWL. Στην πραγματικότητα, το ίδιο ισχύει και για πολλούς ενσωματωμένους τύπους δεδομένων της XML Schema http://www.w3.org/2001/XMLSchema . Ο παρακάτω πίνακας αναφέρει όλους τους τύπους δεδομένων που επιτρέπονται .

xsd:string	xsd:time
xsd:decimal	xsd:gMonthDay
xsd:integer	xsd:token
xsd:nonPositiveInteger	xsd:Name
xsd:long	xsd:boolean
xsd:unsignedLong	xsd:double
xsd:hexBinary	xsd:positiveInteger
xsd:dateTime	xsd:short
xsd:gYear	xsd:unsignedShort
xsd:anyURI	xsd:date
xsd:NMTOKEN	xsd:gDay
xsd:normalizedString	xsd:language
xsd:float	xsd:NCName
xsd:nonNegativeInteger	xsd:byte
xsd:negativeInteger	xsd:unsignedByte
xsd:int	xsd:gYearMonth
xsd:unsignedInt	xsd:gMonth
xsd:base64Binary	rdfs:Literal

Πίνακας 1-Datatype properties

2.2.5 Περιορισμοί ιδιοτήτων

Αντί να καθορίσουμε χαρακτηριστικά στις ιδιότητες, είναι δυνατό να περιορίσουμε περαιτέρω το range μιας ιδιότητας σε συγκεκριμένα πλαίσια με μια ποικιλία διαφορετικών τρόπων. Αυτό είναι εφικτό μέσω των περιορισμών των ιδιοτήτων. Όλοι αυτοί οι περιορισμοί γίνονται μέσα στα πλαίσια του owl:Restriction[3] και είναι οι παρακάτω:

- `owl:AllValuesFrom`, ο συγκεκριμένος περιορισμός καθορίζει ότι για κάθε στιγμιότυπο μιας κλάσης που έχει στιγμιότυπα μιας συγκεκριμένης ιδιότητας, όλες οι δυνατές τιμές αυτής της ιδιότητας αποτελούν μέλη της κλάσης που προσδιορίζεται από το owl:allValuesFrom. Πρόκειται για τοπικό μόνο περιορισμό καθώς χρησιμοποιείται για τον ορισμό της κλάσης των δυνατών τιμών που μπορεί να πάρει η ιδιότητα που καθορίζεται από το στοιχείο owl:onProperty (με άλλα λόγια όλες οι τιμές της ιδιότητας πρέπει να προέρχονται από αυτή την κλάση)

- `owl:hasValue`, χρησιμοποιείται για να καθορίσουμε κλάσεις που είναι βασισμένες στην ύπαρξη μιας συγκεκριμένης τιμής μιας ιδιότητας και καθορίζεται από το στοιχείο owl:onProperty Οπότε, ένα άτομο θα είναι μέλος μιας τέτοιας κλάσης αν οποιαδήποτε στιγμή ισχύει ότι τουλάχιστον μια από τις τιμές αυτής της ιδιότητας είναι ίση με την τιμή του hasValue.

- `owl:someValuesFrom`, ο συγκεκριμένος περιορισμός καθορίζει ότι για κάθε στιγμιότυπο μιας κλάσης που έχει στιγμιότυπα μιας συγκεκριμένης ιδιότητας, η ιδιότητα αυτή θα έχει ως τιμή τουλάχιστον ένα ή περισσότερα μέλη της κλάσης που προσδιορίζεται από το owl:allValuesFrom.

- `owl:Cardinality`, ο συγκεκριμένος περιορισμός προσδίδει ισχυρούς καθορισμούς για ένα αριθμό στοιχείων σε ένα σύνολο ή και ακριβώς ένα καθορισμό στοιχείου από αυτό το σύνολο. Συνηθίζεται να χρησιμοποιείται σε functional ιδιότητες επιβάλλοντας συγκεκριμενοποίηση του στοιχείου. Οι εκφράσεις cardinality που περιέχουν τιμές περιορίζονται στις 0 και 1 και αποτελούν μέρος του OWL Lite. Αυτό επιτρέπει στο χρήστες να επισημαίνουν εκφράσεις που αντιστοιχούν στα λογικά “τουλάχιστον ένα”, “όχι περισσότερο από ένα” και “ακριβώς ένα”. Θετικές ακέραιες τιμές εκτός από το 0 και 1 επιτρέπονται στην OWL DL. Αξιοποιώντας το owl:maxCardinality μπορούμε να καθορίσουμε ένα ανώτατο επιτρεπόμενο όριο, ενώ με το owl:minCardinality μπορούμε να καθορίσουμε το κατώτερο επιτρεπόμενο όριο. Συνδυασμός αυτών το δύο εκφράσεων οδηγεί στην δημιουργία αριθμητικού ορίου στην εκάστοτε ιδιότητα

[3] Owl:Restriction: Ορίζει μία ανώνυμη κλάση η οποία δεν έχει ID,δεν ορίζεται από το στοιχείο owl:Class και έχει μόνο τοπική εμβέλεια. Μπορεί να χρησιμοποιηθεί στο μοναδικό σημείο που εμφανίζεται ο περιορισμός.

2.2.6 Ειδικές ιδιότητες

Μερικές ιδιότητες των στοιχείων ιδιοτήτων είναι:

- Η ***owl:TransitiveProperty*** που ορίζει μια μεταβατική ιδιότητα. π.χ. is_ancestor_of (είναι πρόγονός του)
- Η ***owl:SymmetricProperty*** που ορίζει μια συμμετρική ιδιότητα. π.χ. has_same_grade_as (έχει την ίδια βαθμολογία με)
- Η ***owl:InverseFunctionalProperty*** που ορίζει μια ιδιότητα για την οποία δύο διαφορετικά αντικείμενα δεν μπορούν να έχουν την ίδια τιμή π.χ. Α.Φ.Μ.
- Η ***owl:FunctionalProperty*** που ορίζει μια ιδιότητα που έχει το πολύ μια τιμή για κάθε αντικείμενο, π.χ. age (ηλικία)

2.2.7 Λογικοί Συνδυασμοί

Η ένωση κλάσεων δημιουργείται χρησιμοποιώντας το στοιχείο `owl:unionOf`

```
<owl:Class>
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Door"/>
    <owl:Class rdf:about="#Furniture"/>
    <owl:Class rdf:about="#Window"/>
  </owl:unionOf>
</owl:Class>
```

Η τομή με στοιχείο δημιουργείται χρησιμοποιώντας το στοιχείο `owl:intersectionOf`

```
<owl:Class rdf:ID="facultyInCS">
  <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#faculty"/>
          <owl:Restriction>
                  <owl:onProperty rdf:resource="#belongsTo"/>
                  <owl:hasValue rdf:resource="#CSDepartment"/>
          </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

2.2.8 Απαρίθμηση (Enumerated Class)

Η απαρίθμηση (enumeration) είναι ένα στοιχείο `owl:one of` που χρησιμοποιείται για τον ορισμό μιας κλάσης με παράθεση άλλων στοιχείων της:

```
<owl:Class rdf:ID="weekdays">
  <owl:oneOf rdf:parseType="Collection">
          <owl:Thing rdf:about="#Monday"/>
          <owl:Thing rdf:about="#Tuesday"/>
          <owl:Thing rdf:about="#Wednesday"/>
          <owl:Thing rdf:about="#Thursday"/>
          <owl:Thing rdf:about="#Friday"/>
          <owl:Thing rdf:about="#Saturday"/>
          <owl:Thing rdf:about="#Sunday"/>
  </owl:oneOf>
</owl:Class>
```

2.2.9 Έκδοση οντολογίας

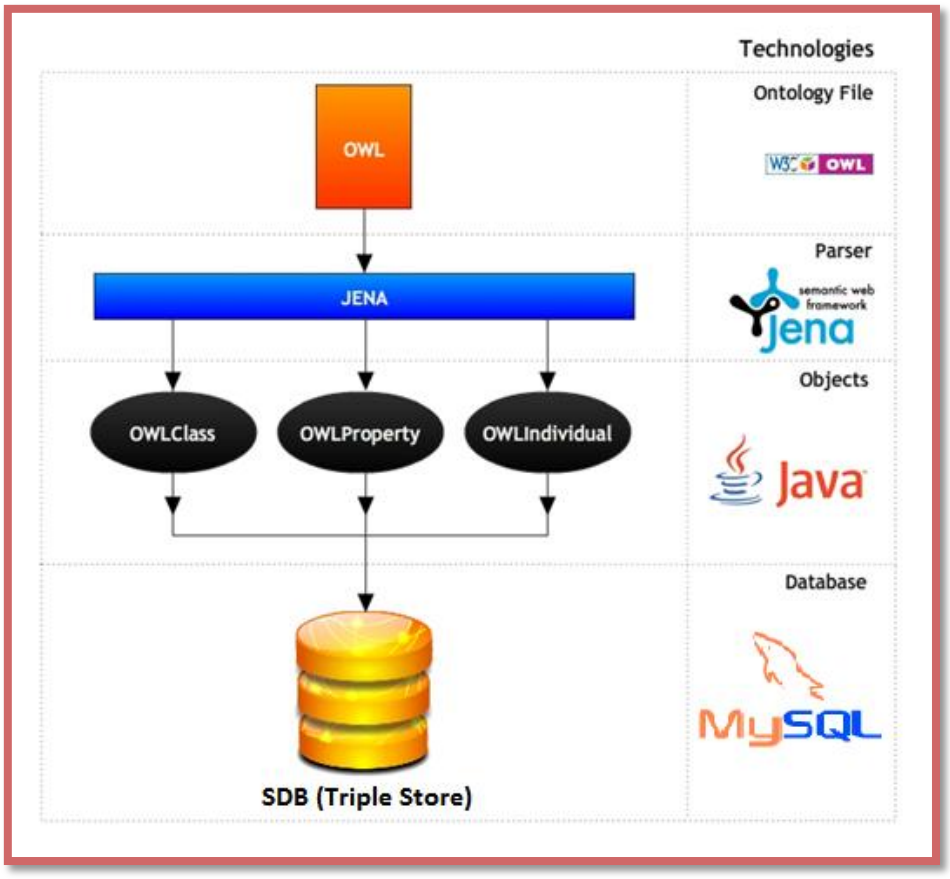
Οι οντολογίες είναι σαν ένα λογισμικό, κάποια στιγμή θα χρειαστούν συντήρηση, πράγμα που θα οδηγήσει στην αλλαγή τους. Η OWL διαθέτει τέσσερις προτάσεις για την ένδειξη πληροφοριών έκδοσης

- Η πρόταση `owl:priorVersion` χρησιμοποιείται ως μέρος των πληροφοριών της επικεφαλίδας για την έκδοση παλιότερων εκδόσεων της τρέχουσας οντολογίας. Οι εκδόσεις μιας οντολογίας μπορεί να είναι ασύμβατες μεταξύ τους. Παραδείγματος χάριν, μια προηγούμενη έκδοση μιας οντολογίας μπορεί να περιέχει δηλώσεις οι οποίες αντικρούονται με την τρέχουσα έκδοση. Εκτός από την owl:priorVersion, η OWL διαθέτει τρεις ακόμα προτάσεις για την ένδειξη περισσότερων ανεπίσημων πληροφοριών έκδοσης. Καμία από αυτές δεν έχει κάποια ουσιαστικό νόημα
- Η `owl:versionInfo` περιέχει γενικά ένα αλφαριθμητικό με πληροφορίες σχετικά με την τρέχουσα έκδοση.
- Η `owl:backwardCompatibleWith` περιέχει αναφορά σε κάποια άλλη οντολογία. Αυτό προσδιορίζει την καθορισμένη οντολογία ως προηγούμενη έκδοση της οντολογίας στην οποία περιέχεται και υποδεικνύει επιπλέον ότι είναι συμβατή προς τα πίσω με αυτή. Συγκεκριμένα αυτό υποδηλώνει ότι όλα τα αναγνωριστικά από την προηγούμενη έκδοση έχουν τις ίδιες ερμηνείες και στη νέα έκδοση. Επομένως είναι μία έκδοση προς τους συγγραφείς των εγγράφων ότι μπορούν να τροποποιήσουν τα έγγραφά τους με ασφάλεια.
- Η `owl:incompatibleWith` από την άλλη πλευρά υποδηλώνει ότι η αναφερόμενη οντολογία αποτελεί προγενέστερη έκδοση της οντολογίας στην οποία περιέχεται αλλά δεν είναι συμβατή προς τα πίσω με αυτήν. Ουσιαστικά αυτό χρησιμεύει στους συγγραφείς οντολογιών οι οποίοι θέλουν να δηλώσουν ρητά ότι τα έγγραφα δεν μπορούν να αναβαθμιστούν για να χρησιμοποιούν την νέα έκδοση χωρίς προηγουμένως να γίνει έλεγχος αν απαιτούνται αλλαγές.

3 ΤΕΧΝΟΛΟΓΙΕΣ ΥΛΟΠΟΙΗΣΗΣ ΠΤΥΧΙΑΚΗΣ

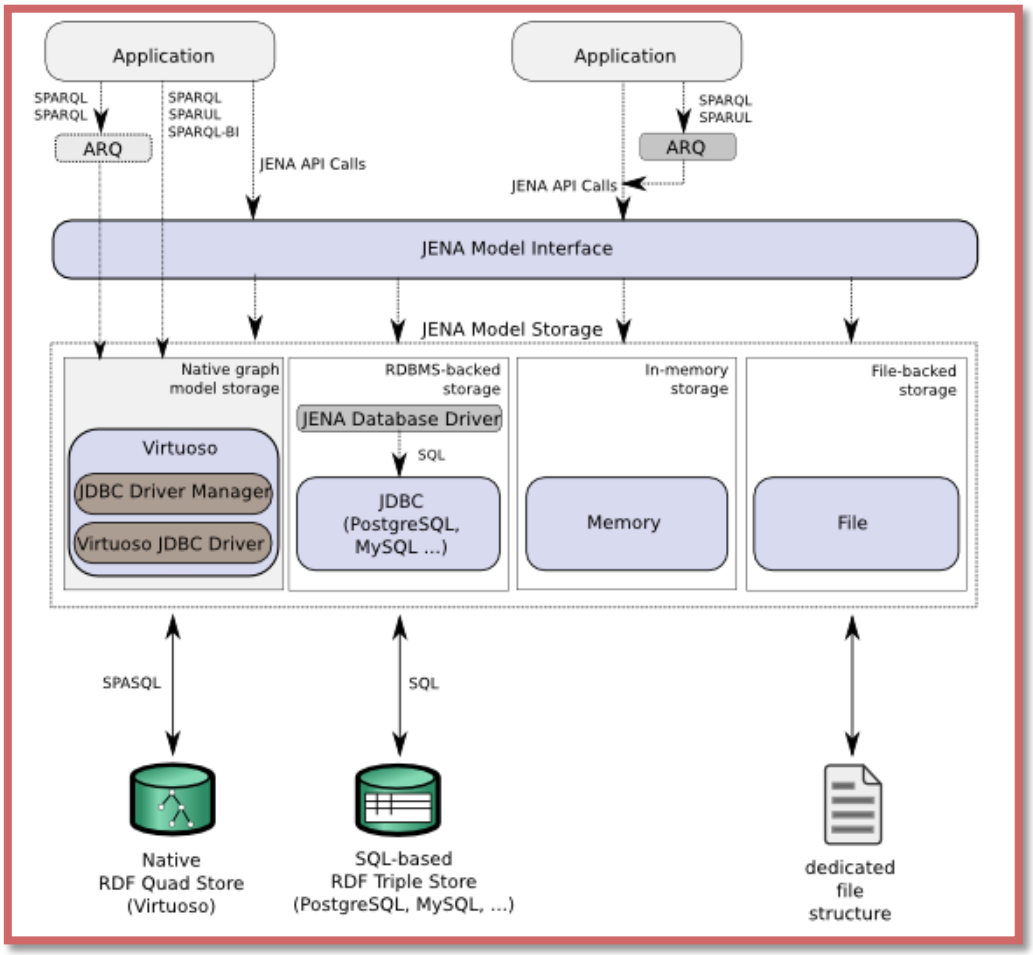
Σε αυτό το σημείο θα παρουσιαστούν οι βασικές τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν για την υλοποίηση αυτής της πτυχιακής. Η εφαρμογή βασίστηκε στην owl οντολογία που δημιουργήθηκε μέσω του προγράμματος protégé και έπειτα μέσω του framework Jena και της java επιτεύχθηκε η φόρτωση αλλά και αποθήκευση αυτής της οντολογίας με τη μορφή τριπλετών σε σχεσιακή βάση δεδομένων (SDB) με τη βοήθεια βάσης SQL. Στη συνέχεια χρησιμοποιήθηκε η τεχνολογία της sparql για την εκτέλεση σημασιολογικών ερωτημάτων στη βάση.

- Τεχνολογίες που χρησιμοποιήθηκαν για την επίτευξη της λειτουργίας σε επίπεδο admin



Εικόνα 3-Τεχνολογίες υλοποίησης πτυχιακής (Επίπεδο Admin)

- Τεχνολογίες που χρησιμοποιήθηκαν για την επίτευξη της λειτουργίας σε επίπεδο user



Εικόνα 4-Τεχνολογίες υλοποίησης πτυχιακής (Επίπεδο User)

Πηγή εικόνας: http://docs.openlinksw.com/images/ui/VirtJenaProvider.png

Στις επόμενες ενότητες θα αναλυθούν όλες οι παραπάνω τεχνολογίες και στη συνέχεια θα παρατεθεί και ο κώδικας υλοποίησης της εφαρμογής σε επίπεδο διαχειριστή (admin) αλλά και σε επίπεδο χρήστη (user).

Συνοπτικά στις επόμενες ενότητες θα δούμε:

- PROTÉGÉ
- JAVA
- APACHE TOMCAT
- JSP
- CSS
- MySQL
- JENA FRAMEWORK- SDB
- JAVASCRIPT
- DOM
- SPARQL

3.1 Protégé

Πηγή εικόνας:
http://protege.stanford.edu/download/protege/4.1/installanywhere/Web_Installers/InstData/com/zerog/ia/installer/images/Splash.gif

Το Protégé αποτελεί μια πλατφόρμα «ανοιχτού κώδικα» (open source) βασισμένη σε JAVA που παρέχει ένα σύνολο από εργαλεία για την μοντελοποίηση πεδίων γνώσης και την ανάπτυξη και επεξεργασία οντολογιών. Πρωτοεμφανίστηκε το 1988 και στην αρχή αποτελούσε απλώς ένα μέσο για τη δημιουργία εργαλείων ανάκτησης γνώσης για έμπειρα συστήματα. Σήμερα το Protégé έχει εξελιχθεί σε ένα σύγχρονο εργαλείο μοντελοποίησης γνώσης, το οποίο εξελίσσεται με ταχύτατους ρυθμούς αφού σύμφωνα με στοιχεία υπάρχουν 200.000 εγγεγραμμένοι χρήστες. Το βασικό τμήμα του Protégé παρέχει ένα πλούσιο σύνολο από δομές μοντελοποίησης γνώσης και λειτουργίες που υποστηρίζουν τη δημιουργία, απεικόνιση και επεξεργασία οντολογιών. Μια οντολογία προσδιορίζει τις έννοιες και τις συσχετίσεις που είναι σημαντικές σε ένα συγκεκριμένο πεδίο γνώσης, παρέχοντας ένα λεξικό για το πεδίο αυτό. Τα τελευταία χρόνια, οι οντολογίες συναντώνται σε αρκετές επιστημονικές και επαγγελματικές κοινότητες ως μέσο διαμερισμού, επαναχρησιμοποίησης και επεξεργασίας γνώσης σε συγκεκριμένα πεδία.

Στην εργασία χρησιμοποιήθηκε μια σημασιολογική οντολογία σχεδίασης και διακόσμησης εσωτερικών χώρων. Η συγκεκριμένη οντολογία εκτός από τα κύρια χαρακτηριστικά ενός αντικειμένου, περιέχει και κάποια άλλα που έχουν πιο αφηρημένη έννοια όπως η λειτουργικότητα χώρου, το στυλ του δωματίου, κτλ. Η οντολογία αναπτύχθηκε με τέτοιο τρόπο ώστε να είναι δυνατή η αναλυτική περιγραφή ενός εσωτερικού χώρου, ενώ ταυτόχρονα η δομή της, επιτρέπει την περιγραφή οποιουδήποτε τύπου δωματίου ενός εσωτερικού χώρου. Η οντολογία αναπτύχθηκε εξ' ολοκλήρου πάνω στην OWL χρησιμοποιώντας το ***Protégé-OWL 3.4.4***. Η συγκεκριμένη οντολογία περιέχει πληροφορίες, που καλύπτουν όλο το φάσμα εκείνων των συντελεστών, που κρίνονται απαραίτητοι για την ακριβή περιγραφή ενός εσωτερικού χώρου. Συγκεκριμένα στην οντολογία μπορούμε να ορίσουμε ποια είναι τα αντικείμενα που υπάρχουν στον χώρο, ποια είναι η μορφή τους, που τοποθετούνται αυτά τα αντικείμενα μέσα στον χώρο σε σχέση με άλλα αντικείμενα και ποια είναι η "προσωπικότητα" του χώρου.

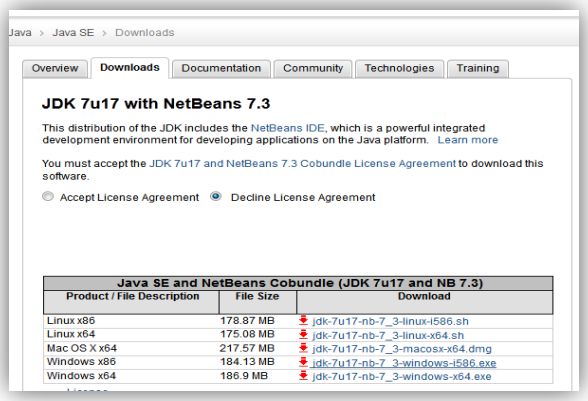
3.2 Java

Πηγές εικόνων: http://www.xblog.gr/wp-content/uploads/2008/02/java-logo.jpg και http://www.kidsil.net/wp-content/uploads/2011/03/netbeans_logo_ok-300x150.jpg

Η Java δημιουργήθηκε το 1991 από τον James Gosling κ.ά. στη Sun Microsystems. Αρχικά, ονομάστηκε Oak. Αρχικός στόχος ήταν η ανάπτυξη μίας γλώσσας που θα ήταν ανεξάρτητη πλατφόρμας, δηλαδή εύκολα θα «έπαιζε» παντού. Λόγω της ανάπτυξης του Διαδικτύου, η Java βρήκε πρόσφορο έδαφος για ανάπτυξη εφαρμογών. Τα προγράμματα που είναι γραμμένα σε Java τρέχουν ακριβώς το ίδιο σε Windows, Linux, Unix χωρίς να χρειαστεί να γίνει μεταγλώττιση (compiling) ή να αλλάξει ο πηγαίος κώδικας για κάθε διαφορετικό λειτουργικό σύστημα. Στις 27 Απριλίου 2010 η εταιρία λογισμικού Oracle Corporation ανακοίνωσε ότι μετά από πολύμηνες συζητήσεις ήρθε σε συμφωνία για την εξαγορά της Sun Microsystems και των τεχνολογιών (πνευματικά δικαιώματα/ πατέντες) που η δεύτερη είχε στην κατοχή της ή δημιουργήσει. Η συγκεκριμένη συμφωνία θεωρείται σημαντική για το μέλλον της Java και του γενικότερου οικοσυστήματος τεχνολογιών γύρω από αυτή μιας και ο έμμεσος έλεγχος της τεχνολογίας και η εξέλιξη της περνάει σε άλλα χέρια.

Το NetBeans είναι ένα επιτυχημένο ερευνητικό έργο ανοικτού κώδικα (open source) με μεγάλο αριθμό χρηστών. Η Sun Microsystems ίδρυσε το ερευνητικό έργο ανοιχτής πηγής NetBeans τον Ιούνιο του 2000.Το NetBeans IDE είναι ένα περιβαλλοντικό ανάπτυγμα IDE - ένα εργαλείο στους προγραμματιστές για να γράψουν, να κάνουν compile, debug και να αναπτύξουν προγράμματα. Είναι γραμμένο σε Java - αλλά μπορεί να υποστηρίξει όλες τις γλώσσες προγραμματισμού. Υπάρχει επίσης ένας μεγάλος αριθμός υπομονάδων (modules) που βοηθάνε στην επέκταση της λειτουργικότητας του NetBeans IDE. Το NetBeans IDE είναι ένα ελεύθερο προϊόν δίχως περιορισμούς στον τρόπο χρησιμοποίησής του.

Στα πλαίσια αυτής της πτυχιακής η έκδοση της java ***(JDK)*** που χρησιμοποιήθηκε είναι η ***7*** σε συνδυασμό με το ***Netbeans 7.3***


Java › Java SE › Downloads

Overview | Downloads | Documentation | Community | Technologies | Training

JDK 7u17 with NetBeans 7.3

This distribution of the JDK includes the NetBeans IDE, which is a powerful integrated development environment for developing applications on the Java platform. Learn more

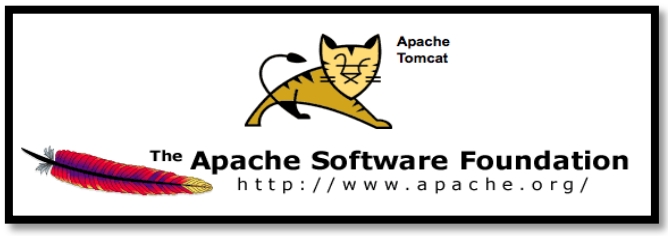
You must accept the JDK 7u17 and NetBeans 7.3 Cobundle License Agreement to download this software.

Accept License Agreement Decline License Agreement

Java SE and NetBeans Cobundle (JDK 7u17 and NB 7.3)		
Product / File Description	File Size	Download
Linux x86	178.87 MB	jdk-7u17-nb-7_3-linux-i586.sh
Linux x64	175.08 MB	jdk-7u17-nb-7_3-linux-x64.sh
Mac OS X x64	217.57 MB	jdk-7u17-nb-7_3-macosx-x64.dmg
Windows x86	184.13 MB	jdk-7u17-nb-7_3-windows-i586.exe
Windows x64	186.9 MB	jdk-7u17-nb-7_3-windows-x64.exe

Εικόνα 5-Έκδοση Netbeans και JDK

3.3 Apache Tomcat



Πηγή εικόνας: http://dipologcity.gov.ph/data/apache-tomcat-startup.png

Το εργαλείο Apache Tomcat είναι μια ανοιχτού κώδικα υλοποίηση της τεχνολογίας των σελίδων διακομιστών Java (Servlet και JSP). Προσφέρει ώθηση στην ανάπτυξη κρίσιμων και μεγάλης κλίμακας δικτυακών εφαρμογών σε επιχειρήσεις και οργανισμούς.

Με το εργαλείο παρέχεται η δυνατότητα:

- ρύθμισης του διαχειριστή ασφάλειας της Java
- χρησιμοποίησης πηγών δεδομένων μέσω JDBC
- υποστήριξης του πρωτοκόλλου SSL για κρυπτογραφημένες συνδέσεις
- χρήσης διαμεσολαβητών (proxy)
- εικονικής λειτουργίας (virtual hosting)
- εκτέλεσης σεναρίων CGI.

Η έκδοση ***apache-tomcat*** που χρησιμοποιήθηκε για αυτή την εργασία είναι η ***7.0.39***

3.4 JSP

Πηγή εικόνας: http://www.zkoss.org/resource/img/product/page/zkjsp-banner_image.png

Τα Web sites διαθέτουν μια εφαρμογή, που αποκαλείται εφαρμογή του Web (Web application), η οποία δέχεται τις αιτήσεις (requests) του χρήστη (επισκέπτη) του site και δημιουργεί μια απάντηση (response) που να ταιριάζει με το κάθε συγκεκριμένο ερώτημα (query) που υποβάλλεται. Επειδή η εφαρμογή του Web δημιουργεί αυτές τις ιστοσελίδες στον αέρα (on-the-fly), λέμε ότι επιστρέφει δυναμικό περιεχόμενο (dynamic content). Αυτό σημαίνει ότι η απόκριση απ' αυτά τα sites, όπως είναι η εμφάνιση πληροφοριών για τα διάφορα τουριστικά μέρη, δημιουργείται δυναμικά κάθε φορά που καταχωρείται ένα συγκεκριμένο ερώτημα.

Οι JSPs (JavaServer Pages) είναι μια τεχνολογία που έχει δημιουργηθεί από την εταιρεία Sun Microsystems για να μπορεί να δημιουργεί δυναμικό περιεχόμενο (dynamic content) στο Web. Πρόκειται για HTML έγγραφα (ιστοσελίδες) τα οποία αναμειγνύονται με

τη γλώσσα προγραμματισμού Java, η οποία και έχει τη δυνατότητα να δημιουργεί αυτό το δυναμικό περιεχόμενο. Οι JSPs είναι μια εφαρμογή στην πλευρά του server (server-side application), που σημαίνει ότι δέχονται μια αίτηση (request) και παράγουν μια απόκριση ή απάντηση (response). Σε γενικές γραμμές, οι αιτήσεις γίνονται από έναν Web client και η απόκριση είναι ένα παραγόμενο HTML έγγραφο (ιστοσελίδα) το οποίο στέλνεται πίσω στον Web client. Επειδή οι JSPs είναι μια εφαρμογή στην πλευρά του server, έχουν πρόσβαση σε πηγές (resources) στον server, όπως είναι τα Servlets, JavaBeans, EJBs, αλλά και σε βάσεις δεδομένων.

3.5 CSS

Πηγή εικόνας: https://encrypted-tbn2.gstatic.com/images?q=tbn:ANd9GcT70F9NcfT2u-XhXHTvZmvygUe-_FimyfN7ZWjycALT7ledtz86

Το CSS είναι μια απλή γλώσσα που μας βοηθάει να ορίσουμε με σαφήνεια και ιδιαίτερη ευελιξία τον τρόπο με τον οποίο θα εμφανίζονται τα διάφορα στοιχεία στην ιστοσελίδα μας. Χρησιμοποιείται δηλαδή για τον έλεγχο της εμφάνισης ενός εγγράφου που γράφτηκε σε γλώσσα HTML. Η CSS είναι μια γλώσσα υπολογιστή προορισμένη να αναπτύσσει στυλιστικά μια ιστοσελίδα δηλαδή να διαμορφώνει περισσότερα χαρακτηριστικά, χρώματα, στοίχιση και δίνει περισσότερες δυνατότητες σε σχέση με την html. Για μια όμορφη και καλοσχεδιασμένη ιστοσελίδα η χρήση της CSS κρίνεται απαραίτητη. Το CSS παρέχει:

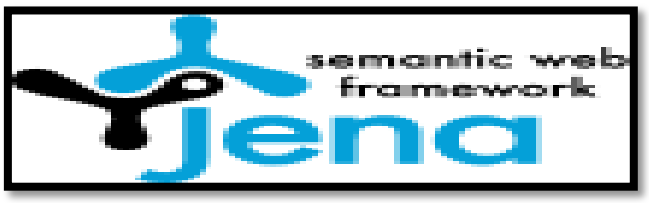
- Πολύ μεγαλύτερη ευελιξία αφού κατέστησε εφικτές μορφοποιήσεις οι οποίες ήταν αδύνατες ή πολύ δύσκολες με την κλασσική HTML.
- Μικρότερο μέγεθος αρχείου,δεδομένου ότι ο κάθε κανόνας μορφοποίησης γράφεται μόνο μια φορά και όχι σε κάθε σημείο που εφαρμόζεται.
- Ευκολότερη συντήρηση των ιστοσελίδων.Η εμφάνιση ενός ολόκληρου site μπορεί να ελέγχεται από ένα μόνο εξωτερικό αρχείο CSS.Έτσι, κάθε αλλαγή στο στυλ της ιστοσελίδας μπορεί να γίνεται με μια μοναδική αλλαγή σε αυτό το αρχείο,αντί για την επεξεργασία πολλών σημείων σε κάθε σελίδα που υπάρχει στο site.
- Καλύτερο SEO (Searchengineoptimization). Οι μηχανές αναζήτησης δεν «μπερδεύονται» ανάμεσα σε περιεχόμενο και τη μορφοποίηση του, αλλά έχουν πρόσβαση στο περιεχόμενο μόνο, οπότε είναι πολύ ευκολότερο να το καταγράψουν και να το αρχειοθετήσουν (indexing).

3.6 MySQL

Πηγή εικόνας: http://www.mysql.com/common/logos/logo-mysql-170x115.png

Η MySQL είναι το πιο αξιόπιστο σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων (RDBMS) ανοικτού κώδικα που χρησιμοποιείται σήμερα. Το γεγονός ότι 9 στις 10 πιο δημοφιλείς εμπορικές ιστοσελίδες στον κόσμο βασίζονται σε MySQL κατά κύριο λόγο οφείλεται στην συνέπεια της σε ετερογενείς πλατφόρμες, τις επιδόσεις, την αξιοπιστία και την ευκολία χρήσης της. Ο MySQL διακομιστής ελέγχει την πρόσβαση στα δεδομένα παρέχοντας πρόσβαση σε πιστοποιημένους χρήστες, δίνοντας τη δυνατότητα να δημιουργεί, να χειρίζεται και να συντηρεί τη βάση. Τα RDBMS χρησιμοποιούνται συνήθως σε μεγάλα διαχειριστικά συστήματα με πολλά δεδομένα απλού τύπου και μικρή διάρκεια συναλλαγών. Παρέχουν βασικούς σχεσιακούς πίνακες, σύνολα εγγραφών ορισμένα από το χρήστη με τύπους γνωστούς από το σύστημα (int, text κλπ).

3.7 Jena

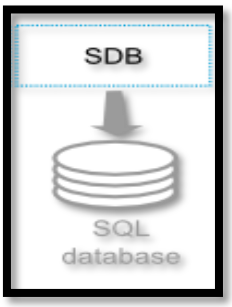


Πηγή εικόνας: http://jena.sourceforge.net/images/jena-logo-small.png

Η Jena είναι μια προγραμματιστική διεπαφή σε java για την ανάπτυξη εφαρμογών σημασιολογικού χαρακτήρα.

- Υλοποιεί API για την χειρισμό οντολογιών που βρίσκονται σε μορφές RDF και OWL.
- Υλοποιεί API για την ανάγνωση, την επεξεργασία και την εγγραφή RDF δεδομένων σε μορφές XML,N-triples και turtles.
- Υλοποιεί ένα κανόνα εξαγωγής συμπερασμάτων για OWL και RDF.
- Παρέχει ένα μηχανισμό ερωτημάτων συμβατό με την SPARQL.
- Η Jena διαθέτει μηχανισμό που επιτρέπει σε μεγάλες οντολογίες να αποθηκεύονται με τη μορφή τριπλετών στο δίσκο καθώς προβλέπει κλιμακούμενη αποθήκευση και αναζήτηση RDF δεδομένων με τη χρήση συμβατικών βάσεων δεδομένων SQL τόσο ανοικτού όσο και κλειστού κώδικα για χρήση σε αυτόνομες εφαρμογές, J2EE.
- Θεμελιώδη κατηγορία της Jena για τους χρήστες είναι το μοντέλο. Ένα μοντέλο μπορεί να δημιουργηθεί από το σύστημα αρχείων ή από ένα απομακρυσμένο αρχείο. Χρησιμοποιώντας JDBC, μπορεί να συνδέεται με μια υπάρχουσα RDBMS, όπως η MySQL ή η PostgreSQL.

3.7.1 SDB



Πηγή εικόνας: http://jena.apache.org/images/jena-architecture.png

Η Jena SDB είναι ένα συστατικό μέρος του framework της Jena και υποστηρίζει αποθήκευση σε RDF δεδομένα αλλά και ερωτήματα χρησιμοποιώντας σχεσιακές βάσεις δεδομένων μέσω του μηχανισμού της SPARQL. Οι υποστηριζόμενες βάσεις δεδομένων είναι οι: Microsoft SQL Server, Oracle 10g, IBM DB2, PostgreSQL, MySQL, HSQLDB, H2 και Apache Derby. Η Jena SDB είναι ανοικτού κώδικα και τρέχει ως server ο οποίος μπορεί να χρησιμοποιηθεί σε εφαρμογές που κάνουν χρήση της βιβλιοθήκης της Jena.

3.7.2 RDF Triple Stores

Η αποθήκευση τριπλετών είναι ένα framework που χρησιμοποιείται για την αποθήκευση και την εκτέλεση ερωτημάτων RDF δεδομένων. Παρέχει ένα μηχανισμό μόνιμης αποθήκευσης και πρόσβασης RDF γράφων. Πρόσφατα έχει υπάρξει μια σημαντική πρωτοβουλία στην ανάπτυξη της επεξεργασίας ερωτημάτων, πρωτοκόλλων πρόσβασης και τεχνολογιών triple-store. Η φιλοσοφία των triple stores άρχισε να αναπτύσσεται ενεργά από τη Jena και το Sesame στις αρχές της δεκαετίας του 2000 στα Garlik JXT, YARS2, BigOWLIM, Jena TDB, Jena SDB, Virtuoso, AllegroGraph, BigData, Mulgara, Sesame, Kowari, 3Store και RDF Gateway. Στο μεταξύ τα Garlik και YARS2 δεν έχουν διανεμηθεί. Άλλα, όπως τα BigOWLIM και AllegroGraph είναι διαθέσιμα στο εμπόριο. Τα υπόλοιπα είναι όλα δωρεάν και open source.
Τα triple stores μπορούν να χωριστούν σε 3 μεγάλες κατηγορίες οι οποίες είναι:

- in-memory,
- native και
- non-native non-memory based.

- Η πρώτη κατηγορία είναι η αποθήκευση τριπλετών στη μνήμη (**in-memory**). Σ'αυτήν αποθηκεύονται οι RDF γράφοι στην κύρια μνήμη. Η αποθήκευση "των πάντων" δεν είναι η πιο αξιόπιστη μέθοδος στην περίπτωση των υπερβολικά μεγάλων σε αριθμό δεδομένων. Ωστόσο μπορεί να λειτουργήσει ως χρήσιμο σημείο αναφοράς και μπορεί να χρησιμοποιηθεί στην εκτέλεση ορισμένων πράξεων, όπως είναι η προσωρινή αποθήκευση δεδομένων από απομακρυσμένες ιστοσελίδες ή η εξαγωγή συμπερασμάτων. Οι περισσότεροι από τους in-memory-stores διαθέτουν αποτελεσματικούς reasoners και μπορούν να βοηθήσουν στην επίλυση του προβλήματος της εξαγωγής συμπερασμάτων σε μόνιμες RDF stores.

- Η δεύτερη κατηγορία είναι η τοπική αποθήκευση τριπλετών (**native triple store**),όπου παρέχεται μόνιμη αποθήκευση για την υλοποίηση βάσεων όπως το Virtuoso,Mulgara-AllegroGraph, Garlik JXT.
- Η τρίτη κατηγορία αποθήκευσης τριπλετών είναι η **non native non memory** αποθήκευση η οποία τρέχει σε ανεξάρτητες σχεσιακές βάσεις δεδομένων όπως είναι η JENA SDB και η οποία μπορεί να συνδυαστεί με βάσεις MySQL, PostsgreSQL, Oracle και άλλες.

Άλλος ένας τυπικός διαχωρισμός είναι ο παρακάτω:

- **Native RDF Stores :** Οι τοπικής αποθήκευσης τριπλέτες (Native stores) εφαρμόζουν ένα ολοκληρωμένο μηχανισμό διαχείρισης βάσεων δεδομένων που έχει βελτιστοποιηθεί για RDF επεξεργασία και λειτουργεί ανεξάρτητα από οποιοδήποτε άλλο σύστημα διαχείρισης βάσεων δεδομένων (DBMS). Τα δεδομένα αποθηκεύονται άμεσα στο σύστημα αρχείων, είτε σε ένα ενιαίο αρχείο είτε σε τμήματα.
- **DMBS:** Τριπλέτες αποθήκευσης τεχνολογίας DBMS κάνουν χρήση των λειτουργιών αποθήκευσης και ανάκτησης που υλοποιούνται σε ήδη υπάρχοντα συστήματα διαχείρισης βάσεων δεδομένων. Αυτό το είδος RDF αποθήκευσης, υποστηρίζεται από σχεσιακές DBMS και από διαφορετικά είδη μοντέλων αποθήκευσης για την αναπαράσταση του RDF μοντέλου στο σχεσιακό σχήμα.
- **Hybrid Stores:** Η RDF αποθήκευση που υποστηρίζει και τις δύο αρχιτεκτονικές δομές (native και DBMS) και εισέρχεται στην κατηγορία της υβριδικής αποθήκευσης.

3.8 Javascript

Η JavaScript (JS) είναι διερμηνευμένη γλώσσα προγραμματισμού για ηλεκτρονικούς υπολογιστές. Αρχικά αποτέλεσε μέρος της υλοποίησης των φυλλομετρητών Ιστού, ώστε τα σενάρια από την πλευρά του πελάτη (client-side scripts) να μπορούν να επικοινωνούν με τον χρήστη, να ανταλλάσσουν δεδομένα ασύγχρονα και να αλλάζουν δυναμικά το περιεχόμενο του εγγράφου που εμφανίζεται. Η JavaScript είναι μια γλώσσα σεναρίων που βασίζεται στα πρωτότυπα (prototype-based), είναι δυναμική, με ασθενείς τύπους και έχει συναρτήσεις ως αντικείμενα πρώτης τάξης. Η σύνταξή της είναι επηρεασμένη από τη C. Η JavaScript αντιγράφει πολλά ονόματα και συμβάσεις ονοματοδοσίας από τη Java, αλλά γενικά οι δύο αυτές γλώσσες δε σχετίζονται και έχουν πολύ διαφορετική σημασιολογία. Το πρότυπο της γλώσσας κατά τον οργανισμό τυποποίησης ECMA ονομάζεται **ECMAscript**

3.8.1 Json

Πηγή εικόνας: http://code.google.com/p/google-gson/%5C

Το JSON (JavaScript Object Notation) είναι ένα ελαφρύ πρότυπο ανταλλαγής δεδομένων. Είναι εύκολο για τους ανθρώπους να το διαβάσουν και γράψουν. Είναι εύκολο

για τις μηχανές να το αναλύσουν (parse) και να το παράγουν (generate). Είναι βασισμένο πάνω σε ένα υποσύνολο της γλώσσας προγραμματισμού JavaScript, Standard ECMA-262 Έκδοση 3η - Δεκέμβριος 1999. Το JSON είναι ένα πρότυπο κειμένου το οποίο είναι τελείως ανεξάρτητο από γλώσσες προγραμματισμού αλλά χρησιμοποιεί πρακτικές (conventions) οι οποίες είναι γνωστές στους προγραμματιστές της οικογένειας προγραμματισμού C, συμπεριλαμβανομένων των C, C++, C#, Java, JavaScript, Perl, Python, και πολλών άλλων. Αυτές οι ιδιότητες κάνουν το JSON μια ιδανική γλώσσα προγραμματισμού ανταλλαγής δεδομένων.

- Το JSON είναι χτισμένο σε δύο δομές:

  - Μια συλλογή από ζευγάρια ονομάτων/τιμών. Σε διάφορες γλώσσες προγραμματισμού, αυτό αντιλαμβάνεται ως ένα object, καταχώριση, δομή, λεξικό, πίνακα hash (hash table), λίστα κλειδιών, ή associative πίνακα.

  - Μία ταξινομημένη λίστα τιμών. Στις περισσότερες γλώσσες προγραμματισμού, αυτό αντιλαμβάνεται ως ένας πίνακας (array), διάνυσμα, λίστα, ή ακολουθία.

Αυτά είναι τα universal data structures. Ουσιαστικά όλες οι μοντέρνες γλώσσες προγραμματισμού τα υποστηρίζουν με τον έναν ή τον άλλον τρόπο. Λογικό είναι πως ένα πρότυπο δεδομένων το οποίο είναι εύκολα μεταβαλλόμενο με γλώσσες προγραμματισμού οι οποίες επίσης είναι βασισμένες σε αυτές τις δομές.

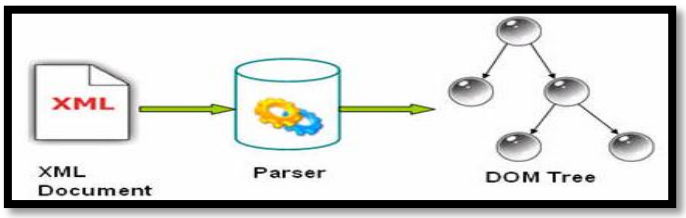
Στα πλαίσια αυτής της πτυχιακής χρησιμοποιήσαμε το Gson της έκδοσης 2.2.4,το οποίο είναι μια java βιβλιοθήκη η οποία χρησιμοποιείται για να μετατρέπει Java αντικείμενα σε JSON αναπαράσταση. Χρησιμοποιείται επίσης για να μετατρέπει χαρακτήρες (strings) σε ισοδύναμα java αντικείμενα.

3.8.2 jQuery

Πηγή εικόνας: https://www.dreamfactory.com/sites/default/files/large_jquery_logo.png

Η jQuery είναι μια βιβλιοθήκη (framework) JavaScript που σχεδιάστηκε για να απλοποιήσει το client-side scripting της HTML.Η jQuery πρωτοεμφανίστηκε τον Ιανουάριο του 2006 στο BarCamp από τον John Resig. Πρόκειται για μια βιβλιοθήκη Javascript ανοιχτού κώδικα, υπό τις άδειες MIT License και την GNU General Public License. Η jQuery χρησιμοποιείται από προγραμματιστές για τη ταχεία ανάπτυξη ιστοσελίδων και διαδικτυακών εφαρμογών που χρειάζονται μεγάλη ευχρηστία και διαδραστικότητα (interactivity).Έχει σχεδιαστεί για να καταστήσει ευκολότερη την πλοήγηση σε ένα έγγραφο, την επιλογή DOM ,το χειρισμό και την ανάπτυξη Ajax εφαρμογών και άλλα.

3.9 DOM



Πηγή εικόνας:
http://wiki.sdn.sap.com/wiki/download/attachments/78053807/dom.JPG?version=2&modificationDate=1234263303804

Το Document Object Model (DOM) είναι μια cross-platform σύμβαση για την αναπαράσταση και την αλληλεπίδραση με αντικείμενα σε HTML, XHTML αλλά και XML έγγραφα. Το DOM επιτρέπει το parsing της XML. Συγκεκριμένα το XML DOM περιλαμβάνει μεθόδους (συναρτήσεις) για τη διάσχιση του XML δέντρου δίνοντας συνάμα την δυνατότητα της εισαγωγής αλλά και της διαγραφής κόμβων (nodes). Ωστόσο ένα έγγραφο XML για να μπορέσει να δεχτεί τις παραπάνω αλλαγές θα πρέπει πρώτα να φορτωθεί σε ένα αντικείμενο XML DOM και στη συνέχεια θα μπορεί να προσεγγιστεί μέσω javascript. Πλέον τα περισσότερα προγράμματα περιήγησης έχουν ενσωματωμένη μονάδα για την ανάλυση του XML.

3.10 Sparql

Πηγή εικόνας: http://semanticweb.com/files/2012/11/alibobo_w3cSPARQL-logo.png

Η SPARQL θεωρείται η επίσημη γλώσσα ερωτημάτων για την RDF, μετά τη σχετική σύσταση της W3C. Η SPARQL είναι για την RDF ότι ακριβώς είναι για τις σχεσιακές βάσεις δεδομένων η SQL. Τα SPARQL ερωτήματα βασίζονται σε μοτίβα τριάδων/τριπλετών (triple patterns). Η μηχανή της SPARQL που εκτελεί ένα ερώτημα αναζητά από όλους τους πόρους εκείνους που επαληθεύουν τα μοτίβα τριάδων του ερωτήματος, σύμφωνα με τις RDF προτάσεις που υπάρχουν στη βάση στην οποία η SPARQL είναι συνδεδεμένη.

Υπάρχουν τέσσερις μορφές ερωτημάτων οι οποίες είναι: SELECT,CONTSTRUCT,ASK και DESCRIBE.

- ***SELECT*** Οι ερωτήσεις αυτής της μορφής, επιστρέφουν για όλες ή μέρος των μεταβλητών, τις τιμές που έχουν ανατεθεί σε αυτές. Η σύνταξη SELECT * επιστρέφει τις τιμές για όλες τις μεταβλητές, σε αντίθετη περίπτωση μετά το SELECT αναφέρονται οι μεταβλητές για τις οποίες θα επιστραφούν αποτελέσματα.
- ***CONSTRUCT*** Οι ερωτήσεις αυτής της μορφής, επιστρέφουν έναν RDF γράφο του οποίου η μορφή έχει προσδιοριστεί από έναν γράφο πρότυπο (graph pattern). Ο γράφος δημιουργείται περνώντας κάθε λύση από την

ακολουθία λύσεων και αντικαθιστώντας τις τιμές των μεταβλητών, στις αντίστοιχες μεταβλητές των σχηματομορφών τριπλέτων του RDF γράφου. Ο τελικός γράφος παράγεται από την ένωση των σχηματομορφών τριπλέτων που έχουν δημιουργηθεί για όλες τις λύσεις της ακολουθίας των λύσεων. Στην περίπτωση που κάποια μεταβλητή μιας σχηματομορφής τριπλέτας δεν έχει ανατεθεί τιμή (δηλαδή η μεταβλητή είναι Unbound), τότε η τριπλέτα αυτή δεν περιλαμβάνεται στον τελικό γράφο

- ***ASK*** Οι ερωτήσεις αυτής της μορφής, μπορούν να χρησιμοποιηθούν από της εφαρμογές για να εκλεχθεί αν μια ερώτηση έχει λύση. Δεν γίνεται επιστροφή πληροφορίας για πιθανές λύσεις της ερώτησης, παρά μόνο αν η ερώτηση έχει λύση ή όχι, επιστρέφοντας yes ή no .

- Μια μεταβλητή συμβολίζεται με το ? ακολουθούμενο από το όνομα της.
- Όλες οι τριπλέτες σε ένα SPARQL query χωρίζονται μεταξύ τους με “.”
- Όταν 2 ή περισσότερες τριπλέτες μοιράζονται το ίδιο subject τότε μπορούμε να το παραλείψουμε χρησιμοποιώντας “;”
- Αν υπάρχει URI αυτό πρέπει να γράφεται μέσα σε γωνιακές παρενθέσεις (π.χ. <http://www.w3c.org/#something>).
- Είναι δυνατό να ορίσουμε προθέματα (prefixes) για κάποια namespaces και να τα χρησιμοποιήσουμε στο ερώτημα
- Το keyword “a” της SPARQL χρησιμοποιείται για να δούμε τον τύπο ενός αντικειμένου (την κλάση του). Ουσιαστικά είναι μια συντομογραφία για το rdf:type.

3.10.1 Παραδείγματα με τη χρήση της γλώσσας SPARQL

```
//Query για την εμφάνιση των υποκλάσεων (subclasses) μιας κλάσης
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX
ns:http://www.semanticweb.org/ontologies/2009/5/Ontology124403
3197062.owl#
SELECT ?subClass WHERE { ?subClass rdfs:subClassOf ns:Room };
```

```
//Το αποτέλεσμα που παίρνουμε με την εκτέλεση του παραπάνω ερωτήματος
```

```
-------------------------------------------------------------------------------------
| subClass                                                                          |
=====================================================================================
| <http://www.semanticweb.org/ontologies/2009/5/Ontology1244033197062.owl#Bedroom>    |
| <http://www.semanticweb.org/ontologies/2009/5/Ontology1244033197062.owl#LivingRoom> |
-------------------------------------------------------------------------------------
```

Όπως παρατηρούμε το ερώτημα που εκτελείται επιστρέφει τις υποκλάσεις της κλάσης Room οι οποίες στην οντολογία που χρησιμοποιούμε είναι το Bedroom και το LivingRoom. Με την ίδια λογική μπορούμε να εμφανίσουμε τις υποκλάσεις οποιαδήποτε οντολογίας αφού πρώτα δηλώσουμε τα prefixes της.

```
//Query για την εμφάνιση των individuals μιας κλάσης
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX
ns:<http://www.semanticweb.org/ontologies/2009/5/Ontology12440
33197062.owl#>"
SELECT ?individuals
WHERE { ?individuals rdf:type ns:Door};
```

```
//Το αποτέλεσμα που παίρνουμε με την εκτέλεση του παραπάνω ερωτήματος

-----------------------------------------------------------------------------------------
| individuals                                                                           |
=========================================================================================
| <http://www.semanticweb.org/ontologies/2009/5/Ontology1244033197062.owl#Door_22> |
| <http://www.semanticweb.org/ontologies/2009/5/Ontology1244033197062.owl#Door_23> |
-----------------------------------------------------------------------------------------
```

Το ερώτημα που εκτελείται επιστρέφει όλους τους individuals μιας συγκεκριμένης κλάσης. Στην παραπάνω περίπτωση η κλάση είναι η Door και η απάντηση του query είναι το Door_22 και Door_23.

```
//Query σε ObjectProperty για την εμφάνιση του range
PREFIX ns:
<http://www.semanticweb.org/ontologies/2009/5/Ontology12440331
97062.owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?ObjectRange WHERE {ns:has_a_Style rdfs:range
?ObjectRange.};
```

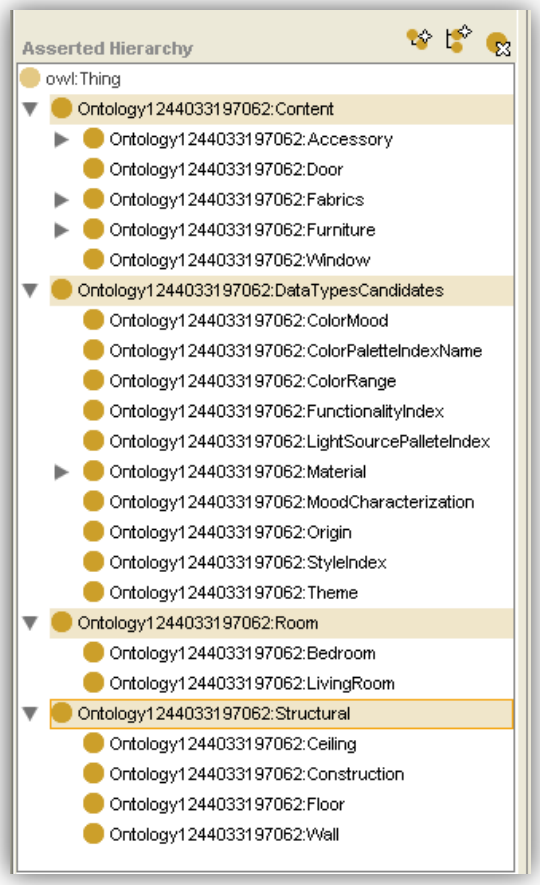
```
//Το αποτέλεσμα που παίρνουμε με την εκτέλεση του παραπάνω ερωτήματος

-----------------------------------------------------------------------------------------
| ObjectRange                                                                           |
=========================================================================================
| <http://www.semanticweb.org/ontologies/2009/5/Ontology1244033197062.owl#StyleIndex> |
-----------------------------------------------------------------------------------------
```

Το ερώτημα που εκτελείται επιστρέφει το range που αντιστοιχεί σε συγκεκριμένο object property. Στην οντολογία έχουμε ένα object property με όνομα has_a_Style και το range του είναι η κλάση StyleIndex,όπως φαίνεται και από το αποτέλεσμα που έχει προκύψει μετά την εκτέλεση του ερωτήματος.

4 ΠΕΡΙΓΡΑΦΗ OWL ΟΝΤΟΛΟΓΙΑΣ

Στα πλαίσια αυτής της πτυχιακής χρησιμοποιήθηκε μια σημασιολογική οντολογία σχεδίασης και διακόσμησης εσωτερικών χώρων. Η οντολογία έχει ως σημείο αναφοράς ένα. Για την εκτενέστερη περιγραφή ενός εσωτερικού χώρου, αναλύσαμε όλα τα φυσικά αντικείμενα που μπορεί να περιέχει (έτσι ώστε κάποιος να μπορεί να καταλάβει τι ακριβώς περιέχει ένα δωμάτιο), τα χαρακτηριστικά κάθε φυσικού αντικειμένου (έτσι ώστε να υπάρχει πληροφορία για το στυλ, τις διαστάσεις, το σχήμα, κτλ), και τελικώς τις σχέσεις μεταξύ αυτών των αντικειμένων (έτσι ώστε κάποιος να μπορεί να καταλάβει την δομή του εσωτερικού χώρου).

Η υπάρχουσα οντολογία είναι μια τροποποίηση παλαιότερης οντολογίας που είχε δημοσιευτεί το 2009 με τίτλο "Xml Annotation Of Conceptual Characteristics In Interior Decoration" [4]



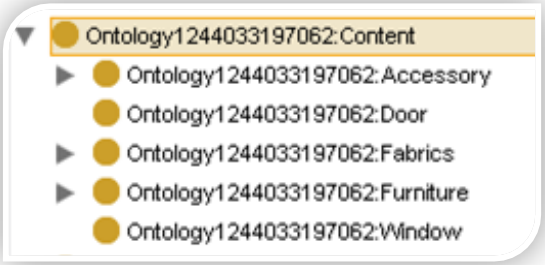
Εικόνα 6-Δομή οντολογίας

[4] Athanasios G.Malamos, Paraskevi V. Sympa, Georgios S.Mamakis, "Xml Annotation Of Conceptual Characteristics In Interior Decoration", 6th International Conference, New Horizons in Industry, Business and Education (NHIBE 2009), 27 - 28 August 2009, Santorini

4.1 Κλάσεις οντολογίας

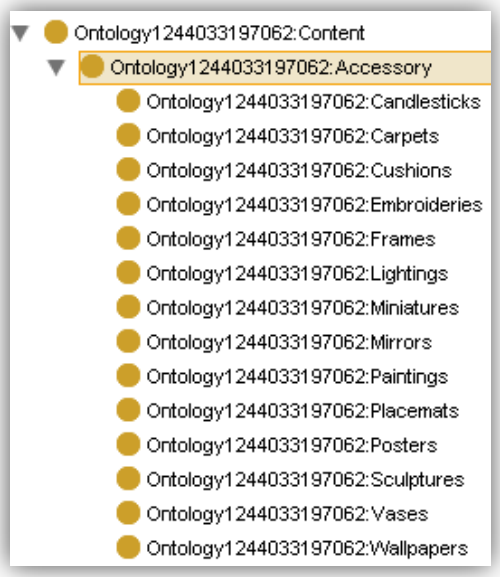
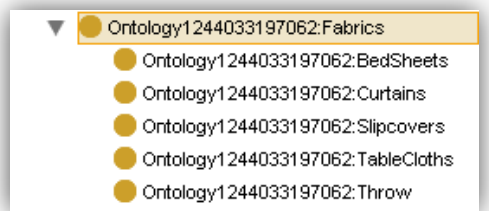
Η κορυφή της οντολογίας είναι η owl:Thing η οποία είναι ρίζα κάθε OWL οντολογίας, και σ' αυτή στηρίζονται όλες οι επόμενες κλάσεις. Ως υποκλάσεις της owl:Thing, ορίσαμε τις: ***Content***, ***DataTypesCandidates*** ,***Room*** και ***Structural***. Αυτές τις υποκλάσεις όπως θα δούμε απαρτίζουν άλλες υποκλάσεις.

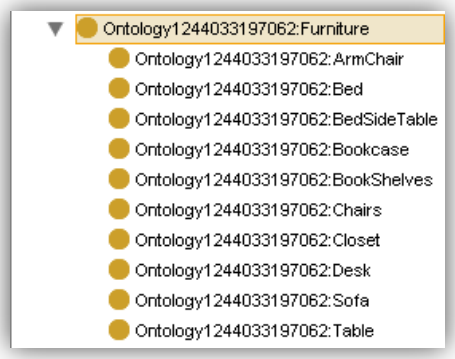
- **Content**, *πρόκειται για την κλάση που περιέχει όλα τα φυσικά αντικείμενα που μπορούν να βρεθούν σε ένα εσωτερικό χώρο*



Πίνακας 2-Κλάση Content

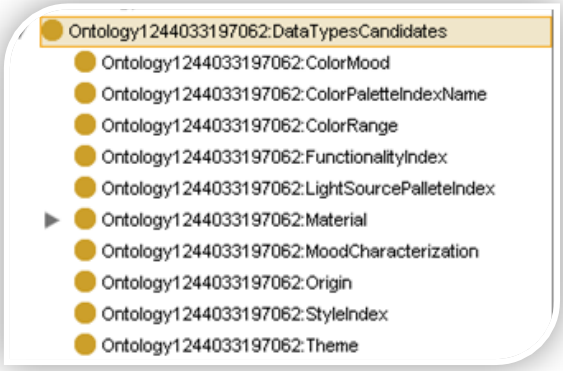
Αποτελείται από τις υποκλάσεις:

Accessory 	Υποκλάση του Content που περιέχει δικές της υποκλάσεις, τα άτομα των οποίων καθορίζουν ένα πλήθος πιθανών αξεσουάρ ενός δωματίου, όπως χαλιά, καθρέπτες, πίνακες κ.α.
Door	Υποκλάση του Content που περιέχει άτομα (individuals) που καθορίζουν τις πόρτες ενός δωματίου
Fabrics 	Υποκλάση του Content που περιέχει δικές της υποκλάσεις άτομα των οποίων καθορίζουν κατασκευές με υφάσματα

Furniture  	Υποκλάση του Content που περιέχει δικές της υποκλάσεις, τα άτομα των οποίων καθορίζουν ένα πλήθος πιθανών επίπλων ενός δωματίου
Window	Υποκλάση του Content που περιέχει άτομα που καθορίζουν τα παράθυρα ενός δωματίου

Πίνακας 3-Υποκλάσεις Content

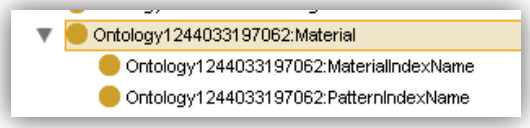
- **DataTypesCandidates**, *πρόκειται για την κλάση που απαριθμεί όλα τα ποιοτικά χαρακτηριστικά ενός δωματίου και των αντικειμένων που περιέχονται σε αυτό*



Εικόνα 7-Κλάση DatatypeCandidates

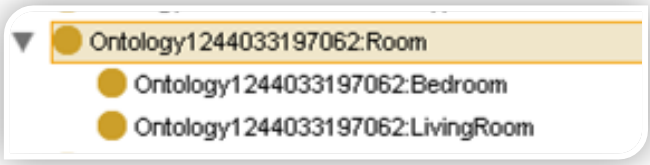
Αποτελείται από τις υποκλάσεις:

ColorMood	Υποκλάση του DataTypesCandidates που περιέχει άτομα που κατατάσσουν ένα σύνολο από ColorRange άτομα, σε μια αυθαίρετη επικρατούσα ατμόσφαιρα
ColorPaletteIndexName	Υποκλάση του DataTypesCandidates που περιέχει άτομα που καθορίζουν ένα χρώμα
ColorRange	Υποκλάση του DataTypesCandidates που περιέχει άτομα που κατατάσσουν ένα σύνολο από σε μια γενικευμένη κατηγορία (παλέτα) χρώματος

FunctionalityIndex	Υποκλάση του DataTypesCandidates που περιέχει άτομα που καθορίζουν την λειτουργικότητα του χώρου
LightSourcePalleteIndex	Υποκλάση του DataTypesCandidates που περιέχει άτομα που καθορίζονται ως πηγές φωτός
Material 	Υποκλάση του DataTypesCandidates που περιέχει τις δύο υποκλάσεις, τα άτομα των οποίων καθορίζουν αντίστοιχα τα υλικά και το σχέδιο κατασκευής ενός αντικειμένου
MoodCharacterization	Υποκλάση του DataTypesCandidates που περιέχει άτομα που καθορίζουν την προσωπικότητα (mood) του δωματίου
Origin	Υποκλάση του DataTypesCandidates που περιέχει άτομα που καθορίζουν τον τόπο καταγωγής ενός αντικειμένου
StyleIndex	Υποκλάση του DataTypesCandidates που περιέχει άτομα που καθορίζουν το στυλ των ατόμων
Theme	Υποκλάση του DataTypesCandidates που περιέχει άτομα που καθορίζουν το θέμα του αντικειμένου

Πίνακας 4-Υποκλάσεις DataTypesCandidates

- **Room**, *πρόκειται για την κλάση που περιέχει τους πιθανούς τύπους ενός εσωτερικού χώρου*



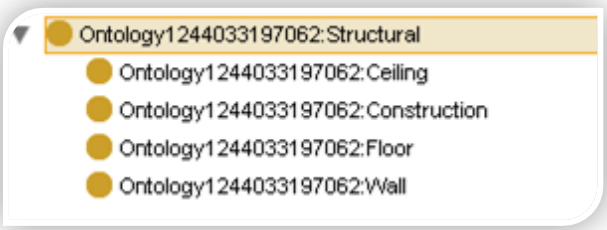
Εικόνα 8-Κλάση Room

Αποτελείται από τις υποκλάσεις:

Bedroom	Υποκλάση του Room που περιέχει άτομα που καθορίζουν ένα υπνοδωμάτιο
LivingRoom	Υποκλάση του Room που περιέχει άτομα που καθορίζουν ένα καθιστικό

Πίνακας 5-Υποκλάσεις Room

- **Structural,** *πρόκειται για την κλάση που περιέχει υποκλάσεις που υποδεικνύουν απαραίτητα άτομα για την δομή και τον σχεδιασμό ενός δωματίου, τα οποία δύσκολα επιδέχονται αλλαγές*



Αποτελείται από τις υποκλάσεις:

Ceiling	Υποκλάση του Structural που περιέχει άτομα που αντιστοιχούν στην οροφή ενός δωματίου
Construction	Υποκλάση του Structural που περιέχει άτομα που καθορίζουν κατασκευές και όχι δομικά αντικείμενα του χώρου
Floor	Υποκλάση του Structural που περιέχει άτομα που αντιστοιχούν στο πάτωμα ενός δωματίου
Wall	Υποκλάση του Structural που περιέχει άτομα που αντιστοιχούν στους τοίχους ενός δωματίου

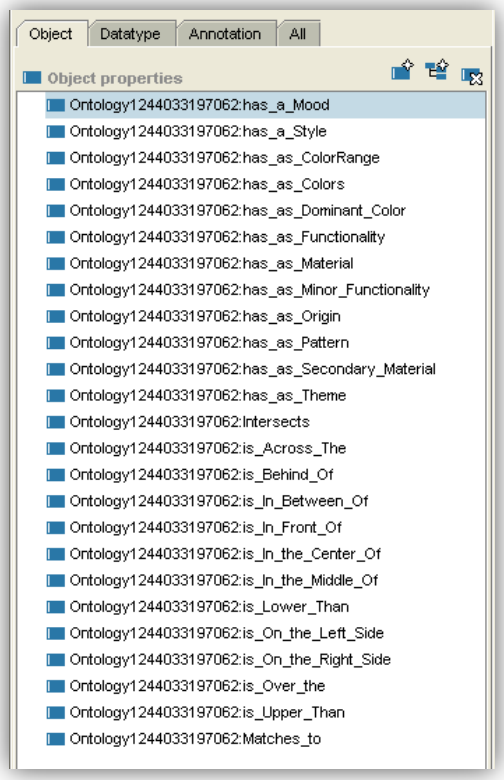
Πίνακας 6-Υποκλάσεις Structural

4.2 Ιδιότητες (Properties) Οντολογίας

Οι ιδιότητες αποτελούν ένα από τα βασικά στοιχεία της OWL και χωρίζονται σε δύο μεγάλες κατηγορίες οι οποίες είναι οι Object properties και οι Datatype properties. Οι Object properties μιας OWL οντολογίας συσχετίζουν τα άτομα των κλάσεων που δηλώνονται στο Domain πεδίο τους, με τα άτομα των κλάσεων που δηλώνονται στο Range πεδίο τους. Αντίθετα, οι datatype properties συσχετίζουν τα άτομα των κλάσεων που δηλώνονται στο Domain πεδίο τους, με ένα datatype που δηλώνεται στο Range πεδίο τους. Τα διαθέσιμα datatypes που επιτρέπεται να δηλωθούν είναι οποιουδήποτε τύπου από τα boolean, float, int, string, data, dataTime και time.

Παρακάτω θα αναλυθούν όλες οι ιδιότητες της οντολογίας ανάλογα με την κατηγορία στην οποία ανήκουν.

4.2.1 Object Properties

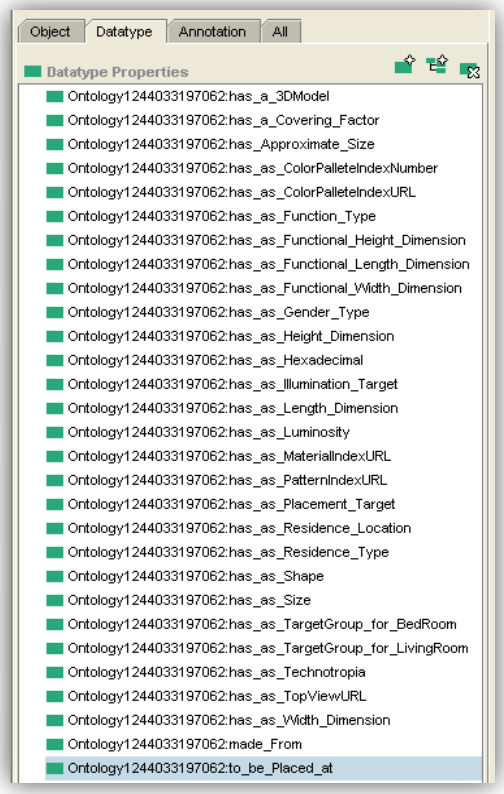


Εικόνα 9-Object properties

- ***has_a_Mood***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Room, έχει κάποια προσωπικότητα, το οποίο είναι άτομο της κλάσης MoodCharacterization.
- ***has_a_Style***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, έχει κάποιο στυλ, το οποίο είναι άτομο της κλάσης StyleIndex.
- ***has_as_ColorRange***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης ColorMood, περιέχει κάποια γενικευμένη κατηγορία, η οποία είναι άτομο της κλάσης ColorRange.
- ***has_as_Colors***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης ColorRange ή Structural ή Content, έχει κάποιο χρώμα, το οποίο είναι άτομο της κλάσης ColorPaletteIndexName.
- ***has_as_Dominant_Color***,πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομα της κλάσης Accessory,έχει κάποιο κυρίαρχο χρώμα,το οποίο είναι άτομο της κλάσης ColorPaletteIndexName.
- ***has_as_Functionality***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Room, έχει κάποια βασική λειτουργικότητα, η οποία είναι άτομο της κλάσης FunctionalityIndex.
- ***has_as_Material***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, προέρχεται από κάποιο κύριο υλικό, το οποίο είναι άτομο της κλάσης MaterialIndexName.
- ***has_as_Minor_Functionality,*** πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Room, έχει κάποια δευτερεύουσα λειτουργικότητα, η οποία είναι άτομο της κλάσης FunctionalityIndex.
- ***has_as_Origin,*** πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Furniture, έχει κάποια καταγωγή, η οποία είναι άτομο της κλάσης Origin.

- ***has_as_Pattern,*** πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, έχει κάποιο σχέδιο, το οποίο είναι άτομο της κλάσης PatternIndexName.
- ***has_as_Secondary_Material,*** πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, προέρχεται από κάποιο δευτερεύων υλικό, το οποίο είναι άτομο της κλάσης MaterialIndexName.
- ***has_as_Theme,*** πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Posters, Paintings ή wallpapers έχει θέμα, το οποίο είναι άτομο της κλάσης Theme
- ***Intersects***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, διασταυρώνεται με κάποιο άλλο άτομο της κλάσης Structural ή Content.
- ***is_Across_The***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content , βρίσκεται απέναντι από κάποιο άλλο άτομο της κλάσης Structural ή Content.
- ***is_Behind_Of***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, βρίσκεται πίσω από κάποιο άλλο άτομο της κλάσης Structural ή Content.
- ***is_In_Between_Of***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, βρίσκεται ανάμεσα σε κάποιο άλλο άτομο της κλάσης Structural ή Content.
- ***is_In_Front_Of***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, βρίσκεται μπροστά από κάποιο άλλο άτομο της κλάσης Structural ή Content.
- ***is_In_the_Center_Of***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, βρίσκεται στο κέντρο κάποιου άλλου ατόμου της κλάσης Structural ή Content.
- ***is_In_the_Middle_Of***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, βρίσκεται στο μέσο σημείο κάποιου άλλου ατόμου της κλάσης Structural ή Content.
- ***is_Lower_Than***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, βρίσκεται χαμηλότερα σε σχέση με κάποιο άλλο άτομο της κλάσης Structural ή Content.
- ***is_On_the_Left_Side***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, βρίσκεται στο αριστερό μέρος κάποιου άλλου ατόμου της κλάσης Structural ή Content.
- ***is_On_the_Right_Side***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, βρίσκεται στο δεξί μέρος κάποιου άλλου ατόμου της κλάσης Structural ή Content.
- ***is_Over_the***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, βρίσκεται πάνω από κάποιο άλλο άτομο της κλάσης Structural ή Content.
- ***is_Upper_Than***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, βρίσκεται υψηλότερα σε σχέση με κάποιο άλλο άτομο της κλάσης Structural ή Content.
- ***Matches_to***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, ταιριάζει στο χώρο απόλυτα με κάποιο άλλο άτομο της κλάσης Structural ή Content.

4.2.2 Datatype Properties



Εικόνα 10-Datatype properties

- ***has_a_3D_Model***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, έχει ένα αλφαριθμητικό (string), το οποίο είναι ένα URI που δείχνει ένα τρισδιάστατο μοντέλο αυτού του ατόμου.
- ***has_a_Covering_Factor***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Room, έχει ένα πραγματικό αριθμό (float), ο οποίος αντιπροσωπεύει την εκατοστιαία αναλογία του δωματίου που καλύπτεται από αντικείμενα.
- ***has_a_Approximate_Size***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Room, έχει ένα αλφαριθμητικό (string), το οποίο αντιπροσωπεύει το μέγεθος του δωματίου κατά προσέγγιση. Αυτό το string έχει συγκεκριμένες τιμές (functional τιμές) που είναι small,medium και large.
- ***has_as_ColorPalleteIndexNumber***, πρόκειται για την ιδιότητα που καθορίζει ότι έναάτομο της κλάσης ColorPaletteIndexName, έχει ένα αλφαριθμητικό (string), το οποίοκαθορίζει έναν μοναδικό αριθμό που αντιπροσωπεύει το χρώμα.
- ***has_as_ColorPalleteIndexURL***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης ColorPaletteIndexName, έχει ένα αλφαριθμητικό (string), το οποίο αναφέρει το URL ή απλώς περιέχει πληροφορία για την διαθεσιμότητα του χρώματος.
- ***has_as_Function_Type***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Curtains, έχει ένα αλφαριθμητικό (string), το οποίο αντιπροσωπεύει τη λειτουργικότητα των κουρτινών. Αυτό το string έχει συγκεκριμένες τιμές (functional τιμές) που είναι Simple,Blinds και Roman.
- ***has_as_Functional_Height_Dimension***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Door,Furniture,Window, έχει ένα πραγματικό αριθμό (float), ο οποίος αντιπροσωπεύει το ύψος στο οποίο θα είναι λειτουργικό το αντικείμενο.

- ***has_as_Functional_Length_Dimension***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Door,Furniture,Window, έχει ένα πραγματικό αριθμό (float), ο οποίος αντιπροσωπεύει το μήκος κατά το οποίο θα είναι λειτουργικό το αντικείμενο.
- ***has_as_Functional_Width_Dimension***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Door,Furniture,Window, έχει ένα πραγματικό αριθμό (float), ο οποίος αντιπροσωπεύει το πλάτος στο οποίο θα είναι λειτουργικό το αντικείμενο.
- ***has_as_Gender_Type***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Furniture, έχει ένα αλφαριθμητικό (string), το οποίο αντιπροσωπεύει την ομάδα των ατόμων που απευθύνεται το συγκεκριμένο έπιπλο. Αυτό το string έχει συγκεκριμένες τιμές (functional τιμές) που είναι Male και Female.
- ***has_as_Height_Dimension***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, έχει ένα πραγματικό αριθμό (float), ο οποίος αντιστοιχεί στην διάσταση του ύψους του αντικειμένου.
- ***has_as_Hexadecimal***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης ColorPaletteIndexName, έχει ένα αλφαριθμητικό (string), το οποίο καθορίζει τον δεκαεξαδικό αριθμό που αντιστοιχεί στο συγκεκριμένο χρώμα.
- ***has_as_Illumination_Target:*** πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Lightings, έχει ένα αλφαριθμητικό (string), το οποίο αντιστοιχεί στον τύπο του φωτός. Αυτό το string έχει συγκεκριμένες τιμές (functional τιμές) που είναι Spotlight,Pointlight και Directional.
- ***has_as_ Length_Dimension***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, έχει ένα πραγματικό αριθμό (float), ο οποίος αντιστοιχεί στην διάσταση του μήκους του αντικειμένου
- ***has_as_Luminosity***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Room, έχει ένα πραγματικό αριθμό (float), ο οποίος αντιπροσωπεύει την φωτεινότητα του εσωτερικού χώρου.
- ***has_as_MaterialIndexURL***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης MaterialIndexName, έχει ένα αλφαριθμητικό (string), το οποίο αναφέρει το URL ή απλώς περιέχει πληροφορία για την διαθεσιμότητα του υλικού.
- ***has_as_PatternIndexURL***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης PatternIndexName, έχει ένα αλφαριθμητικό (string), το οποίο αναφέρει το URL ή απλώς περιέχει πληροφορία για την διαθεσιμότητα του σχεδίου.
- ***has_as_Placement_Target:*** πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Lightings, έχει ένα αλφαριθμητικό (string), το οποίο αντιστοιχεί στην τοποθεσία του αντικειμένου στο χώρο. Αυτό το string έχει συγκεκριμένες τιμές (functional τιμές) που είναι Ceiling,Freestanding ,Desktop και Wall.
- ***has_as_Residence_Location:*** πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Room, έχει ένα αλφαριθμητικό (string), το οποίο αντιστοιχεί στην τοποθεσία που βρίσκεται το δωμάτιο. Αυτό το string έχει συγκεκριμένες τιμές (functional τιμές) που είναι Mountain,Seaside και Urban.
- ***has_as_Residence_Type:*** πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Room, έχει ένα αλφαριθμητικό (string), το οποίο αντιστοιχεί στον τύπο του δωματίου. Αυτό το string έχει συγκεκριμένες τιμές (functional τιμές) που είναι Main και Secondary.
- ***has_as_Shape***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, έχει ένα αλφαριθμητικό (string), το οποίο περιγράφει τα σχηματικά χαρακτηριστικά ενός φυσικού αντικειμένου.
- ***has_as_Size***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Accessory, έχει ένα αλφαριθμητικό (string), το οποίο αναφέρει κατά προσέγγιση το μέγεθος του αντικειμένου.

- ***has_as_TargetGroup_for_BedRoom:*** πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Bedroom, έχει ένα αλφαριθμητικό (string), το οποίο αντιστοιχεί στην ηλικιακή ομάδα που απευθύνεται το υπνοδωμάτιο. Αυτό το string έχει συγκεκριμένες τιμές (functional τιμές) που είναι Kids,Teenager,Adult.
- ***has_as_TargetGroup_for_LivingRoom:*** πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης LivingRoom, έχει ένα αλφαριθμητικό (string), το οποίο αντιστοιχεί στην ομάδα που απευθύνεται το καθιστικό/σαλόνι. Αυτό το string έχει συγκεκριμένες τιμές (functional τιμές) που είναι Family και Student.
- ***has_as_Technotropia***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Fabrics, έχει ένα αλφαριθμητικό (string), το οποίο αναφέρει την τεχνοτροπία που έχει κατασκευαστεί το κάθε ύφασμα.
- ***has_as_TopViewURL***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Room, έχει ένα αλφαριθμητικό (string), το οποίο αναφέρει το URL ή απλώς περιέχει πληροφορία για την κάτοψη του δωματίου.
- ***has_as_Width_Dimension***, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, έχει ένα πραγματικό αριθμό (float), ο οποίος αντιστοιχεί στην διάσταση του πλάτους του αντικειμένου.
- ***has_DoubleBed:*** πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Bed, έχει ένα αλφαριθμητικό (string), το οποίο αντιστοιχεί στην ιδιότητα του κρεβατιού να είναι διπλό ή όχι,. Αυτό το string έχει συγκεκριμένες τιμές (functional τιμές) που είναι Yes και No.
- ***made_From:*** πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Accessory ή Furniture, έχει ένα αλφαριθμητικό (string), το οποίο αντιστοιχεί στον τρόπο κατασκευής του αντικειμένου. Αυτό το string έχει συγκεκριμένες τιμές (functional τιμές) που είναι Industrial και Handmade.
- ***number_of_Females:*** πρόκειται για την ιδιότητα που δείχνει ότι ένα άτομο της κλάσης Bedroom, έχει έναν ακέραιο αριθμό (int),ο οποίος αντιστοιχεί στην πλήθος των κοριτσιών που θα χρησιμοποιούν ένα υπνοδωμάτιο.
- ***number_of_Males:*** πρόκειται για την ιδιότητα που δείχνει ότι ένα άτομο της κλάσης Bedroom, έχει έναν ακέραιο αριθμό (int),ο οποίος αντιστοιχεί στην πλήθος των αγοριών που θα χρησιμοποιούν ένα υπνοδωμάτιο.
- ***to_be_Placed_at:*** πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Curtains, έχει ένα αλφαριθμητικό (string), το οποίο αντιστοιχεί στην θέση τοποθέτησης μιας κουρτίνας. Αυτό το string έχει συγκεκριμένες τιμές (functional τιμές) που είναι Windows και Door.

5 ΕΦΑΡΜΟΓΗ "WIZARD FOR ADMIN"

Η εφαρμογή που αναπτύχθηκε σε αυτό το μέρος της πτυχιακής είναι υπεύθυνη για:

- **την δημιουργία ενός interior space**
- **ή την τροποποίηση του υπάρχοντα interior space**

Συγκεκριμένα οι δυνατότητες που παρέχονται στον διαχειριστή είναι :

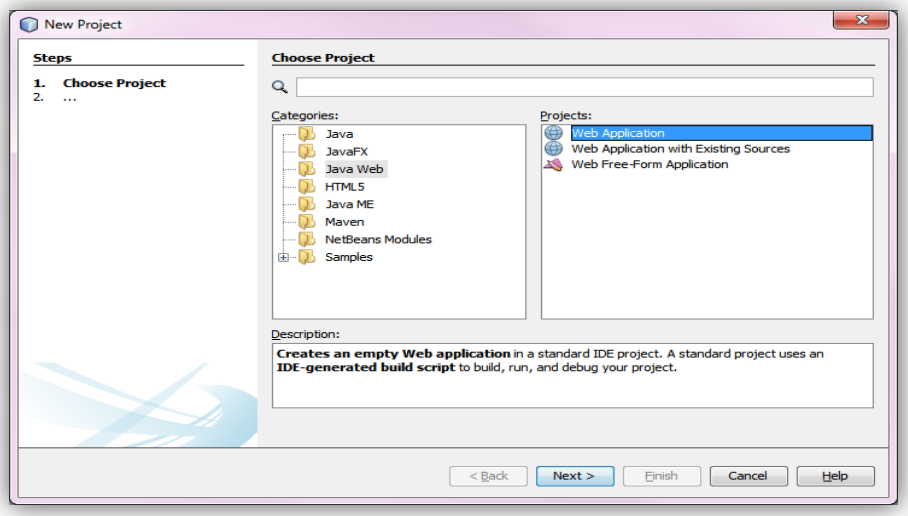
- Η δημιουργία βάσης δεδομένων σε SDB με περιεχόμενο την οντολογία που βρίσκεται σε μορφή owl και το αρχείο της είναι τοποθετημένο στον φάκελο του project ή αντίστοιχα δυνατότητα τροποποίησης μιας υπάρχουσας βάσης σε περίπτωση που το επιθυμεί ο διαχειριστής.
- Η προσθήκη individual σε μια κλάση της οντολογίας
- Η αφαίρεση individual από μία κλάση της οντολογίας
- Η αποθήκευση statements (τριπλετών) στη βάση μέσω των τιμών: όνομα individual, όνομα property και range από property.

Οι τρεις τελευταίες ιδιότητες συνεπάγονται με ενημέρωση της βάσης καθώς οποιαδήποτε αλλαγή γίνεται στην οντολογία , αυτόματα θα πρέπει να ενημερώνεται και η βάση δεδομένων.

5.1 Υλοποίηση "Wizard for Admin"

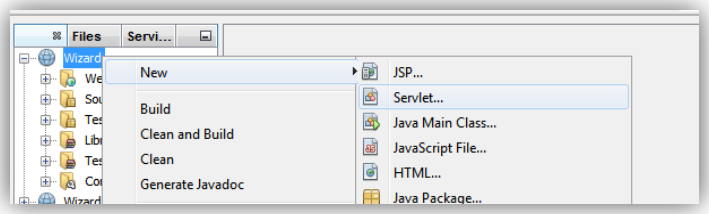
Το περιβάλλον εργασίας που χρησιμοποιήθηκε είναι το NETBEANS IDE,με εγκατεστημένο ως server τον tomcat έτσι ώστε να μας δίνεται η δυνατότητα δημιουργίας web application

- ***File->New Project->Java Web->Web Application***



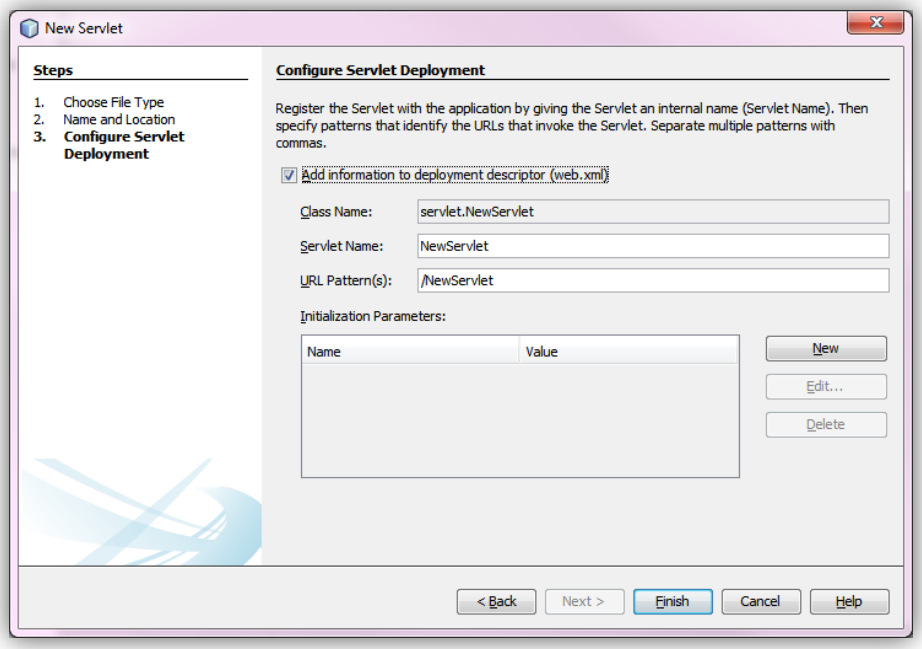
Εικόνα 11-Web Application σε Netbeans

Έπειτα πατώντας πάνω στο φάκελο του web application και επιλέγοντας το μενού New -> Servlet δημιουργήσαμε ένα servlet. Δώσαμε όνομα στο class name αλλά και στο package έτσι ώστε να κάτω από το Source Package του project να δημιουργηθεί ένας νέος φάκελος στον οποίο θα εισάγονται όσα servlets δημιουργούνται



Εικόνα 12-Δημιουργία Servlet (Μέρος Α')

Να σημειωθεί ότι πρέπει να διαλέξουμε στο επόμενο παράθυρο που θα εμφανιστεί την επιλογή **add information to deployment description (web.xml)*



Εικόνα 13-Δημιουργία Servlet (Μέρος Β')

Για την υλοποίηση της εφαρμογής του διαχειριστή (admin) χρησιμοποιήθηκε η τεχνολογία **AJAX** κάνοντας χρήση javascript βιβλιοθηκών όπως είναι η jQuery, το JSON αλλά και η ανάγνωση XML. Μέσω αυτής της τεχνολογίας κατέστη δυνατός ο ***έλεγχος ορθότητας*** πεδίου κατά την εισαγωγή νέου ονόματος βάσης στην SDB, η ***ανανέωση της σελίδας και η επιστροφή κάποια επίπεδα πίσω*** έτσι ώστε να καταστήσουμε δυνατή την τροποποίηση κάποιας επιλογής που πραγματοποιεί ο διαχειριστής και για κάποιο λόγο επιθυμεί να αλλάξει. Αυτό δεν θα μπορούσε να πραγματοποιηθεί αν αφήναμε απλά τον javascript κώδικα να ανανεώνει και να εμφανίζει συγκεκριμένα πεδία της jsp σελίδας μας, καθώς μας απενεργοποιεί τη δυνατότητα της επιστροφής σε προηγούμενη κατάσταση. Επίσης επιτυγχάνεται η εμφάνιση αποτελεσμάτων με τη χρήση ***JSON της απόκρισης του servlet*** που έχει δημιουργηθεί μετά από αίτηση κάποιας javascript συνάρτησης και τέλος η ***ανάλυση και η εμφάνιση*** των αποτελεσμάτων που βρίσκονται σε μορφή ***xml***. Στη συνέχεια αναλύεται ο κώδικας που πραγματοποιεί τα παραπάνω.

Όταν ο χρήστης επιλέξει να δημιουργήσει έναν νέο interior space ή θελήσει να τροποποιήσει έναν interior space εκτελούνται με σειρά τα παρακάτω:

- **Διάβασμα του αρχείου .owl** που βρίσκεται στο μέσα στο web application του netbeans,στην περίπτωση δημιουργίας νέου interior space διαβάζεται το αρχείο της βασικής οντολογίας (masterontology.owl) διαφορετικά γίνεται ανάγνωση του αρχείου owl που αντιστοιχεί στο όνομα της οντολογίας που διάλεξε να τροποποιήσει ο admin.

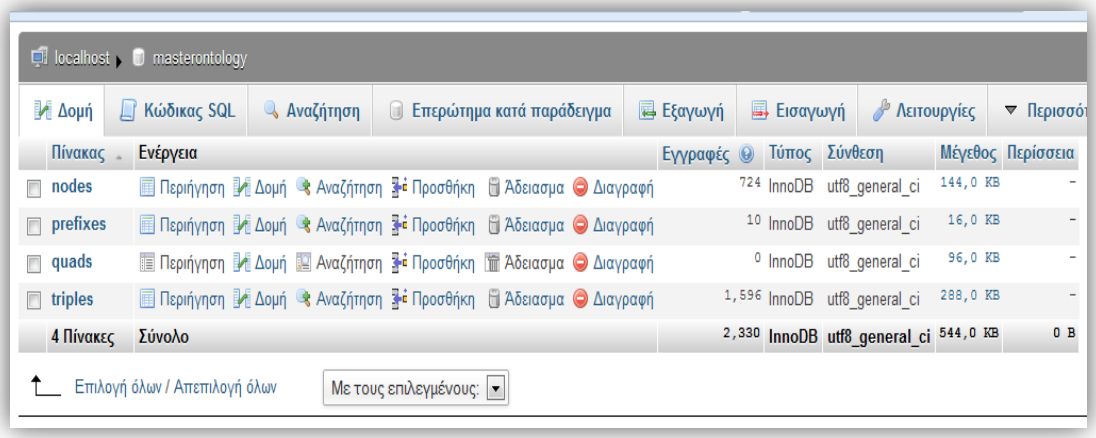
- **Αντιγραφή του περιεχομένου του masterontology.owl σε ένα νέο αρχείο** με το όνομα που έχει δώσει ο admin (στην περίπτωση δημιουργίας νέου interior space μόνο)

- **Δημιουργία βάσης δεδομένων** με το όνομα που έχει δώσει ο admin

- **Αποθήκευση** της οντολογίας **μέσω της SDB**.

```
StoreDesc storeDesc = new
StoreDesc(LayoutType.LayoutTripleNodesHash,DatabaseType.MySQL);
JDBC.loadDriverMySQL();
String jdbcURL = "jdbc:mysql://localhost:3306/" + owlName +
"?autoReconnect=true&useUnicode=true&characterEncoding=UTF-8";
SDBConnection conn = new SDBConnection(jdbcURL, "root", "");
Store store = SDBFactory.connectStore(conn, storeDesc);
store.getTableFormatter().truncate(); //αν η βάση υπάρχει ήδη χρησιμοποιείται αυτή
εντολή
store.getTableFormatter().create();
InputStream in = FileManager.get().open(owlfile);
if (in == null) {
throw new IllegalArgumentException("File: " + owlfile + " not found");
}
Model model = SDBFactory.connectDefaultModel(store);
model.read(in, "");
}
```

Πίνακας 7-Γέμισμα πίνακα με την οντολογία μέσω SDB



Εικόνα 14-Printscreen μετά τη δημιουργία και το γέμισμα της βάσης

- **Εμφάνιση των κύριων κλάσεων** μέσω της μεθόδου listHierarchyRootClasses() της Jena. Οι κλάσεις αποθηκεύονται σ' έναν iterator και αργότερα με τη χρήση json επιστρέφονται στην javascript (jQuery) συνάρτηση από την οποία και καλέστηκε το συγκεκριμένο servlet .

Μετά από αυτό έχει δημιουργηθεί στο javascript κομμάτι συνάρτηση η οποία είναι υπεύθυνη για την εμφάνιση των υποκλάσεων αλλά και των individuals συγκεκριμένης κλάσης. Πιο αναλυτικά η συνάρτηση subclasses() περνάει δύο παραμέτρους στο servlet subclassServlet. Η μία είναι το όνομα του αρχείου (το όνομα αυτό που έχει δώσει ο admin από την αρχή του wizard) και η δεύτερη παράμετρος είναι η τιμή που έχει επιλεχθεί από το select element.

- **Εμφάνιση των υποκλάσεων** :Η μέθοδος της Jena που είναι υπεύθυνη για την εύρεση των υποκλάσεων είναι η listSubClasses με όρισμα true ή false.Η μέθοδος αυτή αναγνωρίζει και έπειτα εμφανίζει τις υποκλάσεις μιας επιλεγμένης κλάσης. Η κλάση δηλώνεται μέσω της getOntClass.

- **Εμφάνιση των Individuals:** Για να μπορέσουμε να εμφανίσουμε τους individuals που αντιστοιχούν στην επιλεγμένη κλάση κάνοντας χρήση των μεθόδων getIndividual αλλά και listIndividuals

Στις περιπτώσεις που θέλουμε να προσθέσουμε ή να αφαιρέσουμε έναν individual

- **Για να προσθέσουμε individual** χρησιμοποιούμε τη μέθοδο createIndividual της Jena αφού έχουμε ορίσει την κλάση στην οποία θέλουμε να προστεθεί ο individual.

```
OntClass a1 = inf.getOntClass(ns + chosenClass);

inf.createIndividual(ns + addIndividual, a1);
```

Στη δική μας περίπτωση το chosenClass αντιστοιχεί στο όνομα της κλάσης.Μέσω της createIndividual δίνουμε το όνομα του individual που θέλουμε να προσθέσουμε (η μεταβλητή addIndividual αντιστοιχεί στο όνομα του individual)

❖ **Για να αφαιρέσουμε individual** χρησιμοποιούμε τις μεθόδους getIndividual και remove

```
Individual indOfClass = inf.getIndividual(ns + removeIndividual);

indOfClass.remove();
```

όπου στη θέση του removeIndividual μπαίνει το όνομα του individual που θέλουμε να αφαιρέσουμε από την κλάση και κατά συνέπεια και από την οντολογία. Η μέθοδος αφαίρεσης είναι η remove.

❖ **Εμφάνιση properties**

Για να εμφανίσουμε τα properties της οντολογίας χρησιμοποιούμε ελέγχους για να προσδιορίσουμε αν τα properties είναι τύπου datatype ή τύπου object. Ωστόσο για τα datatype properties εμφανίζεται ένα πρόβλημα κατά την εκτέλεση της μεθόδου getRange καθώς όταν μια τιμή (xsd μεταβλητή) είναι τύπου integer, string, float επιστρέφεται και τυπώνεται κανονικά, όταν όμως έχουμε μια λίστα με απαριθμημένα στοιχεία (enumerated list) η μέθοδος επιστρέφει null για το range. Για την επίλυση αυτού του προβλήματος κάνουμε χρήση ενός iterator που διασχίζει τις τιμές μιας απαριθμημένης λίστας με τη χρήση enumeratedClass.

```
chosenClassWithNs = ns + chosenClass;
OntClass properOfClass = inf.getOntClass(chosenClassWithNs);

ExtendedIterator itq = properOfClass.listDeclaredProperties();
      while (itq.hasNext()) {
         OntProperty allProperties = (OntProperty) itq.next();

Αν το property είναι τύπου datatype γίνονται τα παρακάτω:

         if (allProperties.isDatatypeProperty()) {
            EnumeratedClass e = null;
            ExtendedIterator<RDFNode> i = null;

Αυτή η if χρησιμοποιείται για datatypes properties που έχουν functional
επιλογές (π.χ. bedroom->has as TargetGroup for BedRoom->kids,teenager,adult)

            if (allProperties.getRange().asClass().isEnumeratedClass()) {
               e = allProperties.getRange().asClass().asEnumeratedClass();
               i = e.getOneOf().iterator();
               RDFNode prop = null;
               String s = null;
               while (i.hasNext()) {
                  prop = i.next();
                  s = prop.asLiteral().toString().split("\\^\\^")[0];   }
else {
allProperties.getRange().getLocalName().toString()}}
```

Πίνακας 8-Κώδικας για την αναγνώριση datatype properties

```
Αν το property είναι τύπου object γίνονται τα παρακάτω:

if (allProperties.isObjectProperty()) {
ExtendedIterator itq4 = allProperties.listRange();
          while (itq4.hasNext()) {
             OntClass allclasses2 = (OntClass) itq4.next();
Η παρακάτω if χρησιμοποιείται για να εμφανίζει το range σε objectProperty
που αποτελείται από περισσότερες από μια κλάσεις.
             if (allclasses2.isUnionClass() == true) {
                List<RDFNode> array = allclasses2.asUnionClass().getOperands().asJavaList();
                for (int i = 0; i < array.size(); i++) {
                array.get(i).toString().split("#")[1];}
             } else if (allclasses2.isHierarchyRoot() == false &&
allclasses2.getSubClass().toString().indexOf("Nothing") != -1) {
allclasses2.getLocalName();}
}
}
```

Πίνακας 9-Κώδικας για την αναγνώριση object properties

Για να μπορέσουμε να παρουσιάσουμε αυτά τα αποτελέσματα μέσω javascript χρειάστηκε να χρησιμοποιήσουμε την τεχνολογία του xml κατά την οποία τα δεδομένα δομούνται σε tags και είναι πιο εύκολο να αποτυπωθούν αργότερα στην jsp σελίδα.

Για να δομηθεί και να δημιουργηθεί το xml αρχείο χρησιμοποιήθηκε η παρακάτω τεχνική. Η δόμηση του xml πραγματοποιήθηκε με τρόπο τέτοιο ώστε να συμπεριλαμβάνονται όλες οι κατηγορίες των properties σε συγκεκριμένα tags.

```
Για να φτιάξουμε το xml
DocumentBuilderFactory docFactory = DocumentBuilderFactory.newInstance();
     DocumentBuilder docBuilder = null;
docBuilder = docFactory.newDocumentBuilder();

Για να ορίσουμε το root element
     org.w3c.dom.Document doc = docBuilder.newDocument();
     org.w3c.dom.Element rootElement = doc.createElement("properties");
     doc.appendChild(rootElement);

Για να προσθέσουμε elements
org.w3c.dom.Element datatypeProperties = doc.createElement("datatypeProperties");
          rootElement.appendChild(datatypeProperties);

Για να ορίσουμε χαρακτηριστικά (attributes) σε ένα element
          Attr attr = doc.createAttribute("dataPropName");
          attr.setValue(allProperties.getLocalName());
          datatypeProperties.setAttributeNode(attr);

Για την αποθήκευση xml στη μνήμη ως ένα stream
TransformerFactory transformerFactory = TransformerFactory.newInstance();
     Transformer transformer = null;
transformer = transformerFactory.newTransformer();
StringWriter stw = new StringWriter();
```

```
transformer.transform(new DOMSource(doc), new StreamResult(stw));
```

Πίνακας 10-Δημιουργία xml που αποθηκεύεται στη μνήμη

Για να αναλύσουμε το XML αρχίζουμε με την εύρεση του tag που επιθυμούμε . Ο πιο γρήγορος τρόπος είναι κάνοντας χρήση των selector της jQuery . Το πρώτο πράγμα που θα γράψουμε είναι η **find** για να πάρουμε το element που θέλουμε. Στη συνέχεια, καλούμε την **each** , η οποία επαναλαμβάνει την αναζήτηση σε κάθε στοιχείο και μέσω του **this** αναγνωρίζουμε το περιεχόμενο της τρέχουσας θέσης. Για να πάρουμε στην συνέχεια της τιμή ενός attribute εκτελούμε την **attr**. Έπειτα επαναλαμβάνουμε την ίδια διαδικασία με find και each . Τέλος για να πάρουμε το περιεχόμενο ενός XML element χρησιμοποιούμε την **text**.

```
            var $prop = $('#propertiesDiv');
            $prop.empty();
            parser = new DOMParser();
            xmlDoc = parser.parseFromString(xmlhttp.responseText, "text/xml");
            $(xmlDoc).find("objectProperties").each(function()
            {
               count++;
               $("#propertiesDiv").append("<b>Object Property Name: </b>" + "<div
id='propName" + count + "'>" + $(this).attr("objPropName") + "</div>");
               var $select = $('<select multiple id="forStatement' + count +
'">').appendTo($('#propertiesDiv'));
               $(this).find("individualsOfClass").each(function() {
                  $select.append("<option>" + $(this).text() + "</option>");
               });
               $("#propertiesDiv").append("<button type='button' onclick='objectTriples(" +
count + ");'>Statement...</button>" + "<br/>");
               $("#propertiesDiv").append("</select>");
               $("#propertiesDiv").append("<br />");
            });
               });
               $("#propertiesDiv").append("<br />");
                                                   …
                                                   …
            });
```

Πίνακας 11-Javascript κώδικας για εμφάνιση των properties από xml

❖ Δημιουργία statement

Για να δημιουργήσουμε statement χρειάζεται να πάρουμε τρία στοιχεία που θα χρησιμοποιηθούν στις παραμέτρους της μεθόδου createStatement

Τα στοιχεία αναφέρονται ως (subject,predicate,object)

- Το ***subject*** είναι το resource στο οποίο αντιστοιχεί ένας individual από την οντολογία
- Το ***predicate*** είναι το όνομα του property και
- Το ***object*** είναι η τιμή του property

Ο κώδικας της jena για την προσθήκη statements με object properties στην οντολογία μας είναι:

```
    statementSubject = inf.getResource(ns + subject);
    statementPredicate = inf.getObjectProperty(ns + predicate);
        for (int i = 0; i < object.length; i++) {
           statementObject = inf.getResource(ns + object[i]);
           System.out.println(statementObject);
           Statement s = ResourceFactory.createStatement(statementSubject,
statementPredicate, statementObject);
           inf.add(s);
        }
```

Πίνακας 12-Κώδικας σε Jena για δημιουργία statement με Object Property

Και ο αντίστοιχος κώδικας για την προσθήκη των statements με datatype properties στην οντολογία μας είναι:

```
statementSubject = inf.getResource(ns + subject);
 statementPredicate = inf.getDatatypeProperty(ns + predicate);
        if (XSDType.equalsIgnoreCase("string")) {
           Literal literalObject = inf.createTypedLiteral(object, XSDDatatype.XSDstring);
           Statement s = ResourceFactory.createStatement(statementSubject,
statementPredicate, literalObject);
           inf.add(s);
        } else if (XSDType.equalsIgnoreCase("float")) {
           Literal literalObject = inf.createTypedLiteral(object, XSDDatatype.XSDfloat);
           Statement s = ResourceFactory.createStatement(statementSubject,
statementPredicate, literalObject);
           inf.add(s);
        } else {
           Literal literalObject = inf.createTypedLiteral(object, XSDDatatype.XSDint);
           Statement s = ResourceFactory.createStatement(statementSubject,
statementPredicate, literalObject);
           inf.add(s);
        }
```

Πίνακας 13-Κώδικας σε Jena για δημιουργία statement με Datatype property

Javascript κώδικας για εμφάνιση αποτελέσματος από servlet με συγκεκριμένη μορφή (π.χ. drop –down μενού)

Για να μπορέσουμε να εμφανίσουμε το αποτέλεσμα της jena σε μορφή drop-down μενού, δημιουργούμε ένα select element δίνοντας κάποιο όνομα όπως φαίνεται παρακάτω:

```
<div id="classes">
  <h1>What would you like to annotate?</h1>
  <select id="selectClass"></select>
  <button type="button" onclick=" ">OK</button>
</div>
```

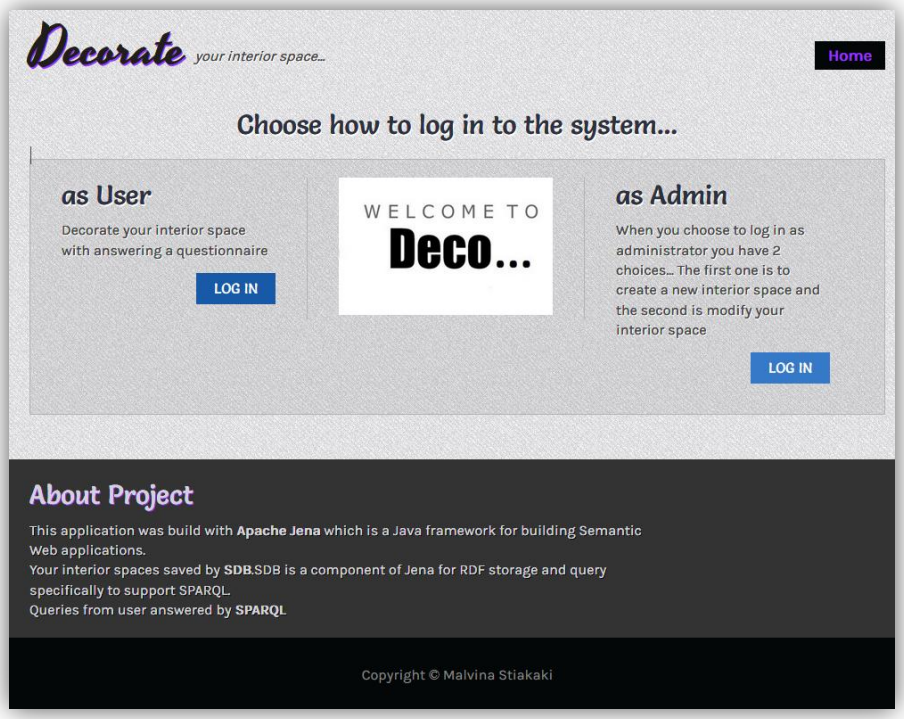
Και έπειτα μέσα στον javascript κώδικα αποδίδουμε σε μία μεταβλητή το select element var $majorClass = $('#selectClass'); και στη συνέχεια του προσθέτουμε περιεχόμενο <option> την απόκριση (responseJson) του servlet.

```
function hierarchyRoot() {
  var file = $('# saveName).val();
  $.get('classServlet', {owl: file}, function(responseJson) {
    $('#classes').show();
    $('#files').hide();
    var $majorClass = $('#selectClass');
    $majorClass.find('option').remove();
    $.each(responseJson, function(index, item) {
      $('<option>').text(item).appendTo($majorClass);
    });
  });
}
```

Πίνακας 14-jQuery για εμφάνιση κλάσεων στον browser μέσω drop-down μενού

Ο κώδικας της συνάρτησης που εμφανίζεται παραπάνω για την απόδοση του αποτελέσματος σε select element είναι σχεδόν ίδιος για όλα το wizard που έχει δημιουργηθεί.

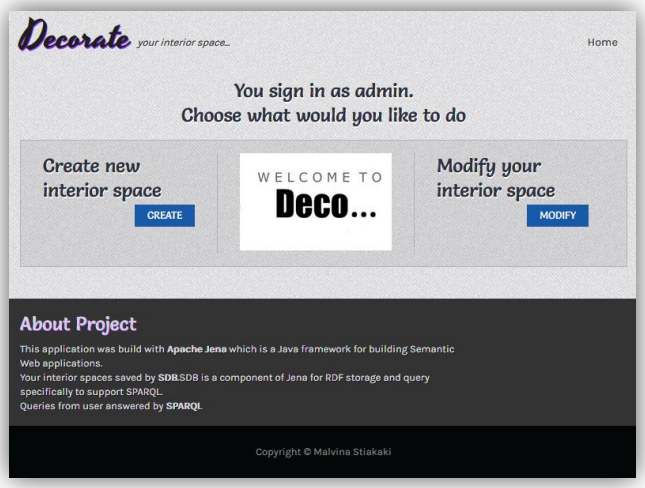
5.2 Αποτέλεσμα εφαρμογής “Wizard for Admin”



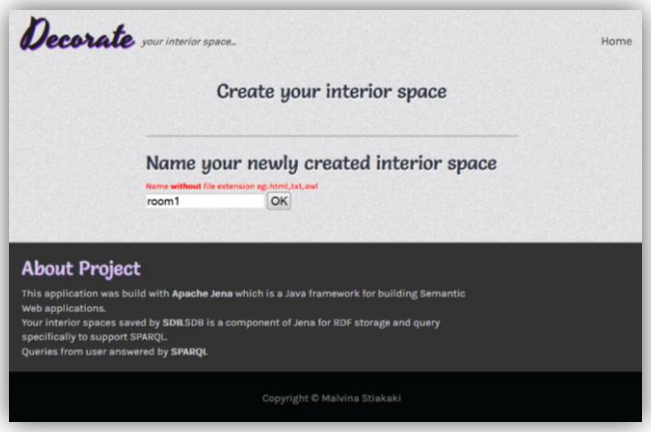
Εικόνα 15-Αρχική σελίδα Wizard

Η παραπάνω εικόνα είναι η αρχική σελίδα της web εφαρμογής. Σ’ αυτό το στάδιο ο χρήστης επιλέγει αν θέλει να εισαχθεί στο σύστημα ως απλός χρήστης ή ως administrator. Για αυτή την ενότητα της πτυχιακής επιλέγουμε να εισαχθούμε στο σύστημα ως administrator.

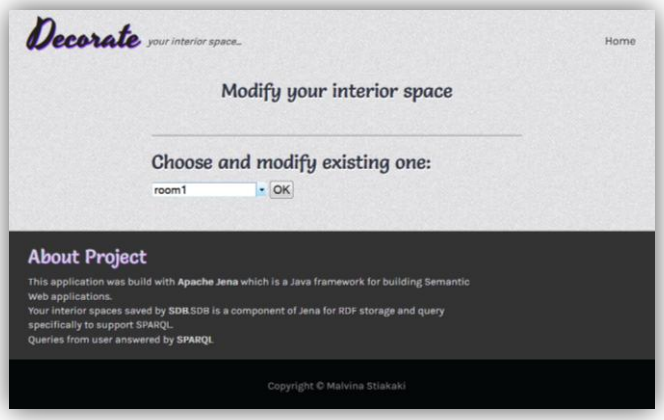
Η επόμενη εικόνα μας παρουσιάζει τις δύο επιλογές που αποκτά κάποιος επιλέγοντας να μπει στο σύστημα ως administrator. Η πρώτη είναι να δημιουργήσει ένα νέο χώρο εσωτερικής διακόσμησης και η δεύτερη επιλογή είναι να τροποποιήσει έναν υπάρχοντα χώρο εσωτερικής διακόσμησης.



Εικόνα 16-Επιλογές admin

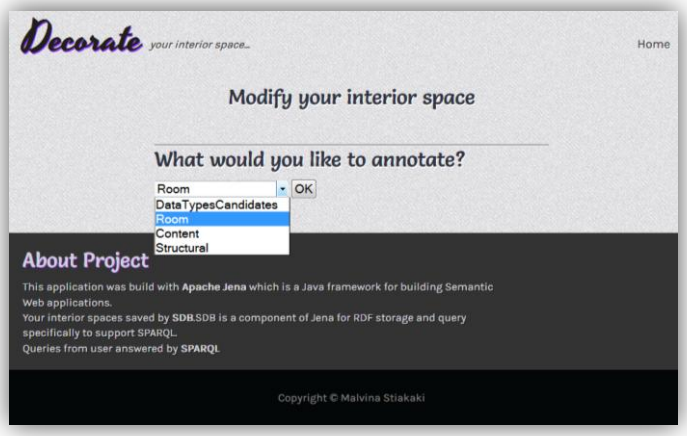


Εικόνα 17-Δημιουργία νέου χώρου εσωτερικής διακόσμησης (από admin)

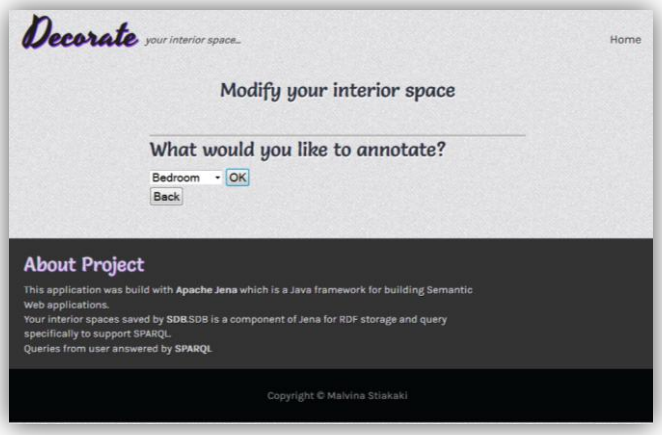


Εικόνα 18-Τροποποίηση υπάρχοντα χώρου εσωτερικής διακόσμησης (admin)

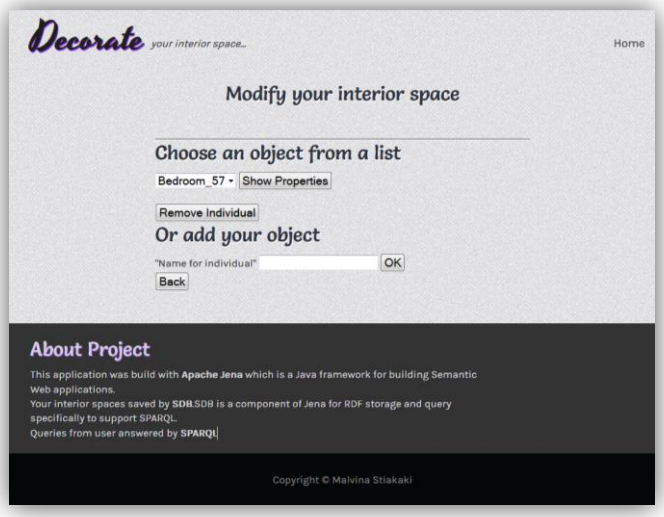
Από την επόμενη εικόνα και μετά παρουσιάζονται όλες οι δυνατότητες που έχει ο χρήστης για να ορίσει ή να τροποποιήσει ένα χώρο εσωτερικής διακόσμησης.



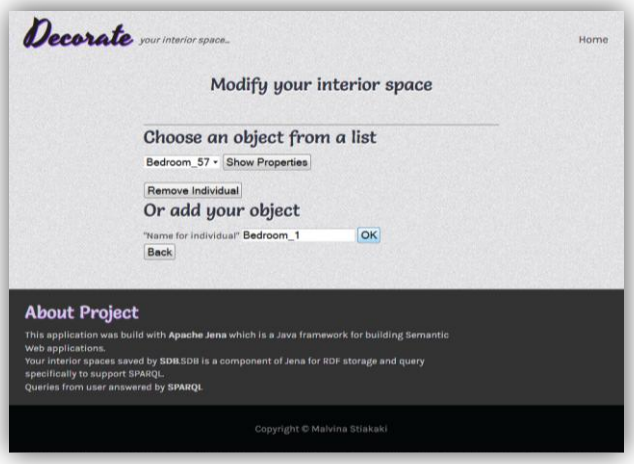
Εικόνα 19-Κύριες κλάσεις οντολογίας σε drop down μενού



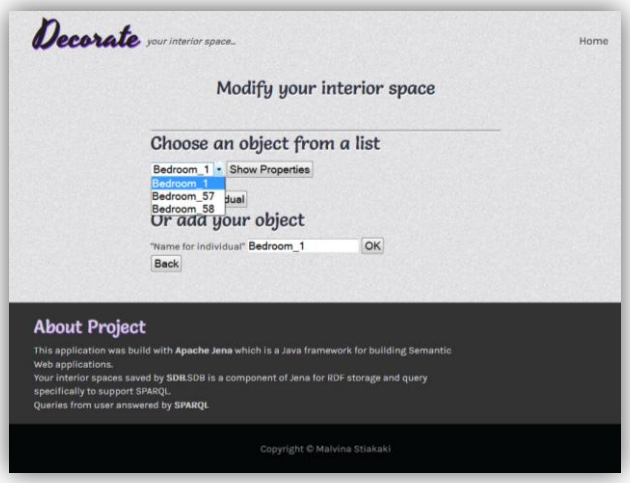
Εικόνα 20-Υποκλάσεις οντολογίας σε drop down μενού



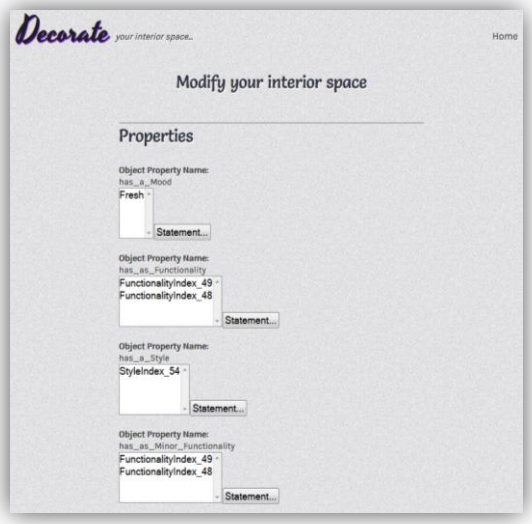
Εικόνα 21-Individuals οντολογίας σε drop down μενού



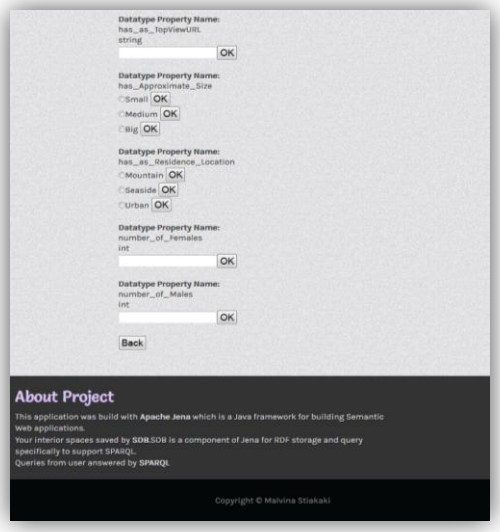
Εικόνα 22-Προσθήκη individual στην οντολογία



Εικόνα 23-Drop down μενού μετά από προσθήκη individual



Εικόνα 24-Εμφάνιση properties (object properties σε multiple select)



Εικόνα 25-Εμφάνιση properties (datatype properties σε radio buttons και input texts)

6 ΕΦΑΡΜΟΓΗ "WIZARD FOR USER"

Στην εφαρμογή που αναπτύχθηκε ο διαχειριστής ορίζει τα πεδία της φόρμας, και ο επισκέπτης μπορεί να συμπληρώσει τη φόρμα και να την υποβάλει. Η παραγόμενη φόρμα με τα συμπληρωμένα πεδία στέλνεται σε μία ρουτίνα που τα μετατρέπει σε ερωτήματα γλώσσας ερωτημάτων σημασιολογικού χαρακτήρα και εμφανίζει τα αντίστοιχα αποτελέσματα. Στην συγκεκριμένη πτυχιακή η γλώσσα ερωτημάτων που έχει χρησιμοποιηθεί είναι η sparql.

Θα πρέπει να σημειωθεί ότι η διαδικασία αυτή χρειάζεται αρκετή εσωτερική μνήμη για να μπορέσει να αποδώσει τα μέγιστα βγάζοντας αποτελέσματα. Ωστόσο αυτό είναι αρκετά δύσκολο να πραγματοποιηθεί σε οικιακούς υπολογιστές. Οι απαιτήσεις της μνήμης είναι πολύ μεγαλύτερες από αυτές που ένας οικιακός υπολογιστής διαθέτει γι 'αυτό και πολλές φορές εμφανίζονται μηνύματα της μορφής "out of memory".

Συγκεκριμένα οι Proudfoot, D., & Copeland, B. J. [5] αναφέρουν σε μια δημοσίευσή τους ότι:

"It's nice to be reminded that the concept of computational decidability is based on abstract Turing machines with infinite memory"

δηλαδή σε μετάφραση:

"Είναι καλό να θυμόμαστε ότι η έννοια της υπολογιστικής απόκρισης είναι βασισμένη πάνω στην αφηρημένη μηχανή Turing με την άπειρη μνήμη"

6.1 Υλοποίηση "Wizard for User"

Το πρώτο κομμάτι που εμφανίζεται στο χρήστη καθώς εισάγεται στο σύστημα ως user είναι ένα μενού που τον ρωτάει για τον τύπο του δωματίου που θέλει να διακοσμήσει. Ανάλογα με την επιλογή που θα κάνει θα ανακατευθυνθεί σ' ένα servlet το οποίο είναι υπεύθυνο

- για να την ***εμφάνιση*** του αντίστοιχου ***ερωτηματολογίου***
- την ***εκτέλεση του ερωτήματος*** μέσω της γλώσσας ***sparql*** και
- την ***παρουσίαση του αποτελέσματος*** στο χρήστη

[5] Proudfoot, D., & Copeland, B. J. (1994). Turing, Wittgenstein and the science of the mind.

Για την εμφάνιση των ερωτηματολογίων ανάλογα με τον τύπο δωματίου που έχει επιλέξει ο χρήστης χρησιμοποιήθηκαν συγκεκριμένες μέθοδοι:

- Η πρώτη μέθοδος αντιστοιχεί στην εμφάνιση των datatype properties
- Η δεύτερη μέθοδος αντιστοιχεί στην εμφάνιση των object properties και
- Η τρίτη μέθοδος αντιστοιχεί στην εμφάνιση individuals μιας κλάσης

Οι ερωτήσεις του ερωτηματολογίου δομούνται μέσω αυτών των μεθόδων. Οι εικόνες από το αποτέλεσμα της εφαρμογής φαίνονται στην επόμενη ενότητα αλλά εδώ θα αναφέρουμε απλά ότι μέσω των κλήσεων των παραπάνω μεθόδων εμφανίζονται τα αποτελέσματα μέσω radio buttons, select elements αλλά και text areas.

Έπειτα από τη σύνταξη αυτών των ερωτηματολογίων , ο ρόλος του διαχειριστή συνεχίζεται καθώς πρέπει να συντάξει το sparql query το οποίο θα επιστρέψει το δωμάτιο που θα έχει τις ίδιες επιλογές με αυτές που έχει επιλέξει ο χρήστης.

Το Sparql query που χρησιμοποιήθηκε στα πλαίσια αυτής της εργασίας έχει την παρακάτω μορφή.

```
String queryString
            = "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> \n"
            + "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  \n"
            +"PREFIX
ns:<http://www.semanticweb.org/ontologies/2009/5/Ontology1244033197062.owl#>  \n"
            +"PREFIX
Ontology1244033197062:<http://www.semanticweb.org/ontologies/2009/5/Ontology1244
033197062.owl#> \n"
            + "PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>  \n"
            + "SELECT ?room \n"
            + "WHERE {\n"
           + "?room ns:has_a_Style ns:" + choice3 + " ;\n"
            + "ns:has_as_Functionality ns:" + choice6 + " ;\n"
            + "ns:has_as_Minor_Functionality ns:" + choice7 + " ;\n"
            + "Ontology1244033197062:has_as_Residence_Type '" + choice1 + "'^^xsd:string
;\n"
            + "Ontology1244033197062:has_as_Residence_Location '" + choice2 +
"'^^xsd:string ;\n"
            + "Ontology1244033197062:has_as_TopViewURL '" + choice4 + "'^^xsd:string
;\n"
            + "Ontology1244033197062:has_Approximate_Size '" + choice5 + "'^^xsd:string
;\n"
            + "Ontology1244033197062:has_as_TargetGroup_for_BedRoom '" + choice8 +
"'^^xsd:string ;\n"
            + "Ontology1244033197062:number_of_Males '" + choice10 + "'^^xsd:int ;\n"
            + "Ontology1244033197062:number_of_Females '" + choice11 + "'^^xsd:int .\n"
            + "?bed Ontology1244033197062:has_DoubleBed '" + choice12 + "'^^xsd:string
\n"
            + "}";
     Query query = QueryFactory.create(queryString);
```

```
Dataset ds = DatasetStore.create(store);
QueryExecution qe = QueryExecutionFactory.create(query, ds);
ds.supportsTransactions();
ResultSet rs = qe.execSelect();
roomName = readXMLFile(db, rs);
qe.close();
```

Πίνακας 15-Sparql query

Στο παραπάνω Sparql query βλέπουμε στην αρχή να δηλώνονται τα prefixes της οντολογίας μας. Στη συνέχεια μέσω της δομής SELECT-FROM-WHERE,που θυμίζει αρκετά τη σύνταξη sql ερωτημάτων ζητάμε να παρουσιαστεί το όνομα του δωματίου (living room ή bedroom) που ταιριάζει στις επιλογές του χρήστη.

Συγκεκριμένα μέσω του SELECT ορίζεται η προβολή, δηλαδή η συγκεκριμένη πληροφορία που επιθυμεί να ανακτήσει ο χρήστης, μέσω του FROM, που δεν υπάρχει στο παραπάνω query καθώς είναι ένα προαιρετικό πεδίο, ορίζεται το RDF έγγραφο στο οποίο θα γίνει η ερώτηση και μέσω του WHERE τίθενται οι περιορισμοί στο σύνολο των πιθανών απαντήσεων του ερωτήματος. Τα has_Approximate_Size , has_a_Style είναι properties της οντολογίας μας και τα choice1,2,3…. που εμφανίζονται στον παραπάνω κώδικα δεν είναι τίποτα άλλο παρά οι επιλογές του χρήστη, οι οποίες έχουν γίνει και έχουν αποσταλεί στο servlet. Η ερώτηση θα επιστρέψει ένα αποτέλεσμα το οποίο θα είναι η μεταβλητή ?room. Πιο αναλυτικά καθώς θα εκτελείται το Sparql ερώτημα στη βάση δεδομένων θα επιστραφούν τα δωμάτια που θα βρεθούν να έχουν τριπλέτες που να αντιστοιχούν στις τιμές που έχει διαλέξει ο χρήστης κάνοντας χρήση του ερωτηματολογίου. Το ερώτημα έχει άλλη μια μεταβλητή η οποία είναι το ?bed και ουσιαστικά αποτελεί άλλο ένα περιορισμό στο αποτέλεσμα του δωματίου καθώς για να είναι έγκυρο ένα δωμάτιο θα πρέπει να περιέχει εκτός από τα properties που έχει επιλέξει ο χρήστης για το δωμάτιο και επιπρόσθετες πληροφορίες οι οποίες θα αναφέρουν αν διαθέτει ή όχι διπλό κρεβάτι (αυτό ισχύει στον παραπάνω κώδικα που αντιστοιχεί στην αναζήτηση bedroom καθώς εκεί υπάρχουν κρεβάτια για να μπορεί να γίνει αντιστοίχηση αποτελέσματος. Στο living room δεν υπάρχει η δεύτερη μεταβλητή και κατά συνέπεια και αυτό το property)

Το αποτέλεσμα τοποθετείται στο ResultSet. (Λίγο παρακάτω γίνεται αναφορά σε όλες τις κλάσεις που χρησιμοποιούνται στον παραπάνω κώδικα συμπεριλαμβανομένου και του ResultSet). Η συνάρτηση readXMLFile δημιουργήθηκε για να μπορούμε να επιστρέφουμε τα αποτελέσματα του ResultSet με κάποιο τρόπο στο browser.

```
public String readXMLFile(String db, ResultSet rs) throws FileNotFoundException,
ParserConfigurationException, IOException, SAXException {
    String roomFile = null;
    File dir = new File(getServletContext().getRealPath("/owl/"));
    OutputStream out = new FileOutputStream(dir.getAbsolutePath() + "/" + db + ".xml");
    ResultSetFormatter.outputAsXML(out, rs);

    DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
    DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
```

```
    Document doc = dBuilder.parse(dir.getAbsolutePath() + "/" + db + ".xml");
    doc.getDocumentElement().normalize();
    NodeList nList = doc.getElementsByTagName("results");
    for (int temp = 0; temp < nList.getLength(); temp++) {
       Node nNode = nList.item(temp);
       System.out.println("\nCurrent Element :" + nNode.getNodeName());
       if (nNode.getNodeType() == Node.ELEMENT_NODE) {
          Element eElement = (Element) nNode;
          System.out.println("Room:                                    "                                    +
eElement.getElementsByTagName("uri").item(0).getTextContent());
          roomFile = eElement.getElementsByTagName("uri").item(0).getTextContent();
       }
    }
    return roomFile;
  }
```

Πίνακας 16-Συνάρτηση για διάβασμα ResultSet από xml

Ο παραπάνω κώδικας παρουσιάζει τον τρόπο με τον οποίο αποθηκεύσαμε τα αποτελέσματα σε xml αρχείο και αργότερα πως καταφέραμε να αντλήσουμε τα περιεχόμενα του tag uri,στο οποίο εμπεριέχονται τα δεδομένα που προέρχονται από το ResultSet.

SPARQL QUERY-Ανάλυση κλάσεων

Για να μπορέσει να γίνει κατανοητός ο τρόπος λειτουργίας ενός Sparql query θα πρέπει να αναφέρουμε ότι η λογική των ερωτήσεων με τη χρήση SPARQL μοιάζει με τη λογική των ερωτήσεων σε σχεσιακές βάσεις δεδομένων. Η διαφορά τους έγκειται στο γεγονός ότι δεν ξέρουμε τα attributes που έχουμε, ξέρουμε μόνο ότι ψάχνουμε σε ένα dataset σε μορφή γράφου.

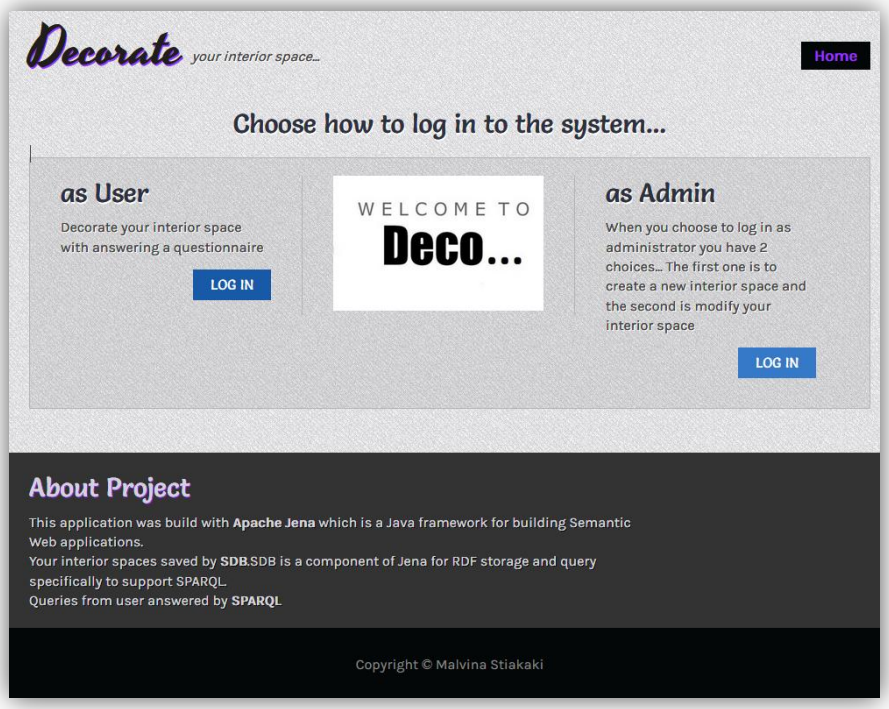
Το framework της Jena ενσωματώνει μια μηχανή ερωτήσεων SPARQL που ονομάζεται ARQ. Το ARQ υποστηρίζει όλα τα χαρακτηριστικά του SPARQL και κάποια επιπλέον χαρακτηριστικά, όπως ελεύθερη αναζήτηση κειμένου, η πρόσβαση και επέκταση της SPARQL άλγεβρας, η δυνατότητα ενημέρωσης ενός RDF γράφου, η υποστήριξη συναρτήσεων FILTER, η υποστήριξη απομακρυσμένης πρόσβασης σε υπηρεσίες SPARQL, η δυνατότητα επέκτασης σε άλλα συστήματα αποθήκευσης και άλλα. Το ARQ μπορεί να γίνει προσβάσιμο από το Jena κυρίως μέσα από το πακέτο `com.hp.hpl.jena.query`. Το πακέτο αυτό περιέχει τις εξής κλάσεις:

- **Query:** η κλάση που αναπαριστά ένα SPARQL ερώτημα. Αντικείμενα αυτής της κλάσης δημιουργούνται καλώντας μία από τις μεθόδους της κλάσης `QueryFactory`, που παρέχουν πρόσβαση σε διάφορους parsers.
- **QueryExecution:** αναπαριστά μία εκτέλεση του ερωτήματος
- **QueryExecutionFactory:** μέρος που χρησιμοποιείται για την τοποθέτηση στιγμιότυπων της `QueryExecution`
- **DatasetFactory:** χρησιμοποιείται για τη δημιουργία συνόλων δεδομένων (Datasets), συμπεριλαμβανομένης της δημιουργίας μιας `DataSource` το οποίο είναι ένα επεκτάσιμο `Dataset`.

Για ερωτήματα `SELECT`, περιλαμβάνει τις κλάσεις:

- **QuerySolution:** μια μοναδική απάντηση στο ερώτημα
- **ResultSet:** Όλες οι απαντήσεις του ερωτήματος σ' έναν iterator
- **ResultSetFormatter:** μετατροπή ενός ResultSet σε διάφορες μορφές, για παράδειγμα κείμενο, RDF γράφος ή απλό XML

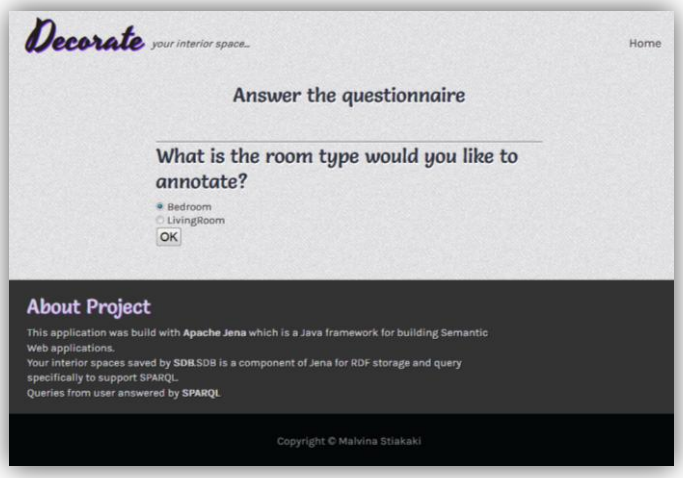
6.2 Αποτέλεσμα εφαρμογής "Wizard for User"



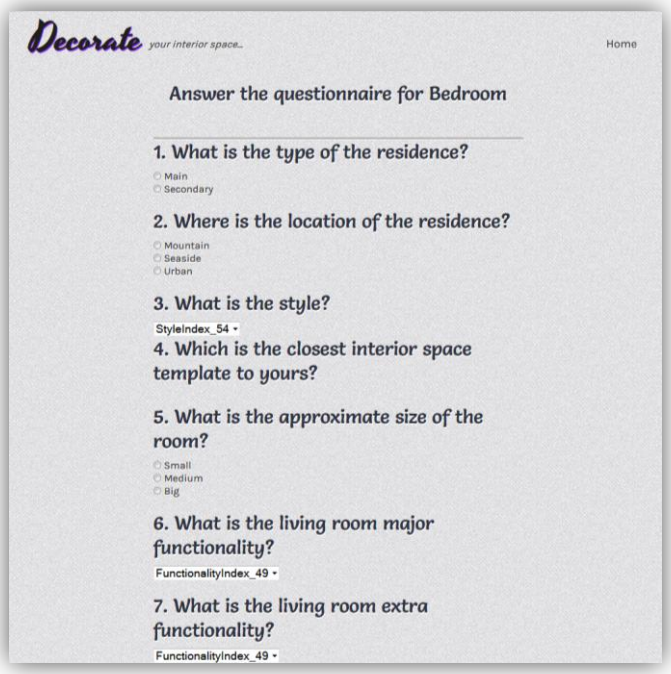
Εικόνα 26-Αρχική σελίδα Wizard

Στην παραπάνω εικόνα παρουσιάζεται η αρχική σελίδα της εφαρμογής. Σε αυτό το σημείο της πτυχιακής ο χρήστης επιλέγει την εισαγωγή του ως απλός χρήστης (as User). Έπειτα από αυτό το σημείο αρχίζει η περιήγηση του στο Wizard αφού απαντώντας σε συγκεκριμένες ερωτήσεις οι οποίες έχουν οριστεί από τον admin θα μπορέσει να βρει τον εσωτερικό χώρο που έχει ως στοιχεία του όλα τα επιλεγμένα πεδία της φόρμας που έχει υποβάλει ο ίδιος.

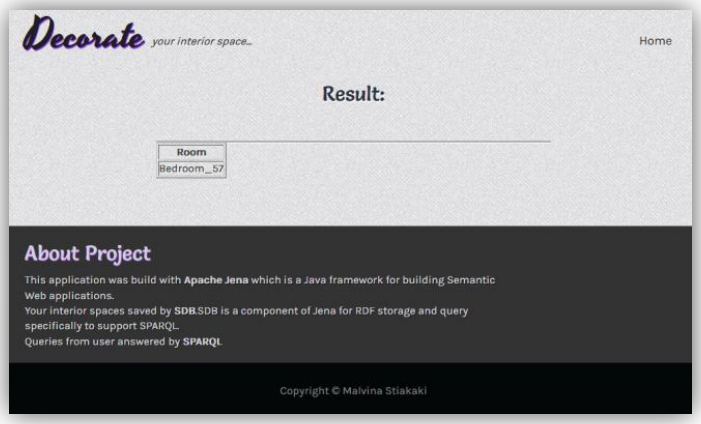
Στα επόμενα printscreens παρουσιάζεται η εξέλιξη του wizard. Οι ερωτήσεις απαντώνται από το χρήστη και μετά υποβάλλονται σε μια ρουτίνα. Η ρουτίνα είναι γραμμένη σε γλώσσα sparql και είναι υπεύθυνη για την παρουσίαση του αποτελέσματος πίσω στο χρήστη.



Εικόνα 27-Wizard χρήστη (Ερωτηματολόγιο 1)



Εικόνα 28-Wizard χρήστη (Ερωτηματολόγιο 2)



Εικόνα 29-Παρουσίαση αποτελέσματος μετά από sparql ερώτημα

7 ΣΥΜΠΕΡΑΣΜΑΤΑ

Στα πλαίσια αυτής της πτυχιακής αναπτύχθηκε εφαρμογή η οποία επιτυγχάνει την ανάγνωση, την επεξεργασία και την εγγραφή RDF δεδομένων σε σχεσιακή βάση μέσω της JENA αλλά και την υποβολή ερωτημάτων μέσω της SPARQL . Τα δεδομένα αυτά, αποτελούν μια οντολογία εσωτερικής διακόσμησης σπιτιού που έχει αναπτυχθεί με το protégé και βρίσκεται σε μορφή OWL-DL. Η πτυχιακή απαρτίζεται από δύο επίπεδα χρηστών. Το πρώτο επίπεδο είναι ο admin ο οποίος είναι υπεύθυνος για την διαχείριση της οντολογίας, δημιουργώντας ή τροποποιώντας τη. Στο επίπεδο του admin έχουν χρησιμοποιηθεί τεχνολογίες AJAX, JENA και SDB. Η AJAX τεχνολογία επιτυγχάνει την ανανέωση συγκεκριμένων σημείων της jsp σελίδας με αποτέλεσμα η εφαρμογή να μειώνει το χρόνο φόρτωσής της αλλά συγχρόνως ακυρώνει τη δυνατότητα επιστροφής της σε προηγούμενο επίπεδο. Η επιτυχής φόρτωση της οντολογίας γίνεται μέσω του framework της JENA. Οι μέθοδοί της βοηθούν στο να γίνει εφικτή η αναγνώριση των συστατικών μερών της οντολογίας παρουσιάζοντας κλάσεις, υποκλάσεις ,individuals αλλά και statements. Στο δεύτερο επίπεδο βρίσκεται ο user. Στο επίπεδο αυτό ο χρήστης συμπληρώνει φόρμες για τα χαρακτηριστικά που θέλει να έχει ένας εσωτερικός χώρος διακόσμησης και έπειτα τα υποβάλει σε γλώσσα ερωτημάτων η οποία είναι η SPARQL. Το αποτέλεσμα που εμφανίζεται είναι η διακοσμητική πρόταση που υπάρχει σε κάποιο από τα δωμάτια που έχει δημιουργήσει ο admin ήδη στη βάση. Θα πρέπει να σημειωθεί ότι η διαδικασία αυτή χρειάζεται αρκετή εσωτερική μνήμη για να μπορέσει να αποδώσει τα μέγιστα και να εμφανίσει αποτελέσματα γι’ αυτό και πολλές φορές εμφανίζονται μηνύματα για αδυναμία συνέχειας του wizard λόγω γεμίσματος της εσωτερικής μνήμης του υπολογιστή.

8 ΒΙΒΛΙΟΓΡΑΦΙΑ-ΑΝΑΦΟΡΕΣ

OWL

Athanasios G.Malamos, Paraskevi V. Sympa, Georgios S.Mamakis, "Xml Annotation Of Conceptual Characteristics In Interior Decoration", 6th International Conference, New Horizons in Industry, Business and Education (NHIBE 2009), 27 - 28 August 2009, Santorini (http://www.medialab.teicrete.gr/media/papers/nhibe2009.pdf)

SEMANTIC WEB

A Semantic Web Primer, Grigoris Antoniou and Frank van Harmelen, 2004

A Semantic Web Primer, second edition,Grigoris Antoniou and Frank van Harmelen, 2008

http://el.wikipedia.org/wiki/%CE%A3%CE%B7%CE%BC%CE%B1%CF%83%CE%B9%CE%BF%CE%BB%CE%BF%CE%B3%CE%B9%CE%BA%CF%8C%CF%82_%CE%99%CF%83%CF%84%CF%8C%CF%82

http://users.iit.demokritos.gr/~alexv/presentations/SemanticWeb_v1.pdf

http://dsmc.eap.gr/semweb.php

http://www.technicalreview.gr/index.php?option=com_content&task=view&id=684

http://www.google.gr/url?sa=t&rct=j&q=&esrc=s&source=web&cd=11&ved=0CCkQFjAAOAo&url=http%3A%2F%2Fwww.ebusinessforum.gr%2Fengine%2Findex.php%3Fop%3Dmodload%26modname%3DDownloads%26action%3Ddownloadsviewfile%26ctn%3D1999%26language%3Del&ei=JbbdUcKXE8qxPLX7gfAG&usg=AFQjCNEfPp2G-6drpdfN38KIa0QVaSAmmg&cad=rja

http://www.google.gr/url?sa=t&rct=j&q=&esrc=s&source=web&cd=50&ved=0CFsQFjAJOCg&url=http%3A%2F%2Fwww.intelligence.tuc.gr%2F~petrakis%2Fcourses%2Fmultimedia%2Ftutorials%2FOWL%2Fsemantic%2520web2.ppt&ei=ILjdUejbBYK5O_XmgPgJ&usg=AFQjCNGB2iZb8l3P_K_BuPQ9oOij5HVFwQ

JENA

http://jena.apache.org/

http://www.slideshare.net/onlyjiny/jena-programming

http://rad.ihu.edu.gr/fileadmin/labsfiles/knowledge_management/TUTORIALS/Jena.pdf

SDB

http://jena.apache.org/documentation/sdb/

http://www.bioontology.org/wiki/images/6/6a/Triple_Stores.pdf

JAVASCRIPT (AJAX) - JQUERY

http://tech.pro/tutorial/877/xml-parsing-with-jquery

http://dspace.lib.uom.gr/bitstream/2159/15626/3/ChamalidouMyrophoraMirelaMsc2013.pdf

http://www.idesigner.gr/%CE%B5%CE%B9%CF%83%CE%B1%CE%B3%CF%89%CE%B3%CE%AE-%CF%83%CF%84%CE%B7%CE%BD-jquery/

SQL

http://vivliothmmy.ee.auth.gr/269/1/DeeBee_%CE%91%CE%BD%CF%84%CE%B9%CE%BA%CE%B5%CE%B9%CE%BC%CE%B5%CE%BD%CE%BF%CF%83%CF%84%CF%81%CE%B1%CF%86%CE%AE%CF%82_%CE%94%CE%B9%CE%B1%CF%87%CE%B5%CE%AF%CF%81%CE%B9%CF%83%CE%B7_%CE%A3%CE%B7%CE%BC%CE%B1%CF%83%CE%B9%CE%BF%CE%BB%CE%BF%CE%B3%CE%B9%CE%BA%CF%8E%CE%BD_%CE%94%CE%B5%CE%B4%CE%BF%CE%BC%CE%AD%CE%BD%CF%89.pdf

SPARQL

Proudfoot, D., & Copeland, B. J. (1994). Turing, Wittgenstein and the science of the mind.

http://www.slideshare.net/apohllo/presentationjena-a-semantic-web-framework-for-java

http://www.music.tuc.gr/GetFile?FILE_TYPE=PUB.FILE&FILE_ID=340

http://estia.hua.gr:8080/dspace/bitstream/123456789/1898/1/Rista,%20Mario.pdf

http://nemertes.lis.upatras.gr/jspui/bitstream/10889/2603/1/diplomatiki.pdf

XML

http://www.google.gr/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CC0QFjAA&url=http%3A%2F%2Fwww.music.tuc.gr%2FGetFile%3FFILE_TYPE%3DPUB.FILE%26FILE_ID%3D340&ei=fznkUcTmOYO8Pa6QgegN&usg=AFQjCNGQylCOydG8bdHuhdN3TnRry4whSg&bvm=bv.48705608,d.ZWU

http://www.w3schools.com/xml/

JSON

http://www.json.org/

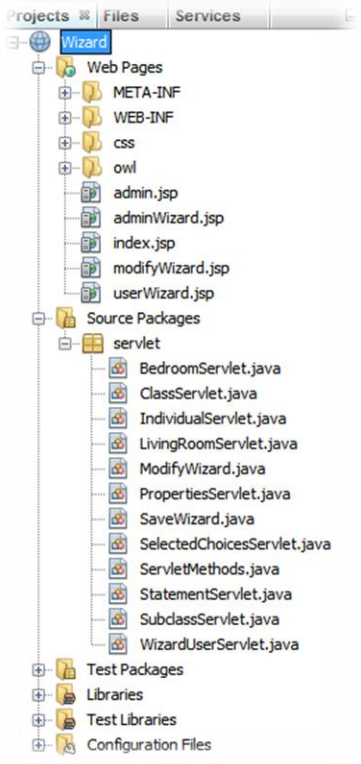
http://www.w3schools.com/json/

RDF

RDF Primer , W3C Recommendation 10 February 2004 (http://www.w3.org/TR/rdf-primer/)

9 ΠΑΡΑΡΤΗΜΑ

Στο παράρτημα αυτό θα γίνει αποτύπωση των μεθόδων που έχουν χρησιμοποιηθεί στην εφαρμογή “Wizard” και συγκεκριμένα στα servlets της εφαρμογής.



Εικόνα 30-Δέντρο εφαρμογής "Wizard" μέσω του netbeans

BedroomServlet

Method Detail

doGet

```
protected void doGet(javax.servlet.http.HttpServletRequest request,
          javax.servlet.http.HttpServletResponse response)
               throws javax.servlet.ServletException,
                      java.io.IOException
```

Create a questionnaire for bedroom interior space based on specific characteristics of the ontology.

Overrides:

`doGet` in class `javax.servlet.http.HttpServlet`

Parameters:

`request` - This servlet is called from WizardUserServlet

`response` - The output result is presented to the user in HTML form

Throws:

`javax.servlet.ServletException`

`java.io.IOException`

ClassServlet

Method Detail

doGet

```
protected void doGet(javax.servlet.http.HttpServletRequest request,
          javax.servlet.http.HttpServletResponse response)
               throws javax.servlet.ServletException,
                      java.io.IOException
```

This method displays the direct subclasses of owl:Thing

Overrides:

doGet in class javax.servlet.http.HttpServlet

Parameters:

request - This servlet is called from any jsp page

response - Results come in JSON format contained into an Arraylist

Throws:

javax.servlet.ServletException

java.io.IOException

```
List<String> list2 = new ArrayList<String>();
    for (Iterator<OntClass> i = inf.listHierarchyRootClasses(); i.hasNext();) {
      OntClass hierarchyRoot = i.next();
      if (hierarchyRoot.getURI() != null && hierarchyRoot.getURI().startsWith(ns)) {
        String hierarchy = hierarchyRoot.getURI().toString().split("#")[1];
        list2.add(hierarchy);
      }
    }
    String json2 = new Gson().toJson(list2);
    response.setContentType("application/json");
    response.setCharacterEncoding("UTF-8");
    response.getWriter().write(json2);
  }
```

Πίνακας 17- Κλάσεις οντολογίας μέσω JENA

IndividualServlet

Method Detail

doGet

```
public void doGet(javax.servlet.http.HttpServletRequest request,
          javax.servlet.http.HttpServletResponse response)
            throws javax.servlet.ServletException,
                   java.io.IOException
```

This method is used for the creation or deletion of an individual.

Overrides:

doGet in class javax.servlet.http.HttpServlet

Parameters:

request - This servlet is called from any jsp page

response - Saves the modified owl file to the disk and afterwards calls the storeSDB method

Throws:

javax.servlet.ServletException

java.io.IOException

storeSDB

```
public void storeSDB(java.lang.String dbName,
              java.lang.String owlfile)
                throws java.lang.ClassNotFoundException,
                       java.lang.InstantiationException,
                       java.lang.IllegalAccessException,
                       org.xmldb.api.base.XMLDBException
```

This method stores the modified owl ontology to RDBMS Triple store.

Parameters:

dbName - The name of database

owlfile - The modified owl file

Throws:

java.lang.ClassNotFoundException

java.lang.InstantiationException

java.lang.IllegalAccessException

org.xmldb.api.base.XMLDBException

LivingRoomServlet

Method Detail

doGet

```
protected void doGet(javax.servlet.http.HttpServletRequest request,
          javax.servlet.http.HttpServletResponse response)
               throws javax.servlet.ServletException,
                      java.io.IOException
```

Create a questionnaire for living room interior space based on specific characteristics of the ontology.

Overrides:

doGet in class javax.servlet.http.HttpServlet

Parameters:

request - This servlet is called from WizardUserServlet

response - The output result is presented to the user in HTML form

Throws:

javax.servlet.ServletException

java.io.IOException

ModifyServlet

Method Detail

doGet

```
protected void doGet(javax.servlet.http.HttpServletRequest request,
          javax.servlet.http.HttpServletResponse response)
               throws javax.servlet.ServletException,
                      java.io.IOException,
                      java.io.FileNotFoundException
```

This servlet is responsible for the connection to the chosen database

Overrides:

doGet in class javax.servlet.http.HttpServlet

Parameters:

request - This servlet is called from modifyWizard jsp page

response - Returns the database to be modified

Throws:

javax.servlet.ServletException

java.io.IOException

java.io.FileNotFoundException

PropertiesServlet

Method Detail

doGet

```
public void doGet(javax.servlet.http.HttpServletRequest request,
          javax.servlet.http.HttpServletResponse response)
            throws javax.servlet.ServletException,
                   java.io.IOException
```

This servlet creates an xml which contains the properties of the chosen individual

Overrides:

doGet in class javax.servlet.http.HttpServlet

Parameters:

request - This servlet is called from adminWizard and modifyWizad jsp pages

response - Writes the generated in memory XML to the disk

Throws:

javax.servlet.ServletException

java.io.IOException

SaveWizard

Method Detail

doGet

```
protected void doGet(javax.servlet.http.HttpServletRequest request,
          javax.servlet.http.HttpServletResponse response)
               throws javax.servlet.ServletException,
                      java.io.IOException
```

This servlet creates an owl file to the disk and a corresponding database to the RDBMS Triple store, based on the name given by the user

Overrides:

doGet in class javax.servlet.http.HttpServlet

Parameters:

request - This servlet is called from adminWizard jsp page

response - This servlet calls the storeSDB method which is contained in ServletMethods

Throws:

javax.servlet.ServletException

java.io.IOException

SelectedChoicesServlet

Method Detail

doGet

```
protected void doGet(javax.servlet.http.HttpServletRequest request,
          javax.servlet.http.HttpServletResponse response)
               throws javax.servlet.ServletException,
                      java.io.IOException
```

This servlet displays the rooms contained in the RDBMS which satisfy the choices made by the user

Overrides:

`doGet` in class `javax.servlet.http.HttpServlet`

Parameters:

`request` - This servlet can be called from either BedroomServlet or LivingRoomServlet

`response` - Results come in the form of an HTML table with each row representing a valid room

Throws:

`javax.servlet.ServletException`

`java.io.IOException`

sparqlQueries

```
public java.lang.String sparqlQueries(java.lang.String db,
                                      javax.servlet.http.HttpServletRequest request)
```

This method creates and executes a sparql query depending on the room type

Parameters:

`db` - The database name retrieved from showdatabases method

`request` - Contains the choices of the user from the questionnaire

Returns:

showdatabases

```
public java.util.List<java.lang.String> showdatabases(javax.servlet.http.HttpServletRequest request)
                                                throws java.io.IOException
```

This method displays the available databases from the RDBMS Triple store

Parameters:

`request` - This method is called from SelectedChoicesServlet

Returns:

Returns the list of databases

Throws:

`java.io.IOException`

listTables

```
public boolean listTables(java.lang.String db)
```

This method displays all the available tables of the chosen database and ccmpares them with a predefined set of tables

Parameters:

db - The name of the database

Returns:

True in case of similar tables and false otherwise

readXMLFile

```
public java.lang.String readXMLFile(java.lang.String db,
                            com.hp.hpl.jena.query.ResultSet rs)
                              throws java.io.FileNotFoundException,
                                     javax.xml.parsers.ParserConfigurationException,
                                     java.io.IOException,
                                     org.xml.sax.SAXException
```

This method outputs the ResultSet of a sparql query in xml format and retrieves the uri node contents

Parameters:

db - The name of the xml

rs - The ResultSet of sparqlQueries method

Returns:

Throws:

java.io.FileNotFoundException

javax.xml.parsers.ParserConfigurationException

java.io.IOException

org.xml.sax.SAXException

ServletMethods

Method Detail

createDatabase

```
public void createDatabase(java.lang.String owlName)
                    throws java.lang.ClassNotFoundException,
                           java.sql.SQLException
```

This method creates a database in the RDBMS Triple store

Parameters:

`owlName` - The name of the database to be created

Throws:

`java.lang.ClassNotFoundException`

`java.sql.SQLException`

```
Connection con = null;
        Class.forName("com.mysql.jdbc.Driver");
        con = (Connection) DriverManager.getConnection("jdbc:mysql://localhost:3306/",
"root", "");
        Statement st = con.createStatement();
        st.executeUpdate("CREATE DATABASE " + owlName + " DEFAULT CHARACTER SET utf8
DEFAULT COLLATE utf8_general_ci");
        System.out.println("Database has been created!!!");
```

Πίνακας 18- Δημιουργία βάσης MySQL

connectSDB

```
public com.hp.hpl.jena.sdb.Store connectSDB(java.lang.String owlName)
```

This method is used for the connection to any database

Parameters:

`owlName` - The database name to be connected

Returns:

storeSDB

```
public boolean storeSDB(java.lang.String owlName,
                java.lang.String owlfile)
                  throws java.lang.ClassNotFoundException,
                         java.lang.InstantiationException,
                         java.lang.IllegalAccessException,
                         org.xmldb.api.base.XMLDBException,
                         java.io.FileNotFoundException
```

This method stores a database to the RDBMS Triple store

Parameters:

`owlName` - The database name to be stored

`owlfile` - The owl file which is used for the store

Returns:

True or false depending on the successful or not storing of the database

Throws:

`java.lang.ClassNotFoundException`

`java.lang.InstantiationException`

`java.lang.IllegalAccessException`

`org.xmldb.api.base.XMLDBException`

`java.io.FileNotFoundException`

modifySDB

```
public void modifySDB(java.lang.String owlName,
                java.lang.String owlfile)
                  throws java.lang.ClassNotFoundException,
                         java.lang.InstantiationException,
                         java.lang.IllegalAccessException,
                         org.xmldb.api.base.XMLDBException,
                         java.io.FileNotFoundException
```

This method modifies a database of the RDBMS Triple store

Parameters:

`owlName` - The database name to be modified

`owlfile` - The owl file which is used for the modification

Throws:

`java.lang.ClassNotFoundException`

`java.lang.InstantiationException`

`java.lang.IllegalAccessException`

`org.xmldb.api.base.XMLDBException`

`java.io.FileNotFoundException`

readOwlFile

```
public com.hp.hpl.jena.ontology.OntModel readOwlFile(java.lang.String owlName,
                                                     com.hp.hpl.jena.ontology.OntModelSpec spec)
```

This method is used to read owl file from the disk

Parameters:

`owlName` - The owl file name

`spec` - The description of the components of the ontology model

Returns:

```
public OntModel readOwlFile(String owlName, OntModelSpec spec) {
   OntModel inf = ModelFactory.createOntologyModel(spec);
   InputStream in = FileManager.get().open(owlName);
   if (in == null) {
      throw new IllegalArgumentException("File: " + owlName + " not found");
   }
   inf.read(in, "");
   return inf;
}
```

Πίνακας 19-Διάβασμα αρχείου owl

prefixNS

```
public java.lang.String prefixNS(com.hp.hpl.jena.ontology.OntModel inf)
```

This method returns the namespace of the ontology

Parameters:

`inf` - The ontology model of the owl file

Returns:

The namespace of the ontology model

showdatabases

```
public java.util.List<java.lang.String> showdatabases()
                                                throws java.io.IOException
```

This method displays the databases of the RDBMS Triple store

Returns:

Returns an ArrayList which contains all the databases

Throws:

`java.io.IOException`

connectModelFromDB

```
public com.hp.hpl.jena.rdf.model.Model connectModelFromDB(java.lang.String RDBMSModel)
```

This method creates a connection to a database

Parameters:

`RDBMSModel` - The database name

Returns:

subclassOfRoom

```
public java.util.List<java.lang.String> subclassOfRoom()
```

This method returns the subclasses of the Room class of the ontology

Returns:

Returns a list which contains the subclasses

datatypePropertyValue

```
public java.util.List<java.lang.String> datatypePropertyValue(java.lang.String propName)
```

This method displays the available values of the datatype properties of the ontology

Parameters:

`propName` - The datatype property name

Returns:

Returns an ArrayList which contains all values

objectPropertyValue

```
public java.util.List<java.lang.String> objectPropertyValue(java.lang.String propName)
```

This method displays the available values of the object properties of the ontology

Parameters:

`propName` - The object property name

Returns:

Returns an ArrayList which contains all values

individualOfClass

```
public java.util.List<java.lang.String> individualOfClass(java.lang.String className)
```

This method returns the individuals of the chosen class

Parameters:

`className` - The class name to be searched for instances

Returns:

Returns a list of the individuals

sameContent

```
public void sameContent(javax.servlet.http.HttpServletResponse response)
                 throws java.io.IOException
```

This method contains questions that can be found in both Bedroom and LivingRoom servlets.

Parameters:

`response` - The provided questions come in HTML form

Throws:

`java.io.IOException`

StatementServlet

Method Detail

doGet

```
protected void doGet(javax.servlet.http.HttpServletRequest request,
          javax.servlet.http.HttpServletResponse response)
               throws javax.servlet.ServletException,
                      java.io.IOException
```

This servlet creates statements which are added in the owl file and in the database

Overrides:

`doGet` in class `javax.servlet.http.HttpServlet`

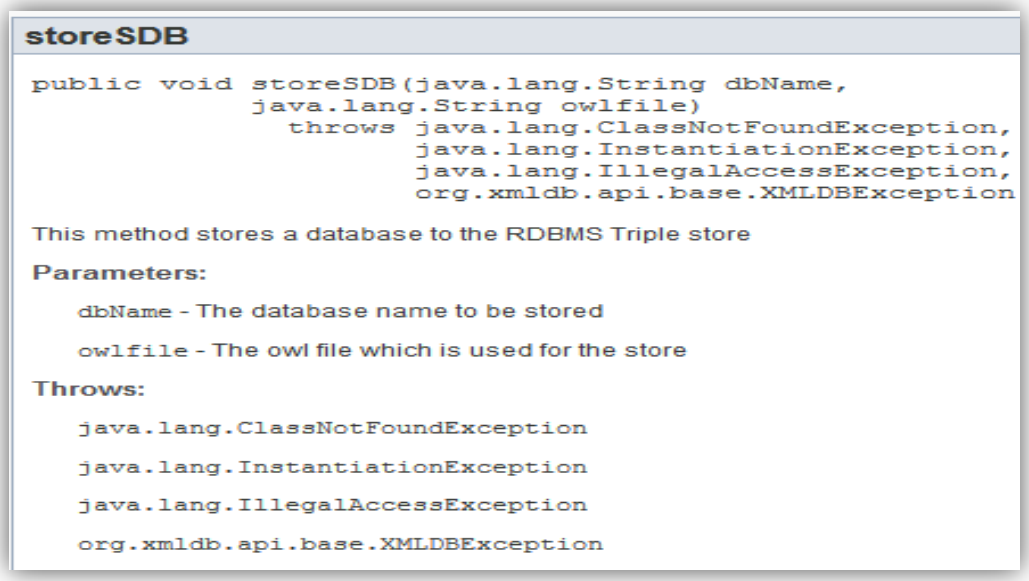
Parameters:

`request` - This servlet is called from adminWizard and modifyWizard jsp pages

Throws:

`javax.servlet.ServletException`

`java.io.IOException`



storeSDB

```
public void storeSDB(java.lang.String dbName,
                     java.lang.String owlfile)
              throws java.lang.ClassNotFoundException,
                     java.lang.InstantiationException,
                     java.lang.IllegalAccessException,
                     org.xmldb.api.base.XMLDBException
```

This method stores a database to the RDBMS Triple store

Parameters:

`dbName` - The database name to be stored

`owlfile` - The owl file which is used for the store

Throws:

`java.lang.ClassNotFoundException`

`java.lang.InstantiationException`

`java.lang.IllegalAccessException`

`org.xmldb.api.base.XMLDBException`

SubclassServlet

Method Detail

doGet

```
public void doGet(javax.servlet.http.HttpServletRequest request,
                  javax.servlet.http.HttpServletResponse response)
           throws javax.servlet.ServletException,
                  java.io.IOException
```

This servlet displays direct subclasses or individuals of the chosen class

Overrides:

`doGet` in class `javax.servlet.http.HttpServlet`

Parameters:

`request` - This method is called from adminWizard and modifyWizard jsp pages

`response` - Returns the subclass or individuals in JSON format

Throws:

`javax.servlet.ServletException`

`java.io.IOException`

```
String chosenMajorClassWithNs = ns + selectSubclass;
OntClass artefact = inf.getOntClass(chosenMajorClassWithNs);
if (artefact.listSubClasses().hasNext()) {
    List<String> list = new ArrayList<String>();
   for (Iterator<OntClass> i = artefact.listSubClasses(true); i.hasNext();) {
      OntClass c = i.next();
      list.add(c.getURI().toString().split("#")[1]);
   }
   list.add("sub");
   String json = new Gson().toJson(list);
```

```
        System.out.println(json);
         response.setContentType("application/json");
         response.setCharacterEncoding("UTF-8");
         response.getWriter().write(json);
      }
```

Πίνακας 20-Κώδικας σε JENA για αναγνώριση υποκλάσεων

```
String chosenClassWithNs = ns + selectSubclass;
Individual indOfClass = inf.getIndividual(chosenClassWithNs);
List<String> list2 = new ArrayList<String>();
for (ExtendedIterator<Individual> individuals = inf.listIndividuals(indOfClass);
individuals.hasNext();;) {
list2.add(individuals.next().getURI().toString().split("#")[1]);
}
list2.add("ind");
         String json = new Gson().toJson(list2);
         System.out.println(json);
         response.setContentType("application/json");
         response.setCharacterEncoding("UTF-8");
         response.getWriter().write(json);
      }
   }
```

Πίνακας 21- Κώδικας σε JENA για αναγνώριση individuals

WizardUserServlet

doGet

```
protected void doGet(javax.servlet.http.HttpServletRequest request,
          javax.servlet.http.HttpServletResponse response)
               throws javax.servlet.ServletException,
                      java.io.IOException
```

This servlet is used for redirection to the appropriate servlet depending on the user's choice

Overrides:

doGet in class javax.servlet.http.HttpServlet

Parameters:

request - The interior room chosen by the user

response - The servlet which is going to be presented next

Throws:

javax.servlet.ServletException

java.io.IOException