ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ & ΠΛΗΡΟΦΟΡΙΚΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Εφαρμογή εξόρυξης δεδομένων από τον παγκόσμιο ιστό και μοντελοποίησή του σε XSD/XML»

Βλαχογεωργακόπουλος Παναγιώτης
ΑΜ: 2221

ΥΠΕΥΘΥΝΟΣ ΚΑΘΗΓΗΤΗΣ:
Χρήστος Μπούρας

ΠΑΤΡΑ, ΦΕΒΡΟΥΑΡΙΟΣ 2012

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον Καθηγητή μου κύριο Μπούρα Χρήστο για τη συνεχή καθοδήγηση και υποστήριξη που μου παρείχε καθ' όλη τη διάρκεια της εκπόνησης της εργασίας μου, για το χρόνο που μου διέθεσε.

Π Ε Ρ Ί Λ Η Ψ Η

Η παρούσα πτυχιακή εργασία πραγματεύεται την ανάπτυξη δομών δεδομένων που σκοπό έχουν την αποθήκευση του περιεχομένου που βρίσκεται στο web, ως εργαλείων τεχνολογικής υποδομής του Σημασιολογικού Ιστού, με σκοπό την διευκόλυνση στην επεξεργασία και την κατηγοριοποίηση ειδησεογραφικού περιεχομένου ή οποιαδήποτε άλλης μορφής πληροφορίας που βρίσκεται στο web.

Στα πλαίσια της εργασίας αυτής χρησιμοποιήθηκαν αρκετά εργαλεία open source καθώς και έγινε ανάπτυξη ενός λογισμικού με σκοπό τη δημιουργία on the fly, xsd σχημάτων που να χαρακτηρίζουν τα δεδομένα που κάνουμε εξόρυξη από το web. Η εξαγωγή των πληροφοριών πραγματοποιείται με τη βοήθεια των wrappers (προγράμματα εξαγωγής πληροφορίας), πάνω στους οποίους δουλέψαμε προκειμένου να υλοποιηθεί η προσπάθεια της παρούσας πτυχιακής εργασίας.

Στην ενότητα αυτή γίνεται μία εισαγωγή στις έννοιες του Σημασιολογικού Ιστού, της εξατομίκευσης υπηρεσιών (υπηρεσίες στα πλαίσια της ανάπτυξης του Σημασιολογικού Ιστού) και των οντολογιών, μέσων απαραίτητων για την σημασιολογική αναπαράσταση της πληροφορίας.

Το φιλτράρισμα στα δεδομένα γίνεται με τους παρακάτω τρόπους: το απλό φιλτράρισμα, το συνεργατικό φιλτράρισμα και το φιλτράρισμα βάση περιεχομένου. Εμείς θα δοκιμάσουμε έναν τρόπο φιλτραρίσματος μέσα από τη

διπλωματική εργασία, που σκοπό έχει να κάνει φιλτράρισμα μέσα από τη δομή περιεχομένου της ιστοσελίδας.

ΠΕΡΙΕΧΟΜΕΝΑ

Περιεχόμενα

ΚΕΦΑΛΑΙΟ 1: ΕΙΣΑΓΩΓΗ

Μέσα σε μικρό χρονικό διάστημα το Διαδίκτυο κατάφερε να προσελκύσει ένα μεγάλο πλήθος χρηστών, κυρίως χάρη στην ευκολία πρόσβασης και παροχής υπηρεσιών στον Παγκόσμιο Ιστό. Επακόλουθο της τόσο μεγάλης προσέλευσης ήταν η παράλληλη αύξηση των ιστοσελίδων που δημοσιεύονται στον Ιστό προκαλώντας αναπόφευκτα την αύξηση της πληροφορίας σε τέτοιο βαθμό ώστε σήμερα να αντιμετωπίζεται το πρόβλημα της *υπερπληροφόρησης* (information overload) στον Παγκόσμιο Ιστό.

Στον Ιστό, λοιπόν, όπου η πληθώρα πληροφοριών είναι τόσο μεγάλη, οι χρήστες, ιδίως εκείνοι που δεν διαθέτουν τις κατάλληλες γνώσεις και δεν είναι επαρκώς ενημερωμένοι, δυσκολεύονται και πολύ συχνά αποτυγχάνουν να εντοπίσουν τις πληροφορίες που αναζητούν. Παράλληλα, οι απαιτήσεις των χρηστών για καλύτερες μεθόδους αναζήτησης και εύρεσης πληροφορίας διαρκώς μεγαλώνουν αναγκάζοντας τις εταιρείες και τις επιχειρήσεις στο Διαδίκτυο να δώσουν προστιθέμενη αξία στις εφαρμογές τους, αυξάνοντας μ' αυτό τον τρόπο άλλωστε και την καθιερωμένη πελατεία τους.

Επίσης, η σημερινή αναπαράσταση της πληροφορίας στο διαδίκτυο στο μεγαλύτερο μέρος της προορίζεται για κατανόηση και διαχείριση από τους ανθρώπους και όχι από τα υπολογιστικά συστήματα. Τα υπολογιστικά συστήματα δεν μπορούν να επεξεργαστούν έννοιες και σημασιολογίες και

κατά συνέπεια να ανταποκριθούν σε σύνθετες αναζητήσεις των χρηστών. Προς την βελτίωση της δόμησης της πληροφορίας, ώστε αυτή να είναι προσπελάσιμη από εφαρμογές υπολογιστών, όχι μόνο για λόγους παρουσίασης της πληροφορίας, αλλά με τελικό στόχο την αυτοματοποίηση, ολοκλήρωση (integration), και επαναχρησιμοποίηση των δεδομένων σε διάφορες εφαρμογές, κατευθύνεται ο Σημασιολογικός Ιστός.

Ο Σημασιολογικός Ιστός (Semantic Web) αποτελεί το επόμενο βήμα του Παγκόσμιου Ιστού. Ένα όραμα και μία πρόταση για την μετεξέλιξη του Διαδικτύου, όπου η πληροφορία αποκτά πλέον δομή και σημασιολογία και ευρύτερος στόχος τίθεται η αποδοτική αναζήτηση και επεξεργασία δεδομένων. Εμπνευστής του όρου Semantic Web και της υλοποίησης της αρχιτεκτονικής του είναι ο Tim Berners-Lee, σύμφωνα με τον οποίο «ο Σημασιολογικός Ιστός δεν είναι ένας ξεχωριστός ιστός αλλά η επέκταση του συντακτικού ιστού, στον οποίο η πληροφορία είναι καλά καθορισμένη καθιστώντας καλύτερη τη συνεργασία ανθρώπων και υπολογιστών» [Berners-Lee, Lassila, Hendler].

Το όραμα του Σημασιολογικού Ιστού είναι να πραγματοποιείται η επεξεργασία δεδομένων από τα υπολογιστικά συστήματα με τέτοιο τρόπο ώστε να ικανοποιούνται όσο το δυνατόν ακόμα πιο σύνθετες απαιτήσεις των χρηστών.

1.1.2 Εξατομίκευση

Στα πλαίσια της ανάπτυξης ενός Σημασιολογικού Ιστού, όπου το περιεχόμενο, το οποίο διακινείται στο διαδίκτυο, μπορεί να τύχει επεξεργασίας και να κατανοηθεί τόσο από τους ανθρώπους όσο και από τις μηχανές, εντάσσεται και η εμφάνιση των «*τεχνικών εξατομικευμένων υπηρεσιών*». Σε έναν ιστό όπου η πληθώρα πληροφοριών δυσκολεύει την αναζήτηση και την ανάκτηση, η εξατομίκευση (personalization) καλείται να αποτελέσει μία έξυπνη πρόταση, όπου το *περιβάλλον του Παγκόσμιου Ιστού* (*Web environment*) προσαρμόζεται στις προτιμήσεις του χρήστη.

Εξατομίκευση είναι η διαδικασία κατά την οποία συλλέγονται και αποθηκεύονται πληροφορίες αναφορικά με τους χρήστες ενός συστήματος, η ανάλυση αυτής της πληροφορίας και η ανάκτηση των κατάλληλων δεδομένων και πληροφοριών για τον συγκεκριμένο χρήστη την κατάλληλη χρονική στιγμή.

Σήμερα, η δομή του Παγκόσμιου Ιστού αναγκάζει σε μία πλοήγηση από ιστοσελίδα σε ιστοσελίδα προκειμένου να εντοπιστούν οι αναζητούμενες πληροφορίες και ειδήσεις. Μία εξατομικευμένη ιστοσελίδα *φέρνει* τα νέα και την πληροφορία απευθείας στον χρήστη προσωπικά. Δεν είναι καθόλου περίεργο, λοιπόν, που πολλές επιχειρήσεις στο Διαδίκτυο άρχιζαν να εφαρμόζουν στις ιστοσελίδες τους τεχνικές εξατομικευμένων υπηρεσιών. Ικανοποιώντας τις ανάγκες των πελατών εξασφαλίζεται σε μεγάλο βαθμό η επαναχρησιμοποίηση των υπηρεσιών, ο πελάτης παραμένει περισσότερο στην ιστοσελίδα και κατά συνέπεια αυξάνεται η εμπορική αξία της ιστοσελίδας.

Πλέον οι χρήστες έχουν συνηθίσει την παρουσία συστημάτων εξατομίκευσης και πολύ συχνά μάλιστα τα θεωρούν δεδομένα. Σύμφωνα με την Global Landscape, το 56% των χρηστών που διαβάζουν ηλεκτρονικές εφημερίδες, σήμερα προτιμούν εξατομικευμένες εφημερίδες. (Αλεξόπουλος, 2003)

1.1.3 Το ζήτημα των οντολογιών στο Σημασιολογικό Ιστό

Απαραίτητο μέσο για πλήθος εφαρμογών στον Σημασιολογικό ιστό, οι οντολογίες προορίζονται να προσφέρουν μία αναπαράσταση γνώσης και ένα λεξιλόγιο από κλάσεις και σχέσεις (κάτι αντίστοιχο με τους βιβλιοθηκονομικούς Θησαυρούς). Οι οντολογίες καταφέρνουν να συνενώσουν δύο ουσιώδη συστατικά, τα οποία συμβάλλουν στην ανάπτυξη του Παγκόσμιου Ιστού. Από τη μία ορίζουν την τυπική σημασιολογία της πληροφορίας διευκολύνοντας την επεξεργασία της από τον υπολογιστή, ενώ ταυτόχρονα από την άλλη ορίζουν σημασιολογία του πραγματικού κόσμου. Μ' αυτό τον τρόπο επιτυγχάνεται η σύνδεση του περιεχομένου, το οποίο τυχαίνει

μηχανικής επεξεργασίας, με τη σημασία που του δίνουν οι άνθρωποι βασιζόμενοι σε κοινά αποδεκτή ορολογία.

Οι οντολογίες μπορούν να διαδραματίσουν έναν πολύ σημαντικό ρόλο στις υπηρεσίες εξατομικευμένης ενημέρωσης, καθώς επιτρέπουν την περιγραφή, επεξεργασία και κατηγοριοποίηση του περιεχομένου τους, μ' έναν τρόπο κοινά κατανοητό τόσο για τους χρήστες όσο και για τα συστήματα. Βασικό ενδιαφέρον των υπηρεσιών αυτών, άλλωστε, είναι η όσο το δυνατόν καλύτερη εξυπηρέτηση των πελατών τους. Η ανάπτυξη, λοιπόν, ενός σημασιολογικά δομημένου συστήματος διευκολύνει την αναζήτηση και τελικά την πρόσβαση στις *αποθήκες* των ειδησεογραφικών τεκμηρίων, που αποτελούν μία πλούσια πηγή διάθεσης πληροφοριών.

Η παρούσα εργασία εστιάζει την μελέτη της κυρίως στην ανάπτυξη οντολογιών για το ειδησεογραφικό πεδίο (News Domain).

ΚΕΦΑΛΑΙΟ 2: ΕΞΑΓΟΝΤΑΣ ΠΛΗΡΟΦΟΡΙΕΣ ΑΠΟ ΤΟ WEB

Στην ενότητα αυτή παρουσιάζονται οι συνηθισμένοι τρόποι εξατομίκευσης, σύμφωνα με τους οποίους στη συνέχεια θα γίνει η παρουσίαση και αξιολόγηση ορισμένων υπηρεσιών εξατομικευμένης ενημέρωσης.

Μία βασική διάκριση στα συστήματα εξατομικευμένης ενημέρωσης γίνεται με βάση τον τρόπο που συλλέγουν πληροφορίες για τον χρήστη. Το σύστημα μπορεί να δέχεται πληροφορίες για τα ενδιαφέροντα του χρήστη άμεσα, συνήθως μέσω κάποιας φόρμας που καλείται να συμπληρώσει ο χρήστης κατά την εγγραφή του στο σύστημα. Επίσης, η συλλογή των πληροφοριών αυτών μπορεί να πραγματοποιείται έμμεσα παρακολουθώντας τις επιλογές του χρήστη. Μία ακόμη πηγή πληροφοριών είναι τα κληροδοτημένα δεδομένα (legacy data). Τέλος, πολύ συχνά τα συστήματα εφαρμόζουν συνδυασμό αυτών των μεθόδων.

Σύμφωνα με τα στοιχεία που συλλέγουν για το χρήστη, τα συστήματα μοντελοποιούν το χρήστη (user modeling), χτίζοντας το προφίλ του. Εκτός

από τον χειρονακτικό τρόπο με τον οποίο κατασκευάζονται συνήθως τα μοντέλα των χρηστών, υπάρχουν, αυτόματοι τρόποι μοντελοποίησης που βασίζονται σε τεχνικές μηχανικής μάθησης.

Ακόμη, τα συστήματα εξατομικευμένης ενημέρωσης διακρίνονται σε προσαρμοστικά και μη. Μόλις το σύστημα συλλέξει τις απαραίτητες για την λειτουργία του πληροφορίες μπορεί είτε να προσαρμόζεται στις αλλαγές των προτιμήσεων του χρήστη είτε να παραμένει στατικό.

Τέλος, τα συστήματα κατηγοριοποιούνται σύμφωνα με τον τρόπο που φιλτράρουν και αναλύουν τα μεταδεδομένα του χρήστη ώστε να επιτευχθεί η εξατομίκευση. Διακρίνουμε, λοιπόν, το απλό φιλτράρισμα, το συνεργατικό φιλτράρισμα και το φιλτράρισμα βάση περιεχομένου. Ορισμένα συστήματα κάνουν συνδυασμό των παραπάνω.

ΚΕΦΑΛΑΙΟ 3: XML/XSD Schema ΔΟΜΗ ΚΑΙ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ με τη χρήση του εργαλείου Astroboa

Σύστημα Διαχείρισης Επιχειρησιακού Περιεχομένου Astroboa

Το Σύστημα Διαχείρισης Επιχειρησιακού Περιεχομένου Astroboa επιτρέπει την κεντρική και ενιαία διαχείριση του συνόλου του περιεχομένου που παράγεται, συλλέγεται, επεξεργάζεται, διακινείται και δημοσιεύεται εσωτερικά σε έναν οργανισμό τοπικής αυτοδιοίκησης (Intranet) ή και εξωτερικά από και πρός τους συνεργάτες και τους πολίτες (Extranet / Portal).

Επίσης παρέχει ένα ιδιαίτερα ευρύ και πλούσιο σε εργαλεία προγραμματιστικό περιβάλλον για την ανάπτυξη των υποδομών περιεχομένου των ΟΤΑ.

Από την οπτική γωνία του τελικού χρήστη (στέλεχος ΟΤΑ) το Astroboa αποτελεί μια **Διαδικτυακή Εφαρμογή** για την συνεργατική και ασφαλή διαχείριση και κατηγοριοποίηση πληροφοριών και εγγράφων

Από την οπτική γωνία του οργανισμού συνολικά (ΚΕΔΚΕ, ΟΤΑ) το Astroboa παρέχει μια **Ενιαία Υποδομή** για την:

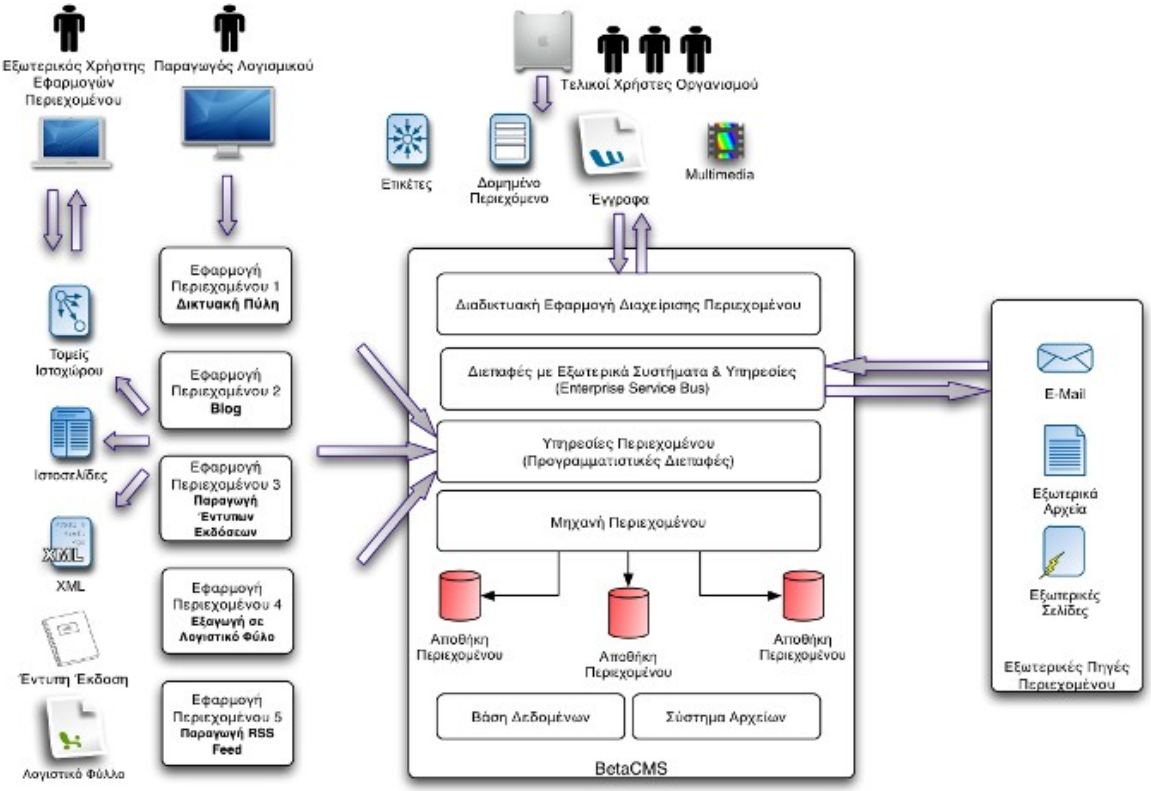
- Μοντελοποίηση

- Προτυποποιημένη Διαχείριση
- Δημοσίευση σε πολλαπλά μέσα και αποδέκτες

των πληροφοριών που συλλέγει και παράγει.

Τέλος, από την οπτική γωνία των παραγωγών λογισμικού (π.χ. των αναδόχων έργων για τους ΟΤΑ) το Astroboa παρέχει ένα **ευρύ φάσμα "Υπηρεσιών Ιστού (web services)"**, και ένα πλούσιο σε εργαλεία προγραμματιστικό περιβάλλον για την ταχεία ανάπτυξη εφαρμογών περιεχομένου (document sharing, blogs, forums, wikis, portals, asset sharing), με έμφαση στην δημιουργία υποδομών κοινωνικών δικτύων (Social Networks).

Το σχήμα που ακολουθεί παρουσιάζει τις λειτουργίες και χρήσεις του Astroboa εσωτερικά και εξωτερικά του οργανισμού. Αποτυπώνεται η χρήση των διαφορετικών τμημάτων που απαρτίζουν το Astroboa από τους διάφορους τύπους εσωτερικών και εξωτερικών χρηστών καθώς και η διασύνδεση με εσωτερικά και εξωτερικά υφιστάμενα συστήματά από τα οποία "τραβάει" και στα οποία στέλνει περιεχόμενο μέσω της υποδομής του Enterprise Service Bus module που περιλαμβάνει.



Τα βασικά χαρακτηριστικά του συστήματος Astroboa μπορούν να συνοψιστούν στα παρακάτω:

Το Astroboa βασίζεται και εφαρμόζει Διεθνή Ανοικτά Πρότυπα για την μοντελοποίηση της πληροφορίας και για την δημιουργία της «Αποθήκης Περιεχομένου» (Content Repository) επιτρέποντας την διαλειτουργικότητα με άλλα συστήματα που υιοθετούν τα πρότυπα.

Στα πλαίσια των ελληνικών κανονισμών και προτύπων, το Astroboa ακολουθεί τις προδιαγραφές που ορίζει το Ελληνικό Πλαίσιο Διαλειτουργικότητας (e-GIF) για τα συστήματα Ηλεκτρονικής Διακυβέρνησης και ειδικότερα για την διαχείριση περιεχομένου.

Στο επίπεδο της μοντελοποίησης των αντικειμένων/τύπων περιεχομένου, το Astroboa υιοθετεί τη χρήση XSD Schema επιτρέποντας την άμεση χρήση διαθέσιμων μοντέλων πληροφορίας υπό την μορφή σχημάτων XSD και υποστηρίζει την εύκολη εισαγωγή στο σύστημα υφιστάμενων XML Documents. Ταυτόχρονα η υιοθέτηση της γραμματικής του XSD Schema, το οποίο αποτελεί ένα από τα πλέον διαδεδομένα και συνιστώμενα πρότυπα μοντελοποίησης πληροφοριών, εξασφαλίζει, για οποιοδήποτε νέο περιεχόμενο μοντελοποιείται και αποθηκεύεται, τη συμβατότητά του με τα διεθνή πρότυπα, τον εύκολο μετασχηματισμό του και την διακίνησή του μεταξύ συνεργαζόμενων συστημάτων.

Για την περιγραφή των διαχειριστικών μετα-δεδομένων υιοθετείται το πρότυπο Dublin Core Metadata.

Το Astroboa είναι γραμμένο στη γλώσσα Java υιοθετώντας τα διεθνή πρότυπα που ορίζει η γλώσσα.

Στο επίπεδο της αποθήκευσης του Περιεχομένου, το Astroboa υιοθετεί το πρότυπο JSR-170 / Java Content Repository που επιτρέπει την αποθήκευση σε μια πρότυπη μορφή ανεξάρτητη υλοποίησης (Vendor Independent) και εξασφαλίζει την πρόσβαση στο περιεχόμενο από οποιαδήποτε εφαρμογή μέσω μιας ενιαίας προγραμματιστικής διεπαφής (API).

Στο επίπεδο των Web Services υιοθετεί τα πρότυπα JAX-WS, JAX-RS και JAXB.

Το Astroboa εγκαθίσταται ως μέρος ενός JEE5 Application Server ακολουθώντας τα πρότυπα που ορίζονται κάτω από το JEE5 όπως JMS για message queues, JTA για transaction management, JCA για σύνδεση με data sources, JAAS για την ασφάλεια των εφαρμογών, Servlets για την δημιουργία Web Εφαρμογών, JSF για την δημιουργία της διεπαφής χρήστη σε Web εφαρμογές.

Επίσης χρησιμοποιούνται οι πλέον διαδεδομένες τεχνολογίες και βιβλιοθήκες ανοικτού λογισμικού για την σύνθεση των τμημάτων του Astroboa όπως το Spring Application Framework που προσφέρει δυνατότητες Dependency Injection και Aspect Oriented Programming, το JBoss Seam Framework που αποτελεί ένα ολοκληρωμένο πλαίσιο δημιουργίας Web εφαρμογών, οι βιβλιοθήκες Hibernate και iBatis για Object to Relational Mapping, η μηχανή διαχείρισης επιχειρησιακών κανόνων (Business Rules Management System) Drools (χρησιμοποιείται στη διαχείριση κανόνων ασφάλειας), η μηχανή διαχείρισης ροών εργασίας JBPM, η πλατφόρμα Mule που παρέχει ένα Enterprise Service Bus για τη διασύνδεση του Astroboa με τρίτα συστήματα, και πολλές μικρότερες βιβλιοθήκες ανοικτού λογισμικού.

Εν γένει το Astroboa βασίζεται αποκλειστικά σε ανοικτό λογισμικό και το σύνολο του πηγαίου κώδικα όλων των τμημάτων του αποτελούν ανοικτό λογισμικό το οποίο διατίθεται με την άδεια LGPL.

ΒΑΣΙΚΕΣ ΔΟΜΕΣ ΜΟΝΤΕΛΟΠΟΙΗΣΗΣ

Στην λίστα που ακολουθεί παρουσιάζονται οι βασικές δομές μοντελοποίησης περιεχομένου που παρέχει το Astroboa.

- **Αντικείμενο περιεχομένου** με σύνθετες και απλές ιδιότητες (**Content Object**)
- **Σύνθετη Ιδιότητα Αντικειμένου** / Δυναμική Συνιστώσα Ιδιοτήτων (**Complex Content Object Property** / Dynamic Content Object Aspect)
- **Ψηφιακό Κανάλι** (Ιδιότητα Αντικειμένου με ψηφιακό περιεχόμενο, **BinaryChannel**)
- **Ταξινομία** / Ιεραρχικό Δέντρο Δεσμευμένων Όρων (**Taxonomy**)

- **Λίστα Ετικετών Χρήστη** (**Folksonomy**)
- **Ταξινομικός Κόμβος** / Όρος Λεξικού / Δεσμευμένος Όρος / Ετικέτα Χρήστη (**Taxonomy Topic** / Vocabulary Item / User Tag)
- **Εικονικός Χώρος Χρήστη** (User Space, Organization Space ή απλά **Space**)
- **Χρήστης Αποθήκης Περιεχομένου** / Παραγωγός Περιεχομένου (**RepositoryUser**)

Οι δομές απευθύνονται σε τέσσερις βασικούς ρόλους που εμπλέκονται στην δημιουργία του μοντέλου της αποθήκης περιεχομένου καθώς και στην παραγωγή του περιεχομένου:

- Για τον ρόλο που πραγματοποιεί την μοντελοποίηση περιεχομένου σε επίπεδο σχεδίασης (content modeler) οι δομές αποτελούν τις κεντρικές έννοιες με τις οποίες θα περιγράψει σε αφαιρετικό επίπεδο την δομή της αποθήκης περιεχομένου του οργανισμού. Ο προσδιορισμός των Αντικειμένων Περιεχομένου και των ιδιοτήτων τους, η περιγραφή των σύνθετων ιδιοτήτων καθώς και των απαραίτητων ιεραρχικών δέντρων όρων για την κατηγοριοποίηση της πληροφορίας αποτελούν βασικές εργασίες αυτού του ρόλου.
- Για τον ρόλο που υλοποιεί το μοντέλο περιεχομένου οι κεντρικές δομές αντιστοιχούν στα βασικά XSD Schemas που παρέχει. Ο ρόλος υλοποιεί νέα XSD Schemas προεκτείνοντας τα βασικά σχήματα του Astroboa. Όταν τα σχήματα εισαχθούν στο σύστημα παράγονται αυτόματα οι φόρμες εισαγωγής για την διαδικτυακή εφαρμογή διαχείρισης περιεχομένου και επίσης οι μοντελοποιημένες δομές γίνονται αυτόματα διαθέσιμες στις προγραμματιστικές διεπαφές και τις υπηρεσίες ιστού του Astroboa.
- Για τον παραγωγό λογισμικού οι δομές αντιστοιχούν στις προγραμματιστικές οντότητες (object classes, object attributes) που παρέχονται μέσω των προγραμματιστικών διεπαφών του Astroboa. Ο παραγωγός λογισμικού χρησιμοποιεί τις διαθέσιμες προγραμματιστικές οντότητες και τις μεθόδους που περιέχουν για να χτίσει εφαρμογές που γράφουν και διαβάζουν περιεχόμενο.
- Για τον τελικό χρήστη που εισάγει και αναζητεί περιεχόμενο μέσω της διαδικτυακής εφαρμογής που παρέχει το Astroboa, οι παραπάνω δομές

απεικονίζονται στις διαθέσιμες λειτουργίες που παρέχει η εφαρμογή για να κάνει εφικτή την διαχείρισή τους. Ο χρήστης μπορεί να επιλέξει ένα τύπο αντικειμένου περιεχομένου από αυτούς που έχουν μοντελοποιηθεί και να δημιουργήσει νέα στιγμιότυπα με την βοήθεια των φορμών εισαγωγής που όπως αναφέρθηκε παραπάνω παράγονται αυτόματα. Επίσης μπορεί να δημιουργήσει νέες ταξινομίες και να προσθέσει ή να αλλάξει τις ετικέτες, να προσθέσει χρήστες, να αναζητήσει περιεχόμενο με βάση τις μοντελοποιημένες ιδιότητές του κλπ.

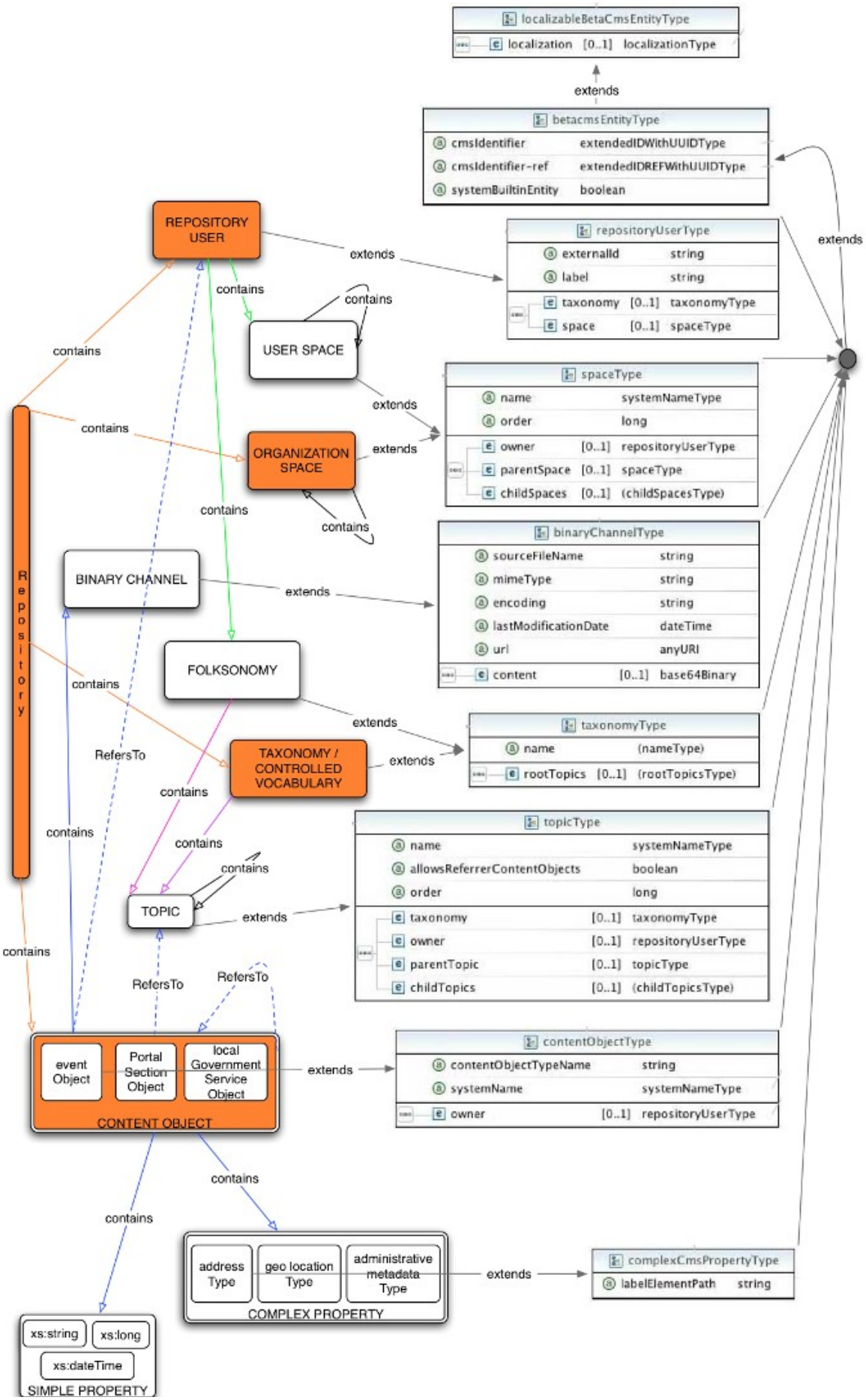
Οι πολλαπλοί επεξηγηματικοί όροι που εμφανίζονται για κάθε μία από τις παραπάνω δομές εκφράζουν το γεγονός ότι η ίδια βασική δομή μπορεί να έχει πολλαπλές χρήσεις ή και διαφορετικό νόημα κατά την διάρκεια της μοντελοποίησης και ανάλογα με το γνωστικό υπόβαθρο του προσώπου που πραγματοποιεί την μοντελοποίηση. Κάποιοι όροι είναι πιο συναφείς και πιο προσιτοί εννοιολογικά ανάλογα με την χρήση που γίνεται στις δομές και ανάλογα με το υπόβαθρο του προσώπου που τις χρησιμοποιεί (π.χ. για ένα απλό χρήστη ο όρος ετικέτα είναι πιο προσιτός ενώ για κάποιον ειδικό στην κατηγοριοποίηση η έννοια "Λεξικό Δεσμευμένων Όρων" είναι εξίσου κατανοητή. Και στις δύο περιπτώσεις στο Astroboa η υλοποίηση γίνεται μέσω Taxonomies και Topics).

Το σχήμα που ακολουθεί παρουσιάζει τον τρόπο που οι παραπάνω δομικές οντότητες συνδυάζονται για την δημιουργία μιας "Αποθήκης Περιεχομένου" (Content Repository). Εμφανίζονται τρεις βασικές σχέσεις μεταξύ των δομικών οντοτήτων:

- Η σχέση **"Περιέχει" (contains)** που δηλώνει ότι η μία δομή περιέχει την άλλη. Π.χ. Η αποθήκη περιεχομένου περιέχει τα αντικείμενα περιεχομένου που με την σειρά τους περιέχουν απλές και σύνθετες ιδιότητες.
- Η σχέση **"Προεκτείνει" (extends)** που δηλώνει πως η οντότητα παράγεται κληρονομώντας και προεκτείνοντας τα χαρακτηριστικά κάποιας άλλης συνήθως πρότυπης / αφαιρετικής οντότητας (σε αναλογία με τις οντοκεντρικές γλώσσες προγραμματισμού). Π.χ. ένα Αντικείμενο Περιεχομένου (άρθρο, υπηρεσία δήμου, κλπ.) κληρονομεί και προεκτείνει τις ιδιότητες του βασικού τύπου αντικειμένου που παρέχει το Astroboa (contentObjectType).

- Η σχέση **"Αναφορά σε άλλη Οντότητα" (RefersTo)** υποδηλώνει ότι μια οντότητα της αποθήκης περιέχει μια "**αναφορά**" (ένα σύνδεσμο) σε κάποια άλλη οντότητα. Η διαφορά σε σύγκριση με τη σχέση "Περιέχει" είναι οτι η συνδεδεμένη οντότητα δεν περικλείεται εσωτερικά κάποιας άλλης αλλά είναι ανεξάρτητη και μπορεί να υφίσταται μόνο ως σύνδεσμος μέσα σε άλλες οντότητες (κατά αναλογία με τα foreign keys στις βάσεις δεδομένων). Π.χ. στο σχήμα φαίνεται ότι ένα αντικείμενο περιεχομένου μπορεί να περιέχει αναφορές σε άλλα αντικείμενα περιεχομένου ή σε ετικέτες (δεσμευμένους όρους από ένα λεξικό) ή σε χρήστες ενώ περιέχει εσωτερικά τα κανάλια ψηφιακού περιεχομένου (BinaryChannel).

Στο αριστερό μέρος της εικόνας εμφανίζονται οι δομές με τις οποίες έρχεται σε επαφή ο υπεύθυνος μοντελοποίησης ή ο παραγωγός λογισμικού. Στο δεξί τμήμα φαίνονται οι συνδέσεις των βασικών δομών με βασικά XSD Schemas από τα οποία παράγονται.


localizableBetaCmsEntityType
localization [0..1] localizationType
extends
betacmsEntityType
cmsIdentifier extendedIDWithUUIDType
cmsIdentifier-ref extendedIDREFWithUUIDType
systemBuiltinEntity boolean
REPOSITORY USER
extends
repositoryUserType
externalId string
label string
taxonomy [0..1] taxonomyType
space [0..1] spaceType
contains
USER SPACE
spaceType
name systemNameType
order long
owner [0..1] repositoryUserType
parentSpace [0..1] spaceType
childSpaces [0..1] (childSpacesType)
ORGANIZATION SPACE
Repository
BINARY CHANNEL
binaryChannelType
sourceFileName string
mimeType string
encoding string
lastModificationDate dateTime
url anyURI
content [0..1] base64Binary
FOLKSONOMY
TAXONOMY / CONTROLLED VOCABULARY
taxonomyType
name (nameType)
rootTopics [0..1] (rootTopicsType)
RefersTo
TOPIC
topicType
name systemNameType
allowsReferrerContentObjects boolean
order long
taxonomy [0..1] taxonomyType
owner [0..1] repositoryUserType
parentTopic [0..1] topicType
childTopics [0..1] (childTopicsType)
event Object
Portal Section Object
local Government Service Object
CONTENT OBJECT
contentObjectType
contentObjectTypeName string
systemName systemNameType
owner [0..1] repositoryUserType
address Type
geo location Type
administrative metadata Type
COMPLEX PROPERTY
complexCmsPropertyType
labelElementPath string
xs:string
xs:long
xs:dateTime
SIMPLE PROPERTY

Με εξαίρεση το «Αντικείμενο Περιεχομένου» (Content Object) και την «Σύνθετη Ιδιότητα Αντικειμένου» (Complex Property) που εμφανίζονται με διπλή γραμμή στο σχήμα, οι ιδιότητες όλων των υπόλοιπων έτοιμων δομών (Topic, Taxonomy, Space, Repository User) είναι προκαθορισμένες από το σύστημα και δεν μπορούν να αλλαχτούν από το χρήστη. Για αυτές ο χρήστης μπορεί να δημιουργεί στιγμιότυπα αλλά όχι να αλλάζει τον αριθμό και το είδος των ιδιοτήτων τους.

Το **«Αντικείμενο Περιεχομένου» δεν έχει προκαθορισμένες ιδιότητες**. Όπως φαίνεται από το σχήμα κάθε νέος τύπος περιεχομένου (π.χ. newsITem) κληρονομεί κάποιες βασικές ιδιότητες από τον γενικό αφαιρετικό τύπο "contentObjectType" και από κει και πέρα μπορεί να προεκτείνει με οποιοδήποτε τρόπο την βασική δομή προσθέτοντας ιδιότητες.

Με βάση το παραπάνω το **«Αντικείμενο Περιεχομένου»** αποτελεί τη κεντρική δομή μοντελοποίησης και όλες οι ιδιότητές του μπορούν να καθοριστούν και να παραμετροποιηθούν πλήρως από το χρήστη του συστήματος. Το μοντέλο ιδιοτήτων ενός αντικειμένου περιεχομένου προσδιορίζει τον **Τύπο** του αντικειμένου (π.χ. άρθρο, βιογραφικό, ανακοίνωση, δελτίο τύπου, εικόνα, video, απόφαση συμβουλίου, εκδήλωση, ερωτηματολόγιο, δημοσκόπηση, blog entry, σχόλιο, οργανισμός, πρόσωπο, κλπ.) και καθορίζεται μέσω ενός XSD Schema που **επεκτείνει** το βασικό XSD Schema "contentObjectType" που παρέχει το Astroboa. Ουσιαστικά δηλαδή κάθε τύπος αντικειμένου που αποθηκεύεται στο Astroboa ορίζεται μέσω ενός XSD Schema και περιγράφει/αντιστοιχεί σε όλα τα XML Documents των οποίων η μορφή (η πληροφορία που μπορούν να περιγράψουν) καθορίζεται από την γραμματική του XSD Schema.

Στην απλούστερη μορφή του ένα αντικείμενο περιεχομένου έχει μόνο **ιδιότητες απλού τύπου (simple type property)** που αντιστοιχούν στους απλούς τύπους που υποστηρίζει το XSD Schema όπως Integer, Double, Boolean, Binary, Long, Date, String, κλπ.

Για πιο σύνθετες μορφές αντικειμένων, ο χρήστης μπορεί να **ομαδοποιήσει τις ιδιότητες των αντικειμένων** υπό την μορφή **complex content object properties** που απαρτίζονται από simple type properties ή με τη σειρά τους

από άλλα complex content object properties σε οσοδήποτε βάθος (arbitrary nesting). Στο XSD σχήμα τα complex content object properties αντιστοιχούν και περιγράφονται ως Complex Types. Αν ο ορισμός τους γίνει ανεξάρτητα ενός τύπου αντικειμένου (έξω από τον ορισμό του αντικειμένου), τα complex content object properties αποτελούν **επαναχρησιμοποιήσιμους τύπους στατικών ή δυναμικών ιδιοτήτων (Static Complex Property ή Dynamic Content Object Aspect)** και μπορούν να επαναχρησιμοποιηθούν σε οποιοδήποτε τύπο αντικειμένου είτε στατικά στον ορισμό των ιδιοτήτων του είτε δυναμικά κατά την διάρκεια δημιουργίας στιγμιοτύπων χωρίς να χρειάζεται να έχουν προηγουμένως ορισθεί στο XSD Schema του.

Ετσι παράλληλα με το στατικό ορισμό των ιδιοτήτων για κάθε τύπο αντικειμένου περιεχομένου, όλα τα complex content object properties που οριζονται στο XSD Schema σαν ανεξάρτητα complex types αποτελούν «Δυναμικές Συνιστώσες Ιδιοτήτων» οι οποίες δεν είναι απαραίτητο να εισάγονται στατικά κατά τον ορισμό του τύπου του αντικειμένου αλλά μπορούν να προστίθενται δυναμικά σε ένα στιγμιότυπο (Instance) αντικειμένου κατά την διάρκεια της δημιουργίας του. Με αυτό τον τρόπο μπορούν να υλοποιηθούν προαιρετικά set από ιδιότητες που χρησιμοποιούνται κατά περίπτωση όταν τις χρειάζεται ένα συγκεκριμένο αντικείμενο. Παράδειγμα τέτοιων ιδιοτήτων αποτελούν οι ιδιότητες που καθορίζουν τη δημοσίευσή του. Χωρίς να χρειάζεται να προστεθούν σε κάθε τύπο αντικειμένου που δημιουργείται είναι διαθέσιμες εφόσον κάποιος επιθυμεί να δημοσιεύσει ένα αντικείμενο και μπορεί να τις προσθέτει δυναμικά μόνο σε αυτή την περίπτωση.

Μέσω έτοιμων προ-μοντελοποιημένων ιδιοτήτων και παραμέτρων στα XSD Schemata που παρέχει το Astroboa, υποστηρίζεται η ασφαλής πρόσβαση στο περιεχόμενο προσφέροντας έλεγχο σε επίπεδο αντικειμένων περιεχομένου αλλά και ιδιοτήτων των αντικειμένων (fine level control). Κατά την δημιουργία κάθε αντικειμένου ο χρήστης / προγραμματιστής μπορεί να ορίζει τους χρήστες που έχουν το δικαίωμα να διαβάσουν, να τροποποιήσουν, να σβήσουν το αντικείμενο καθώς και τους χρήστες που μπορούν να προσθέτουν δικές τους ετικέτες στο αντικείμενο. Κατά την διάρκεια ορισμού του τύπου κάθε αντικειμένου είναι επίσης δυνατό να οριστούν μέσω του XSD Schema

ποιοι ρόλοι χρηστών μπορούν να δουν ή να τροποποιήσουν κάθε πεδίο του αντικειμένου.

Επίσης στην αρχική του εγκατάσταση, το Astroboa παρέχει έτοιμα XSD Schemata που μοντελοποιούν το πρότυπο μεταδεδόμενων Dublic Core, το πρότυπο open social για την μοντελοποίηση προσώπων και ένα περιεκτικό σύνολο απο βασικούς τύπους περιεχομένου και ιδιοτήτες για τη δημιουργία δικτυακών τόπων. Με αυτό τον τρόπο το Astroboa, χωρίς να απαιτεί καμία επιπλέον μοντελοποίηση παρέχει out of the box την λειτουργία ενός Web Content Management εργαλείου ενώ ταυτόχρονα επιτρέπει την περαιτέρω ανάπτυξη του προσφερόμενου μοντέλου εφόσον ο οργανισμός επιθυμεί να δημιουργήσει ένα πληρέστερο μοντέλο της πληροφορίας του πλέον του Δικτυακού Τόπου. Ενδεικτικά αναφέρονται μερικοί από τους μοντελοποιημένους τύπους περιεχομένου:

- Διαχειριστικά Μεταδεδομένα (Dublin Core Metadata)
- Βασικό Περιεχόμενο
- Σχετικό Περιεχόμενο
- Λίστα Αναφορών σε Αντικείμενα Βασικού Περιεχομένου
- Πρόσωπο (Open Social compatible)
- Οργανισμός
- Τοποθεσία
- Σημείο Γεωγραφικού Εντοπισμού
- Σύνδεσμος
- Λίστα Συνδέσμων
- Αρχείο
- Λίστα Αρχείων
- Λίστα Αναφορών σε Αρχεία
- Εμβόλιμο Πολυμεσικό Περιεχόμενο
- Εκδήλωση
- Προγραμματισμένη Περιοχή Περιεχομένου
- Περιοχή Δυναμικού Περιεχομένου
- Δικτυακή Πύλη
- Τομέας Δικτυακής Πύλης
- Στατιστικά Ανάγνωσης Περιεχομένου

- Σχόλιο σε Περιεχόμενο
- Βαθμολόγηση Περιεχομένου
- Στοιχεία Ασφάλειας Αντικειμένου Περιεχομένου
- Στοιχεία Δημοσίευσης Αντικειμένου Περιεχομένου
- Στοιχεία ροής εργασίας Αντικειμένου Περιεχομένου

Η μοντελοποίηση των Dublin Core Metadata θεωρείται βασικός τύπος σύνθετης ιδιότητας αντικειμένου (complex content object property / XSD Complex Type) που παρέχεται από το σύστημα και προστίθεται υποχρεωτικά σε όλους του τύπους αντικειμένων για να περιγράψει τα διαχειριστικά μετα-δεδομένα τους (Ιδιοκτήτης, Δημιουργός, Ημ/νία Δημιουργίας, Οργανισμός, Θεματικές Ενότητες Κατηγοριοποίησης, κλπ.).

Συνδυασμός χαρακτηριστικών

Το Astroboa διαθέτει ένα μοναδικό συνδυασμό χαρακτηριστικών που προσφέρουν σημαντική ευελιξία τόσο στους τελικούς χρήστες που διαχειρίζονται το περιεχόμενου του οργανισμού όσο και στους παραγωγούς λογισμικού που χρησιμοποιούν το Astroboa ως κεντρική υποδομή μοντελοποίησης και αποθήκευσης περιεχομένου και ως πλατφόρμα υπηρεσιών για να χτίσουν νέες εφαρμογές περιεχομένου. Μια συνοπτική περιγραφή των σημαντικότερων χαρακτηριστικών του Astroboa παρουσιάζεται στην λίστα που ακολουθεί. Τα χαρακτηριστικά είναι κατηγοριοποιημένα σύμφωνα με:

- Τα υποστηριζόμενα πρότυπα και τις δυνατότητες μοντελοποίησης περιεχομένου
- Τις δυνατότητες που προσφέρει η διαδικτυακή εφαρμογή Διαχείρισης στους τελικούς χρήστες
- Τις προσφερόμενες προγραμματιστικές διεπαφές και υπηρεσίες ιστού για την ανάπτυξη εφαρμογών περιεχομένου και την ενσωμάτωση σε αρχιτεκτονικές Servise Oriented Architecture (SOA)
- Την απόδοση, την συνέπεια και συνοχή των πληροφοριών, και τις δυνατότητες κλιμάκωσης
- Την προσφερόμενη ασφάλεια

Μοντελοποίηση Περιεχομένου & Πρότυπα

Ισχυρή υποστήριξη για προτυποποιημένη μοντελοποίηση του αποθηκευμένου περιεχομένου μέσω του προτύπου των XSD Schemas (Πρότυπη Γραμματική XML). Μοντελοποίηση οσοδήποτε πολύπλοκης δομής περιεχομένου και δυνατότητα εύκολης μετανάστευσης δεδομένων που υφίστανται ή μπορούν να εξαχθούν υπό μορφή XML.

Η χρήση των XSD Schemas παρέχει ασφάλεια ως προς την προσβασιμότητα στο περιεχόμενο ανεξάρτητα του εργαλείου διαχείρισης και επίσης κάνει εφικτή την χρήση των έτοιμων μοντέλων πληροφορίας που είναι διαθέσιμα υπό μορφή XSD Schema (π.χ. NewsML, eML, κλπ.).

Εισαγωγή και εξαγωγή οποιουδήποτε αντικειμένου περιεχομένου ή οποιασδήποτε λίστας αποθηκευμένων αντικειμένων σε XML documents που ακολουθούν τα περιγεγραμμένα XSD Schemas.

Αυτόματη αντιστοίχιση των XSD Schemas σε προγραμματιστικά αντικείμενα (Schema Elements σε Java Objects και Schema Complex Types σε Java Object Properties).

Εύκολη αλλά και ευέλικτη μοντελοποίση από μη ειδικούς χρήστες (με επιχειρησιακή αλλά όχι τεχνική κατάρτιση) μέσω έξι (6) Αφαιρετικών Εννοιών Μοντελοποίησης:

1. Αντικείμενα Περιεχομένου
2. Στατικά & Δυναμικά Πεδία αντικειμένων
3. Ταξινομίες (Δέντρα δεσμευμένων όρων / κατηγοριών)
4. Ετικέτες Χρηστών
5. Χώροι
6. Χρήστες / Παραγωγοί Περιεχομένου

Δυνατότητα επαναχρησιμοποιήσιμων σύνθετων τύπων πεδίων (XSD Schema complex elements) σε οσοδήποτε βάθος (arbitrary nesting). Οι τύποι πληροφορίας ορίζονται μόνο μια φορά (π.χ. Διεύθυνση) και μπορούν να χρησιμοποιηθούν σε οποιοδήποτε αντικείμενο (π.χ. ένα πρόσωπο έχει διεύθυνση, ένας οργανισμός έχει διεύθυνση, κλπ.)

Δυνατότητα δυναμικής σύνθεσης περιεχομένου στο επίπεδο των στιγμιότυπων ενός τύπου περιεχομένου (content object instances). Είναι δηλαδή δυνατή η προσθήκη δυναμικών πεδίων που ισχύουν μόνο για το συγκεκριμένο στιγμιότυπο ενός τύπου αντικειμένου. Π.χ. μπορούμε δυναμικά να προσθέσουμε γεωγραφικές συντεταγμένες σε άρθρα, εκδηλώσεις και ομιλίες ακόμα και αν δεν έχει οριστεί στα σχετικά XSD Schemas αυτών των τύπων πληροφορίας ότι έχουν πεδία για γεωγραφικές συντεταγμένες. Σε αυτή την περίπτωση τα πεδία προσθέτονται αυτόματα από το Astroboa.

-Υποστήριξη ιεραρχικών Δέντρων Όρων (taxonomies) για content tagging και δημιουργία λεξικών δεσμευμένων όρων

-Υποστήριξη ετικετών συστήματος και προσωπικών ετικετών για κάθε παραγωγό περιεχομένου (taxonomies και folksonomies)

-Ισχυρή υποστήριξη πολυγλωσσικού περιεχομένου από το επίπεδο της μοντελοποίησης με την δυνατότητα πολυγλωσσικών πεδίων και πολυγλωσσικών ετικετών / όρων λεξικού

-Χρήση του προτύπου Dublin Core για την δημιουργία διαχειριστικών μεταδεδομένων ανά αντικείμενο περιεχομένου

-Out of the box παροχή έτοιμου μοντέλου περιεχομένου για την διαχείριση εγγράφων, multimedia αντικειμένων, τοποθεσιών, geotags, προσώπων (opensocial standard), γεγονότων και ευρύτερα περιεχομένου με προσανατολισμό την δημοσίευση σε δικτυακό τόπο (δημιουργία portal sections, δυναμικών περιοχών περιεχομένου, προγραμματισμένων περιοχών περιεχομένου).

-Αποθήκευση του περιεχομένου σε ιεραρχική μορφή κόμβων και ιδιοτήτων με χρήση του προτύπου JSR-170 (Java Content Repository) για την υλοποίηση της αποθήκης περιεχομένου. Στον διεθνή επιχειρησιακό χώρο οι προτυποποιημένες ιεραρχικές αποθήκες περιεχομένου αντικαθιστούν σταδιακά την απευθείας χρήση βάσεων δεδομένων παρέχοντας μια αφαιρετική και καταλληλότερη προσέγγιση στην αποθήκευση δεδομένων που είναι από τη φύση τους ιεραρχικής μορφής. Επιπλέον προσφέρουν σημαντικές υπηρεσίες που επιτυγχάνονται πιο δύσκολα με τις βάσεις δεδομένων.

Αυτόματη αντιστοίχιση και αποθήκευση των αντικειμένων και τύπων που ορίζονται στα XSD schemas σε κόμβους (nodes) και ιδιότητες (properties) στην αποθήκη περιεχομένου JSR-170

Δυνατότητα πολλαπλών εκδόσεων (versions) ανά αντικείμενο περιεχομένου

-Full text search για όλα τα πεδία οποιουδήποτε τύπου περιεχομένου και αυτόματο indexing και αναζήτηση σε πεδία που περιέχουν ψηφιακά αρχεία τύπου doc, excel, powerpoint, openoffice formats , pdf

Διαδικτυακή Εφαρμογή Διαχείρισης Περιεχομένου και Συνεργασίας

AJAX Web Based Περιβάλλον Διαχείρισης και Εισαγωγής Περιεχομένου με υποστήριξη πολλαπλών τοπικών και απομακρυσμένων Content Servers που ο καθένας μπορεί να διαχειρίζεται πολλαπλές αποθήκες περιεχομένου (content repositories), που η κάθε μια μπορεί να παράγει πολλαπλά portals / web sites.

Αυτόματη παραγωγή των φορμών εισαγωγής περιεχομένου από τα XSD Schema definitions χωρίς περιορισμό στο βάθος των σύνθετων πεδίων (subforms)

Αυτόματη επικύρωση (validation) των πεδίων στις φόρμες μέσω του XSD Schema Definition

Δυναμική σύνθεση νέων τύπων περιεχομένου από υφιστάμενους κατά την διάρκεια εισαγωγής περιεχομένου στις φόρμες, χωρίς καμία συγγραφή κώδικα και χωρίς την ανάγκη περιγραφής σε κάποιο XML Schema ή άλλο configuraton file.

Ροή εργασίας για δημοσίευση αντικειμένων στο web με βήματα έγκρισης, προσωρινής απόρριψης για αλλαγές και μόνιμης απόρριψης. Στην επόμενη έκδοση θα υποστηρίζεται δυνατότητα εγκατάστασης εναλλακτικών ροών εργασίας μέσω ανεξάρτητου workflow engine.

Διαχείριση πολυγλωσσικών ταξινομιών (Ιεραρχικών Δέντρων Όρων) και των ετικετών των χρηστών με δυνατότητα ανάταξης των ιεραρχικών δέντρων μέσω drag & drop

Δυνατότητα από το διαχειριστικό περιβάλλον εξαγωγής και εισαγωγής των Ιεραρχικών Δέντρων Όρων μέσω XML

Πλοήγηση στο περιεχόμενο με βάση τα ιεραρχικά δέντρα όρων (θεματικές ενότητες, όροι λεξικών), τις ετικέτες χρηστών, τους τύπους περιεχομένου και την χρονολογία εισαγωγής τους στο σύστημα, τις ομάδες χρηστών και τους χρήστες

Κατηγοριοποίηση περιεχομένου και εισαγωγή δεσμευμένων όρων στα πεδία της φόρμας μέσω drag & drop

Dashboard για γρήγορη επισκόπηση των τελευταίων εισαγωγών, των πιο δημοφιλών δημοσιευμένων αντικειμένων και των αντικειμένων που περιμένουν έγκριση δημοσίευσης. Στην επόμενη έκδοση θα υπάρχει δυνατότητα προσθήκης από το χρήστη widgets με προσωποποιημένες αναζητήσεις.

Αναζήτηση περιεχομένου με full text search (αλα google) ή σύνθετες αναζητήσεις με δυναμικά κριτήρια που συνθέτονται από το χρήστη.

Υποστήριξη γραφικού εργαλείου δημιουργίας σύνθετων ερωτήσεων ανάκτησης περιεχομένου και δυνατότητα αποθήκευσης των ερωτήσεων για επαναχρησιμοποίηση.

Υποστήριξη μέσα από το διαχειριστικό περιβάλλον εξαγωγής οποιουδήποτε αντικειμένου ή οποιουδήποτε αποτελέσματος αναζήτησης σε XML και Excel

Υποστήριξη Virtual Spaces που εξομοιώνουν την λειτουργία του File System και δυνατότητα μεταφοράς των αντικειμένων σε Virtual Space κατά την δημιουργία τους ή μέσω drag & drop.

Υποδομή Υπηρεσιών Περιεχομένου & Προγραμματιστικές Διεπαφές (content APIs)

Ταχύτατη ανάπτυξη (rapid development) νέων επιχειρησιακών εφαρμογών περιεχομένου μέσω των παρεχόμενων προγραμματιστικών διεπαφών και των βοηθητικών υπηρεσιών και βιβλιοθηκών

Εννιαίο client interface (προγραμματιστική διεπαφή για το client application) για local ή remote πρόσβαση στα content repositories, υλοποιημένο σαν EJB3 local και Remote content-api

SOAP & Restful Content-API με έξοδο XML και JSON

Ισχυρό content search API με Hibernate-Like search criteria

Ισχυρό RESTful search API μέσω expression language για την σύνθεση των κριτηρίων αναζήτησης.

ESB Module για σύνδεση με εξωτερικά συστήματα (Databases, Filesystems, Mail Servers, FTP Servers, web services, κλπ.) κυρίως για άντληση περιεχομένου

Maven Archetype για την αυτόματη παραγωγή template content client application.

Έτοιμη Βιβλιοθήκη με portal-api, content retrieval utilities και ολοκλήρωση των τεχνολογιών Seam Framework, Spring Framework, Javaserver Faces και Facelets για την εύκολη δημιουργία δυναμικών πυλών περιεχομένου. Δυνατότητα δυναμικής παραγωγής των σελίδων σαν HTML, RSS ή ATOM feeds μέσω παραμέτρων του portal-api

Παροχή έτοιμου Template Web Site (wiki like) για γρήγορη δημιουργία δικτυακών τόπων

Χρήση προτύπων σε όλα τα σημεία ανάπτυξης (APIs, Web Services, SW Component synthesis).

Απόδοση & Κλιμάκωση Συστήματος / Συνέπεια και Συνάφεια Αποθήκης Περιεχομένου

Υποστήριξη full distributed transactions για χρήση σε κατανεμημένα συστήματα και διασφάλιση της συνέπειας και της συνάφειας των δεδομένων

Global 2-nd level content / query caching με έτοιμες περιοχές για caching 1, 5, 10, 20, 30 λεπτών, κλπ. Ο μηχανισμός caching μπορεί να ενεργοποιηθεί ανά ερώτημα σε κάθε αποθήκη περιεχομένου και εξασφαλίζει ταχεία ανάκτηση του περιεχομένου σε περιβάλλοντα με πολύπλοκες αναζητήσεις και πολλές ταυτόχρονες προσβάσεις (αυτόματη κλιμάκωση).

Lazy loading των αντικειμένων περιεχομένου και των binary data (video, images) που επιτρέπει τη ανάκτηση δεκάδων χιλιάδων αντικειμένων σε μία αναζήτηση (ακόμη και αν περιέχουν binary channels πολλών δεκάδων megabytes) χωρίς επιβάρυνση του συστήματος. Τα περιεχόμενα πολύπλοκων αντικειμένων που περιέχουν πολλαπλά ψηφιακά αρχεία και πολλά σύνθετα πεδία ανακτώνται αυτόματα μόνο όταν ζητούνται από το χρήστη εξασφαλίζοντας άμεση κλιμάκωση σε υψηλά φορτία.

Ασφάλεια

Υπηρεσίες ασφάλειας στο επίπεδο του API με κλειδιά ή username / password και αυτόματη ανταλλαγή tokens μεταξύ client content application και Repository Server.

Identity Store Agnostic: Το Astroboa παρέχει out of the box αποθήκευση των χρηστών σε προδιαγεγραμμένη βάση δεδομένων και επιτρέπει την διαχείρισή τους μέσω της προσφερόμενης διαδικτυακής εφαρμογής διαχείρισης περιεχομένου. Ταυτόχρονα μπορεί να συνδεθεί και με οποιοδήποτε εξωτερικό identity store (DB / LDAP) εφόσον προϋπάρχει στον οργανισμό σύστημα διαχείρισης χρηστών. Η σύνδεση με υφιστάμενα συστήματα διαχείρισης χρηστών γίνεται μέσω του επιχειρησιακού προτύπου JAAS και υποστηρίζεται διαφορετική βάση χρηστών ανά αποθήκη περιεχομένου (separate JAAS Policy Domain per content repository). Στην υλοποίηση για τους ΟΤΑ δεν χρησιμοποιείται η παρεχόμενη από το Astroboa βάση χρηστών αλλά εξωτερικό σύστημα διαχείρισης των ταυτοτήτων των χρηστών που βασίζεται σε LDAP.

Λεπτομερείς κανόνες ασφάλειας στο επίπεδο των αντικειμένων περιεχομένου. Μπορεί να ρυθμιστεί η πρόσβαση στις λειτουργίες ανάγνωσης, αλλαγής, αφαίρεσης και κατηγοριοποίησης ανά στιγμιότυπο αντικειμένου περιεχομένου (π.χ. μπορεί να αποκλειστεί η πρόσβαση σε συγκεκριμένο χρήστη μόνο για ένα έγγραφο).

Rule Based engine με Security Rules στο επόμενο Version.

Τεχνολογίες & Ανοικτό Λογισμικό

Ανάπτυξη με τις τελευταίες και πιο διαδεδομένες τεχνολογίες ανοικτού λογισμικού για επιχειρησιακά συστήματα όπως JEE5, Spring Framework, EJB3, Jboss Cache, JSF, Rich Faces, Seam Framework, Mule ESB, Postgres, jbpm workflow engine, Drools rule engine, κλπ.

Παρέχεται μέσω της άδειας ανοικτού λογισμικού LGPL που επιτρέπει την χρήση και υιοθέτησή του ακόμα και από κλειστά περιβάλλοντα.

ΚΕΦΑΛΑΙΟ 4: TAXONOMIES/TOPICS/ΣΧΗΜΑΤΑ με τη χρήση του restfull API από το λογισμικό Astroboa

Retrieve a taxonomy collection from a repository

Retrieve taxonomies of a repository either in XML or JSON format.

URL: http://<Astroboa Host or IP>/resource-api/<repository-id>/taxonomies

HTTP Method: GET

Query Parameters: output, callback, depth

- **output** [optional]:
  Use *output* to specify the output format: XML or JSON
- **callback** [optional]:
  Use *callback* to specify the name of a JavaScript function callback that the web-server will use to wrap the returned response in. Function callbacks allow for making cross-site requests - JSON with Padding (JSON-P) or XML with Padding (XML-P)
- **depth** [optional]:
  Use *depth* to specify whether topics of a taxonomy should be included in the response

Depth	Description
0	Fetch only the taxonomy
1	Fetch the taxonomy and its root topics (default)
-1	Fetch the taxonomy and the whole topic tree beneath

Supported Response Body Formats: XML (default), JSON

Users may use one of the following ways in order to specify the desired response body format:

1. Use the *output* query parameter.
2. Use the file name suffix 'xml' or 'json'.
3. Use the HTTP **Accept** header with the values

- '*text/xml*' or '*application/xml*' for XML response body format
- '*application/json*' for JSON response body format

URL-Based negotiation overrides any Accept header provided by the clent, therefore option (1) overrides all other options and option (2) overrides option (3). If none of the above is provided, the response body will be returned in XML format. Usage examples below, provide a case for each one of the available options.

Response:

- 200 HTTP Code (OK) : if one or more taxonomies were successfuly retrieved

Notes:

- http://<Astroboa Host or IP>/resource-api/<repository-id>/taxonomy has been deprecated and will not be supported in the 4.x releases

Create a new taxonomy

Create new taxonomies in a repository. You should always provide a system name for a taxonomy. A system name should be composed only of alphanumeric characters, dash(-), undescore(_), dot(.) and colon(:). If not, the provided system name will be modified to comply with the above rule. It is also possible to import a full taxonomy hierarchy, i.e. a taxonomy with topic which may have subtopics, etc. See more in the usage examples.

URL: http://<Astroboa Host or IP>/resource-api/<repository-id>/taxonomies
HTTP Method: POST
Headers [*mandatory*]: Authorization: Basic base64encode(username:password)
Supported Request Body Formats: XML, JSON

Although, the request body format is automatically recognized by Astroboa, users may explicitly specify it with the use of the HTTP **Content-Type** header with the values

- '*text/xml*' or '*application/xml*' for XML request body format
- '*application/json*' for JSON request body format

Response:

- 201 HTTP Code (OK) : if a taxonomy was successfuly created
- 401 HTTP Code (UNAUTHORIZED): if user is not authorized to create a taxonomy

Notes:

- http://<Astroboa Host or IP>/resource-api/<repository-id>/taxonomy has been deprecated and will not be supported in the 4.x releases

ΚΕΦΑΛΑΙΟ 5: Αντικείμενο περιεχομένου, σχεδιασμός σε σχήμα δυναμικά

Ένα αντικείμενο περιεχομένου απεικονίζει τη δομή που έχει ένας συγκεκριμένος τύπος πληροφορίας που παράγεται ή διακινείται εντός ή εκτός κάποιου οργανισμού (π.χ δελτία τύπου, έγγραφα, τιμολόγια, ενημερωτικά φυλλάδια, κλπ.) Το αντικείμενο περιεχομένου παρέχει την υποδομή για να περιγραφούν:

- Οι ιδιότητες του τύπου πληροφορίας υπό την μορφή πεδίων / ιδιοτήτων που συνθέτουν το αντικείμενο (π.χ. title, description, dayOfPublication, κλπ.)
- Το είδος της πληροφορίας που αποθηκεύεται σε κάθε πεδίο του αντικειμένου (π.χ κείμενο, δεκαδικός αριθμός αριθμός, ημερομηνία, κλπ.)
- Το ελάχιστο και μέγιστο επιτρεπόμενο αριθμό τιμών που μπορούν να καταχωρηθούν σε κάθε πεδίο
- Οι επιτρεπόμενες τιμές που δέχεται κάθε πεδίο (π.χ. μόνο κεφαλαία ελληνικά)
- Το όνομα συστήματος με το οποίο είναι προσβάσιμη η πληροφορία σε κάθε πεδίο (όνομα συστήματος για κάθε πεδίο)

- Το όνομα συστήματος με το οποίο είναι η προσβάσιμη η πληροφορία που ομαδοποιεί το αντικείμενο, δηλαδή το όνομα του τύπου του αντικειμένου (π.χ. event, pressRelease, κλπ.)
- Τα εναλλακτικά ονόματα σε διάφορες γλώσσες με τα οποία θα γίνεται αναφορά στα πεδία ανάλογα με τη γλώσσα του χρήστη που "βλέπει" το αντικείμενο (π.χ. Τίτλος, Περιγραφή, Ημέρα Δημοσίευσης)
- Τα εναλλακτικά ονόματα σε διάφορες γλώσσες με τα οποία θα γίνεται αναφορά στον τύπο του αντικειμένου (π.χ. Εκδήλωση, Δελτίο Τύπου, κλπ.)
- Ο τρόπος χρήσης του αντικειμένου για να βοηθάει όσους το χρησιμοποιούν ή μοντελοποιούν συναφή αντικείμενα
- Ο τρόπος χρήσης του κάθε πεδίου για να βοηθάει όσους παράγουν περιεχόμενο με βάση το συγκεκριμένο αντικείμενο (π.χ. συμπληρώνουν στη Διαδικτυακή Εφαρμογή Διαχείρισης του Astroboa τις φόρμες που αντιστοιχούν στα πεδία του αντικειμένου)

Παράδειγμα Αντικειμένου Περιεχομένου:

Εκδήλωση

- Τίτλος [Απλό Κείμενο 1:1]
- Περιγραφή [Απλό Κείμενο 1:1]
- Θέμα / Λέξεις Κλειδία [Αναφορά σε Ταξινομικό Όρο / Ετικέτα 0:N]
- Κείμενο Εκδήλωσης [HTML Κείμενο 1:1]
- Ημ/νίες Εκδήλωσης [Ημερομηνία 1:N]
- Δωρεάν [Λογική Τιμή (true/false) 0:1]
- Thumbnail [Ψηφιακό Κανάλι 0:N]
- Φωτογραφία [Ψηφιακό Κανάλι 0:N]
- Τιμή [Ακέραιος Αριθμός 0:1]
- Γεωγραφικός Εντοπισμός [Σύνθετος Τύπος / Ιδιότητα:Γεω-Εντοπισμός 0:1]

Η σημείωση MinValues:MaxValues που ακολουθεί τον τύπο κάθε πεδίου υποδηλώνει τον ελάχιστο και μέγιστο αριθμό τιμών για το πεδίο (cardinality).

Οπότε:

- Η σημείωση 0:1 σημαίνει ότι το πεδίο δεν είναι υποχρεωτικό (επιτρέπεται να μην έχει τιμη) και δέχεται το πολύ μία τιμή.
- Η σημείωση 1:1 σημαίνει ότι το πεδίο είναι υποχρεωτικό και δέχεται μόνο μια τιμή.
- Η σημείωση 1:N σημαίνει ότι το πεδίο είναι υποχρεωτικό και δέχεται οσεσδήποτε τιμές (δηλαδή πρέπει να έχει τουλάχιστον μια τιμή ή και περισσότερες χωρίς περιορισμό στον αριθμό τους).

Ο τύπος Γεω-Εντοπισμός είναι σύνθετος. Ορίζεται δηλαδή με βάση τις παρεχόμενες δομές του Astroboa ως "Σύνθετη Ιδιότητα Αντικειμένου Περιεχομένου" και περιέχει με τη σειρά του τα πεδία που χρειάζονται για την καταγραφή των γεωγραφικών συντεταγμένων. Μπορούμε εναλλακτικά να ορίσουμε τα πεδία για τις γεωγραφικές συντεταγμένες εντός του αντικειμένου αλλά είναι πιο αποδοτικό να ορίσουμε ξεχωριστά μια σύνθετη ιδιότητα που φέρει τα πεδία και μπορεί να χρησιμοποιηθεί και σε άλλους τύπους αντικειμένων που χρειάζονται γεωγραφική πληροφορία.

Είναι σημαντικό να παρατηρήσουμε ότι οι σύνθετες ιδιότητες ορίζουν μια σειρά πεδίων όπως συμβαίνει και με τα αντικείμενα περιεχομένου όμως διαφέρουν ως προς αυτά στο γεγονός ότι δεν αποτελούν ανεξάρτητες οντότητες αλλά υφίστανται μόνο ως τμήματα άλλων αντικειμένων.

Μπορούμε να πούμε δηλαδή ότι οι σύνθετες ιδιότητες αποτελούν τμήματα πληροφορίας που είναι κοινά στα διάφορα αντικείμενα. Για να μην περιγράφουμε τα ίδια πεδία ξανά και ξανά σε κάθε τύπο αντικειμένου αποσπούμε τα κοινά πεδία σε ένα επαναχρησιμοποιήσιμο τύπο που αποκαλούμε σύνθετη ιδιότητα αντικειμένου και της δίνουμε κάποιο όνομα όπως στο παράδειγμα το όνομα "Γεω-Εντοπισμός". Κατόπιν αν χρειαζόμαστε αυτή την ομάδα πεδίων σε κάποιο νέο τύπο αντικειμένου φτιάχνουμε ένα πεδίο το οποίο ορίζουμε να έχει σαν τύπο το όνομα της Σύνθετης Ιδιότητας. Στο παράδειγμά μας το πεδίο "Γεωγραφικός Εντοπισμός" δεν παίρνει απλές τιμές όπως τα άλλα πεδία αλλά έχει οριστεί να κληρονομεί τα πεδία που έχουν οριστεί για την σύνθετη ιδιότητα με το όνομα "Γεω-Εντοπισμός". Δηλαδή το Astroboa "καταλαβαίνει" ένα σύνθετο πεδίο και αυτόματα εισάγει στο

αντικείμενο όλα τα πεδία που ορίζει η σύνθετη ιδιότητα σαν να είχαν περιγραφεί εσωτερικά του αντικειμένου.

Παρακάτω φαίνεται ένα παράδειγμα με το XSD Schema που πρέπει να γράψουμε και να εισάγουμε στο Astroboa για να μπορούμε να βάζουμε "εκδηλώσεις" στην αποθήκη περιεχομένου. Το tag **<xs:extension base="bccmsmodel:contentObjectType">** που εμφανίζεται σημειωμένο με κόκκινα γράμματα στο σχήμα υποδηλώνει ότι το element "eventObject" στο XSD Schema προεκτείνει τον τύπο "**bccmsmodel:contentObjectType**". Όλα τα αντικείμενα περιεχομένου του Astroboa πρέπει να δηλώνονται στο XSD Schema με αυτό τον τρόπο, δηλαδή ως root elements που προεκτείνουν τον αφαιρετικό τύπο "**bccmsmodel:contentObjectType**". Οι ιδιότητες

Τίτλος, Περιγραφή και Θέμα που αναφέρονται στο παράδειγμα παραπάνω δεν εμφανίζονται στο σχήμα διότι αποτελούν μέρος του σύνθετου τύπου "administrativeMetadataType" που περιέχει όλα τα διαχειριστικά μεταδεδομένα του αντικειμένου (Dublin core Metadata) και κληρονομείται αυτόματα σε όλα τα αντικείμενα περιεχομένου (ουσιαστικά με όρους XSD κληρονομείτε σε όλα τα elements που προεκτείνουν τον αφαιρετικό τύπο "bccmsmodel:contentObjectType").

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.betaconceptframework.org/schema/astroboa/web/event"
        xmlns:bccmsmodel="http://www.betaconceptframework.org/schema/astroboa/model"
        xmlns:geoTagType="http://www.betaconceptframework.org/schema/astroboa/web/geoTagType">

   <xs:import namespace="http://www.betaconceptframework.org/schema/astroboa/model"
         schemaLocation="astroboa-model-3.0.0b1.xsd" />
  <xs:import namespace="http://www.betaconceptframework.org/schema/astroboa/web/geoTagType"
         schemaLocation="goeTag-1.0.xsd" />

   <xs:element name="eventObject"
         bccmsmodel:description="This element defines a content object which models events
            that take place at a certain place in a specified period of time.">
      <xs:annotation>
```

```
            <xs:documentation xml:lang="en">Event</xs:documentation>
            <xs:documentation xml:lang="el">Εκδήλωση</xs:documentation>
      </xs:annotation>
      <xs:complexType>
            <xs:complexContent>
               <xs:extension base="bccmsmodel:contentObjectType">
                     <xs:sequence>
                          <xs:element name="thumbnail"
                             minOccurs="0"
                             maxOccurs="1"
                             type="bccmsmodel:binaryChannelType"
                             bccmsmodel:description="A small image
related to the event">
                             <xs:annotation>
                                 <xs:documentation
xml:lang="en">Thumbnail</xs:documentation>
                                 <xs:documentation
xml:lang="el">Εικονίδιο</xs:documentation>
                             </xs:annotation>
                          </xs:element>
                          <xs:element name="image"
                             minOccurs="0"
                             maxOccurs="1"
                             type="bccmsmodel:binaryChannelType"
                             bccmsmodel:description="Image related to
the event">
                             <xs:annotation>
                                 <xs:documentation
xml:lang="en">Image</xs:documentation>
                                 <xs:documentation
xml:lang="el">Εικόνα</xs:documentation>
                             </xs:annotation>
                          </xs:element>
                          <xs:element name="body"
                             minOccurs="0"
                             maxOccurs="1"
                             type="xs:string"
                             bccmsmodel:description="Event Text Body"
                             bccmsmodel:stringFormat="RichText">
                             <xs:annotation>
                                 <xs:documentation xml:lang="en">Event
Text Body</xs:documentation>
                                 <xs:documentation xml:lang="el">Κείμενο
Εκδήλωσης</xs:documentation>
                             </xs:annotation>
                          </xs:element>
                          <xs:element name="location"
                             minOccurs="0"
                             maxOccurs="1"
                             type="bccmsmodel:contentObjectType"
                             bccmsmodel:acceptedContentTypes="location"
                             bccmsmodel:description="The event location
modeled as a separate content object in
                                       repository. May be used bedides
the address.">
                             <xs:annotation>
                                 <xs:documentation xml:lang="en">Event
Location</xs:documentation>
                                 <xs:documentation
xml:lang="el">Τόπος / Χώρος Εκδήλωσης</xs:documentation>
                             </xs:annotation>
```

```
                        </xs:element>
                        <xs:element name="pricingInfo"
                            minOccurs="0"
                            maxOccurs="1"
                            type="xs:string"
                            bccmsmodel:stringFormat="RichText"
                            bccmsmodel:maxStringLength="350"
                            bccmsmodel:description="pricingInfo">
                            <xs:annotation>
                                <xs:documentation
xml:lang="en">Pricing info</xs:documentation>
                                <xs:documentation
xml:lang="el">Πληροφορίες χρέωσης</xs:documentation>
                            </xs:annotation>
                        </xs:element>
                         <xs:element name="isFreeEvent"
                            minOccurs="0"
                            maxOccurs="1"
                            type="xs:boolean"
                            bccmsmodel:description="Whether the event
is free">
                            <xs:annotation>
                                <xs:documentation xml:lang="en">Free
event</xs:documentation>
                                <xs:documentation xml:lang="el">Δωρεάν
εκδήλωση</xs:documentation>
                            </xs:annotation>
                        </xs:element>
                        <xs:element name="eventDate"
                            minOccurs="0"
                            maxOccurs="unbounded"
                            type="xs:date"
                            bccmsmodel:description="The date when the
event takes place.">
                            <xs:annotation>
                                <xs:documentation xml:lang="en">Event
Date</xs:documentation>
                                <xs:documentation xml:lang="el">Ημ/νία
Εκδήλωσης</xs:documentation>
                            </xs:annotation>
                        </xs:element>
                        <xs:element name="geoTag"
                            minOccurs="0"
                            maxOccurs="1"
                            type="geoTagType:geoTagType"
                            bccmsmodel:description="The geo location
where the event takes place.">
                            <xs:annotation>
                                <xs:documentation
xml:lang="en">GeoLocation</xs:documentation>
                                <xs:documentation
xml:lang="el">Γεωγραφικός Εντοπισμός</xs:documentation>
                            </xs:annotation>
                        </xs:element>
                        </xs:sequence>
                    </xs:extension>
                </xs:complexContent>
            </xs:complexType>
        </xs:element>
</xs:schema>
```

ΚΕΦΑΛΑΙΟ 6: ΕΦΑΡΜΟΓΗ DATABLOG

Στην υλοποίηση της εφαρμογής που ονομάζουμε Datablog, υλοποιούμε το εξής παράδειγμα. Με τον parser σε java που κατασκευάσαμε, κατεβάζουμε ιστοσελίδες με όλα τα links που ακολουθούν. Σε κάθε σελίδα, δημιουργούμε ένα νέο XSD, στο παραπάνω πρότυπο, με μία κοινή βάση. Σε όλα τα επιπλέον πεδία που διαβάζουμε, δημιουργούμε ένα πρόσθετο πεδίο και αποθηκεύουμε τη σελίδα στο repository astroboa που είδαμε στο 4ο και 5ο κεφάλαιο.

Για το παράδειγμα μας, θα κάνουμε parsing, μόνο το περιεχόμενο, τα flashobjects, google adsense, καθώς τις εικόνες με τα links που υπάρχουν. Για την υλοποίηση και το παράδειγμα χρήσης του λογισμικού, εκτελέσαμε τον αλγόριθμο σε περισσότερα από 1000 sites/blogs.

Στο τέλος της υλοποίησης μας ενδιαφέρει να βρούμε σε κάθε εκτέλεση μια μορφή σχήματος που να μπορεί να αποθηκεύει το σύνολο από τα δεδομένα που υπάρχουν στο web. Αυτό το σχήμα παράγεται αυτόματα από το εργαλείο που υλοποιήσαμε (¨datablog").

Μέσα από την consola, μπορούμε πλέον να αναζητήσουμε το περιεχόμενο με περισσότερους τρόπους καθώς και να αποκτήσουμε τη δυνατότητα να αναπαράγουμε το περιεχόμενο σε πολλές μορφές, αλλά με βασικό χαρακτηριστικό την ίδια τη δομή της κάθε ιστοσελίδας, που έχουμε αποθηκεύσει το περιεχόμενο.

ΚΕΦΑΛΑΙΟ 7: ΕΠΙΛΟΓΟΣ

Στα κεφάλαια που προηγήθηκαν έγινε αναφορά στην αξιοποίηση ενός συγκεκριμένου τρόπου αποθήκευσης δεδομένων που μπορεί να παράγει και να αποθηκεύσει η μηχανή κατά τη διάδρασή του με μια ιστοσελίδα. Είδαμε

έναν διαφορετικό τρόπο αποθήκευσης και εκμετάλλευσης των δεδομένων, πέρα από τις συνηθισμένες δομές που έχουμε γνωρίσει από τα μέχρι τώρα εργαλεία εξαγωγής δεδομένων.

Τέλος, αναφέρθηκαν και κάποια προβλήματα που μπορούν να διορθωθούν και που προκύπτουν με την καταγραφή και εκμετάλλευση των δεδομένων αυτών και τη μετέπειτα κατηγοριοποίηση, ενώ έγινε και μια συνοπτική παρουσίαση του εργαλείου Astroboa.

ΠΑΡΑΡΤΗΜΑ:

ΚΩΔΙΚΑΣ ΕΦΑΡΜΟΓΩΝ

Parser.java

```
package org.htmlparser;

import java.io.Serializable;
import java.net.HttpURLConnection;
import java.net.URLConnection;

import org.htmlparser.filters.TagNameFilter;
import org.htmlparser.filters.NodeClassFilter;
import org.htmlparser.http.ConnectionManager;
import org.htmlparser.http.ConnectionMonitor;
import org.htmlparser.http.HttpHeader;
import org.htmlparser.lexer.Lexer;
import org.htmlparser.lexer.Page;
import org.htmlparser.util.DefaultParserFeedback;
import org.htmlparser.util.IteratorImpl;
import org.htmlparser.util.NodeIterator;
import org.htmlparser.util.NodeList;
import org.htmlparser.util.ParserException;
import org.htmlparser.util.ParserFeedback;
import org.htmlparser.visitors.NodeVisitor;

public class Parser
    implements
        Serializable,
        ConnectionMonitor
{
    public static final double
    VERSION_NUMBER = 1.6
```

```
    ;

    public static final String
    VERSION_TYPE = "Release Build"
    ;

    public static final String
    VERSION_DATE = "Jun 10, 2006"
    ;

    // End of formatting

    public static final String VERSION_STRING =
            "" + VERSION_NUMBER
            + " (" + VERSION_TYPE + " " + VERSION_DATE + ")";

    protected ParserFeedback mFeedback;

    protected Lexer mLexer;

    public static final ParserFeedback DEVNULL =
        new DefaultParserFeedback (DefaultParserFeedback.QUIET);

    public static final ParserFeedback STDOUT = new
DefaultParserFeedback ();

    static
    {
        getConnectionManager ().getDefaultRequestProperties ().put (
            "User-Agent", "HTMLParser/" + getVersionNumber ());

    }

    public static String getVersion ()
    {
        return (VERSION_STRING);
    }

    /**
     * Return the version number of this parser.
     * @return A floating point number, the whole number part is the
major
     * version, and the fractional part is the minor version.
     */
    public static double getVersionNumber ()
    {
        return (VERSION_NUMBER);
    }

    /**
     * Get the connection manager all Parsers use.
     * @return The connection manager.
     * @see #setConnectionManager
     */
    public static ConnectionManager getConnectionManager ()
    {
        return (Page.getConnectionManager ());
    }

    /**
     * Set the connection manager all Parsers use.
```

```
     * @param manager The new connection manager.
     * @see #getConnectionManager
     */
    public static void setConnectionManager (ConnectionManager manager)
    {
        Page.setConnectionManager (manager);
    }

    /**
     * Creates the parser on an input string.
     * @param html The string containing HTML.
     * @param charset <em>Optional</em>. The character set encoding
that will
     * be reported by {@link #getEncoding}. If charset is
<code>null</code>
     * the default character set is used.
     * @return A parser with the <code>html</code> string as input.
     * @exception IllegalArgumentException if <code>html</code> is
<code>null</code>.
     */
    public static Parser createParser (String html, String charset)
    {
        Parser ret;

        if (null == html)
            throw new IllegalArgumentException ("html cannot be null");
        ret = new Parser (new Lexer (new Page (html, charset)));

        return (ret);
    }

    //
    // Constructors
    //

    /**
     * Zero argument constructor.
     * The parser is in a safe but useless state parsing an empty
string.
     * Set the lexer or connection using {@link #setLexer}
     * or {@link #setConnection}.
     * @see #setLexer(Lexer)
     * @see #setConnection(URLConnection)
     */
    public Parser ()
    {
        this (new Lexer (new Page ("")), DEVNULL);
    }

    /**
     * Construct a parser using the provided lexer and feedback object.
     * This would be used to create a parser for special cases where the
     * normal creation of a lexer on a URLConnection needs to be
customized.
     * @param lexer The lexer to draw characters from.
     * @param fb The object to use when information,
     * warning and error messages are produced. If <em>null</em> no
feedback
     * is provided.
     */
    public Parser (Lexer lexer, ParserFeedback fb)
```

```
    {
        setFeedback (fb);
        setLexer (lexer);
        setNodeFactory (new PrototypicalNodeFactory ());
    }

    /**
     * Constructor for custom HTTP access.
     * This would be used to create a parser for a URLConnection that
needs
     * a special setup or negotiation conditioning beyond what is
available
     * from the {@link #getConnectionManager ConnectionManager}.
     * @param connection A fully conditioned connection. The connect()
     * method will be called so it need not be connected yet.
     * @param fb The object to use for message communication.
     * @throws ParserException If the creation of the underlying Lexer
     * cannot be performed.
     */
    public Parser (URLConnection connection, ParserFeedback fb)
        throws
            ParserException
    {
        this (new Lexer (connection), fb);
    }

    /**
     * Creates a Parser object with the location of the resource (URL
or file)
     * You would typically create a DefaultHTMLParserFeedback object
and pass
     * it in.
     * @see #Parser(URLConnection,ParserFeedback)
     * @param resource Either a URL, a filename or a string of HTML.
     * The string is considered HTML if the first non-whitespace
character
     * is a &lt;. The use of a url or file is autodetected by first
attempting
     * to open the resource as a URL, if that fails it is assumed to be
a file
     * name.
     * A standard HTTP GET is performed to read the content of the URL.
     * @param feedback The HTMLParserFeedback object to use when
information,
     * warning and error messages are produced. If <em>null</em> no
feedback
     * is provided.
     * @throws ParserException If the URL is invalid.
     */
    public Parser (String resource, ParserFeedback feedback)
        throws
            ParserException
    {
        setFeedback (feedback);
        setResource (resource);
        setNodeFactory (new PrototypicalNodeFactory ());
    }

    /**
     * Creates a Parser object with the location of the resource (URL
or file).
```

```
     * A DefaultHTMLParserFeedback object is used for feedback.
     * @param resource Either HTML, a URL or a filename (autodetects).
     * @throws ParserException If the resourceLocn argument does not
resolve
     * to a valid page or file.
     * @see #Parser(String,ParserFeedback)
     */
    public Parser (String resource) throws ParserException
    {
        this (resource, STDOUT);
    }

    /**
     * Construct a parser using the provided lexer.
     * A feedback object printing to {@link #STDOUT System.out} is used.
     * This would be used to create a parser for special cases where the
     * normal creation of a lexer on a URLConnection needs to be
customized.
     * @param lexer The lexer to draw characters from.
     */
    public Parser (Lexer lexer)
    {
        this (lexer, STDOUT);
    }

    /**
     * Construct a parser using the provided URLConnection.
     * This would be used to create a parser for a URLConnection that
needs
     * a special setup or negotiation conditioning beyond what is
available
     * from the {@link #getConnectionManager ConnectionManager}.
     * A feedback object printing to {@link #STDOUT System.out} is used.
     * @see #Parser(URLConnection,ParserFeedback)
     * @param connection A fully conditioned connection. The connect()
     * method will be called so it need not be connected yet.
     * @throws ParserException If the creation of the underlying Lexer
     * cannot be performed.
     */
    public Parser (URLConnection connection) throws ParserException
    {
        this (connection, STDOUT);
    }

    //
    // Bean patterns
    //

    /**
     * Set the html, a url, or a file.
     * @param resource The resource to use.
     * @exception IllegalArgumentException if <code>resource</code> is
<code>null</code>.
     * @exception ParserException if a problem occurs in connecting.
     */
    public void setResource (String resource)
        throws
            ParserException
    {
        int length;
        boolean html;
```

```
        char ch;

        if (null == resource)
            throw new IllegalArgumentException ("resource cannot be
null");
        length = resource.length ();
        html = false;
        for (int i = 0; i < length; i++)
        {
            ch = resource.charAt (i);
            if (!Character.isWhitespace (ch))
            {
                if ('<' == ch)
                    html = true;
                break;
            }
        }
        if (html)
            setLexer (new Lexer (new Page (resource)));
        else
            setLexer (new Lexer (getConnectionManager ().openConnection
(resource)));
    }

    /**
     * Set the connection for this parser.
     * This method creates a new <code>Lexer</code> reading from the
connection.
     * @param connection A fully conditioned connection. The connect()
     * method will be called so it need not be connected yet.
     * @exception ParserException if the character set specified in the
     * HTTP header is not supported, or an i/o exception occurs
creating the
     * lexer.
     * @see #setLexer
     * @see #getConnection
     * @exception IllegalArgumentException if <code>connection</code>
is <code>null</code>.
     * @exception ParserException if a problem occurs in connecting.
     */
    public void setConnection (URLConnection connection)
        throws
            ParserException
    {
        if (null == connection)
            throw new IllegalArgumentException ("connection cannot be
null");
        setLexer (new Lexer (connection));
    }

    /**
     * Return the current connection.
     * @return The connection either created by the parser or passed
into this
     * parser via {@link #setConnection}.
     * @see #setConnection(URLConnection)
     */
    public URLConnection getConnection ()
    {
        return (getLexer ().getPage ().getConnection ());
    }
```

```
    /**
     * Set the URL for this parser.
     * This method creates a new Lexer reading from the given URL.
     * Trying to set the url to null or an empty string is a no-op.
     * @param url The new URL for the parser.
     * @throws ParserException If the url is invalid or creation of the
     * underlying Lexer cannot be performed.
     * @exception ParserException if a problem occurs in connecting.
     * @see #getURL
     */
    public void setURL (String url)
        throws
            ParserException
    {
        if ((null != url) && !"".equals (url))
            setConnection (getConnectionManager ().openConnection
(url));
    }

    /**
     * Return the current URL being parsed.
     * @return The current url. This is the URL for the current page.
     * A string passed into the constructor or set via setURL may be
altered,
     * for example, a file name may be modified to be a URL.
     * @see Page#getUrl
     * @see #setURL
     */
    public String getURL ()
    {
        return (getLexer ().getPage ().getUrl ());
    }

    /**
     * Set the encoding for the page this parser is reading from.
     * @param encoding The new character set to use.
     * @throws ParserException If the encoding change causes characters
that
     * have already been consumed to differ from the characters that
would
     * have been seen had the new encoding been in force.
     * @see org.htmlparser.util.EncodingChangeException
     * @see #getEncoding
     */
    public void setEncoding (String encoding)
        throws
            ParserException
    {
        getLexer ().getPage ().setEncoding (encoding);
    }

    /**
     * Get the encoding for the page this parser is reading from.
     * This item is set from the HTTP header but may be overridden by
meta
     * tags in the head, so this may change after the head has been
parsed.
     * @return The encoding currently in force.
     * @see #setEncoding
     */
```

```
    public String getEncoding ()
    {
        return (getLexer ().getPage ().getEncoding ());
    }

    /**
     * Set the lexer for this parser.
     * The current NodeFactory is transferred to (set on) the given
lexer,
     * since the lexer owns the node factory object.
     * It does not adjust the <code>feedback</code> object.
     * @param lexer The lexer object to use.
     * @see #setNodeFactory
     * @see #getLexer
     * @exception IllegalArgumentException if <code>lexer</code> is
<code>null</code>.
     */
    public void setLexer (Lexer lexer)
    {
        NodeFactory factory;
        String type;

        if (null == lexer)
            throw new IllegalArgumentException ("lexer cannot be null");
        // move a node factory that's been set to the new lexer
        factory = null;
        if (null != getLexer ())
            factory = getLexer ().getNodeFactory ();
        if (null != factory)
            lexer.setNodeFactory (factory);
        mLexer = lexer;
        // warn about content that's not likely text
        type = mLexer.getPage ().getContentType ();
        if (type != null && !type.startsWith ("text"))
            getFeedback ().warning (
                "URL "
                + mLexer.getPage ().getUrl ()
                + " does not contain text");
    }

    /**
     * Returns the lexer associated with the parser.
     * @return The current lexer.
     * @see #setLexer
     */
    public Lexer getLexer ()
    {
        return (mLexer);
    }

    /**
     * Get the current node factory.
     * @return The current lexer's node factory.
     * @see #setNodeFactory
     */
    public NodeFactory getNodeFactory ()
    {
        return (getLexer ().getNodeFactory ());
    }

    /**
```

```
     * Set the current node factory.
     * @param factory The new node factory for the current lexer.
     * @see #getNodeFactory
     * @exception IllegalArgumentException if <code>factory</code> is
<code>null</code>.
     */
    public void setNodeFactory (NodeFactory factory)
    {
        if (null == factory)
            throw new IllegalArgumentException ("node factory cannot be
null");
        getLexer ().setNodeFactory (factory);
    }

    /**
     * Sets the feedback object used in scanning.
     * @param fb The new feedback object to use. If this is null a
     * {@link #DEVNULL silent feedback object} is used.
     * @see #getFeedback
     */
    public void setFeedback (ParserFeedback fb)
    {
        if (null == fb)
            mFeedback = DEVNULL;
        else
            mFeedback = fb;
    }

    /**
     * Returns the current feedback object.
     * @return The feedback object currently being used.
     * @see #setFeedback
     */
    public ParserFeedback getFeedback()
    {
        return (mFeedback);
    }

    //
    // Public methods
    //

    /**
     * Reset the parser to start from the beginning again.
     * This assumes support for a reset from the underlying
     * {@link org.htmlparser.lexer.Source} object.
     * <p>This is cheaper (in terms of time) than resetting the URL,
i.e.
     * <pre>
     * parser.setURL (parser.getURL ());
     * </pre>
     * because the page is not refetched from the internet.
     * <em>Note: the nodes returned on the second parse are new
     * nodes and not the same nodes returned on the first parse. If you
     * want the same nodes for re-use, collect them in a NodeList with
     * {@link #parse(NodeFilter) parse(null)} and operate on the
NodeList.</em>
     */
    public void reset ()
    {
        getLexer ().reset ();
```

```
    }

    /**
     * Returns an iterator (enumeration) over the html nodes.
     * {@link org.htmlparser.nodes Nodes} can be of three main types:
     * <ul>
     * <li>{@link org.htmlparser.nodes.TagNode TagNode}</li>
     * <li>{@link org.htmlparser.nodes.TextNode TextNode}</li>
     * <li>{@link org.htmlparser.nodes.RemarkNode RemarkNode}</li>
     * </ul>
     * In general, when parsing with an iterator or processing a
NodeList,
     * you will need to use recursion. For example:
     * <code>
     * <pre>
     * void processMyNodes (Node node)
     * {
     *     if (node instanceof TextNode)
     *     {
     *         // downcast to TextNode
     *         TextNode text = (TextNode)node;
     *         // do whatever processing you want with the text
     *         System.out.println (text.getText ());
     *     }
     *     if (node instanceof RemarkNode)
     *     {
     *         // downcast to RemarkNode
     *         RemarkNode remark = (RemarkNode)node;
     *         // do whatever processing you want with the comment
     *     }
     *     else if (node instanceof TagNode)
     *     {
     *         // downcast to TagNode
     *         TagNode tag = (TagNode)node;
     *         // do whatever processing you want with the tag itself
     *         // ...
     *         // process recursively (nodes within nodes) via
getChildren()
     *         NodeList nl = tag.getChildren ();
     *         if (null != nl)
     *             for (NodeIterator i = nl.elements ();
i.hasMoreElements (); )
     *                 processMyNodes (i.nextNode ());
     *     }
     * }
     *
     * Parser parser = new Parser ("http://www.yahoo.com");
     * for (NodeIterator i = parser.elements (); i.hasMoreElements (); )
     *     processMyNodes (i.nextNode ());
     * </pre>
     * </code>
     * @throws ParserException If a parsing error occurs.
     * @return An iterator over the top level nodes (usually {@.html
<html>}).
     */
    public NodeIterator elements () throws ParserException
    {
        return (new IteratorImpl (getLexer (), getFeedback ()));
    }

    /**
```

```
     * Parse the given resource, using the filter provided.
     * This can be used to extract information from specific nodes.
     * When used with a <code>null</code> filter it returns an
     * entire page which can then be modified and converted back to HTML
     * (Note: the synthesis use-case is not handled very well; the
parser
     * is more often used to extract information from a web page).
     * <p>For example, to replace the entire contents of the HEAD with a
     * single TITLE tag you could do this:
     * <pre>
     * NodeList nl = parser.parse (null); // here is your two node list
     * NodeList heads = nl.extractAllNodesThatMatch (new TagNameFilter
("HEAD"))
     * if (heads.size () > 0) // there may not be a HEAD tag
     * {
     *     Head head = heads.elementAt (0); // there should be only one
     *     head.removeAll (); // clean out the contents
     *     Tag title = new TitleTag ();
     *     title.setTagName ("title");
     *     title.setChildren (new NodeList (new TextNode ("The New
Title")));
     *     Tag title_end = new TitleTag ();
     *     title_end.setTagName ("/title");
     *     title.setEndTag (title_end);
     *     head.add (title);
     * }
     * System.out.println (nl.toHtml ()); // output the modified HTML
     * </pre>
     * @return The list of matching nodes (for a <code>null</code>
     * filter this is all the top level nodes).
     * @param filter The filter to apply to the parsed nodes,
     * or <code>null</code> to retrieve all the top level nodes.
     * @throws ParserException If a parsing error occurs.
     */
    public NodeList parse (NodeFilter filter) throws ParserException
    {
        NodeIterator e;
        Node node;
        NodeList ret;

        ret = new NodeList ();
        for (e = elements (); e.hasMoreNodes (); )
        {
            node = e.nextNode ();
            if (null != filter)
                node.collectInto (ret, filter);
            else
                ret.add (node);
        }

        return (ret);
    }

    /**
     * Apply the given visitor to the current page.
     * The visitor is passed to the <code>accept()</code> method of
each node
     * in the page in a depth first traversal. The visitor
     * <code>beginParsing()</code> method is called prior to processing
the
```

```
     * page and <code>finishedParsing()</code> is called after the
processing.
     * @param visitor The visitor to visit all nodes with.
     * @throws ParserException If a parse error occurs while traversing
     * the page with the visitor.
     */
    public void visitAllNodesWith (NodeVisitor visitor) throws
ParserException
    {
        Node node;
        visitor.beginParsing();
        for (NodeIterator e = elements(); e.hasMoreNodes(); )
        {
            node = e.nextNode();
            node.accept(visitor);
        }
        visitor.finishedParsing();
    }

    /**
     * Initializes the parser with the given input HTML String.
     * @param inputHTML the input HTML that is to be parsed.
     * @throws ParserException If a error occurs in setting up the
     * underlying Lexer.
     * @exception IllegalArgumentException if <code>inputHTML</code> is
<code>null</code>.
     */
    public void setInputHTML (String inputHTML)
        throws
            ParserException
    {
        if (null == inputHTML)
            throw new IllegalArgumentException ("html cannot be null");
        if (!"".equals (inputHTML))
            setLexer (new Lexer (new Page (inputHTML)));
    }

    /**
     * Extract all nodes matching the given filter.
     * @see Node#collectInto(NodeList, NodeFilter)
     * @param filter The filter to be applied to the nodes.
     * @throws ParserException If a parse error occurs.
     * @return A list of nodes matching the filter criteria,
     * i.e. for which the filter's accept method
     * returned <code>true</code>.
     */
    public NodeList extractAllNodesThatMatch (NodeFilter filter)
        throws
            ParserException
    {
        NodeIterator e;
        NodeList ret;

        ret = new NodeList ();
        for (e = elements (); e.hasMoreNodes (); )
            e.nextNode ().collectInto (ret, filter);

        return (ret);
    }

    //
```

```
    // ConnectionMonitor interface
    //

    /**
     * Called just prior to calling connect.
     * Part of the ConnectionMonitor interface, this implementation just
     * sends the request header to the feedback object if any.
     * @param connection The connection which is about to be connected.
     * @throws ParserException <em>Not used</em>
     * @see ConnectionMonitor#preConnect
     */
    public void preConnect (HttpURLConnection connection)
        throws
            ParserException
    {
        getFeedback ().info (HttpHeader.getRequestHeader (connection));
    }

    public void postConnect (HttpURLConnection connection)
        throws
            ParserException
    {
        getFeedback ().info (HttpHeader.getResponseHeader (connection));
    }

    /**
     * The main program, which can be executed from the command line.
     * @param args A URL or file name to parse, and an optional tag
name to be
     * used as a filter.
     */
    public static void main (String [] args)
    {
        Parser parser;
        NodeFilter filter;

        if (args.length < 1 || args[0].equals ("-help"))
        {
            System.out.println ("HTML Parser v" + getVersion () + "\n");
            System.out.println ();
            System.out.println ("Syntax : java -jar htmlparser.jar"
                    + " <file/page> [type]");
            System.out.println ("   <file/page> the URL or file to be
parsed");
            System.out.println ("   type the node type, for example:");
            System.out.println ("     A - Show only the link tags");
            System.out.println ("     IMG - Show only the image tags");
            System.out.println ("     TITLE - Show only the title tag");
            System.out.println ();
            System.out.println ("Example : java -jar htmlparser.jar"
                    + " http://www.yahoo.com");
            System.out.println ();
        }
        else
            try
            {
                parser = new Parser ();
                if (1 < args.length)
                    filter = new TagNameFilter (args[1]);
                else
                {
```

```
                    filter = null;
                    // for a simple dump, use more verbose settings
                    parser.setFeedback (Parser.STDOUT);
                    getConnectionManager ().setMonitor (parser);
                }
                getConnectionManager ().setRedirectionProcessingEnabled
(true);
                getConnectionManager ().setCookieProcessingEnabled
(true);
                parser.setResource (args[0]);
                System.out.println (parser.parse (filter));
            }
            catch (ParserException e)
            {
                e.printStackTrace ();
            }
    }
}
```

Prototypical.java

```
package org.htmlparser;

import java.io.Serializable;
import java.util.Hashtable;
import java.util.Locale;
import java.util.Map;
import java.util.Set;
import java.util.Vector;

import org.htmlparser.lexer.Page;
import org.htmlparser.nodes.TextNode;
import org.htmlparser.nodes.RemarkNode;
import org.htmlparser.nodes.TagNode;
import org.htmlparser.tags.AppletTag;
import org.htmlparser.tags.BaseHrefTag;
import org.htmlparser.tags.BodyTag;
import org.htmlparser.tags.Bullet;
import org.htmlparser.tags.BulletList;
import org.htmlparser.tags.DefinitionList;
import org.htmlparser.tags.DefinitionListBullet;
import org.htmlparser.tags.Div;
import org.htmlparser.tags.DoctypeTag;
import org.htmlparser.tags.FormTag;
import org.htmlparser.tags.FrameSetTag;
import org.htmlparser.tags.FrameTag;
import org.htmlparser.tags.HeadingTag;
import org.htmlparser.tags.HeadTag;
import org.htmlparser.tags.Html;
import org.htmlparser.tags.ImageTag;
import org.htmlparser.tags.InputTag;
import org.htmlparser.tags.JspTag;
import org.htmlparser.tags.LabelTag;
import org.htmlparser.tags.LinkTag;
import org.htmlparser.tags.MetaTag;
import org.htmlparser.tags.ObjectTag;
import org.htmlparser.tags.OptionTag;
import org.htmlparser.tags.ParagraphTag;
import org.htmlparser.tags.ProcessingInstructionTag;
```

```
import org.htmlparser.tags.ScriptTag;
import org.htmlparser.tags.SelectTag;
import org.htmlparser.tags.Span;
import org.htmlparser.tags.StyleTag;
import org.htmlparser.tags.TableColumn;
import org.htmlparser.tags.TableHeader;
import org.htmlparser.tags.TableRow;
import org.htmlparser.tags.TableTag;
import org.htmlparser.tags.TextareaTag;
import org.htmlparser.tags.TitleTag;

/**
 * A node factory based on the prototype pattern.
 * This factory uses the prototype pattern to generate new nodes.
 * These are cloned as needed to form new {@link Text}, {@link Remark}
and
 * {@link Tag} nodes.
 * <p>Text and remark nodes are generated from prototypes accessed
 * via the {@link #setTextPrototype(Text) textPrototype} and
 * {@link #setRemarkPrototype(Remark) remarkPrototype} properties
respectively.
 * Tag nodes are generated as follows:
 * <p>Prototype tags, in the form of undifferentiated tags, are held in
a hash
 * table. On a request for a tag, the attributes are examined for the
name
 * of the tag to be created. If a prototype of that name has been
registered
 * (exists in the hash table), it is cloned and the clone is given the
 * characteristics ({@link Attribute Attributes}, start and end
position)
 * of the requested tag.</p>
 * <p>In the case that no tag has been registered under that name,
 * a generic tag is created from the prototype acessed via the
 * {@link #setTagPrototype(Tag) tagPrototype} property.</p>
 * <p>The hash table of registered tags can be automatically populated
with
 * all the known tags from the {@link org.htmlparser.tags} package when
 * the factory is constructed, or it can start out empty and be
populated
 * explicitly.</p>
 * <p>Here is an example of how to override all text issued from
 * {@link org.htmlparser.nodes.TextNode#toPlainTextString()
 * Text.toPlainTextString()},
 * in this case decoding (converting character references),
 * which illustrates the use of setting the text prototype:
 * <pre>
 * PrototypicalNodeFactory factory = new PrototypicalNodeFactory ();
 * factory.setTextPrototype (
 *     // create a inner class that is a subclass of TextNode
 *     new TextNode () {
 *         public String toPlainTextString()
 *         {
 *             String original = super.toPlainTextString ();
 *             return (org.htmlparser.util.Translate.decode (original));
 *         }
 *     });
 * Parser parser = new Parser ();
 * parser.setNodeFactory (factory);
 * </pre></p>
 * <p>Here is an example of using a custom link tag, in this case just
```

```
 * printing the URL, which illustrates registering a tag:
 * <pre>
 *
 * class PrintingLinkTag extends LinkTag
 * {
 *     public void doSemanticAction ()
 *         throws
 *             ParserException
 *     {
 *         System.out.println (getLink ());
 *     }
 * }
 * PrototypicalNodeFactory factory = new PrototypicalNodeFactory ();
 * factory.registerTag (new PrintingLinkTag ());
 * Parser parser = new Parser ();
 * parser.setNodeFactory (factory);
 * </pre></p>
 */
public class PrototypicalNodeFactory
    implements
        Serializable,
        NodeFactory
{
    /**
     * The prototypical text node.
     */
    protected Text mText;

    /**
     * The prototypical remark node.
     */
    protected Remark mRemark;

    /**
     * The prototypical tag node.
     */
    protected Tag mTag;

    /**
     * The list of tags to return.
     * The list is keyed by tag name.
     */
    protected Map mBlastocyst;

    /**
     * Create a new factory with all tags registered.
     * Equivalent to
     * {@link #PrototypicalNodeFactory()
PrototypicalNodeFactory(false)}.
     */
    public PrototypicalNodeFactory ()
    {
        this (false);
    }

    /**
     * Create a new factory.
     * @param empty If <code>true</code>, creates an empty factory,
     * otherwise create a new factory with all tags registered.
     */
    public PrototypicalNodeFactory (boolean empty)
```

```
    {
        clear ();
        mText = new TextNode (null, 0, 0);
        mRemark = new RemarkNode (null, 0, 0);
        mTag = new TagNode (null, 0, 0, null);
        if (!empty)
            registerTags ();
    }

    /**
     * Create a new factory with the given tag as the only registered
tag.
     * @param tag The single tag to register in the otherwise empty
factory.
     */
    public PrototypicalNodeFactory (Tag tag)
    {
        this (true);
        registerTag (tag);
    }

    /**
     * Create a new factory with the given tags registered.
     * @param tags The tags to register in the otherwise empty factory.
     */
    public PrototypicalNodeFactory (Tag[] tags)
    {
        this (true);
        for (int i = 0; i < tags.length; i++)
            registerTag (tags[i]);
    }

    /**
     * Adds a tag to the registry.
     * @param id The name under which to register the tag.
     * <strong>For proper operation, the id should be uppercase so it
     * will be matched by a Map lookup.</strong>
     * @param tag The tag to be returned from a {@link #createTagNode}
call.
     * @return The tag previously registered with that id if any,
     * or <code>null</code> if none.
     */
    public Tag put (String id, Tag tag)
    {
        return ((Tag)mBlastocyst.put (id, tag));
    }

    /**
     * Gets a tag from the registry.
     * @param id The name of the tag to return.
     * @return The tag registered under the <code>id</code> name,
     * or <code>null</code> if none.
     */
    public Tag get (String id)
    {
        return ((Tag)mBlastocyst.get (id));
    }

    /**
     * Remove a tag from the registry.
     * @param id The name of the tag to remove.
```

```
     * @return The tag that was registered with that <code>id</code>,
     * or <code>null</code> if none.
     */
    public Tag remove (String id)
    {
        return ((Tag)mBlastocyst.remove (id));
    }

    /**
     * Clean out the registry.
     */
    public void clear ()
    {
        mBlastocyst = new Hashtable ();
    }

    /**
     * Get the list of tag names.
     * @return The names of the tags currently registered.
     */
    public Set getTagNames ()
    {
        return (mBlastocyst.keySet ());
    }

    /**
     * Register a tag.
     * Registers the given tag under every {@link Tag#getIds() id} that
the
     * tag has (i.e. all names returned by {@link Tag#getIds()
tag.getIds()}.
     * <p><strong>For proper operation, the ids are converted to
uppercase so
     * they will be matched by a Map lookup.</strong>
     * @param tag The tag to register.
     */
    public void registerTag (Tag tag)
    {
        String[] ids;

        ids = tag.getIds ();
        for (int i = 0; i < ids.length; i++)
            put (ids[i].toUpperCase (Locale.ENGLISH), tag);
    }

    /**
     * Unregister a tag.
     * Unregisters the given tag from every {@link Tag#getIds() id} the
tag has.
     * <p><strong>The ids are converted to uppercase to undo the
operation
     * of registerTag.</strong>
     * @param tag The tag to unregister.
     */
    public void unregisterTag (Tag tag)
    {
        String[] ids;

        ids = tag.getIds ();
        for (int i = 0; i < ids.length; i++)
            remove (ids[i].toUpperCase (Locale.ENGLISH));
```

```
    }

    /**
     * Register all known tags in the tag package.
     * Registers tags from the {@link org.htmlparser.tags tag package}
by
     * calling {@link #registerTag(Tag) registerTag()}.
     * @return 'this' nodefactory as a convenience.
     */
    public PrototypicalNodeFactory registerTags ()
    {
        registerTag (new AppletTag ());
        registerTag (new BaseHrefTag ());
        registerTag (new Bullet ());
        registerTag (new BulletList ());
        registerTag (new DefinitionList ());
        registerTag (new DefinitionListBullet ());
        registerTag (new DoctypeTag ());
        registerTag (new FormTag ());
        registerTag (new FrameSetTag ());
        registerTag (new FrameTag ());
        registerTag (new HeadingTag ());
        registerTag (new ImageTag ());
        registerTag (new InputTag ());
        registerTag (new JspTag ());
        registerTag (new LabelTag ());
        registerTag (new LinkTag ());
        registerTag (new MetaTag ());
        registerTag (new ObjectTag ());
        registerTag (new OptionTag ());
        registerTag (new ParagraphTag ());
        registerTag (new ProcessingInstructionTag ());
        registerTag (new ScriptTag ());
        registerTag (new SelectTag ());
        registerTag (new StyleTag ());
        registerTag (new TableColumn ());
        registerTag (new TableHeader ());
        registerTag (new TableRow ());
        registerTag (new TableTag ());
        registerTag (new TextareaTag ());
        registerTag (new TitleTag ());
        registerTag (new Div ());
        registerTag (new Span ());
        registerTag (new BodyTag ());
        registerTag (new HeadTag ());
        registerTag (new Html ());


        return (this);
    }

    /**
     * Get the object that is cloned to generate text nodes.
     * @return The prototype for {@link Text} nodes.
     * @see #setTextPrototype
     */
    public Text getTextPrototype ()
    {
        return (mText);
    }
```

```
    /**
     * Set the object to be used to generate text nodes.
     * @param text The prototype for {@link Text} nodes.
     * If <code>null</code> the prototype is set to the default
     * ({@link TextNode}).
     * @see #getTextPrototype
     */
    public void setTextPrototype (Text text)
    {
        if (null == text)
            mText = new TextNode (null, 0, 0);
        else
            mText = text;
    }

    /**
     * Get the object that is cloned to generate remark nodes.
     * @return The prototype for {@link Remark} nodes.
     * @see #setRemarkPrototype
     */
    public Remark getRemarkPrototype ()
    {
        return (mRemark);
    }

    /**
     * Set the object to be used to generate remark nodes.
     * @param remark The prototype for {@link Remark} nodes.
     * If <code>null</code> the prototype is set to the default
     * ({@link RemarkNode}).
     * @see #getRemarkPrototype
     */
    public void setRemarkPrototype (Remark remark)
    {
        if (null == remark)
            mRemark = new RemarkNode (null, 0, 0);
        else
            mRemark = remark;
    }

    /**
     * Get the object that is cloned to generate tag nodes.
     * Clones of this object are returned from {@link #createTagNode}
when no
     * specific tag is found in the list of registered tags.
     * @return The prototype for {@link Tag} nodes.
     * @see #setTagPrototype
     */
    public Tag getTagPrototype ()
    {
        return (mTag);
    }

    /**
     * Set the object to be used to generate tag nodes.
     * Clones of this object are returned from {@link #createTagNode}
when no
     * specific tag is found in the list of registered tags.
     * @param tag The prototype for {@link Tag} nodes.
     * If <code>null</code> the prototype is set to the default
     * ({@link TagNode}).
```

```
     * @see #getTagPrototype
     */
    public void setTagPrototype (Tag tag)
    {
        if (null == tag)
            mTag = new TagNode (null, 0, 0, null);
        else
            mTag = tag;
    }

    //
    // NodeFactory interface
    //

    /**
     * Create a new string node.
     * @param page The page the node is on.
     * @param start The beginning position of the string.
     * @param end The ending position of the string.
     * @return A text node comprising the indicated characters from the
page.
     */
    public Text createStringNode (Page page, int start, int end)
    {
        Text ret;

        try
        {
            ret = (Text)(getTextPrototype ().clone ());
            ret.setPage (page);
            ret.setStartPosition (start);
            ret.setEndPosition (end);
        }
        catch (CloneNotSupportedException cnse)
        {
            ret = new TextNode (page, start, end);
        }

        return (ret);
    }

    /**
     * Create a new remark node.
     * @param page The page the node is on.
     * @param start The beginning position of the remark.
     * @param end The ending positiong of the remark.
     * @return A remark node comprising the indicated characters from
the page.
     */
    public Remark createRemarkNode (Page page, int start, int end)
    {
        Remark ret;

        try
        {
            ret = (Remark)(getRemarkPrototype ().clone ());
            ret.setPage (page);
            ret.setStartPosition (start);
            ret.setEndPosition (end);
        }
        catch (CloneNotSupportedException cnse)
```

```
        {
            ret = new RemarkNode (page, start, end);
        }

        return (ret);
    }

    /**
     * Create a new tag node.
     * Note that the attributes vector contains at least one element,
     * which is the tag name (standalone attribute) at position zero.
     * This can be used to decide which type of node to create, or
     * gate other processing that may be appropriate.
     * @param page The page the node is on.
     * @param start The beginning position of the tag.
     * @param end The ending positiong of the tag.
     * @param attributes The attributes contained in this tag.
     * @return A tag node comprising the indicated characters from the
page.
     */
    public Tag createTagNode (Page page, int start, int end, Vector
attributes)
    {
        Attribute attribute;
        String id;
        Tag prototype;
        Tag ret;

        ret = null;

        if (0 != attributes.size ())
        {
            attribute = (Attribute)attributes.elementAt (0);
            id = attribute.getName ();
            if (null != id)
            {
                try
                {
                    id = id.toUpperCase (Locale.ENGLISH);
                    if (!id.startsWith ("/"))
                    {
                        if (id.endsWith ("/"))
                            id = id.substring (0, id.length () - 1);
                        prototype = (Tag)mBlastocyst.get (id);
                        if (null != prototype)
                        {
                            ret = (Tag)prototype.clone ();
                            ret.setPage (page);
                            ret.setStartPosition (start);
                            ret.setEndPosition (end);
                            ret.setAttributesEx (attributes);
                        }
                    }
                }
                catch (CloneNotSupportedException cnse)
                {
                    // default to creating a generic one
                }
            }
        }
        if (null == ret)
```

```
        {   // generate a generic node
            try
            {
                ret = (Tag)getTagPrototype ().clone ();
                ret.setPage (page);
                ret.setStartPosition (start);
                ret.setEndPosition (end);
                ret.setAttributesEx (attributes);
            }
            catch (CloneNotSupportedException cnse)
            {
                ret = new TagNode (page, start, end, attributes);
            }
        }

        return (ret);
    }
}
```

Βιβλιογραφία/Πηγές

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. http://www.astroboa.org/
2. Antoniou, G., F. van Harmelen. 2003, Web Ontology Language: OWL, Springer-Verlag
3. Antoniou, G., M. Baldoni, C. Baroglio, V. Patti, R. Baumgartner, T. Eiter, M. Herzog, R. Schindlauer, H. Tompits, F. Bry, S. Schaffert, N. Henze, W. May. 2004, Reasoning Methods for Personalization on the Semantic Web. *Annals of Mathematics, Computing and Teleinformatics (AMCT) 1(2),* Institute for Informatics, Georg-August-Univesität Göttingen, pp. 1-24
4. Antoniou, G., E. Franconi, F. van Harmelen. Introduction to Semantic Web Ontology Languages
5. Baader, F., I. Horrocks and U. Sattler. 2005, Description Logics as Ontology Languages for the Semantic Web, Springer
6. BBC Greek,
7. Berners-Lee, T., J. Handler, and O. Lassila. 2001, The Semantic Web, *Scientific American*
8. Bonett, M. Personalization of Web Services: Oportunities and Challenges. *Ariadne,* available at http://ariadne.ac.uk/issue28/personalization/
9. CNN, http://www.cnn.com/

10. Decker, S., S. Malnik, F. van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann, I. Horrocks. 2000, The Semantic Web: The Roles of XML and RDF. In *IEEE Internet Computing*
11. Fernandez-Garcia, N., L. Sanchez-Fernandez. 2004, Building an Ontology for NEWS Applications. *In proceedings of the 3rd International Semantic Web Conference ISWC2004,* http://iswc2004.semanticweb.org
12. Henze, N., Kriesell, M. 2004, Personalization Functionality for the Semantic Web: *Architectural Outline and First Sample Implementations,* Distributed Systems Institute – Knowledge Based Systems
13. Horrocks, I., P. F. Patel-Schneider, F. van Harmelen. 2002, Reviewing the Desing of DAML+OIL: *An Ontology Language for the Semantic Web,* American Association Intelligence, www.aaai.org
14. IEEE Internet Computing
http://computer.org/internet
15. Information Interchange Model (*IIM*). Available at:
http://www.iptc.org/IIM/
16. IPTC NewsML Web. Available at: http://www.newsml.org
17. Kalfoglou, Y., J. Domingue, E. Motta, M. Vargas-Vera, S. Buckingham Shum. MyPlanet: *An Ontology-driven Web-based personilised news service,* Knowledge Media Institute (KMi)
18. Kinecta: Industry Standards: PRISM
http://www.kinecta.com/resources/prism.html
19. Kravatz, H. 2000, Desining Web Personalization Features
20. Liu, L., W. Han, D. Buttler, C. Pu, W. Tang.1999, An XML-based Wrapper Generator for Web Information Extraction, *in Proceedings of ACM SIGMOD International Conference on Management of Data*
21. Markellou, P., M. Rigou, S. Sirmakessis & A. Tsakalidis. 2004, Personalization in the Semantic Web Era: *a glance ahead*. *WIT Press*
22. Manola, F. 2002, The Semantic Web and the Role of Information Systems Research, *The MITRE Corporation.* Available at the Workshop Web site as:
http://lsdis.cs.uga.edu/SemNSF/Manoloa-Position.doc
23. McIlraith, Sheila A., T.C. Son, H. Zeng. 2001, Semantic Web Services
24. Muslea, I. 1999, Extraction Patterns for Information Extraction Tasks: A Survey. In *IEEE Intelligent Systems,* available at
http://www.ai.sri.com/~muslea/PS/ml4ie-aaai99.pdf

25. Neptuno Project (The). *Networked Semantic Team (NETS),* http://nets.ii.uam.es/neptuno
26. NEWS (*News Engine Web Services*) Home. Available at: http://www.news-project.com
27. NIFT: *News Industry Text Format.* Available at: http://www.nift.org
28. Noy, N. F., D. L. McGuiness. 2001, Ontology Development 101: *A guide to Creating Your First Ontology,* Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880
29. OWL, Web Ontology Language, WC3 Recommendation, Feb. 2004 http://www.w3.org/TR/owl-features/
30. Potock, T.E., M.T. Elmore, J.W. Reed, and N.F. Samatova. 2002, An Ontology-based HTML to XML Conversion Using Intelligent Agents, *In Proceedings of the 35th Hawaii International Conference on System Sciences (HICSS-35'02)*
31. PRISM (Publishing Requirements for Industry Standard Metadata) http://www.prismstandard.org/
32. RDF *Vocabulary Description Language1.0: RDF Schema.* Available at: http://www.w3.org/TR/rdf-schema
33. Sanfilippo, A., A. Bernardi, L. van Elst, L. S. Fernandez, M. Sintek. 2003, Position Paper: *Integrating Ontologies for Semantic Web Applications, ENABLER/ELSNET Workshop on International Roadmap for Languages Resources,* Paris
34. Shahabi, C., Yi-Shin Chen. Web Information Personalization: *Challenges and Approaches*
35. Shih-Hung, W., Tzing-Han Tsai and Wen-Lian Hsu. 2003, Domain Event Extraction and Representation with Domain Ontology, *Workshop on Information Intergration on the Web (IIWeb-03)*
36. Metadata: Subject Reference System and NewsML topicsets. Available at: http://www.iptc.org/metadata/
37. UMBC eBiquity http://ebiquity.umbc.edu/us