ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Εφαρμογή αντικειμενο-σχεσιακής απεικόνισης (Object Relational Mapping) στο περιβάλλον διαχείρισης ιστοσελίδων μαθημάτων του Ε.Μ.Π.

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΨΑΡΟΥΔΑΚΗ ΗΡΑΚΛΗ

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2010

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Εφαρμογή αντικειμενο-σχεσιακής απεικόνισης (Object Relational Mapping) στο περιβάλλον διαχείρισης ιστοσελίδων μαθημάτων του Ε.Μ.Π.

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΨΑΡΟΥΔΑΚΗ ΗΡΑΚΛΗ

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 8η Ιουλίου 2010.

.....	.....	.....
Τιμολέων Σελλής Καθηγητής Ε.Μ.Π.	Ιωάννης Βασιλείου Καθηγητής Ε.Μ.Π.	Νεκτάριος Κοζύρης Αναπλ. Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2010

.....

ΨΑΡΟΥΔΑΚΗΣ ΗΡΑΚΛΗΣ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Ο σκοπός της διπλωματικής εργασίας ήταν η εφαρμογή μεθόδων αντικειμενο-σχεσιακής απεικόνισης (Object-Relational Mapping) στο περιβάλλον διαχείρισης ιστοσελίδων μαθημάτων NCCMS (.NET Course Content Management System) που έχει αναπτυχθεί από το Εργαστήριο Συστημάτων Βάσεων Γνώσεων και Δεδομένων και φιλοξενεί μεγάλο αριθμό ιστοσελίδων μαθημάτων του Ε.Μ.Π.

Η εργασία αποτελείται από τρία τμήματα: (α) Καταγραφή του μοντέλου δεδομένων και των λειτουργιών του περιβάλλοντος NCCMS και του υποκείμενου περιβάλλοντος διαχείρισης περιεχομένου NCommet (.NET Content Management Methodologies, Environments and Tools) του ΕΣΒΓΔ. (β) Περιγραφή των βασικών αρχών της αντικειμενο-σχεσιακής απεικόνισης για την αποθήκευση των δεδομένων μιας αντικειμενοστραφούς εφαρμογής και παρουσίαση των σημαντικότερων εργαλείων αντικειμενο-σχεσιακής απεικόνισης. (γ) Σχεδίαση και υλοποίηση της αντικειμενο-σχεσιακής απεικόνισης του περιβάλλοντος NCCMS με τη χρήση του εργαλείου NHibernate.

Λέξεις Κλειδιά: διατήρηση δεδομένων, αντικειμενοστραφής προγραμματισμός, C#, .NET Framework, αντικειμενο-σχεσιακή απεικόνιση, ORM, NHibernate, HQL, NCCMS, NCommet, δενδρικές δομές, XML σειριακοποίηση

Abstract

The scope of this thesis was the application of Object-Relational Mapping techniques on the web environment NCCMS (.NET Course Content Management System). NCCMS is used to administrate web pages for courses. It has been developed by the Knowledge and Database Systems Laboratory and it hosts a large number of courses from all over NTUA.

The work of the thesis was composed by three main tasks: (a) The recording of the data model and the operations of NCCMS and of the underlying content management environment NCommet (.NET Content Management Methodologies, Environments and Tools), developed also by DBLab. (b) The description of the main principles of Object-Relational Mapping for the persistence of data of object-oriented applications and the presentation of the most important ORM tools. (c) The design and implementation of Object-Relational Mapping for the NCCMS environment, with the use of the NHibernate tool.

Keywords: data persistence, object-oriented programming, C#, .NET Framework, object-relational mapping, ORM, NHibernate, HQL, NCCMS, NCommet, tree structures, XML serialization

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον καθηγητή μου, κύριο Τίμο Σελλή, για την εποικοδομητική συνεργασία μας, την υποστήριξη του και την ευκαιρία που μου έδωσε να ασχοληθώ με ένα τόσο ενδιαφέρον έργο λογισμικού.

Επίσης, θα ήθελα να ευχαριστήσω τον κύριο Κυριάκο Διακονικολάου ο οποίος με βοήθησε και με καθοδήγησε στην εκπόνηση της παρούσας διπλωματικής εργασίας.

Ένα τελευταίο και μεγάλο ευχαριστώ στους γονείς μου Εύα και Βαγγέλη που με στηρίζουν καθ' όλη τη διάρκεια της ζωής μου μέχρι και σήμερα. Ευελπιστώ ότι μέσα απ' όλη μου την πορεία θα μπορέσω κάποτε να τους ανταποδώσω αυτά που μου έχουν προσφέρει.

Ιούλιος 2010,
Ηρακλής Ψαρουδάκης

Πίνακας περιεχομένων

1

Εισαγωγή

1.1 Διατήρηση δεδομένων σε αντικειμενοστραφείς εφαρμογές

[Wik10k]

Στην επιστήμη των υπολογιστών, η έννοια της διατήρησης δεδομένων (persistence) αναφέρεται στη διαδικασία αποθήκευσης των δεδομένων ενός προγράμματος σε ένα μόνιμο αποθηκευτικό μέσο, ώστε αυτά να επιζήσουν μετά τον τερματισμό της διεργασίας που τα δημιούργησε. Χωρίς τη διατήρηση δεδομένων, αυτά θα υπήρχαν μόνο στη μνήμη του υπολογιστή και θα χάνονταν κατά τον τερματισμό της λειτουργίας του.

Κατά τη σχεδίαση ενός μεγάλου έργου λογισμικού, πρέπει να ληφθεί μια σημαντική απόφαση που αφορά τον τρόπο αποθήκευσης των δεδομένων που χρειάζεται να διατηρούνται (π.χ. σε αρχεία δεδομένων, βάσεις δεδομένων κλπ). Η πλέον διαδεδομένη επιλογή είναι η χρήση σχεσιακών βάσεων δεδομένων. Οι σχεσιακές βάσεις δεδομένων αποτελούν μια ώριμη, ευέλικτη και αποδοτική λύση για τη διαχείριση δεδομένων. Επιπλέον, στηρίζονται σε ένα ισχυρό μαθηματικό υπόβαθρο. Τα πλεονεκτήματα τους είναι αρκετά, όπως η εξυπηρέτηση πολλαπλών εφαρμογών, η υποστήριξη μιας SQL γλώσσας για τη διαχείριση των δεδομένων, οι προχωρημένες λειτουργίες αναζήτησης, ταξινόμησης, ομαδοποίησης δεδομένων κ.α.

Εκτός τούτου, πρέπει να ληφθεί και η απόφαση που αφορά τη γλώσσα προγραμματισμού που θα χρησιμοποιηθεί. Οι σύγχρονες τάσεις της τεχνολογίας λογισμικού επικεντρώνονται στη χρησιμοποίηση αντικειμενοστραφών γλωσσών προγραμματισμού, όπου ο χειρισμός των

σχετιζόμενων δεδομένων και των διαδικασιών που επενεργούν σε αυτά, γίνεται από κοινού μέσω μιας δομής δεδομένων που ονομάζεται αντικείμενο. Τα πλεονεκτήματα είναι αρκετά όπως η απόκρυψη πληροφορίας, η εύκολη μοντελοποίηση πραγματικών οντοτήτων, η επαναχρησιμοποιησιμότητα των αντικειμένων κ.α.

Κατά την αντικειμενοστραφή ανάπτυξη λογισμικού, που αποτελεί την πλέον χρησιμοποιούμενη τεχνική ανάπτυξης σήμερα, η διατήρηση των δεδομένων σε σχεσιακές βάσεις δεδομένων χρειάζεται ειδική αντιμετώπιση. Η διατήρηση των αντικειμένων μιας αντικειμενοστραφούς εφαρμογής σε μια σχεσιακή βάση δεδομένων δεν είναι μια 1:1 διαδικασία, καθώς η αναπαράσταση των δεδομένων είναι δομικά διαφορετική.

Ο μηχανικός λογισμικού πρέπει να λύσει αυτήν την αναντιστοιχία (Object-Relation impedance mismatch). Ο πιο στοιχειώδης τρόπος είναι η σύνταξη προσαρμοσμένων SQL ερωτημάτων, κατευθείαν στον κώδικα, για κάθε αντικείμενο που πρέπει η εφαρμογή να διατηρήσει στη βάση δεδομένων. Όμως ο τρόπος αυτός έχει αρκετά μειονεκτήματα: Δεν είναι ευέλικτος, είναι χρονοβόρος και η συντήρηση του κώδικα είναι δύσκολη. Αυτά τα μειονεκτήματα αντιτίθενται στις βασικές αρχές της τεχνολογίας λογισμικού που απαιτούν κώδικα ανώτερου επιπέδου, περισσότερο συντηρήσιμο και ταχύτερης κατασκευής.

1.2 Αντικείμενο διπλωματικής

Στην παρούσα διπλωματική εργασία αναπτύσσουμε ένα νέο επίπεδο διατήρησης δεδομένων για το διαδικτυακό περιβάλλον διαχείρισης ιστοσελίδων μαθημάτων του Ε.Μ.Π. ονόματι NCCMS (.NET Course Content Management System). Το περιβάλλον NCCMS έχει αναπτυχθεί από το Εργαστήριο Συστημάτων Βάσεων Γνώσεων και Δεδομένων και φιλοξενεί μεγάλο αριθμό ιστοσελίδων μαθημάτων του Ε.Μ.Π.

Η υλοποίηση του νέου επιπέδου διατήρησης δεδομένων βασίζεται σε τεχνικές αντικειμενο-σχεσιακής απεικόνισης (Object-Relational Mapping). Οι τεχνικές ORM ξεφεύγουν από τη συνήθη τακτική της χρησιμοποίησης προσαρμοσμένων SQL ερωτημάτων για την αποθήκευση ή την ανάκτηση του κάθε αντικειμένου της εφαρμογής. Είναι προηγμένες τεχνικές που αυτοματοποιούν τον τρόπο διασύνδεσης του μοντέλου αντικειμένων μιας εφαρμογής με τη σχεσιακή βάση δεδομένων, χρησιμοποιώντας μετα-δεδομένα για την περιγραφή του τρόπου διασύνδεσης.

Το νέο επίπεδο διατήρησης δεδομένων στοχεύει να προσφέρει τα εξής πλεονεκτήματα στην πλατφόρμα NCCMS:

- Μείωση του χρόνου ανάπτυξης λογισμικού. Σε μελλοντικές επεκτάσεις, όσον αφορά το θέμα της διατήρησης δεδομένων, ο μηχανικός λογισμικού θα χρειαστεί απλώς να ρυθμίσει τα κατάλληλα μετα-δεδομένα της απεικόνισης των νέων αντικειμένων.

Έπειτα θα έχει στη διάθεση του τις λειτουργίες του επίπεδου διατήρησης δεδομένων για να διαχειρίζεται με εύκολο τρόπο την αποθήκευση και την ανάκτηση τους.

- Μείωση του κώδικα. Οι ORM τεχνικές αναλαμβάνουν την αυτόματη σύνταξη των απαραίτητων SQL ερωτημάτων για την αποθήκευση/ανάκτηση των αντικειμένων της εφαρμογής.
- Ευκολότερη συντήρηση. Ο κώδικας είναι περισσότερο καθαρός και ευανάγνωστος αφού επικεντρώνεται πλέον στις πραγματικές επιχειρηματικές ανάγκες της εφαρμογής παρά στη διαχείριση της βάσης δεδομένων.

Συνοπτικά η παρούσα διπλωματική εργασία: (α) καταγράφει το μοντέλο δεδομένων και των λειτουργιών του NCCMS, (β) περιγράφει τις βασικές αρχές της αντικειμενο-σχεσιακής απεικόνισης και (γ) σχεδιάζει και υλοποιεί το νέο επίπεδο διατήρησης δεδομένων για το NCCMS χρησιμοποιώντας το ORM εργαλείο NHibernate.

1.2.1 Συνεισφορά

Η συνεισφορά της διπλωματικής συνοψίζεται ως εξής:

1. Μελετήθηκαν τα δύο κυρίαρχα διαθέσιμα ORM εργαλεία για .NET Framework εφαρμογές: (α) Το Microsoft ADO.NET Entity Framework και (β) το NHibernate. Από αυτά επιλέξαμε να χρησιμοποιήσουμε το NHibernate για την ανάπτυξη του επιπέδου διατήρησης δεδομένων του NCCMS.
2. Μελετήθηκε ο τρόπος λειτουργίας του NHibernate καθώς και τα μετα-δεδομένα που χρειάζεται να ορίσει μια εφαρμογή και έχουν σχέση με τον τρόπο απεικόνισης των αντικειμένων της στη βάση δεδομένων.
3. Αναλύθηκε και αποτυπώθηκε η αρχιτεκτονική του NCCMS, τα συστατικά του, το μοντέλο δεδομένων του και ο τρόπος λειτουργίας του.
4. Αναλύθηκε το κατώτερο επίπεδο της αρχιτεκτονικής του NCCMS που αποτελείται από τη βιβλιοθήκη διαχείρισης δενδρικών δομών NCommet. Το NCommet αποτελείται από υποσυστήματα που καλούνται Υπηρεσίες. Μία από αυτές είναι η «Υπηρεσία Αποθήκευσης» και αποτελεί το επίπεδο διατήρησης δεδομένων που είναι υπεύθυνο για τη διατήρηση των δενδρικών δομών της εφαρμογής (άρα και του NCCMS) σε μια σχεσιακή βάση δεδομένων.
5. Διατυπώθηκαν οι λειτουργικές απαιτήσεις της νέας «Υπηρεσίας Αποθήκευσης» του NCommet λαμβάνοντας υπ’ όψιν τις υπάρχουσες λειτουργικές απαιτήσεις της πλατφόρμας και ότι θα χρησιμοποιηθούν ORM τεχνικές.
6. Ολοκληρώθηκε η υλοποίηση της νέας «Υπηρεσίας Αποθήκευσης» του NCommet χρησιμοποιώντας το NHibernate: (α) Συντάχθηκαν τα απαραίτητα αρχεία ρυθμίσεων

απεικόνισης των κλάσεων των αντικειμένων της εφαρμογής στη βάση δεδομένων και (β) αναπτύχθηκε ο απαραίτητος κώδικας που υλοποιεί τις λειτουργικές απαιτήσεις που θέσαμε.

7. Ενσωματώθηκε η νέα «Υπηρεσία Αποθήκευσης» του NCommet σε πρότυπο σύστημα, ονόματι SimpleCMSPlus, που αποτελεί μια απλουστευμένη έκδοση του NCCMS.
8. Ελέγχθηκε το SimpleCMSPlus πραγματοποιώντας κατάλληλα σενάρια λειτουργίας.

Τα ποιοτικά χαρακτηριστικά της διπλωματικής εργασίας είναι τα εξής:

- Συγγραφή ποιοτικού κώδικα ο οποίος συνάδει με τις καλές πρακτικές του αντικειμενοστραφούς προγραμματισμού, όπως η ορθή χρήση των διεπαφών (interfaces), του πολυμορφισμού, της κληρονομικότητας, της απόκρυψης πληροφορίας κ.α.
- Χρήση τεχνικών τεχνολογίας λογισμικού όπως σχεδιαστικά μορφήματα (design patterns) και ανάλυση της αρχιτεκτονικής του συστήματος με UML διαγράμματα.
- Η υλοποίηση της Υπηρεσίας Αποθήκευσης έχει γίνει με τέτοιο τρόπο ώστε να είναι ικανή να αποθηκεύει, με ελάχιστο κόπο, οποιαδήποτε αντικείμενα μιας εφαρμογής που χρησιμοποιεί το NCommet (όπως είναι το SimpleCMSPlus και το NCCMS).
- Έχει δοθεί έμφαση στην υψηλή συνεκτικότητα της Υπηρεσίας Αποθήκευσης καθώς και στη χαμηλή σύζευξη της με τις υπόλοιπες Υπηρεσίες του NCommet.

1.3 Οργάνωση κειμένου

Παρακάτω περιγράφονται συνοπτικά τα κεφάλαια της παρούσας διπλωματικής εργασίας. Στο 2^{ο} κεφάλαιο πραγματοποιείται μια σύντομη περιγραφή της πλατφόρμας NCCMS και των λειτουργιών της.

Στο 3^{ο} κεφάλαιο περιγράφεται τι είναι η αντικειμενο-σχεσιακή απεικόνιση (ORM) και παρουσιάζονται δύο κυρίαρχα εργαλεία για .NET Framework εφαρμογές: το (α) ADO.NET Entity Framework και το (β) NHibernate. Δίνεται έμφαση στην ανάλυση του NHibernate, στον τρόπο λειτουργίας του και στις ρυθμίσεις του, καθώς χρησιμοποιείται για την υλοποίηση της διπλωματικής εργασίας. Επίσης, στο τέλος του κεφαλαίου, περιγράφεται συνοπτικά η έννοια της XML Σειριακοποίησης.

Το 4^{ο} κεφάλαιο προχωρά στην ανάλυση των απαιτήσεων του συστήματος. Αρχικά αναλύεται η γενική αρχιτεκτονική του SimpleCMSPlus, το οποίο είναι μια απλουστευμένη έκδοση του NCCMS. Ύστερα, δίνεται έμφαση στην περιγραφή του κατώτερου επίπεδου της αρχιτεκτονικής του SimpleCMSPlus που αποτελείται από τη βιβλιοθήκη διαχείρισης

δενδρικών δομών NCommet. Περιγράφεται η αρχιτεκτονική του NCommet η οποία αποτελείται από τον πυρήνα και διάφορες Υπηρεσίες (υποσυστήματα), στις οποίες περιλαμβάνεται και η «Υπηρεσία Αποθήκευσης» που αποτελεί το επίπεδο διατήρησης δεδομένων. Έπειτα, διατυπώνονται οι λειτουργικές απαιτήσεις του πυρήνα του NCommet και της νέας «Υπηρεσίας Αποθήκευσης» που αναπτύξαμε με την παρούσα διπλωματική εργασία. Τέλος, παρουσιάζεται το μοντέλο οντοτήτων συσχετίσεων του NCommet, που καλείται να διατηρήσει η «Υπηρεσία Αποθήκευσης» στη βάση δεδομένων.

Στο 5ο κεφάλαιο ακολουθεί η λεπτομερής ανάλυση της σχεδίασης του πυρήνα του NCommet. Περιγράφονται η αρχιτεκτονική του, οι βασικές κλάσεις του, οι ρυθμίσεις του και πως οι Υπηρεσίες του ορίζονται με τη μορφή διεπαφών (interfaces). Τέλος, παρουσιάζεται το σχεσιακό διάγραμμα της βάσης δεδομένων.

Στο 6ο κεφάλαιο παρουσιάζεται λεπτομερώς η νέα «Υπηρεσία Αποθήκευσης» του NCommet, που υλοποιήθηκε στην παρούσα διπλωματική εργασία, με τη βοήθεια του ORM εργαλείου NHibernate. Περιγράφονται οι ρυθμίσεις απεικόνισης καθώς και πως η «Υπηρεσία Αποθήκευσης» αποθηκεύει τα αντικείμενα των εφαρμογών που χρησιμοποιούν το NCommet (όπως του NCCMS και του SimpleCMSPlus). Ύστερα, αναλύονται οι κλάσεις της υλοποίησης. Στο τέλος του κεφαλαίου, παρουσιάζονται τα εργαλεία και οι πλατφόρμες ανάπτυξης του συστήματος.

Στο 7ο κεφάλαιο, ακολουθεί η περιγραφή της διαδικασίας του ελέγχου του συστήματος. Ο έλεγχος πραγματοποιήθηκε με την εκτέλεση κατάλληλων σεναρίων λειτουργίας και την καταγραφή των SQL ερωτημάτων προς/από τη βάση δεδομένων.

Στο 8ο κεφάλαιο καταλήγει η διπλωματική εργασία συνοψίζοντας τους στόχους που επετεύχθησαν και παραθέτοντας τις δυνατότητες για μελλοντική επέκταση της.

2

Τι είναι το περιβάλλον ανάπτυξης ιστοσελίδων μαθημάτων «NCCMS»

Πριν την παρουσίαση των απαιτήσεων του επιπέδου διατήρησης δεδομένων, κρίνεται σκόπιμη η σύντομη παρουσίαση των ιστοσελίδων μαθημάτων NCCMS.

2.1 Σύντομο ιστορικό

Το Περιβάλλον Ανάπτυξης Ιστοσελίδων Μαθημάτων **NCCMS** (.NET Course Content Management System) αναπτύχθηκε στα πλαίσια των «Δράσεων Κεντρικής Υποστήριξης του Εκπαιδευτικού Έργου του ΕΜΠ» από την ομάδα ανάπτυξης διαδικτυακών εφαρμογών του Εργαστηρίου Βάσεων Γνώσεων & Δεδομένων του ΕΜΠ.

Έχει στόχο να διευκολύνει τα μέλη ΔΕΠ του ιδρύματος να δημιουργήσουν δικτυακούς τόπους για εκπαιδευτικούς σκοπούς, και πιο συγκεκριμένα:

- Δικτυακούς τόπους προπτυχιακών και μεταπτυχιακών μαθημάτων.
- Επιστημονικούς οδηγούς.
- Δικτυακούς τόπους προβολής διπλωματικών ή διδακτορικών διατριβών.
- Ιστοσελίδες συνεδρίων, ημερίδων κλπ.
- Ιστοσελίδες εργαστηρίων ή ερευνητικών ομάδων.

Οι λειτουργικές προδιαγραφές του NCCMS βασίστηκαν στην ανάλυση των πιο διαδεδομένων συστημάτων ανάπτυξης ηλεκτρονικών μαθημάτων (Course Management Systems) αφαιρώντας τη λειτουργικότητα που σχετίζεται με θέματα τηλε-εκπαίδευσης (e-learning), καθώς και πιο γνωστών συστημάτων διαχείρισης περιεχομένου (Content Management Systems) ανοικτού κώδικα.

2.2 Περιβάλλον λειτουργίας και ανάπτυξης

Η σχεδίαση και η ανάπτυξη του NCCMS βασίστηκε σε βέλτιστες πρακτικές υλοποίησης πληροφοριακών συστημάτων. Χρησιμοποιήθηκαν οι πιο πρόσφατες και γενικά αποδεκτές μεθοδολογίες, τεχνολογίες και περιβάλλοντα ανάπτυξης λογισμικού, πρότυπα και πρωτόκολλα.

- Μεθοδολογίες: Domain driven design, Object oriented programming, Pattern based & test-driven development, Aspect oriented programming, Agile software development, Service oriented architecture.
- Τεχνολογίες & περιβάλλοντα ανάπτυξης: Microsoft NET 3.0, Visual Studio 2005 (application development), WWF (Windows Workflow Foundation), WCF (Windows Communication Foundation), MSMQ (Message Queuing), Shibboleth (Federated Identity).
- Πρότυπα & πρωτόκολλα: XML (XML schemas και XSLT), Web Services (WSDL, SOAP, UDDI, WS-Security), Federated Identity (SAML).

2.3 Δικτυακός Τόπος Μαθημάτων του ΕΜΠ

Ο κεντρικός NCCMS **Δικτυακός Τόπος Μαθημάτων** του ΕΜΠ, βρίσκεται στη διεύθυνση http://courses.dbnet.ntua.gr.

Εάν δεν προηγηθεί διαδικασία ταυτοποίησης (login), η σελίδα αυτή εμφανίζει τους δικτυακούς τόπους μόνο των μαθημάτων που έχουν δηλωθεί από το δημιουργό τους ως «ανοικτά», δηλαδή επιτρέπουν την πρόσβαση στον ανώνυμο χρήστη.

Χρειάζεται να γίνει διαδικασία ταυτοποίησης (login), στις εξής περιπτώσεις που ο χρήστης είναι ένας:

- Καθηγητής, που θέλει να δημιουργήσει νέο μάθημα ή να διαχειριστεί υπάρχον μάθημά του.
- Φοιτητής, που θέλει τα πρόσθετα δικαιώματα πρόσβασης, δηλαδή:
    - Να δηλώσει την πρόθεσή του να γίνει μέλος σε «κλειστά» μαθήματα (ή σε «ανοικτά» μαθήματα με «κλειδωμένο» περιεχόμενο).

- Να πλοηγηθεί στα «κλειστά» μαθήματα όπου έχει γίνει μέλος (ή να δει το «κλειδωμένο» περιεχόμενο «ανοιχτών» μαθημάτων όπου έχει γίνει μέλος).

Για την διαδικασία ταυτοποίησης (**login**), πρέπει να χρησιμοποιηθεί το username και password που του έχει αποδοθεί από το Κέντρο Ηλεκτρονικού Υπολογιστή του ΕΜΠ. Οι καθηγητές αποκτούν αυτόματα τη δυνατότητα να δημιουργούν μαθήματα, χωρίς κάποια έγκριση από τον γενικό διαχειριστή του NCCMS.

Επιπλέον, ο Δικτυακός Τόπος Μαθημάτων του ΕΜΠ διαθέτει και επιπρόσθετο κατάλογο χρηστών, στους οποίους μπορεί να δοθεί το δικαίωμα δημιουργίας, διαχείρισης ή ανάγνωσης ιστοσελίδων. Τη διαχείριση των χρηστών του ειδικού καταλόγου έχει ο γενικός διαχειριστής.

2.4 Δικτυακός Τόπος ενός Μαθήματος

Ο **Δικτυακός Τόπος ενός Μαθήματος** είναι οι ιστοσελίδες του μαθήματος. Στο περιβάλλον NCCMS, Ο Δικτυακός Τόπος ενός Μαθήματος μπορεί να περιλάβει:

- Τα στοιχεία ταυτότητας του μαθήματος (Σχολή, Τομέας, Τίτλος, Διδάσκοντες, κλπ).
- Οποιοδήποτε υλικό έχει σχέση με το μάθημα (περιγραφές, ασκήσεις, εργασίες, βιβλιογραφικό υλικό του μαθήματος, διδάσκοντες και βοηθοί διδασκαλίας, κλπ):
    - Σε μορφή άμεσα αναγνώσιμη από τις ιστοσελίδες (*HTML format*),
    - Ως επισυναπτόμενα έγγραφα (*Word documents, PDF documents*, κλπ),
    - Σε μορφή συνδέσμων διαδικτύου (*αναφορές σε άλλες ιστοσελίδες*).
- Ανακοινώσεις ή λίστες από χρονικά ταξινομημένα γεγονότα.

Η οργάνωση του Δικτυακού Τόπου ενός Μαθήματος έχει ως εξής:

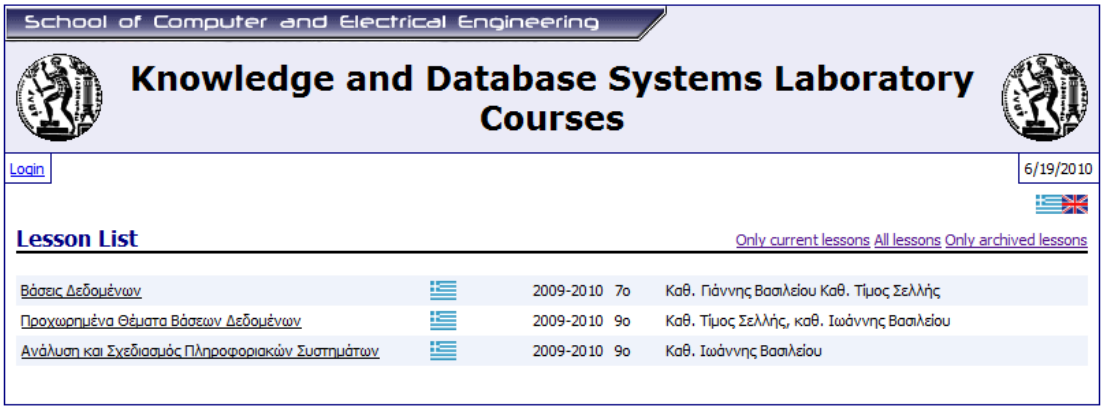
- Το **υλικό** του μαθήματος οργανώνεται σε **φακέλους υλικού**.
- Οι φάκελοι υλικού τοποθετούνται σε **ιστοσελίδες**.
- Μία ιστοσελίδα, εκτός από φακέλους υλικού, μπορεί να έχει υπο-σελίδες.

Οι ρυθμίσεις του NCCMS που έχουν γίνει στο Δικτυακό Τόπο Μαθημάτων του ΕΜΠ, επιβάλλουν τους εξής κανόνες στη δομή του περιεχομένου ενός μαθήματος:

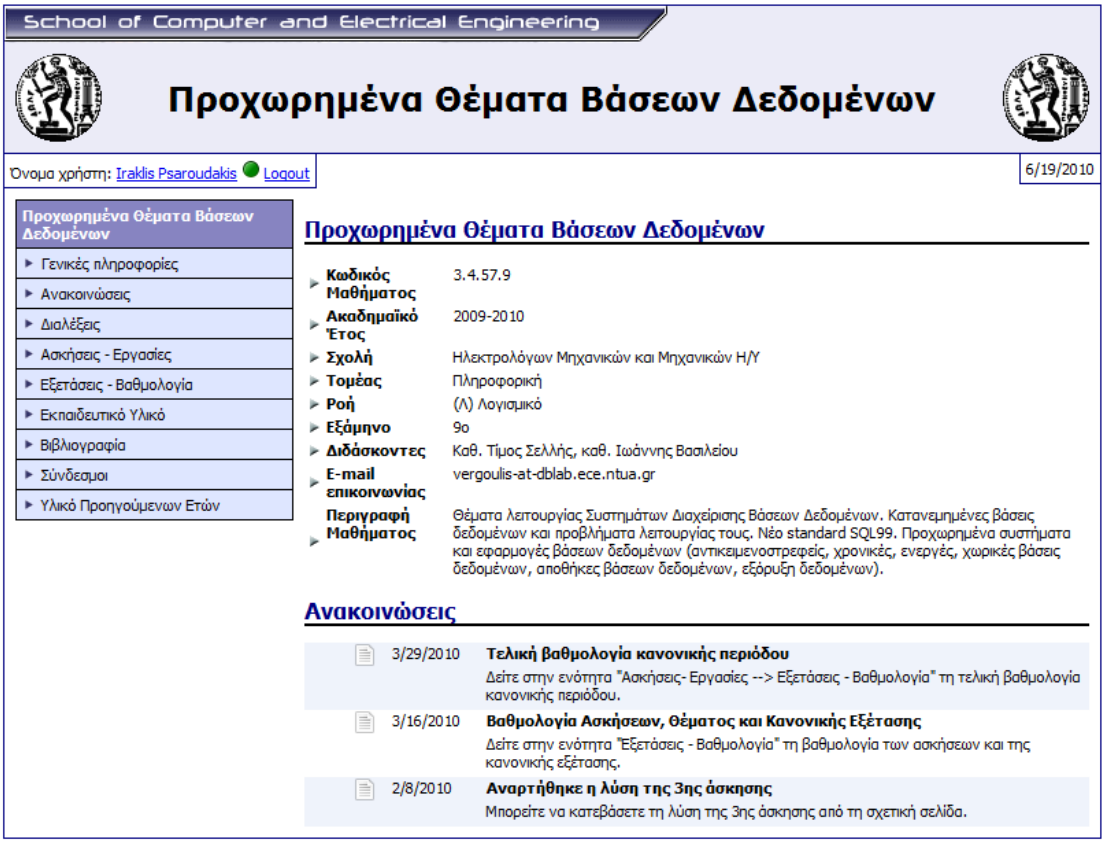
- Ο Δικτυακός Τόπος Μαθημάτων του ΕΜΠ μπορεί να φιλοξενήσει οσαδήποτε *μαθήματα.*
- Ένα μάθημα (δικτυακός τόπος μαθήματος) περιέχει οσεσδήποτε *ιστοσελίδες*.
    - Τύποι ιστοσελίδων: Κεντρική σελίδα μαθήματος, Απλή Σελίδα, Σελίδα διαλέξεων, Σελίδα Διάλεξης.
- Μία ιστοσελίδα περιέχει οσουσδήποτε *φακέλους*.

- Τύποι φακέλων: Φάκελος Περιεχομένου, Φάκελος Ανακοινώσεων, Φάκελος Προσώπων, Φάκελος με μορφοποιημένο κείμενο.

- Ένας φάκελος περιέχει οσαδήποτε τεμάχια *υλικού*.
  - Τύποι υλικού: Κείμενο, Έγγραφο, Σύνδεσμος, Πρόσωπο.



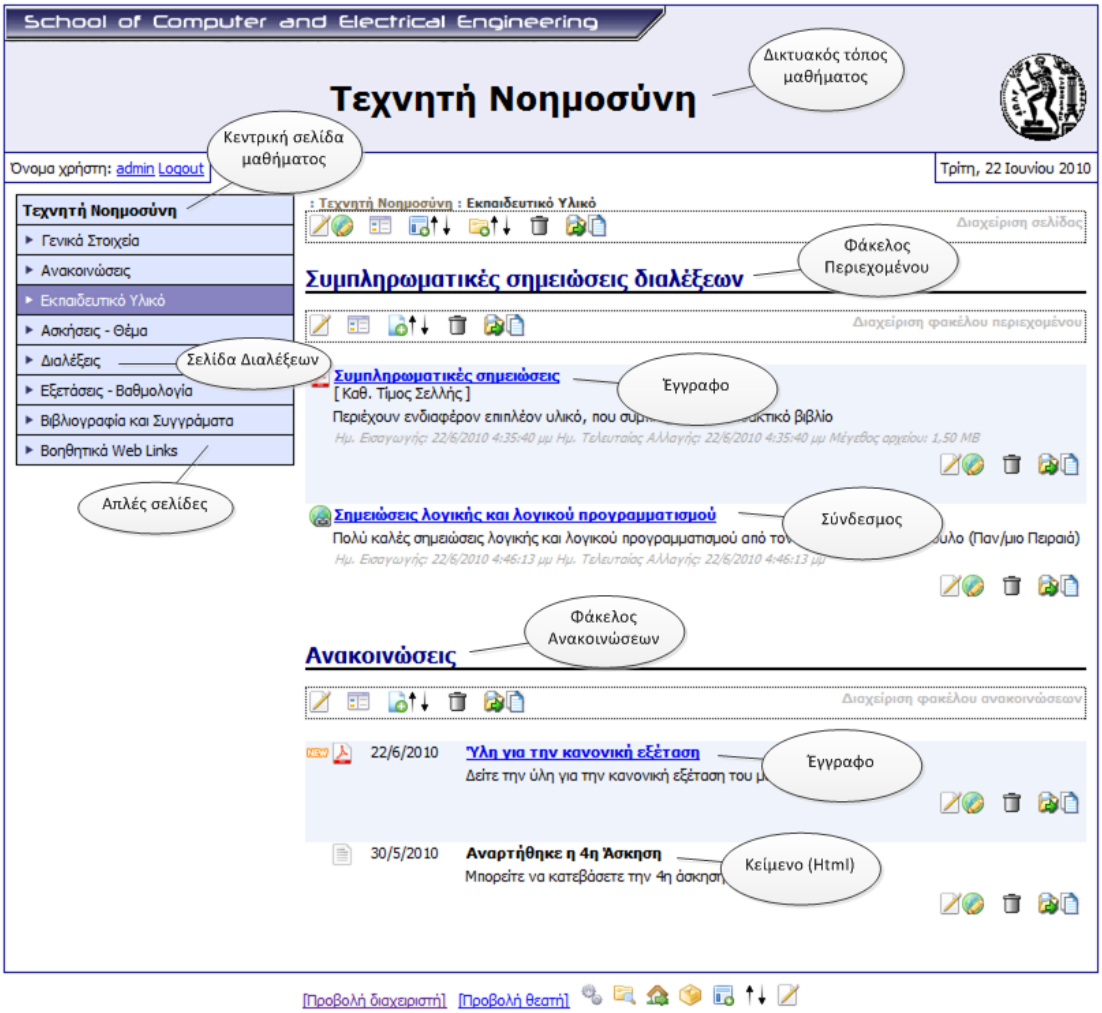
Εικόνα 2-1 Η αρχική σελίδα του Δικτυακού Τόπου Μαθημάτων του ΕΜΠ



Εικόνα 2-2 Η κεντρική σελίδα του Δικτυακού Τόπου ενός Μαθήματος

Εικόνα 2-3 Μία απλή σελίδα του Δικτυακού Τόπου ενός Μαθήματος, με φακέλους και υλικό μέσα στους φακέλους

Εικόνα 2-4 Η σελίδα διαλέξεων του Δικτυακού Τόπου ενός Μαθήματος



Εικόνα 2-5 Μία απλή σελίδα του Δικτυακού Τόπου ενός Μαθήματος, για ένα διαχειριστή μαθημάτων. Επισημαίνονται οι τύποι των ιστοσελίδων, των φακέλων και του υλικού.

2.5 *Εξουσιοδότηση χρηστών*

2.5.1 *Κατηγορίες χρηστών των Ιστοσελίδων ενός Μαθήματος*

Οι Ιστοσελίδες *ενός* Μαθήματος στο περιβάλλον NCCMS, έχουν τρεις κατηγορίες χρηστών:

- **Ανώνυμοι χρήστες**, είναι όσοι προσπελαύνουν τις ιστοσελίδες του μαθήματος, χωρίς να έχουν πραγματοποιήσει login (διαδικασία ταυτοποίησης και εξουσιοδότησης).
- **Θεατές**, είναι οι φοιτητές στους οποίους έχει δοθεί η σχετική εξουσιοδότηση. Οι θεατές ενός μαθήματος μπορούν να δουν υλικό που έχει δηλωθεί ως *διαβαθμισμένο* (μη ορατό στους ανώνυμους χρήστες). Επιπλέον, μπορούν να δουν τις σελίδες του μαθήματος όταν αυτό δηλωθεί ως *κλειστό μάθημα*. Για να αποκτήσουν τα σχετικά δικαιώματα, πρέπει να πραγματοποιήσουν login.

- **Διαχειριστές**, είναι όσοι έχουν δικαίωμα διαχείρισης του περιεχομένου και της εμφάνισης των web σελίδων του μαθήματος. Οι διαχειριστές, επιπλέον, καθορίζουν τα επίπεδα πρόσβασης στο μάθημα ή σε μεμονωμένο υλικό του μαθήματος. Τέλος, εγκρίνουν τους θεατές των ιστοσελίδων του μαθήματος (φοιτητές που έχουν υποβάλει σχετικό αίτημα).

Σε επίπεδο *ολόκληρου* του Δικτυακού Τόπου Μαθημάτων, το περιβάλλον NCCMS έχει επιπλέον τους εξής ρόλους:

- **Δημιουργοί μαθημάτων**, είναι όσοι έχουν το δικαίωμα να δημιουργήσουν νέα μαθήματα.
- **Γενικός Διαχειριστής**, είναι ο υπεύθυνος συντήρησης ολόκληρου του Δικτυακού Τόπου Μαθημάτων, με δυνατότητες διαμόρφωσης των γενικών χαρακτηριστικών του Δικτυακού Τόπου.

2.5.2 Εξουσιοδότηση – Απόδοση ρόλων

Ένας Δικτυακός Τόπος Μαθημάτων στο περιβάλλον NCCMS, μπορεί να χρησιμοποιήσει πολλαπλές πηγές (καταλόγους) για την ταυτοποίηση των χρηστών κατά τη διαδικασία εισόδου (login). Ο Δικτυακός Τόπος Μαθημάτων του Ε.Μ.Π. χρησιμοποιεί κυρίως τον κατάλογο χρηστών του Ε.Μ.Π. Κατά συνέπεια, κατά τη διαδικασία εισόδου, οι χρήστες πρέπει να χρησιμοποιούν το username και password που τους έχει αποδοθεί από το Κέντρο Ηλεκτρονικού Υπολογιστή (ΚΗΥ) του ιδρύματος.

Για την απόδοση ρόλων (δικαιωμάτων πρόσβασης), ισχύουν τα εξής:

- Κάθε μέλος ΔΕΠ που συνδέεται με το username και password που του έχει αποδοθεί από το ΚΗΥ, γίνεται αυτόματα δημιουργός μαθημάτων.
- Όταν ένα Μέλος ΔΕΠ δημιουργήσει ένα νέο μάθημα, αυτόματα αποκτά το ρόλο *διαχειριστή* του μαθήματος αυτού.

Ο *διαχειριστής* ενός μαθήματος έχει τη δυνατότητα:

- Να επιτρέψει το ρόλο *διαχειριστή* στο συγκεκριμένο μάθημα, σε οποιονδήποτε χρήστη έχει κάνει σχετική αίτηση (συνήθως σε άλλα μέλη ΔΕΠ).
- Να επιτρέψει το ρόλο *θεατή* στο συγκεκριμένο μάθημα, σε οποιονδήποτε χρήστη έχει κάνει σχετική αίτηση (συνήθως σε φοιτητές).

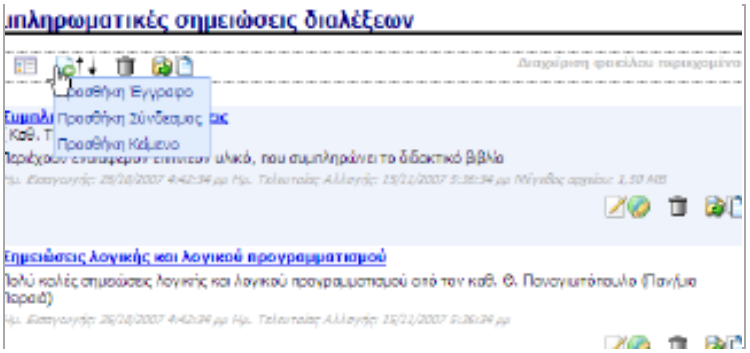
2.6 Επιπρόσθετες δυνατότητες

Το NCCMS δίνει τις εξής δυνατότητες διαχείρισης (σε επίπεδο μαθήματος):

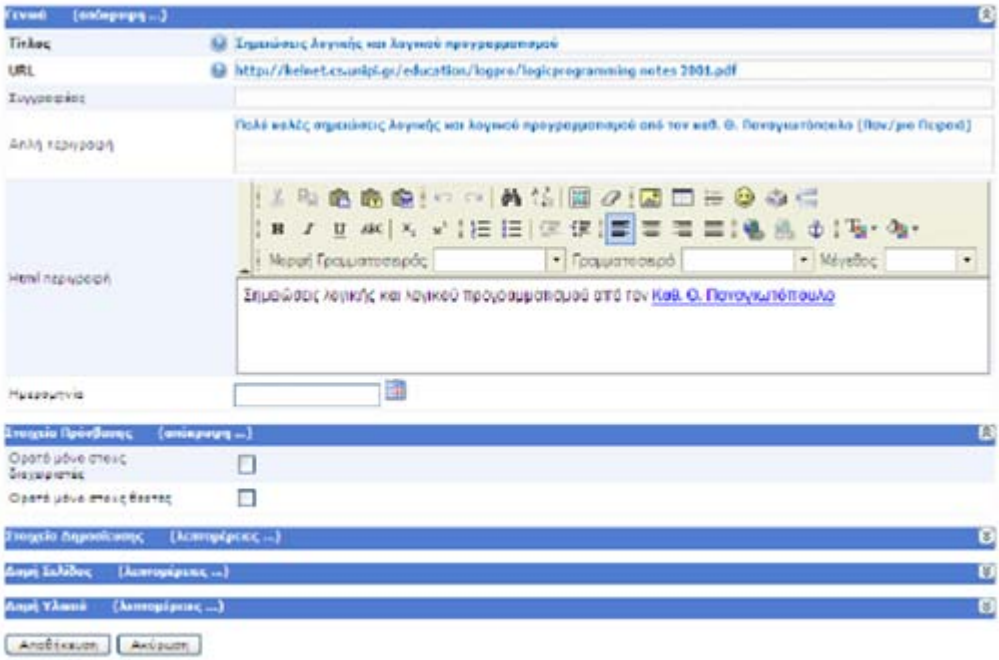
- **Αντιγραφή μαθήματος**: Με την αντιγραφή ενός μαθήματος δημιουργείται ένα νέο Μάθημα πανομοιότυπο με το πρωτότυπο που περιέχει αντίγραφα όλου του υλικού του πρωτοτύπου (σελίδες, φάκελοι υλικού, περιεχόμενο, αρχεία, κλπ).
- **Αντίγραφα ασφαλείας και διαδικασία επαναφοράς (staging)**: Ένας διδάσκον μπορεί ανά πάσα στιγμή να δημιουργήσει ένα αντίγραφο ασφαλείας (Backup) ενός Μαθήματος, το οποίο περιλαμβάνει όλη την απαραίτητη πληροφορία ώστε με βάση μόνο αυτό, να μπορέσει να επαναφέρει το μάθημα (Restore) ακριβώς όπως ήταν τη στιγμή της λήψης του αντιγράφου.
- **Καθορισμός γλώσσας μαθήματος**: Εάν ένας διδάσκον επιθυμεί οι Web Σελίδες του Μαθήματός του να είναι σε διαφορετική γλώσσα από την προκαθορισμένη γλώσσα του Δικτυακού Τόπος Μαθημάτων που φιλοξενεί το μάθημα, μπορεί να καθορίσει τη γλώσσα.

2.7 Παράδειγμα διαχείρισης περιεχομένου

Έστω ότι ο διαχειριστής θέλει να προσθέσει έναν καινούργιο *υπερσύνδεσμο* μέσα σε έναν υπάρχον *φάκελο*. Τα βήματα που χρειάζονται φαίνονται στα παρακάτω στιγμιότυπα οθόνης.



Εικόνα 2-6 Κλήση της δημιουργίας πληροφοριακού Υλικού



Εικόνα 2-7 Φόρμα προσθήκης/επεξεργασίας συνδέσμου διαδικτύου.

3

Θεωρητικό υπόβαθρο

Η παρούσα διπλωματική εργασία προϋποθέτει ότι ο αναγνώστης είναι εξοικειωμένος με την ανάπτυξη .NET Framework εφαρμογών με τη γλώσσα προγραμματισμού C#, καθώς και με τη γλώσσα SQL για την δημιουργία ερωτημάτων προς μια βάση δεδομένων.

Θα περιγράψουμε αρχικά τί είναι η αντικείμενο-σχεσιακή απεικόνιση (ORM) και ύστερα θα παρουσιάσουμε δύο σημαντικά ORM εργαλεία: (α) Το ADO.NET Entity Framework και (β) το NHibernate. Θα επικεντρωθούμε στην ανάλυση του NHibernate, καθώς θα το χρησιμοποιήσουμε για την υλοποίηση της Υπηρεσίας Αποθήκευσης του NCommet.

Επιπλέον, στο τέλος της ενότητας θα δώσουμε μια σύντομη περιγραφή της τεχνικής XML Σειριακοποίησης (Serialization) για .NET Framework εφαρμογές.

3.1 Εργαλεία αντικειμενο-σχεσιακής απεικόνισης

[Kua09]

3.1.1 Διατήρηση δεδομένων στις αντικειμενοστραφείς εφαρμογές

3.1.1.1 Διατήρηση δεδομένων σε σχεσιακές βάσεις δεδομένων

Τα περισσότερα προγράμματα που αναπτύσσουμε αποθηκεύουν στην μνήμη του υπολογιστή δεδομένα και τα επεξεργάζονται. Όμως σπανίως αυτά τρέχουν συνεχώς. Χρησιμοποιούμε τον όρο «διατήρηση δεδομένων» (**persistency**) για να περιγράψουμε τη διαδικασία αποθήκευσης

των δεδομένων ακόμα και όταν το πρόγραμμα μας δεν τρέχει. Όταν το πρόγραμμα επανέλθει σε λειτουργία, έχει τη δυνατότητα να φορτώσει τα αποθηκευμένα δεδομένα και να συνεχίσει την επεξεργασία τους εκ νέου.

Όσον αφορά την ίδια την αποθήκευση, πρέπει να αποφασιστεί ποιο μέσο αποθήκευσης θα χρησιμοποιηθεί. Υπάρχουν διάφορες επιλογές όπως αρχεία κειμένου, απλά δυαδικά αρχεία κ.α. όμως οι περισσότερες απ' αυτές τις επιλογές περιορίζονται σε μέγεθος και αποδοτικότητα.

Η πλέον διαδομένη επιλογή είναι η χρήση σχεσιακών βάσεων δεδομένων. Οι βάσεις δεδομένων αποτελούν μια ώριμη, ευέλικτη και αποδοτική λύση για τα προβλήματα διαχείρισης δεδομένων. Μάλιστα, οι σχεσιακές βάσεις δεδομένων στηρίζονται σε ένα ισχυρό μαθηματικό υπόβαθρο.

Ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων (Relational Database Management System ή RDBMS) έχει τρόπους να προσφέρει δεδομένα σε πολλές διαφορετικές εφαρμογές. Σε μια βάση δεδομένων μπορούν να έχουν πρόσβαση προγράμματα που είναι γραμμένα σε διαφορετικές γλώσσες όπως C# ή Java. Η πρόσβαση στα RDBMS και στις βάσεις δεδομένων το ως, καθώς και η διαχείριση τους, στηρίζονται σε γλώσσες που βασίζονται στην SQL (Structured Query Language). Οι γλώσσες αυτές προσφέρουν από απλές λειτουργίες όπως CRUD εντολές (Create – Read – Update – Delete εντολές) μέχρι προχωρημένες λειτουργίες αναζήτησης, ταξινόμησης και ομαδοποίησης δεδομένων.

Ειδικότερα για τις αντικειμενοστραφείς εφαρμογές, οι σύγχρονες τεχνικές ανάπτυξης λογισμικού απαιτούν τη διατήρηση των δεδομένων των αντικειμένων τους σε σχεσιακές βάσεις δεδομένων. Έτσι τα αντικείμενα που χρησιμοποιούνται από την αντικειμενοστραφή εφαρμογή θα μπορέσουν να επιζήσουν της διεργασίας που τα δημιούργησε και να επαναχρησιμοποιηθούν σε επόμενη λειτουργία της εφαρμογής. Τα αντικείμενα αυτά τα ονομάζουμε persisted. Βεβαίως δεν είναι όλα τα αντικείμενα persisted, καθώς υπάρχουν και τα παροδικά (transient) αντικείμενα που έχουν περιορισμένο κύκλο ζωής, εξαρτώμενο από τον κύκλο ζωής της διεργασίας που τα δημιούργησε.

Επειδή τα αντικείμενα συνθέτουν ολόκληρους γράφους αντικειμένων, ο προγραμματιστής καλείται να βρει έναν τρόπο να αποθηκεύσει αυτούς τους γράφους αντικειμένων καθώς και τα δεδομένα τους στη βάση δεδομένων. Ο τρόπος επίτευξης της αποθήκευσης επαφίεται στην κρίση του προγραμματιστή. Για παράδειγμα, μια απλή τεχνική αποθήκευσης δεδομένων είναι η χρησιμοποίηση προσαρμοσμένων SQL ερωτημάτων κάθε φορά που η εφαρμογή χρειάζεται ανάκτηση ή τροποποίηση δεδομένων.

3.1.1.2 Ιεραρχία επιπέδων

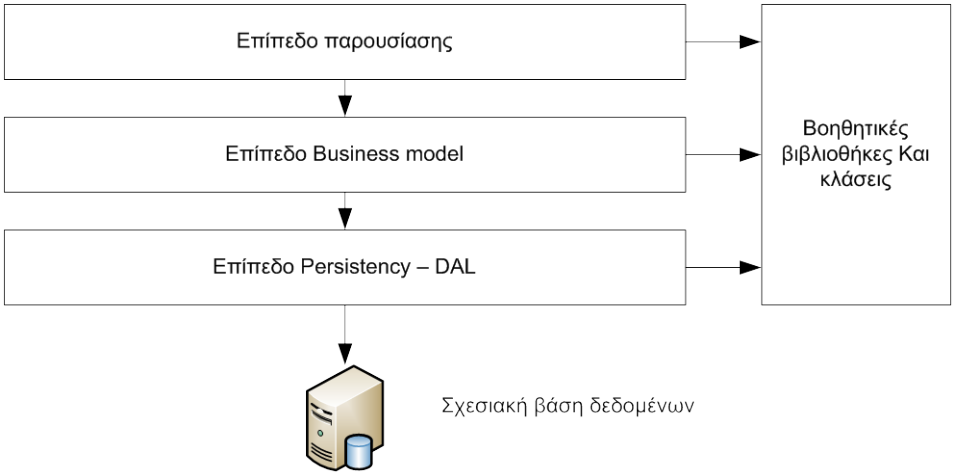
[Wik10h]

Η απλή τεχνική χρησιμοποίησης προσαρμοσμένων SQL ερωτημάτων κάθε φορά που η εφαρμογή θέλει να διαχειριστεί τα δεδομένα της βάσης, δεν είναι ιδιαίτερα ευέλικτη.

Οι σύγχρονες τάσεις στις αντικειμενοστραφείς εφαρμογές μεγάλων απαιτήσεων, προτείνουν την οργάνωση μιας εφαρμογής σε μια ιεραρχία επιπέδων. Η τεχνική επιπέδων ορίζει ότι σε κάθε επίπεδο εκτελείται μια ξεχωριστή δουλειά της εφαρμογής που έχει ένα δικό της σκοπό. Σε ένα επίπεδο, για να επιτευχθεί ο σκοπός του, χρησιμοποιούνται στοιχεία και μεθοδολογίες που προσφέρονται από το επίπεδο που βρίσκεται παρακάτω. Ενώ ταυτόχρονα έχουμε απόλυτη άγνοια για οποιαδήποτε επίπεδα βρίσκονται παραπάνω.

Στις αντικειμενοστραφής εφαρμογές, το τελευταίο επίπεδο αυτής της ιεραρχίας είναι συνήθως το Data Access Layer (DAL). Το επίπεδο αυτό σκοπό έχει να διαχωρίσει την υπόλοιπη εφαρμογή από τις λεπτομέρειες της βάσης δεδομένων. Μόνο αυτό το επίπεδο επικοινωνεί με τη βάση δεδομένων. Το παραπάνω επίπεδο ζητά από το DAL την ανάκτηση/τροποποίηση αντικειμένων από/στη βάση δεδομένων. Το DAL εκτελεί τα κατάλληλα CRUD ερωτήματα για να διαχειριστεί τα δεδομένα της βάσης δεδομένων και να γυρίσει ένα αντικείμενο στην εφαρμογή.

Ένας σύνηθες παράδειγμα ιεραρχίας επιπέδων είναι το εξής:



- **Επίπεδο παρουσίασης**: Εδώ βρίσκεται όλη η προγραμματιστική λογική για τη δημιουργία της διεπαφής χρήστη (user interface), της παρουσίασης των δεδομένων, και της εισαγωγής δεδομένων από τον χρήστη.
- **Επίπεδο business model**: Εδώ βρίσκονται τα αντικείμενα που αναπτύσσουμε και τα οποία θέλουμε να αποθηκεύουμε και να φορτώνουμε κατά βούληση, χρησιμοποιώντας το DAL. Τα ονομάζουμε “business entities” ή “business objects”. Δημιουργούνται και σχεδιάζονται με γνώμονα την λύση κάποιων συγκεκριμένων προβλημάτων. Η λογική που υλοποιούν, δηλαδή το “business logic”, ασχολείται κυρίως με την επεξεργασία γράφων από συνδεδεμένα μεταξύ τους αντικείμενα και

την επεξεργασία των ιδιοτήτων τους. Όλο το επίπεδο περιλαμβάνει το λεγόμενο "business model" της εφαρμογής.

- **Επίπεδο persistency - DAL**: Αναλαμβάνει να αποθηκεύσει μόνιμα στη βάση δεδομένων, τα παροδικά αντικείμενα με τα οποία ασχολείται το επίπεδο business model. Επίσης προσφέρει μεθόδους ανάκτησης δεδομένων από τη βάση δεδομένων ή αντικειμένων τα οποία συντίθενται αυτόματα από τα δεδομένα της βάσης.

3.1.1.3 Τρόποι υλοποίησης του DAL σε .NET εφαρμογές

Πιο συγκεκριμένα, σε .NET εφαρμογές, οι πιο γνωστές διαθέσιμες επιλογές για την υλοποίηση του DAL είναι οι εξής:

- **Κώδικας και SQL** ερωτήματα γραμμένα από τον προγραμματιστή: Σε αυτήν την κατηγορία ανήκουν οι πρακτικές οι οποίες προσπαθούν να δημιουργήσουν το DAL επίπεδο εξ' ολοκλήρου από την αρχή. Είναι σίγουρο ότι εμπεριέχουν αρκετή δουλειά, καθώς θα χρειαστεί να αναπτυχθεί κάποιο είδος αφαιρετικότητας ή αυτοματοποίησης για την εκτέλεση των SQL ερωτημάτων. Επίσης, θα πρέπει να αναπτυχθούν μέθοδοι για τη φόρτωση και την αποθήκευση των αντικειμένων.
- **DataSets**: Πρόκειται για μία τεχνική ενσωματωμένη στο .NET Framework. Επιτρέπει την πρόσβαση στο σχεσιακό μοντέλο της βάσης δεδομένων μέσω αντικειμένων.
- **LINQ-to-SQL**: Παρέχει επέκταση μιας .NET γλώσσας για την σύνταξη ενός ερωτήματος μέσα στον ίδιο τον κώδικα, χωρίς να γραφεί SQL μέσα σε Strings.
- **Object-Relational Mapping** (ORM) εργαλεία: Σε αυτήν την κατηγορία βρίσκεται η τεχνική στην οποία θα επικεντρωθούμε. Τα ORM εργαλεία επιτρέπουν την εύκολη και αυτοματοποιημένη αποθήκευση ολόκληρων γράφων από συνδεδεμένα μεταξύ τους αντικείμενα σε μια σχεσιακή βάση δεδομένων. Την αποθήκευση και το φόρτωμα των αντικειμένων τα αναλαμβάνει το ORM εργαλείο, εκτελώντας αυτόματα τα κατάλληλα SQL ερωτήματα. Εμείς απλώς πρέπει να το τροφοδοτήσουμε με την κατάλληλη μετά-πληροφορία για την απεικόνιση των αντικειμένων μας στη σχεσιακή βάση δεδομένων. Θα περιγράψουμε δύο ORM εργαλεία:
    - ADO.NET Entity Framework (EF).
    - NHibernate, το οποίο θα χρησιμοποιήσουμε κατά την υλοποίηση της διπλωματικής εργασίας.

3.1.2 Τι είναι τα ORM συστήματα

[Wik10g]

Ως αντικειμενο-σχεσιακή απεικόνιση (Object-Relational Mapping ή ORM) ορίζεται ένας αυτοματοποιημένος τρόπος διασύνδεσης του μοντέλου αντικειμένων (object model) μίας αντικειμενοστραφούς εφαρμογής με τη σχεσιακή βάση δεδομένων της εφαρμογής, χρησιμοποιώντας μετά-δεδομένα (metadata) για την περιγραφή του τρόπου της διασύνδεσης.

Με τα ORM συστήματα, μπορούμε να φτιάξουμε στην ουσία μια εικονική βάση δεδομένων από αντικείμενα, η οποία μπορεί να χρησιμοποιηθεί από την αντικειμενοστραφή γλώσσα προγραμματισμού που υποστηρίζει το ORM σύστημα.

Ουσιαστικά, τα ORM συστήματα αναλαμβάνουν το επίπεδο DAL της εφαρμογής. Επιτρέπουν την εύκολη αποθήκευση ολόκληρων γράφων από συνδεδεμένα αντικείμενα σε μια σχεσιακή βάση δεδομένων. Το ORM διαχειρίζεται μόνο του αυτή τη διαδικασία, διαμορφώνοντας αυτόματα τα κατάλληλα SQL ερωτήματα προς τη βάση.

3.1.3 Object-Relational impedance mismatch

[Wik10f]

Το O-R impedance mismatch (ή paradigm mismatch) αναφέρεται σε ένα σύνολο από θεμελιώδη εννοιολογικά και τεχνικά προβλήματα που συναντάμε κατά τη χρησιμοποίηση σχεσιακών βάσεων δεδομένων για την αποθήκευση των αντικειμένων που χρησιμοποιούμε σε ένα αντικειμενοστραφές πρόγραμμα.

Το κύριο πρόβλημα της αντικειμενο-σχεσιακής απεικόνισης είναι ότι τα αντικείμενα είναι μη-βαθμωτά μεγέθη ενώ οι σχεσιακές βάσεις δεδομένων μπορούν να διαχειρίζονται μόνο βαθμωτά μεγέθη τα οποία είναι οργανωμένα σε πίνακες. Επιπρόσθετα, ενώ μια σχεσιακή βάση δεδομένων στηρίζεται σε έναν καθαρά μαθηματικό αλγόριθμο, το αντικειμενοστραφές μοντέλο βασίζεται στην αναπαράσταση της ζωής και του περιβάλλοντος ενός αντικειμένου.

Αναφέρουμε συνοπτικά κάποια από τα προβλήματα που θέτει το O-R impedance mismatch. Αυτά τα προβλήματα καλούνται να λύσουν ή να μετριάσουν τα ORM εργαλεία.

- Το πρόβλημα της κληρονομικότητας και του πολυμορφισμού. Ενώ οι αντικειμενοστραφείς γλώσσες προγραμματισμού υποστηρίζουν την έννοια της κληρονομικότητας, οι σχεσιακές βάσεις δεδομένων δεν την υποστηρίζουν. Έτσι, δεν υπάρχει ευθύς τρόπος της αποθήκευσης μιας ιεραρχίας κλάσεων και υποκλάσεων σε μια βάση δεδομένων.
- Με την ίδια αναλογία, υπάρχει το πρόβλημα της πρόσβασης στα αντικείμενα. Ενώ στο αντικειμενοστραφές μοντέλο προγραμματισμού, η πρόσβαση σε ένα αντικείμενο μπορεί να διαφοροποιηθεί ανάλογα με τη διεπαφή (interface) που χρησιμοποιείται προς αυτό, στις βάσεις δεδομένων πρέπει να χρησιμοποιηθούν όψεις (views) για την διαφοροποίηση των προοπτικών με τις οποίες βλέπουμε τα δεδομένα.

- Το πρόβλημα των σχέσεων. Στα μοντέλα αντικειμένων, ένα αντικείμενο μπορεί να σχετιστεί με ένα άλλο, χρησιμοποιώντας μια απλή αναφορά (object reference). Όμως στις σχεσιακές βάσεις δεδομένων, μια σχέση αναπαριστάται μέσω ενός ξένου κλειδιού (foreign key). Οι διαφορές μεταξύ των δύο είναι λεπτές και μικρές, όπως η κατευθυντικότητα.
- Το πρόβλημα των τύπων δεδομένων. Οι βαθμωτοί τύποι δεδομένων σε μια αντικειμενοστραφή γλώσσα προγραμματισμού διαφέρουν συνήθως από τους βαθμωτούς τύπους που μπορεί να αποθηκεύσει μια σχεσιακή βάση δεδομένων. Ένα παράδειγμα είναι η διαφορετική αντιμετώπιση ενός String. Ενώ στις βάσεις δεδομένων τα strings πρέπει να έχουν collation και να έχουν κάποιο μέγιστο μέγεθος, στις αντικειμενοστραφείς γλώσσες δεν υπάρχουν αυτοί οι περιορισμοί γιατί το collation χρειάζεται μόνο σε ορισμένες λειτουργίες (όπως η ταξινόμηση) ενώ το μέγεθος του String αυξομειώνεται στη μνήμη.
- Το πρόβλημα της ταυτότητας. Ενώ στις σχεσιακές βάσεις δεδομένων η ταυτότητα μιας γραμμής εκφράζεται από την τιμή του πρωτεύοντος κλειδιού (primary key), στα αντικείμενα η ταυτότητα συνήθως βασίζεται στην τοποθεσία του αντικειμένου στη μνήμη ή σε μια προσαρμοσμένη μέθοδο ισότητας αντικειμένων.
- Το πρόβλημα της ενθυλάκωσης. Τα αντικειμενοστραφή προγράμματα σχεδιάζονται με μεθόδους που καταλήγουν σε ενθυλακωμένα αντικείμενα των οποίων οι λεπτομέρειες είναι κρυμμένες. Αντίθετα, στις σχεσιακές βάσεις δεδομένων δεν υπάρχουν περιορισμοί για τη σχεδίαση ενθυλακωμένων αντικειμένων που έχουν κρυμμένες λεπτομέρειες, αλλά τα δεδομένα είναι ελεύθερα προς χρήση. Κατ' αυτήν την έννοια, η απεικόνιση των κρυμμένων λεπτομερειών σε μια βάση δεδομένων, καθιστά τις συγκεκριμένες βάσεις δεδομένων ευαίσθητες σύμφωνα με τις αρχές του αντικειμενοστραφούς προγραμματισμού.
- Το πρόβλημα των μηχανισμών προστασίας των ενθυλακωμένων αντικειμένων. Τα RDBMS τείνουν να χρησιμοποιούν μηχανισμούς προστασίας που στηρίζονται σε κανόνες και ρόλους. Κατά την αρχικοποίηση των δεδομένων, τους δίνεται ένα προκαθορισμένο σύνολο από κανόνες και λειτουργίες. Για την τροποποίηση των κανόνων, πρέπει να «αφαιρεθούν» κάποιοι κανόνες από το προκαθορισμένο σετ, γι' αυτό λέμε ότι τα RDBMS στηρίζονται σε «αφαιρετικούς» μηχανισμούς προστασίας. Αντίθετα, στο αντικειμενοστραφές μοντέλο, ένα ενθυλακωμένο αντικείμενο δεν είναι προσβάσιμο από τον έξω κόσμο μέχρι να οριστεί μια διεπαφή (interface) που να παρέχει αυτήν την πρόσβαση.
- Το πρόβλημα των συναλλαγών. Ενώ οι σχεσιακές βάσεις δεδομένων υποστηρίζουν τις μεγάλες συναλλαγές, στον OOP οι συναλλαγές είναι πολύ μικρές και

περιορίζονται συνήθως στην ανάθεση μιας τιμής. Στον OOP δεν υπάρχουν οι ανάλογες έννοιες της απομόνωσης και της μονιμότητας.

3.1.4 Χαρακτηριστικά των ORM συστημάτων

[Mic10c]

Η αυτόματη απεικόνιση των αντικειμένων μιας αντικειμενοστραφούς εφαρμογής σε μια σχεσιακή βάση δεδομένων προσφέρει πολλά πλεονεκτήματα έναντι άλλων τεχνικών πρόσβασης δεδομένων. Μερικά από τα χαρακτηριστικά και τα πλεονεκτήματα των ORM συστημάτων είναι τα εξής:

- Δραματική μείωση του χρόνου ανάπτυξης λογισμικού, του κόστους ανάπτυξης και του κόστους συντήρησης, γιατί το ORM αυτοματοποιεί τη διαδικασία του persistency των αντικειμένων. Ουσιαστικά αναλαμβάνει όλο το DAL που παρουσιάσαμε παραπάνω. Ο προγραμματιστής δεν χρειάζεται να ασχοληθεί με το γράψιμο πολύπλοκων SQL ερωτημάτων και την υλοποίηση του DAL επιπέδου.
- Δίνει τη δυνατότητα στον προγραμματιστή να μοντελοποιήσει τις οντότητες της εφαρμογής βασιζόμενος στις πραγματικές business έννοιες και ανάγκες παρά στην δομή της βάσης δεδομένων.
- Λιγότερο κώδικα σε σύγκριση με την συγγραφή SQL ερωτημάτων μέσα στον κώδικα. Βοηθάει τον προγραμματιστή να επικεντρωθεί στη business λογική της εφαρμογής παρά στην δημιουργία των CRUD SQL ερωτημάτων.
- Μεγαλύτερη ευελιξία στην δημιουργία ερωτημάτων. Τα ORM εργαλεία προσφέρουν συνήθως μια αντικειμενοστραφή γλώσσα ερωτημάτων. Οι γλώσσες αυτές επικεντρώνονται στο μοντέλο αντικειμένων παρά στη μορφή της δομής μιας βάσης δεδομένων. Το ORM εργαλείο θα μεταφράσει αυτόματα την γλώσσα αυτή σε κατάλληλα SQL ερωτήματα.
- Διαφανής πλοήγηση μεταξύ των σχετιζόμενων αντικειμένων (lazy loading). Τα συνδεδεμένα αντικείμενα φορτώνονται δυναμικά από το ORM (ανακτώνται δυναμικά από τη βάση δεδομένων) καθώς ο χρήστης πλοηγείται σε αυτά. Έτσι βελτιστοποιείται η χρήση της μνήμης του εξυπηρετητή.
- Συγχρονισμός. Υποστηρίζονται πολλαπλοί χρήστες που τροποποιούν τα ίδια δεδομένα ταυτόχρονα.
- Caching. Τα αντικείμενα γίνονται cached στην μνήμη για την καλύτερη απόδοση της εφαρμογής και για την μείωση του φόρτου της βάσης δεδομένων.
- Υποστήριξη για συναλλαγές (transactions): Όλες οι αλλαγές σε αντικείμενα γίνονται μέσα σε μια συναλλαγή. Όλη η συναλλαγή είτε θα γίνει committed είτε rolled back.

Πολλές συναλλαγές μπορούν να πραγματοποιούνται ταυτόχρονα και οι αλλαγές της μίας είναι απομονωμένες από τις αλλαγές της άλλης.

- Διαφανής διατήρηση (transparent persistence): Το ORM εργαλείο παρακολουθεί τις αλλαγές που γίνονται στα αντικείμενα. Όταν κληθεί έπειτα να τα αποθηκεύσει, ξέρει τι αλλαγές πρέπει να κάνει στη βάση δεδομένων και δημιουργεί αυτόματα τα CRUD ερωτήματα που πρέπει να εκτελεστούν.
- Μη παρεμβατικό persistence: Ένα καλό ORM εργαλείο δεν θέτει περιορισμούς όσον αφορά τα αντικείμενα του business model της εφαρμογής. Δηλαδή τα αντικείμενα δεν πρέπει να χρειάζεται να κληρονομήσουν κάποια συγκεκριμένη κλάση ή interface.
- Υποστήριξη πολλών RDBMS όπως Oracle, DB2, Microsoft SQL Server, PostgreSQL, MySQL κ.α.

Βέβαια πρέπει να τονίσουμε ότι τα ORM εργαλεία δεν είναι πάντα η καλύτερη λύση για την υλοποίηση του DAL επιπέδου μια εφαρμογής. Για παράδειγμα, για μικρές εφαρμογές, η διαδικασία της ρύθμισης της σωστής απεικόνισης των αντικειμένων σε πίνακες καθώς και η διαδικασία της βελτίωσης της αποδοτικότητας των SQL ερωτημάτων που παράγονται αυτόματα, ίσως είναι περισσότερο χρονοβόρες από την χρησιμοποίηση απλούστερων τεχνικών όπως SQL ερωτημάτων. Από την άλλη, για μεγάλες business εφαρμογές, πρέπει να ζυγιστούν κατάλληλα τα πλεονεκτήματα των ORM εργαλείων που αναφέραμε παραπάνω.

3.2 ADO.NET Entity Framework (EF)

[Wik10i], [Ler09], [Mic06]

3.2.1 Γενικά

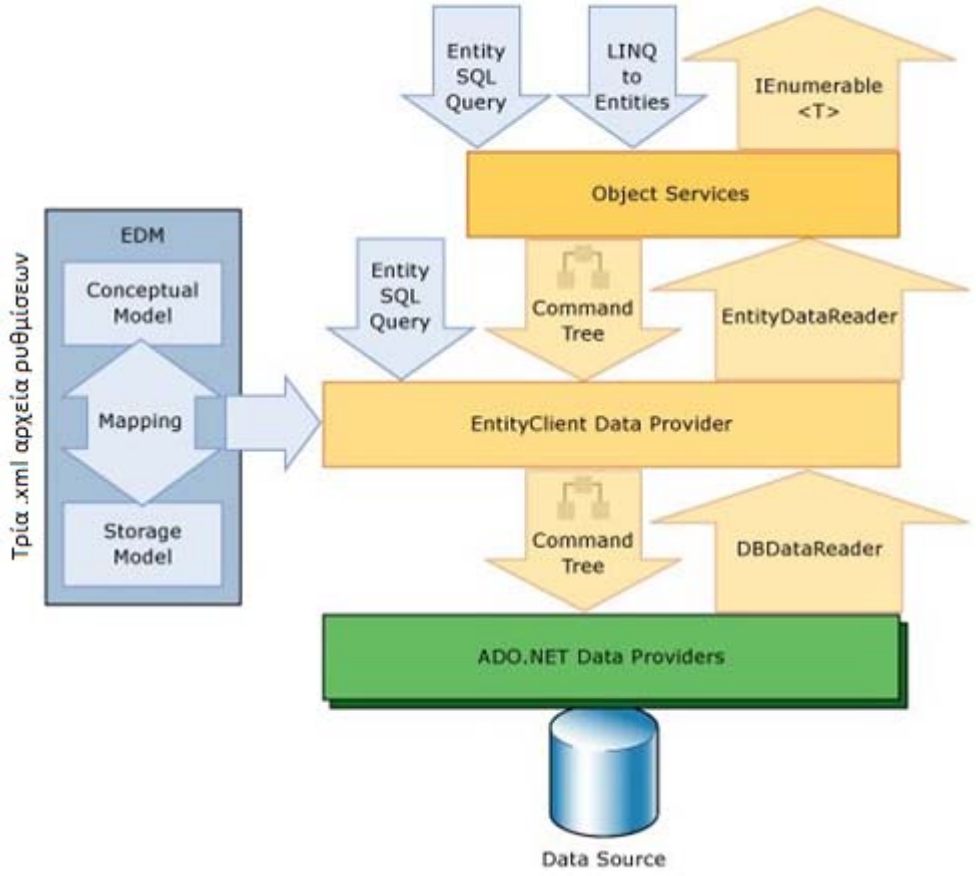
Το ADO.NET Entity Framework είναι ένα από τα καινούργια εργαλεία ORM που κυκλοφόρησε η Microsoft τον Αύγουστο του 2008 με το Visual Studio 2008 Service Pack 1 και το .NET Framework 3.5 Service Pack 1.

Το EF στοχεύει να «αφαιρέσει» το σχεσιακό σχήμα (relational schema) των δεδομένων που αποθηκεύονται στη βάση δεδομένων και να παρουσιάσει στην εφαρμογή το εννοιολογικό σχήμα (conceptual schema) των αντικειμένων. Δηλαδή η εφαρμογή δεν χρειάζεται να ασχοληθεί με το πως τα αντικείμενα αποθηκεύονται σε πίνακες, αλλά να ασχοληθεί απλά με τα business objects της.

Το EF χρησιμοποιεί ένα μοντέλο που ονομάζεται Entity Data Model (**EDM**). Το EDM ορίζει το εννοιολογικό σχήμα των δεδομένων μέσω ενός Entity-Relationship Model (ERM), το οποίο συντίθεται από τις οντότητες (entities) και τις σχέσεις μεταξύ τους (relationships).

Επιπρόσθετα του EDM, πρέπει να οριστεί και η απεικόνιση των στοιχείων του εννοιολογικού σχήματος στο σχεσιακό σχήμα.

Η γενική αρχιτεκτονική του EF φαίνεται στο παρακάτω σχήμα.



Εικόνα 3-1 Γενική αρχιτεκτονική του ADO.NET Entity Framework

Η οντότητα (entity) είναι μία κύρια έννοια του EDM. Μια οντότητα ορίζει το σχήμα ενός αντικειμένου αλλά όχι και τη συμπεριφορά του. Μάλιστα η εφαρμογή μπορεί να έχει πρόσβαση στις οντότητες μέσω διάφορων τρόπων:

- Ως συνήθη .NET αντικείμενα. Η υπηρεσία «Object Services» του EF αυτόματα παράγει .NET αντικείμενα που εκθέτουν τις ίδιες ιδιότητες με μια οντότητα, επιτρέποντας σε μια αντικειμενοστραφή εφαρμογή να έχει πρόσβαση στην οντότητα με τη μορφή .NET αντικειμένων.
- Ως Web Services.

Τα κύρια χαρακτηριστικά του EF είναι τα εξής:

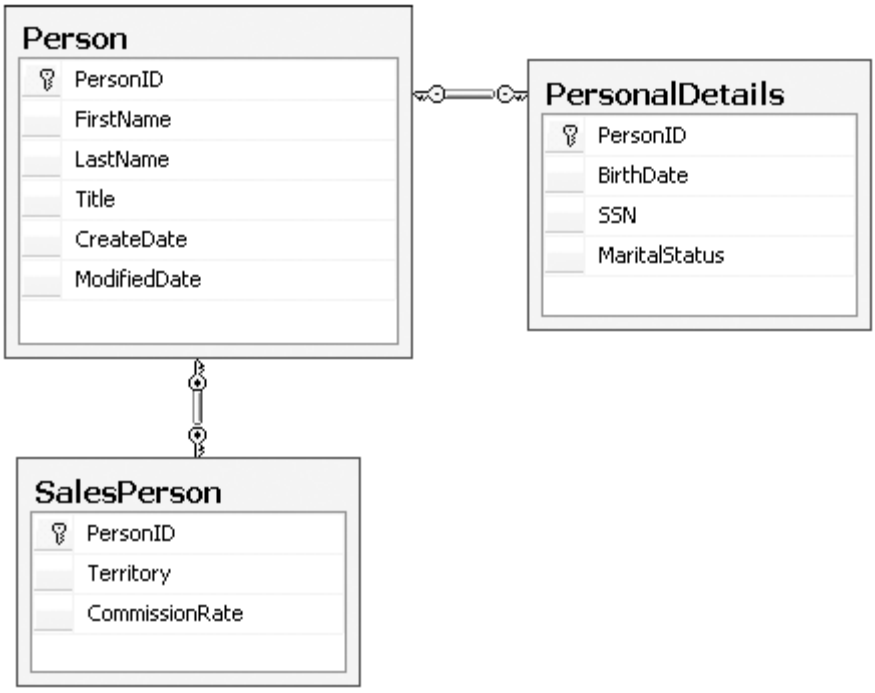
- Δημιουργεί αυτόματα κλάσεις από το EDM και τις ενημερώνει δυναμικά αν το EDM αλλάξει.
- Το EDM ορίζεται μέσω 3 XML αρχείων όπως περιγράφονται στην Εικόνα 3-1.

- Περιέχει ένα σύνολο από σχεδιαστικά εργαλεία για τη σχεδίαση του EDM. Ο EDM Designer μας δίνει τη δυνατότητα να επεξεργαστούμε οπτικά το EDM παρά με τα XML αρχεία. Ο EDM Wizard μας δίνει τη δυνατότητα αυτόματης παραγωγής των XML αρχείων και του EDM από μια βάση δεδομένων.
- Παρέχει μια γλώσσα ερωτημάτων (Entity SQL ή eSQL) η οποία στοχεύει το εννοιολογικό μοντέλο, δηλαδή στοχεύει τις οντότητες, και όχι το σχεσιακό μοντέλο.
- LINQ-to-Entities: Παρέχει επέκταση μιας .NET γλώσσας για τη σύνταξη ενός ερωτήματος μέσα στον ίδιο τον κώδικα, χωρίς να γραφεί eSQL μέσα σε Strings. Μάλιστα με τη συνδυασμένη χρήση του EF και της LINQ, η εφαρμογή μπορεί να μην χρησιμοποιήσει καμία γλώσσα τύπου SQL, και ο κώδικας να φαίνεται ως καθαρά business λογική.
- Η υπηρεσία "Object Services" αυτόματα εντοπίζει τις αλλαγές στα αντικείμενα του μοντέλου, καθώς η εφαρμογή τα αλλάζει, και αναλαμβάνει να πραγματοποιήσει τα κατάλληλα CRUD ερωτήματα για να ανανεώσει τα αντίστοιχα δεδομένα της βάσης δεδομένων.
- Η υπηρεσία "Object Services" διαχειρίζεται και τις συσχετίσεις μεταξύ των οντοτήτων. Μάλιστα οι συσχετίσεις μεταξύ των αντικειμένων (που συντίθενται από τις οντότητες) είναι και αυτές αντικείμενα.
- Υποστήριξη Data Binding σε .NET εφαρμογές.
- Το EntityClient είναι ένα επίσης σημαντικό API του EF. Παρέχει όλη τη λειτουργικότητα για την εργασία με τη βάση δεδομένων και το μετασχηματισμό των δεδομένων ώστε να ταιριάζουν με το EDM. Μπορούμε να εργαστούμε είτε με το EntityClient είτε με τα Object Services, τα οποία βρίσκονται σε υψηλότερο επίπεδο από το EntityClient. Η διαφορά έγκειται στο ότι στην περίπτωση που δουλεύουμε με το EntityClient, τα αποτελέσματα είναι read-only και σε μορφή πίνακα, ενώ όταν δουλεύουμε με τα Object Services, τα αποτελέσματα μετατρέπονται σε τροποποιήσιμα αντικείμενα.

3.2.2 Σχεσιακό και εννοιολογικό σχήμα

Όπως είπαμε, το EF προσπαθεί να «αφαιρέσει» το σχεσιακό σχήμα, το οποίο ανήκει στη βάση δεδομένων, και να παρουσιάσει στην εφαρμογή μόνο το εννοιολογικό σχήμα.

Τα δύο σχήματα δεν είναι ίδια. Τα αντικείμενα μπορούν να μοντελοποιηθούν με διαφορετικό τρόπο στη βάση δεδομένων. Ας πάρουμε για παράδειγμα το εξής κανονικοποιημένο σχεσιακό σχήμα μιας βάσης δεδομένων:

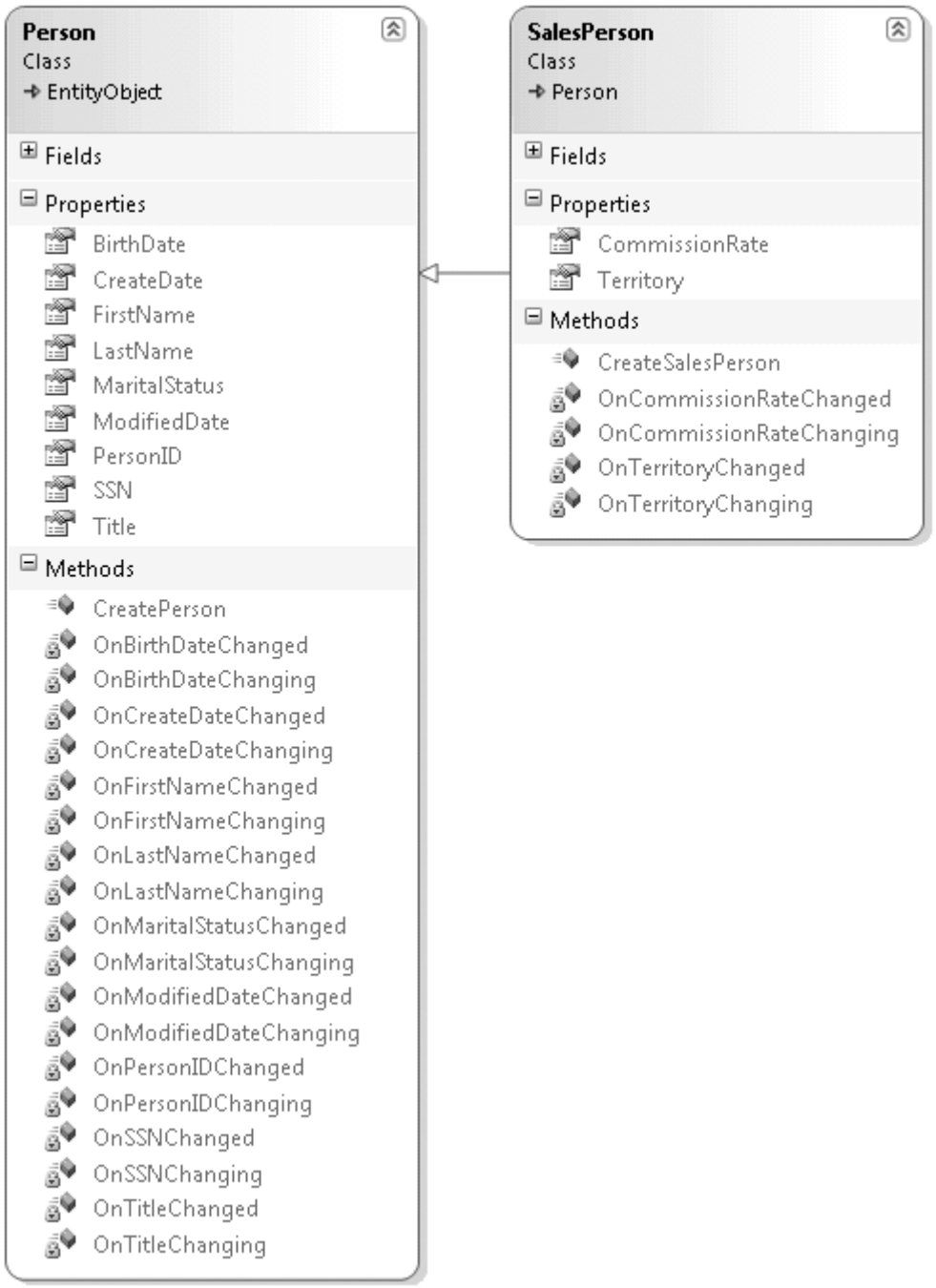


Εικόνα 3-2 Παράδειγμα σχεσιακού σχήματος μιας βάσης δεδομένων

Το σχήμα αυτό περιγράφει τα δεδομένα ενός προσώπου. Ο πίνακας PersonalDetails περιέχει λεπτομέρειες του προσώπου. Ενώ ο πίνακας SalesPerson περιέχει πρόσωπα τα οποία είναι και πωλητές. Οι πωλητές έχουν τις επιπλέον ιδιότητες που φαίνονται στον πίνακα.

Στον κώδικα της εφαρμογής, για να μπορέσουμε να πλοηγηθούμε στους πίνακες της βάσης δεδομένων και να ανακτήσουμε τα απαραίτητα δεδομένα για ένα άτομο, θα πρέπει να γράψουμε SQL ερωτήματα τα οποία θα είναι γεμάτα inner και outer joins.

Ας περάσουμε τώρα στο εννοιολογικό σχήμα που θέλει να ορίσει η εφαρμογή. Το εννοιολογικό σχήμα δεν χρειάζεται να είναι κατ' ανάγκην 1:1 με το σχεσιακό σχήμα. Το εννοιολογικό σχήμα στοχεύει στον καθορισμό των οντοτήτων που θέλει να χρησιμοποιήσει η εφαρμογή. Για παράδειγμα, ένα εννοιολογικό σχήμα είναι το εξής:



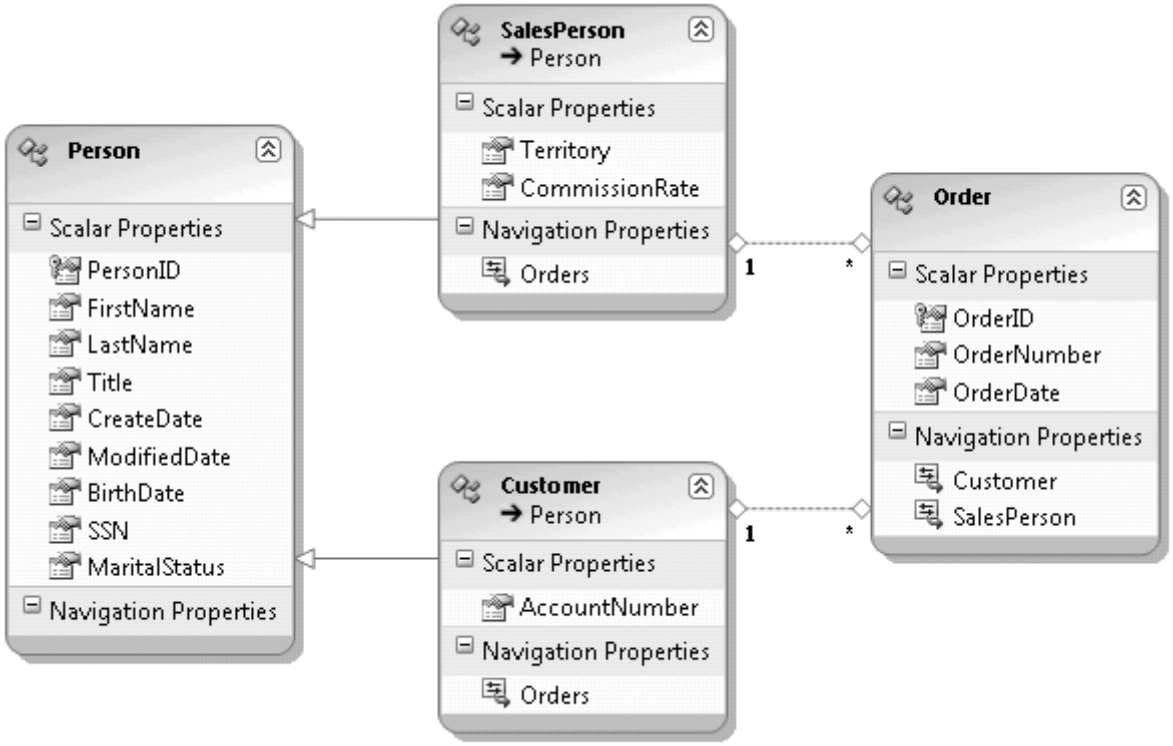
Εικόνα 3-3 Παράδειγμα εννοιολογικού σχήματος

Οι οντότητες διαφέρουν από τα κοινά αντικείμενα του business model, γιατί απλώς περιγράφουν την δομή ενός αντικειμένου και όχι την συμπεριφορά του. Περιέχουν μόνο μεθόδους που επιτρέπουν τον εντοπισμό των αλλαγών των ιδιοτήτων (properties) της οντότητας.

Για την πρόσθεση επιχειρηματικής λογικής στις οντότητες, η εφαρμογή πρέπει να χρησιμοποιήσει τα “Object Services” του EF, ώστε να αντιστοιχήσει τις παραπάνω οντότητες

στα business objects που θα χρησιμοποιήσει. Τα business objects περιέχουν την επιχειρηματική λογική που ορίζει η εφαρμογή.

Επιπλέον, μπορούν να ορισθούν σχέσεις μεταξύ των οντοτήτων. Για παράδειγμα, στο παρακάτω σχήμα έχει προστεθεί μια οντότητα Customer (που προέρχεται από το Person) και μια οντότητα Order. Φαίνονται και οι σχέσεις που έχουμε ορίσει.

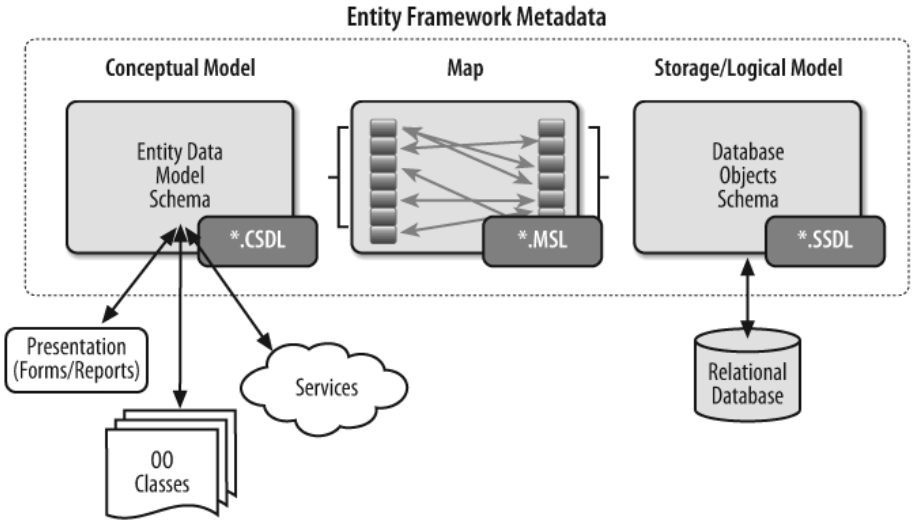


Εικόνα 3-4 Ορισμός σχέσεων μεταξύ των οντοτήτων

3.2.3 XML Schemas του EDM

Όπως δείξαμε στην Εικόνα 3-1, το EDM αποτελείται από τρία διαφορετικά XML αρχεία:

- Το Conceptual Schema, γραμμένο σε CSDL (Conceptual Schema Definition Language). Περιγράφει τις οντότητες και τις σχέσεις, δηλαδή το εννοιολογικό σχήμα των δεδομένων.
- Το Storage/Logical Schema, γραμμένο σε SSDL (Stored Schema Definition Language). Περιγράφει το σχεσιακό σχήμα των δεδομένων, δηλαδή την αναπαράσταση τους στη βάση δεδομένων.
- Το Mapping Layer, γραμμένο σε MSL (Mapping Specification Language). Περιγράφει την απεικόνιση του εννοιολογικού σχήματος στο σχεσιακό σχήμα.



Εικόνα 3-5 Τα τρία αρχεία ρυθμίσεων του EDM

3.3 NHibernate

[Kua09], [NHi10], [Wik10j]

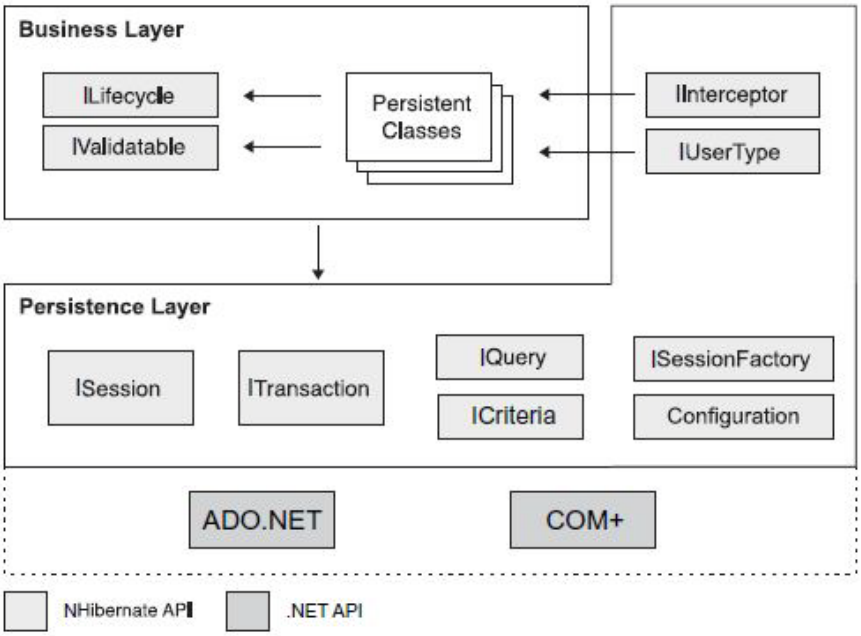
3.3.1 Γενικά

Το NHibernate είναι ένα ORM εργαλείο για το .NET Framework. Είναι δωρεάν λογισμικό και ανοιχτού κώδικα. Το NHibernate είναι μια μεταφορά του ORM εργαλείου Hibernate της Java στο .NET Framework.

Το NHibernate υποστηρίζει τη διαφανή διατήρηση των Plain Old CLR Objects (POCOs). Τα POCOs αναφέρονται σε απλά .NET αντικείμενα. Δηλαδή το NHibernate δεν απαιτεί από τα αντικείμενα μιας εφαρμογής να κληρονομούν κάποια ειδική κλάση ή να υλοποιούν κάποια ειδική διεπαφή (interface) ώστε να μπορεί να τα αποθηκεύσει. Οι μόνοι περιορισμοί που τίθενται από το NHibernate για τα POCOs είναι οι εξής:

- Να έχουν έναν constructor χωρίς παραμέτρους, όχι κατ' ανάγκη public.
- Για τη σωστή συμπεριφορά σε σχέση με αρκετές λειτουργίες του NHibernate που βασίζονται στην ταυτότητα των αντικειμένων, ίσως χρειαστεί να γίνουν overridden οι μέθοδοι Equals() και GetHashCode(). Το ζήτημα αυτό περιγράφεται αρκετά παρακάτω, στην ενότητα 3.3.5.6 «Το πρόβλημα της ταυτότητας και της ισότητας».

Η αρχιτεκτονική του NHibernate και της εφαρμογής που το χρησιμοποιεί, φαίνεται στο παρακάτω σχήμα.



Εικόνα 3-6 Αρχιτεκτονική του NHibernate και της εφαρμογής που το χρησιμοποιεί

Το Business Layer περιέχει ουσιαστικά τα POCOs (στιγμιότυπα των "Persistent Classes") τα οποία καλείται το NHibernate να αποθηκεύσει.

Το Persistence Layer περιέχει τις διεπαφές (interfaces) και τις κλάσεις με τις οποίες αλληλεπιδρά η εφαρμογή με το NHibernate. Τις σημαντικότερες διεπαφές θα περιγράψουμε σε επόμενη ενότητα.

Το NHibernate κάνει χρήση υπαρχόντων .NET APIs όπως το ADO.NET και το ITransaction API του. Το ADO.NET προσφέρει ένα πρωταρχικό επίπεδο αφαιρετικότητας και λειτουργικότητας που είναι κοινό για τις βάσεις δεδομένων και επιτρέπει στο NHibernate να υποστηρίξει σχεδόν οποιαδήποτε βάση δεδομένων απλώς συνδέοντας τον κατάλληλο ADO.NET driver.

Οι ρυθμίσεις απεικόνισης (mapping) των κλάσεων και των σχέσεων μεταξύ των κλάσεων στη βάση δεδομένων, πρέπει να γίνει από την εφαρμογή. Αυτό γίνεται κυρίως με δύο τρόπους:

- Με τη χρησιμοποίηση κατάλληλων XML αρχείων. Τα αρχεία αυτά έχουν συνήθως κατάληξη .hbm.xml. Σε αυτόν τον τρόπο ρύθμισης θα επικεντρωθούμε.
- Με τη χρησιμοποίηση κατάλληλων .NET Attributes που πρέπει να κοσμούν τις κλάσεις που θα απεικονίσει το NHibernate στη σχεσιακή βάση δεδομένων.

3.3.2 Γιατί επιλέξαμε το NHibernate

Το NHibernate είναι ένα πλήρες ORM εργαλείο το οποίο χρησιμοποιείται αρκετό καιρό από πολλούς επαγγελματίες .NET προγραμματιστές. Είναι κοινά αποδεκτό ότι το NHibernate

είναι πιο ώριμο ως προϊόν και διαθέτει περισσότερα ORM χαρακτηριστικά από το Entity Framework.

Το Entity Framework είναι ακόμα σε πρώιμο στάδιο. Η πρώτη έκδοση του EF είχε μερικά αρνητικά χαρακτηριστικά. Ένα από αυτά ήταν ότι το GUI το οποία έφτιαχνε αυτόματα τις κλάσεις και τα αρχεία απεικόνισης δεν ήταν πλήρες, και τα παραγόμενα αρχεία είχαν πάρα πολλές γραμμές κώδικα που τα καθιστούσαν δύσκολα στη συντήρηση. Άλλο ένα σημαντικό χαρακτηριστικό ήταν ότι δεν παρείχε αυτόματο lazy loading των σχετιζόμενων αντικειμένων: Όταν ο προγραμματιστής ανακτούσε στιγμιότυπα μιας κλάσης, έπρεπε να φορτώσει χειρονακτικά τυχόν σχετιζόμενες κλάσεις.

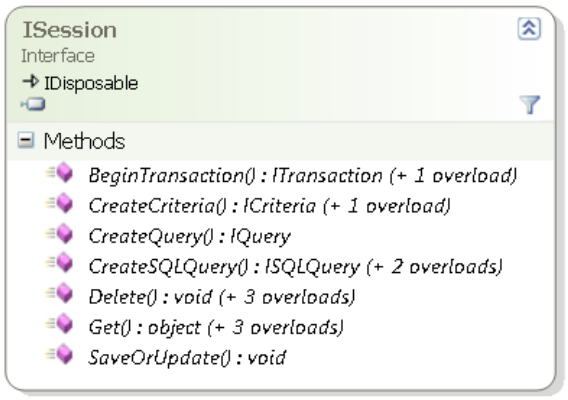
Το EF μόλις έχει φτάσει την 2η έκδοση του, η οποία εκδίδεται με το Visual Studio 2010 και το .NET Framework 4.0 και βελτιώνει πολλά από τα αρνητικά χαρακτηριστικά της πρώτης. Παρόλα αυτά, το NHibernate έχει περάσει τη δοκιμασία του χρόνου. Ακόμα και χωρίς την (επίσημη) υποστήριξη με γραφικό περιβάλλον, η κοινότητα που έχει δημιουργήσει είναι πολύ ενεργή. Μάλιστα προσφέρει μια πιο ξεκάθαρη απεικόνιση των αντικειμένων στη σχεσιακή βάση δεδομένων. Επίσης, προσφέρει περισσότερες δυνατότητες ρυθμίσεων και παραμετροποίησης απ᾽ ότι η πρώτη έκδοση του EF.

3.3.3 Περιγραφή των διεπαφών

Επικεντρωνόμαστε κυρίως στις πιο σημαντικές διεπαφές και στις πιο σημαντικές μεθόδους τους που χρησιμοποιήσαμε κατά την υλοποίηση της Υπηρεσίας Αποθήκευσης του NCommet.

3.3.3.1 ISession

Το ISession είναι η σημαντικότερη διεπαφή. Εκθέτει τις μεθόδους του NHibernate για εύρεση, αποθήκευση, ενημέρωση και διαγραφή αντικειμένων. Η λέξη “session” (ή συνεδρία) υπονοεί ένα τμήμα εργασίας (unit of work) μέσα στο οποίο δουλεύουμε με μια συλλογή αντικειμένων. Το NHibernate μπορεί να εντοπίζει τις αλλαγές που γίνονται στα αντικείμενα μέσα στο session ώστε να μπορέσει να τα ενημερώσει αυτόματα στη βάση δεδομένων όταν του ζητηθεί.



Εικόνα 3-7 Διάγραμμα κλάσης της διεπαφής ISession

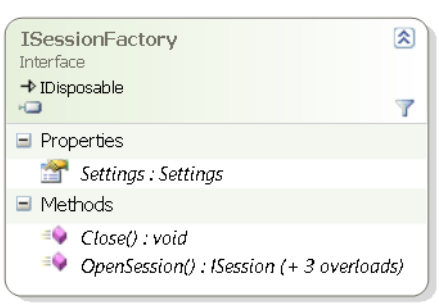
Σημαντικές μέθοδοι:

- Delete(object obj): Διαγράφει ένα persistent αντικείμενο από τη βάση δεδομένων.
- Get<T>(object id): Γυρίζει το αντικείμενο που αντιστοιχεί στο δοσμένο αναγνωριστικό (μπορεί να είναι και σύνθετο αναγνωριστικό).
- SaveOrUpdate(object obj): Είτε κάνει Save() το αντικείμενο είτε Update(), ανάλογα με το αν είναι persistent το αντικείμενο ή παροδικό.

3.3.3.2 ISessionFactory

Το ISessionFactory δημιουργεί στιγμιότυπα του ISession. Συνήθως υπάρχει ένα ISessionFactory ανά εφαρμογή, το οποίο δημιουργείται στην αρχή της εφαρμογής, μετά τη ρύθμιση του NHibernate.

Όταν η εφαρμογή θέλει να επικοινωνήσει με το NHibernate, ζητάει από το ISessionFactory ένα καινούργιο ISession με το οποίο θα δουλέψει ένα unit of work του, δηλαδή ένα συγκεκριμένο κομμάτι δουλειάς του.

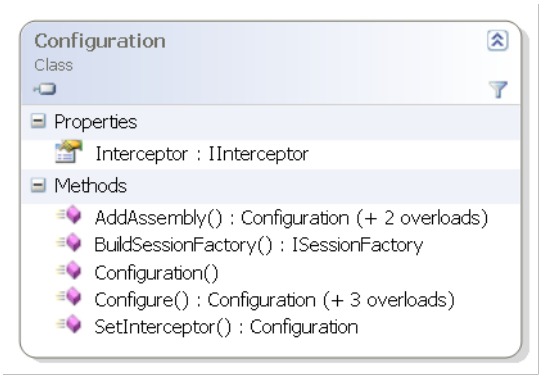


Εικόνα 3-8 Διάγραμμα κλάσης της διεπαφής ISessionFactory

3.3.3.3 Configuration

Η κλάση αυτή χρησιμοποιείται στην αρχή μιας εφαρμογής για να ρυθμίσει το NHibernate, να φορτώσει τις ρυθμίσεις απεικόνισης (από τα .hbm.xml αρχεία) και να τροποποιήσει

συγκεκριμένες ρυθμίσεις του NHibernate. Έπειτα χρησιμοποιείται για τη δημιουργία ενός ISessionFactory.



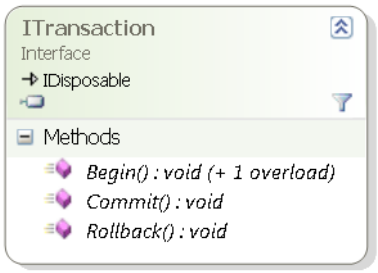
Εικόνα 3-9 Διάγραμμα κλάσης του Configuration

Σημαντικές μέθοδοι:

- AddAssembly(Assembly assembly): Φορτώνει τις ρυθμίσεις απεικόνισης από τα .hbm.xml αρχεία που έχουν γίνει embedded στο δοσμένο assembly.
- BuildSessionFactory(): Καλείται στην αρχή του προγράμματος για να ρυθμίσει το NHibernate και να δημιουργήσει ένα ISessionFactory με τις παρούσες ρυθμίσεις. Πριν κληθεί η μέθοδος, πρέπει να επιβεβαιωθεί ότι έχουν γίνει οι κατάλληλες ρυθμίσεις στο app.config αρχείο ρυθμίσεων της εφαρμογής και επίσης ότι έχει κληθεί η μέθοδος AddAssembly() για να φορτωθούν τα κατάλληλα .hbm.xml αρχεία.
- SetInterceptor(IInterceptor interceptor): Ορίζει την υλοποίηση της διεπαφής IInterceptor που θα ενημερώνεται για τυχόν συμβάντα (events) του NHibernate.

3.3.3.4 ITransaction

Πρόκειται για μια προαιρετική διεπαφή η οποία προσφέρει ένα ενιαίο API όσον αφορά τη διαχείριση συναλλαγών από την εφαρμογή. Το NHibernate μπορεί έτσι να χρησιμοποιήσει διαφορετικές τεχνολογίες για την υποστήριξη των συναλλαγών, είτε του ADO.NET είτε κάποια άλλη.



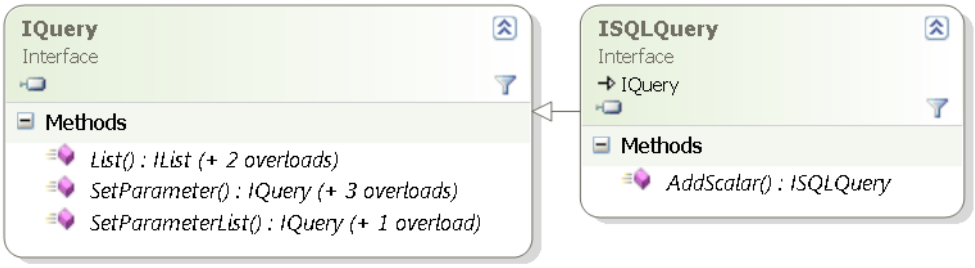
Εικόνα 3-10 Διάγραμμα κλάσης της διεπαφής ITransaction

3.3.3.5 IQuery

Το IQuery δίνει τη δυνατότητα στην εφαρμογή να εκτελέσει ερωτήματα προς το NHibernate και προς τη βάση δεδομένων.

Τα ερωτήματα μπορεί να είναι γραμμένα σε:

- HQL (Hibernate Query Language). Το IQuery πρέπει να δημιουργηθεί μέσω της μεθόδου ISession.CreateQuery(). Στην ενότητα 3.3.8 «Hibernate Query Language (HQL)» θα παρουσιάσουμε, εν συντομία, την HQL.
- SQL της βάσης δεδομένων. Το ISQLQuery (υποκλάση του IQuery) πρέπει να δημιουργηθεί μέσω της μεθόδου ISession.CreateSQLQuery().



Εικόνα 3-11 Διάγραμμα κλάσης της διεπαφής IQuery και ISQLQuery

Για την εκτέλεση του ερωτήματος καλείται απλά η μέθοδος List() η οποία εκτελεί το ερώτημα και γυρίζει το αποτέλεσμα σε μορφή IList. Η μέθοδος προσφέρεται και σε generic μορφή και γυρίζει αποτελέσματα σε μορφή IList<T>.

Το ερώτημα είτε σε HQL είτε σε SQL μπορεί να περιέχει ονομαστικές παραμέτρους. Οι παράμετροι τοποθετούνται μέσα στο ερώτημα με τη μορφή «:parameterName». Πριν την εκτέλεση του ερωτήματος, πρέπει να τεθούν οι τιμές των παραμέτρων χρησιμοποιώντας τις μεθόδους SetParameter() και SetParameterList().

3.3.3.6 ICriteria

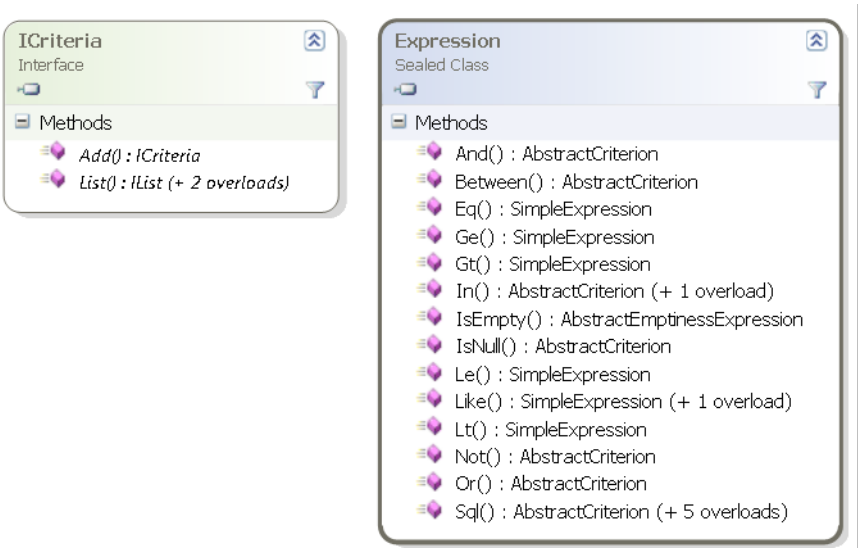
Το ICriteria δίνει και αυτό τη δυνατότητα δημιουργίας ερωτημάτων για τη βάση δεδομένων. Όμως τα ερωτήματα δεν γράφονται ως κείμενο, αλλά δημιουργούνται βάσει αντικειμενοστραφών κριτηρίων.

Για παράδειγμα, αν θέλουμε να βρούμε έναν χρήστη με ένα όνομα, μπορούμε να εκτελέσουμε το εξής ICriteria:

```
ICriteria criteria = session.CreateCriteria(typeof(User));
criteria.Add(Expression.Like("name", "A Name"));
IList result = criteria.List();
```

Το θετικό του ICriteria είναι ότι ο προγραμματιστής μπορεί να προσθέσει κριτήρια δυναμικά, διαχειρίζοντας αντικείμενα κριτηρίων στο χρόνο εκτέλεσης. Από την άλλη, ο κώδικας με κριτήρια είναι λιγότερο ευανάγνωστος από ένα HQL IQuery ερώτημα.

Τα κριτήρια μπορούν να δημιουργηθούν εύκολα μέσω μεθόδων της στατικής κλάσης NHibernate.Expression.Expression.



Εικόνα 3-12 Διάγραμμα κλάσης της διεπαφής ICriteria και της κλάσης Expression

3.3.4 Ρυθμίσεις απεικόνισης

Η εφαρμογή πρέπει να ορίσει σε αρχεία .hbm.xml τις ρυθμίσεις απεικόνισης των αντικειμένων της στη βάση δεδομένων. Έτσι γνωστοποιείται στο NHibernate ο τρόπος με τον οποίο πρέπει να αποθηκεύει και να ανακτά τα αντικείμενα από τη βάση δεδομένων.

Οι ρυθμίσεις απεικόνισης του NHibernate είναι πολλές ώστε να καλύπτουν ακόμα και τις πιο εξεζητημένες ανάγκες. Εμείς θα επικεντρωθούμε στις ρυθμίσεις που θα μας χρειαστούν κατά την υλοποίηση της Υπηρεσίας Αποθήκευσης του NCommet.

3.3.4.1 Απεικόνιση κλάσης

Συνήθως για κάθε κλάση δημιουργούμε το δικό της .hbm.xml αρχείο. Ο βασικός σκελετός του αρχείου είναι ο εξής:

```
<?xml version="1.0" encoding="utf-8" ?>
<hibernate-mapping xmlns="urn:nhibernate-mapping-2.2" auto-
import="true">
  <class name="ItemsExample.Item, ItemsExample" table="Items"
lazy="false">
    ...
  </class>
```

```
</hibernate-mapping>
```

Η ρύθμιση lazy=”true” σημαίνει ότι κατά την άντληση ενός στιγμιότυπου της κλάσης Item από τη βάση δεδομένων, δεν διαβάζονται άλλα στιγμιότυπα με τα οποία σχετίζεται (ή στα οποία αναφέρεται) το Item. Για παράδειγμα, αν είχαμε ορίσει μια ιδιότητα Children : IList<Item>, τότε δεν θα αντλούνταν τα παιδιά. Τα παιδιά θα αντληθούν δυναμικά και αυτόματα από το NHibernate μόλις η εφαρμογή προσπαθήσει να διαβάσει την ιδιότητα Children. Η ιδιότητα αυτή είναι ευρέως γνωστή ως «lazy loading». Η ρύθμιση lazy="false" κάνει το αντίθετο, φορτώνει τα παιδιά του στιγμιότυπου κατευθείαν.

3.3.4.2 Απεικόνιση ιδιοτήτων

Η βασικότερη ρύθμιση είναι η απεικόνιση των ιδιοτήτων και των πεδίων ενός αντικειμένου. Για παράδειγμα, για την απεικόνιση της ιδιότητας Name μιας κλάσης στη στήλη Name, χρησιμοποιούμε την εξής ρύθμιση:

```
<property name="Name" column="Name" />
```

Επιπλέον XML attributes της <property>:

- type: Ο τύπος που αντιστοιχεί στην ιδιότητα. Αν δεν διευκρινιστεί, το NHibernate θα προσπαθήσει να εντοπίσει αυτόματα τον τύπο χρησιμοποιώντας το Reflection του .NET Framework.
- access: property/field. Το access διευκρινίζει πως το NHibernate θα πρέπει να παίρνει και να θέτει την τιμή της ιδιότητας. Προκαθορισμένη τιμή είναι η property.
    - property: Σημαίνει ότι αναφερόμαστε σε μια ιδιότητα (property) του αντικειμένου. Το NHibernate χρησιμοποιεί τον getter και setter της ιδιότητας.
    - field: Σημαίνει ότι δεν αναφερόμαστε σε ιδιότητα της κλάσης, αλλά σε πεδίο του αντικειμένου. Το NHibernate τροποποιεί κατευθείαν το πεδίο.

3.3.4.3 Απεικόνιση αναγνωριστικού

Μια βασική έννοια στις σχεσιακές βάσεις δεδομένων είναι η ταυτότητα μιας γραμμής ενός πίνακα, η οποία εκφράζεται από την τιμή του πρωτεύοντος κλειδιού (primary key). Το πρωτεύον κλειδί μπορεί να είναι μια στήλη ή ένας συνδυασμός από στήλες των οποίων οι τιμές (σε συνδυασμό) πρέπει να είναι μοναδικές για κάθε γραμμή του πίνακα.

Συνήθως, στις business εφαρμογές, ορίζεται ως πρωτεύον κλειδί μια στήλη id, δηλαδή ένας αναγνωριστικός αριθμός για κάθε γραμμή. Μάλιστα, η τακτική αυτή είναι τόσο σύνηθες που τα RDBMS έχουν ειδικούς κανόνες για στήλες με αναγνωριστικούς αριθμούς.

Στις αντικειμενοστραφείς εφαρμογές που βασίζονται στο NHibernate, τα αντικείμενα πρέπει να εκθέτουν το primary key με το οποίο αποθηκεύονται στη βάση δεδομένων. Στην συνήθη

περίπτωση που το primary key είναι ένας αναγνωριστικός αριθμός, τότε αρκεί η εξής ρύθμιση:

```
<id name="id" column="id" access="field">
    <generator class="native" />
</id>
```

Επιπλέον XML attributes της <id>:

- Unsaved-value: any/none/null/id_value

  Δηλώνει την τιμή του αναγνωριστικού η οποία δηλώνει ότι το αντικείμενο είναι νέο και παροδικό, δηλαδή δεν έχει ακόμα αποθηκευτεί. Η τιμή αυτή είναι σημαντική γιατί ξεχωρίζει το νέο αντικείμενο από άλλα detached αντικείμενα τα οποία είχαν αποθηκευτεί σε κάποιο άλλο προηγούμενο ISession. Για αναγνωριστικούς αριθμούς, η προκαθορισμένη τιμή είναι το 0.

Η ρύθμιση <generator> ορίζει τη γεννήτρια παραγωγής νέων αναγνωριστικών για καινούργια αντικείμενα που πρόκειται να αποθηκευτούν στη βάση δεδομένων. Υπάρχουν διάφορα είδη γεννητριών ανάλογα με τις ικανότητες του συγκεκριμένου RDBMS που χρησιμοποιείται.

3.3.4.4 Απεικόνιση σύνθετου αναγνωριστικού

Αν το primary key μιας οντότητας δεν είναι ένας αναγνωριστικός αριθμός, αλλά μια σύνθεση στηλών ενός πίνακα, ή αν το primary key δεν μπορεί να πάρει τιμές που παράγονται αυτόματα από τις γεννήτριες του NHibernate, τότε πρέπει να ρυθμίσουμε το αναγνωριστικό της οντότητας ως composite-id.

Όταν χρησιμοποιείται ένα σύνθετο αναγνωριστικό, το NHibernate δεν μπορεί να το διαχειριστεί αυτόματα. Πρέπει ο προγραμματιστής να αναθέσει την τιμή του primary key (δηλαδή να αναθέσει τιμές στις κατάλληλες ιδιότητες) πριν το αντικείμενο αποθηκευτεί στην βάση δεδομένων.

Για παράδειγμα:

```
<class name="Person" table="Persons">
  <composite-id>
    <key-property name="FirstName"/>
    <key-property name="LastName"/>
  </composite-id>
  <property name="FirstName" />
  <property name="LastName" />
  <property name="Telephone" />
</class>
```

Πέραν του <key-property>, μπορούν επίσης να προστεθούν xml elements <key-many-to-one>. Αυτό γίνεται στην περίπτωση που το αντικείμενο είναι μέρος μιας σχέσης πολλά-προς-

ένα και αποτελεί μια συντόμευση για να ορίσουμε την σχέση και να δηλώσουμε ότι η αντίστοιχη ιδιότητα είναι μέρος του composite-id.

3.3.4.5 Απεικόνιση value-type αντικειμένων

Οντότητα, ή entity, αποκαλούμε τα αντικείμενα που έχουν δικό τους αναγνωριστικό και είναι αυτοτελή, δηλαδή μπορούν να αντληθούν απλώς αναφέροντας στο NHibernate το αναγνωριστικό τους. Αποκαλούμε value-type τα αντικείμενα που δεν έχουν δικό τους αναγνωριστικό. Τα αντικείμενα αυτά δεν μπορούν να υπάρχουν από μόνα τους και ανήκουν αναγκαστικά σε μια οντότητα.

Όταν θέλουμε τα value-type αντικείμενα να αποθηκεύονται στον ίδιο πίνακα της οντότητας, χρησιμοποιούμε τη ρύθμιση απεικόνισης <component>, μέσα στην <class> του Person, ως εξής:

```
<component name="HomeAddress" class="Address">
  <property name="Street" type="String" column="HOME_STREET" />
  <property name="City" type="String" column="HOME_CITY" />
</component>
```

3.3.5 Απεικόνιση Σχέσεων και Συλλογών

Ένα από τα θετικά χαρακτηριστικά των ORM εργαλείων είναι η εύκολη διαχείριση των σχέσεων. Οι σχέσεις περιλαμβάνουν τις εξής μορφές:

- One-to-One: Μια σχέση ένα-προς-ένα.
- Many-to-One: Μια σχέση πολλά-προς-ένα.
- Many-to-Many: Μια σχέση πολλά-προς-πολλά.

Οι σχέσεις είναι κατά βάση μονής κατεύθυνσης. Όμως με κατάλληλη ρύθμιση μπορούν να γίνουν αμφίδρομες. Θα επικεντρωθούμε στις αμφίδρομες σχέσεις μεταξύ οντοτήτων.

Στα .NET αντικείμενα, το άκρο μιας σχέσης που αναφέρεται σε πολλά αντικείμενα, συνήθως εκφράζεται ως μια ιδιότητα τύπου Συλλογής. Οι συλλογές που μπορούν να χρησιμοποιηθούν είναι οι:

- Συνήθεις συλλογές του .NET όπως ICollection, IList, IDictionary, IList<T> κ.α.
- Επιπλέον συλλογές που υποστηρίζει το NHibernate και δεν υπάρχουν σε όλες τις εκδόσεις του .NET όπως οι ISet και ISet<T>. Το ISet είναι μια συλλογή αντικειμένων χωρίς διπλότυπα αντικείμενα και η ταξινόμηση τους δεν έχει σημασία.

3.3.5.1 Σχέση πολλά-προς-ένα

Μια συνήθης σχέση πολλά-προς-ένα η σχέση μεταξύ γονέα και παιδιών. Για παράδειγμα:

```
<class name="ParentClass" table="Parents">
```

```
    <!-- ...other mapping configuration options... -->
    <set name="Children" inverse="true" generic="true">
      <key column="Parent_Id"/>
      <one-to-many class="ChildClass"/>
    </set>
  </class>
    <class name="ChildClass " table="Children">
    <!-- ...other mapping configuration options... -->
    <many-to-one name="Parent" column="Parent_Id"
class="ParentClass" />
  </class>
```

Η ρύθμιση <set generic=”true”> δηλώνει ότι ο τύπος της ιδιότητας Children της κλάσης ParentClass είναι ISet<ChildClass>. Η στήλη Parent_Id του πίνακα Children ορίζει ουσιαστικά την σχέση πολλά-προς-ένα και κρατάει το αναγνωριστικό του πατέρα του παιδιού.

Η ρύθμιση inverse=”true” δηλώνει ότι η σχέση είναι αμφίδρομη. Η ρύθμιση θα μπορούσε να υπάρχει στο άλλο άκρο της σχέσης. Το άκρο που είναι inversed, δεν συγχρονίζεται άμεσα με τη βάση. Δηλαδή, αν προστεθεί ένα παιδί στην συλλογή Children του γονέα οι αλλαγές δεν θα διατηρηθούν στην βάση κατευθείαν.

3.3.5.2 Σχέση ένα-προς-ένα

Μια συνήθης ένα-προς-ένα σχέση μεταξύ δύο οντοτήτων είναι όταν ένας πίνακας στη βάση δεδομένων έχει ένα ξένο κλειδί προς το πρωτεύον κλειδί ενός άλλου πίνακα. Για παράδειγμα:

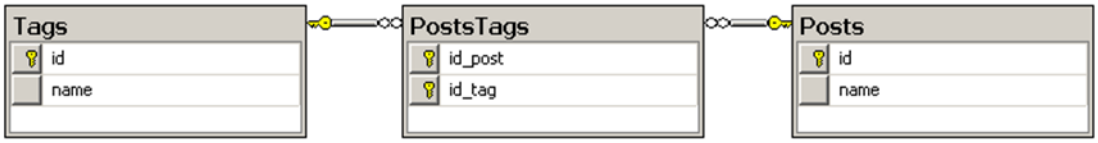
```
    <class name="Person" table="PERSON">
      <id name="Id" column="PERSON_ID">
        <generator class="foreign">
          <param name="property">Employee</param>
        </generator>
      </id>
      <!-- ...other mapping configuration options... -->
      <one-to-one name="Employee" class="Employee"
constrained="true"/>
    </class>
      <class name="Employee" table="EMPLOYEE">
      <id name="Id">
        <generator class="native" />
      </id>
      <!-- ...other mapping configuration options... -->
```

```
    <one-to-one name="Person" class="Person"/>
  </class>
```

Η ειδική ρύθμιση <generator class=”foreign”> μαζί με την ρύθμιση constrained=”true”, δηλώνουν στο NHibernate ότι: Όταν αποθηκεύσει ένα καινούργιο στιγμιότυπο του Person πρέπει να του αναθέσει στο primary key την τιμή του primary key του Employee που αναφέρει.

3.3.5.3 Σχέση πολλά-προς-πολλά

Ας πάρουμε για παράδειγμα το εξής σχεσιακό σχήμα:



Εικόνα 3-13 Σχεσιακό διάγραμμα του παραδείγματος

Για την απεικόνιση της σχέσης, αρκούν οι εξής ρυθμίσεις απεικόνισης στο .hbm.xml αρχείο:

```
<class name="Tag" table="Tags">
  <id name="Id">
    <generator class="native"></generator>
  </id>
  <!-- ...other mapping configuration options... -->
</class>
<class name="Post" table="Posts">
  <id name="Id">
    <generator class="native"></generator>
  </id>
  <!-- ...other mapping configuration options... -->
  <bag name="Tags" table="PostsTags">
    <key column="id_post"></key>
    <many-to-many class="Tag" column="id_tag"></many-to-many>
  </bag>
</class>
```

Για ευκολία, στις παραπάνω ρυθμίσεις ορίσαμε τη σχέση ως μονής κατεύθυνσης. Αυτό φαίνεται στο ότι δε χρησιμοποιήσαμε τη ρύθμιση “inversed=true”.

Βλέπουμε ότι ο ενδιάμεσος πίνακας που αντιπροσωπεύει τη σχέση πολλά-προς-πολλά δεν χρειάζεται να αντιπροσωπευθεί με κάποιο .NET αντικείμενο. Οι .NET Συλλογές των ιδιοτήτων των κλάσεων Tag και Post αρκούν για την αναφορά στα αντικείμενα της σχέσης.

3.3.5.4 *Περισσότερα για τις συλλογές*

Στις παραπάνω ρυθμίσεις απεικόνισεις, αντί για τη <set> ρύθμιση, που αντιπροσωπεύει μια ιδιότητα τύπου ISet, μπορούν να χρησιμοποιηθούν με ανάλογο τρόπο οι εξής ρυθμίσεις:

- <bag>: Αντιπροσωπεύει μια συλλογή σαν τη <set> όπου όμως επιτρέπονται διπλότυπα. Η σειρά δεν έχει σημασία. Το <bag> προσομοιώνεται στο .NET μέσω μιας IList συλλογής.
- <list>: Αντιπροσωπεύει μια IList συλλογή. Η σειρά των αντικειμένων έχει σημασία. Επιτρέπονται διπλότυπα.
- <map>: Αντιπροσωπεύει μια IDictionary συλλογή. Αποθηκεύει μια αφαιρετική δομή που αποτελείται από μία συλλογή κλειδιών και μια συλλογή τιμών, όπου κάθε κλειδί σχετίζεται με μια τιμή.

Το NHibernate προσφέρει διάφορες επιπλέον ρυθμίσεις για τις συλλογές:

- generic: true/false. Έχει ήδη εξηγηθεί.
- lazy: true/false. Έχει ήδη εξηγηθεί.
- <index column="…" />: Για τις συλλογές όπου η σειρά είναι σημαντική, η στήλη αυτή κρατάει την θέση ενός αντικειμένου μέσα στη συλλογή.
- Για τις συλλογές που αντιπροσωπεύει το άκρο μια σχέσης (με πολλαπλότητα μεγαλύτερη το υ 1) υπάρχει και μια ακό α πιθανή ρύθμιση, η "cascade", τη σημασία της οποίας θα αναλύσουμε στην ενότητα 3.3.7.2 «Transitive persistence και Cascading persistence».

3.3.5.5 *Απεικόνιση συλλογών από value-type αντικείμενα*

Για παράδειγμα, έστω η εξής κλάση Person η οποία έχει μια ιδιότητα Contacts : HashedSet<String>. Για να αντιπροσωπεύσουμε τη συλλογή των strings σε έναν ξεχωριστό πίνακα "Contacts", αρκούν οι εξής ρυθμίσεις:

```
<class name="Person" table="Persons">
  <!-- ...other mapping configuration options... -->
  <set name="Contacts">
    <key column="PersonId" />
    <element column="Name" type="String"/>
  </set>
</class>
```

3.3.5.6 *Το πρόβλημα της ταυτότητας και της ισότητας*

Πιο συγκεκριμένα, στις αντικειμενοστραφείς εφαρμογές που χρησιμοποιούν ένα ORM εργαλείο όπως το NHibernate, μας απασχολούν τρεις μορφές ταυτότητας ή ισότητας:

- **Η ταυτότητα των αντικειμένων**: Δύο αντικείμενα είναι ίδια αν αναφέρονται στην ίδια τοποθεσία στη μνήμη. Αυτό μπορεί να ελεγχθεί χρησιμοποιώντας τη μέθοδο object.ReferenceEquals() του .NET Framework.
- **Η ισότητα των αντικειμένων**: Δύο αντικείμενα, που δεν είναι κατ' ανάγκη ίδια, είναι ίσα αν η μέθοδος Equals(object o) γυρίζει true.
- **Η ταυτότητα της βάσης δεδομένων**: Δύο αντικείμενα που έχουν αντληθεί από μια βάση δεδομένων είναι ίδια εφόσον μοιράζονται την ίδια τιμή του πρωτεύοντος κλειδιού του ίδιου πίνακα, δηλαδή εφόσον αναφέρονται στην ίδια γραμμή.

Ο προγραμματιστής πρέπει να προσέξει ιδιαίτερα τις τρεις αυτές μορφές. Όταν στην εφαρμογή χρησιμοποιούνται συλλογές αντικειμένων, πρέπει να προσέχει όταν σε μια συλλογή υπάρχουν αντικείμενα που έχουν αντληθεί από διαφορετικά ISessions. Αυτά τα αντικείμενα θα έχουν διαφορετική .NET ταυτότητα, γιατί θα αναφέρονται σε διαφορετικές τοποθεσίες μνήμης.

Για παράδειγμα, αν σε μια ISession αντληθεί ένα αντικείμενο και τοποθετηθεί σε μια ISet συλλογή, και σε μια δεύτερη ISession αντληθεί ένα αντικείμενο με το ίδιο id και τοποθετηθεί στην ίδια ISet συλλογή, τότε στη συλλογή θα υπάρχουν δύο αντικείμενα που έχουν την ίδια ταυτότητα στη βάση δεδομένων. Όμως τα δύο αυτά αντικείμενα έχουν διαφορετικές .NET ταυτότητες, αφού αναφέρονται σε διαφορετικές τοποθεσίες μνήμης.

Προβλήματα όπως το παραπάνω καλείται ο προγραμματιστής να διορθώσει υλοποιώντας κατάλληλα τις μεθόδους Equals(object o) και GetHashCode() των αντικειμένων που πρόκειται να αποθηκεύονται με το NHibernate. Οι δύο αυτές μέθοδοι πρέπει να είναι συμβατές μεταξύ τους (δηλαδή αν δύο αντικείμενα είναι ίσα σύμφωνα με την Equals, πρέπει και η GetHashCode να γυρίζει το ίδιο hash code).

Μια συνήθης τακτική για την υλοποίηση των δύο μεθόδων βασίζεται στα αναγνωριστικά των αντικειμένων. Δηλαδή παίρνουν υπ' όψιν την .NET ταυτότητα των αντικειμένων αλλά και την ισότητα των αναγνωριστικών αριθμών των αντικειμένων. Αυτήν την τακτική ακολουθούμε και στην υλοποίηση της Υπηρεσίας Αποθήκευσης του NCommet.

Η παραπάνω τακτική όμως δεν είναι σωστή για όλες τις περιπτώσεις. Ξέρουμε ότι το NHibernate δεν δίνει αναγνωριστικά σε καινούργια παροδικά αντικείμενα. Έτσι δεν μπορούμε να είμαστε σίγουροι ότι δύο καινούργια αντικείμενα είναι ίσα, όταν βασιζόμαστε μόνο στα αναγνωριστικά και στις αναφορές τους.

Για να είμαστε σίγουροι, πρέπει να πάρουμε υπ' όψιν κάποια ιδιότητα των αντικειμένων που ξέρουμε ότι είναι μοναδική για κάθε αντικείμενο και έχει πάντα τιμή. Για παράδειγμα, αν θεωρήσουμε ότι ένα άτομο έχει πάντα μοναδικό όνομα, θα πρέπει να πάρουμε υπ' όψιν το όνομα του.

3.3.6 Απεικόνιση Κληρονομικότητας

[Amb10]

Υπάρχουν τρεις τακτικές για την απεικόνιση της ιεραρχίας κληρονομικότητας των κλάσεων:

- **Ένας πίνακας ανά τελική κλάση** (table per concrete class): Δεν αποθηκεύουμε λεπτομέρειες για τον πολυμορφισμό και τις σχέσεις κληρονομικότητας.
- **Ένας πίνακας ανά ιεραρχία κλάσεων** (table per class hierarchy): Αποθηκεύουμε σε έναν πίνακα όλες τις κλάσεις που ανήκουν στην ίδια ιεραρχία κληρονομικότητας.
- **Ένας πίνακας ανά υποκλάση** (table per subclass): Απεικονίζουμε την ιεραρχία κληρονομικότητας μέσω σχέσεων της βάσης δεδομένων.

3.3.6.1 Ένας πίνακας ανά τελική κλάση

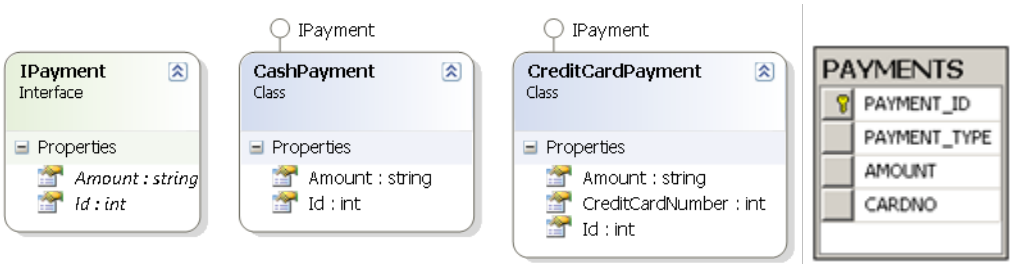
Όλες οι ιδιότητες μιας κλάσης, συμπεριλαμβανομένου και των κληρονομούμενων ιδιοτήτων, απεικονίζονται σε στήλες ενός πίνακα. Κάθε κλάση θα απεικονίζεται στον δικό της ξεχωριστό πίνακα.

Η τακτική αυτή παρουσιάζει αρκετά προβλήματα. Το σημαντικότερο είναι ότι δεν αποθηκεύονται οι σχέσεις κληρονομικότητας. Γι' αυτό το λόγο, δεν μπορούμε να εκτελέσουμε πολυμορφικά HQL ερωτήματα.

3.3.6.2 Ένας πίνακας ανά ιεραρχία κλάσεων

Όλες οι κλάσεις μιας ιεραρχίας κληρονομικότητας απεικονίζονται στον ίδιο πίνακα. Ο πίνακας αυτός περιέχει στήλες για όλες τις ιδιότητες όλων των κλάσεων της ιεραρχίας. Για να ξέρουμε σε ποια υποκλάση ανήκει μια γραμμή του πίνακα, χρησιμοποιούμε μια ειδική discriminator στήλη η οποία περιέχει τον τύπο της υποκλάσης.

Για παράδειγμα, έστω ότι έχουμε μια διεπαφή (interface) IPayment και δύο κλάσεις που το υλοποιούν: CreditCardPayment και CashPayment, των οποίων το διάγραμμα κλάσεων φαίνεται παρακάτω. Τότε, με κατάλληλες ρυθμίσεις απεικόνισης, τα στιγμιότυπα των δύο κλάσεων μπορούν να αποθηκευτούν σε έναν πίνακα, το σχεσιακό διάγραμμα του οποίου φαίνεται παρακάτω.

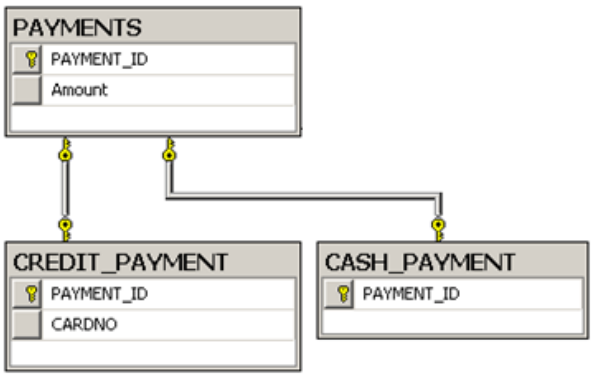


Εικόνα 3-14 Διάγραμμα κλάσεων και σχεσιακό διάγραμμα της περίπτωσης «Ένας πίνακας ανά ιεραρχία κλάσεων»

3.3.6.3 Ένας πίνακας ανά υποκλάση

Με αυτή την τακτική απεικονίζουμε πλήρως τις σχέσεις κληρονομικότητας μέσω foreign-key σχέσεων στη βάση δεδομένων. Κάθε υποκλάση (συμπεριλαμβανομένου abstract κλάσεων και interfaces), έχει το δικό της πίνακα.

Παίρνοντας υπ' όψιν το προηγούμενο παράδειγμα του IPayment, και με τις κατάλληλες ρυθμίσεις απεικόνισης, το αντίστοιχο σχεσιακό διάγραμμα θα έχει ως εξής:



Εικόνα 3-15 Σχεσιακό διάγραμμα περίπτωσης «Ένας πίνακας ανά υποκλάση»

Μόνο η υπερκλάση έχει γεννήτρια αναγνωριστικών αριθμών. Αυτό γιατί κάθε γραμμή ενός πίνακα μιας υποκλάσης πρέπει να αναφέρεται σε μια γραμμή του πίνακα της υπερκλάσης.

3.3.7 Λοιπά θέματα

3.3.7.1 Η ζωή των persistent αντικειμένων

Τα .NET αντικείμενα με τα οποία δουλεύει η εφαρμογή μπορεί να βρίσκονται σε διάφορες καταστάσεις σε σχέση με ένα ISession του NHibernate. Για παράδειγμα, μπορεί το αντικείμενο να είναι καινούργιο και να πρέπει να εισαχθεί στη βάση δεδομένων.

Υπάρχουν τρεις καταστάσεις στις οποίες μπορείς να βρίσκεται ένα αντικείμενο:

- Παροδικά (transient) αντικείμενα.
  Τα καινούργια αντικείμενα δεν διαχειρίζονται αυτόματα από το NHibernate, ούτε οι αλλαγές τους ανιχνεύονται αυτόματα από τις συναλλαγές ενός ISession. Αυτό συμβαίνει γιατί δεν απεικονίζονται ακόμα στη βάση δεδομένων. Για να γίνει το παροδικό αντικείμενο persistent, πρέπει ο προγραμματιστής είτε να το κάνει Save(), είτε να αναφερθεί σε αυτό μέσω ενός ήδη persistent αντικειμένου.
- Διατηρημένα (persistent) αντικείμενα.
  Ένα persistent αντικείμενο έχει απεικονιστεί στη βάση δεδομένων και έχει αποκτήσει μια τιμή από το πρωτεύον κλειδί του πίνακα στον οποίο έχει αποθηκευτεί. Τα persistent αντικείμενα πάντα αντλούνται από ένα ISession, ή γίνονται Save() μέσω αυτού. Επίσης, οι αλλαγές τους ανιχνεύονται αυτόματα από το NHibernate και

μπορούν να πάρουν μέρος στις συναλλαγές του ISession. Όταν ένα persistent αντικείμενο γίνει Delete(), τότε περνάει στην κατάσταση transient.

- Αποσπασμένα (detached) αντικείμενα.
  Όταν ένα ISession κλείνει, τότε τα persistent αντικείμενα που σχετιζόταν με το ISession περνάνε στην detached κατάσταση. Αυτό σημαίνει ότι δεν διαχειρίζονται πλέον από το NHibernate, οπότε τα δεδομένα τους δεν είναι κατ' ανάγκην συγχρονισμένα με τη βάση δεδομένων. Ένα detached αντικείμενο μπορεί να ξαναπεράσει στην κατάσταση persistent αν σχετιστεί με ένα καινούργιο ISession.

3.3.7.2 Transitive persistence και Cascading persistence

Η κατάσταση κατά την οποία ένα παροδικό αντικείμενο γίνεται persistent όταν αναφερθεί από ένα ήδη persistent αντικείμενο λέγεται transitive persistence. Υπάρχουν αρκετές τεχνικές υλοποίησης του transitive persistence. Το NHibernate χρησιμοποιεί το δικό του μοντέλο, το Cascading Persistence.

Το Cascading Persistence ρυθμίζεται μέσω της ρύθμισης απεικόνισης "cascade" για τις συλλογές που αντιπροσωπεύουν σχέσεις μεταξύ οντοτήτων. Ορίζοντας αυτή τη ρύθμιση κατάλληλα, το NHibernate μπορεί να διαχειριστεί αυτόματα το persistency αντικειμένων μιας συλλογής που σχετίζονται με ένα persistent αντικείμενο που τροποποιεί ο προγραμματιστής.

Οι επιλογές του cascade είναι οι εξής:

- "none": Προκαθορισμένη επιλογή. Η διαχείριση του persistency αφήνεται στον προγραμματιστή.
- "save-update": Όταν ένα αντικείμενο αποθηκεύεται, το NHibernate διαβαίνει τη συλλογή της σχέσης και σώζει τα παροδικά αντικείμενα αυτόματα.
- "delete": Όταν ένα αντικείμενο γίνεται Delete(), το NHibernate διαβαίνει τη συλλογή της σχέσης και διαγράφει τα αντικείμενα αυτόματα.
- "all": Συνδυάζει το "save-update" και το "delete".
- "all-delete-orphan": Είναι όπως το "all", αλλά επιπρόσθετα το NHibernate διαγράφει τυχόν οντότητες που έχουν αφαιρεθεί από τη συλλογή της σχέσης.
- "delete-orphan": Το NHibernate διαγράφει τυχόν οντότητες που έχουν αφαιρεθεί από τη συλλογή μιας σχέσης.

3.3.7.3 Τύποι δεδομένων

Το NHibernate έχει μια αυτόματη αντιστοιχία μεταξύ των τύπων που ορίζονται για τις ιδιότητες στα αρχεία ρυθμίσεων και των τύπων που θα χρησιμοποιηθούν για τις αντίστοιχες στήλες στη βάση δεδομένων.

Το NHibernate δεν κάνει κατευθείαν την αντιστοιχία σε τύπους του SQL Server που χρησιμοποιούμε. Επειδή το NHibernate μπορεί να χρησιμοποιηθεί και για άλλα RDBMS, η αντιστοιχία γίνεται βάσει των τύπων DbType.* του ADO.NET του .NET Framework. Οι τύποι αυτοί μετατρέπονται στο τέλος σε vendor-specific τύπους από την ενσωματωμένη υποστήριξη του NHibernate για SQL dialects.

3.3.8 Hibernate Query Language (HQL)

[NHi10], [Kua09]

Η HQL είναι μια γλώσσα διατύπωσης ερωτημάτων ανάκτησης δεδομένων για το NHibernate. Μοιάζει αρκετά με SQL αλλά είναι πλήρως αντικειμενοστραφής και αντιλαμβάνεται έννοιες όπως κληρονομικότητα, πολυμορφισμός και σχέσεις. Χρησιμοποιείται μόνο για ανάκτηση δεδομένων και όχι για τροποποίηση δεδομένων.

Η δύναμη της HQL έγκειται στο ότι διατυπώνουμε ερωτήματα επί των κλάσεων των αντικειμένων της εφαρμογής μας. Κατά την εκτέλεση του ερωτήματος, το NHibernate επεξεργάζεται το ερώτημα και, μέσω των ρυθμίσεων απεικόνισης, διαμορφώνει και στέλνει δυναμικά το κατάλληλο SQL ερώτημα προς τη βάση δεδομένων. Τελικά μας γυρίζει ένα αντικειμενοστραφές αποτέλεσμα το οποίο είναι άμεσα κατανοητό από την εφαρμογή μας και δεν χρειάζεται καμία μετατροπή.

Στην ενότητα 3.3.3.5 «IQuery» περιγράψαμε τη διεπαφή IQuery με την οποία μπορούμε να εκτελέσουμε ένα HQL ερώτημα και να λάβουμε τα αποτελέσματα σε μορφή IList ή IList<T>. Το IQuery μας δίνει επίσης τη δυνατότητα ορισμού ονομαστικών παραμέτρων μέσα στο ερώτημα.

Στις επόμενες υπο-ενότητες θα περιγράψουμε, μέσω απλών παραδειγμάτων, τους βασικούς όρους και τις βασικές προτάσεις που υποστηρίζει η HQL. Πολλά από αυτά τα παραδείγματα βασίζονται στην εξής κλάση μιας εικονικής γάτας.

Εικόνα 3-16 Διάγραμμα της κλάσης Cat

3.3.8.1 Πρόταση from

Το απλούστερο ερώτημα είναι της μορφής:

```
from Cat as cat
```

Επιστρέφει όλα τα στιγμιότυπα της κλάσης Cat. Μάλιστα, επιστρέφει και τα στιγμιότυπα όλων των υποκλάσεων της Cat. Η πρόταση from μπορεί να συνδυάσει αρκετές κλάσεις, με αποτέλεσμα ένα καρτεσιανό γινόμενο. Επίσης, υποστηρίζεται η χρήσης του όρου “order by” για την ταξινόμηση (αύξουσα ή φθίνουσα) των αποτελεσμάτων βάσει μιας ιδιότητας τους.

Πρέπει να σημειώσουμε ότι τα ερωτήματα δεν είναι case-sensitive, αλλά τα ονόματα των .NET κλάσεων και των ιδιοτήτων τους είναι.

3.3.8.2 Σχέσεις και ενώσεις

Οι όροι των ενώσεων (joins) δανείζονται από την ANSI SQL και είναι οι εξής:

- inner join ή join
- left outer join ή left join
- right outer join ή right join
- full join

Για παράδειγμα:

```
from Cat as cat
    inner join cat.Mate as mate
    where mate.Name = 'Jack'
```

Όμως, η HQL μπορεί να συνάγει, μέσω των ρυθμίσεων απεικόνισης, τις ενώσεις για κλάσεις που σχετίζονται. Το προηγούμενο παράδειγμα μπορεί να γραφεί ως εξής:

```
from Cat cat where cat.Mate.Name = 'Perfect'
```

3.3.8.3 Πρόταση select

Το select επιλέγει τα αντικείμενα και τις ιδιότητες τους που θα γυρίσει το NHibernate στη λίστα των αποτελεσμάτων. Για παράδειγμα:

```
select cat.Mate from Cat cat
```

3.3.8.4 Συναθροίσματα

Οι όροι των συναθροισμάτων (aggregate functions) δανείζονται από την ANSI SQL και είναι οι εξής:

- avg(…), sum(…), min(…), max(…)
- count(*)
- count(…), count(distinct …), count(all …)

Επιπλέον, υποστηρίζεται η SQL έκφραση «group by» για την ομαδοποίηση αποτελεσμάτων με συναθροίσματα βάσει οποιασδήποτε ιδιότητας των αντικειμένων. Για παράδειγμα:

```
select cat.Color, sum(cat.Weight), count(cat)
from Eg.Cat cat
group by cat.Color
```

Σημείωση: Επιτρέπεται επίσης η χρήση της έκφρασης «having».

3.3.8.5 Πρόταση where

Ο όρος where χρησιμοποιείται για να περιορίσει τα αποτελέσματα των στιγμιότυπων που γυρίζει το NHibernate. Σε ένα where, μπορεί να χρησιμοποιηθεί ο τελεστής «=» για σύγκριση ολόκληρων στιγμιότυπων. Για παράδειγμα:

```
from Eg.Cat cat, Eg.Cat rival where cat.Mate = rival.Mate
```

3.3.8.6 Εκφράσεις

Οι εκφράσεις που μπορούν να γραφούν στην πρόταση where περιλαμβάνουν τις περισσότερες εκφράσεις της SQL:

- Μαθηματικοί τελεστές +, -, *, /.
- Τελεστές σύγκρισης =, >=, <=, <>, !=, like.
- Λογικοί τελεστές and, or, not.
- Ένωση αλφαριθμητικών ||
- SQL scalar συναρτήσεις όπως upper() και lower()
- Οι παρενθέσεις ( ) υποδεικνύουν την ομαδοποίηση εκφράσεων
- Τελεστές in, between, is null.

Στις εκφράσεις μπορούν να χρησιμοποιηθούν και οι εξής ειδικές συναρτήσεις:

- size() για τον αριθμό μιας συλλογής (collection).
- maxindex() ή minindex() για την αναφορά στο μέγιστο ή ελάχιστο δείκτη (index) μιας συλλογής.
- maxelement() ή minelement() για την αναφορά στο μέγιστο ή ελάχιστο στοιχείο μιας συλλογής.
- Οι SQL συναρτήσεις any(), some(), all(), exists(), in() μπορούν να χρησιμοποιηθούν με τα στοιχεία ή τους δείκτες μιας συλλογής. Τα στοιχεία μιας συλλογής μπορούν να αναφέρονται με χρήση της ειδικής συνάρτησης elements() και οι δείκτες της συλλογής με τη συνάρτηση indices().
- Τα στοιχεία μιας συλλογής με δείκτες (πίνακες, λίστες, maps) μπορούν να αναφέρονται χρησιμοποιώντας τον δείκτη τους (είτε αριθμητικό είτε αλφαριθμητικό δείκτη) ως εξής:

```
select item from Item item, Order order
where order.Items[ size(order.Items) - 1 ] = item
```

3.4 XML Σειριακοποίηση

[Mic10a]

Η Σειριακοποίηση (Serialization), είναι η διαδικασία της μετατροπής ενός αντικειμένου σε μια μορφή που μπορεί να μεταφέρεται εύκολα. Για παράδειγμα, ένα αντικείμενο μπορεί να σειριακοποιηθεί και να μεταδοθεί στο Internet μέσω HTTP μεταξύ ενός εξυπηρετητή και ενός πελάτη. Στην άλλη άκρη, η αποσειριακοποίηση ανασυγκροτεί το αντικείμενο από την ροή (Stream).

Η XML Σειριακοποίηση του .NET Framework, σειριακοποιεί τα public πεδία και τις public ιδιότητες ενός αντικειμένου σε μια ροή XML. Δεν συμπεριλαμβάνει όμως πληροφορίες για τον τύπο της κλάσης του αντικειμένου.

Για να μπορεί ένα στιγμιότυπο μιας κλάσης να σειριακοποιηθεί πρέπει:

- Η κλάση να έχει κοσμηθεί με το Attribute “[Serializable]” του .NET Framework.
- Η κλάση να έχει μια προκαθορισμένη constructor μέθοδο. Σημείωση: Αν δεν έχει οριστεί καμία constructor μέθοδος σε μια κλάση, ο μεταγλωττιστής προσπαθεί αυτόματα να προσθέσει μια constructor μέθοδο χωρίς παραμέτρους.
- Οι ιδιότητες και τα πεδία, προς σειριακοποίηση, πρέπει να ορίζονται ως public.

Η κεντρική κλάση της XML Σειριακοποίησης είναι η κλάση XmlSerializer του .NET Framework. Οι πιο σημαντικές μέθοδοι της κλάσης είναι οι Serialize() και Deserialize().

Τα πλεονεκτήματα της σειριακοποίησης είναι τα εξής:

- Οι περιορισμοί που τίθενται είναι απλοί και μπορούν να ικανοποιηθούν εύκολα από τις κλάσεις των business objects μιας εφαρμογής.
- Η εφαρμογή μπορεί να δημιουργήσει νέες ιδιότητες ή να αφαιρέσει άλλες από τις κλάσεις των business objects, χωρίς να χρειάζεται να αλλάξει κάτι στην σειριακοποίηση. Απλώς, όταν θα σειριακοποιηθεί το νέο αντικείμενο, θα αλλάξει η XML που το αντιπροσωπεύει.
- Η εφαρμογή μπορεί να ελέγξει τη διαδικασία της σειριακοποίησης, μέσω κατάλληλων Attributes του .NET Framework. Για παράδειγμα, μπορεί να ελέγξει αν μια ιδιότητα θα αποθηκευτεί ως XML Attribute ή ως ένα XML Element.

Το κυριότερο μειονέκτημα της σειριακοποίησης είναι ότι η εφαρμογή πρέπει να προσέχει τις προσθαφαιρέσεις και τις αλλαγές των ιδιοτήτων μιας κλάσης. Για παράδειγμα, έστω ότι μια εφαρμογή χρησιμοποιεί την XML Σειριακοποίηση για να αποθηκεύει τα business objects της σε μορφή XML αρχείων. Αν προσθέσει μια καινούργια ιδιότητα σε κάποια κλάση, τότε τα παλαιά στιγμιότυπα της κλάσης που έχουν ήδη σειριακοποιηθεί και αποθηκευτεί, δεν θα έχουν τιμή για αυτήν την ιδιότητα. Αν η καινούργια ιδιότητα είναι απαραίτητη, τότε θα υπάρχει πρόβλημα κατά την αποσειριακοποίηση (deserialization) των παλαιών αντικειμένων. Υπάρχουν διάφορες λύσεις για το παραπάνω πρόβλημα, όπως:

- Οι καινούργιες ιδιότητες να μην είναι απαραίτητες.
- Κατά την αποσειριακοποίηση της XML, αν δεν βρεθεί τιμή για κάποια απαραίτητη καινούργια ιδιότητα, να δίνει μια προκαθορισμένη τιμή σε αυτήν την ιδιότητα.

3.4.1 Παράδειγμα σειριακοποίησης

Έστω μια κλάση Article, το διάγραμμα της οποίας φαίνεται παρακάτω.

Εικόνα 3-17 Διάγραμμα της κλάσης Article

Με την ανάπτυξη του κατάλληλου κώδικα, τα στιγμιότυπα αυτής της κλάσης μπορούν να σειριακοποιηθούν στην εξής XML μορφή.

```
<?xml version="1.0" encoding="utf-16"?>
<Article xmlns:xsi="http://www.w3.org/200
```

```
1/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <Title>A nice
 article</Title>
    <Body>A really long text.</Body>
</Article>
```

Έπειτα, η ίδια XML μπορεί να αποσειριακοποιηθεί για την παραγωγή το υ ίδιου στιγμιότυπου.

4

Ανάλυση Απαιτήσεων Συστήματος

Η ενότητα είναι οργανωμένη ως εξής: Αρχίζουμε περιγράφοντας τη γενική αρχιτεκτονική του συστήματος μέσα στο οποίο λειτουργεί το NCCMS. Ύστερα περιγράφουμε τη λειτουργία του NCommet, της βασικής βιβλιοθήκης οργάνωσης δενδρικών δομών δεδομένων που χρησιμοποιεί το NCCMS. Το NCommet αποτελείται από Υπηρεσίες, μία εκ των οποίων είναι και η Υπηρεσία Αποθήκευσης που αποτελεί το επίπεδο διατήρησης της εφαρμογής που χρησιμοποιεί το NCommet (άρα και του NCCMS). Στη συνέχεια, διατυπώνουμε τις λειτουργικές απαιτήσεις της νέας Υπηρεσίας Αποθήκευσης που θα υλοποιήσουμε στην ενότητα 6 «Υλοποίηση». Τέλος, δίνουμε το μοντέλο οντοτήτων συσχετίσεων των βασικών οντοτήτων του NCommet το οποίο καλείται να διατηρήσει η νέα Υπηρεσίας Αποθήκευσης.

4.1 Γενική Αρχιτεκτονική

Το NCCMS είναι ένα πλήρες περιβάλλον διαχείρισης ιστοσελίδων μαθημάτων, με πλειάδα σύνθετων λειτουργιών και τύπων περιεχομένου.

Για το σκοπό της διπλωματικής, αφαιρέθηκαν πολλές λειτουργίες του NCCMS οι οποίες δεν έχουν σχέση με το αντικείμενο μας. Για παράδειγμα, αφαιρέθηκε η δυνατότητα ταυτοποίησης χρηστών μέσω του καταλόγου χρηστών του Ε.Μ.Π., αφού η συγκεκριμένη δυνατότητα δεν μας ενδιαφέρει για την εφαρμογή της αντικειμενο-σχεσιακής απεικόνισης των δομών δεδομένων.

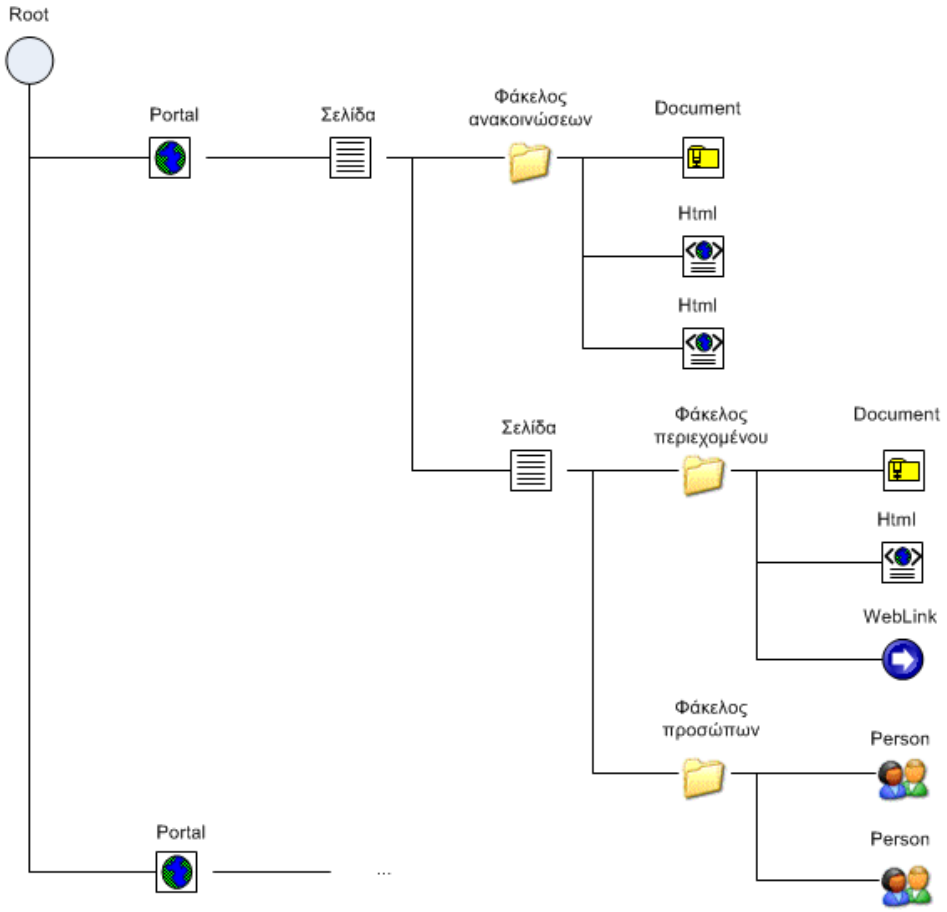
Το αποτέλεσμα ονομάστηκε SimpleCMSPlus και αποτελεί μια απλουστευμένη έκδοση του NCCMS.

4.1.1 SimpleCMSPlus

Στόχος: Να υλοποιεί απλές εφαρμογές διαχείρισης περιεχομένου στο web (Content Management εφαρμογές), με τις εξής γενικές προδιαγραφές:

- Root (1:N) Portal.
- Portal (1:N) Page.
- Page (1:N) Page.
- Page (1:N) Module.
- Module (1:N) ContentItem.
- ContentItem: Document, Html, Weblink, Person.
- Module: Announcements List, Content List, Persons List.

Ένα παράδειγμα που μπορεί να φτιαχτεί σύμφωνα με τα παραπάνω είναι το εξής:



Εικόνα 4-1 Παράδειγμα περιεχομένου

4.1.2 NCommet

Εδώ περιγράφουμε το NCommet εν συντομία. Στην ενότητα 4.2 «Επισκόπηση του NCommet», το περιγράφουμε αναλυτικά.

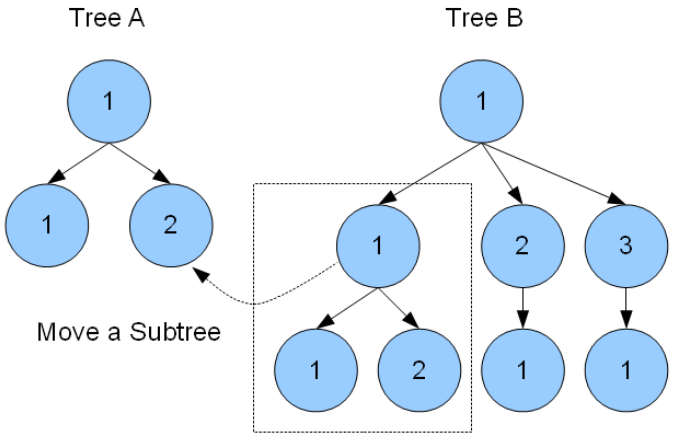
Το NCommet είναι μια γενική βιβλιοθήκη για οργάνωση δενδρικών δομών δεδομένων. Έχει τα εξής χαρακτηριστικά:

- **Typed**: Κάθε κόμβος του δένδρου μπορεί να έχει συγκεκριμένο τύπο (type).
- **Ordered**: Τα παιδιά ενός κόμβου έχουν συγκεκριμένη μεταξύ τους σειρά: $1^{ο}$ παιδί, $2^{ο}$ παιδί κλπ.
- **Content-capable**: Κάθε κόμβος μπορεί να περιέχει ένα αντικείμενο που φέρει χρήσιμο περιεχόμενο (document, html, weblink, person κλπ). Οι κλάσεις των αντικειμένων περιεχομένου ονομάζονται κλάσεις περιεχομένου.

Επιπλέον, το NCommet δίνει τη δυνατότητα συσχέτισης κόμβων, πέραν της δενδρικής δομής, ορίζοντας συσχετίσεις που μπορούν να φέρουν **τύπο** και **κατεύθυνση**. Δηλαδή οι κόμβοι μπορούν να σχηματίσουν typed directed graphs.

Το NCommet δεν δίνει βιβλιοθήκες για τυπική διαχείριση γράφων, παρότι οι κόμβοι του μπορούν να οργανωθούν σε typed directed graphs.

Δίνει βιβλιοθήκες για προχωρημένες ρουτίνες διαχείρισης typed ordered trees, όπως για τη μεταφορά ενός υπόδενδρου από έναν κόμβο σε έναν άλλον κόμβο:



Εικόνα 4-2 Παράδειγμα ρουτίνας διαχείρισης δένδρου: "Μεταφορά υπόδενδρου"

Το NCommet περιλαμβάνει επιπρόσθετη λειτουργικότητα που χωρίζεται σε υπηρεσίες:

- Υπηρεσία αποθήκευσης. Υποστηρίζει την ανάκτηση και την αποθήκευση των κόμβων, των αντικειμένων περιεχομένου και των συσχετίσεων.
- Υπηρεσία αναγνώρισης χρήστη. Συνδέεται με ένα εξωτερικό σύστημα για την αναγνώριση του τρέχοντος χρήστη και των ρόλων του.

- Υπηρεσία δικαιωμάτων πρόσβασης (Authorization). Υποστηρίζει τον ορισμό δικαιωμάτων πρόσβασης ανά κόμβο και ανά ρόλο χρηστών.
- Υπηρεσία κανόνων εγκυρότητας (Typed tree validation). Υποστηρίζει τον ορισμό κανόνων εγκυρότητας επί των κόμβων και της δενδρικής δομής τους, βάσει των τύπων των κόμβων. Για παράδειγμα μπορεί να οριστεί κανόνας της μορφής: Ένας κόμβος τύπου T1 μπορεί να έχει παιδιά μόνο κόμβους τύπων T2 και T3.
- Υπηρεσία αναζήτησης περιεχομένου. Υποστηρίζει την αναζήτηση κειμένου σε ιδιότητες των αντικειμένων περιεχομένων των κόμβων.
- Υπηρεσία ταξινόμησης κόμβων. Υποστηρίζει την ταξινόμηση κόμβων βάσει μιας ιδιότητας των αντικειμένων περιεχομένων τους.
- Υπηρεσία επεκτασιμότητας. Υποστηρίζει εξωτερικά modules τα οποία χρησιμοποιούν τα συμβάντα (events) του NCommet για να εκτελέσουν τις λειτουργίες τους.

Για τις υπηρεσίες αυτές, το NCommet ορίζει interfaces (διεπαφές). Η υλοποίηση των διεπαφών (πλην αυτών που αφορούν την *υπηρεσία αποθήκευσης*) είναι προαιρετική. Δηλαδή θα μπορούσε μια πολύ απλή εφαρμογή να μην έχει υλοποιήσει ούτε την υπηρεσία δικαιωμάτων πρόσβασης, ούτε την υπηρεσία κανόνων εγκυρότητας, κ.ο.κ. Φυσικά, σε αυτήν την περίπτωση δεν θα μπορούσε να υποστηρίξει την αντίστοιχη λειτουργικότητα.

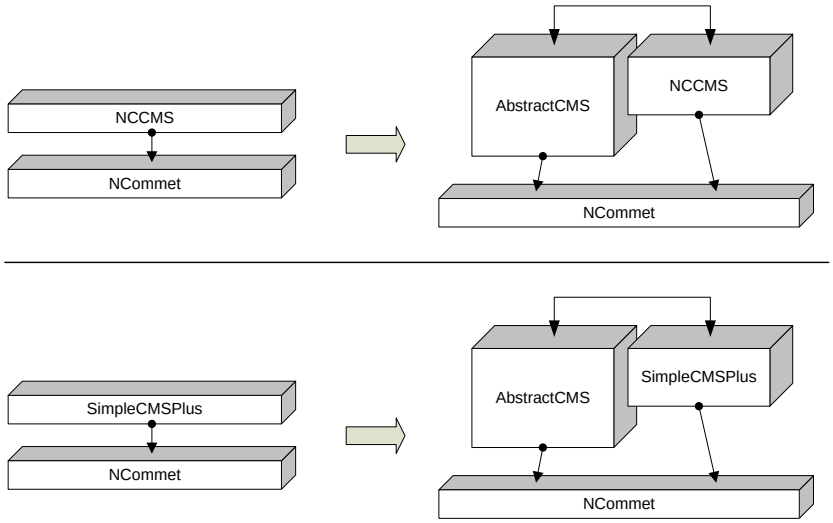
Η υλοποίηση της *υπηρεσίας αποθήκευσης* είναι υποχρεωτική, διότι για λόγους επίδοσης, πολλές λειτουργίες διαχείρισης της δενδρικής δομής αποθηκεύουν και ανακτούν τα δεδομένα τους από το μέσο αποθήκευσης.

Για όλες τις διεπαφές (πλην της αναζήτησης), το NCommet παρέχει απλές υλοποιήσεις. Μια εφαρμογή διαχείρισης περιεχομένου με πιο σύνθετες απαιτήσεις μπορεί να ορίσει δικές της υλοποιήσεις για οποιοδήποτε διεπαφή. Η αντιστοίχιση των διεπαφών και των υλοποιήσεων γίνεται στο αρχείο ρυθμίσεων της εφαρμογής.

4.1.3 Γενική αρχιτεκτονική των CMS που βασίζονται στο NCommet

Κατά την υλοποίηση CMS συστημάτων με βάση το NCommet (όπως το σύστημα διαχείρισης Portals «SimpleCMSPlus» και το σύστημα διαχείρισης μαθημάτων «NCCMS»), διαπιστώθηκε ότι μεγάλο μέρος της λειτουργικότητας τους είναι κοινό, ανεξάρτητα από τις ειδικότερες απαιτήσεις του κάθε CMS. Για το λόγο αυτό, η κοινή λειτουργικότητα τοποθετήθηκε σε μια βιβλιοθήκη που ονομάστηκε *AbstractCMS*.

Για τα CMS «SimpleCMSPlus» και «NCCMS», η γενική αρχιτεκτονική τους πριν και μετά τη χρησιμοποίηση του AbstractCMS, φαίνεται παρακάτω:



Εικόνα 4-3 Αρχιτεκτονική των NCCMS και SimpleCMSPlus

Κατά την υλοποίηση μίας νέας CMS εφαρμογής το AbstractCMS και το NCommet χρησιμοποιούνται ως έχουν.

4.1.4 AbstractCMS

Το AbstractCMS ορίζει υπηρεσίες με τη μορφή interfaces (διεπαφές). Με την ίδια λογική όπως και στο NCommet, υπάρχουν υλοποιήσεις των διεπαφών οι οποίες βρίσκονται είτε στο AbstractCMS (προκαθορισμένες – default) είτε στο CMS είτε και στα δύο. Η αντιστοίχιση των διεπαφών με συγκεκριμένες υλοποιήσεις γίνεται στο αρχείο ρυθμίσεων της εφαρμογής.

Πολλές υπηρεσίες είναι αναγκαίες για τη λειτουργία του CMS. Όσες από αυτές είναι μερικώς υλοποιημένες από το AbstractCMS (με τη μορφή abstract κλάσεων), πρέπει να υλοποιηθούν στο CMS.

Στον παρακάτω πίνακα φαίνονται οι σημαντικότερες υπηρεσίες που παρέχει το AbstractCMS, ποιές από αυτές πρέπει να υλοποιήσει το CMS και για ποιές από αυτές προσφέρει το AbstractCMS μια πλήρη υλοποίηση.

Υπηρεσία	**Περιγραφή**	**Υπάρχει υλοποίηση στο AbstractCMS;**	**Απαιτείται υλοποίηση;**
Ιεραρχία ASPX Σελίδων	Δυνατότητα ορισμού των τύπων κόμβων που είναι πλοηγήσιμοι (δηλαδή μπορούν να παρουσιαστούν σε δική τους aspx ιστοσελίδα). Ένας κόμβος που δεν είναι πλοηγήσιμος συνήθως εμφανίζεται από το CMS ως περιεχόμενο στη σελίδα ενός πλοηγήσιμου πρόγονου.	Ναι	-

Αντιστοίχιση τύπων κόμβων και κλάσεων περιεχομένου	Δυνατότητα ορισμού των έγκυρων συνδυασμών τύπων κόμβων και κλάσεων περιεχομένου. Για παράδειγμα το CMS μπορεί να ορίσει ότι ο τύπος κόμβου «portal» μπορεί να έχει ως κλάση περιεχομένου την κλάση Portal.	Ναι	-
Διαχειριστής Attributes	Ανιχνεύει τα .NET Attributes που έχουν επισημανθεί στις κλάσεις περιεχομένου που ορίζονται από την CMS εφαρμογή.	Ναι	-
Business Model	Παρέχει μεθόδους που σχετίζονται με το περιεχόμενο. Προβλέπει διάφορα κοινά χαρακτηριστικά του περιεχομένου.	Όχι	Ναι
Υποστήριξη Φιλικών URLs	Παρέχει υποστήριξη για φιλικά URLs των πλοηγήσιμων κόμβων.	Ναι	-
Fireli Repositories	Διαχείριση αποθηκών αρχείων.	Ναι	-
Παροχέας URLs	Δίνει τα URLs προς τους κόμβους, τα αρχεία και το Backoffice.	Όχι	Ναι
Πολυγλωσσικότητα	Υποστηρίζονται πολυγλωσσικά portals μέσω συσχετίσεων του NCommet.	Όχι	Όχι
Ομαδοποίηση επεξεργάσιμων ιδιοτήτων	Το Backoffice εμφανίζει τις επεξεργάσιμες ιδιότητες σε ομάδες που ορίζονται στο αρχείο ρυθμίσεων του CMS.	Ναι	-
Υποστήριξη Εκδόσεων	Υποστηρίζει την αναβάθμιση του business model σε άλλη έκδοση και την τήρηση ιστορικού εκδόσεων.	Ναι	Όχι

Πίνακας 4-1 Πίνακας υπηρεσιών του AbstractCMS

4.1.4.1 Υπηρεσίες με επιπρόσθετες ρυθμίσεις στο αρχείο ρυθμίσεων

Κάποιες από τις παραπάνω υλοποιήσεις υπηρεσιών του AbstractCMS απαιτούν επιπρόσθετες ρυθμίσεις στο αρχείο ρυθμίσεων. Για παράδειγμα, η υπηρεσία *«Αντιστοίχιση τύπων κόμβων και κλάσεων περιεχομένου»* απαιτεί τον ορισμό των έγκυρων συνδυασμών στο αρχείο ρυθμίσεων.

4.1.4.2 Έτοιμα controls

Υπάρχουν έτοιμα controls που μπορούν να χρησιμοποιηθούν από τη CMS εφαρμογή.

4.1.4.3 Περιβάλλον διαχείρισης «Backoffice»

Στο AbstractCMS περιλαμβάνεται και ένα γενικού σκοπού περιβάλλον διαχείρισης (Backoffice) το οποίο λειτουργεί με οποιοδήποτε business model. Πρόκειται για μια web εφαρμογή με την οποία γίνεται η επεξεργασία της δενδρικής δομής των κόμβων καθώς και των αντικειμένων περιεχομένου των κόμβων (βλ. Εικόνα 2-7).

Για την επεξεργασία των αντικειμένων περιεχομένου, η λειτουργία του Backoffice στηρίζεται στην ανίχνευση των επεξεργάσιμων ιδιοτήτων της κλάσης περιεχομένου (μέσω κατάλληλων Attributes) και στη δυναμική δημιουργία controls για την επεξεργασία τους. Ένα παράδειγμα μιας ιδιότητας κειμένου είναι το εξής:

```
[AreaTextFieldEdit("Απλή περιγραφή")]
public virtual string DescriptionText { get; set; }
```

Το control επεξεργασίας που θα δείξει το backoffice είναι το εξής:

Εικόνα 4-4 Control επεξεργασίας του Backoffice

4.1.5 Διασύνδεση του SimpleCMSPlus με το AbstractCMS και το NCommet

4.1.5.1 Διασύνδεση των υπηρεσιών του AbstractCMS με τις υλοποιήσεις

Η διασύνδεση των υπηρεσιών του AbstractCMS και των υλοποιήσεων που επιθυμεί να χρησιμοποιήσει το SimpleCMSPlus ορίζεται στο αρχείο ρυθμίσεων web.config του SimpleCMSPlus. Για παράδειγμα, για τις υπηρεσίες *«Business Model»*, *«Παροχέας URLs»*, το web.config περιλαμβάνει τα εξής:

```
<abstractcms
    businessModel="SimpleCMSPlus.BusinessModel.AbstractCMSImpl.B
usinessModel, SimpleCMSPlus.BusinessModel"
    urlHelper="SimpleCMSPlus.BusinessModel.AbstractCMSImpl.UrlHe
lper, SimpleCMSPlus.BusinessModel">
</abstractcms>
```

4.1.5.1.1 Επιπλέον ρυθμίσεις για υπηρεσίες του AbstractCMS

Για άλλες υπηρεσίες του AbstractCMS, πιθανώς να χρειάζονται επιπλέον ρυθμίσεις στο web.config. Για παράδειγμα, για τις υπηρεσίες *«Αντιστοίχιση τύπων κόμβων και κλάσεων περιεχομένου»* και *«Ιεραρχία ASPX Σελίδων»*, το web.config του SimpleCMSPlus περιλαμβάνει τα εξής:

```
<itemContentCombos>
```

```
    <add key="root" itemType="root"
contentType="SimpleCMSPlus.BusinessModel.Root,
SimpleCMSPlus.BusinessModel" contentClass="1" isPortal="false"
isAspxItem="true"/>
    <add key="portal" itemType="portal"
contentType="SimpleCMSPlus.BusinessModel.Portal,
SimpleCMSPlus.BusinessModel" contentClass="1" isPortal="true"
isAspxItem="true"/>
    …
</itemContentCombos>
```

Στον παρακάτω πίνακα βλέπουμε συνοπτικά όλους τους συνδυασμούς που ορίζει.

Key του συνδυασμού	**Τύπος κόμβου**	**Κλάση περιεχομένου**
Κατηγορία "Navigation" (1)		
Root	"root"	Root
Portal	"portal"	Portal
Σελίδα	"page"	Page
Κατηγορία "Modules" (2)		
Φάκελος ανακοινώσεων	"ann_list"	Module
Φάκελος περιεχομένου	"cont_list"	Module
Φάκελος προσώπων	"prsn_list"	Module
Κατηγορία "Items" (3)		
Έγγραφο	"cont_item"	Document
Κείμενο Html	"cont_item"	Html
Σύνδεσμος	"cont_item"	WebLink
Πρόσωπο	"prsn_item"	Person

Πίνακας 4-2 Συνδυασμοί τύπων κόμβων και κλάσεων περιεχομένου που ορίζει το SimpleCMSPlus

Τους κόμβους του πίνακα τους επισημάναμε στο στιγμιότυπο οθόνης στην Εικόνα 2-5.

4.1.5.2 Διασύνδεση των υπηρεσιών του NCommet με τις υλοποιήσεις

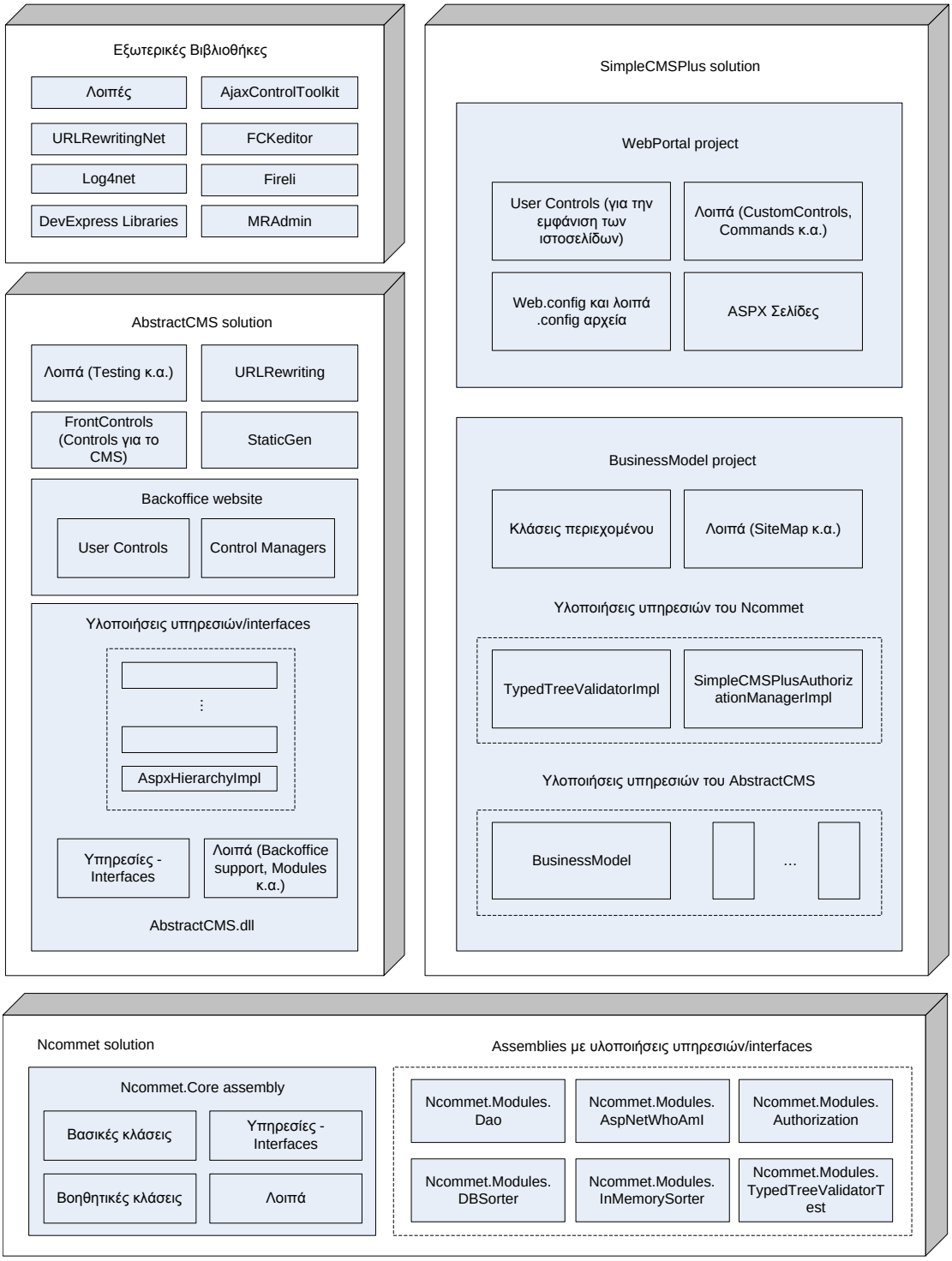
Η διασύνδεση των υπηρεσιών του NCommet και των υλοποιήσεων που επιθυμεί να χρησιμοποιήσει το SimpleCMSPlus ορίζεται στο αρχείο ρυθμίσεων web.config. Για παράδειγμα, για την *υπηρεσία αποθήκευσης* και την *υπηρεσία των κανόνων εγκυρότητας*:

```
 <ncommet persister="NCommet.Modules.Dao.Storage,
NCommet.Modules.Dao"
          daoEventsSink="NCommet.Modules.Dao.DaoInterceptor,
NCommet.Modules.Dao"
         typedTreeValidator="SimpleCMSPlus.BusinessModel.
TypedTreeValidatorImpl, SimpleCMSPlus.BusinessModel" >
    <modules>…</modules>
</ncommet>
```

Σημείωση: Η υλοποίηση της *υπηρεσίας αποθήκευσης*, που αναπτύξαμε με την παρούσα διπλωματική εργασία και περιλαμβάνεται στο assembly NCommet.Modules.Dao, χρειάζεται και άλλες ρυθμίσεις στο web.config. Θα τις περιγράψουμε αναλυτικότερα στην ενότητα 6.1.3.3.

4.1.6 Διάγραμμα των NCommet, AbstractCMS και SimpleCMSPlus

Παρουσιάζουμε ένα γενικό block διάγραμμα των solutions που απαρτίζουν τη CMS εφαρμογή του SimpleCMSPlus.



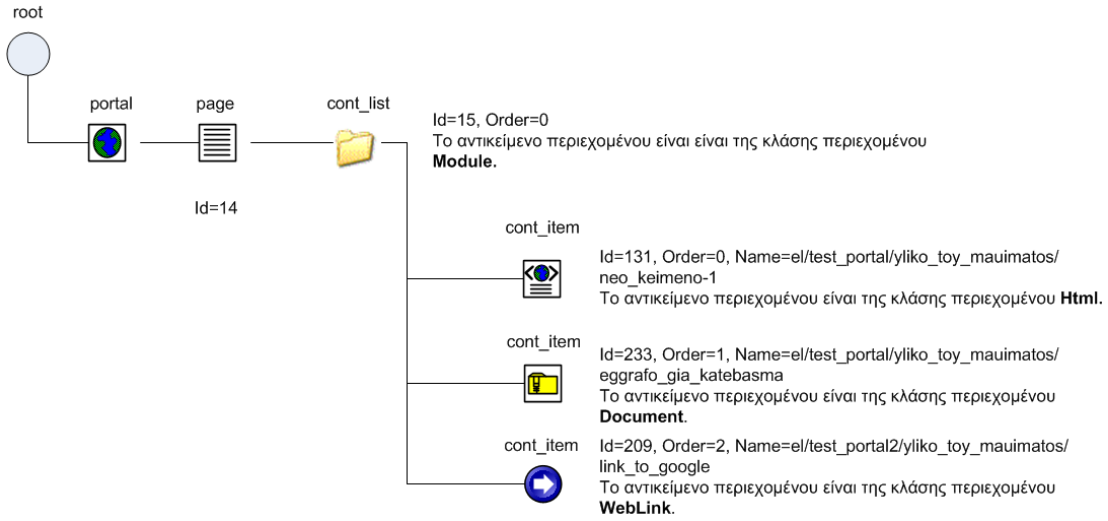
Εικόνα 4-5 Block διάγραμμα των NCommet, AbstractCMS και SimpleCMSPlus

4.1.7 Παράδειγμα λειτουργίας της πολύ-επίπεδης αρχιτεκτονικής

Θα πραγματοποιήσουμε μια top-down ανάλυση της διαδικασίας με την οποία το SimpleCMSPlus εμφανίζει ένα Φάκελο Περιεχομένου. Επικεντρωνόμαστε μόνο στο φάκελο και δεν ξεφεύγουμε σε περιγραφές άλλων κομματιών της ιστοσελίδας ή άλλων ιδιάζουσων περιπτώσεων. Θέλουμε μόνο να δείξουμε τον τρόπο με τον οποίο το SimpleCMSPlus

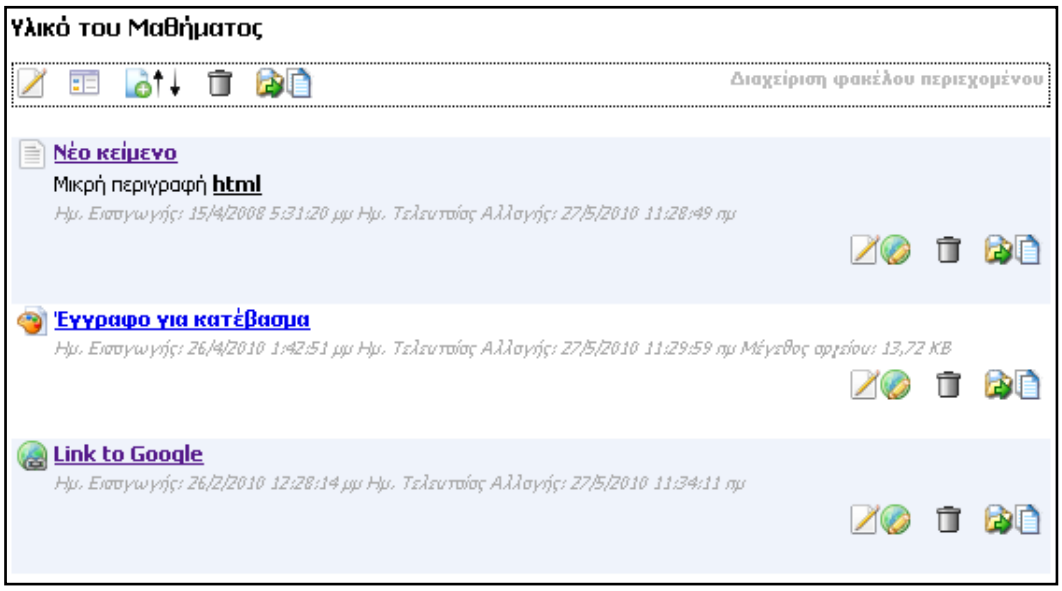
χρησιμοποιεί τις διορισμένες υλοποιήσεις των υπηρεσιών του AbstractCMS και του NCommet.

Έστω η εξής δενδρική δομή κόμβων του SimpleCMSPlus.



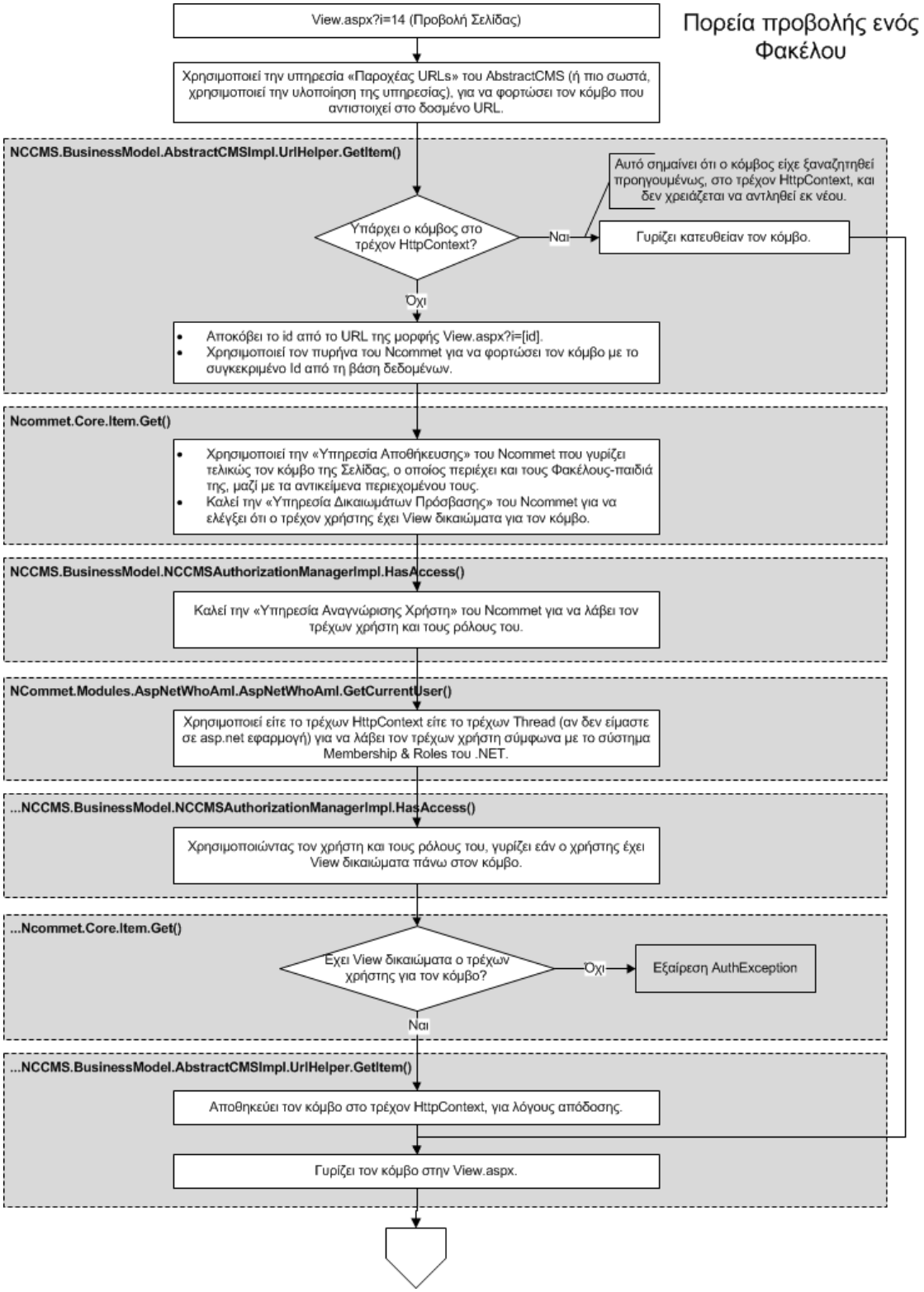
Εικόνα 4-6 Δενδρική δομή κόμβων του παραδείγματος

Έστω ότι έχουμε συνδεθεί στην ιστοσελίδα ως Διαχειριστής (ώστε να βλέπουμε και τις εργαλειοθήκες διαχείρισης). Αν πλοηγηθούμε στη σελίδα View.aspx?i=14 (το id της Σελίδας που περιέχει τον Φάκελο), ο Φάκελος θα παρουσιαστεί ως εξής:

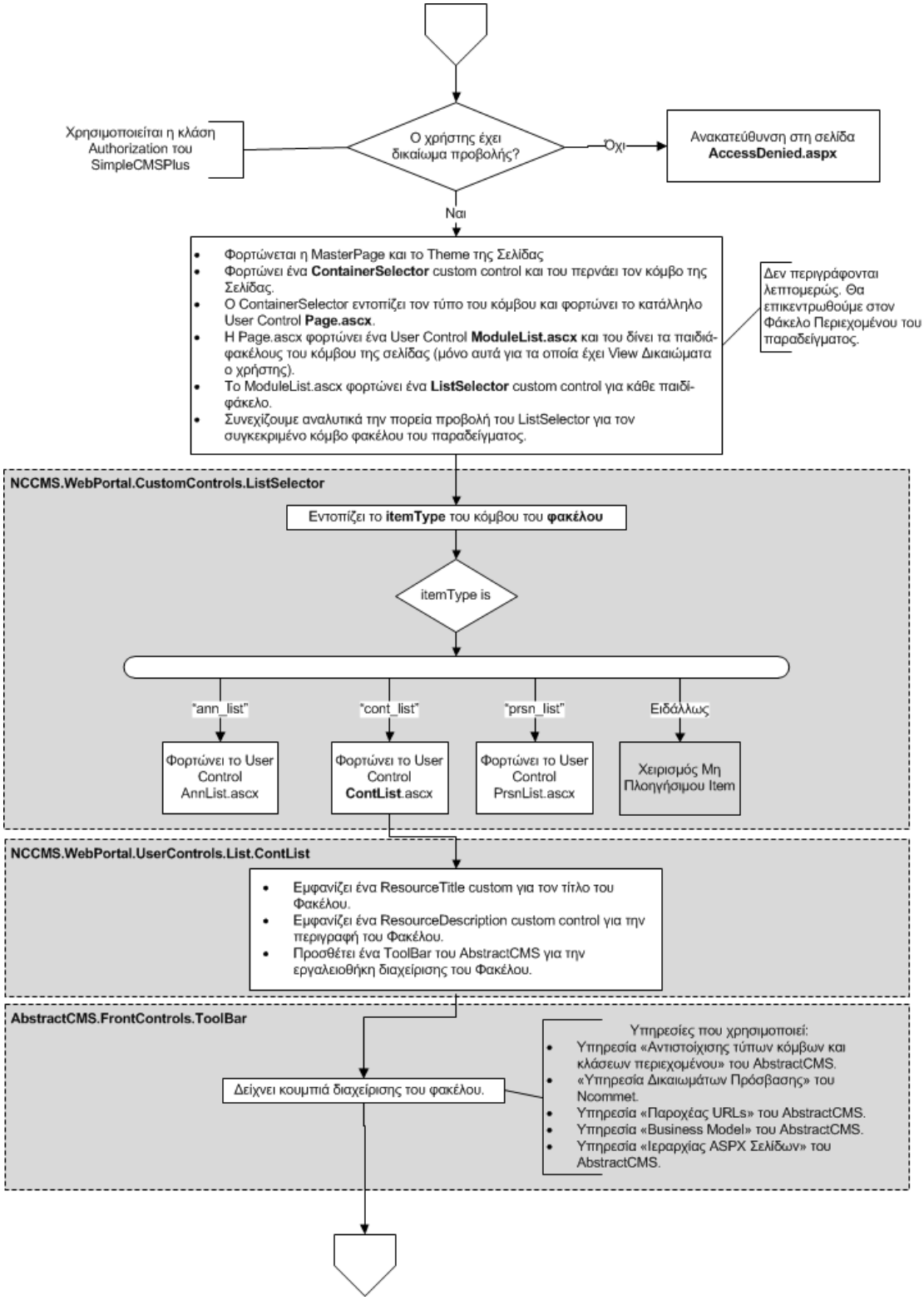


Εικόνα 4-7 Παρουσίαση του φακέλου του παραδείγματος από το SimpleCMSPlus

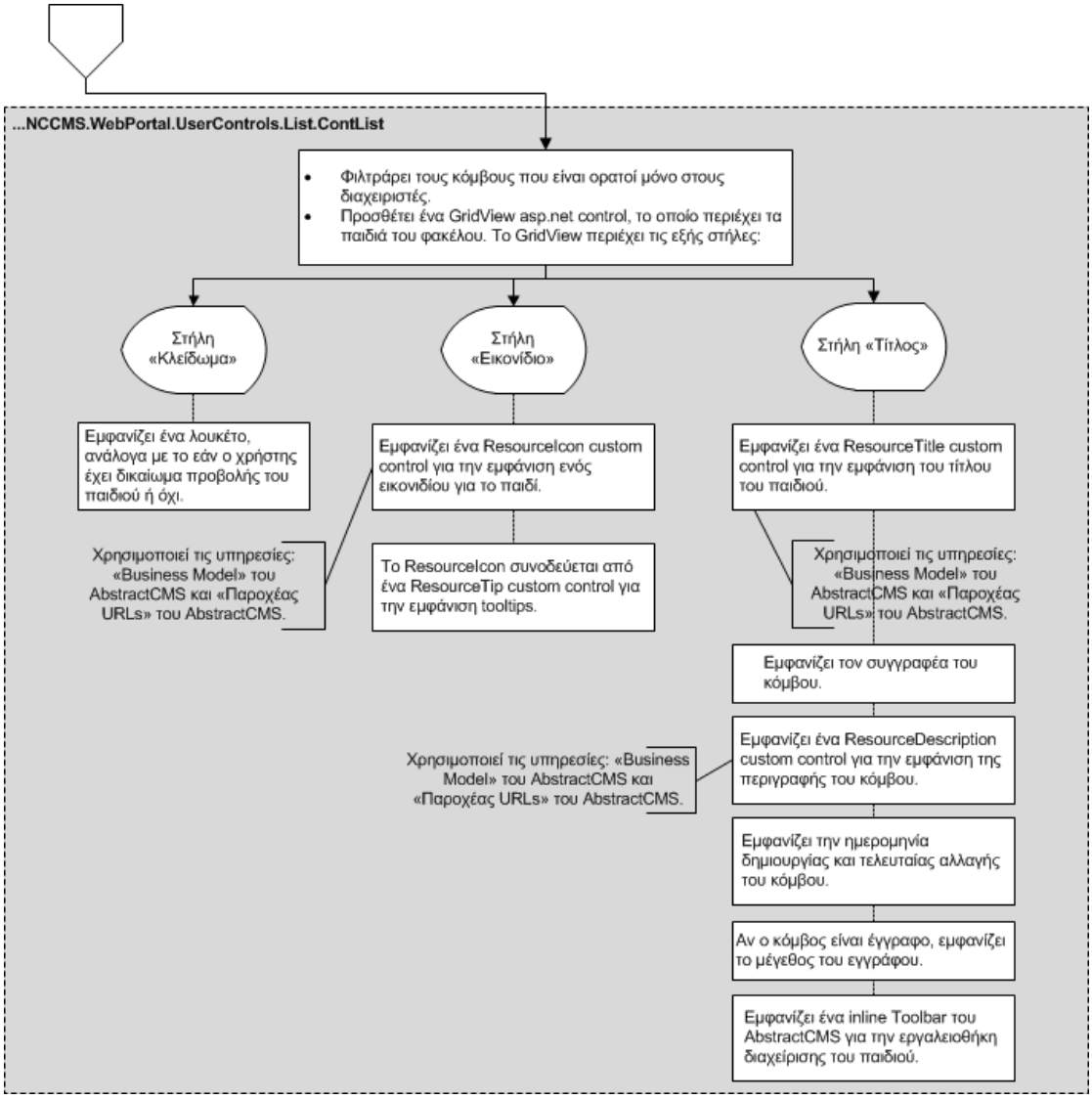
Παρακάτω, επεξηγούμε την πορεία εμφάνισης του φακέλου με τη βοήθεια flows.



Εικόνα 4-8 Πορεία προβολής ενός Φακέλου του SimpleCMSPlus, μέρος 1/3



Εικόνα 4-9 Πορεία προβολής ενός Φακέλου του SimpleCMSPlus, μέρος 2/3



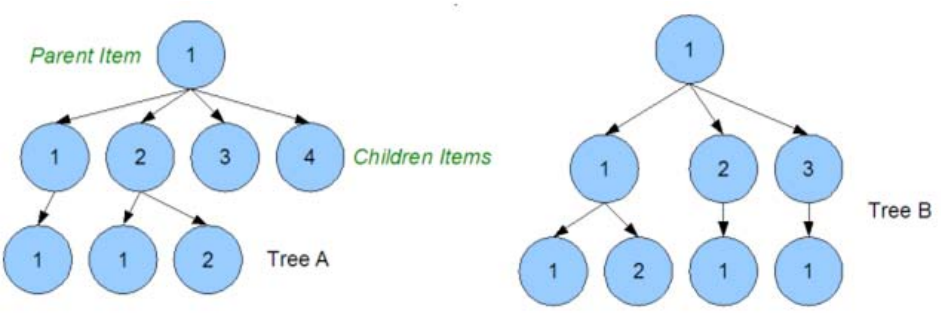
Εικόνα 4-10 Πορεία προβολής ενός Φακέλου του SimpleCMSPlus, μέρος 3/3

Παρατηρείστε ότι η νέα διεπαφή της «Υπηρεσίας Αποθήκευσης» που αναπτύξαμε, έχει τόσο χαμηλή σύζευξη με το AbstractCMS και το SimpleCMSPlus ώστε καλείται μόνο στην αρχή του flow για να πάρει τον κόμβο προς εμφάνιση. Μετέπειτα, οι λειτουργίες του AbstractCMS και του SimpleCMSPlus, που δεν θα μας απασχολήσουν περισσότερο, απλώς διαχειρίζονται τον κόμβο που επέστρεψε το ORM εργαλείο (το NHibernate). Δεν χρειάζονται άλλες κλήσεις προς την Υπηρεσία Αποθήκευσης, αφού χρησιμοποιώντας lazy-loading ανακτώνται δυναμικά τυχόν σχετιζόμενα αντικείμενα που θέλει (παιδιά, συσχετίσεις κ.α.).

4.2 Επισκόπηση του NCommet

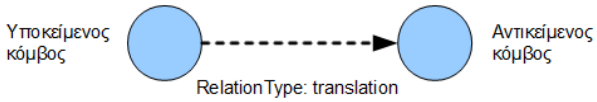
Το NCommet είναι μια γενική βιβλιοθήκη που αποσκοπεί στην οργάνωση πληροφορίας μέσω δενδρικών δομών.

Πιο αναλυτικά για το NCommet, μπορούμε να έχουμε κόμβους-ρίζες (root nodes) κάτω από τους οποίους δημιουργείται ένα δένδρο. Ένα δένδρο περιέχει κόμβους με συγκεκριμένους **τύπους** (typed tree). Τα παιδιά ενός κόμβου έχουν συγκεκριμένη **σειρά** (ordered tree). Ένα παράδειγμα δένδρων που μπορούν να δημιουργηθούν με το NCommet είναι το εξής:



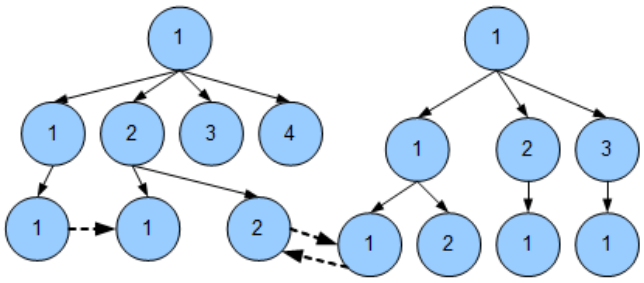
Εικόνα 4-11 Παράδειγμα δένδρων που μπορούν να δημιουργηθούν με το NCommet

Επιπρόσθετα, το NCommet προσφέρει τη δυνατότητα να ορισθούν **συσχετίσεις** μεταξύ δύο κόμβων. Συγκεκριμένα, μια συσχέτιση ορίζεται μεταξύ ενός υποκείμενου κόμβου (Owner ή Subject item) και ενός αντικείμενου κόμβου (Relative ή Object item), δηλαδή η συσχέτιση έχει κατεύθυνση (directed relation). Μια συσχέτιση έχει έναν συγκεκριμένο τύπο (typed relation) που καθορίζει το είδος της.



Εικόνα 4-12 Μια συσχέτιση του NCommet

Με αυτή τη δυνατότητα, το NCommet μπορεί να σχηματίσει κατευθυνόμενους γράφους. Γενικά, δεν προσφέρει τις τυπικές λειτουργίες διαχείρισης γράφων (εύρεση ελάχιστου μονοπατιού κλπ.).



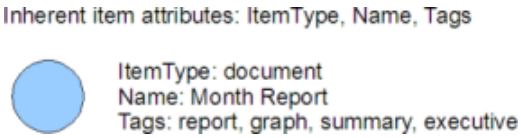
Εικόνα 4-13 Δυνατότητα σχηματισμού κατευθυνόμενων γράφων μέσω συσχετίσεων

4.2.1 Ιδιότητες κόμβων

Η λειτουργικότητα που προσφέρει το NCommet για τη διαχείριση δένδρων υπερβαίνει τη λειτουργικότητα συνηθισμένων συστημάτων διαχείρισης δένδρων, γιατί επικεντρώνεται στην περικλειόμενη πληροφορία των κόμβων (items). Στις τελικές εφαρμογές, οι κόμβοι (items) αντιστοιχούν σε οντότητες (entities) του κόσμου.

Ένας κόμβος έχει τις εξής ιδιότητες (properties):

- **Όνομα (name).** Χρησιμεύει ως τίτλος του κόμβου. Δεν έχει ιδιαίτερη σημασία για τη δενδρική ιεραρχία.
- **Τύπος κόμβου (ItemType)**. Χρησιμεύει ως η ειδοποιός διαφορά μεταξύ των κόμβων. Οι διάφοροι κόμβοι μπορούν να κατηγοριοποιηθούν ανά τον τύπο κόμβο και να οριστούν κανόνες ως προς την ιεραρχία των κόμβων. Είναι μια υποχρεωτική ιδιότητα που **δεν** μπορεί να αλλάξει μόλις ορισθεί.
- **Λεξάντες (tags)**. Χρησιμεύουν ως λεξάντες/ετικέτες των κόμβων, δίνοντας επιπρόσθετες πληροφορίες κατηγοριοποίησης. Δεν έχουν ιδιαίτερη σημασία για τη δενδρική ιεραρχία.



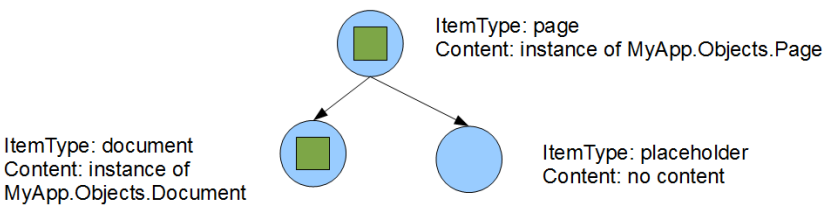
Εικόνα 4-14 Ιδιότητες των κόμβων του NCommet

4.2.2 Αντικείμενα περιεχομένου

Για την αποθήκευση οποιασδήποτε πληροφορίας στους κόμβους, κάθε κόμβος μπορεί να κατέχει ένα αντικείμενο περιεχομένου (content object) που περιέχει όλες τις επιπλέον πληροφορίες που χρειάζεται η εφαρμογή.

Το αντικείμενο περιεχομένου μπορεί να είναι στιγμιότυπο οποιασδήποτε .NET κλάσης μιας εφαρμογής. Οι κλάσεις των αντικειμένων περιεχομένου λέγονται κλάσεις περιεχομένου.

Με την προσθήκη αντικειμένων περιεχομένου, το NCommet γίνεται ένα πλήρως λειτουργικό σύστημα διαχείρισης περιεχομένου (content management system). Ένα παράδειγμα ενός δένδρου που θα μπορούσε να υλοποιήσει ένας ιστότοπος είναι το εξής:



Εικόνα 4-15 Αντικείμενα περιεχομένου

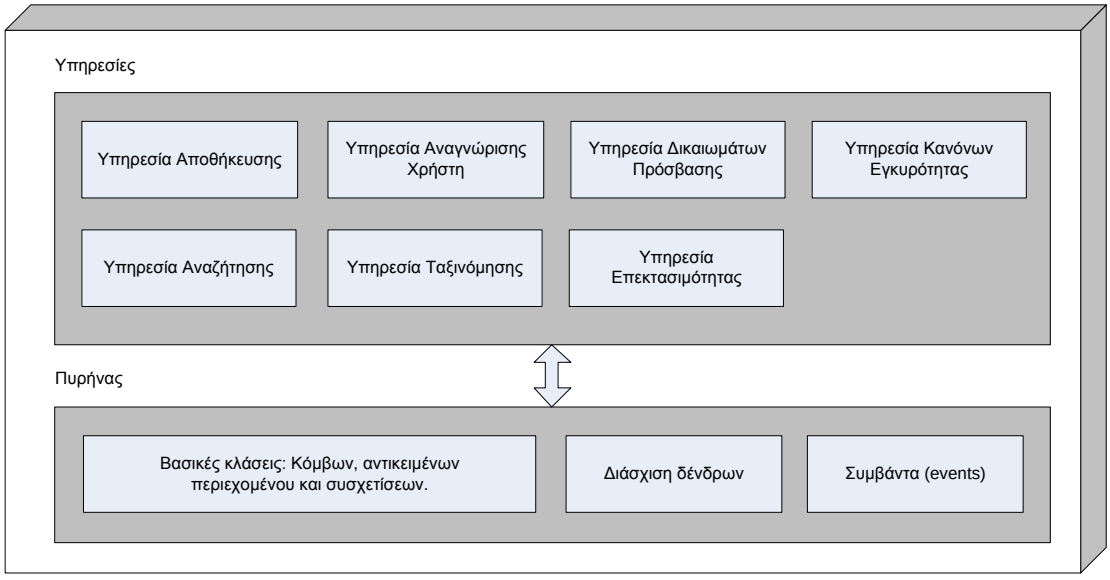
Συνήθως στις εφαρμογές, για κάθε τύπο κόμβου θα αντιστοιχεί και μια συγκεκριμένη κλάση περιεχομένου. Όμως αυτό δεν είναι αναγκαίο: Ένας τύπος κόμβου μπορεί να χρησιμοποιηθεί με πολλές κλάσεις περιεχομένου, και αντίστροφα.

4.2.3 Υπηρεσίες του NCommet

Πέραν των βασικών λειτουργιών διαχείρισης δενδρικών δομών, το NCommet υποστηρίζει και άλλες λειτουργίες οι οποίες έχουν διαχωριστεί σε υπηρεσίες (υποσυστήματα). Αυτές τις υπηρεσίες τις απαριθμήσαμε στην σύντομη περιγραφή του NCommet που έγινε στην ενότητα 4.1.2 «NCommet». Στις υπηρεσίες συμπεριλαμβάνεται και η «Υπηρεσία αποθήκευσης» που υποστηρίζει την ανάκτηση και την αποθήκευση των κόμβων, των αντικειμένων περιεχομένου και των συσχετίσεων.

Κάθε υπηρεσία έχει υψηλή συνεκτικότητα καθεαυτή και χαμηλή σύζευξη με τις υπόλοιπες υπηρεσίες του NCommet. Η σύζευξη της κάθε υπηρεσίας περιορίζεται με τον πυρήνα του NCommet.

4.2.4 Αρχιτεκτονική



Εικόνα 4-16 Block διάγραμμα των συστατικών του NCommet

4.3 Λειτουργικές απαιτήσεις

Στην παρούσα ενότητα, θα απαριθμήσουμε τις λειτουργικές απαιτήσεις της νέας Υπηρεσίας Αποθήκευσης που σκοπεύει να υλοποιήσει η παρούσα διπλωματική εργασία χρησιμοποιώντας τεχνικές αντικειμενο-σχεσιακής απεικόνισης.

Όμως, λόγω της φύσης της, η Υπηρεσίας Αποθήκευσης παρουσιάζει υψηλή σύζευξη με τον πυρήνα του NCommet. Αυτό είναι φυσικό αφού αρκετές λειτουργίες προϋποθέτουν την

άντληση των κόμβων από τη βάση δεδομένων. Άλλες λειτουργίες μεταβάλλουν τους ίδιους τους κόμβους ή την δενδρική δομή, με αποτέλεσμα να καλείται η Υπηρεσία Αποθήκευσης να διατηρήσει τις αλλαγές στη βάση δεδομένων. Γι' αυτό το λόγο, παρουσιάζουμε επιπλέον τις λειτουργικές απαιτήσεις του πυρήνα του NCommet.

Παραλείπουμε τις λειτουργικές απαιτήσεις των υπολοίπων υπηρεσιών του NCommet, οι οποίες δε σχετίζονται άμεσα με την Υπηρεσία Αποθήκευσης.

4.3.1 Πυρήνας

4.3.1.1 Διαχείριση κόμβων

- Δημιουργία ενός κόμβου, με συγκεκριμένο όνομα, τύπο.
- Τροποποίηση του αντικείμενου περιεχομένου ενός κόμβου.
- Τροποποίηση της ημερομηνίας δημιουργίας ενός κόμβου.
- Τροποποίηση τους ονόματος ενός κόμβου.
- Τροποποίηση των λεζάντων ενός κόμβου.
- Αποθήκευση ενός κόμβου, του αντικείμενου περιεχομένου του και των συσχετίσεων του.

4.3.1.2 Διαχείριση δενδρικής δομής

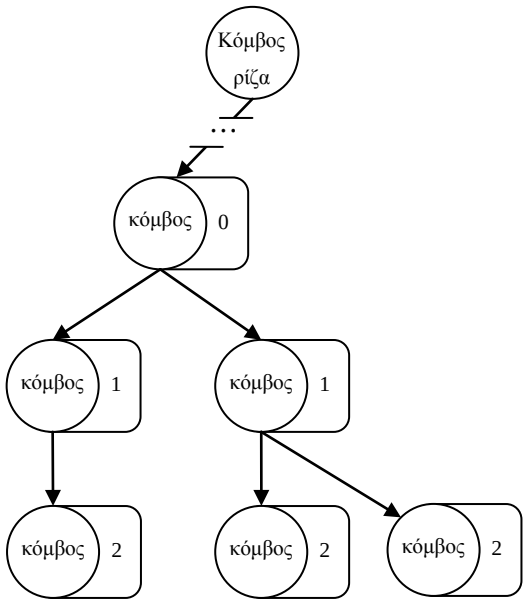
- Προσθήκη ενός κόμβου κάτω από έναν υπάρχοντα κόμβο γονέα.
- Επιστροφή της θέσης ενός κόμβου ανάμεσα στα αδέρφια του.
- Μετακίνηση ενός κόμβου σε συγκεκριμένη θέση ανάμεσα στα αδέρφια του (μετακινώντας κατάλληλα τα αδέρφια που επηρεάζονται).
- Εναλλαγή (swap) δύο αδερφιών.
- Επιστροφή των παιδιών ενός κόμβου (με δυνατότητα ορισμού του αριθμού των παιδιών), σύμφωνα με τη σειρά τους.
- Επιστροφή του γονέα ενός κόμβου.
- Επιστροφή των παιδιών ενός κόμβου με φιλτράρισμα ανάλογα με τον τύπο τους.
- Έλεγχος για το αν ένας κόμβος περιέχει παιδί συγκεκριμένου τύπου.
- Έλεγχος για το αν ένας κόμβος είναι γονέας ενός κόμβου.
- Διαγραφή ενός κόμβου και των παιδιών του.
- Διαγραφή ενός παιδιού και των παιδιών του.
- Διαγραφή των παιδιών ενός κόμβου.
- Απόσπαση ενός κόμβου από τον γονέα του. Γίνεται έτσι κόμβος-ρίζα.

- Απόσπαση ενός παιδιού από τον γονέα του. Γίνεται έτσι κόμβος-ρίζα.
- Αλλαγή του γονέα ενός κόμβου.
- Κλωνοποίηση ενός κόμβου και του αντικείμενου περιεχομένου του (shallow clone).
- Κλωνοποίηση ενός κόμβου και του υπόδενδρου του και των αντικειμένων περιεχομένων τους (deep clone).
- Κλωνοποίηση ενός κόμβου και του υπόδενδρου του και των αντικειμένων περιεχομένων τους (deep clone), με ταυτόχρονη τοποθέτηση του κλώνου κάτω από καθορισμένο κόμβο γονέα.
- Ταυτοποίηση (εάν δύο κόμβοι είναι ο ίδιος κόμβος ή όχι, όσον αφορά την απεικόνιση τους στη βάση δεδομένων).
- Υπολογισμός του επιπέδου που βρίσκεται ένας κόμβος μέσα σε ένα δένδρο.
- Επιστροφή της ρίζας του δένδρου μέσα στο οποίο βρίσκεται ένας κόμβος.

4.3.1.3 Διάσχιση δένδρων

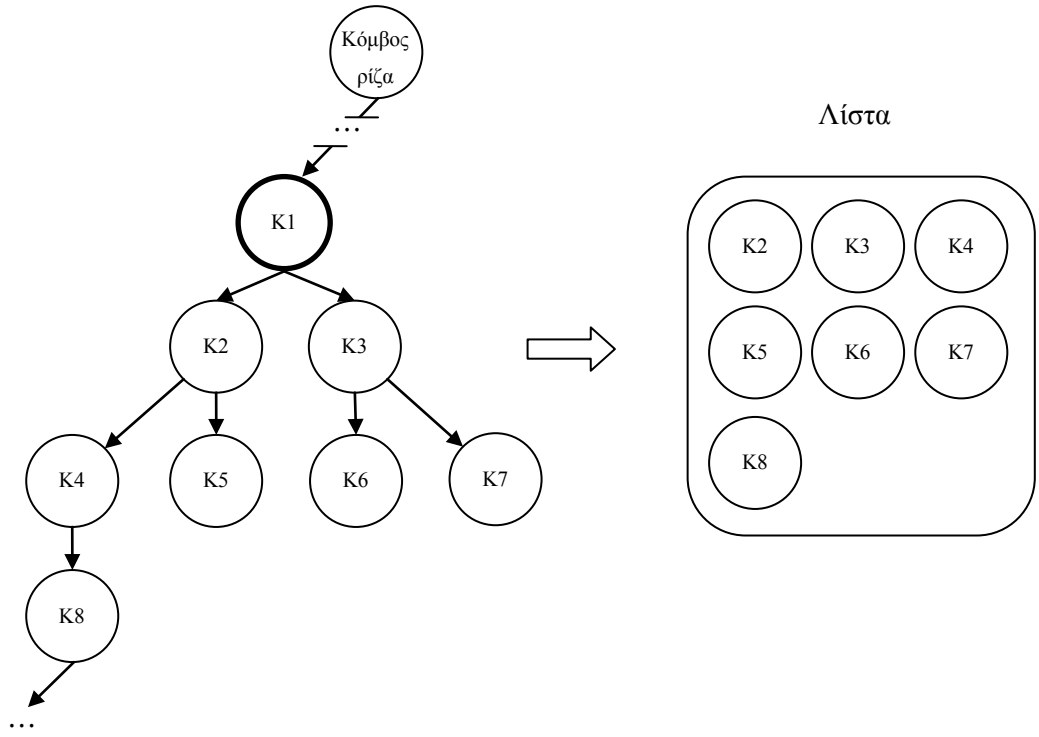
- Εύρεση των κόμβων ριζών του δάσους[1] (με δυνατότητα περιορισμού σε συγκεκριμένους τύπους κόμβων).
- Εύρεση του μονοπατιού των προγόνων ενός κόμβου (με δυνατότητα περιορισμού σε τύπους κόμβων που θέλουμε και στο επίπεδο μέχρι το οποίο θέλουμε να ψάξουμε).
- Εύρεση του πιο κοντινού προγόνου με συγκεκριμένο τύπο κόμβου.
- Εύρεση του προγόνου που βρίσκεται συγκεκριμένα επίπεδα ψηλότερα.
- Έλεγχος για το αν ένας κόμβος είναι πρόγονος ενός κόμβου.
- Εύρεση της ιεραρχίας απογόνων ενός κόμβου, όπου κάθε κόμβος θα συνοδεύεται και από το επίπεδο (σχετικό ως προς τον δοσμένο κόμβο) στο οποίο βρίσκεται (με δυνατότητα περιορισμού σε τύπους κόμβων που θέλουμε, σε λεζάντες που θέλουμε οι κόμβοι να έχουν και στο επίπεδο μέχρι το οποίο θέλουμε να ψάξουμε). Για παράδειγμα:

[1] Δάσος είναι το σύνολο όλων των κόμβων (σε όποιο δένδρο κι αν ανήκουν).



Εικόνα 4-17 Εύρεση της ιεραρχίας απογόνων ενός κόμβου

- Εύρεση των απογόνων ενός κόμβου που έχουν συγκεκριμένους τύπους κόμβων, με δυνατότητα ορισμού του μέγιστου βάθους μέχρι το οποίο θέλουμε να ψάξουμε. Για παράδειγμα, έστω ότι αναζητούμε τους απογόνους τους κόμβου Κ1, με μέγιστο βάθος 3:



Εικόνα 4-18 Εύρεση των απογόνων ενός κόμβου

- Εύρεση του αριθμού των απογόνων ενός κόμβου.

4.3.1.4 Διαχείριση συσχετίσεων

- Δημιουργία μιας συσχέτισης μεταξύ ενός υποκείμενου κόμβου (Owner ή Subject item) και ενός αντικείμενου κόμβου (Relative ή Object item), με συγκεκριμένο τύπο σχέσης. Σημείωση: Δεν μπορούν να υπάρχουν πολλαπλές σχέσεις μεταξύ δύο κόμβων με τον ίδιο τύπο σχέσης.
- Επιστροφή των συσχετίσεων ενός υποκείμενου κόμβου (με δυνατότητα φιλτραρίσματος ανάλογα με τον τύπο σχέσης και τον τύπο των αντικείμενων κόμβων).
- Επιστροφή των συσχετίσεων ενός αντικείμενου κόμβου.
- Έλεγχος για το αν δύο κόμβοι συσχετίζονται.
- Έλεγχος για το αν δύο κόμβοι συσχετίζονται με συγκεκριμένο τύπο σχέσης.
- Διαγραφή μιας συσχέτισης.
- Διαγραφή των συσχετίσεων ενός κόμβου, στις οποίες συμμετέχει ως υποκείμενος κόμβος.
- Διαγραφή των συσχετίσεων ενός κόμβου, στις οποίες συμμετέχει ως αντικείμενος κόμβος.

4.3.1.5 Συμβάντα (events)

- Θα πρέπει να υποστηρίζονται συμβάντα (events), σύμφωνα με το αντικειμενοστραφές μοντέλο προγραμματισμού.
- Τα συμβάντα πρέπει να περιλαμβάνουν:
    - Τη δημιουργία ενός κόμβου.
    - Την προσάρτηση ενός κόμβου σε έναν κόμβο γονέα.
    - Πριν την αποθήκευση ενός κόμβου.
    - Μετά την αποθήκευση ενός κόμβου.
    - Πριν την αλλαγή γονέα ενός κόμβου, με δυνατότητα ακύρωσης της ενέργειας.
    - Μετά την αλλαγή γονέα ενός κόμβου.
    - Πριν τη διαγραφή ενός κόμβου, με δυνατότητα ακύρωσης της ενέργειας.
    - Μετά τη διαγραφή ενός κόμβου.
    - Την αλλαγή σειράς των παιδιών ενός κόμβου.

- Την διαγραφή των παιδιών ενός κόμβου.
- Την shallow κλωνοποίηση ενός κόμβου.
- Την deep κλωνοποίηση ενός κόμβου.

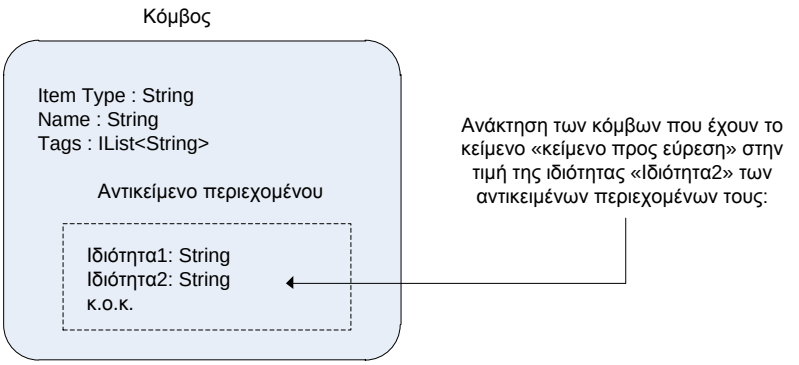
4.3.2 Νέα Υπηρεσία Αποθήκευσης

Τα αντικείμενα του NCommet (κόμβοι, συσχετίσεις και αντικείμενα περιεχομένου) πρέπει να μπορούν να αποθηκευτούν σε μόνιμο αποθηκευτικό μέσο. Μια αναγκαία υλοποίηση της υπηρεσίας κρίνεται η αποθήκευση των αντικειμένων σε μια σχεσιακή βάση δεδομένων (RDBMS) με αντικειμενο-σχεσιακές τεχνικές (ORM).

Η υπηρεσία αποθήκευσης απαιτεί ένας κόμβος να χαρακτηρίζεται από έναν μοναδικό αναγνωριστικό αριθμό (ID), βάσει του οποίου αποθηκεύεται στη βάση δεδομένων.

Οι λειτουργικές απαιτήσεις της υπηρεσίας αποθήκευσης είναι οι εξής:

- Κάθε κόμβος που έχει αποθηκευτεί στη βάση δεδομένων, πρέπει να χαρακτηρίζεται από έναν μοναδικό αναγνωριστικό αριθμό (ID).
- Να ανακτά έναν κόμβο που έχει συγκεκριμένο ID.
- Να ανακτά τα IDs των παιδιών ενός κόμβου με συγκεκριμένο ID.
- Να ανακτά τους κόμβους που έχουν συγκεκριμένα IDs.
- Να ανακτά όλους τους κόμβους.
- Να ανακτά τους κόμβους που έχουν συγκεκριμένο όνομα.
- Να ανακτά τους κόμβους που έχουν συγκεκριμένο pattern τύπου. Το pattern είναι μορφής που δέχεται ο τελεστής LIKE της SQL. Για παράδειγμα αν θέλουμε την ανάκτηση των κόμβων με τύπο «%t», θα μας γυρίσει κόμβους με τύπους «post», «comment» κ.ο.κ.
- Να ανακτά τους κόμβους που περιέχουν ένα αντικείμενο περιεχομένου που περιλαμβάνει κάποιο δεδομένο string σε μια συγκεκριμένη ιδιότητά του (με δυνατότητα περιορισμού σε συγκεκριμένο τύπο κόμβου).

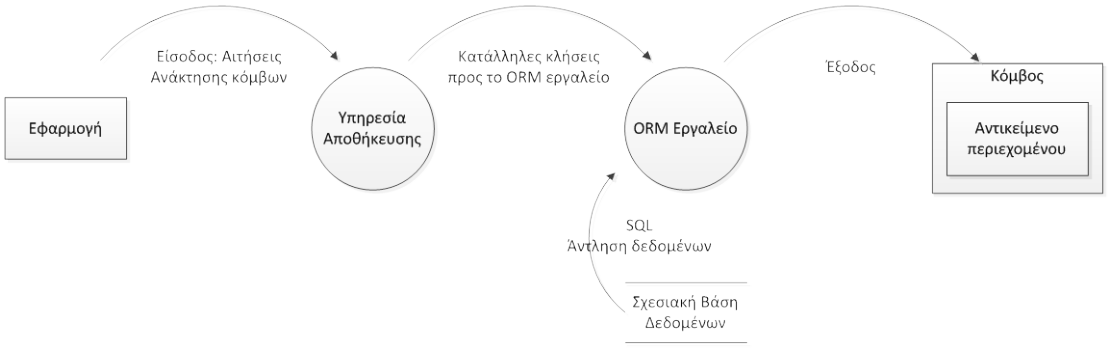


Εικόνα 4-19 Ανάκτηση κόμβων βάσει ενός δοσμένου κειμένου

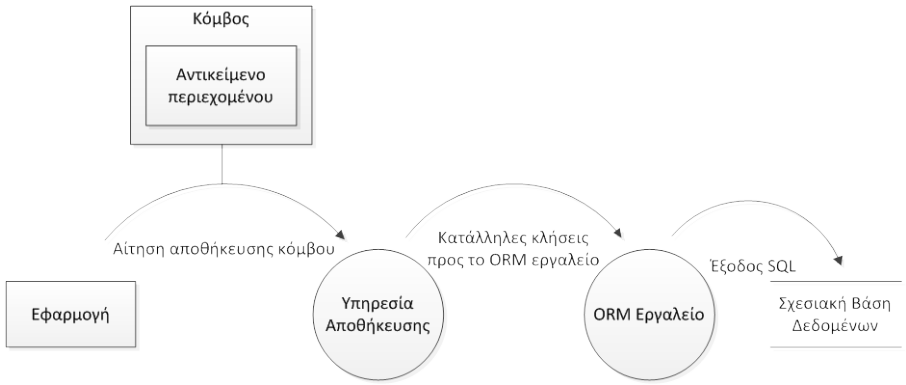
- Να ανακτά τον αριθμό των κόμβων που έχουν συγκεκριμένο όνομα.
- Να αποθηκεύει ένα οποιοδήποτε αντικείμενο.
- Να διαγράφει ένα οποιοδήποτε αντικείμενο.
- Να ανακτά τους κόμβους ρίζες (με δυνατότητα περιορισμού σε συγκεκριμένους τύπους κόμβων).
- Να ανακτά τα παιδιά ενός κόμβου που έχουν συγκεκριμένο τύπο.
- Να ανακτά τα παιδιά ενός κόμβου που έχουν συγκεκριμένους τύπους και συγκεκριμένες λεζάντες.
- Να ανακτά τα παιδιά ενός κόμβου που έχουν συγκεκριμένους τύπους.
- Να ανακτά τις συσχετίσεις στις οποίες συμμετέχει ένας κόμβος ως υποκείμενος κόμβος (με δυνατότητα περιορισμού σε συγκεκριμένους τύπους συσχετίσεων και σε συγκεκριμένους τύπους αντικείμενων κόμβων).
- Να ανακτά τις συσχετίσεις στις οποίες συμμετέχει ένας κόμβος ως αντικείμενος κόμβος.
- Να διαγράφει τις συσχετίσεις ενός κόμβου, με δυνατότητα ορισμού του τύπου της συσχέτισης και του αν ο κόμβος συμμετέχει στη συσχέτιση ως υποκείμενος ή αντικείμενος κόμβος.
- Να διαγράφει τις συσχετίσεις ενός κόμβου και αναδρομικά και των παιδιών του, με δυνατότητα ορισμού του τύπου της συσχέτισης και του αν ο κόμβος συμμετέχει στη συσχέτιση ως υποκείμενος ή αντικείμενος κόμβος.
- Να ξεκινά μια συναλλαγή (transaction).
- Να κάνει commit μια συναλλαγή (transaction).
- Να κάνει rollback μια συναλλαγή (transaction).

- Να παρέχει τρόπο ελέγχου για το αν ένας κόμβος είναι παροδικός (transient).
- Να παρέχει τρόπο ελέγχου για το αν ένα αντικείμενο περιεχομένου είναι παροδικό (transient).

Παρουσιάζουμε μερικά διαγράμματα ροής δεδομένων για την καλύτερη κατανόηση της λειτουργικότητας της υπηρεσίας αποθήκευσης, όσον αφορά την ανάκτηση και την τροποποίηση κόμβων.



Εικόνα 4-20 Ανάκτηση κόμβων με την υπηρεσία Αποθήκευσης

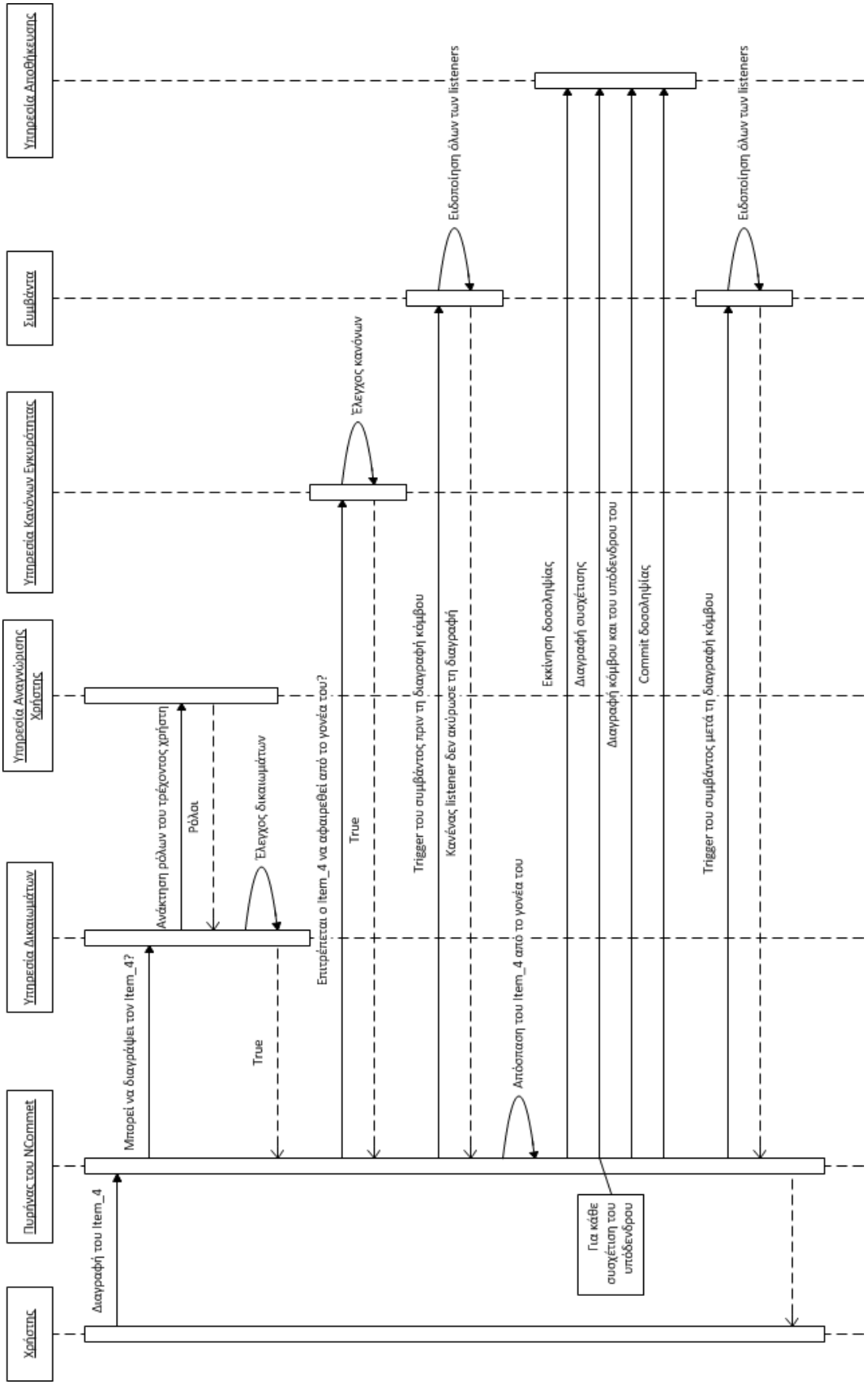


Εικόνα 4-21 Τροποποίηση κόμβων με την υπηρεσία Αποθήκευσης

4.3.3 Συνεργασία πυρήνα και υπηρεσιών

Ο πυρήνας του NCommet πρέπει να συνεργάζεται με τις υπηρεσίες, συμπεριλαμβανομένου και της Υπηρεσίας Αποθήκευσης, πριν και μετά την διεκπεραίωση οποιασδήποτε ενέργειας επί των κόμβων και της δενδρικής δομής τους.

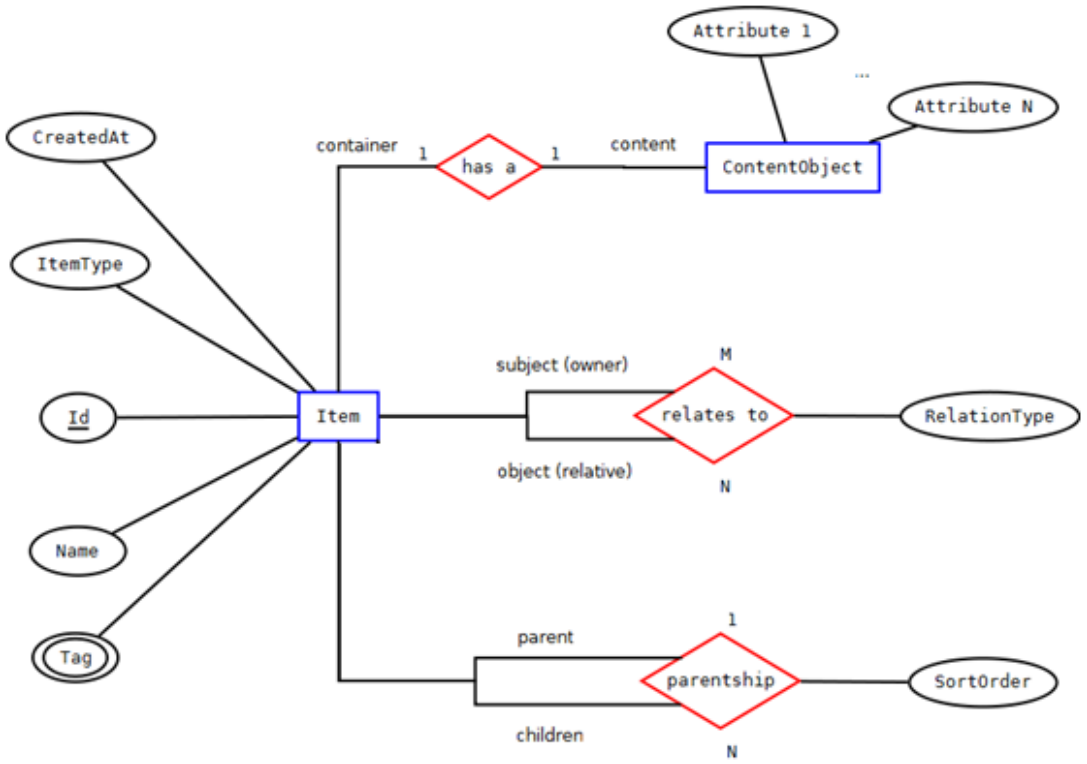
Για παράδειγμα, έστω ότι ο χρήστης ζητάει τη διαγραφή ενός κόμβου. Πριν τη διεκπεραίωση της διαγραφής καθεαυτής, πρέπει να γίνουν οι εξής ενέργειες:



Εικόνα 4-22 Ακολουθιακό διάγραμμα διαγραφής ενός κόμβου που δείχνει τη συνεργασία του πυρήνα και των υπηρεσιών

4.4 Μοντέλο Οντοτήτων Συσχετίσεων του NCommet

Παρουσιάζουμε το μοντέλο οντοτήτων συσχετίσεων (Entity-Relation diagram) των βασικών οντοτήτων του NCommet, το οποίο καλείται να διατηρήσει στη βάση δεδομένων η νέα Υπηρεσίας Αποθήκευσης του NCommet.



Εικόνα 4-23 Μοντέλο Οντοτήτων Συσχετίσεων (E-R diagram) του NCommet

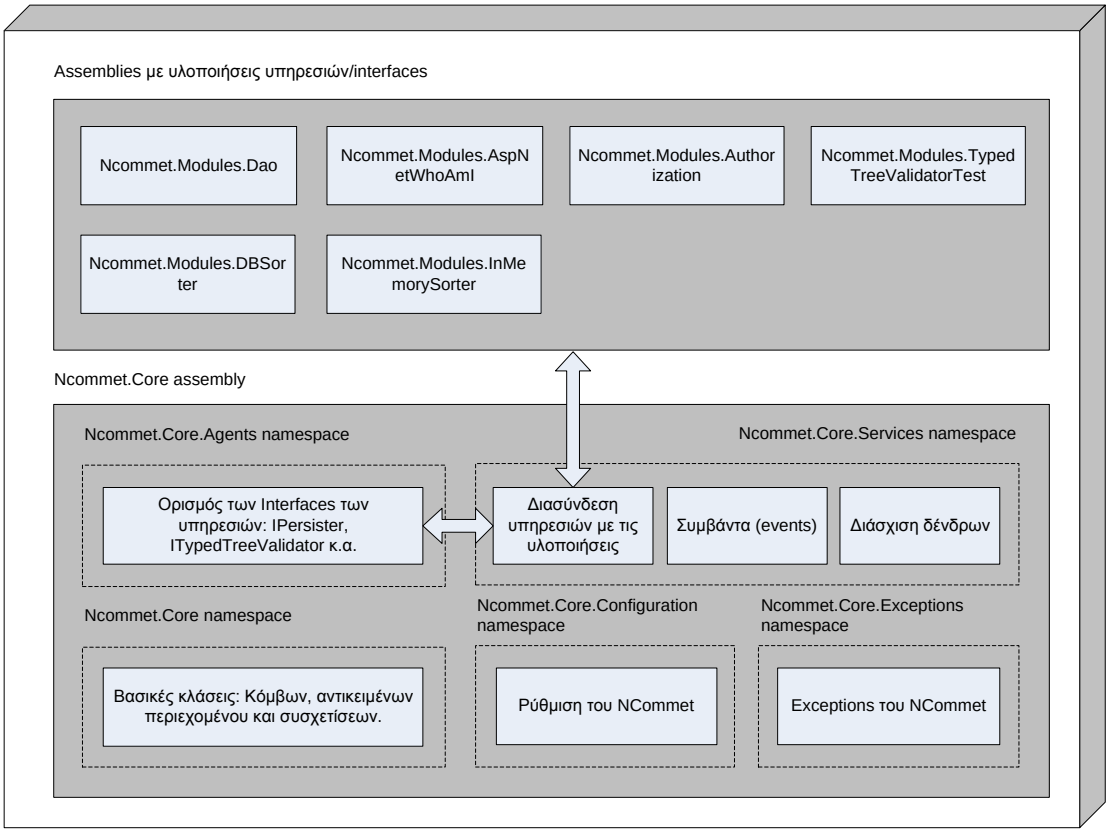
5

Σχεδίαση Συστήματος

Στην παρούσα ενότητα θα παρουσιάσουμε τη γενική σχεδίαση του NCommet, το οποίο αποτελείται από τον πυρήνα και τις έτοιμες υλοποιήσεις των υπηρεσιών του. Επειδή σκοπός της διπλωματικής είναι η υλοποίηση της νέας Υπηρεσίας Αποθήκευσης του NCommet (η οποία αναλύτεται στην ενότητα 6 «Υλοποίηση»), θα επικεντρωθούμε στην ανάλυση μόνο του πυρήνα και δεν θα αναλύσουμε τις υλοποιήσεις των υπολοίπων υπηρεσιών του NCommet.

5.1 Αρχιτεκτονική

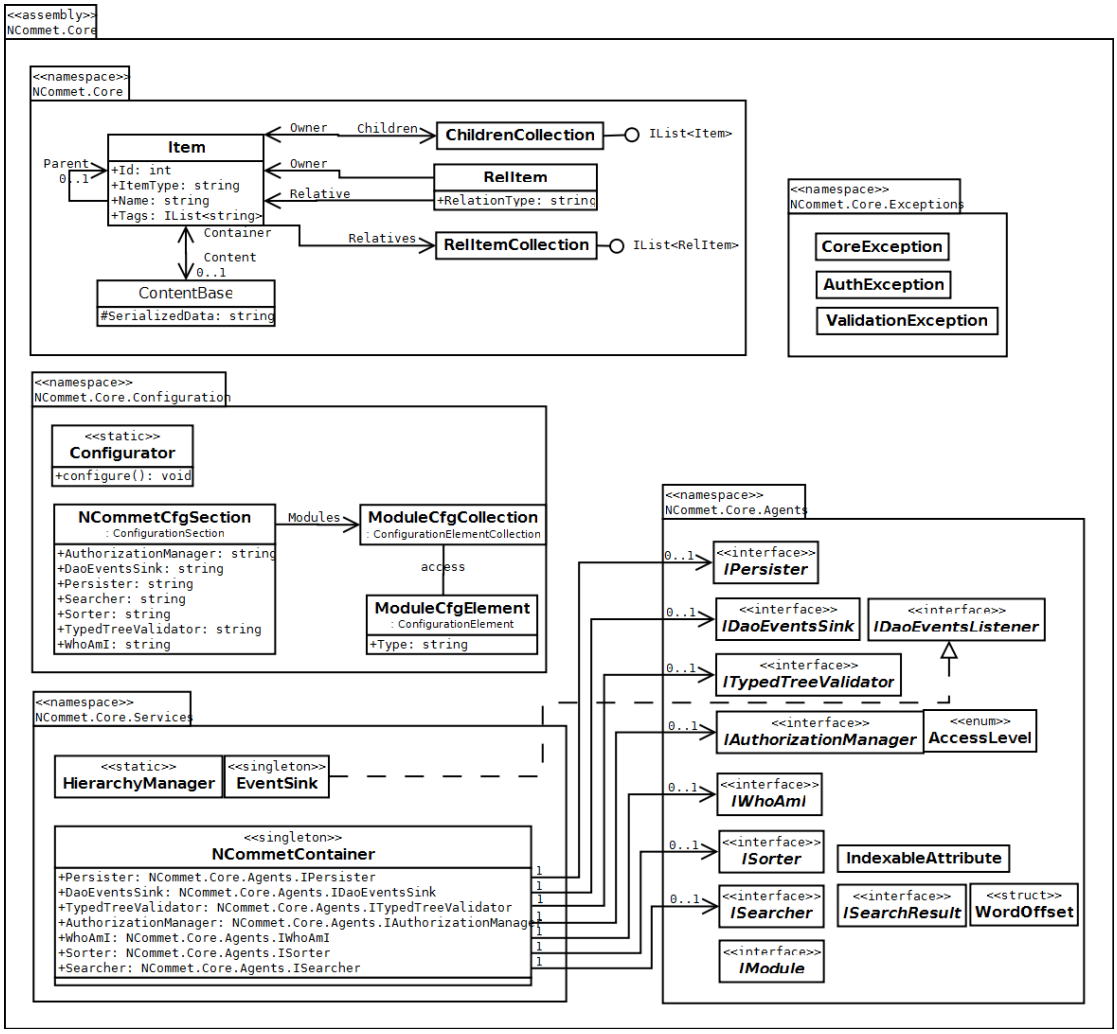
Τα συστατικά όλου του NCommet solution φαίνονται στο παρακάτω block διάγραμμα:



Εικόνα 5-1 Γενική αρχιτεκτονική του NCommet

5.1.1 NCommet.Core assembly

Στο παρακάτω διάγραμμα κλάσεων, φαίνονται οι κλάσεις (συνοπτικά) του NCommet.Core assembly, το οποίο υλοποιεί τις λειτουργικές απαιτήσεις του πυρήνα του NCommet.



Εικόνα 5-2 Συνοπτικό διάγραμμα κλάσεων του NCommet.Core assembly

Περιγραφή των συστατικών του assembly NCommet.Core:

- NCommet.Core namespace: Περιέχει τις βασικές κλάσεις του NCommet.
    - Item. Αντιπροσωπεύει έναν κόμβο.
    - ContentBase. Αντιπροσωπεύει ένα αντικείμενο περιεχομένου.
    - ChildrenCollection. Αντιπροσωπεύει τη λίστα των παιδιών ενός κόμβου.
    - RelItem. Αντιπροσωπεύει μια συσχέτιση μεταξύ ενός υποκείμενου και αντικείμενου κόμβου.
    - RelItemCollection. Αντιπροσωπεύει τις συσχετίσεις ενός κόμβου, στις οποίες συμμετέχει ως υποκείμενος κόμβος.
- NCommet.Core.Agents namespace: Το NCommet ορίζει τις υπηρεσίες του με τη μορφή διεπαφών (interfaces), στις οποίες έχει διαμοιραστεί η λειτουργικότητα του. Οι διεπαφές αυτές είναι:

- IPersister. Αντιπροσωπεύει την νέα Υπηρεσία Αποθήκευσης. Αποθηκεύει τα αντικείμενα του NCommet (κόμβους, αντικείμενα περιεχομένου και συσχετίσεις) σε ένα μέσο αποθήκευσης.
- IDaoEventsSink. Η επιπρόσθετη αυτή διεπαφή που δημιουργήσαμε, βοηθάει την νέα Υπηρεσία Αποθήκευσης να μεταδώσει τα συμβάντα (events) που έχουν σχέση με την αποθήκευση, στον EventSink. Ο EventSink υλοποιεί τη διεπαφή IDaoEventsListener ώστε να ακούει τα συμβάντα του IDaoEventsSink και να τα αναπαράγει αυτόματα.
- ITypedTreeValidator. Αντιπροσωπεύει την Υπηρεσία Κανόνων Εγκυρότητας. Υποστηρίζει τον ορισμό κανόνων εγκυρότητας επί των κόμβων και της δενδρικής δομής τους, βάσει των τύπων των κόμβων.
- IAuthorizationManager. Αντιπροσωπεύει την Υπηρεσία Δικαιωμάτων Πρόσβασης. Ελέγχει την εξουσιοδότηση που έχουν οι ρόλοι χρηστών, προστατεύοντας τους κόμβους σύμφωνα με τα δικαιώματα των ρόλων που τους έχουν οριστεί.
- IWhoAmI. Αντιπροσωπεύει την Υπηρεσία Αναγνώρισης Χρήστη. Παρέχει πληροφορίες για τους χρήστες και τους ρόλους τους.
- ISorter. Αντιπροσωπεύει την Υπηρεσία Ταξινόμησης. Ταξινομεί μια λίστα κόμβων σύμφωνα με τις τιμές μιας ιδιότητας των αντικειμένων περιεχομένου τους.
- ISearcher. Αντιπροσωπεύει την Υπηρεσία Αναζήτησης. Εκτελεί αναζήτηση κειμένου στους κόμβους.
- IModule. Αντιπροσωπεύει μία εξωτερική μονάδα της Υπηρεσίας Επεκτασιμότητας.

- NCommet.Core.Services namespace: Βοηθητικές κλάσεις.
  - NCommetContainer. Singleton κλάση που συγκεντρώνει τις υλοποιήσεις των διεπαφών των υπηρεσιών του NCommet.
  - EventSink. Singleton κλάση που υποστηρίζει τα συμβάντα (events) του NCommet.
  - HierarchyManager. Στατική κλάση που υποστηρίζει τη διάσχιση δένδρων του NCommet.
- NCommet.Core.Configuration namespace: Κλάσεις που υποστηρίζουν την αυτόματη ρύθμιση του NCommet.

- Configurator. Αντλεί και εφαρμόζει τις ρυθμίσεις του αρχείου ρυθμίσεων (app.config ή web.config) της εφαρμογής. Στο αρχείο ρυθμίσεων ορίζονται επίσης οι υλοποιήσεις των διεπαφών των υπηρεσιών του NCommet, που θα χρησιμοποιήσει η εφαρμογή. Γι' αυτό το λόγο, ο Configurator ρυθμίζει αυτόματα τον NCommetContainer.

- NCommet.Core.Exceptions namespace: Οι εξειδικευμένες εξαιρέσεις του NCommet.

5.1.2 Assemblies με έτοιμες υλοποιήσεις των υπηρεσιών

Για να χρησιμοποιήσει μια εφαρμογή τις υπηρεσίες του NCommet, **πρέπει πιο πριν να συνδέσει τις διεπαφές των υπηρεσιών που χρειάζεται, με τις υλοποιήσεις τους**. Η σύνδεση της διεπαφής μιας υπηρεσίας με την υλοποίηση της, ορίζεται στην κλάση NCommetContainer. Η σύνδεση των διεπαφών με τις υλοποιήσεις τους, μπορεί να επιτευχθεί αυτόματα χρησιμοποιώντας την κλάση Configurator.

Στο NCommet solution, προσφέρονται έτοιμες υλοποιήσεις για καθεμία από τις διεπαφές (interfaces) των υπηρεσιών, που αναφέραμε παραπάνω, εκτός της διεπαφής ISearcher, και τις οποίες μπορεί να χρησιμοποιήσει μια εφαρμογή. Οι έτοιμες υλοποιήσεις έχουν χωρισθεί σε δικά τους assemblies.

Πιο συγκεκριμένα, υπάρχουν οι εξής έτοιμες υλοποιήσεις:

Διεπαφή	**Assembly**	**Κλάση υλοποίησης**
IAuthorizationManager	NCommet.Modules.Authorization	AuthorizationManagerImpl
IPersister	NCommet.Modules.Dao	Storage
IDaoEventsSink	NCommet.Modules.Dao	DaoInterceptor
ISorter	NCommet.Modules.InMemorySorter	InMemorySorter
	NCommet.Modules.DBSorter	DBSorter
IModule	NCommet.Modules.DBSorter	DBIndexer (χρησιμοποιείται από τον DBSorter)
ITypedTreeValidator	NCommet.Modules.BRE	TypeTreeValidatorImpl
	NCommet.Modules.TypedTreeValidatorTest	BlogsTTV
IWhoAmI	NCommet.Modules.AspNetWhoAmI	AspNetWhoAmI

ISearcher		Δεν υπάρχει

Πίνακας 5-1 Έτοιμες υλοποιήσεις των υπηρεσιών του NCommet

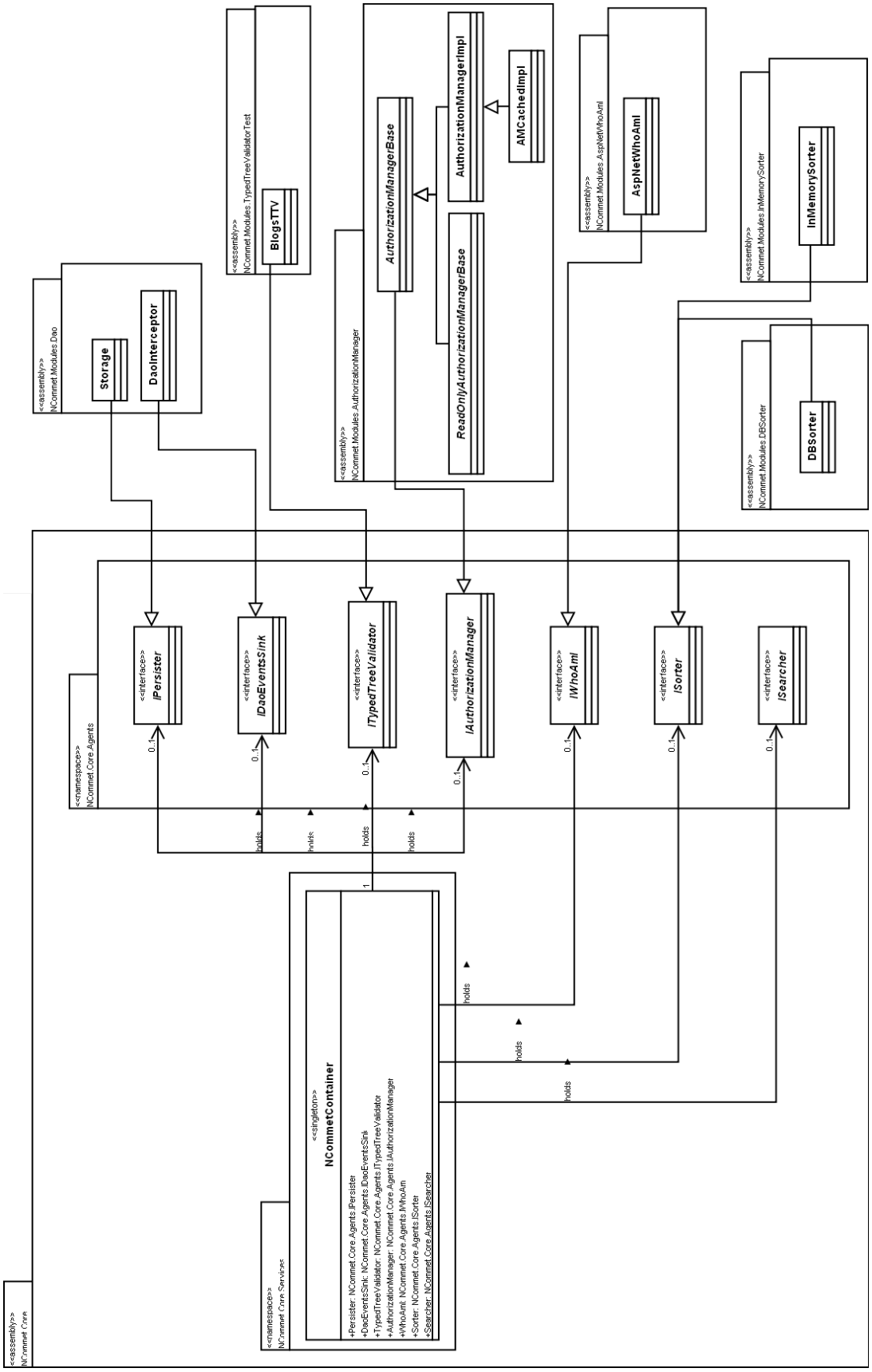
Τα assemblies και οι κλάσεις του παραπάνω πίνακα δεν περιγράφονται αναλυτικά στην παρούσα διπλωματική εργασία. Θα περιγραφεί μόνο το NCommet.Modules.Dao assembly στην ενότητα 6 «Υλοποίηση». Το assembly αυτό περιέχει την έτοιμη υλοποίηση της Υπηρεσίας Αποθήκευσης του NCommet με τη βοήθεια του NHibernate, το οποίο είναι και το κύριο αντικείμενο της διπλωματικής εργασίας.

Πρέπει να σημειώσουμε ότι μια εφαρμογή μπορεί να δημιουργήσει τις δικές της υλοποιήσεις για οποιαδήποτε διεπαφή επιθυμεί.

Η υλοποίηση των διεπαφών δεν είναι απαραίτητη για τη λειτουργία του NCommet, καθώς και χωρίς καμία υλοποίηση, μια εφαρμογή μπορεί να χρησιμοποιήσει τις βασικές κλάσεις για να δημιουργήσει ένα δένδρο στη μνήμη. Δηλαδή, το NCommet μπορεί να χρησιμοποιηθεί ως έχει, χωρίς η εφαρμογή να χρειάζεται να υλοποιήσει κάποια διεπαφή.

Όμως υπάρχει μια ιδιάζουσα εξαίρεση: αν η εφαρμογή χρησιμοποιήσει μεθόδους που χρησιμοποιούν το μέσο αποθήκευσης, και δεν έχει υλοποιηθεί η διεπαφή IPersister, τότε θα προκύψουν exceptions (π.χ. αν η εφαρμογή προσπαθήσει να αποθηκεύσει έναν κόμβο). **Κατ' αυτήν την έννοια, απ' όλες τις διεπαφές, η υλοποίηση της διεπαφής IPersister της Υπηρεσίας Αποθήκευσης, κρίνεται αναγκαία**. Εάν, για το IPersister χρησιμοποιηθεί η έτοιμη υλοποίηση NCommet.Modules.Dao.Storage, τότε θα χρειαστεί η εφαρμογή να κάνει μερικές ακόμα ρυθμίσεις που περιγράφονται στη ενότητα 6.1.3.3.

Στο παρακάτω διάγραμμα φαίνονται όλα τα assemblies με τις έτοιμες υλοποιήσεις που προσφέρει το NCommet. Βλέπουμε επίσης πως η κλάση NCommetContainer κρατάει, για κάθε διεπαφή, μια αναφορά σε μία εκ των έτοιμων υλοποιήσεων που προσφέρονται.



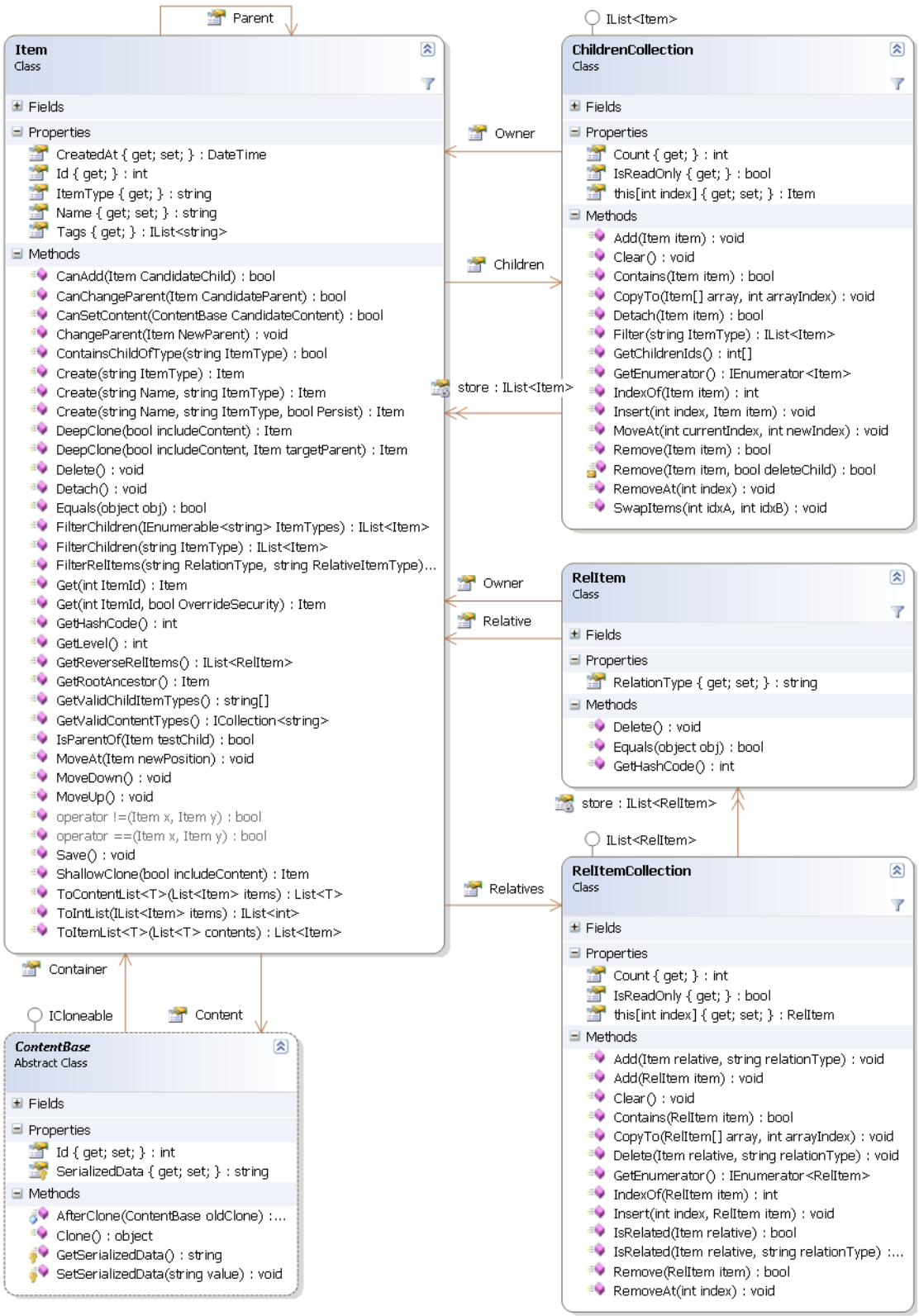
Εικόνα 5-3 Απεικόνιση των έτοιμων υλοποιήσεων των διεπαφών των υπηρεσιών του NCommet

5.2 Περιγραφή των κλάσεων του NCommet.Core assembly

Στην ενότητα αυτή περιγράφουμε, με διαγράμματα κλάσεων, τις κλάσεις του πυρήνα του NCommet. Δεν περιγράφουμε αναλυτικά τα μέλη (πεδία, ιδιότητες, μέθοδοι) των κλάσεων, καθώς αντιστοιχούν στις λειτουργικές απαιτήσεις που απαριθμήσαμε στην ενότητα 4.3 «Λειτουργικές απαιτήσεις».

5.2.1 NCommet.Core namespace

Παρακάτω φαίνεται το διάγραμμα κλάσεων για τις βασικές κλάσεις του NCommet, οι οποίες βρίσκονται στο namespace NCommet.Core.



Εικόνα 5-4 Βασικές κλάσεις του πυρήνα του NCommet

5.2.1.1 Item

Η σημαντικότερη κλάση του NCommet είναι η Item, η οποία αντιπροσωπεύει έναν κόμβο. Ένας κόμβος μπορεί να περιέχει οποιοδήποτε αντικείμενο περιεχομένου της εφαρμογής, όμως αυτό το αντικείμενο πρέπει να κληρονομεί την κλάση ContentBase.

5.2.1.2 ContentBase

Η ContentBase είναι μια αφαιρετική (abstract) κλάση. Κάθε κλάση περιεχομένου της εφαρμογής πρέπει να την κληρονομεί. Επιπλέον, κάθε κλάση περιεχομένου πρέπει να είναι XML Serializable (βλ. 3.4 «XML Σειριακοποίηση»). Οι περιορισμοί που θέσαμε για τις κλάσεις περιεχομένου, επιτρέπουν στην ιδιότητα SerializedData να σειριακοποιεί και να αποσειριακοποιεί το αντικείμενο περιεχομένου.

Η ουσιαστική χρησιμότητα της ιδιότητας SerializedData είναι ότι επιτρέπει στην Υπηρεσία Αποθήκευσης του NCommet να σειριακοποιεί οποιοδήποτε αντικείμενο περιεχομένου και να αποθηκεύει την XML έξοδο στο μέσο αποθήκευσης. Επιπλέον, αποθηκεύοντας και πληροφορίες για την κλάση περιεχομένου του αντικειμένου, μπορεί, μεταγενέστερα, να ανακτήσει το αντικείμενο περιεχομένου από το αποθηκευτικό μέσο εκ νέου.

5.2.1.3 ChildrenCollection

Η κλάση ChildrenCollection ανήκει πάντα σε ένα συγκεκριμένο κόμβο και περιέχει τους κόμβους που είναι άμεσα παιδιά του. Η θέση του κάθε παιδιού μεταξύ των αδερφιών του είναι σημαντική και ορίζεται από την εφαρμογή.

5.2.1.4 RelItem

Η κλάση RelItem αντιπροσωπεύει την συσχέτιση μεταξύ ενός υποκείμενου κόμβου (ιδιότητα Owner) και ενός αντικείμενου κόμβου (ιδιότητα Relative). Ο τύπος της συσχέτισης αντιπροσωπεύεται από την ιδιότητα RelationType.

5.2.1.5 RelItemCollection

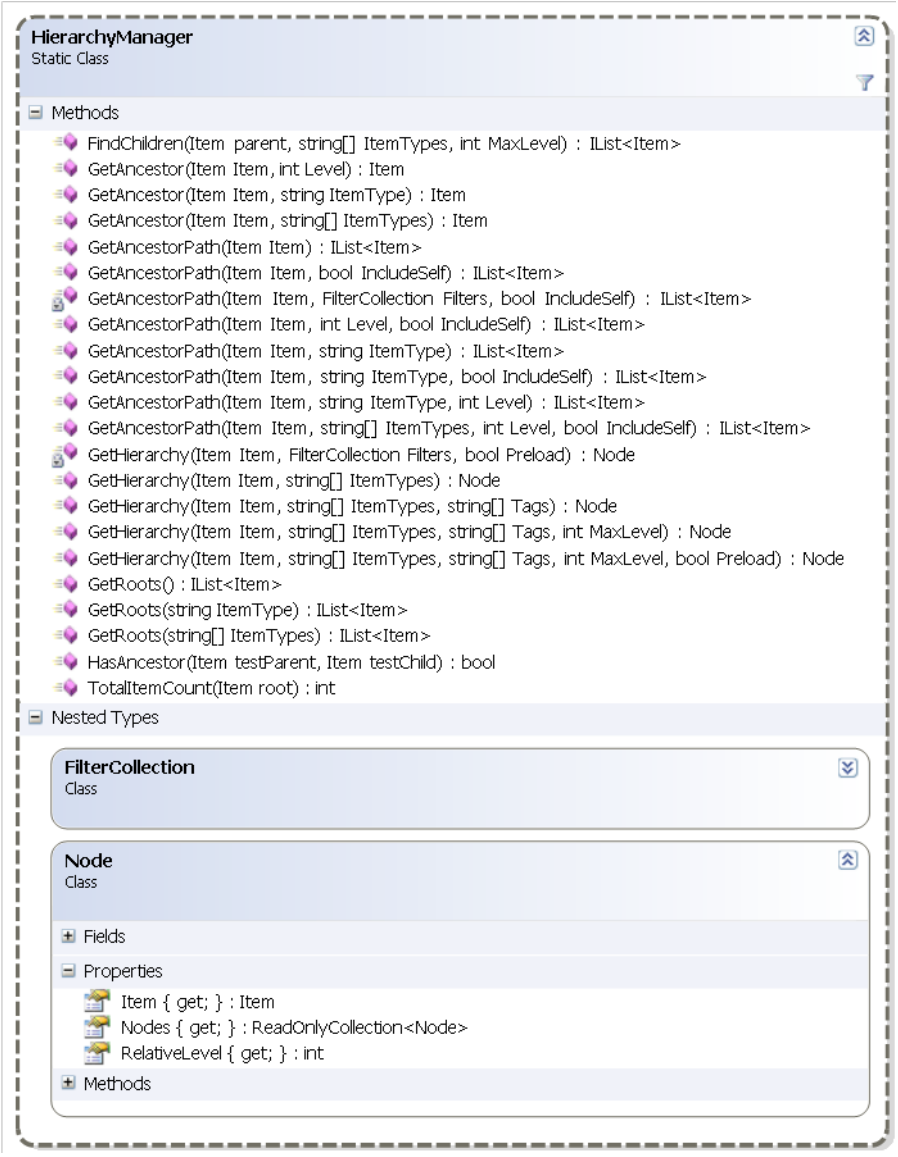
Η κλάση RelItemCollection ανήκει πάντα σε ένα συγκεκριμένο κόμβο και περιέχει τα RelItem αντικείμενα με τις συσχετίσεις του κόμβου, στις οποίες συμμετέχει ως υποκείμενος κόμβος.

5.2.2 NCommet.Core.Services namespace

Στο namespace NCommet.Core.Services υπάρχουν μερικές κλάσεις που χρησιμεύουν ως βοηθητικές κλάσεις του NCommet.

5.2.2.1 *HierarchyManager*

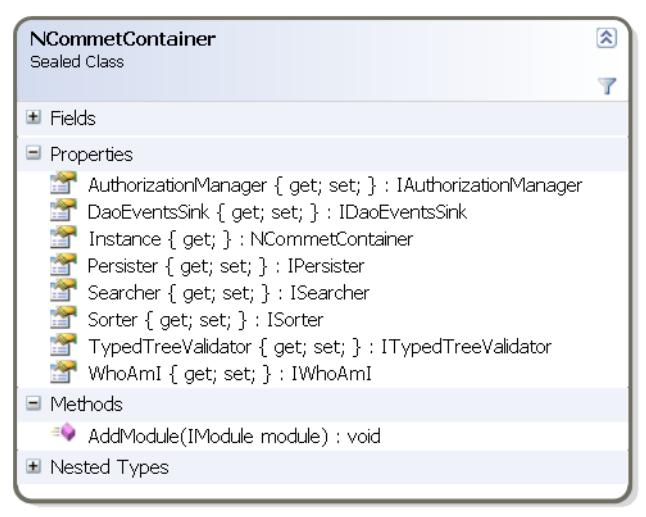
Οι μέθοδοι της στατικής κλάσης HierarchyManager υλοποιούν τις λειτουργικές απαιτήσεις που αφορούν τη διάσχιση δένδρων.



Εικόνα 5-5 Διάγραμμα κλάσης του HierarchyManager

5.2.2.2 *NCommetContainer*

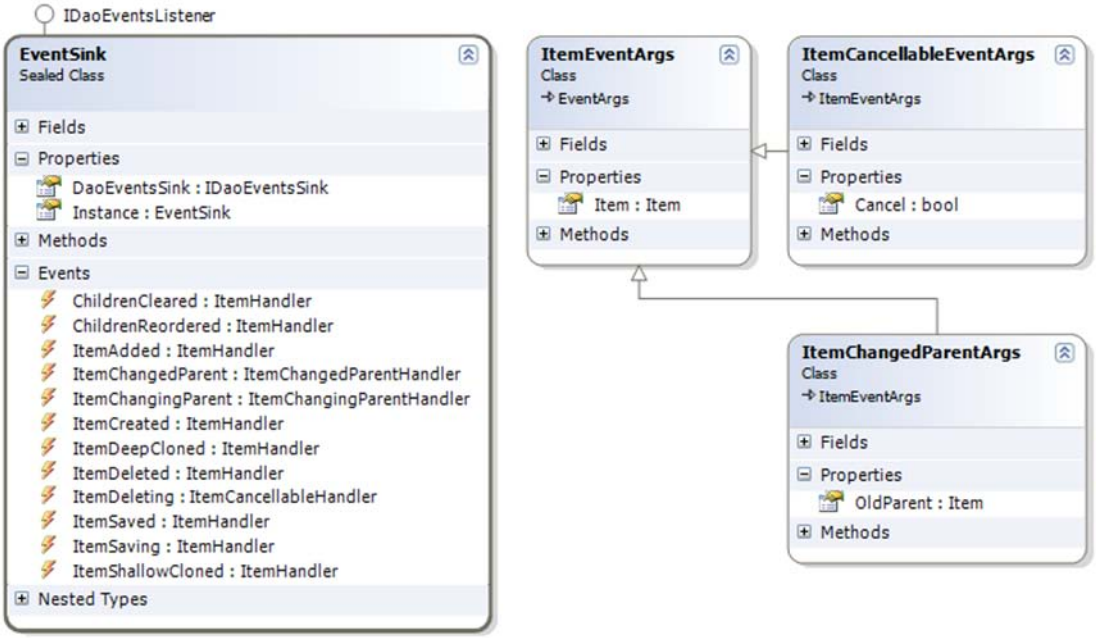
Συγκεντρώνει τις concrete υλοποιήσεις των διεπαφών. Μέσα στο NCommet, ο κώδικας απευθύνεται σε αυτήν την κλάση όταν θέλει να χρησιμοποιήσει κάποια από τις υπηρεσίες.



Εικόνα 5-6 Διάγραμμα κλάσης του NCommetContainer

5.2.2.3 EventSink

Η κλάση EventSink είναι μια singleton κλάση, την οποία χρησιμοποιεί η εφαρμογή ή τα IModules για να εγγραφούν στα διάφορα συμβάντα (events) του NCommet. Ο EventSink υλοποιεί τις λειτουργικές απαιτήσεις του NCommet που αφορούν τα συμβάντα.



Εικόνα 5-7 Διάγραμμα κλάσης του EventSink

Επεξηγήσεις:

- Ο EventSink αντιστοιχεί πάντα την ιδιότητα DaoEventsSink με την ιδιότητα DaoEventsSink του NCommetContainer. Η ιδιότητα αυτή ορίζει σε ποια υλοποίηση της διεπαφής IDaoEventsSink θα εγγραφεί ο EventSink για να αντλεί τα συμβάντα

που έχουν σχέση με την Υπηρεσία Αποθήκευσης και να τα αναπαράγει στα δικά του συμβάντα.

5.2.3 NCommet.Core.Configuration namespace

Στο namespace NCommet.Core.Configuration, υπάρχει η στατική κλάση Configurator. Η κλάση αυτή χρησιμεύει για να αντλήσει και να εφαρμόσει τις ρυθμίσεις του αρχείου ρυθμίσεων (app.config ή web.config) της εφαρμογής.

Η κλάση Configurator προσφέρει μόνο τη μέθοδο Configure() η οποία πρέπει να κληθεί στην αρχή της εφαρμογής, πριν χρησιμοποιηθεί το NCommet.

Οι ρυθμίσεις περιλαμβάνονται στο τμήμα <ncommet>…</ncommet> του αρχείου ρυθμίσεων. Εκεί ορίζονται τα εξής:

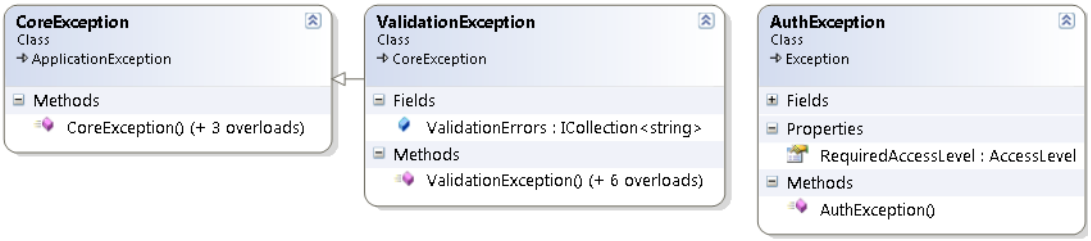
- Οι υλοποιήσεις των διεπαφών των υπηρεσιών του NCommet που θα χρησιμοποιήσει η εφαρμογή. Ο Configurator φορτώνει δυναμικά τις υλοποιήσεις στο NCommetContainer.
- Οι εξωτερικές μονάδες τις οποίες θα χρησιμοποιήσει η εφαρμογή. Ο Configurator φορτώνει αυτόματα την κάθε εξωτερική μονάδα και καλεί την μέθοδο AddModule του NCommetContainer.

Ένα παράδειγμα ενός αρχείου ρυθμίσεων φαίνεται παρακάτω, για την εφαρμογή SimpleCMSPlus:

```
<ncommet
   persister="NCommet.Modules.Dao.Storage, NCommet.Modules.Dao"
   daoEventsSink="NCommet.Modules.Dao.DaoInterceptor,
NCommet.Modules.Dao"
   sorter="…" typedTreeValidator="…" authorizationManager="…"
whoAmI="…">
    <modules>
          <add type="AbstractCMS.Modules.InterPortalMover,
AbstractCMS"/>
          …
          <add type="SimpleCMSPlus.BusinessModel.Auditor,
SimpleCMSPlus.BusinessModel"/>
    </modules>
</ncommet>
```

5.2.4 NCommet.Core.Exceptions namespace

Στο namespace NCommet.Core.Exceptions, υπάρχουν 3 κλάσεις που αντιπροσωπεύουν ειδικές περιπτώσεις εξαιρέσεων.



Εικόνα 5-8 Διάγραμμα κλάσεων των εξαιρέσεων του NCommet

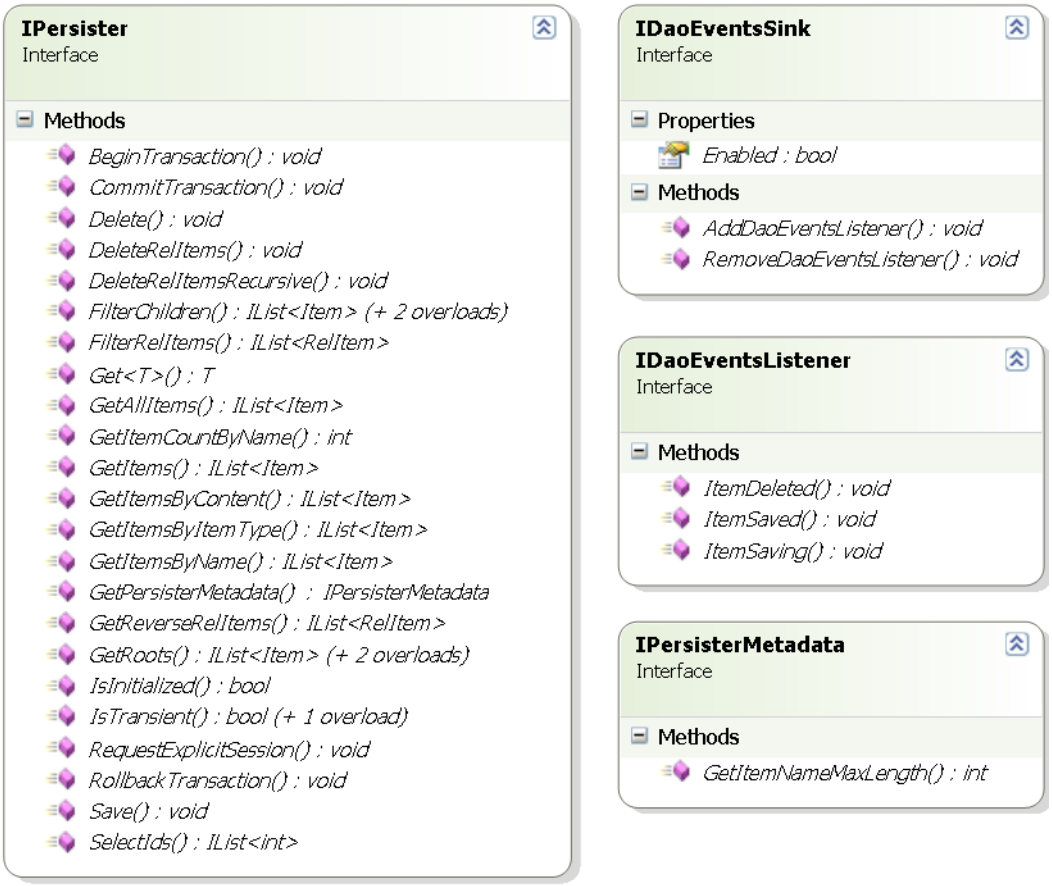
Η CoreException πετάγεται από τις βασικές κλάσεις του πυρήνα του NCommet. Η ValidationException πετάγεται από την υλοποίηση της Υπηρεσία Κανόνων Εγκυρότητας του NCommet. Η AuthException πετάγεται από την υλοποίηση της Υπηρεσίας Δικαιωμάτων Πρόσβασης του NCommet.

5.2.5 NCommet.Core.Agents namespace

Στο namespace NCommet.Core.Agents υπάρχουν οι διεπαφές (interfaces) των υπηρεσιών του NCommet. Θα παρουσιάσουμε μόνο τις διεπαφές της νέας Υπηρεσίας Αποθήκευσης της οποίας η υλοποίηση θα παρουσιασθεί στην ενότητα 6 «Υλοποίηση».

5.2.5.1 IPersister και IDaoEventsSink

Οι διεπαφές IPersister και IDaoEventsSink του NCommet φαίνονται στο εξής διάγραμμα κλάσεων, μαζί με τις βοηθητικές κλάσεις του.



Εικόνα 5-9 Διάγραμμα κλάσεων των διεπαφών IPersister και IDaoEventsSink

Οι μέθοδοι που ορίζει ο IPersister υλοποιούν τις λειτουργικές απαιτήσεις του NCommet που αφορούν την νέα Υπηρεσία Αποθήκευσης. Ο IPersister έχει φτιαχτεί έχοντας υπ' όψιν ότι θα υλοποιηθεί με κάποιο ORM εργαλείο, συγκεκριμένα το NHibernate.

Ο IPersister ορίζει μερικές παραπάνω μεθόδους απ' ότι απαιτούν οι λειτουργικές απαιτήσεις που περιγράψαμε στην ενότητα 4.3 «Λειτουργικές απαιτήσεις». Πρόκειται για βοηθητικές μεθόδους και αυτές είναι οι εξής:

- GetPersisterMetadata(): Γνωστοποιεί στην εφαρμογή το μέγιστο μήκος ονόματος ενός κόμβου που επιτρέπεται να αποθηκευτεί.
- IsInitialized(object proxy): Η μέθοδος αυτή χρησιμεύει για υλοποιήσεις της διεπαφής IPersister με ORM εργαλεία, που υποστηρίζουν lazy loading (βλ. 3.3 «NHibernate»). Θα πάρουμε ως παράδειγμα το NHibernate, με το οποίο φτιάξαμε την έτοιμη υλοποίηση της διεπαφής IPersister της Υπηρεσίας Αποθήκευσης του NCommet που υπάρχει στο assembly NCommet.Modules.Dao και περιγράφεται στην ενότητα 6 «Υλοποίηση». Αν το lazy loading έχει ενεργοποιηθεί για κάποια αντικείμενα, τότε όταν καλούμε το NHibernate να διαβάσει το αντικείμενο, δεν διαβάζει μαζί και τα σχετιζόμενα αντικείμενα. Για παράδειγμα, αν καλέσουμε το NHibernate να φορτώσει έναν κόμβο, δεν διαβάζει μαζί και τις συσχετίσεις του. Στη θέση των συσχετίσεων,

τοποθετεί αντικείμενα proxy. Τα proxy αντικείμενα δημιουργούνται δυναμικά από το NHibernate και έχουν την ίδια διεπαφή με τις συσχετίσεις (κλάση RelItem), αλλά είναι διαφορετικής κλάσης και δεν έχουν φορτωθεί τα δεδομένα τους. Όταν η εφαρμογή προσπαθήσει να προσπελάσει ένα proxy αντικείμενο, τότε το NHibernate αντλεί δυναμικά τα δεδομένα, μετατρέποντας το proxy αντικείμενο σε στιγμιότυπο της κλάσης RelItem που περιμένει η εφαρμογή. Όμως, αν η εφαρμογή προσπαθήσει να προσπελάσει ένα proxy αντικείμενο το οποίο έχει δημιουργηθεί σε διαφορετική συνεδρία (ISession) από την οποία βρισκόμαστε, θα πεταχτεί μια εξαίρεση του NHibernate. Η μέθοδος IsInitialized() γυρνάει true εάν το proxy αντικείμενο έχει αρχικοποιηθεί από την ίδια συνεδρία στην οποία βρισκόμαστε. Αν γυρίσει true, τότε η εφαρμογή μπορεί να προσπελάσει το proxy και το NHibernate θα αντλήσει δυναμικά τα δεδομένα του για να το μετατρέψει σε κανονικό στιγμιότυπο της κλάσης που περιμένει η εφαρμογή.

- RequestExplicitSession(): Με τη μέθοδο αυτή, η εφαρμογή αιτείται μιας καινούργιας συνεδρίας από το ORM εργαλείο. Για παράδειγμα, στο NHibernate δημιουργείται ένα καινούργιο ISession.

Η διεπαφή IDaoEventsSink έχει φτιαχτεί για να αντλεί τα συμβάντα (events) που έχουν σχέση με την αποθήκευση και τη διαγραφή κόμβων και να ενημερώνει τους IDaoEventsListeners που έχουν εγγραφεί σε αυτόν.

Όπως έχει ήδη διαφανεί, η Υπηρεσία Αποθήκευσης προσπαθεί να αποκρύψει τα συμβάντα που συμβαίνουν στην πηγή δεδομένων. Εφόσον ήδη προσφέρεται ο EventSink, προσπαθεί να αναπαράγει τα συμβάντα της πηγής δεδομένων στα αντίστοιχα συμβάντα του EventSink. Τελικά, η εφαρμογή δεν γνωρίζει τίποτα για την προέλευση των συμβάντων και εγγράφεται μόνο στον EventSink για όλα τα συμβάντα (και του πυρήνα του NCommet και της Υπηρεσίας Αποθήκευσης).

Ο τρόπος με τον οποίο επιτυγχάνεται αυτό είναι ο εξής: Η κλάση EventSink υλοποιεί τη διεπαφή IDaoEventsListener και εγγράφεται αυτόματα στα συμβάντα της υλοποίησης της υπηρεσίας IDaoEventsSink.

5.3 Βάση Δεδομένων

Όπως αναφέραμε στην ενότητα 5.1 «Αρχιτεκτονική», μια εφαρμογή μπορεί να χρησιμοποιήσει οσεσδήποτε από τις υπηρεσίες του NCommet χρειάζεται. Για κάθε υπηρεσία μπορεί να ορίσει (στο αρχείο ρυθμίσεων) οποιαδήποτε υλοποίηση της αντίστοιχης διεπαφής (interface) επιθυμεί. Μπορεί μάλιστα να χρησιμοποιήσει μία εκ των έτοιμων υλοποιήσεων που προσφέρει το NCommet και οι οποίες απαριθμούνται στην ενότητα 5.1.2 «Assemblies με

έτοιμες υλοποιήσεις των υπηρεσιών». Γι' αυτό το λόγο, το σχήμα της βάσης δεδομένων που χρησιμοποιεί η εφαρμογή διαφέρει ανάλογα με τις υλοποιήσεις των υπηρεσιών που έχει ορίσει.

Παρακάτω περιγράφεται το σχήμα της βάσης δεδομένων που απαιτεί η έτοιμη υλοποίηση NCommet.Modules.Dao.Storage. Η κλάση Storage αποτελεί την υλοποίηση της Υπηρεσίας Αποθήκευσης που αναπτύξαμε στην παρούσα διπλωματική εργασία και περιγράφεται στην ενότητα 6 «Υλοποίηση».

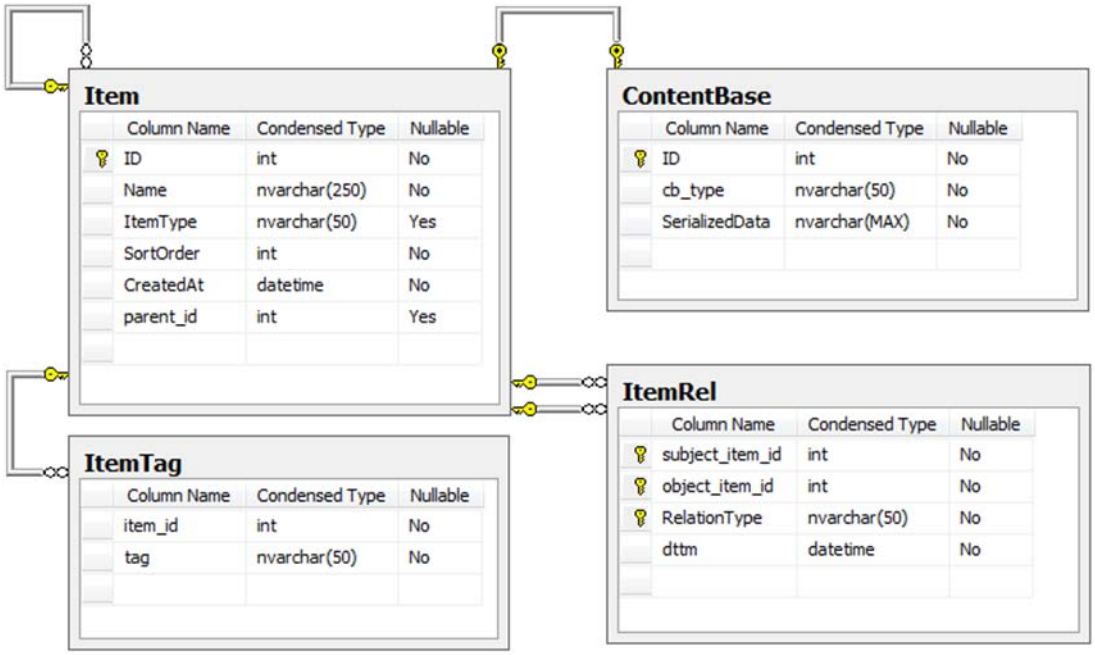
Σημείωση: Δεν δίνουμε, στην παρούσα ενότητα, λεπτομέρειες για το πώς χρησιμοποιούνται οι πίνακες.

Πεδίο	**Τύπος**	**Περιγραφή**
Πίνακας «Item»		
ID	int	Το id του κόμβου.
Name	nvarchar(250)	Το όνομα του κόμβου.
ItemType	nvarchar(50)	Ο τύπος του κόμβου.
SortOrder	int	Η θέση του κόμβου μεταξύ των αδερφιών του (με βάση το 0).
CreatedAt	datetime	Ημερομηνία και ώρα δημιουργίας του κόμβου.
Parent_id	int	Το id του κόμβου γονέα (ή NULL αν δεν έχει γονέα).
Πίνακας «ContentBase»		
ID	int	Το id του κόμβου που περιέχει το αντικείμενο περιεχομένου.
cb_type	nvarchar(50)	Εδώ μπαίνει το πλήρες όνομα της υποκλάσης του ContentBase, της οποίας στιγμιότυπο είναι το συγκεκριμένο αντικείμενο περιεχομένου.
SerializedData	nvarchar(MAX)	Έδω μπαίνει το string της XML σειριακοποίησης του αντικειμένου περιεχομένου.
Πίνακας «ItemRel»		
subject_item_id	int	Το id του υποκείμενου κόμβου της συσχέτισης.
object_item_id	int	Το id του αντικείμενου κόμβου της συσχέτισης.

RelationType	nvarchar(50)	Ο τύπος της συσχέτισης.
dttm	datetime	Timestamp εγγραφής του πίνακα.
Πίνακας «ItemTag»		
item_id	int	Το id του κόμβου που περιέχει τη συγκεκριμένη λεζάντα.
tag	nvarchar(50)	Η λεζάντα (tag).

Πίνακας 5-2 Απαραίτητοι πίνακες της βάσης δεδομένων για την έτοιμη υλοποίηση NCommet.Modules.Dao.Storage.

Το σχεσιακό διάγραμμα των παραπάνω πινάκων είναι το εξής.



Εικόνα 5-10 Σχεσιακό διάγραμμα των απαιτούμενων πινάκων της έτοιμης υλοποίησης NCommmet.Modules.Dao.Storage.

Στο assembly NCommet.Modules.Dao περιλαμβάνεται ένα SQL script με το οποίο μπορούν να φτιαχτούν οι απαραίτητοι πίνακες.

6

Υλοποίηση

Στην ενότητα αυτή συζητάμε τα εξής:

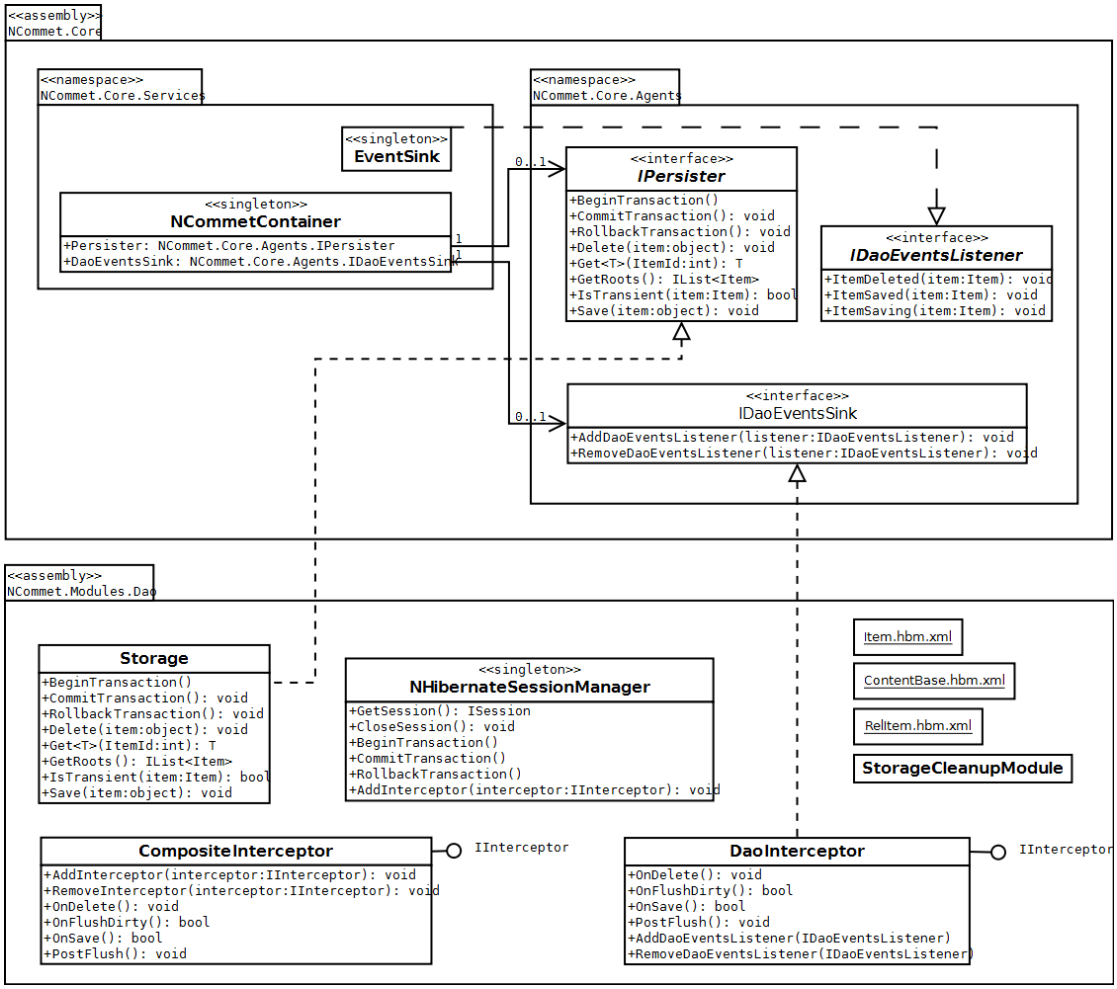
- Την έτοιμη υλοποίηση της νέας Υπηρεσίας Αποθήκευσης του NCommet με το NHibernate, που ήταν ο κύριος σκοπός της παρούσας διπλωματικής εργασίας. Αναλύουμε τις κλάσεις που περιέχει το assembly NCommet.Modules.Dao καθώς και τις ρυθμίσεις απεικόνισης των βασικών κλάσεων του NCommet.
- Τις πλατφόρμες και τα προγραμματιστικά εργαλεία που χρησιμοποιούνται από ολόκληρο το SimpleCMSPlus.

6.1 Υπηρεσία Αποθήκευσης

6.1.1 NCommet.Modules.Dao assembly

Στο assembly NCommet.Modules.Dao βρίσκεται η κλάση Storage, η οποία αποτελεί μια έτοιμη υλοποίηση της διεπαφής IPersister της νέας Υπηρεσίας Αποθήκευσης του NCommet. Χρησιμοποιεί το NHibernate και μια βάση δεδομένων. Για μια ανασκόπηση του NHibernate και των ρυθμίσεων απεικόνισης, βλ. 3 « Θεωρητικό υπόβαθρο».

Στο επόμενο διάγραμμα κλάσεων φαίνονται οι κλάσεις και τα συστατικά του assembly, καθώς και οι σχέσεις τους με μερικές κλάσεις του NCommet.Core assembly.



Εικόνα 6-1 Διάγραμμα κλάσεων του NCommet.Modules.Dao assembly

Περιγραφή των συστατικών του assembly NCommet.Modules.Dao:

- **Κλάση Storage.** Αποτελεί την έτοιμη υλοποίηση της διεπαφής IPersister της Υπηρεσίας Αποθήκευσης του NCommet. Εσωτερικά, χρησιμοποιεί την κλάση NHibernateSessionManager.
- Τα **.hbm.xml αρχεία** περιέχουν τις ρυθμίσεις απεικόνισης για τις βασικές κλάσεις του NCommet (κόμβοι, αντικείμενα περιεχομένου και συσχετίσεις), τις οποίες χρειάζεται το NHibernate ώστε να μπορεί να αποθηκεύει τα στιγμιότυπα των κλάσεων στη βάση δεδομένων και να τα ανακτά εκ νέου.
    - Item.hbm.xml
    - ContentBase.hbm.xml
    - RelItem.hbm.xml
- **Κλάση NHibernateSessionManager.** Είναι μια singleton κλάση της οποίας κύριος σκοπός είναι να διαχειρίζεται κάθε συνεδρία (session) και συναλλαγή (transaction)

του NHibernate. Επίσης, κατά την πρώτη αναφορά στην κλάση, ρυθμίζεται το NHibernate.

- **Create.sql**. Είναι ένα SQL script που δημιουργεί όλους τους απαραίτητους πίνακες στη βάση δεδομένων (βλ. 5.3 «Βάση Δεδομένων»).
- **Κλάση DaoInterceptor.** Αποτελεί μια έτοιμη υλοποίηση της διεπαφής IDaoEventsSink του NCommet. Επιπλέον, υλοποιεί τη διεπαφή IInterceptor του NHibernate. Με τον IInterceptor, ο DaoInterceptor εγγράφεται στα συμβάντα (events) του NHibernate, όπως η αποθήκευση και η διαγραφή αντικειμένων. Έπειτα περνάει αυτά τα συμβάντα στον EventSink ο οποίος έχει εγγραφεί στον DaoInterceptor (βλ. 5.2.5.1 «IPersister και IDaoEventsSink»). Έτσι ο EventSink μπορεί να αντιστοιχήσει τα συμβάντα του NHibernate στα δικά του. Τελικώς, η εφαρμογή απλώς εγγράφεται στον EventSink για όλα τα συμβάντα που την ενδιαφέρουν (είτε πηγάζουν από το NCommet είτε πηγάζουν από την Υπηρεσία Αποθήκευσης).
- **Κλάση CompositeInterceptor**. Επιτρέπει τη χρησιμοποίηση περισσοτέρων του ενός IInterceptor για το NHibernate, γιατί το NHibernate υποστηρίζει μόνο έναν ανά SessionFactory του NHibernate. Ο NHibernateSessionManager χρησιμοποιεί τον CompositeInterceptor ως τον μοναδικό IInterceptor του SessionFactory του. Στον CompositeInterceptor, προστίθεται αυτόματα ο DaoInterceptor.
- **Κλάση StorageCleanupModule**. Ένα IHttpModule που βοηθάει την κλάση NHibernateSessionManager να διεκπεραιώσει κάποιες λειτουργίες της.

Για να χρησιμοποιήσει μια εφαρμογή το assembly NCommet.Modules.Dao πρέπει να ικανοποιήσει μερικές απαιτήσεις, όπως τη ρύθμιση των υλοποιήσεων των διεπαφών του NCommet και τη δημιουργία απαραίτητων .hbm.xml αρχείων για τις κλάσεις περιεχομένου. Τις απαιτήσεις αυτές θα απαριθμήσουμε στην ενότητα 6.1.3.3.

6.1.2 Τα αρχεία ρυθμίσεων απεικόνισης

Οι ρυθμίσεις απεικόνισης περιγράφονται μέσω των .hbm.xml αρχείων και αντικατοπτρίζουν τη μορφή των πινάκων της βάσης δεδομένων που περιγράψαμε στην ενότητα 5.3 «Βάση Δεδομένων». Μέσω αυτών των αρχείων, το NHibernate ενημερώνεται για το πώς πρέπει να αποθηκεύει τα αντικείμενα του NCommet (κόμβους και συσχετίσεις) και της εφαρμογής (αντικείμενα περιεχομένου) στη βάση δεδομένων. Για μια ανασκόπηση του NHibernate και των αρχείων ρυθμίσεων απεικόνισης, βλ. 3.3.4 «Ρυθμίσεις απεικόνισης».

6.1.2.1 Τα κρυμμένα πεδία των βασικών κλάσεων

Στον αντικειμενοστραφή προγραμματισμό με C#, υπάρχει μία συνήθης τακτική όσον αφορά τα δεδομένα που χρειάζεται μία κλάση: Για κάθε δεδομένο, ορίζεται ένα protected ή private[2] πεδίο (field) το οποίο περιέχει το δεδομένο. Με αυτόν τον τρόπο, το πεδίο δεν είναι άμεσα προσβάσιμο από το εξωτερικό της κλάσης. Αν η κλάση χρειάζεται να παρέχει εξωτερική πρόσβαση στο δεδομένο, τότε ορίζεται και μια συνονόματη public ιδιότητα (property) η οποία διαχειρίζεται την τιμή του πεδίου.

Για παράδειγμα, στην κλάση Item υπάρχει μία public ιδιότητα Name, μέσω της οποίας διαχειριζόμαστε το όνομα ενός κόμβου. Η ιδιότητα αυτή στην ουσία μεταβάλλει την τιμή του protected πεδίου name της κλάσης Item.

Item
#name : string +Name : string

Στα αρχεία ρυθμίσεων απεικόνισης που ακολουθούν, συνήθως προτιμούμε να ρυθμίζουμε το NHibernate ώστε να αποθηκεύει στη βάση δεδομένων τα πεδία μιας κλάσης παρά τις ιδιότητες της. Οι λόγοι είναι οι εξής:

- Τα πεδία περιέχουν τα δεδομένα που θέλουμε να αποθηκεύσουμε, ενώ οι αντίστοιχες ιδιότητες ίσως περιέχουν επιπλέον επιχειρηματική λογική (στους getters και setters τους) την οποία δεν θέλουμε να εκτελεί το NHibernate κάθε φορά που διαβάζει ή θέτει τα δεδομένα.
- Μερικές φορές ίσως να μην υπάρχουν ιδιότητες οι οποίες παρέχουν εξωτερική πρόσβαση στα κρυμμένα πεδία της κλάσης. Ή μπορεί η ιδιότητα να παραλείπει τον έναν από τους getters/setters του.

Ίσως να φαίνεται ότι παραβιάζουμε το θεωρητικό σκοπό των access modifiers της C#. Δηλαδή τα πεδία έχουν οριστεί ως protected ή private ακριβώς επειδή δεν πρέπει μια εξωτερική κλάση να έχει άμεση πρόσβαση σε αυτά αλλά πρέπει να χρησιμοποιεί τις αντίστοιχες public ιδιότητες. Όμως το NHibernate αποτελεί μια ειδική περίπτωση ενδιάμεσου εργαλείου, το οποίο θα θέλαμε να έχει πρόσβαση σε αυτά τα πεδία ώστε να διαβάζει και να θέτει τα δεδομένα κατευθείαν χωρίς να ενεργοποιεί τη λογική που ίσως περιέχουν οι αντίστοιχες ιδιότητες. Το NHibernate μπορεί να προσπελαύνει τα κρυμμένα πεδία των κλάσεων χρησιμοποιώντας το Reflection του .NET Framework.

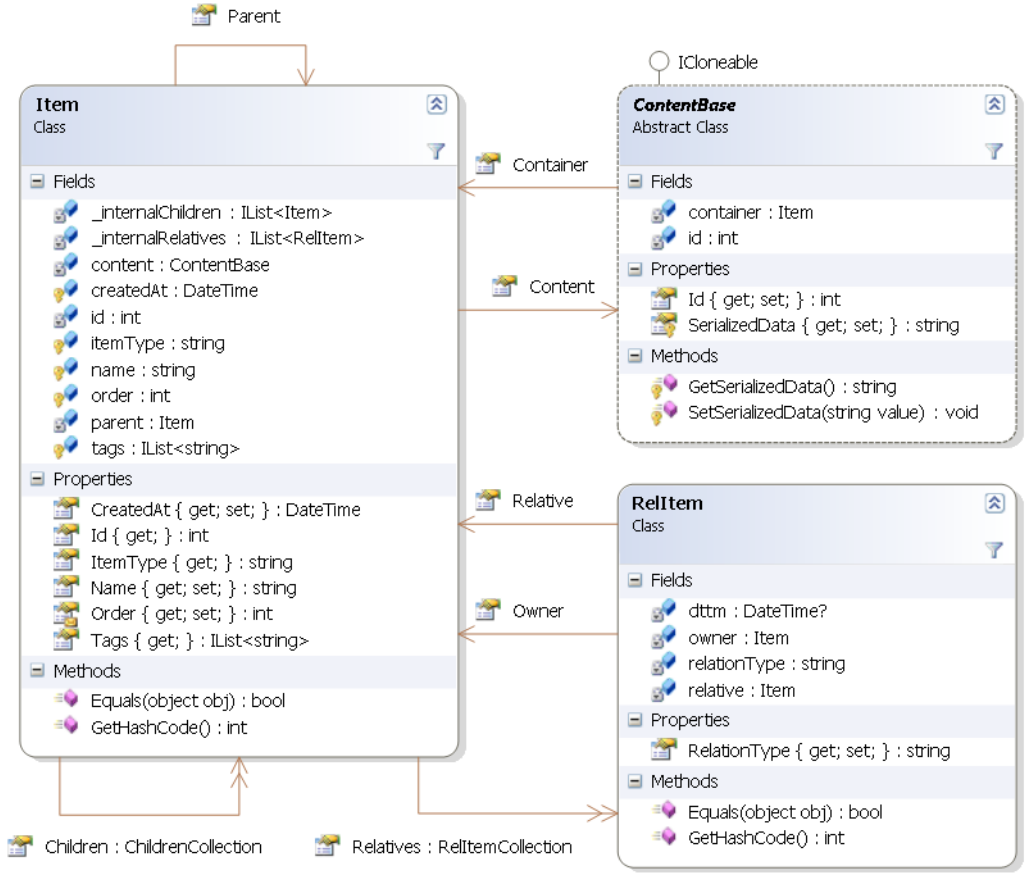
Υπάρχει βέβαια ένα μικρό μειονέκτημα με την παραπάνω τακτική που επιλέξαμε: Στα HQL ερωτήματα που αναπτύσσουμε πρέπει να γράφουμε τα ονόματα των πεδίων και όχι των ιδιοτήτων[3]. Επειδή τα ονόματα των πεδίων δεν είναι τόσο ευανάγνωστα όσο των ιδιοτήτων,

[2] Το πεδίο είναι protected ή private ανάλογα με το αν θέλουμε ή όχι οι υποκλάσεις να έχουν πρόσβαση στο πεδίο.

[3] Το NHibernate προσφέρει λύση και για αυτό το πρόβλημα. Μπορούν να χρησιμοποιηθούν τα "naming strategies" τα οποία επιτρέπουν την χρήση των ονομάτων των ιδιοτήτων στα

τα HQL ερωτήματα που θα γράψουμε γίνονται περισσότερο δυσανάγνωστα. Όμως αυτό δεν αποτελεί ιδιαίτερο πρόβλημα καθώς τα HQL ερωτήματα θα τα αναπτύξουμε μέσα στον ίδιο τον κώδικα.

Στην ενότητα 5.2.1 «NCommet.Core namespace» παρουσιάσαμε το διάγραμμα των βασικών κλάσεων του NCommet μόνο με τα public μέλη τους (ιδιότητες και μεθόδους). Επειδή στις ρυθμίσεις απεικόνισης θα χρησιμοποιήσουμε τα κρυμμένα πεδία τους, παρουσιάζουμε εκ νέου το διάγραμμα των βασικών κλάσεων, αυτή τη φορά περιλαμβάνοντας τα private ή protected πεδία τα οποία θα ρυθμίσουμε να προσπελαύνει το NHibernate. Δεν δείχνουμε τις μεθόδους των κλάσεων, παρά μόνο τις μεθόδους Equals() και GetHashCode(), τη σημασία των οποίων θα επεξηγήσουμε παρακάτω.



Εικόνα 6-2 Διάγραμμα των βασικών κλάσεων του NCommet μαζί με τα κρυμμένα πεδία τους

HQL ερωτήματα. Έπειτα το NHibernate αυτόματα μετατρέπει το όνομα της ιδιότητας στο όνομα του αντίστοιχου πεδίου, χρησιμοποιώντας τη naming strategy που έχουμε ορίσει (π.χ. camelcase, camelcase-underscore κ.α.).

6.1.2.2 *Ρυθμίσεις απεικόνισης της κλάσης Item*

Παρουσιάζουμε το αρχείο Item.hbm.xml που περιέχει τις ρυθμίσεις απεικόνισης της κλάσης Item στον πίνακα Item της βάσης δεδομένων. Ύστερα επεξηγούμε τις αξιοσημείωτες ρυθμίσεις ξεχωριστά.

```
<?xml version="1.0" encoding="utf-8" ?>
<hibernate-mapping xmlns="urn:nhibernate-mapping-2.2">
<!-- The instances of the Item class will be stored in the
"Item" table -->
<class
  name="NCommet.Core.Item, NCommet.Core"
  table="Item" lazy="true">
  <!-- Identity -->
  <id
    name="id" access="field"
    column="ID" type="Int32" unsaved-value="0">
    <generator class="identity" />
  </id>
  <!-- Fields -->
  <property
    name="name" access="field"
    column="Name" type="String" length="250" not-null="true" />
  <property
    name="itemType" access="field"
    column="ItemType" type="String" length="50" not-null="false"
  />
  <property
    name="order" access="field"
    column="SortOrder" type="Int32" not-null="true" />
  <property
    name="createdAt" access="field"
    column="CreatedAt" type="DateTime" not-null="true" />
  <!-- Field "parent": Many-to-One association between children
Items and their parent Item. -->
  <many-to-one
    name="parent" access="field"
    column="parent_id" not-null="false"
    class="NCommet.Core.Item, NCommet.Core" cascade="none"
    unique="false" />
```

```
  <!-- Field "_internalChildren": One-to-Many association
between the parent Item and its children Items. -->
  <list
    name="_internalChildren" access="field"
    table="Item" cascade="all" inverse="true"
    generic="true" batch-size="20">
      <key column="parent_id" />
      <index column="SortOrder" type="Int32" />
      <one-to-many class="NCommet.Core.Item, NCommet.Core" />
  </list>
  <!-- Field "tags": The strings will be stored in the "ItemTag"
table -->
  <bag
    name="tags" access="field"
    table="ItemTag">
      <key column="item_id"/>
      <element column="tag" type="String" length="50" not-
null="true" />
  </bag>
  <!-- Field "_internalRelatives": One-to-Many association
between the subject Item and its relatives. -->
  <bag
    name="_internalRelatives" access="field"
    table="ItemRel" cascade="all-delete-orphan" inverse="true">
      <key column="subject_item_id"/>
      <one-to-many class="NCommet.Core.RelItem, NCommet.Core" />
  </bag>
  <!-- Field "content": One-to-One association between an Item
and its ContentBase object. -->
  <one-to-one
    name="content" access="field"
    class="NCommet.Core.ContentBase, NCommet.Core" cascade="all"
  />
</class>
</hibernate-mapping>
```

Ρύθμιση	**Επεξήγηση**
• <class lazy=”true”>	Σημαίνει ότι κατά την άντληση ενός στιγμιότυπου από τη βάση

	δεδομένων, χρησιμοποιείται lazy-loading, δηλαδή δεν διαβάζονται άλλα αντικείμενα με τα οποία σχετίζεται. Πιο συγκεκριμένα, δεν διαβάζονται τα παιδιά και οι συσχετιζόμενοι κόμβοι. Τα αντικείμενα αυτά θα αντληθούν δυναμικά από το NHibernate όταν το σχετικό πεδίο προσπελαστεί στον χρόνο εκτέλεσης. Σημείωση: Λόγω της σχέσης One-to-One μεταξύ του κόμβου και του αντικειμένου περιεχομένου, το NHibernate πάντα διαβάζει και το αντικείμενο περιεχομένου μαζί με τον κόμβο.
• <many-to-one>	Δηλώνει μια σχέση πολλά-προς-ένα μεταξύ των παιδιών και του γονικού κόμβου τους. Η ρύθμιση <many-to-one> δηλώνει το ένα άκρο της αμφίδρομης σχέσης. Το άλλο άκρο της σχέσης δηλώνεται στην ρύθμιση <list> που επεξηγούμε παρακάτω.
• name=”…”	Ο γονικός κόμβος περιέχεται στο πεδίο “parent” του στιγμιότυπου.
• column =”…”	Το αναγνωριστικό του γονικού κόμβου αποθηκεύεται στη στήλη “parent_id”.
• unique =”false”	Εφόσον ένας γονικός κόμβος μπορεί να έχει πολλά παιδιά, οι τιμές της στήλης δεν μπορούν να είναι μοναδικές.
• cascade =”none”	Βλ. 3.3.7.2 «Transitive persistence και Cascading persistence». Ορίζει ότι όταν οι αλλαγές του γονικού κόμβου αποθηκεύονται, το NHibernate δεν πρέπει να αποθηκεύει αυτόματα τις αλλαγές που έχουν γίνει στα παιδιά.
• <list>	Μέσω της ρύθμισης <one-to-many>, που θα δούμε παρακάτω, ορίζουμε το άλλο άκρο της σχέσης μεταξύ των παιδιών και του γονικού κόμβου. Με τη <list> δηλώνουμε ότι τα παιδιά αποτελούν ένα ταξινομημένο σύνολο.
• name=”…”	Τα παιδιά θα αποθηκεύονται στο πεδίο “_internalChildren” του γονικού κόμβου, το οποίο είναι τύπου IList<Item>.
• table=”…”	Τα παιδιά θα αποθηκεύονται στον πίνακα “Item”.
• cascade =”all”	Βλ. 3.3.7.2 «Transitive persistence και Cascading persistence». Δηλώνει ότι όταν αποθηκεύονται οι αλλαγές ενός παιδιού, θα αποθηκεύονται αυτόματα και οι αλλαγές που έχουν γίνει στον γονικό κόμβο. Επίσης δηλώνει ότι όταν διαγράφεται ένα παιδί, θα διαγράφονται αναδρομικά και τα παιδιά του, με αποτέλεσμα τη

	διαγραφή ολόκληρου του υπόδενδρου του.
• <one-to-many>	Δηλώνει τη σχέση ένα-προς-πολλά μεταξύ του γονικού κόμβου και των παιδιών του.
• <key>	Δηλώνει ότι το ξένο κλειδί της σχέσης περιέχεται στη στήλη “parent_id”. Στη στήλη αυτή θα αποθηκεύεται ο αναγνωριστικός αριθμός του γονικού κόμβου.
• <index>	Η σειρά των παιδιών έχει σημασία. Η θέση του κάθε παιδιού θα αποθηκεύεται στη στήλη “SortOrder” του πίνακα.
• inverse =”true”	Δηλώνουμε ότι η ρύθμιση <one-to-many> αποτελεί το άλλο άκρο της σχέσης <many-to-one>.
• batch-size=”20”	Δηλώνει ότι τα παιδιά του γονικού κόμβου θα πρέπει να μεταφέρονται από τη βάση δεδομένων κατά εικοσάδες, για λόγους αποδοτικότητας.
• <bag>	Δηλώνει ότι οι λεζάντες αποτελούν ένα μη-ταξινομημένο σύνολο, το οποίο αποθηκεύεται στο πεδίο “tags” του κόμβου (τύπου IList<string>). Στη βάση δεδομένων οι λεζάντες αποθηκεύονται στον πίνακα “ItemTag”. Οι λεζάντες αποτελούν συλλογή από value-type αντικείμενα, βλ. 3.3.5.5 «Απεικόνιση συλλογών από value-type αντικείμενα».
• <bag>	Με το <one-to-many> που επεξηγούμε παρακάτω, δηλώνουμε μια σχέση ένα-προς-πολλά μεταξύ ενός υποκείμενου κόμβου και των συσχετίσεων του (RelItem αντικείμενα). Το <bag> δηλώνει ότι οι συσχετίσεις αποτελούν ένα μη-ταξινομημένο σύνολο. Σε αυτό το αρχείο ορίζουμε το ένα άκρο της σχέσης. Το άλλο άκρο της σχέσης ορίζεται στο RelItem.hbm.xml.
• cascade= ”all-delete-orphan ”	Βλ. 3.3.7.2 «Transitive persistence και Cascading persistence». Επιπρόσθετα της επίδρασης του “all” που είδαμε παραπάνω, κατά την αποθήκευση του υποκείμενου κόμβου, κάθε συσχέτιση η οποία έχει αφαιρεθεί από τη λίστα _internalRelatives θα διαγράφεται από τη βάση δεδομένων.
• <one-to-many>	Δηλώνει τη σχέση ένα-προς-πολλά μεταξύ του υποκείμενου κόμβου και των συσχετίσεων του. Οι συσχετίσεις είναι στιγμιότυπα της κλάσης RelItem.

• <key>	Δηλώνει ότι το ξένο κλειδί της σχέσης περιέχεται στη στήλη “subject_item_id” του πίνακα στον οποίο απεικονίζονται οι συσχετίσεις. Στη στήλη αυτή θα αποθηκεύεται ο αναγνωριστικό αριθμός του υποκείμενου κόμβου.
• inverse =”true”	Δηλώνουμε ότι ορίζουμε το άλλο άκρο της σχέσης <many-to-one> που ορίζεται στο RelItem.hbm.xml.
• <one-to-one>	Δηλώνει μια σχέση ένα-προς-ένα μεταξύ ενός κόμβου και του αντικειμένου περιεχομένου του. Το αντικείμενο περιεχομένου θα αποθηκεύεται στο πεδίο “content” του κόμβου, τύπου ContentBase. Στο ContentBase.hbm.xml αρχείο ορίζεται το άλλο άκρο της σχέσης.

Πίνακας 6-1 Επεξήγηση των ρυθμίσεων απεικόνισης της κλάσης Item

Μας μένει ακόμα μια εκκρεμότητα για να ολοκληρώσουμε την απεικόνιση των κόμβων. Πρέπει να λύσουμε το πρόβλημα της ταυτότητας / ισότητας (βλ. 3.3.5.6 «Το πρόβλημα της ταυτότητας και της ισότητας»). Για το λόγο αυτό πρέπει να υλοποιήσουμε τις μεθόδους Equals() και GetHashCode() κατάλληλα.

Για την υλοποίηση των δύο μεθόδων, έχουμε στηριχθεί στην κλασσική τακτική που συνδυάζει το πρωτεύον κλειδί του κόμβου και τις .NET αναφορές των στιγμιότυπων για να εξάγει την ισότητα. Οι μέθοδοι έχουν υλοποιηθεί ως εξής:

```
public override bool Equals(object obj)
{
    return obj is Item && this == (Item)obj;
}
public override int GetHashCode()
{
    if (this.id > 0)
        return this.id;
    else
        return base.GetHashCode();
}
public static bool operator ==(Item x, Item y)
{
    // If both are null, or both are same instance, return true.
    if (System.Object.ReferenceEquals(x, y))
        return true;
    // If one is null, but not both, return false.
    if (((object)x == null) || ((object)y == null))
```

```
            return false;
        // different references
        return x.Id == y.Id && x.Id > 0;
    }
```

Σημείωση: Υπενθυμίζουμε ότι το ID ενός κόμβου είναι 0 αν το αντικείμενο είναι παροδικό, δηλαδή αν δεν έχει ακόμα αποθηκευτεί από το NHibernate ώστε να πάρει το ID τιμή.

6.1.2.3 Ρυθμίσεις απεικόνισης της κλάσης RelItem

Παρουσιάζουμε το αρχείο RelItem.hbm.xml που περιέχει τις ρυθμίσεις απεικόνισης της κλάσης RelItem στον πίνακα ItemRel της βάσης δεδομένων. Ύστερα επεξηγούμε τις αξιοσημείωτες ρυθμίσεις ξεχωριστά.

```
<?xml version="1.0" encoding="utf-8" ?>
<hibernate-mapping xmlns="urn:nhibernate-mapping-2.2">
<!-- The instances of the RelItem class will be stored in the
"ItemRel" table -->
<class name="NCommet.Core.RelItem, NCommet.Core" table="ItemRel"
lazy="false">
  <!-- Composite Identity -->
  <composite-id>
    <!-- Field "owner": Many-to-One association between RelItems
and their subject item. -->
    <key-many-to-one
      name="owner" access="field"
      column="subject_item_id"
      class="NCommet.Core.Item, NCommet.Core" />
    <!-- Field "relative": Many-to-One association between
RelItems and their object item. -->
    <key-many-to-one
      name="relative" access="field"
      column="object_item_id"
      class="NCommet.Core.Item, NCommet.Core" />
    <!-- Field "relationType" -->
    <key-property
      name="relationType" access="field"
      column="RelationType" type="String" length="50" />
  </composite-id>
  <!-- Field "dttm" -->
  <timestamp access="field" column="dttm" name="dttm" />
```

```
  <!-- Reinforce the Many-to-One association between RelItems
and their subject item. -->
  <many-to-one
    name="Owner" access="property"
    column="subject_item_id"  not-null="true"
    class="NCommet.Core.Item, NCommet.Core"
    cascade="save-update" insert="false" update="false" />
  <!-- Reinforce the Many-to-One association between RelItems
and their object item. -->
  <many-to-one
    name="Relative" access="property"
    column="object_item_id" not-null="true"
    class="NCommet.Core.Item, NCommet.Core"
    cascade="save-update" insert="false" update="false" />
</class>
</hibernate-mapping>
```

Ρύθμιση	**Επεξήγηση**
• <composite-id>	Δηλώνει ένα σύνθετο πρωτεύον κλειδί.
• <key-many-to-one>	Δηλώνει ότι η τιμή του πεδίου “owner” είναι μέρος του σύνθετου κλειδιού και αποθηκεύεται στη στήλη “subject_item_id” του πίνακα ItemRel. Επιπρόσθετα, ορίζει μια σχέση πολλά-προς-ένα μεταξύ των συσχετίσεων και του υποκείμενου κόμβου (τύπου Item) που συμμετέχει σε αυτές. Γι’ αυτό το λόγο στην στήλη “subject_item_id” αποθηκεύεται ο αναγνωριστικός αριθμός του υποκείμενου κόμβου της συσχέτισης. Σε αυτό το σημείο ορίζεται το άλλο άκρο της σχέσης One-to-Many που είδαμε στο αρχείο Item.hbm.xml που αφορούσε τις συσχετίσεις.
• <key-many-to-one>	Δηλώνει ότι η τιμή του πεδίου “relative” είναι μέρος του σύνθετου κλειδιού και αποθηκεύεται στη στήλη “object_item_id” του πίνακα ItemRel. Επιπρόσθετα, ορίζει μια σχέση πολλά-προς-ένα (μονής κατεύθυνσης) μεταξύ των συσχετίσεων και του αντικείμενου κόμβου (τύπου Item) που συμμετέχει σε αυτές. Γι’ αυτό το λόγο στην στήλη “object_item_id” αποθηκεύεται ο αναγνωριστικός αριθμός του

	αντικείμενου κόμβου της συσχέτισης.
• <key-property>	Δηλώνει ότι η τιμή του πεδίου “relationType” είναι μέρος του σύνθετου κλειδιού και θα αποθηκεύεται στη στήλη “RelationType”.
• <timestamp>	Δηλώνει ότι η τιμή του πεδίου “dttm” θα αποθηκεύεται στη στήλη “dttm”. Το timestamp το διαχειρίζεται το NHibernate και αποτελεί ένα μέτρο με το οποίο μπορεί να διακρίνει αν άλλαξε ένα στιγμιότυπο από την τελευταία φορά που αποθηκεύτηκε. Το timestamp δεν έχει σημασία για το NCommet.

Πίνακας 6-2 Επεξήγηση των ρυθμίσεων απεικόνισης της κλάσης RelItem

Σε αυτό το σημείο ίσως φαίνεται περίεργο ότι στις τελευταίες ρυθμίσεις <many-to-one> ορίζουμε εκ νέου τις ίδιες σχέσεις πολλά-προς-ένα που ορίσαμε στις ρυθμίσεις <key-many-to-one>.

Αν δεν ορίζαμε τις ρυθμίσεις <many-to-one>, δημιουργείται το εξής πρόβλημα: Αν η εφαρμογή συσχετίσει δύο παροδικούς κόμβους και ύστερα τους αποθηκεύσει, τότε θα δημιουργηθεί μια εξαίρεση που εξηγεί ότι δεν επιτρέπονται κενές τιμές στη στήλη “object_item_id” του πίνακα ItemRel. Αυτό που συμβαίνει είναι ότι το NHibernate προσπαθεί να αποθηκεύσει τη συσχέτιση πριν αποθηκεύσει τον παροδικό αντικείμενο κόμβο, με αποτέλεσμα να μην έχει πάρει ακόμα αναγνωριστικό αριθμό και έτσι να μην μπορεί να αποθηκευτεί η συσχέτιση.

Δυστυχώς το NHibernate έχει την παραπάνω δυσκολία για την απεικόνιση σχέσεων που ορίζονται ως μονής κατεύθυνσης και όχι αμφίδρομες (όπως η σχέση μεταξύ των συσχετίσεων και του αντικείμενου κόμβου), μέσα μάλιστα σε σύνθετο αναγνωριστικό.

Με τις πλεονάζουσες ρυθμίσεις <many-to-one>, μπορούμε να λύσουμε το παραπάνω πρόβλημα ορίζοντας τη ρύθμιση cascade=”save-update”, η οποία αναγκάζει το NHibernate να αποθηκεύσει τους κόμβους πριν αποθηκεύσει τη συσχέτιση. Βεβαίως, ορίζουμε τις ρυθμίσεις ως insert=”false” και update=”false”, ειδάλλως το NHibernate σε κάθε εισαγωγή ή ανανέωση μιας γραμμής του πίνακα ItemRel θα προσπαθούσε να θέσει τις στήλες «subject_item_id» και «object_item_id» εις διπλούν καταλήγοντας σε εξαίρεση.

Οι αναγκαίες ρυθμίσεις <many-to-one> έχουν το εξής μειονέκτημα: Δεν μπορούν να αναφέρονται στο ίδιο πεδίο της κλάσης που αναφέρονται οι ρυθμίσεις <key-many-to-one>. Το NHibernate απαγορεύει δύο ή περισσότερες ρυθμίσεις που αναφέρονται στην ίδια ιδιότητα ή στο ίδιο πεδίο μιας κλάσης. Γι’ αυτό το λόγο, οι ρυθμίσεις <many-to-one> αναφέρονται στις ιδιότητες Owner και Relative που τελικώς αναφέρονται στα πεδία owner και relative της κλάσης RelItem.

Μας μένει ακόμα η εκκρεμότητα να λύσουμε το πρόβλημα της ταυτότητας / ισότητας, υλοποιώντας τις μεθόδους Equals() και GetHashCode(), όπως κάναμε παραπάνω για την απεικόνιση της κλάσης Item.

Για το RelItem έχουμε ακολουθήσει διαφορετική τακτική απ' ότι για το Item, γιατί το RelItem δεν έχει ως πρωτεύον κλειδί έναν αναγνωριστικό αριθμό αλλά ένα συνδυασμό από τρεις στήλες και πρέπει να τις πάρουμε υπ' όψιν κατά την υλοποίηση των δύο μεθόδων.

Οι μέθοδοι έχουν υλοποιηθεί ως εξής:

```
public override bool Equals(object obj)
{
    if (this == obj)
        return true;
    if (obj == null || !(obj is RelItem))
        return false;
    RelItem that = (RelItem)obj;
    return
        this.owner == that.owner &&
        this.relative == that.relative &&
        this.relationType == that.relationType;
}
public override int GetHashCode()
{
    int result = 0;
    if (this.owner != null)
        result = this.owner.GetHashCode();
    if (this.relative != null)
        result = result * 11 + this.relative.GetHashCode();
    if (this.relationType != null)
        result = result * 13 + this.relationType.GetHashCode();
    return result;
}
```

Πρέπει να τονίσουμε ότι στη μέθοδο GetHashCode(), δημιουργούμε έναν ακέραιο αριθμό ο οποίος είναι μοναδικός για κάθε διαφορετική συσχέτιση. Για αυτό δημιουργούμε έναν ακέραιο ο οποίος βασίζεται στον πολλαπλασιασμό πρώτων αριθμών με τα hash codes των αντικειμένων που αποτελούν το πρωτεύον κλειδί της συσχέτισης. Μόνο οι συσχετίσεις οι οποίες είναι ίσες, δηλαδή έχουν τον ίδιο τύπο και τους ίδιους υποκείμενους και αντικείμενους κόμβους, θα έχουν το ίδιο hash code.

6.1.2.4 Ρυθμίσεις απεικόνισης των κλάσεων περιεχομένου της εφαρμογής

Στην ενότητα 5.2.1.2 «ContentBase», αναφέραμε ότι οι κλάσεις περιεχομένου που ορίζει η εφαρμογή πρέπει να:

- Κληρονομούν την abstract κλάση ContentBase, δηλαδή να είναι υποκλάσεις της.
- Να είναι XML Serializable ώστε μέσω της ιδιότητας SerializedData να μπορούμε να έχουμε το String (μορφής XML) το οποίο αντιπροσωπεύει το στιγμιότυπο και από το οποίο μπορεί να ανακτηθεί το στιγμιότυπο εκ νέου.

Επειδή η ContentBase κληρονομείται, πρέπει να χρησιμοποιήσουμε μία από τις τεχνικές απεικόνισης της κληρονομικότητας που προσφέρει το NHibernate (βλ. 3.3.6 «Απεικόνιση Κληρονομικότητας»). Η τεχνική κληρονομικότητας που θα χρησιμοποιήσουμε είναι “ένας πίνακας ανά ιεραρχία κλάσεων”. Με αυτήν την τεχνική, τα στιγμιότυπα όλων των υποκλάσεων θα αποθηκεύονται στον ίδιο πίνακα, ονόματι ”ContentBase”. Δηλαδή κάθε γραμμή του πίνακα θα μπορεί να απεικονίζει οποιαδήποτε υποκλάση της ContentBase.

Θα χρησιμοποιήσουμε μία στήλη του πίνακα, ονόματι “cb_type”, η οποία θα αποτελεί την ειδοποιό διαφορά (discriminator) μεταξύ των υποκλάσεων. Η στήλη θα περιέχει τον τύπο της υποκλάσης που απεικονίζει η συγκεκριμένη γραμμή του πίνακα. Όταν το στιγμιότυπο χρειαστεί να ανακτηθεί, τότε, ανάλογα με την τιμή της στήλης, το NHibernate θα δημιουργεί ένα στιγμιότυπο της κατάλληλης υποκλάσης.

Εφόσον όλες οι υποκλάσεις είναι XML Serializable, για την αποθήκευση των στιγμιότυπων τους μας αρκεί μόνο μια στήλη, που θα την ονομάσουμε “SerializedData” και η οποία θα περιέχει το serialized string των αντικειμένων περιεχομένου. Δεν χρειάζονται παραπάνω στήλες για τις επιπλέον ιδιότητες που πιθανόν να ορίζουν οι υποκλάσεις.

Με τα παραπάνω, το μόνο που χρειάζεται να κάνει μια εφαρμογή είναι να δημιουργήσει ένα .hbm.xml αρχείο για κάθε κλάση περιεχομένου της. Το κάθε .hbm.xml αρχείο θα περιέχει απλώς μια ρύθμιση <subclass> η οποία ενημερώνει το NHibernate ότι η συγκεκριμένη κλάση είναι υποκλάση της ContentBase. Έτσι το NCommet θα γνωρίζει πώς να αποθηκεύει τα στιγμιότυπα των κλάσεων περιεχομένου της εφαρμογής στον πίνακα “ContentBase”. Στην ενότητα 6.1.3.3, παρουσιάζουμε πως πρέπει η εφαρμογή να ορίσει τα .hbm.xml αρχεία της.

Έχοντας εξηγήσει τα παραπάνω, παρουσιάζουμε το αρχείο ContentBase.hbm.xml που περιέχει τις ρυθμίσεις απεικόνισης των στιγμιότυπων των υποκλάσεων της κλάσης ContentBase στον πίνακα “ContentBase” της βάσης δεδομένων. Ύστερα επεξηγούμε τις αξιοσημείωτες ρυθμίσεις ξεχωριστά.

```
<?xml version="1.0" encoding="utf-8" ?>
<hibernate-mapping xmlns="urn:nhibernate-mapping-2.2">
```

```
<!-- The instances of the ContentBase class will be stored in
the "ContentBase" table -->
<class name="NCommet.Core.ContentBase, NCommet.Core"
table="ContentBase">
      <!-- Identity -->
  <id
    name="id" access="field"
    column="ID" type="Int32" unsaved-value="0">
    <!-- References the Item's ID -->
    <generator class="foreign">
      <param name="property">container</param>
    </generator>
  </id>
  <!-- Column "cb_type": The default discriminator value of
subclasses is their class name. -->
  <discriminator
    column="cb_type" type="String" length="50" not-null="true"/>
  <!-- Property "SerializedData" -->
  <property
    name="SerializedData" access="property"
    column="SerializedData" type="String" length="4000" not-
null="true" />
  <!-- Field "container": One-to-One association between a
ContentBase object and its container Item. -->
  <one-to-one
    name="container" access="field"
    class="NCommet.Core.Item, NCommet.Core" constrained="true"
/>
</class>
</hibernate-mapping>
```

Ρύθμιση	**Επεξήγηση**
• <id>	
• <generator>	Δηλώνει ότι ο αναγνωριστικός αριθμός είναι και ξένο κλειδί του πίνακα. Το ξένο κλειδί δείχνει τον αναγνωριστικό αριθμό του κόμβου που περιέχει το αντικείμενο περιεχομένου.
• <discriminator>	Η ρύθμιση αυτή δηλώνει στο NHibernate ποια στήλη αποτελεί την

	ειδοποιό διαφορά μεταξύ των στιγμιότυπων των διαφορετικών υποκλάσεων της ContentBase που θα αποθηκεύονται στον πίνακα “ContentBase”, σύμφωνα με την τεχνική κληρονομικότητας “table per class hierarchy”. Η στήλη περιέχει το πλήρες όνομα (fully qualified name) της υποκλάσης.
• <property>	Δηλώνει ότι η τιμή της ιδιότητας “SerializedData” θα αποθηκεύεται στη στήλη “SerializedData”. Η τιμή είναι τύπου String, περιορίζεται στους 4000 χαρακτήρες και δεν πρέπει να είναι ποτέ κενή. Σημείωση: Ξεφεύγουμε από τη συνήθη τακτική να αποθηκεύουμε πεδία. Στην περίπτωση της ιδιότητας SerializedData, δεν υπάρχει αντίστοιχο πεδίο. Γι’ αυτό το λόγο, ορίζουμε το access=”property” για να δηλώσουμε στο NHibernate ότι πρέπει να προσπελάσει την ιδιότητα και όχι κάποιο πεδίο.
• <one-to-one>	Δηλώνει μια σχέση ένα-προς-ένα μεταξύ του αντικειμένου περιεχομένου και του κόμβου που το περιέχει. Ο κόμβος θα αποθηκεύεται στο πεδίο “container” του αντικειμένου περιεχομένου, τύπου Item. Στο Item.hbm.xml αρχείο ορίζεται το άλλο άκρο της σχέσης.
• constrained =”true”	Δηλώνει ότι το πρωτεύον κλειδί του αντικειμένου περιεχομένου αποτελεί και ένα ξένο κλειδί προς τον πίνακα που περιλαμβάνει το άλλο άκρο της one-to-one σχέσης. Δηλαδή δηλώνει ότι η στήλη “id” του πίνακα “ContentBase” αποτελεί και ένα ξένο κλειδί προς τη στήλη “id” του πίνακα “Item”.

Πίνακας 6-3 Επεξήγηση των ρυθμίσεων απεικόνισης των υποκλάσεων της κλάσης ContentBase

Σε αντίθεση με τις κλάσεις Item και RelItem, δεν χρειάζεται να λύσουμε το πρόβλημα της ταυτότητας / ισότητας για τα στιγμιότυπα της ContentBase.

Αυτό συμβαίνει γιατί ένα persistent αντικείμενο περιεχομένου υπάρχει μόνο μέσα σε έναν υπάρχον κόμβο. Δεν έχει από μόνο του πρωτεύον κλειδί, αλλά μοιράζεται το πρωτεύον κλειδί του κόμβου που το περιέχει. Για το λόγο αυτό, δεν χρειάζεται να υλοποιήσουμε τις μεθόδους Equals() και GetHashCode().

6.1.3 Περιγραφή των κλάσεων του assembly

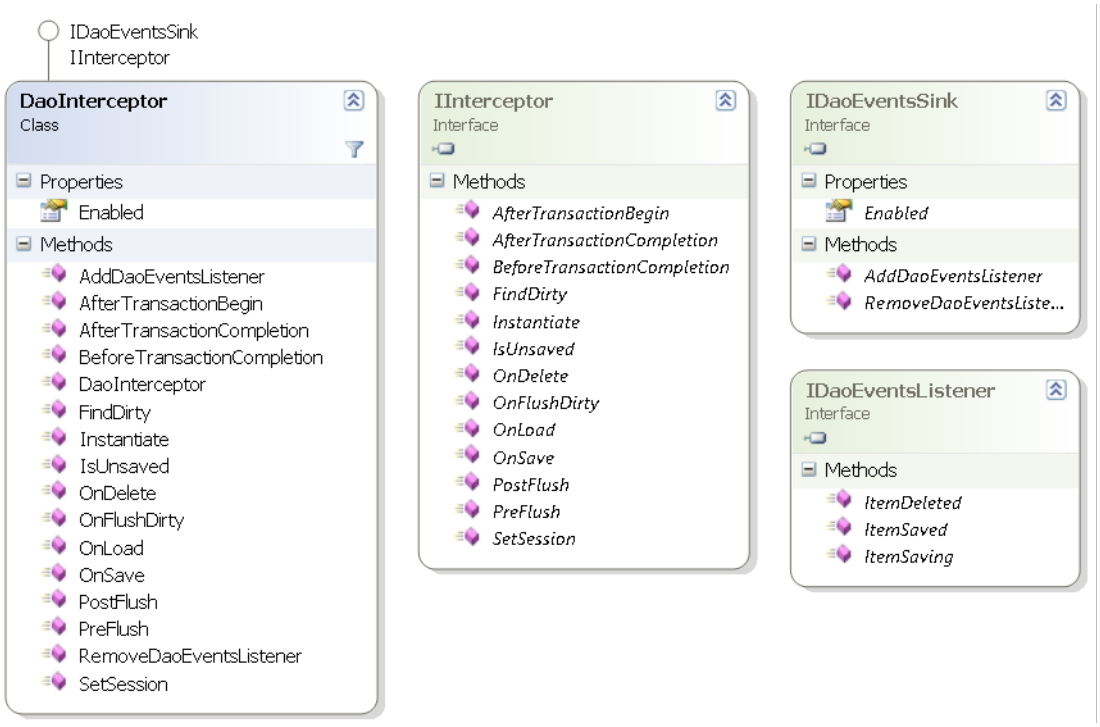
6.1.3.1 CompositeInterceptor

Το NHibernate έχει τον περιορισμό ότι μόνο ένας IInterceptor μπορεί να χρησιμοποιηθεί ανά SessionFactory. Το NCommet.Modules.Dao ορίζει την κλάση CompositeInterceptor η οποία υλοποιεί τη διεπαφή IInterceptor και έχει σκοπό να άρει τον παραπάνω περιορισμό. Δηλαδή υποστηρίζει πολλαπλούς IInterceptors.

Χρησιμοποιώντας τις μεθόδους AddInterceptor() και RemoveInterceptor() μπορούμε να προσθαφαιρέσουμε υλοποιήσεις της διεπαφής IInterceptor. Οι υπόλοιπες μέθοδοι του CompositeInterceptor, που καλούνται από το NHibernate, με τη σειρά τους καλούν τις αντίστοιχες μεθόδους των υλοποιήσεων του IInterceptor που έχουμε προσθέσει στον CompositeInterceptor.

6.1.3.2 DaoInterceptor

Σκοπός του DaoInterceptor είναι να ανακατευθύνει τα συμβάντα του NHibernate στον EventSink του NCommet με διαφανή τρόπο, δηλαδή με τρόπο που δεν θα απασχολεί τον χρήστη. Η εφαρμογή θα ενημερώνεται για όλα τα συμβάντα, που είτε συμβαίνουν στο NCommet είτε συμβαίνουν στο NHibernate, μόνο μέσω του EventSink.



Εικόνα 6-3 Διάγραμμα κλάσης του DaoInterceptor

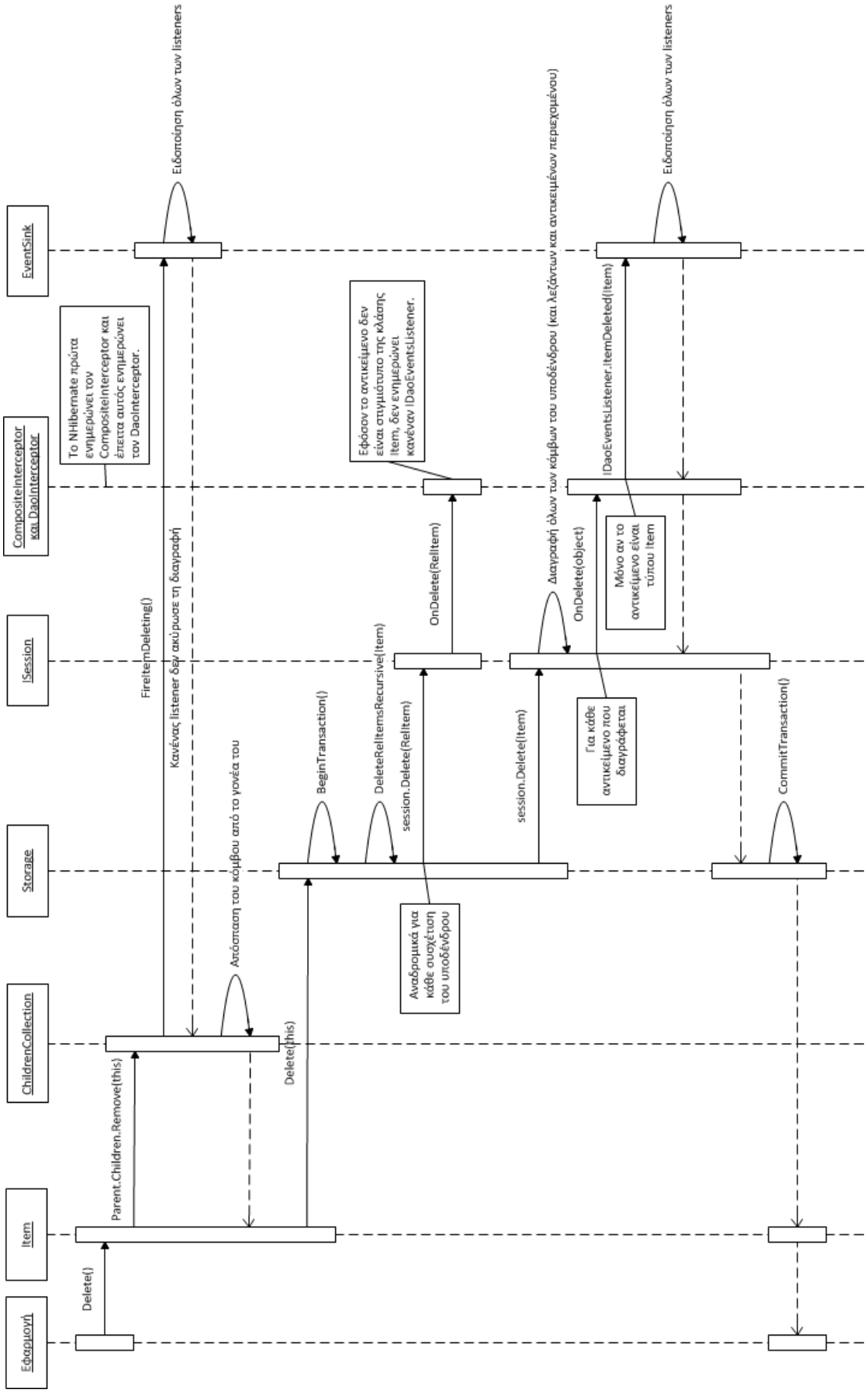
Για να πετύχει το σκοπό του, η κλάση DaoInterceptor υλοποιεί τις εξής διεπαφές:

- Υλοποιεί τη διεπαφή IInterceptor ώστε να ενημερώνεται για τα συμβάντα (events) του NHibernate. Ο DaoInterceptor προστίθεται αυτόματα στον CompositeInterceptor[4] που χρησιμοποιεί ο NHibernateSessionManager.
- Υλοποιεί τη διεπαφή IDaoEventsSink ώστε να μπορεί ο EventSink του NCommet, που υλοποιεί τη διεπαφή IDaoEventsListener, να εγγραφεί σε αυτόν.

Ο DaoInterceptor πρέπει να ρυθμίζεται από την εφαρμογή (στις ρυθμίσεις του NCommet του αρχείου ρυθμίσεων της) ως η υλοποίηση της διεπαφής IDaoEventsSink του NCommet.

Για να καταλάβουμε καλύτερα τη λειτουργία των συμβάντων ας δούμε ένα παράδειγμα διαγραφής ενός κόμβου. Στο παρακάτω ακολουθιακό διάγραμμα φαίνεται η διαφανής ενημέρωση του EventSink.

[4] Το NCommet.Modules.Dao χρειάζεται να χρησιμοποιήσει τον DaoInterceptor ως IInterceptor του NHibernate για να υποστηρίξει τα συμβάντα του NCommet με διαφανή τρόπο. Ο CompositeInterceptor δημιουργήθηκε επειδή το NHibernate επιτρέπει μόνο έναν IInterceptor ανά SessionFactory και, για λόγους λειτουργικότητας, δεν έπρεπε να στερήσουμε τη δυνατότητα χρησιμοποίησης επιπλέον IInterceptors.

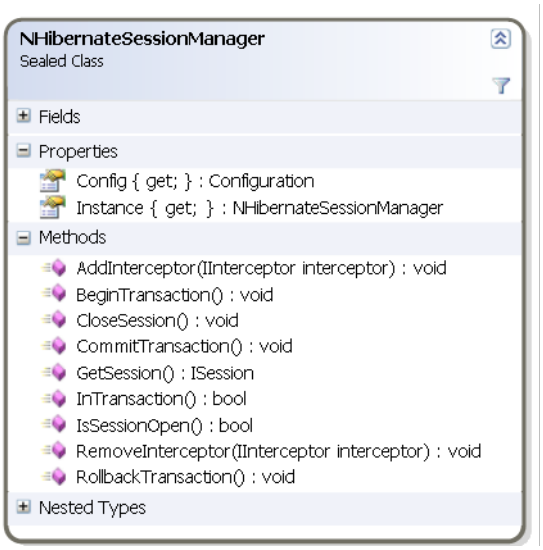


Εικόνα 6-4 Ακολουθιακό διάγραμμα της διαγραφής ενός κόμβου

Επίσης, στο παραπάνω διάγραμμα βλέπουμε την ουσιαστική χρησιμότητα της διεπαφής IInterceptor. Χωρίς αυτή τη διεπαφή δεν θα μπορούσαμε να αντλήσουμε τα συμβάντα για τη διαγραφή κάθε κόμβου του υπόδενδρου. Θα μπορούσαμε μόνο να παράγουμε ένα συμβάν για τη διαγραφή του αρχικού κόμβου του υπόδενδρου τον οποίο ζητάει η εφαρμογή να διαγράψει.

6.1.3.3 NHibernateSessionManager και StorageCleanupModule

Είναι μια singleton κλάση με σκοπό τη δημιουργία και τη διαχείριση των συνεδριών (sessions) του NHibernate που θα χρησιμοποιήσει η κλάση Storage.



Εικόνα 6-5 Διάγραμμα κλάσης του NHibernateSessionManager

Κατά τη δημιουργία της κλάσης, ρυθμίζεται το NHibernate και δημιουργείται το SessionFactory που θα χρησιμοποιείται. Για το SessionFactory ορίζεται ως IInterceptor ο CompositeInterceptor.

Οι ρυθμίσεις για το NHibernate αντλούνται αυτόματα από το αρχείο ρυθμίσεων της εφαρμογής app.config ή web.config. Ένα παράδειγμα ρυθμίσεων είναι οι εξής:

```
  <nhibernate>
    <add key="hibernate.connection.provider"
         value="NHibernate.Connection.DriverConnectionProvider"/>
    <add key="hibernate.dialect"
         value="NHibernate.Dialect.MsSql2000Dialect"/>
    <add key="hibernate.connection.driver_class"
         value="NHibernate.Driver.SqlClientDriver"/>
    <add key="hibernate.connection.connection_string" value="Data
Source=.;Database=TestBlogs;Trusted_Connection=yes;"/>
  </nhibernate>
```

Επίσης, κατά τη ρύθμιση, αντλούνται αυτόματα τα assemblies που περιέχουν τα .hbm.xml που περιέχουν τις ρυθμίσεις απεικόνισης που θα ληφθούν υπ' όψιν από το NHibernate. Τα assemblies αυτά πρέπει να έχουν οριστεί στο αρχείο ρυθμίσεων της εφαρμογής ως εξής:

```
<appSettings>
  …
  <add key="HBM_ASSEMBLY_COUNT" value="2"/>
  <add key="HBM_ASSEMBLY_1" value="NCommet.Modules.Dao"/>
  <add key="HBM_ASSEMBLY_2" value="[Fully qualified name of the
application's assembly]"/>
</appSettings>
```

Σημείωση: Μπορούν να οριστούν περισσότερα των 2 assemblies. Όμως το NCommet.Modules.Dao πρέπει να είναι απαραίτητα πρώτο.

Το assembly NCommet.Modules.Dao έχει ενσωματωμένα τα αρχεία ρυθμίσεων απεικόνισης των βασικών κλάσεων του NCommet που περιγράψαμε στην ενότητα 6.1.2 «Τα αρχεία ρυθμίσεων απεικόνισης». Το assembly της εφαρμογής πρέπει να έχει embedded τα .hbm.xml αρχεία που ρυθμίζουν τις κλάσεις περιεχομένου της εφαρμογής ως <subclass> (υποκλάσεις) της κλάσης ContentBase. Έτσι το NHibernate θα γνωρίζει ότι όλα τα αντικείμενα περιεχομένου πρέπει να απεικονίζονται σύμφωνα με τις ρυθμίσεις της ContentBase.

Για παράδειγμα, έστω ότι μια εφαρμογή ορίζει την εξής κλάση περιεχομένου:

```
[Serializable]
public class Blog : ContentBase
{
    public virtual string Title { get; set; }
    public virtual string Author { get; set; }
}
```

Τότε θα πρέπει να ενσωματώσει στο assembly της το εξής Blog.hbm.xml αρχείο:

```
<?xml version="1.0" encoding="utf-8" ?>
<hibernate-mapping xmlns="urn:nhibernate-mapping-2.2">
    <subclass name="Blog" extends="NCommet.Core.ContentBase,
NCommet.Core" />
</hibernate-mapping>
```

Ο NHibernateSessionManager υποστηρίζει μόνο μια τρέχουσα συνεδρία ανά thread και μόνο μια τρέχουσα συναλλαγή ανά thread. Το επιτυγχάνει αποθηκεύοντας τη συνεδρία ή τη συναλλαγή στο HttpContext (αν το thread είναι μιας web εφαρμογής) είτε στο CallContext. Αυτή η τακτική έχει τα εξής πλεονεκτήματα:

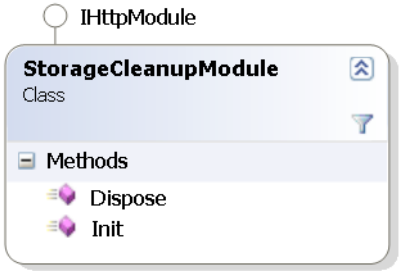
- Αποφεύγει τη δημιουργία περιττών συνεδριών και περιττών συναλλαγών.

- Δίνει τη δυνατότητα αναφοράς στην ίδια συνεδρία ή στην ίδια συναλλαγή από οποιοδήποτε σημείο του κώδικα που εκτελείται από το ίδιο thread.

Όμως, ειδικότερα για τις διαδικτυακές asp.net εφαρμογές, προτείνεται στο αρχείο ρυθμίσεων web.config να προστεθεί στην ενότητα των <httpModules>, η κλάση StorageCleanupModule, ως εξής:

```
<httpModules>
      …
      <add name="StorageCleanupModule"
type="NCommet.Modules.Dao.StorageCleanupModule,
NCommet.Modules.Dao"/>
</httpModules>
```

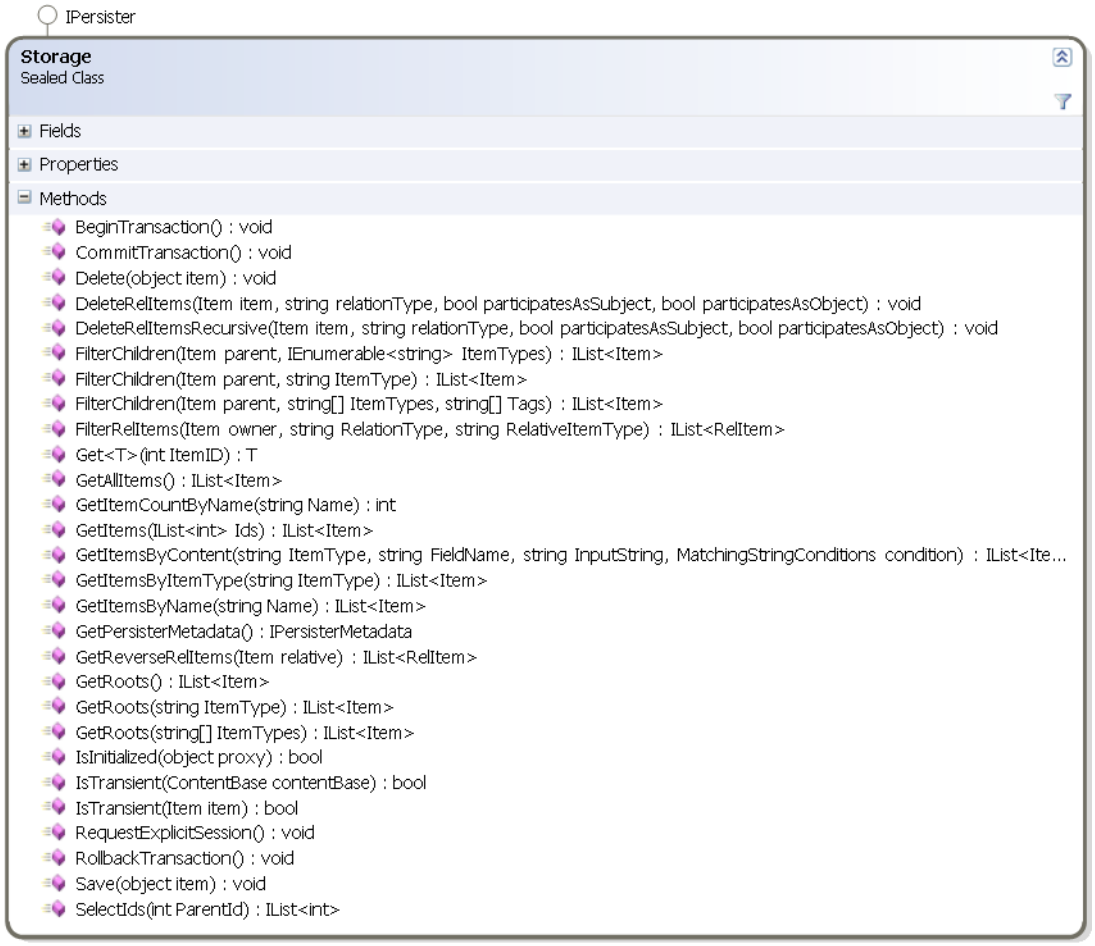
Η κλάση αυτή υλοποιεί τη διεπαφή IHttpModule της ASP.NET ώστε, κατά τον τερματισμό ενός HttpContext, δηλαδή κατά τον τερματισμό μιας αίτησης, να κάνει commit τη συναλλαγή που ίσως έχει δημιουργηθεί στο τρέχων HttpContext και να κλείνει τη συνεδρία που έχει δημιουργηθεί στο τρέχων HttpContext.



Εικόνα 6-6 Διάγραμμα κλάσης του StorageCleanupModule

6.1.3.4 Storage

Η κλάση Storage αποτελεί την έτοιμη υλοποίηση της διεπαφής IPersister της Υπηρεσίας Αποθήκευσης του NCommet που περιγράψαμε στην ενότητα 5.2.5.1 «IPersister και IDaoEventsSink».



Εικόνα 6-7 Διάγραμμα κλάσης της Storage

Εσωτερικά οι περισσότερες μέθοδοι χρησιμοποιούν την κλάση NHibernateSessionManager για την ανάκτηση μιας συνεδρίας ή μιας συναλλαγής (transaction) του NHibernate. Επίσης, οι μέθοδοι ανάκτησης δεδομένων χρησιμοποιούν την γλώσσα HQL.

Στον παρακάτω πίνακα περιγράφουμε μερικές αξιοσημείωτες μεθόδους και τον τρόπο με τον οποίο υλοποιούνται.

Μέθοδος	**Περιγραφή**
GetRoots (string[] ItemTypes)	Αντλεί τους κόμβους-ρίζες του συστήματος. Χρησιμοποιεί την εξής εντολή εκτέλεσης HQL ερωτήματος: `return session.CreateQuery("FROM Item it WHERE it.parent IS NULL AND it.itemType IN (:itemTypes)").SetParameterList("itemTypes", ItemTypes).List<Item>();`
DeleteRelItems (…)	Διαγραφή των συσχετίσεων ενός κόμβου, στις οποίες συμμετέχει είτε ως υποκείμενος είτε ως αντικείμενος κόμβος είτε ως και με τους δύο ρόλους. Εκτελεί ένα HQL ερώτημα της μορφής:

	`"FROM RelItem ri WHERE (ri.owner=:owner OR ri.relative=:relative) AND ri.relationType=:relationType"` και διαγράφει τα αντικείμενα που επιστρέφονται.
Delete (object item)	Διαγραφή ενός αντικειμένου από τη βάση δεδομένων. Αν δεν υπάρχει ήδη μια συναλλαγή, δημιουργεί μία μέσα στην οποία εκτελεί τη διαγραφή. Το αντικείμενο μπορεί να είναι στιγμιότυπο οποιασδήποτε κλάσης για την οποία το NHibernate έχει ρυθμίσεις απεικόνισης. Αν κληθεί για να διαγράψει έναν κόμβο, τότε, πριν τη διαγραφή του υπόδενδρου, διαγράφονται αναδρομικά και οι συσχετίσεις στις οποίες συμμετέχουν οι κόμβοι του υπόδενδρου.
Get<T> (int itemID)	Ανάκτηση ενός αντικειμένου με συγκεκριμένο αναγνωριστικό αριθμό. Πρέπει το NHibernate να έχει τις ρυθμίσεις απεικόνισης της κλάσης T. Συνήθως καλείται για την ανάκτηση ενός κόμβου.
GetItemsByContent (…)	Ανάκτηση των κόμβων που έχουν συγκεκριμένο τύπο κόμβου και που η δοσμένη ιδιότητα του αντικειμένου περιεχομένου τους περιέχει το δοσμένο κείμενο. Εκτελεί ένα HQL ερώτημα της μορφής: `"SELECT it FROM Item it WHERE it.content.SerializedData LIKE :matchingClause AND it.itemType=:itemType"` Για παράδειγμα, έστω ότι θέλουμε να ψάξουμε το κείμενο «3» στην ιδιότητα «Age» με MatchingStringConditions.STARTS_WITH για να βρούμε π.χ. τους πελάτες που βρίσκονται στην τρίτη δεκαετία της ζωής τους. Τότε το matchingClause είναι «`%<Age>3%</Age>%`». Το ερώτημα παίρνει υπ' όψιν ότι, όπως περιγράψαμε στην ενότητα 6.1.2.4 «Ρυθμίσεις απεικόνισης των κλάσεων περιεχομένου της εφαρμογής», οι ιδιότητες των αντικειμένων περιεχομένου έχουν αποθηκευτεί στον πίνακα ContentBase στη στήλη SerializedData σε XML σειριακοποιημένη μορφή. Σημείωση: Το MatchingStringConditions μπορεί να πάρει τιμές STARTS_WITH, CONTAINS και ENDS_WITH.
GetItemsByItemType (…)	Ανάκτηση όλων των κόμβων που έχουν το συγκεκριμένο pattern τύπου. Εκτελεί ένα HQL ερώτημα της μορφής:

	`"FROM Item it WHERE it.itemType LIKE :itemType"`
IsTransient (Item / ContentBase)	Ελέγχει απλώς αν ο αναγνωριστικός αριθμός του κόμβου (ή του κόμβου που περιέχει το αντικείμενο περιεχομένου) είναι μηδέν.
Save (object item)	Αποθήκευση ή ενημέρωση ενός αντικειμένου στη βάση δεδομένων. Αν δεν υπάρχει ήδη μια συναλλαγή, δημιουργεί μία μέσα στην οποία εκτελεί την αποθήκευση. Το αντικείμενο μπορεί να είναι στιγμιότυπο οποιασδήποτε κλάσης για την οποία το NHibernate έχει ρυθμίσεις απεικόνισης.

Πίνακας 6-4 Επεξήγηση της υλοποίησης των μεθόδων της κλάσης Storage

6.2 Πλατφόρμες και προγραμματιστικά εργαλεία

Στην παρούσα ενότητα αναφέρονται τα προγραμματιστικά εργαλεία και οι πλατφόρμες ανάπτυξης λογισμικού που χρησιμοποιούνται από το SimpleCMSPlus, το AbstractCMS και το NCommet στο σύνολο τους.

6.2.1 Εργαλεία

6.2.1.1 Microsoft .NET Framework

[Wik10a] [Wik10d]

Πρόκειται για ένα πλαίσιο ανάπτυξης λογισμικού, το οποίο μπορεί να εγκατασταθεί σε υπολογιστές με λειτουργικό σύστημα Microsoft Windows. Το .NET Framework περιλαμβάνει μια μεγάλη βιβλιοθήκη για κοινά προγραμματιστικά προβλήματα και περιλαμβάνει μια εικονική μηχανή (virtual machine) για τη διαχείριση και την εκτέλεση προγραμμάτων που είναι γραμμένα για αυτό. Υποστηρίζει πολλές γλώσσες προγραμματισμού, συμπεριλαμβανομένου της C#, με τέτοιο τρόπο ώστε κάθε γλώσσα να μπορεί να χρησιμοποιήσει κώδικα που έχει γραφεί με άλλη γλώσσα.

Στο .NET Framework περιλαμβάνεται και η ASP.NET. Πρόκειται για ένα πλαίσιο ανάπτυξης web εφαρμογών που επιτρέπουν στον προγραμματιστή να φτιάξει δυναμικές ιστοσελίδες, εφαρμογές διαδικτύου και διαδικτυακές υπηρεσίες.

6.2.1.2 Microsoft Internet Information Services (IIS)

[Wik10e]

Πρόκειται για έναν εξυπηρετητή ιστού σχεδιασμένο από την Microsoft για το λειτουργικό σύστημα Microsoft Windows. Ο IIS είναι ένας από τους δημοφιλέστερους εξυπηρετητές ιστού. Συνοπτικά, τα πρωτόκολλα που υποστηρίζει είναι τα εξής: HTTP/HTTPS, FTP, FTPS,

SMTP και NNTP. Με την προσθήκη επεκτάσεων, μπορεί να υποστηρίξει και άλλα πρωτόκολλα όπως το WebDAV.

Οι ASP.NET εφαρμογές συνήθως εξυπηρετούνται από τον IIS, καθώς προσφέρει ειδικές δυνατότητες εκτέλεσης και διαχείρισης τους.

6.2.1.3 Microsoft SQL Server

[Wik10b]

Πρόκειται για ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων (RDBMS). Είναι σχεδιασμένος ώστε να εκτελείται σε λειτουργικό σύστημα Microsoft Windows. Χρησιμοποιείται ευρέως από πολλές εταιρίες ανάπτυξης λογισμικού και αποτελεί ένα από τα σημαντικότερα RDBMS που κυκλοφορούν.

Τα κύρια χαρακτηριστικά του SQL Server είναι τα εξής:

- Υποστήριξη των γλωσσών ερωτημάτων T-SQL (Transact SQL) και ANSI SQL.
- Υποστήριξη για stored procedures, indexes, constraints, triggers, views.
- Υποστήριξη για συναλλαγές (transactions).
- Υποστήριξη Full-Text Search.
- Υποστήριξη User-Defined Types.
- Προσφέρει πολλές άλλες υπηρεσίες, οι οποίες παρέχουν επιπλέον λειτουργίες, όπως ο Service Broker, τα Replication Services κ.α.

Για όλους τους παραπάνω λόγους το SimpleCMSPlus χρησιμοποιεί τον SQL Server για τη βάση δεδομένων του.

6.2.1.4 C#

[Wik10c]

Πρόκειται για μια γλώσσα προγραμματισμού που περιλαμβάνει πολλές προγραμματιστικές έννοιες όπως imperative, functional, generic, object-oriented και component-oriented disciplines. Έχει αναπτυχθεί από την Microsoft κυρίως ως γλώσσα προγραμματισμού εφαρμογών που σχεδιάζονται για το .NET Framework.

Ο κύριος σχεδιαστικός στόχος της C# είναι να είναι μια απλή, σύγχρονη, γενικού σκοπού, αντικειμενοστραφής γλώσσα προγραμματισμού. Επίσης, να παρέχει υποστήριξη για πολλές σύγχρονες αρχές της τεχνολογίας λογισμικού όπως string type checking, αυτόματη συλλογή σκουπιδιών κ.α.

6.2.1.5 Λοιπά εργαλεία

Πέραν των παραπάνω βασικών εργαλείων, στο σύνολο του SimpleCMSPlus χρησιμοποιούνται και τα εξής:

- Microsoft Visual Studio: Πρόκειται για ένα IDE (Integrated Development Environment) που ανέπτυξε η Microsoft. Χρησιμοποιείται για την ανάπτυξη εφαρμογών κονσόλας, GUI (Graphical User Interface applications), ιστοσελίδων, web εφαρμογών, διαδικτυακών υπηρεσιών (web services) κ.α. Υποστηρίζει και κώδικα C++ αλλά και managed κώδικα για το .NET Framework.
- NAnt: Πρόκειται για ένα δωρεάν εργαλείο αυτοματοποίησης του compilation και του deployment μιας εφαρμογής που είναι γραμμένη για το .NET Framework.
- NHibernate: Πρόκειται για ένα δωρεάν ORM (Object-Relational mapping) εργαλείο για το .NET Framework. Παρέχει ένα πλαίσιο αυτοματοποιημένης απεικόνισης του αντικειμενοστραφούς μοντέλου δεδομένων μιας εφαρμογής σε παραδοσιακές σχεσιακές βάσεις δεδομένων.
- Log4Net: Είναι ένα δωρεάν εργαλείο για το .NET Framework, το οποίο βοηθάει τον προγραμματιστή να εξάγει μηνύματα ελέγχου και debugging. Υποστηρίζει εμφάνιση μηνυμάτων στην κονσόλα αλλά και την αποθήκευση τους σε αρχεία κειμένου, σε βάση δεδομένων κ.α.
- Fireli: Πρόκειται για μια .NET βιβλιοθήκη που διαχειρίζεται αρχεία και φακέλους με αφαιρετικό τρόπο. Τα αρχεία μπορούν στην πραγματικότητα να είναι αποθηκευμένα σε οποιοδήποτε μέσο όπως το σύστημα αρχείων των Windows ή σε μια σχεσιακή βάση δεδομένων.
- AjaxControlToolkit: Πρόκειται για μια δωρεάν .NET βιβλιοθήκη της Microsoft, η οποία περιέχει ένα σετ από controls και εργαλεία που χρησιμοποιούν την τεχνολογία AJAX (Asynchronous JavaScript and XML). Τη βιβλιοθήκη αυτή τη χρησιμοποιούν κυρίως ASP.NET εφαρμογές.
- URLRewritingNET: Πρόκειται για μια δωρεάν .NET βιβλιοθήκη που χρησιμοποιείται σε ASP.NET εφαρμογές. Προσφέρει τη δυνατότητα αλλαγής των URLs που αιτείται ο χρήστης και την ανακατεύθυνση τους σε άλλα URLs που ορίζει η εφαρμογή.
- MRAdmin: Πρόκειται για μια ASP.NET εφαρμογή που διαχειρίζεται τους χρήστες και τους ρόλους μιας εφαρμογής που χρησιμοποιεί το σύστημα Membership & Roles του .NET [Mic10b].
- FCKEditor: Πρόκειται για ένα δωρεάν επεξεργαστή κειμένου ανοιχτού κώδικα, που μπορεί να χρησιμοποιηθεί σε ιστοσελίδες. Είναι γραμμένος σε JavaScript.
- SharpZipLib: Πρόκειται για μια δωρεάν .NET βιβλιοθήκη για τη διαχείριση συμπιεσμένων αρχείων Zip, GZip, Tar και BZip2.

7

Έλεγχος

Στο κεφάλαιο αυτό αξιολογούμε τη λειτουργία της υλοποίησης της νέας Υπηρεσίας Αποθήκευσης του NCommet που παρουσιάσαμε στην ενότητα 6 «Υλοποίηση». Επικεντρωθήκαμε στη σωστή απεικόνιση του NHibernate και στη σωστή αποθήκευση των δεδομένων που εισάγει ή τροποποιεί ο χρήστης στην CMS εφαρμογή.

7.1 Μεθοδολογία ελέγχου

Κατά τον έλεγχο, χρησιμοποιήσαμε το SimpleCMSPlus ως τη CMS εφαρμογή που χρησιμοποιεί το AbstractCMS και το NCommet. Ως υλοποίηση της Υπηρεσίας Αποθήκευσης του NCommet χρησιμοποιήσαμε το NCommet.Modules.Dao assembly που αναπτύξαμε στην ενότητα 6.1 «Υπηρεσία Αποθήκευσης».

Σκοπός μας είναι να επαληθεύσουμε:

- Τη σωστή υλοποίηση των λειτουργικών απαιτήσεων του NCommet που έχουν σχέση με την Υπηρεσία Αποθήκευσης (βλ. 4.3 «Λειτουργικές απαιτήσεις»).
- Τη σωστή απεικόνιση των αντικειμένων στη βάση δεδομένων, που πραγματοποιεί το NHibernate, σύμφωνα με τα αρχεία ρυθμίσεων απεικόνισης που αναπτύξαμε (βλ. 6.1.2 «Τα αρχεία ρυθμίσεων απεικόνισης»).
- Τη σωστή λειτουργία των κλάσεων που αναπτύχθηκαν στο assembly NCommet.Modules.Dao (βλ. 6.1.3 «Περιγραφή των κλάσεων του assembly»).

Εκτελέσαμε μερικά σενάρια λειτουργίας, όπου χρησιμοποιούνται οι λειτουργικές απαιτήσεις της αποθήκευσης. Τα σενάρια αυτά είναι τα εξής:

- Προβολή περιεχομένου.
- Επεξεργασία περιεχομένου και συναλλαγές.
- Μετακίνηση, αντιγραφή και αναδιάταξη κόμβων.
- Δημιουργία καινούργιου κόμβου.
- Διαχείριση συσχετίσεων.

Ο τρόπος επαλήθευσης των σεναρίων λειτουργίας είχε ως εξής:

1. Εκτελέσαμε το σενάριο λειτουργίας. Ενίοτε χρησιμοποιούσαμε το Backoffice του AbstractCMS. Για παράδειγμα, χρησιμοποιώντας τη σελίδα επεξεργασίας του Backoffice, μπορούσαμε να τροποποιήσουμε ένα πεδίο του αντικειμένου περιεχομένου ενός κόμβου.
2. Επαληθεύσαμε ότι το σενάριο λειτουργίας εκτελέστηκε σωστά ελέγχοντας ότι η ιστοσελίδα του SimpleCMSPlus δείχνει τους τροποποιημένους κόμβους σωστά.
3. Κρατούσαμε δύο αρχεία καταγραφής «log.txt», ένα για το SimpleCMSPlus και ένα για το Backoffice, όπου καταγράφονταν όλες οι ενέργειες και τα SQL ερωτήματα του NHibernate κατά την εκτέλεση του σεναρίου λειτουργίας. Στο τέλος της εκτέλεσης, ελέγξαμε ότι το αρχείο καταγραφής δεν περιείχε σφάλματα και ότι περιείχε αναμενόμενες ενέργειες και SQL ερωτήματα.
4. Για μερικά σενάρια, ελέγξαμε ότι τα δεδομένα που τυχόν τροποποιήθηκαν στη βάση δεδομένων ήταν τα αναμενόμενα, και ότι δεν υπήρξαν παρενέργειες.

Για να δείξουμε τον τρόπο με τον οποίο πραγματοποιήθηκαν τα σενάρια λειτουργίας, παρουσιάζουμε στην ενότητα 7.2 «Αναλυτική παρουσίαση ελέγχου» μερικά από αυτά.

7.1.1 Αρχεία καταγραφής

Για την καταγραφή των ενεργειών και των ερωτημάτων του NHibernate, χρησιμοποιήσαμε την βιβλιοθήκη Apache log4net [Apa10], η οποία υποστηρίζεται αυτόματα από το NHibernate.

Απλώς χρειάστηκε να ρυθμίσουμε κατάλληλα το αρχείο ρυθμίσεων web.config του SimpleCMSPlus ώστε να περιλαμβάνει τις απαραίτητες ρυθμίσεις που κατευθύνουν το NHibernate να καταγράφει τις ενέργειες του, μέσω του log4net, στα αρχεία καταγραφής.

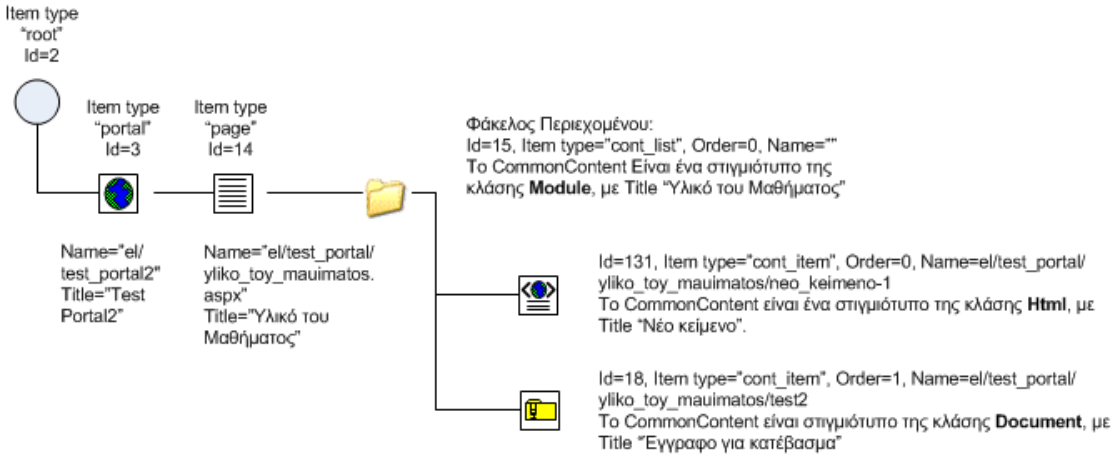
7.2 Αναλυτική παρουσίαση ελέγχου

Στην παρούσα ενότητα παρουσιάζουμε αναλυτικά μερικά από τα σενάρια λειτουργίας που αναφέραμε στην ενότητα 7.1 «Μεθοδολογία ελέγχου». Τα υπόλοιπα σενάρια λειτουργίας πραγματοποιήθηκαν με ανάλογο τρόπο.

7.2.1 Σενάριο προβολής περιεχομένου

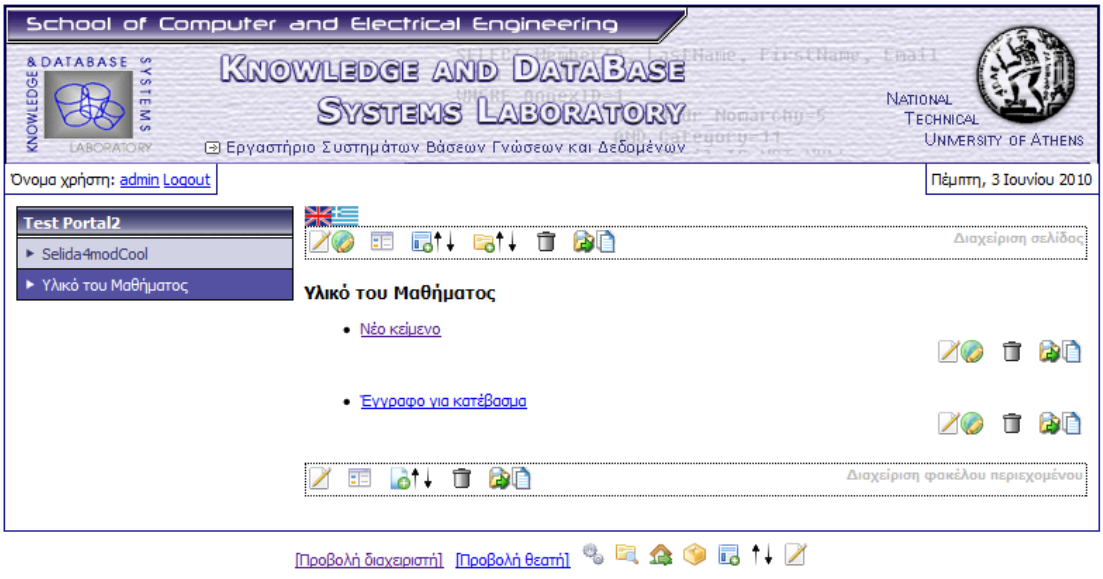
Στο σενάριο αυτό ελέγχουμε ότι το NCommet αντλεί σωστά τους κόμβους από τη βάση δεδομένων ώστε να τους παρουσιάσει το SimpleCMSPlus.

Έστω η εξής ιεραρχία κόμβων (η οποία είναι αποθηκευμένη και στη βάση δεδομένων):



Εικόνα 7-1 Παράδειγμα κόμβου Φακέλου

Αν επισκεφτούμε την τοποθεσία http://.../el/test_portal/yliko_toy_mauimatos.aspx του SimpleCMSPlus (το Url αντιστοιχεί στη Σελίδα), τότε θα παρουσιαστεί η εξής ιστοσελίδα:



Εικόνα 7-2 Παρουσίαση του κόμβου Φακέλου

Από το παραπάνω στιγμιότυπο οθόνης, επιβεβαιώνουμε ότι το NCommet αντλεί σωστά τα δεδομένα από τη βάση δεδομένων.

Μάλιστα, για να επαληθεύσουμε τη λειτουργία του NHibernate, μπορούμε να ελέγξουμε το αρχείο καταγραφής. Ας επικεντρωθούμε μόνο στο Φάκελο περιεχομένου. Οι σχετικές καταχωρήσεις είναι οι εξής:

- **Άντληση του αντικειμένου περιεχομένου του Φακέλου:**

```
SELECT contentbas0_.ID as ID0_1_,
contentbas0_.SerializedData as Serializ3_0_1_,
contentbas0_.cb_type as cb2_1_, item1_.ID as ID1_0_,
item1_.Name as Name1_0_, item1_.ItemType as ItemType1_0_,
item1_.SortOrder as SortOrder1_0_, item1_.CreatedAt as
CreatedAt1_0_, item1_.parent_id as parent6_1_0_
FROM ncommet_SimpleCMSPlus.dbo.ContentBase contentbas0_
inner join ncommet_SimpleCMSPlus.dbo.Item item1_
on contentbas0_.ID=item1_.ID
WHERE contentbas0_.ID=@p0; @p0 = '15'
```

- **Άντληση των θυγατρικών κόμβων του Φακέλου, μαζί με τα αντικείμενα περιεχομένων τους:**

```
SELECT internalc0_.parent_id as parent6___2_, internalc0_.ID
as ID2_, internalc0_.SortOrder as SortOrder__2_,
internalc0_.ID as ID1_1_, internalc0_.Name as Name1_1_,
internalc0_.ItemType as ItemType1_1_, internalc0_.SortOrder
as SortOrder1_1_, internalc0_.CreatedAt as CreatedAt1_1_,
internalc0_.parent_id as parent6_1_1_, contentbas1_.ID as
ID0_0_, contentbas1_.SerializedData as Serializ3_0_0_,
contentbas1_.cb_type as cb2_0_
FROM ncommet_SimpleCMSPlus.dbo.Item internalc0_
left outer join ncommet_SimpleCMSPlus.dbo.ContentBase
contentbas1_
on internalc0_.ID=contentbas1_.ID
WHERE internalc0_.parent_id in (@p0, @p1, @p2);
@p0 = '15', @p1 = '14', @p2 = '3'
```

Σημείωση: Στο ερώτημα, αντλούνται επιπλέον και τα παιδιά της Σελίδας και του Portal (θα χρειαστούν από το SimpleCMSPlus).

- **Άντληση των θυγατρικών κόμβων του Φακέλου, που έχουν συγκεκριμένους τύπους κόμβων (περιεχομένου):**

```
select item0_.ID as ID1_, item0_.Name as Name1_,
item0_.ItemType as ItemType1_, item0_.SortOrder as
```

```
SortOrder1_, item0_.CreatedAt as CreatedAt1_,
item0_.parent_id as parent6_1_
from ncommet_SimpleCMSPlus.dbo.Item item0_
where (item0_.parent_id=@p0 )AND(item0_.ItemType IN(@p1 ,
@p2 , @p3))
order by  item0_.SortOrder;
@p0 = '15', @p1 = 'cont_item', @p2 = 'prsn_item', @p3 =
'sg_entry'
```

- Κατ' επέκταση υπάρχουν καταχωρήσεις για την άντληση δεδομένων σχετικά με τον κόμβο-ρίζα, τον κόμβο του portal, τις σελίδες του portal (ώστε να φανούν στο μενού επιλογής που βρίσκεται στα αριστερά της ιστοσελίδας), κ.ο.κ.

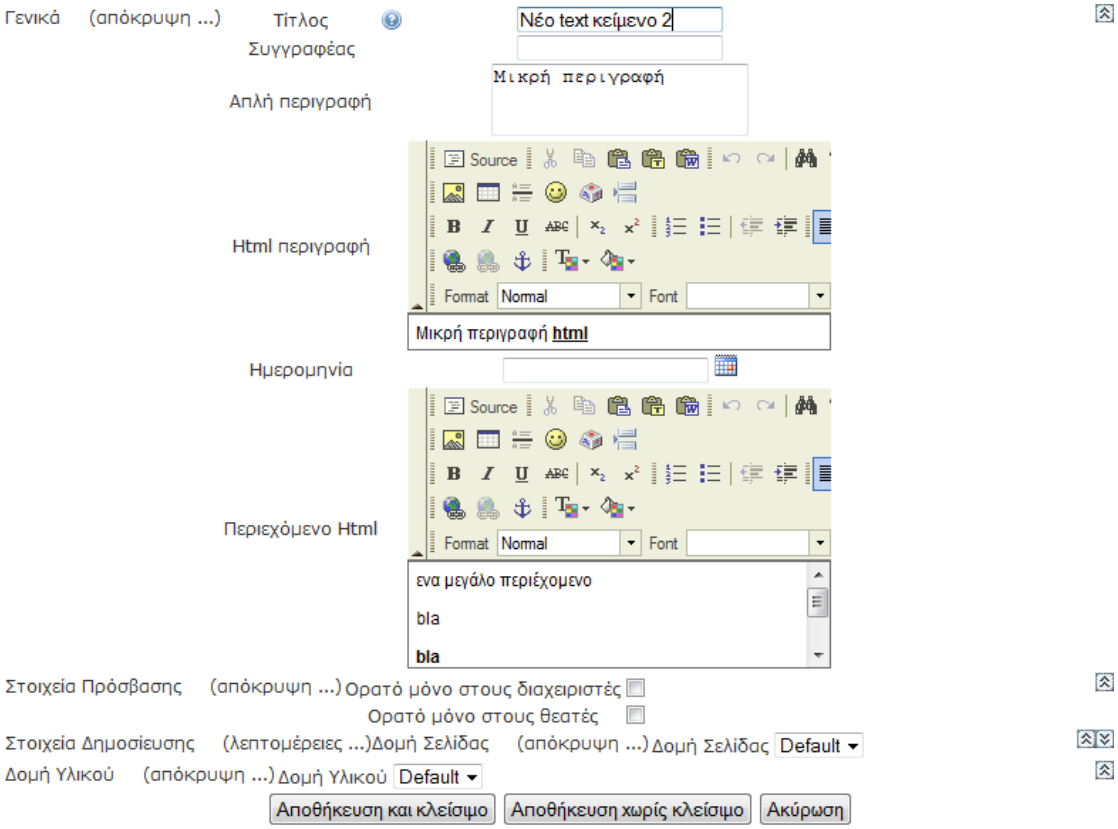
7.2.2 Σενάριο επεξεργασίας περιεχομένου

Επειδή θέλουμε να ελέγξουμε μόνο το NCommet, δεν θα μας απασχολήσουν οι κλάσεις περιεχομένου του SimpleCMSPlus, όπως η κλάση Html ή η κλάση WebLink. Η Υπηρεσία Αποθήκευσης του NCommet διαχειρίζεται όλα τα αντικείμενα περιεχομένου με ενιαίο τρόπο, όπως περιγράψαμε στην ενότητα 6.1.2.4 «Ρυθμίσεις απεικόνισης των κλάσεων περιεχομένου της εφαρμογής». Συγκεκριμένα, αποθηκεύει την XML σειριακοποιημένη μορφή του αντικείμενου περιεχομένου στη βάση δεδομένων.

Όσον αφορά τα αντικείμενα περιεχομένου, θα αρκεστούμε στο να επαληθεύσουμε ότι το NCommet μπορεί να αντλήσει και να τροποποιήσει σωστά ένα αντικείμενο περιεχομένου. Γι' αυτό το λόγο θα χρησιμοποιήσουμε μόνο την ιδιότητα Title των αντικειμένων περιεχομένου του SimpleCMSPlus, η οποία είναι κοινή σε όλα τα αντικείμενα περιεχομένου του SimpleCMSPlus.

Θα βασιστούμε στο παράδειγμα της προηγούμενης περίπτωσης (βλ. παραπάνω).

Θα επεξεργαστούμε την ιδιότητα Τίτλου του πρώτου κόμβου κάτω από τον Φάκελο Περιεχομένου «Υλικό του Μαθήματος». Αν επιλέξουμε το κουμπί επεξεργασίας του κόμβου με τίτλο «Νέο κείμενο», θα εμφανιστεί σε ένα νέο παράθυρο η σελίδα επεξεργασίας του Backoffice του SimpleCMSPlus.



Εικόνα 7-3 Σελίδα επεξεργασίας κόμβου του Backoffice

Αλλάζοντας τον τίτλο σε «Νέο text κείμενο 2» και πατώντας το κουμπί αποθήκευσης, το SimpleCMSPlus χρησιμοποιεί την Υπηρεσία Αποθήκευσης του NCommet για την αποθήκευση του τροποποιημένου κόμβου. Ο τροποποιημένος τίτλος φαίνεται σωστά στην ανανεωμένη ιστοσελίδα:

Εικόνα 7-4 Εμφάνιση τροποποιημένου τίτλου

Από το παραπάνω στιγμιότυπο οθόνης, επιβεβαιώνουμε ότι το NCommet τροποποιεί σωστά ένα αντικείμενο περιεχομένου.

Μάλιστα, για να επαληθεύσουμε την λειτουργία του NHibernate, μπορούμε να ελέγξουμε το αρχείο καταγραφής του Backoffice. Οι σχετικές καταχωρήσεις είναι οι εξής:

- **Άντληση του κόμβου και του αντικείμενου περιεχομένου του:**

```
SELECT item0_.ID as ID1_1_, item0_.Name as Name1_1_,
item0_.ItemType as ItemType1_1_, item0_.SortOrder as
SortOrder1_1_, item0_.CreatedAt as CreatedAt1_1_,
item0_.parent_id as parent6_1_1_, contentbas1_.ID as ID0_0_,
contentbas1_.SerializedData as Serializ3_0_0_,
contentbas1_.cb_type as cb2_0_
FROM ncommet_SimpleCMSPlus.dbo.Item item0_
left outer join ncommet_SimpleCMSPlus.dbo.ContentBase
contentbas1_
on item0_.ID=contentbas1_.ID
WHERE item0_.ID=@p0;
@p0 = '131'
```

- **Ενημέρωση του τροποποιημένου κόμβου:**

```
UPDATE ncommet_SimpleCMSPlus.dbo.ContentBase
SET SerializedData = @p0 WHERE ID = @p1;
@p0 = '
<?xml version="1.0" encoding="utf-16"?>
<Html xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<Title>Νέο text κείμενο 2</Title>
…
<DescriptionText>Μικρή περιγραφή</DescriptionText>
</Html>', @p1 = '131'
```

Αν επισκεφτούμε και τον πίνακα ContentBase στη βάση δεδομένων, μπορούμε να επιβεβαιώσουμε την εγγραφή του τροποποιημένου κόμβου:

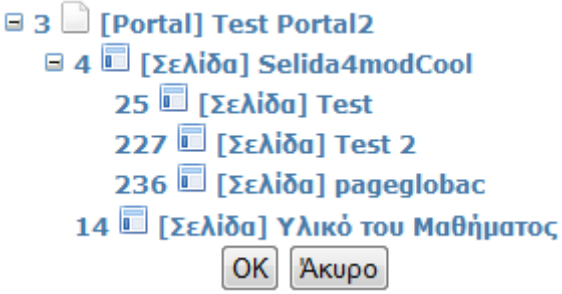
ID	**cb_type**	**SerializedData**
131	SimpleCMSPlus.BusinessModel.Html	<?xml version="1.0" encoding="utf-16"?><Html xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"><Title>Νέο text κείμενο 2</Title> … </Html>

Πίνακας 7-1 Περιεχόμενο του πίνακα ContentBase μετά το σενάριο επεξεργασίας περιεχομένου.

7.2.3 Σενάριο μετακίνησης κόμβου

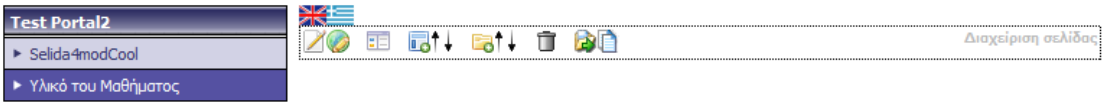
Θα βασιστούμε στο παράδειγμα της προηγούμενης περίπτωσης (βλ. παραπάνω).

Θα μετακινήσουμε τον Φάκελο Περιεχομένου σε μια άλλη Σελίδα του SimpleCMSPlus. Επιλέγουμε το κατάλληλο κουμπί «Μετακίνησης» από την εργαλειοθήκη διαχείρισης του Φακέλου. Εμφανίζεται η εξής σελίδα του Backoffice:

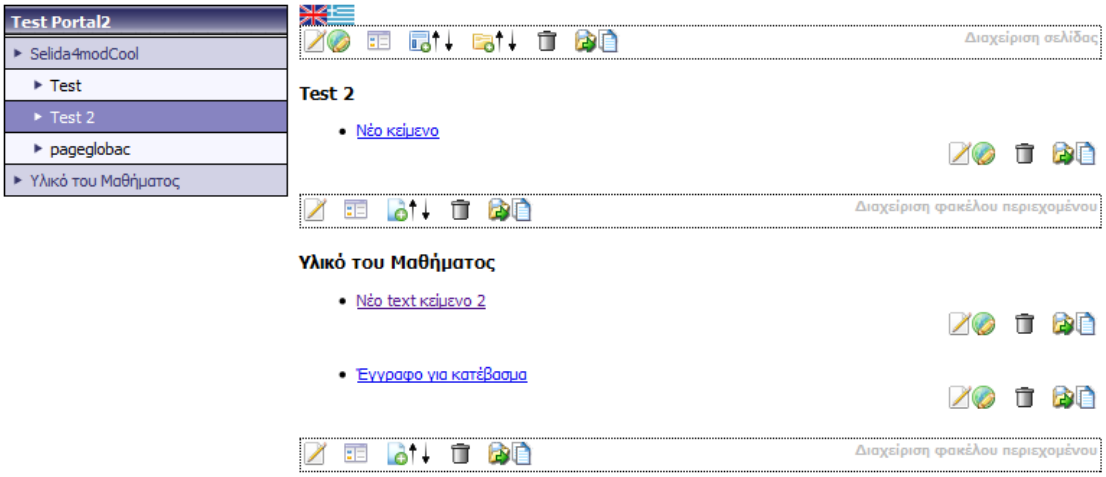


Εικόνα 7-5 Σελίδα μετακίνησης του Φακέλου

Επιλέγοντας τη σελίδα "Test 2" (κόμβος με id="227"), και πατώντας το κουμπί OK, η σελίδα «Υλικό του Μαθήματος» θα μείνει κενή, ενώ η σελίδα «Test 2» θα έχει επιπλέον τον φάκελο που μόλις μετακινήσαμε:



Εικόνα 7-6 Σελίδα «Υλικό του Μαθήματος» μετά την μετακίνηση του Φακέλου



Εικόνα 7-7 Σελίδα «Test 2» μετά την μετακίνηση του Φακέλου

Από τα παραπάνω στιγμιότυπα οθόνης, επιβεβαιώνουμε ότι το NCommet μετακινεί σωστά ένα υπόδενδρο. Μάλιστα, για να επαληθεύσουμε την λειτουργία του NHibernate, μπορούμε να ελέγξουμε το αρχείο καταγραφής του Backoffice. Οι σχετικές καταχωρήσεις είναι οι εξής:

- **Απόσπαση του κόμβου από τον πατέρα του:**

```
UPDATE ncommet_SimpleCMSPlus.dbo.Item
SET Name = @p0, ItemType = @p1, SortOrder = @p2, CreatedAt =
@p3, parent_id = @p4 WHERE ID = @p5;
```

```
@p0 = '', @p1 = 'cont_list', @p2 = '0',
@p3 = '3/31/2008 2:22:38 PM', @p4 = '', @p5 = '15'
```

- **Αλλαγή γονικού κόμβου:**

```
UPDATE ncommet_SimpleCMSPlus.dbo.Item
SET Name = @p0, ItemType = @p1, SortOrder = @p2, CreatedAt =
@p3, parent_id = @p4 WHERE ID = @p5;
@p0 = '', @p1 = 'cont_list', @p2 = '1',
@p3 = '3/31/2008 2:22:38 PM', @p4 = '227', @p5 = '15'
```

Αν επισκεφτούμε και τον πίνακα Item στη βάση δεδομένων, μπορούμε να επιβεβαιώσουμε την εγγραφή του τροποποιημένου κόμβου:

ID	Name	ItemType	SortOrder	CreatedAt	parent_id
227	el/test_portal/selida1mod5/test_2	page	5	2010-04-14 13:51:04.000	4
15		cont_list	1	2008-03-31 14:22:38.000	227
131	el/test_portal/yliko_toy_mauimatos/neo_keimeno-1	cont_item	0	2008-03-31 14:22:38.000	15
233	el/test_portal/yliko_toy_mauimatos/eggrafo_gia_katebasma	cont_item	1	2010-04-26 13:42:51.000	15

Πίνακας 7-2 Περιεχόμενο του πίνακα Item μετά τη μετακίνηση φακέλου.

Ο Φάκελος (κόμβος με id 15) έχει πλέον πατέρα την σελίδα “Test 2” (κόμβος με id 227). Τα δύο παιδιά του Φακέλου έμειναν ανέπαφα.

8

Επίλογος

Στο τελευταίο κεφάλαιο, συνοψίζεται η παρούσα διπλωματική εργασία και τα αποτελέσματα της. Παρουσιάζονται επιγραμματικά οι στόχοι που επετεύχθησαν. Τέλος πραγματοποιείται μία σύντομη αναφορά σε μελλοντικές επεκτάσεις που έχουν νόημα και ενδιαφέρον να υλοποιηθούν.

8.1 Σύνοψη και συμπεράσματα

Στην παρούσα διπλωματική εργασία αναπτύχθηκε ένα νέο επίπεδο διατήρησης δεδομένων για την πλατφόρμα NCCMS. Η πλατφόρμα NCCMS έχει αναπτυχθεί από το Εργαστήριο Συστημάτων Βάσεων Γνώσεων και Δεδομένων και φιλοξενεί μεγάλο αριθμό ιστοσελίδων μαθημάτων του Ε.Μ.Π. Οι καθηγητές μπορούν να χρησιμοποιήσουν την πλατφόρμα για την επεξεργασία του υλικού των ιστοσελίδων, την οργάνωση του υλικού και τη διαχείριση των χρηστών και των δικαιωμάτων πρόσβασης τους.

Όπως είναι φυσικό, το NCCMS αποτελεί μια μεγάλη και πολύπλοκη διαδικτυακή πλατφόρμα. Αρχικά, ασχοληθήκαμε με την κατανόηση της αρχιτεκτονικής του NCCMS, του μοντέλου δεδομένων του και του τρόπου λειτουργίας του. Το NCCMS χρησιμοποιεί δύο μεγάλες βιβλιοθήκες, το AbstractCMS και το NCommet. Το NCommet βρίσκεται στο τελευταίο επίπεδο της αρχιτεκτονικής και αποτελεί μια γενική βιβλιοθήκη διαχείρισης δενδρικών δομών. Η βιβλιοθήκη αυτή είναι υπεύθυνη για τη μοντελοποίηση και τη διατήρηση των δεδομένων του NCCMS.

Συνεχίσαμε αναλύοντας τη σχεδίαση της βιβλιοθήκης NCommet, η οποία αποτελείται από τον πυρήνα και από διάφορες «Υπηρεσίες» στις οποίες έχει διαμοιραστεί η λειτουργικότητα του. Σκοπός της διπλωματικής εργασίας ήταν η ανάπτυξη μιας νέας υλοποίησης της «Υπηρεσίας Αποθήκευσης» του NCommet, η οποία λειτουργεί ως το επίπεδο διατήρησης δεδομένων των εφαρμογών που το χρησιμοποιούν (όπως το NCCMS).

Για την υλοποίηση της «Υπηρεσίας Αποθήκευσης», δουλέψαμε πάνω σε ένα πρότυπο σύστημα ονόματι SimpleCMSPlus, το οποίο αποτελεί μια απλουστευμένη έκδοση του NCCMS, και δεν περιέχει υποσυστήματα και λειτουργίες οι οποίες δεν έχουν σχέση με τη διατήρηση δεδομένων. Οι αλλαγές που κάναμε στον κώδικα του SimpleCMSPlus και του AbstractCMS ήταν ελάχιστες και επικεντρωθήκαμε στις αλλαγές μόνο μέσα στο NCommet. Αυτό γιατί το NCommet αποτελεί μια βιβλιοθήκη και οι αλλαγές πρέπει να απευθύνονται σε οποιαδήποτε εφαρμογή τη χρησιμοποιεί.

Η υλοποίηση της «Υπηρεσίας Αποθήκευσης» είχε σκοπό να είναι ευέλικτη, εύκολη στη συντήρηση, και να εξυπηρετεί όλες τις εφαρμογές που χρησιμοποιούν το NCommet. Για το λόγο αυτό, χρησιμοποιήσαμε σύγχρονες τεχνικές αντικειμενο-σχεσιακής απεικόνισης.

Το ORM εργαλείο που χρησιμοποιήσαμε ήταν το NHibernate. Για τη χρησιμοποίηση του, ήταν αναγκαία αρχικά η κατανόηση του τρόπου λειτουργίας του και των ρυθμίσεων του. Έπειτα, συντάξαμε τα κατάλληλα αρχεία ρυθμίσεων απεικόνισης και φροντίσαμε να είναι εύκολη η αποθήκευση των αντικειμένων περιεχομένου οποιασδήποτε εφαρμογής χρησιμοποιεί το NCommet. Το τελευταίο επετεύχθη χρησιμοποιώντας τεχνικές XML Σειριακοποίησης.

Η υλοποίηση της «Υπηρεσίας Αποθήκευσης» είναι αρκετά ευέλικτη ώστε να μπορεί να χρησιμοποιηθεί με ευκολία από καινούργιες εφαρμογές για τη διαχείριση της διατήρησης των αντικειμένων τους. Επιπρόσθετα, ο κώδικας που αναπτύχθηκε είναι ευκολότερος στην συντήρηση, αφού είναι πλέον αποσυνδεδεμένος από τη βάση δεδομένων και χρησιμοποιεί το NHibernate για την ανάκτηση και την αποθήκευση των αντικειμένων. Έτσι επικεντρώνεται περισσότερο στις επιχειρηματικές ανάγκες της εφαρμογής παρά στη διαχείριση της βάσης δεδομένων μέσω προσαρμοσμένων SQL ερωτημάτων.

Τα κύρια πλεονεκτήματα που προσφέρει η υλοποίηση της νέας «Υπηρεσίας Αποθήκευσης» του NCommet στο περιβάλλον NCCMS, όσον αφορά τη διατήρηση δεδομένων, είναι η μείωση του χρόνου ανάπτυξης λογισμικού, η μείωση του κώδικα, και η ευκολότερη συντήρηση του λογισμικού.

8.2 Μελλοντικές επεκτάσεις

8.2.1 Βελτιώσεις στην υλοποίηση με NHibernate

Μια βασική επέκταση που μπορεί να γίνει στο σύστημα, είναι μια διεξοδική και αναλυτική μελέτη της αποδοτικότητας των SQL ερωτημάτων που παράγονται αυτόματα από το NHibernate για την ανάκτηση δεδομένων του NCCMS. Σκοπός θα είναι η αύξηση της ταχύτητας της ανταπόκρισης των ιστοσελίδων του NCCMS, βελτιστοποιώντας τις τεχνικές caching του NHibernate ή χρησιμοποιώντας προσαρμοσμένες τεχνικές caching.

Για την αύξηση της ταχύτητας, μια άλλη δυνατότητα είναι η ανάπτυξη ενός τρόπου για lazy loading των ιδιοτήτων των αντικειμένων περιεχομένου. Αυτή τη στιγμή, τα αντικείμενα περιεχομένου αποθηκεύονται και ανακτώνται ολόκληρα σε XML σειριακοποιημένη (serialized) μορφή. Μια πιο προχωρημένη λύση θα ήταν η αποθήκευση της XML σε στήλη τύπου «xml» και όχι «varchar» του SQL Server. Έπειτα, πρέπει να ερευνηθεί αν μπορεί να δημιουργηθεί κατάλληλος προσαρμοσμένος τύπος στο NHibernate ο οποίος, κατά την ανάκτηση του αντικειμένου περιεχομένου δεν θα ανακτά όλη την XML, αλλά θα λειτουργεί ως proxy. Δηλαδή, μόνο κατά την ανάκτηση κάποιας ιδιότητας θα εκτελούνται τα κατάλληλα ερωτήματα προς τη βάση δεδομένων για την ανάκτηση της. Με αυτόν τον τρόπο, θα αποφευχθεί η φόρτωση της υπόλοιπης XML που ίσως να μην χρειάζεται να διαβάσει η εφαρμογή.

Άλλη μια επέκταση μπορεί να πραγματοποιηθεί στο θέμα της αυτόματης ρύθμισης του NHibernate και των ρυθμίσεων απεικόνισης των αντικειμένων περιεχομένου της εφαρμογής. Αυτή τη στιγμή, η εφαρμογή πρέπει να συντάξει ένα αρχείο ρυθμίσεων απεικόνισης για κάθε αντικείμενο περιεχομένου της που να το ορίζει ως <subclass> της ContentBase του NCommet (βλ. 6.1.3.3). Εφόσον αυτές οι ρυθμίσεις είναι τετριμμένες, μια νέα επέκταση θα μπορούσε να δημιουργεί αυτόματα τις απαραίτητες ρυθμίσεις απεικόνισης των αντικειμένων περιεχομένου της εφαρμογής για το NHibernate, χρησιμοποιώντας το σύστημα Reflection του .NET Framework.

8.2.2 Εναλλακτικό ORM εργαλείο

Στην περίοδο που εκπονήθηκε η παρούσα διπλωματική εργασία, το NHibernate ήταν το πιο αξιόπιστο και ευρέως χρησιμοποιούμενο εργαλείο αντικειμενο-σχεσιακής απεικόνισης. Το επόμενο πιο κυρίαρχο εργαλείο ήταν το ADO.NET Entity Framework που εξέδωσε η Microsoft τον Αύγουστο του 2008 με το .NET Framework 3.5 SP1. Περιγράψαμε τις βασικές αρχές του στην ενότητα 3.2 «ADO.NET Entity Framework (EF)». Όμως το Entity Framework βρισκόταν ακόμα στην πρώτη του έκδοση και είχε υιοθετηθεί από περιορισμένο

αριθμό προγραμματιστών. Ήταν κοινά αποδεκτό ότι το NHibernate ήταν πιο ώριμο ως προϊόν και διέθετε περισσότερα ORM χαρακτηριστικά από το Entity Framework.

Τον Απρίλιο του 2010, η Microsoft εξέδωσε τη δεύτερη έκδοση του Entity Framework μαζί με το .NET Framework 4.0. Η δεύτερη έκδοση διορθώνει πολλά από τα αρνητικά χαρακτηριστικά της πρώτης έκδοσης, για τα οποία υπήρχε αρκετή κριτική. Μερικές βελτιώσεις που προσφέρει είναι οι εξής [Mic10d]:

- Στην πρώτη έκδοση υπήρχε η δυνατότητα παραγωγής του Entity Data Model (EDM) μόνο από τη βάση δεδομένων. Στη δεύτερη έκδοση προστέθηκε η αντίστροφη δυνατότητα, δηλαδή η παραγωγή του σχεσιακού μοντέλου από το EDM.
- Το EF επιτρέπει την αποθήκευση απλών αντικειμένων της εφαρμογής. Τα αντικείμενα δεν χρειάζεται να υλοποιήσουν καμία διεπαφή που να έχει σχέση με τη διατήρηση των δεδομένων τους. Βεβαίως υπάρχει ακόμα η ανάγκη να οριστούν τα μετα-δεδομένα του EDM.
- Υποστήριξη για αυτόματο lazy loading. Στην πρώτη έκδοση δεν υπήρχε αυτόματο lazy loading: Όταν ο προγραμματιστής ανακτούσε στιγμιότυπα μιας κλάσης, έπρεπε να φορτώσει χειρονακτικά τυχόν σχετιζόμενες κλάσεις. Αυτό το πρόβλημα ήταν και το σπουδαιότερο για τους περισσότερους χρήστες της πρώτης έκδοσης του EF. Ήταν και το κεντρικό σημείο σύγκρισης με άλλα ORM εργαλεία όπως το NHibernate.
- Έχοντας ορίσει το EDM, ο προγραμματιστής μπορεί να χρησιμοποιήσει τα καινούργια χαρακτηριστικά της δεύτερης έκδοσης του EF για να δημιουργήσει τους απαραίτητους πίνακες στη σχεσιακή βάση δεδομένων.
- Βελτιστοποίηση των SQL ερωτημάτων που παράγονται αυτόματα.

Η δεύτερη έκδοση του EF έχει βελτιωθεί αρκετά σε σχέση με την πρώτη. Μάλιστα, όλο και περισσότεροι μηχανικοί λογισμικού το χρησιμοποιούν για τις εφαρμογές τους.

Το ΕΣΒΓΔ, το οποίο είναι υπεύθυνο για το NCCMS, πιστεύει ότι θα ήταν αρκετά σημαντική η χρησιμοποίηση του EF ως ORM εργαλείο για τη διατήρηση δεδομένων του NCCMS. Μάλιστα, αφού η παρούσα διπλωματική εργασία εφάρμοσε την αντικειμενο-σχεσιακή απεικόνιση για το NCCMS, η μετάβαση στο EF θα είναι αρκετά ευκολότερη.

9

Βιβλιογραφία

[Amb10] Scott Ambler. (2010, Ιούνιος) Mapping Objects to Relational Databases: O/R Mapping In Detail. [Online]. http://www.agiledata.org/essays/mappingObjects.html

[Apa10] Apache Software Foundation. (2010, Ιούνιος) About Apache log4net. [Online]. http://logging.apache.org/log4net/

[Kua09] Pierre Henri Kuaté, Tobin Harris, Christian Bauer, and Gavin King, *NHibernate in Action*, Cynthia Kane, Ed. Greenwich, United States of America: Manning Publications Co., 2009.

[Ler09] Julia Lerman, *Programming Entity Framework*, 1st ed., John Osborn and Audrey Doyle, Eds. Sebastopol, United States of America: O'Reilly Media, Inc., 2009.

[Mic06] Microsoft Corp. (2006, Ιούνιος) The ADO.NET Entity Framework Overview. [Online]. http://msdn.microsoft.com/en-us/library/aa697427(VS.80).aspx

[Mic10a] Microsoft Corporation. (2010, Ιούνιος) MSDN - Introducing XML Serialization. [Online]. http://msdn.microsoft.com/en-us/library/182eeyhh%28v=VS.100%29.aspx

[Mic10b] Microsoft Corporation. (2010, Ιούνιος) MSDN - How To: Use Role Manager in ASP.NET 2.0. [Online]. http://msdn.microsoft.com/en-us/library/ff647401.aspx

[Mic10c] Microsoft Corp. (2010, Ιούνιος) Ten advantages of an ORM (Object Relational

Mapper). [Online]. http://blogs.msdn.com/b/gblock/archive/2006/10/26/ten-advantages-of-an-orm.aspx

[Mic10d] Microsoft Corp. (2010, Απρίλιος) Announcing the release of Entity Framework 4. [Online]. http://blogs.msdn.com/b/efdesign/archive/2010/04/12/announcing-the-release-of-entity-framework-4.aspx

[NHi10] NHibernate. (2010, Ιούνιος) NHibernate Reference Documentation. [Online]. http://nhforge.org/doc/nh/en/index.html

[Wik10a] Wikipedia. (2010, Ιούνιος).NET Framework. [Online]. http://en.wikipedia.org/wiki/.NET_Framework

[Wik10b] Wikipedia. (2010, Ιούνιος) Microsoft SQL Server. [Online]. http://en.wikipedia.org/wiki/Microsoft_SQL_Server

[Wik10c] Wikipedia. (2010, Ιούνιος) C Sharp (programming language). [Online]. http://en.wikipedia.org/wiki/C_Sharp_%28programming_language%29

[Wik10d] Wikipedia. (2010, Ιούνιος) ASP.NET. [Online]. http://en.wikipedia.org/wiki/ASP.NET

[Wik10e] Wikipedia. (2010, Ιούνιος) Internet Information Services. [Online]. http://en.wikipedia.org/wiki/Internet_Information_Services

[Wik10f] Wikipedia. (2010, Ιούνιος) Object-relational impedance mismatch. [Online]. http://en.wikipedia.org/wiki/Object-relational_impedance_mismatch

[Wik10g] Wikipedia. (2010, Ιούνιος) Object-relational mapping. [Online]. http://en.wikipedia.org/wiki/Object-relational_mapping

[Wik10h] Wikipedia. (2010, Ιούνιος) Data access layer. [Online]. http://en.wikipedia.org/wiki/Data_access_layer

[Wik10i] Wikipedia. (2010, Ιούνιος) ADO.NET Entity Framework. [Online]. http://en.wikipedia.org/wiki/ADO.NET_Entity_Framework

[Wik10j] Wikipedia. (2010, Ιούνιος) NHibernate. [Online]. http://en.wikipedia.org/wiki/NHibernate

[Wik10k] Wikipedia. (2010, Ιούνιος) Persistence (computer science). [Online]. http://en.wikipedia.org/wiki/Persistence_%28computer_science%29