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

Ανάπτυξη εφαρμογής διαχείρισης μαθημάτων με χρήση τεχνολογιών Java 2 Enterprise Edition (J2EE)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Νικόλαος Ι. Κουρής

Επιβλέπων : Νικόλαος Παπασπύρου
Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2008

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

Ανάπτυξη εφαρμογής διαχείρισης μαθημάτων με χρήση τεχνολογιών Java 2 Enterprise Edition (J2EE)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Νικόλαος Ι. Κουρής

Επιβλέπων : Νικόλαος Παπασπύρου
Επίκουρος Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 24η Νοεμβρίου 2008.

.....	.....	.....
Νικόλαος Παπασπύρου	Ανδρέας Παπασαλούρος	Κώστας Κοντογιάννης
Επίκουρος Καθηγητής Ε.Μ.Π	Λέκτορας Πανεπιστημίου Αιγαίου	Αναπληρωτής Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2008

.....

Νικόλαος Ι. Κουρής

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

Περίληψη

Το αντικείμενο αυτής της διπλωματικής εργασίας είναι η ανάπτυξη μιας εφαρμογής για διαχείριση μαθημάτων. Η υλοποίηση έγινε με σκοπό την διερεύνηση των πλεονεκτημάτων και μειονεκτημάτων που παρέχει η ανάπτυξη μιας εφαρμογής σε Java στην πλατφόρμα ανάπτυξης εφαρμογών, Java 2 Enterprise Edition (J2EE) την οποία και χρησιμοποιήσαμε.

Η πλατφόρμα J2EE, για αρκετά χρόνια αποτελούσε «πονοκέφαλο» για τους ανθρώπους που ασχολούνταν με την ανάπτυξη επιχειρησιακών εφαρμογών και όχι άδικα αφού παρουσίαζε αρκετές δυσκαμψίες και μεγάλη πολυπλοκότητα. Με την εισαγωγή τα τελευταία χρόνια της καινούριας έκδοσης της τεχνολογίας Enterprise Java Bean (EJB), γνωστής ως έκδοσης EJB 3 ξεπερνιούνται αρκετά προβλήματα του παρελθόντος και διευκολύνεται κατά πολύ ή ανάπτυξη εφαρμογών με την πλατφόρμα J2EE.

Η παρούσα διπλωματική αναφέρεται στη σχεδίαση και υλοποίηση μιας εφαρμογής χρησιμοποιώντας τεχνολογίες J2EE (servlet, xml, EJB κτλ.). Κύριος σκοπός της ανάπτυξης, ήταν η δημιουργία μιας εφαρμογής Ιστού (Web application) που θα χρησιμοποιεί μια βάση δεδομένων πραγματικών απαιτήσεων, που θα υλοποιείται κάνοντας χρήση της τεχνολογίας Enterprise Java Bean (EJB3). Η σχεδίαση της εφαρμογής, και κατά ακολουθία της βάσης δεδομένων, έγινε με γνώμονα την παρουσίαση όσο το δυνατόν περισσοτέρων χαρακτηριστικών που συναντά κανείς σε μια σχεσιακή βάση δεδομένων, ώστε μέσω της υλοποίησης να αναδειχθεί καταφανώς το πλεονέκτημα της εξολοκλήρου αντιστοίχησης της αντικειμενοστραφούς σχεδίασης των κλάσεων της Java της εφαρμογής με τη σχεδίαση στο επίπεδο οντοτήτων-συσχετίσεων της σχεσιακής βάσης δεδομένων, που παρέχει η υλοποίηση με την προδιαγραφή EJB3.

Η εργασία αυτή προσπαθεί να δείξει και να βγάλει συμπεράσματα κατά πόσο είναι αποδοτική, η υλοποίηση μιας επιχειρησιακής εφαρμογής, ακολουθώντας κανείς την προδιαγραφή J2EE μαζί με ότι καινούριο αυτή προτείνει.

Λέξεις Κλειδιά: Enterprise Java Bean (EJB3),, Java 2 Enterprise Edition J2EE, Server Side εφαρμογή.

Abstract

The object of this thesis is the development of an application for course management. The implementation aimed at investigating benefits and drawbacks of Java 2 Enterprise Edition (J2EE) platform, which was used for the development of the application.

J2EE was not widely adopted by the enterprise application development community due to its rigidity and complexity. With the advent of the new version of the Enterprise Java Bean (EJB3) most of the problems of previous versions are overcome and application development becomes easier.

This thesis refers to the design and development of an application using particular J2EE technologies such as servlets and EJB. Main purpose of development was a web application using a database covering realistic requirements. Object to relational mapping feature of EJB3 was used in order to synchronize database data with business objects developed in Java.

The main objective of this thesis is to inquire whether the adoption of J2EE for enterprise application development is efficient from a development point of view.

Keywords: Enterprise Java Bean (EJB3), Java 2 Enterprise Edition J2EE, server side application.

Πρόλογος

Η παρούσα διπλωματική εκπονήθηκε στη σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου το ακαδημαϊκό έτος 2007-2008, στα πλαίσια των δραστηριοτήτων του Τομέα Τεχνολογίας Πληροφορικής και Υπολογιστών, υπό την επίβλεψη του Επίκουρου Καθηγητή κου Νικόλαου Παπασπύρου, τον οποίο θα ήθελα να ευχαριστήσω για τη δυνατότητα που μου έδωσε να ασχοληθώ με το συγκεκριμένο θέμα, καθώς και για το ενδιαφέρον του και την βοήθεια που μου παρείχε.

Ιδιαίτερα θα ήθελα να ευχαριστήσω τον Λέκτορα του Πανεπιστημίου Αγαίου του τμήματος Μαθηματικών στο Καρλόβασι της Σάμου, Ανδρέα Παπασαλούρο. Αρχικώς, που με παρότρυνε να εργαστώ σε ένα τόσο σύγχρονο και ταυτόχρονα ενδιαφέρον αντικείμενο. Επίσης, για την καθοδήγηση, την υπομονή, τη συνδρομή, τις παρατηρήσεις του και τη στήριξη που μου παρείχε καθόλο το διάστημα της εκπόνησης αυτής της εργασίας, και τέλος για την άριστη συνεργασία που είχαμε, παρόλο που η επικοινωνία μας γινόταν από απόσταση, Αθήνα – Σάμος.

Οφείλω να εκφράσω ιδιαιτέρως τις ευχαριστίες μου καθώς και την ευγνωμοσύνη μου προς την οικογένεια μου, για την αμέριστη συμπαράσταση και υποστήριξη που μου παρείχε κατά τη διάρκεια των σπουδών μου. Επίσης στους φίλους μου, να εκφράσω την ευγνωμοσύνη μου , που με την παρουσία τους και την αγάπη τους, αποτελούν καθημερινό στήριγμα για μένα .

Τέλος, θα ήθελα να ευχαριστήσω τους φίλους και συμφοιτητές μου, για όλα αυτά που μοιραστήκαμε, όλα αυτά τα χρόνια στην κοινή μας διαδρομή.

Κουρής Νίκος

Νοέμβρης 2008

Περιεχόμενα

1

Εισαγωγή

Στο κεφάλαιο αυτό αναλύεται το αντικείμενο της παρούσας διπλωματικής εργασίας και επίσης περιγράφεται η οργάνωση του τόμου της διπλωματικής αυτής όπως ακολουθεί στα επόμενα κεφάλαια.

1.1 Αντικείμενο της διπλωματικής

Η παρούσα διπλωματική αναφέρεται στη σχεδίαση και υλοποίηση μιας εφαρμογής χρησιμοποιώντας τεχνολογίες J2EE (servlet, xml, EJB κτλ.). Κύριος σκοπός της ανάπτυξης, ήταν η δημιουργία μιας εφαρμογής Ιστού (Web application) που θα χρησιμοποιεί μια βάση δεδομένων πραγματικών απαιτήσεων, η οποία θα υλοποιείται κάνοντας χρήση της τεχνολογίας Enterprise Java Bean (EJB3). Η σχεδίαση της εφαρμογής, και κατά ακολουθία της βάσης δεδομένων, έγινε με γνώμονα την παρουσίαση όσο το δυνατόν περισσοτέρων χαρακτηριστικών που συναντά κανείς σε μια σχεσιακή βάση δεδομένων, ώστε μέσω της υλοποίησης να αναδειχθεί καταφανώς, το πλεονέκτημα της εξολοκλήρου αντιστοίχησης, της αντικειμενοστραφούς σχεδίασης των κλάσεων Java της εφαρμογής με τη σχεδίαση στο επίπεδο οντοτήτων-συσχετίσεων της σχεσιακής βάσης δεδομένων που παρέχει η υλοποίηση με την προδιαγραφή EJB3.

1.2 Οργάνωση του τόμου

Ο τόμος αυτής της διπλωματικής εργασίας αποτελείται 10 κεφάλαια, από το κεφάλαιο 1 της εισαγωγής, από τα επόμενα 8 κεφάλαια τα οποία συνιστούν τα δύο μέρη του τόμου, και το κεφάλαιο 10 της βιβλιογραφίας.

Στο Α΄ μέρος, που αποτελείται από τα κεφάλαια 2 και 3, παρουσιάζονται οι τεχνολογίες J2EE και EJB 3, εστιάζοντας σε αυτές, περισσότερο στα σημεία που χρησιμοποιήθηκαν στο

κατασκευαστικό μέρος της διπλωματικής. Στο Β΄ μέρος , που αποτελείται από τα κεφάλαια 4, 5, 6, 7, 8 και 9, παρουσιάζονται στοιχεία της ανάπτυξης (σχεδιασμού-υλοποίησης) της εφαρμογής : οι πλατφόρμες ανάπτυξης και εκτέλεσης, οι ρυθμίσεις εγκατάστασης, τα προγραμματιστικά περιβάλλοντα και τα σχεδιαστικά διαγράμματα. Επίσης, γίνεται μια μικρή παρουσίαση της λειτουργίας της εφαρμογής, καθώς εξάγονται και κάποια γενικά συμπεράσματα από την διαδικασία ανάπτυξης της εφαρμογής .

Το πρώτο κεφάλαιο είναι η εισαγωγή η οποία περιγράφει το αντικείμενο της διπλωματικής, και επίσης αναφέρονται κάποιοι στόχοι της εφαρμογής καθώς και η χρήση για την οποία προορίζεται η εφαρμογή.

Στο δεύτερο κεφάλαιο γίνεται μια παρουσίαση της πλατφόρμας ανάπτυξης Java 2 Enterprise Edition (J2EE), όπου αρχικά αναφέρεται η ανάγκη ανάπτυξης της συγκεκριμένη πλατφόρμας. Στη συνέχεια παρουσιάζεται η αρχιτεκτονική της και ο τρόπος λειτουργίας της, καθώς επίσης το πλήθος των τεχνολογιών που χρησιμοποιεί και τα είδη των εφαρμογών που μπορούν να αναπτυχθούν με αυτή. Τέλος, αναφέρονται όλοι οι λόγοι που την καθιστούν ελκυστική για ανάπτυξη εφαρμογών.

Στο τρίτο κεφάλαιο γίνεται μια λεπτομερής παρουσίαση των χρησιμοποιούμενων τεχνολογιών και EJB 3 και Java Persistence API. Αναφέρονται τα χαρακτηριστικά, ο τρόπος χρήσης και τα πλεονεκτήματα που προκύπτουν από τη χρήση τους.

Στο τέταρτο κεφάλαιο γίνεται μια μικρή αναφορά της εφαρμογής διαχείρισης μαθημάτων: SchoolApplication που αναπτύξαμε, όπου αναφέρονται οι στόχοι της ανάπτυξης καθώς και τι παρέχει η εφαρμογή από πλευράς χρήσης της.

Στο πέμπτο κεφάλαιο παρουσιάζονται, η ανάλυση απαιτήσεων της εφαρμογής με τη μορφή διαγραμμάτων περιπτώσεων χρήσης, καθώς και το διάγραμμα οντοτήτων.

Στο έκτο κεφάλαιο παρουσιάζεται η ανάπτυξη της εφαρμογής καθώς και τα διαγράμματα, κλάσεων, οντοτήτων – συσχετίσεων και σχεσιακό για την βάση δεδομένων το παραταξιακό , τα ακολουθιακά και το πλοήγησης για την περιγραφή της λειτουργίας της εφαρμογής.

Στο έβδομο κεφάλαιο γίνεται μια περιγραφή της υλοποίησης, δηλαδή της εγκατάστασης και της σύνδεσης των διαφόρων πλατφόρμων που χρησιμοποιήσαμε.

Στο όγδοο κεφάλαιο γίνεται η παρουσίαση της εφαρμογής , με εικόνες από τη χρήση της.

Τέλος, στο ένατο κεφάλαιο, εξάγονται κάποια γενικά συμπεράσματα, τα οποία έχουν να κάνουν, με τη χρήση τεχνολογιών J2EE κατά τη διαδικασία της ανάπτυξης, σε εφαρμογές όπως αυτή που αναπτύξαμε.

Μέρος Α΄

2

Η πλατφόρμα ανάπτυξης Java 2 Enterprise Edition (J2EE)

2.1 Η ανάγκη για μια πλατφόρμα ανάπτυξης επιχειρησιακών εφαρμογών ανεξάρτητη από δεσμεύσεις

Η J2EE (Java 2 Enterprise Edition) είναι μια πλατφόρμα για την εκτέλεση εφαρμογών εξυπηρετητή (server side) που είναι γραμμένες σε γλώσσα Java. Πριν την J2EE, οι εφαρμογές αυτές γραφόντουσαν κάνοντας χρήση των ειδικών προγραμματιστικών διεπαφών API (Application Programming Interface) των κατασκευαστών λογισμικού. Ό κάθε κατασκευαστής είχε τα δικά του μοναδικά API και εφάρμοζε τις δικές του αρχιτεκτονικές κατά την ανάπτυξη εφαρμογών εξυπηρετητή με γλώσσα Java . Αποτέλεσμα της παραπάνω κατάστασης ήταν η απαίτηση, για γνώση μεγάλου όγκου πληροφορίας και ικανότητας προγραμματισμού για κάθε API από τους προγραμματιστές και αρχιτέκτονες, και τελικά υψηλότερο κόστος ανάπτυξης εφαρμογών. Επακόλουθο, ήταν η διαίρεση της κοινότητας και η απομόνωση όσων έκαναν ανάπτυξη τότε εφαρμογών σε γλώσσα Java. Η περίοδος εκείνη χαρακτηρίστηκε ως αρνητική , αφού δεν προώθησε το κλίμα για την ανάπτυξη εφαρμογών και έκανε πολύ δύσκολο το κτίσιμο σοβαρών επιχειρησιακών (enterprise) εφαρμογών με γλώσσα Java.

Τη λύση στην παραπάνω κατάσταση την έφερε η εισαγωγή της αρχιτεκτονικής J2EE, που δεν είναι τίποτα άλλο από μια πλατφόρμα ανάπτυξης Java κατανεμημένων επιχειρησιακών εφαρμογών, η οποία αναπτύχθηκε από τη Sun Microsystems. Η πλατφόρμα αυτή έχει υιοθετηθεί σήμερα, από τους περισσότερους κατασκευαστές λογισμικού μιας και ήταν προϊόν κοινής τους συμφωνίας και προτυποποίησης των ίδιων τους των API.

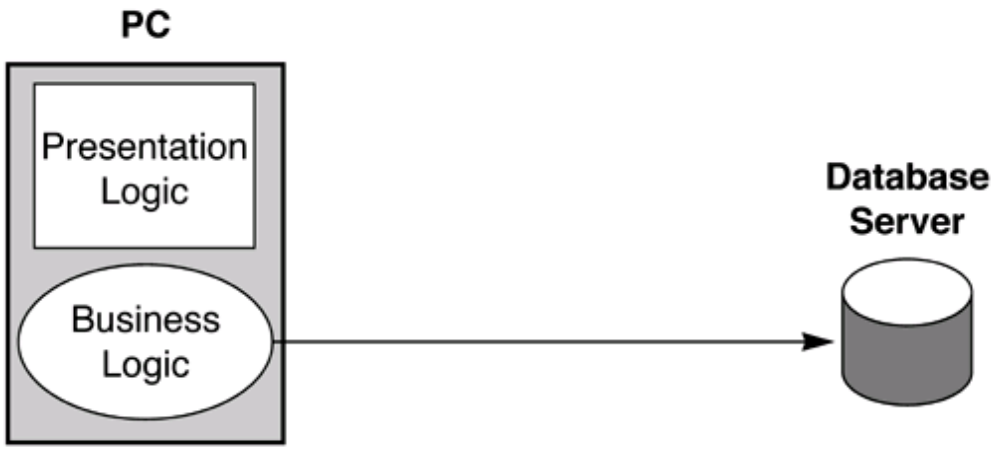
2.2 Γιατί οι εφαρμογές J2EE είναι επιθυμητές ;

2.2.1 Η αρχιτεκτονική πελάτη - εξυπηρετητή (client - server)

Στις αρχές του 1990, τα Πληροφοριακά Συστήματα (Information Systems) συχνά χρησιμοποιούσαν το μοντέλο αρχιτεκτονικής πελάτη - εξυπηρετητή (client - server). Η διεπαφή (interface) του χρήστη της εφαρμογής συνήθως έτρεχε σε έναν επιτραπέζιο υπολογιστή που βρίσκονταν στην πλευρά του πελάτη. Τα επιχειρησιακά δεδομένα που προσπελαζόντουσαν από τον πελάτη - εξυπηρετητή βρίσκονταν στη βάση δεδομένων και προσφερόντουσαν μέσω του εξυπηρετητή.

Η παραπάνω προσέγγιση αρχικά υποσχέθηκε μεγάλη ανάπτυξη και ευελιξία. Η μετέπειτα εμπειρία έδειξε μολαταύτα, ότι το κτίσιμο και η συντήρηση ενός ευέλικτου κατανεμημένου συστήματος είναι δύσκολο να γίνει χρησιμοποιώντας το μοντέλο πελάτη - εξυπηρετητή. Για παράδειγμα το τμήμα της επιχειρησιακής λογικής (business logic), αυτό που έχει να κάνει με τον κύριο μηχανισμό επεξεργασίας της εφαρμογής, ήταν τότε στην πλευρά του πελάτη. Κάθε φορά που το κομμάτι της επιχειρησιακής λογικής (business logic) της εφαρμογής χρειαζόταν τροποποίηση , η αναθεωρημένη έκδοση της εφαρμογής έπρεπε να εγκατασταθεί σε κάθε μηχανή πελάτη της επιχείρησης. Η συντήρηση εκείνα τα χρόνια ήταν ένας εφιάλτης. Επίσης, αυτές οι εφαρμογές έπρεπε να διαχειρίζονται τις διάφορες δοσοληψίες (transactions) με την μεγαλύτερη δυνατή ασφάλεια, και να επεξεργάζονται τα δεδομένα με τρόπο αποδοτικό ενώ ταυτόχρονα να παρέχουν και ένα ελκυστικό, εύκολα κατανοητό, περιβάλλον αλληλεπίδρασης στους χρήστες.

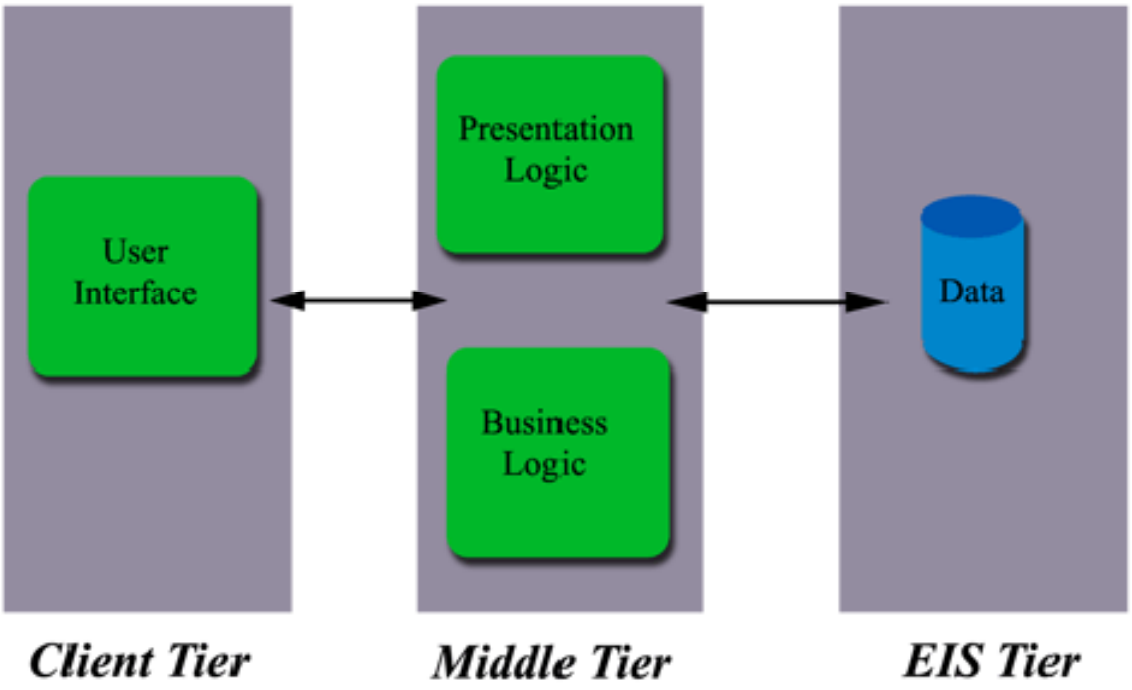
Λίγοι είναι οι προγραμματιστές οι οποίοι έχουν ταλέντο σε όλα αυτά τα αντικείμενα. Ενώ το πρότυπο πελάτη - εξυπηρετητή είναι επαρκές για κάποια περιβάλλοντα γίνεται φανερό ότι δεν είναι δυνατόν να ικανοποιήσει τις σύγχρονες απαιτήσεις στο πεδίο των επιχειρησιακών εφαρμογών.



Σχήμα 2.1 Αρχιτεκτονική εφαρμογής δύο στρωμάτων (client – server)

2.2.2 Το πολυστρωματικό πρότυπο αρχιτεκτονικής των εφαρμογών

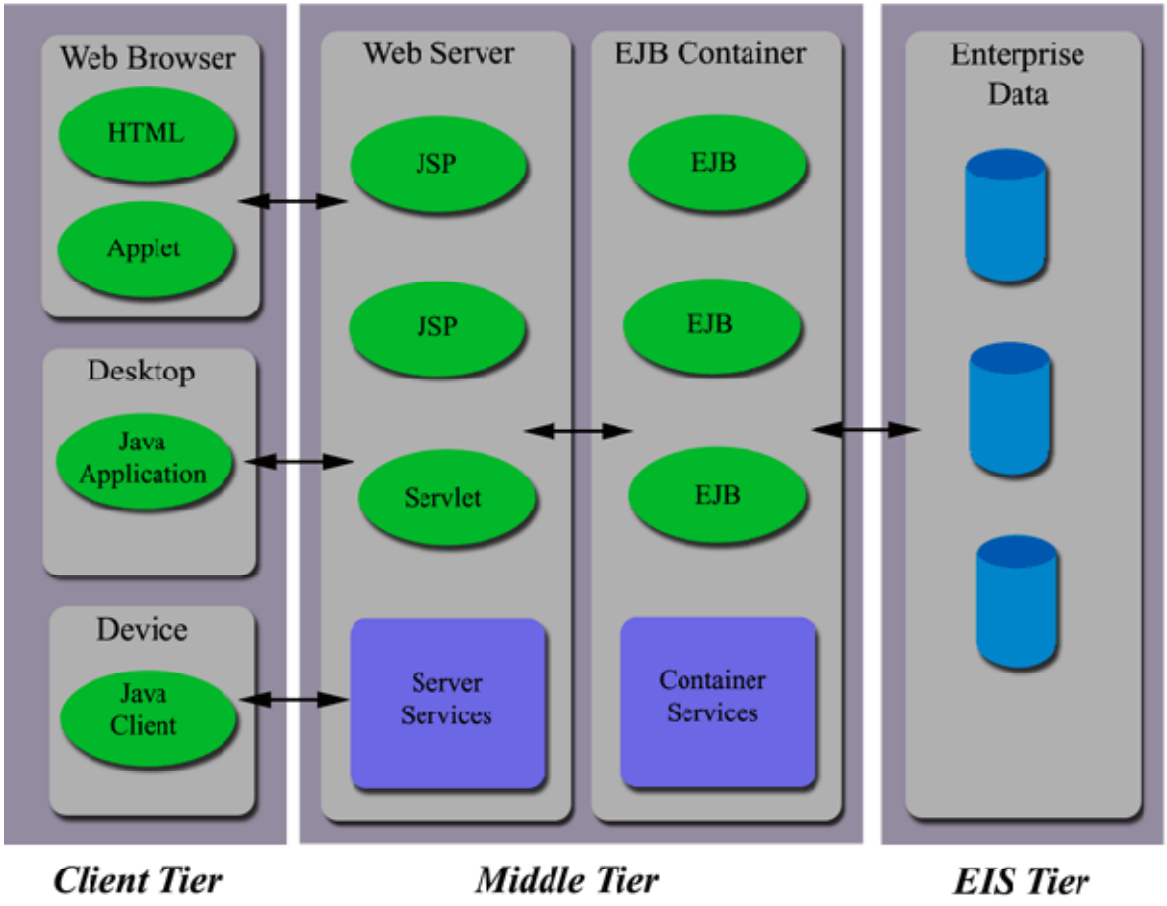
Με δεδομένους τους παραπάνω περιορισμούς, αναζητήθηκε μια νέα προσέγγιση. Το αποτέλεσμα αυτής της αναζήτησης είναι το πολυστρωματικό (multi-tier) πρότυπο αρχιτεκτονικής, το οποίο φαίνεται στο παρακάτω δομικό διάγραμμα για επιπλέον (Σχήμα 2.2).



Σχήμα 2.2 Αρχιτεκτονική εφαρμογής πολλών στρωμάτων - πολυστρωματική προδιαγραφή

Στην πολυστρωματική (multi-tier) προδιαγραφή η παρουσία του χρήστη στο μεσαίο στρώμα βρίσκεται υπό την μορφή διεπαφής χρήστη. Το κομμάτι της επιχειρησιακής λογικής (business logic) της εφαρμογής βρίσκεται επίσης στο μεσαίο στρώμα και επικοινωνεί με το χρήστη, μέσω της διεπαφής χρήστη . Όταν μια αλλαγή χρειαστεί για την εφαρμογή, θα ενημερωθεί ένα σημείο, αυτό της διεπαφής χρήστη στο μεσαίο στρώμα, αντί να ενημερωθεί κάθε μηχανή πελάτη όπως γινόταν παλαιότερα.

Το παρακάτω πιο αναλυτικό δομικό διάγραμμα (Σχήμα 2.3) δείχνει τα διάφορα στοιχεία λογισμικού τα οποία εκτελούνται στα επιμέρους στρώματα της εφαρμογής στην πλατφόρμα J2EE.



Σχήμα 2.3 Η πολυστρωματική προδιαγραφή με παρουσίαση των συστατικών λογισμικού (components) που εκτελούνται σε κάθε στρώμα

2.3 Συστατικά λογισμικού (components)

Ο όρος συστατικό λογισμικού (component) έχει πάρει παρά πολλά νοήματα στο χώρο της ανάπτυξης λογισμικού, και έτσι είναι αναγκαίο να δοθεί καταρχάς ένας ορισμός. Ένα συστατικό λογισμικού είναι μια αυτοτελής (self-contained), επαναχρησιμοποιήσιμη μονάδα λογισμικού η οποία ολοκληρώνεται μέσα σε μια εφαρμογή. Οι πελάτες αλληλεπιδρούν με αυτές τις λογισμικές μονάδες μέσω καλά ορισμένων διεπαφών. Στην γλώσσα Java το πιο απλό συστατικό λογισμικού είναι το Java Bean.

Στον επιχειρησιακό χώρο, τα συστατικά λογισμικού εστιάζονται περισσότερο στην υλοποίηση επιχειρησιακών (business) υπηρεσιών (και λιγότερο για υπηρεσίες ιστού - συστατικά λογισμικού ιστού), με τη σύμβαση τους να ορίζεται με τους όρους των επιχειρησιακών λειτουργιών, τις οποίες

μπορεί να διεξάγουν. Η βασική προδιαγραφή συστατικού λογισμικού για την πλατφόρμα J2EE είναι η προδιαγραφή των Enterprise Java Beans(EJBs), η οποία ορίζει τρόπους για τη συσκευασία (packaging) και εγκατάσταση μιας επιχειρησιακής εφαρμογής καθώς και διασύνδεσης με αυτόνομες υπηρεσίες. Η προδιαγραφή EJB ορίζει τις συμβάσεις που απαιτούνται για την επικοινωνία των επιμέρους συστατικών λογισμικού μέσω προτυποποιημένων διεπαφών στη γλώσσα Java .

Αρκετές εφαρμογές παρακάμπτουν τα EJBs. Σε αυτές τις εφαρμογές, τα συστατικά λογισμικού του μεσαίου στρώματος (JSP/servlets) προσπελάζουν απευθείας τη βάση δεδομένων. Η χρήση των συστατικών λογισμικού απαιτεί την οργάνωση της εφαρμογής σε στρώματα, με τις επιχειρησιακές υπηρεσίες να βρίσκονται μέσα στο επιχειρησιακό (business) στρώμα ακολουθώντας το πρότυπο συστατικών EJB , όπως φαίνεται στο Σχήμα 2.3.

Ιστορικά, μια από τις αντιρρήσεις για την υιοθέτηση των συστατικών λογισμικού EJB από την Java EE ήταν η πολυπλοκότητα στην υλοποίηση τους. Το πρόβλημα αυτό έχει λυθεί κατά ένα μεγάλο μέρος, και έτσι αυτό που έχει μείνει δεν είναι τίποτα άλλο από τα πλεονεκτήματα που προσφέρουν το πλήθος των καλά ορισμένων υπηρεσιών σε μια εφαρμογή:

- Χαλαρή σύζευξη: Η χρήση συστατικών λογισμικού για την υλοποίηση υπηρεσιών βοηθά στην χαλαρή σύζευξή μεταξύ των στρωμάτων μιας εφαρμογής. Η υλοποίηση ενός συστατικού λογισμικού μπορεί να αλλάζει χωρίς να επηρεάζει τους πελάτες ή άλλα συστατικά λογισμικού από τα οποία εξαρτάται.

- Διαχείριση εξαρτήσεων : Οι εξαρτήσεις ενός συστατικού λογισμικού δηλώνονται με τη μορφή μεταδεδομένων (metadata) και μετά αυτομάτως διευθετούνται από τον υποδοχέα (container) , δηλαδή από το περιβάλλον του εξυπηρετητή εφαρμογών (application server) όπου εγκαθίσταται και εκτελείται το συστατικό λογισμικού.

- Διαχείριση κύκλου ζωής : Ο κύκλος ζωής των συστατικών λογισμικού είναι καλά προσδιορισμένος και είναι διαχειριζόμενος από τον εξυπηρετητή εφαρμογών. Οι υλοποιήσεις των συστατικών λογισμικού μπορούν να συμμετέχουν στις λειτουργίες κύκλου ζωής για την απόκτηση και απελευθέρωση πόρων ή για την εκτέλεση άλλων αρχικοποιήσεων και κλεισιμάτων.

- Ορισμός υπηρεσιών υποδοχέα (container) : Οι επιχειρησιακές (business)

μέθοδοι των συστατικών λογισμικού μπορούν να διακόπτονται από τον εξυπηρετητή εφαρμογών για την παραχώρηση προτεραιότητας σε αυτοματοποιημένες υπηρεσίες της εφαρμογής, όπως: συγχρονισμού, διαχείρισης δοσοληψιών, ασφάλειας και διαχείριση απομακρυσμένων κλήσεων (remoting).

- Φορητότητα: Τα συστατικά λογισμικού που ακλουθούν τις προδιαγραφές Java EE και εγκαθίστανται σε κατάλληλους εξυπηρετητές , οι οποίοι επίσης ακολουθούν αυτές τις προδιαγραφές, μπορούν πολύ εύκολα να μεταφερθούν από έναν εξυπηρετητή σε έναν άλλο, με την προϋπόθεση ότι ο τελευταίος ακολουθεί επίσης τις παραπάνω προδιαγραφές.

- Επεκτασιμότητα και αξιοπιστία: Οι εξυπηρετητές εφαρμογών σχεδιάζονται ώστε να διασφαλίζεται ότι διαχειρίζονται τα συστατικά λογισμικού αποδοτικά και με προοπτική πάντα την επεκτασιμότητα τους. Οι λειτουργίες που υλοποιούνται χρησιμοποιώντας συστατικών λογισμικού, είναι πάντα εξαρτημένες από τον εκάστοτε τύπο συστατικών καθώς και από και τις ρυθμίσεις (configurations) του κάθε εξυπηρετητή. Έτσι μπορεί να επαναλαμβάνουν εσφαλμένες μεθόδους ή να αποτυγχάνουν να λειτουργήσουν όταν εκτελεστούν σε κάποιον άλλο εξυπηρετητή.

2.4 Πλεονεκτήματα του μοντέλου πολλαπλής στρωμάτωσης

Η πολυστρωματική προσέγγιση που έχει υιοθετηθεί από την J2EE πλατφόρμα έχει αρκετά πλεονεκτήματα:

- Μειώνει την πολυπλοκότητα της κατανεμημένης ανάπτυξης κάνοντας χρήση μιας απλοποιημένης αρχιτεκτονικής και μοιράζοντας το φόρτο εργασίας μεταξύ διαφόρων ρόλων.

- Το κομμάτι της επιχειρησιακής λογικής της εφαρμογής εκτελείται στο μεσαίο στρώμα, μέσα στον Enterprise Java Bean (EJB) υποδοχέα και/ή πάνω στον εξυπηρετητή ιστού. Οι συγκεκριμένοι υποδοχείς και εξυπηρετητές μπορούν να διαχειριστούν δύσκολες εργασίες αντί για τους προγραμματιστές. Για παράδειγμα, ένας EJB υποδοχέας μπορεί να διαχειριστεί, τις δοσοληψίες (transactions) μεταξύ των διαφόρων συστατικών λογισμικού, τη διαχείριση – άντληση στιγμιοτύπων (instance pooling), και τη διαχείριση των

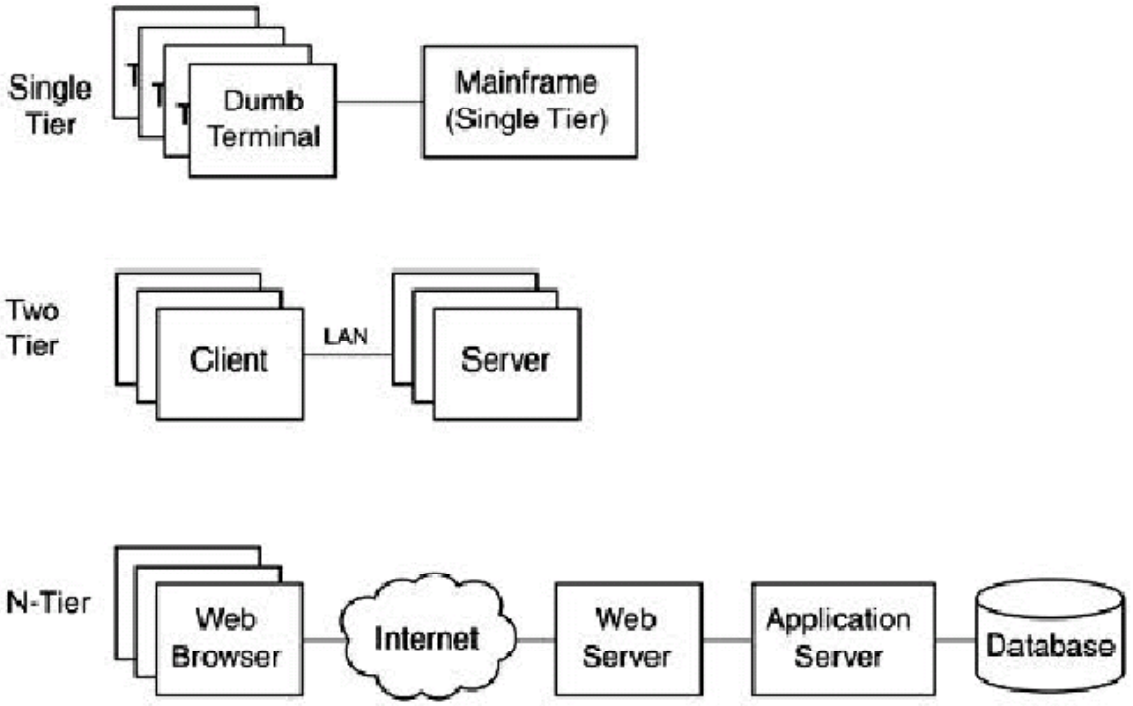
δεδομένων της βάσης (data persistence) , χωρίς την απαίτηση από τον προγραμματιστή να ασχοληθεί με την συγγραφή κώδικα που να εκτελεί τις σύνθετες αυτές τις λειτουργίες. Ένας εξυπηρετητής ιστού μπορεί να δημιουργεί και να διαχειρίζεται τα στιγμιότυπα των κλάσεων των servlet και να διαχειρίζεται πολλαπλά νήματα εκτέλεσης και κανάλια επικοινωνίας. Ο προγραμματιστής εφαρμογών, αντί του γραψίματος κώδικα για την πραγμάτωση αυτών των λειτουργιών προσδιορίζει απλώς τη συγκεκριμένη συμπεριφορά τους κατά το χρόνο της εγκατάστασης των εφαρμογών στο εξυπηρετητή.

- Τα μέλη που απαρτίζουν την ομάδα ανάπτυξης έχουν συνήθως διαφορετικούς ρόλους. Ο καθένας είναι ειδικός σε ένα η περισσότερα πεδία. Για παράδειγμα, το περιεχόμενο μιας HTML σελίδας ή ενός εκμαγείου στιλ σελίδων (stylesheet) θα μπορούσε να είχε δημιουργηθεί από έναν σχεδιαστή γραφικών. Ένας έμπειρος προγραμματιστής είναι υπεύθυνος για την υλοποίηση του επιχειρησιακού τμήματος της εφαρμογής το οποίο είναι ενσωματωμένο στα συστατικά λογισμικού Enterprise Java Bean . Αυτός ο οποίος αναπτύσσει το κομμάτι ιστού της εφαρμογής οφείλει να αναπτύσσει και τη διεπαφή χρήστη και γενικότερα την πολιτική παρουσίασης συστατικών λογισμικού (presentation logic) χρησιμοποιώντας JSPs (Java Server Pages) και servlets. Αυτός που έχει το ρόλο του κτισίματος της εφαρμογής παίρνει τα διάφορα συστατικά λογισμικού της εφαρμογής και τα τοποθετεί όλα μαζί, δημιουργώντας ένα EAR αρχείο και ένα αρχείο περιγραφέα εγκατάστασης το οποίο περιγράφει πως ή εφαρμογή είναι διατεταγμένη (deployed).

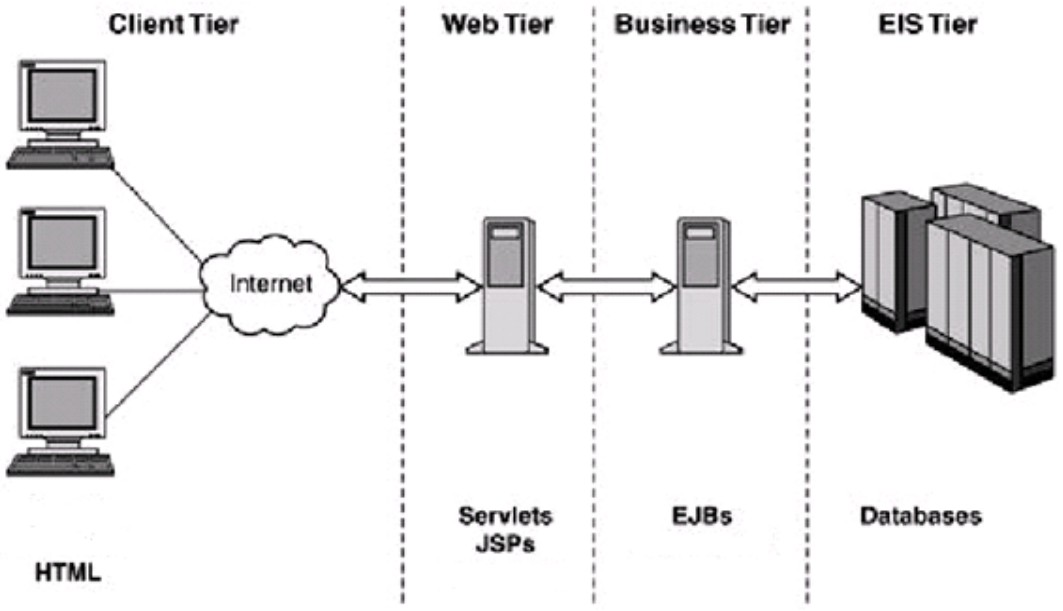
- Είναι επεκτάσιμη, επιτρέποντας την ανάπτυξη συστημάτων που συναθροίζουν διαφορετικές ανάγκες οι οποίες μπορούν να μεταβάλλονται συχνά.

- Όταν οι απαιτήσεις του συστήματος αυξάνονται, το κομμάτι του λογισμικού που εμπεριέχει την επιχειρησιακή λογική της εφαρμογής μπορεί να ενημερώνεται εύκολα σε ένα σημείο του μεσαίου επιπέδου χωρίς να υπάρχει η ανάγκη της ενημέρωσης κάθε μηχανής πελάτη.

Το παρακάτω δομικά διαγράμματα, (Σχήμα 2.4 και 2.5) δείχνουν την εξέλιξη της αρχιτεκτονικής και τον τρόπο κατανομής των στρωμάτων των επιχειρησιακών εφαρμογών αντιστοίχως.



Σχήμα 2.4 Η εξέλιξη της αρχιτεκτονικής των επιχειρησιακών εφαρμογών

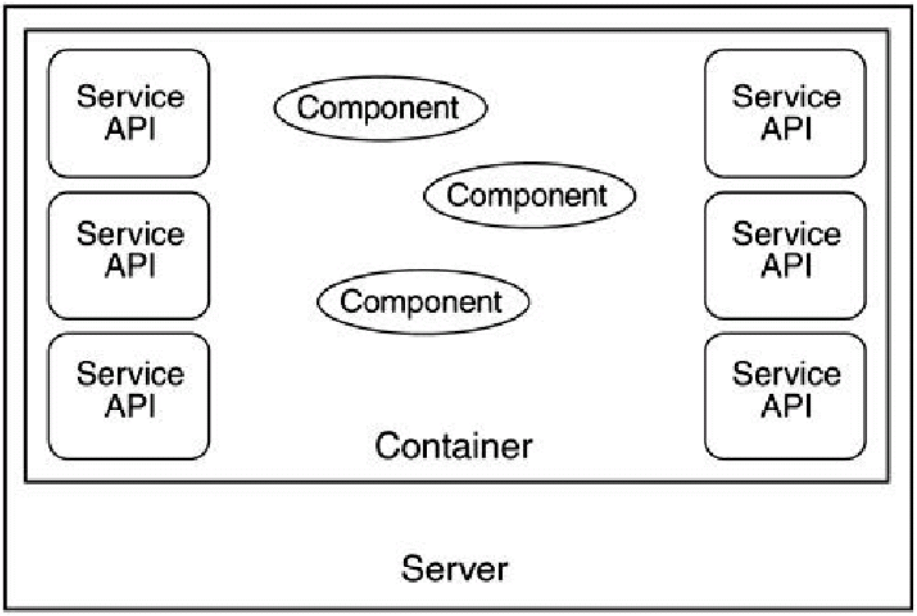


Σχήμα 2.5 Η κατανεμημένη αρχιτεκτονική, των επιχειρησιακών εφαρμογών

2.5 Αρχιτεκτονική της πλατφόρμας J2EE

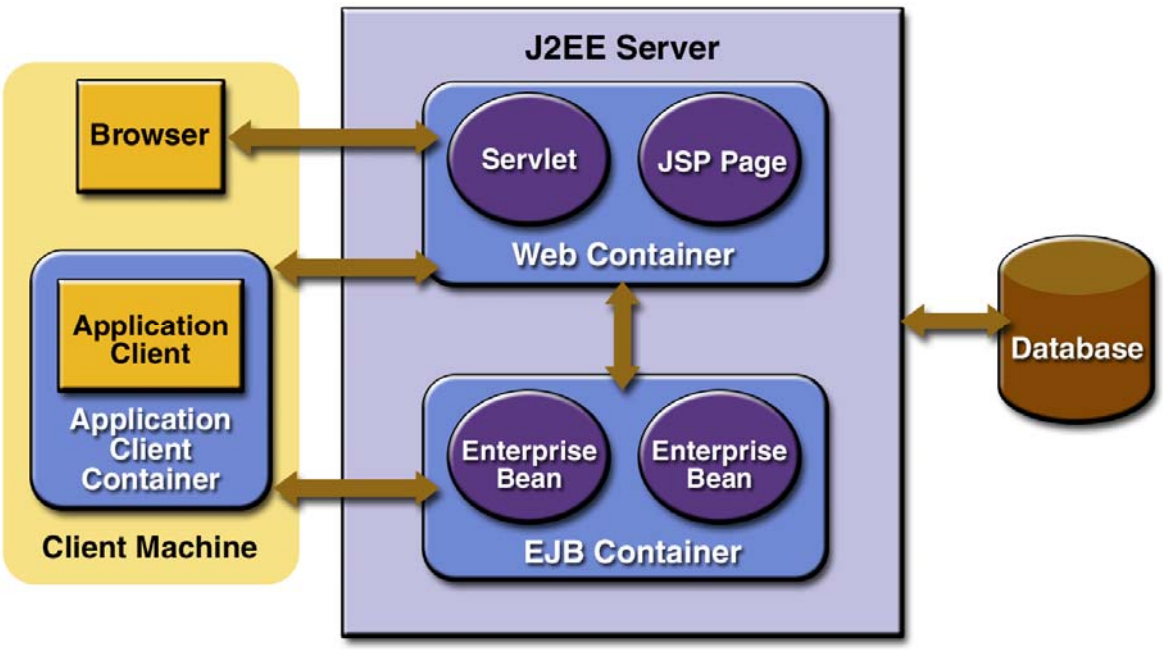
Μελετώντας την αρχιτεκτονική της πλατφόρμας J2EE σαν μια στοίβα στρωμάτων, όπως θα την έβλεπε ένας προγραμματιστής, στο κατώτερο στρώμα της στοίβας υπάρχει η υποδομή της Java 2 Standard Edition (J2SE).

Οι εξυπηρετητές εφαρμογών J2EE εκτελούν τις εφαρμογές στο επίπεδο Java Virtual Machine (JVM) και παρέχουν διεπαφές στους προγραμματιστές των εφαρμογών (API). Δύο (κυρίως) τύποι εφαρμογών μπορούν να αναπτυχθούν στους εξυπηρετητές εφαρμογών J2EE – οι εφαρμογές ιστού και οι EJB(Enterprise Java Bean) εφαρμογές. Αυτές οι εφαρμογές εγκαθίστανται και εκτελούνται μέσα σε κατάλληλους υποδοχείς (containers). Η προδιαγραφή J2EE ορίζει υποδοχείς για τη διαχείριση του κύκλου ζωής των συστατικών λογισμικού μιας εφαρμογής από την πλευρά του εξυπηρετητή.



Σχήμα 2.6 Γενική άποψη ενός υποδοχέα (container)

Υπάρχουν δύο τύποι υποδοχέων οι υποδοχείς servlet και οι υποδοχείς EJB. Οι υποδοχείς servlet διαχειρίζονται τον κύκλο ζωής των εφαρμογών ιστού ενώ οι υποδοχείς EJB διαχειρίζονται τον κύκλο ζωής των EJB.

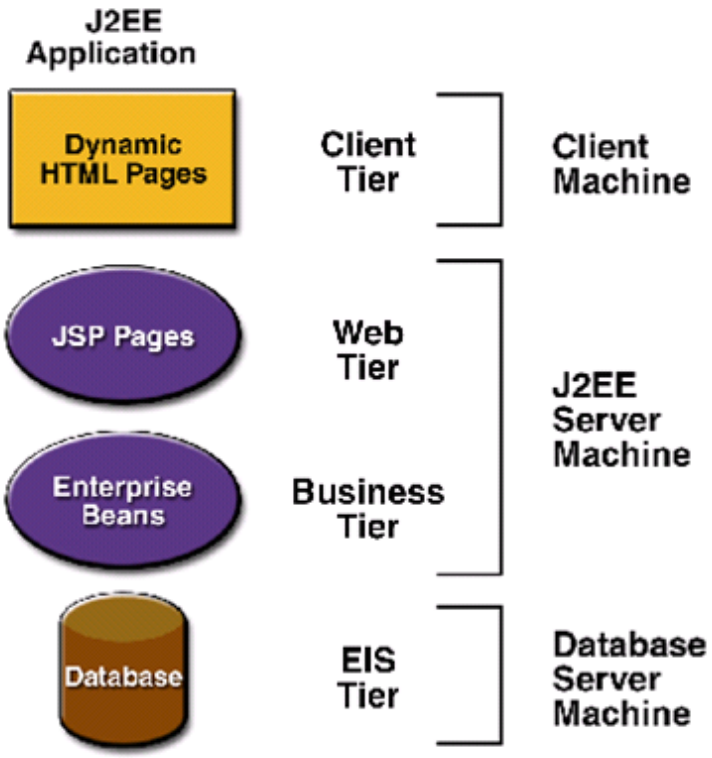


Σχήμα 2.7 Γενική άποψη των υποδοχέων ιστού (web) και EJB ενός εξυπηρετητή εφαρμογών J2EE

2.5.1 Κατανεμημένες πολυστρωματικές εφαρμογές

Τα διάφορα στοιχεία λογισμικού που συνιστούν την εφαρμογή J2EE εγκαθίστανται σε διαφορετικές μηχανές, ανάλογα με το στρώμα που ανήκουν στην πολυστρωματική δομή J2EE. Το Σχήμα 2.8 δείχνει μια πολυστρωματική εφαρμογή J2EE χωρισμένη σε στρώματα τα οποία περιγράφονται στην παρακάτω λίστα συστατικών λογισμικού J2EE :

- Στρώμα πελάτη (client) το οποίο εκτελείται στη μηχανή του πελάτη.
- Στρώμα ιστού (web) που εκτελείται στον εξυπηρετητή εφαρμογών J2EE.
- Λογισμικό επιχειρησιακού (business) στρώματος που εκτελείται στον εξυπηρετητή εφαρμογών J2EE.
- Στρώμα Συστήματος Επιχειρησιακών Πληροφοριών (Enterprise Information System -EIS) που εκτελείται στον εξυπηρετητή EIS.



Σχήμα 2.8 Πολυστρωματική Εφαρμογή

2.6 Τεχνολογικά Χαρακτηριστικά

Η πλατφόρμα J2EE, για τη μείωση του κόστους και την ταχύτερη σχεδίαση και ανάπτυξη, παρέχει μια προσέγγιση βασισμένη σε συστατικά λογισμικού, για τη σχεδίαση ,ανάπτυξη, δόμηση, και εγκατάσταση των επιχειρησιακών εφαρμογών. Η πλατφόρμα J2EE προσφέρει μια κατανεμημένη, πολυστρωματική προδιαγραφή εφαρμογών, επαναχρησιμοποιήσιμα συστατικά λογισμικού, ένα ενιαίο πρότυπο ασφάλειας, ευέλικτο έλεγχο δοσοληψιών, και υποστήριξη υπηρεσιών ιστού δια της ολοκληρωμένης ανταλλαγής δεδομένων πάνω σε γλώσσα XML(eXtensible Markup Language) - βασιζόμενη σε ανοικτές προτυποποιήσεις και πρωτόκολλα.

Η ανάπτυξή ενός επιχειρησιακού λογισμικού αποτελεί μια σύνθετη εργασία ή οποία χρειάζεται βαθιά γνώση σε αρκετά και διαφορετικά πεδία. Για παράδειγμα, μια τυπική επιχειρησιακή εφαρμογή απαιτεί ότι ο προγραμματιστής θα είναι εξοικειωμένος με θέματα επικοινωνίας μεταξύ διαδικασιών , ασφάλειας, με ειδικά ερωτήματα βάσεων δεδομένων(SQL Queries), κτλ. Η πλατφόρμα J2EE παρέχει την υποστήριξη, με αρκετά διαφανή τρόπο τέτοιων και παρόμοιων υπηρεσιών. Τα παραπάνω έχουν ως αποτέλεσμα οι προγραμματιστές να εστιάζουν στην υλοποίηση του κώδικα του επιχειρησιακού κομματιού

, παρά στο γράψιμο κώδικα ο οποίος παρέχει βασική υποστήριξη στις εφαρμογές.
Ίσως ένα από τα μεγαλύτερα πλεονεκτήματα της πλατφόρμας J2EE είναι η υποστήριξη συστατικών λογισμικού. Το λογισμικό το οποίο είναι βασισμένο σε συστατικά λογισμικού έχει ένα πλήθος πλεονεκτημάτων σε σχέση με την παραδοσιακή , συνηθισμένη ανάπτυξη:

- **Υψηλή παραγωγικότητα**: Λιγότεροι μπορούν να πετύχουν περισσότερα μέσω της τοποθέτησης, μιας εφαρμογής η οποία είχε πραγματοποιηθεί για παραπλήσιο σκοπό, μαζί με ήδη ξαναδουλεμένα στοιχεία λογισμικού αντί για μια υλοποίηση από την αρχή με το συνηθισμένο τρόπο.

- **Γρήγορή ανάπτυξη**: Συστατικά λογισμικού που ήδη υπάρχουν, μπορούν να συνδυαστούν με γρήγορο τρόπο και να δημιουργηθούν νέες εφαρμογές.

- **Υψηλή ποιότητα**: Αποφυγή ελέγχου συνολικά μιας εφαρμογής. Ο έλεγχος των εφαρμογών που βασίζεται σε συστατικά λογισμικού μπορεί να γίνει πιο ευέλικτος με τον έλεγχο όλης της εφαρμογής μέσω του ελέγχου των υπαρχόντων συστατικών λογισμικού.

- **Ευκολότερη συντήρηση**: Τα συστατικά λογισμικού αποτελούν αυτόνομες οντότητες και έτσι η συντήρηση τους έτσι είναι πιο εύκολη και λιγότερο δαπανηρή.

2.7 Τεχνολογίες J2EE

Για την κατανόηση των τεχνολογιών J2EE είναι απαραίτητη, καταρχάς η κατανόηση του ρόλου του υποδοχέα (container) στην αρχιτεκτονική J2EE. Όλες οι τρέχουσες τεχνολογίες που χρησιμοποιούνται στην προδιαγραφή J2EE στηρίζονται σε αυτή την ισχυρή, ακόμα, έννοια.
Ένας υποδοχέας δεν είναι τίποτα άλλο από μια οντότητα λογισμικού οι όποια εκτελείται εντός του εξυπηρετητή και είναι υπεύθυνη για τη διαχείριση ειδικών τύπων συστατικών λογισμικού. Παρέχει το περιβάλλον εκτέλεσης για τα συστατικά λογισμικού που ένας προγραμματιστής αναπτύσσει. Είναι προφανές ότι υποδοχείς σαν και αυτούς που παρέχει η αρχιτεκτονική J2EE, είναι ικανοί να παρέχουν ανεξαρτησία μεταξύ των διαδικασιών της ανάπτυξης μιας εφαρμογής από την εγκατάσταση της, και να παρέχουν τη δυνατότητα μεταφοράς, μεταξύ διαφόρων τύπων εξυπηρετητών μεσαίου στρώματος.
Ο υποδοχέας επίσης είναι υπεύθυνος για τη διαχείριση του κύκλου ζωής των

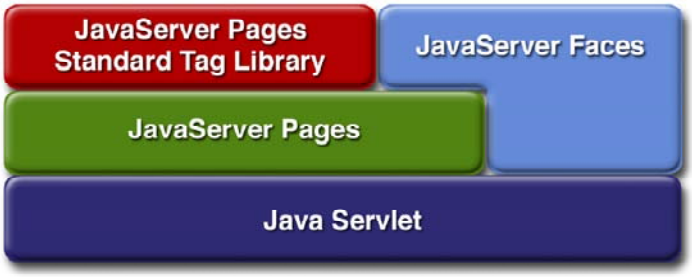
συστατικών λογισμικού που είναι εγκαταστημένα μέσα σε αυτόν και για λειτουργίες όπως αυτές της διαχείρισης πόρων (pooling resource) καθώς και της επιβολής της ασφάλειας. Για παράδειγμα, ο διαχειριστής της εφαρμογής έχει τη δυνατότητα να περιορίσει την πρόσβαση σε μια συγκεκριμένη μέθοδο για μια μικρή ομάδα χρηστών. Ο υποδοχέας θα επέβαλλε τον παραπάνω περιορισμό μέσω της ανάσχεσης των κλήσεων της συγκεκριμένης μεθόδου, βεβαιώνοντας ταυτόχρονα ότι η πρόσβαση της οντότητας είναι στη λίστα των εχόντων δικαιώματα.
Εξαρτάται από τον τύπο του υποδοχέα εάν θα παρέχεται πρόσβαση σε κάποια ή σε όλα από τα APIs J2EE. Όλα τα συστατικά λογισμικού J2EE εγκαθίστανται και εκτελούνται εντός ενός υποδοχέα κάποιου τύπου. Για παράδειγμα, τα EJBs εκτελούνται στον υποδοχέα EJB, και τα servlets εκτελούνται στον υποδοχέα ιστού. Συνολικά η πλατφόρμα έχει τέσσερα τύπους υποδοχέων:

- **Υποδοχέας εφαρμογών** (application container): Φιλοξενεί αυτόνομες εφαρμογές Java
- **Υποδοχέας μικροεφαρμογών** (applet container): Παρέχει ένα περιβάλλον εκτέλεσης για applets(μικροεφαρμογές).
- **Υποδοχέας** ιστού (web container) : Φιλοξενεί εφαρμογές ιστού , όπως servlets και Java Server Pages (JSP)
- **Επιχειρησιακός υποδοχέας** (enterprise container) : Φιλοξενεί συστατικά λογισμικού EJB

2.7.1 Servlets (μικρουπηρεσίες)

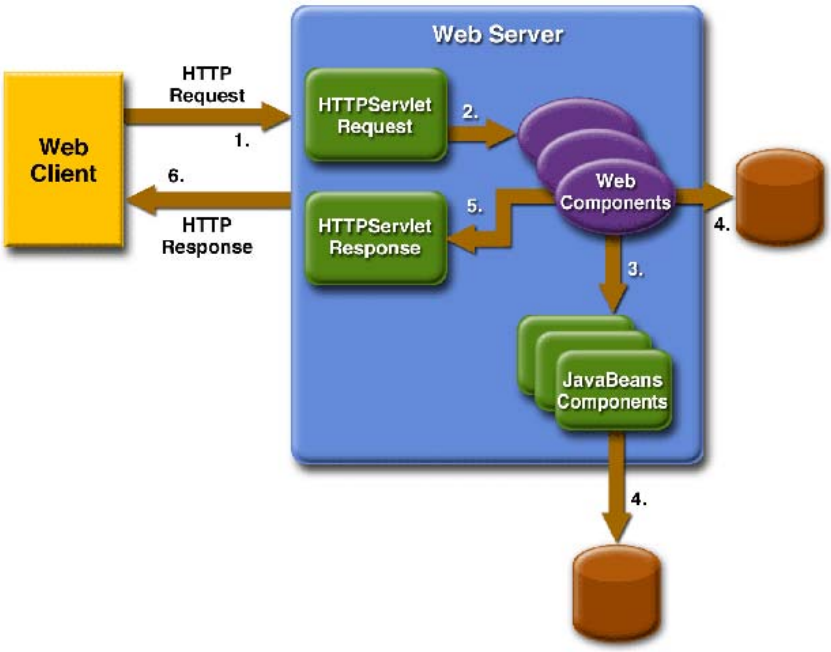
Τα servlets είναι συστατικά λογισμικού ιστού ικανά για τη δημιουργία δυναμικού περιεχομένου. Είναι από τα πιο συχνά χρησιμοποιούμενα συστατικά λογισμικού J2EE στον παγκόσμιο ιστό (World Wide Web) σήμερα. Παρέχουν ένα αποδοτικό μηχανισμό, για διάδραση μεταξύ πελάτη που βρίσκεται σε περιβάλλον ιστού και του επιχειρησιακού (business) κομματιού της εφαρμογής που βρίσκεται στον εξυπηρετητή. Επίσης είναι σαφώς πιο «ελαφριά» και περισσότερο εύκολα στη διαχείριση από την παλαιοτέρων ετών δημοφιλή προσέγγιση CGI.
Επειδή τα servlets είναι απλούστερα και χρειάζονται γενικά λιγότερους πόρους, αρκετοί προγραμματιστές προτιμούν να τα χρησιμοποιούν, μαζί με JSPs, σχεδόν αποκλειστικά στις υλοποιήσεις τους από το να κάνουν χρήση των αρκετά πολύπλοκων συστατικών λογισμικού

EJB. Η παραπάνω πρακτική είναι αποδοτική για πολύ απλές εφαρμογές, αλλά υπολείπεται όταν απαιτείται υποστήριξη δοσοληψιών (transaction) από την εφαρμογή.



Σχήμα 2.9 Τεχνολογίες ιστού Java που έχουν ως υποδομή την τεχνολογία των servlet

Τα servlets είναι πιο αποδοτικά όταν διαχειρίζονται απλές εφαρμογές, όπως η συλλογή παραμέτρων και ο έλεγχος πληροφορίας εισόδου από τα πεδία εισόδου μιας σελίδας ιστού. Όταν οι αρχικοί έλεγχοι γίνουν, η πληροφορία πρέπει να διοχετεύεται σε πιο κατάλληλα συστατικά λογισμικού για εκτέλεση. Τα servlets εκτελούνται μέσα στον υποδοχέα servlet (πολλές φορές μπορεί να αναφέρεται και ως servlet μηχανή-engine) που φιλοξενείται μέσα σε έναν εξυπηρετητή ιστού. Ο υποδοχέας servlet διαχειρίζεται τις κλήσεις ιστού του πελάτη, οι οποίες γίνονται μέσω του πρωτοκόλλου HTTP (Hypertext Transfer Protocol) , και τις μετασχηματίζει σε κλήσεις για αντικείμενα (objects). Παρομοίως, ο υποδοχέας servlet
μετασχηματίζει τις αποκρίσεις των servlets και τις αντιστοιχεί σε αποκρίσεις για αντικείμενα κατάλληλου πρωτοκόλλου ιστού.



Σχήμα 2.10 Διαχείριση αίτησης εφαρμογής ιστού

2.7.2 Java Server Pages (JSPs)

Τα JSP είναι ένας άλλος τύπος συστατικού λογισμικού ιστού J2EE και έχει σχέση με την τεχνολογία servlet. Στην πραγματικότητα, τμήματα των JSP μεταφράζονται σε servlets τα οποία στη συνέχεια εκτελούνται εντός του περιβάλλοντος του servlet container. Τα JSP δημιουργήθηκαν για να γίνεται πιο εύκολη η συντήρηση τμημάτων ενός συστήματος, τα οποία υποστηρίζουν το κομμάτι της παρουσίασης ιστού της εφαρμογής, από τα μέλη μιας ομάδας ανάπτυξης χωρίς αυτά κατά ανάγκη να είναι προγραμματιστές, όπως διαφορετικά συνηθίζεται. Οι σχεδιαστές σελίδων ιστού (web designers) συντηρούν τον κώδικά παρουσίασης, σε γλώσσα HTML (HyperText Markup Language). Αυτό είναι σαφώς πιο δύσκολο να γίνει όταν ή HTML δημιουργείται από εντολές Java που περιέχονται εντός των servlet.

2.7.3 Enterprise Java Bean - EJB (Επιχειρησιακά συστατικά λογισμικού)

Η EJB προδιαγραφή αποτελεί τον πυρήνα της πλατφόρμας J2EE. Αυτή ορίζει ένα αναλυτικό μοντέλο συστατικών λογισμικού για αποδοτικό κτίσιμο εφαρμογών και κατανεμημένων, απ'την πλευρά του εξυπηρετητή , επιχειρησιακών συστατικών λογισμικού Java.

Υπάρχουν τρείς τύποι EJB:

- Τα session bean, προορίζονται για την υλοποίηση υπηρεσιών. Δεν συγκρατούν τη κατάστασή τους(δεν είναι persistent) και συνήθως εμπεριέχουν το μεγαλύτερο επιχειρησιακό κομμάτι σε μία επιχειρησιακή εφαρμογή Java. Τα session bean μπορεί να είναι stateful, πράγμα που σημαίνει ότι διατηρούν τη σχέση τους με έναν πελάτη, μεταξύ αλλεπάλληλων επαφών με αυτόν. Ο άλλος τύπος των session beans είναι τα stateless. Στην περίπτωση των stateless session bean κάθε αλλεπάλληλη κλήση του ίδιου session bean από τον ίδιο πελάτη εκλαμβάνεται ως νέα διαδικασία, μη συσχετιζόμενη με τυχόν προηγούμενες.

- Τα entity bean υπάρχουν για την καταχώριση πληροφορίας της βάσης δεδομένων σε ένα μόνιμο μέσω αποθήκευσης, δηλαδή αποθήκευση τυπικά μια πλήρους ή ενός τμήματος γραμμής πληροφορίας από αυτές που βρίσκονται σε ένα πίνακα μιας βάσης δεδομένων. Τα

entity bean παρέχουν αυτοματοποιημένες υπηρεσίες για την επιβεβαίωση ότι η αντικειμενοστραφής απεικόνιση αυτών των μόνιμων δεδομένων παραμένει συγχρονισμένη συνεχώς με τα ενεργά δεδομένα της υποδομής τη βάσης δεδομένων. Επίσης τα entity bean χρησιμοποιούνται για τη μορφοποίηση των δεδομένων των βάσεων δεδομένων, είτε για να βοηθήσουν σε μια εργασία ή απ' την άλλη να προετοιμάσουν τα δεδομένα για παρουσίαση σε μια σελίδα ιστού. Για παράδειγμα, σε ένα πίνακα υπαλλήλων μιας βάσης δεδομένων, κάθε έγγραφή (γραμμή του πίνακα) αντιστοιχίζεται σε ένα στιγμιότυπο π.χ. του Employee entity bean.

- Τα **message-driven beans** έχουν σχεδιαστεί για να παρέχουν ασύγχρονή ανταλλαγή μηνυμάτων JMS (Java Messaging Service), δηλαδή ό καλών δεν περιμένει απάντηση κατά την κλήση μιας λειτουργίας. Διαφορετικά από τα session και τα entity beans, τα message-driven beans δεν έχουν δημοσιευμένες διεπαφές χρήστη. Τα message-driven beans δρουν ανώνυμα στο παρασκήνιο. Τα message-driven beans δεν διατηρούν την κατάσταση τους (είναι stateless) και είναι σχετικά ένας καινούριος τύπος συστατικού λογισμικού EJB ο οποίος εισήχθηκε με την έκδοση 1.3 της προδιαγραφής J2EE.

2.7.4 APIs και άλλες τεχνολογίες J2EE

Υποστηρίζονται αρκετά προγραμματιστικές διεπαφές (API) με την πλατφόρμα J2EE. Παρόλο που η χρήση τους δεν περιορίζεται σε κάποιο αποκλειστικά αρχιτεκτονικό επίπεδο, αυτές οι τεχνολογίες είναι που κάνουν τα διάφορα στρώματα να επικοινωνούν. Μερικά από τα πιο δημοφιλή παρουσιάζονται στις επόμενες παραγράφους.

2.7.4.1 Java Database Connectivity (JDBC)

Η διασύνδεση με βάσεις δεδομένων του στρώματος των Επιχειρησιακών Πληροφοριακών Συστημάτων(Enterprise Information Systems (EIS)), αποτελεί ένα ζωτικό σημείο για μια επιχειρησιακή Java εφαρμογή. Το JDBC API δημιουργήθηκε για να κάνει την παραπάνω διαδικασία ευκολότερη, και ορίζει ένα Java API που μέσω αυτού κάποιος προγραμματιστής χρησιμοποιεί για να γράψει SQL ερωτήματα τα όποια στέλνονται στη βάση δεδομένων, και κάνει έτσι ευκολότερη τη ζωή για τους προγραμματιστές επιχειρησιακών Java εφαρμογών. Το JDBC

API, το όποιο έχει παρόμοιο πνεύμα με αυτό του API της Microsoft (του ODBC -Open Database Connectivity), απλοποιεί την πρόσβαση στις σχεσιακές βάσεις δεδομένων και αποτελεί μια γενική, ανεξάρτητη από κατασκευαστή, διεπαφή για βάσεις δεδομένων. Η χρήση του JDBC προσδίδει στις εφαρμογές φορητότητα και κάνει τις δυνατότητες της βάσης δεδομένων που χρησιμοποιεί να είναι εφαρμόσιμες σε ένα πλατύτερο εύρος πλατφόρμων κατασκευαστών. Η πλειονότητα του JDBC API αποτελεί κομμάτι της Java (J2SE), και δεν περιορίζεται η χρήση του μόνο στην πλατφόρμα J2EE. Παρόλα αυτά, υπάρχουν αρκετές επεκτάσεις που η έκδοση J2EE προσθέτει, κατά κύριο λόγο την υποστήριξη κάποιων προχωρημένων λειτουργιών για τους υποδοχείς J2EE που χρησιμοποιεί, όπως εναλλαγή συνδέσεων (connection pooling) καθώς επίσης και κάποια επιπρόσθετη υποστήριξη για Java Bean.

Το JDBC API παρέχει μια κοινόχρηστη διεπαφή για να απομονώσει τον χρήστη από τις διαφορετικές προδιαγραφές του κάθε κατασκευαστή . Έτσι διαφορετικές υλοποιήσεις βάσεων δεδομένων, διαφόρων κατασκευαστών, μπορούν να υπάρχουν ξεχωριστά κάτω από τη διαπροσωπεία του JDBC.

Στις επιχειρησιακές εφαρμογές, δεν είναι απαραίτητη η απ'ευθείας χρήση του JDBC. Για παράδειγμα, μπορεί κανείς να χρησιμοποιήσει entity beans να κάνουν τις απαραίτητες κλήσεις στη βάση δεδομένων αντί για τον χρήστη. Η πρακτική της απευθείας χρήσης JDBC αναμένεται να περιοριστεί όταν οι εξυπηρετητές εφαρμογών παρέχουν περισσότερη και πιο συντονισμένη υποστήριξη για entity beans.

2.7.4.2 Java Naming and Directory Interface (JNDI)

Όλοι οι J2EE servers χρησιμοποιούν JNDI, η οποία είναι μια υπηρεσία αναζήτησης αντικειμένων νε βάση το όνομα που χρησιμοποιείται για την εύρεση κατανεμημένων αντικειμένων (object). Η υπηρεσίες σαν και αυτή επιτρέπουν σε κάποιον να καλεί τις υπηρεσίες ενός αντικειμένου η αναφορά στο οποίο δεν είναι γνωστή κατά το χρόνο μεταγλώττισης του προγράμματος. Ένα σύστημα αρχείων είναι ένα παράδειγμα υπηρεσίας αναζήτησης αντικειμένων νε βάση το όνομα. Μια υπηρεσία καταλόγου (directory) είναι παρόμοια με την υπηρεσία αναζήτησης αντικειμένων νε βάση το όνομα και παρέχει βελτιωμένες δυνατότητες αναζήτησης. Στην πραγματικότητα, μια υπηρεσία καταλόγου περιέχει μια υπηρεσία αναζήτησης αντικειμένων νε βάση το όνομα (και όχι το αντίθετο).

Υπάρχει ποικιλία διαθέσιμων υπηρεσιών αναζήτησης αντικειμένων νε βάση το όνομα και καταλόγου, έτσι οι κλήσεις που δέχεται αυτή η διασύνδεση είναι παρόμοιες με εκείνες των βάσεων δεδομένων. Το JNDI έχει σχεδιαστεί για την διευθέτηση αυτών των κλήσεων μέσω της παροχής

ενός γενικού και ενιαίου τρόπου για την προσπέλαση των υπηρεσιών. Το ολοκληρωμένο JNDI παρόλο που έχει καταχωρηθεί ως επιχειρησιακού τύπου τεχνολογία API υπάρχει ήδη ως μέρος της J2SE. Οι πιο πολλές κατανεμημένες επιχειρησιακές εφαρμογές κάνουν χρήση αυτής της υπηρεσίας σε κάποιο σημείο τους. Για παράδειγμα, κάθε χρήση των EJBs σε μια επιχειρησιακή εφαρμογή καθιστά αναγκαία ότι το JNDI χρησιμοποιείται για την εύρεση της συσχετιζόμενης EJB Home διεπαφής.

2.7.4.3 Java Message Service (JMS)

Είναι μια υπηρεσία μηνυμάτων που επιτρέπει επικοινωνία μεταξύ κατανεμημένων εφαρμογών. Κατευθύνει τα μηνύματα που ανταλλάσσονται μεταξύ διαδικασιών και λογισμικών στοιχείων σε μια κατανεμημένη εφαρμογή, χρησιμοποιώντας αυτόνομα entities, τα οποία είναι υπεύθυνα για την κλήση των μηνυμάτων. Αυτή του είδους η επικοινωνία είναι τυπικά ασύγχρονη. Οι διάφοροι κατασκευαστές παρέχουν υπηρεσίες μηνυμάτων οι οποίες είναι προσανατολισμένες στο μεσαίο στρώμα των εφαρμογών. Η JMS υπηρεσία παρέχει μια ενιαία και γενική διεπαφή στο παραπάνω στρώμα. Η JMS υπηρεσία μπορεί να χρησιμοποιηθεί απευθείας σε μια επιχειρησιακή εφαρμογή ή μέσω ενός τύπου EJB γνωστού ως message-driven bean.

2.7.4.4 Java Authentication and Authorization Service (JAAS)

Το JAAS (Java Authentication and Authorization Service) είναι μια υπηρεσία που δίνει τη δυνατότητα σε μια J2EE εφαρμογή, για πιστοποίηση (authentication) και εξουσιοδότηση (authorization) χρηστών ή ομάδων χρηστών που έχουν τη δυνατότητα να την χρησιμοποιήσουν. Το JAAS είναι μια έκδοση της γλώσσας προγραμματισμού Java που προτυποποιεί την Pluggable Authentication Module (PAM) πλατφόρμα ανάπτυξης (framework), η οποία επεκτείνει την αρχιτεκτονική ασφάλειας της πλατφόρμας Java 2 για την παροχή εξουσιοδότησης βασισμένη στο χρήστη.

2.7.4.5 Remote Method Invocation (RMI)

Αποτελεί προτυποποίηση για κλήσεις σε απομακρυσμένα αντικείμενα στη γλώσσα Java. Τα EJBs χρησιμοποιούν RMI. Το RMI ενεργοποιεί την πρόσβαση σε συστατικά λογισμικού, σε ένα κατανεμημένο περιβάλλον, επιτρέποντας σε αντικείμενα Java να καλούν μεθόδους απομακρυσμένων αντικειμένων Java. Η μέθοδος στην πραγματικότητα καλείται από ένα εξουσιοδοτημένο αντικείμενο, το οποίο έπειτα είναι υπεύθυνο για τη διευθέτηση του περάσματος των παραμέτρων της μεθόδου στο απομακρυσμένο αντικείμενο το οποίο παρέχει την απόκριση πίσω στο αντικείμενο το οποίο ξεκίνησε την κλήση της απομακρυσμένης μεθόδου. Το RMI δεν

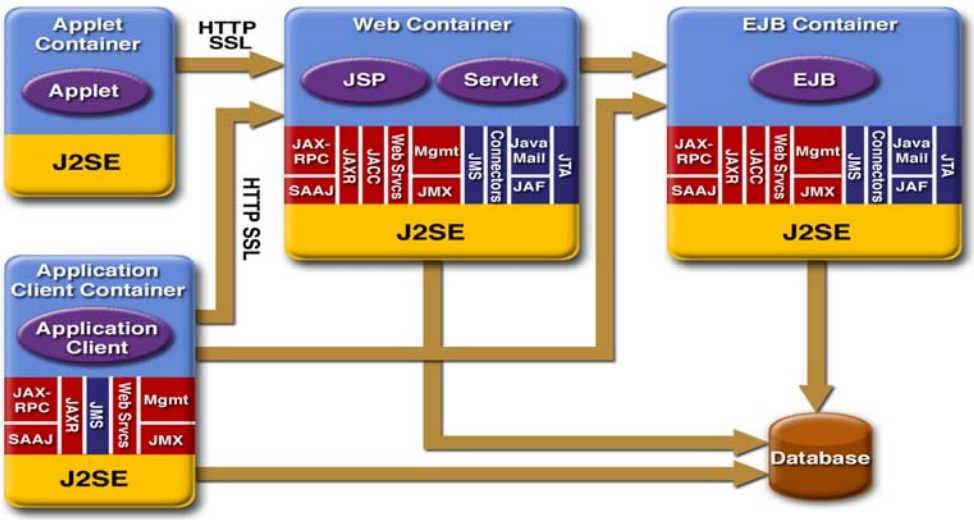
υπάρχει αποκλειστικά για την J2EE πλατφόρμα. Παρόλα αυτά, το RMI είναι η καρδία κάποιων τεχνολογιών J2EE, όπως της EJB.

2.7.4.6 Extensible Markup Language (XML)

Αν και δεν είναι και τόσο μια τεχνολογία J2EE, η XML χρησιμοποιείται από πολλές άλλες τεχνολογίες J2EE. Για παράδειγμα, στα συστατικά λογισμικού ιστού και στα επιχειρησιακά συστατικά λογισμικού (EJBs) οι περιγραφείς εγκατάστασης (deployment descriptors) τους είναι γραμμένοι σε γλώσσα XML. Αυτοί οι περιγραφείς εγκατάστασης περιγράφουν πως τα στοιχεία λογισμικού συμπεριφέρονται μιας και εγκατασταθούν.

2.7.4.7 Java Transaction API

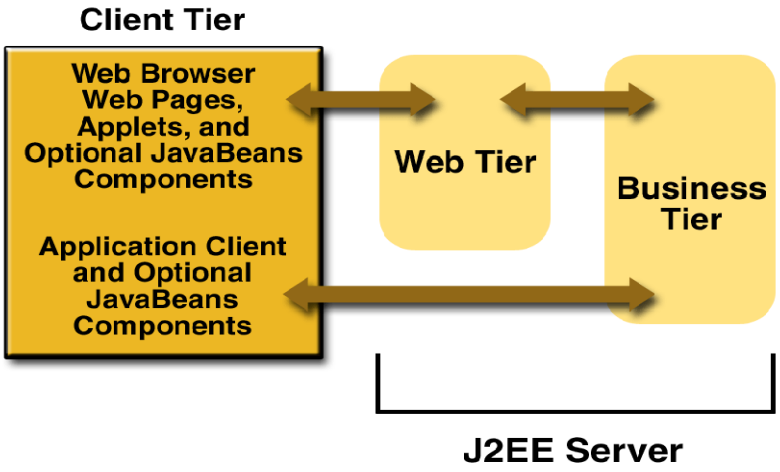
Το Java Transaction API (JTA) αποτελεί προδιαγραφή για την οριοθέτηση των δοσοληψιών (transactions). Η αρχιτεκτονική J2EE παρέχει ένα προκαθορισμένο τρόπο αυτόματης διενέργειας δοσοληψιών. Η έννοια αυτόματη διενέργεια δοσοληψιών σημαίνει , ότι οποιαδήποτε άλλη εφαρμογή έχει αναφορά στα δεδομένα που επηρεάζει μια δοσοληψία θα μπορεί να παρέμβει, στα ενημερωμένα δεδομένα, μόνο μετά από κάθε λειτουργία διαβάσματος ή γραψίματος της βάσης δεδομένων. Παρόλα αυτά, εάν μια εφαρμογή εκτελεί δύο διαφορετικές λειτουργίες πρόσβασης οι οποίες έχουν αλληλεξάρτηση μεταξύ τους, θα πρέπει να γίνει χρήση του JTA API για να οριοθετήσει , την εκκίνηση, το κλείσιμο και την και διενέργεια, μιας εσωτερικής δοσοληψίας που εμπεριέχει μια διπλή λειτουργία.



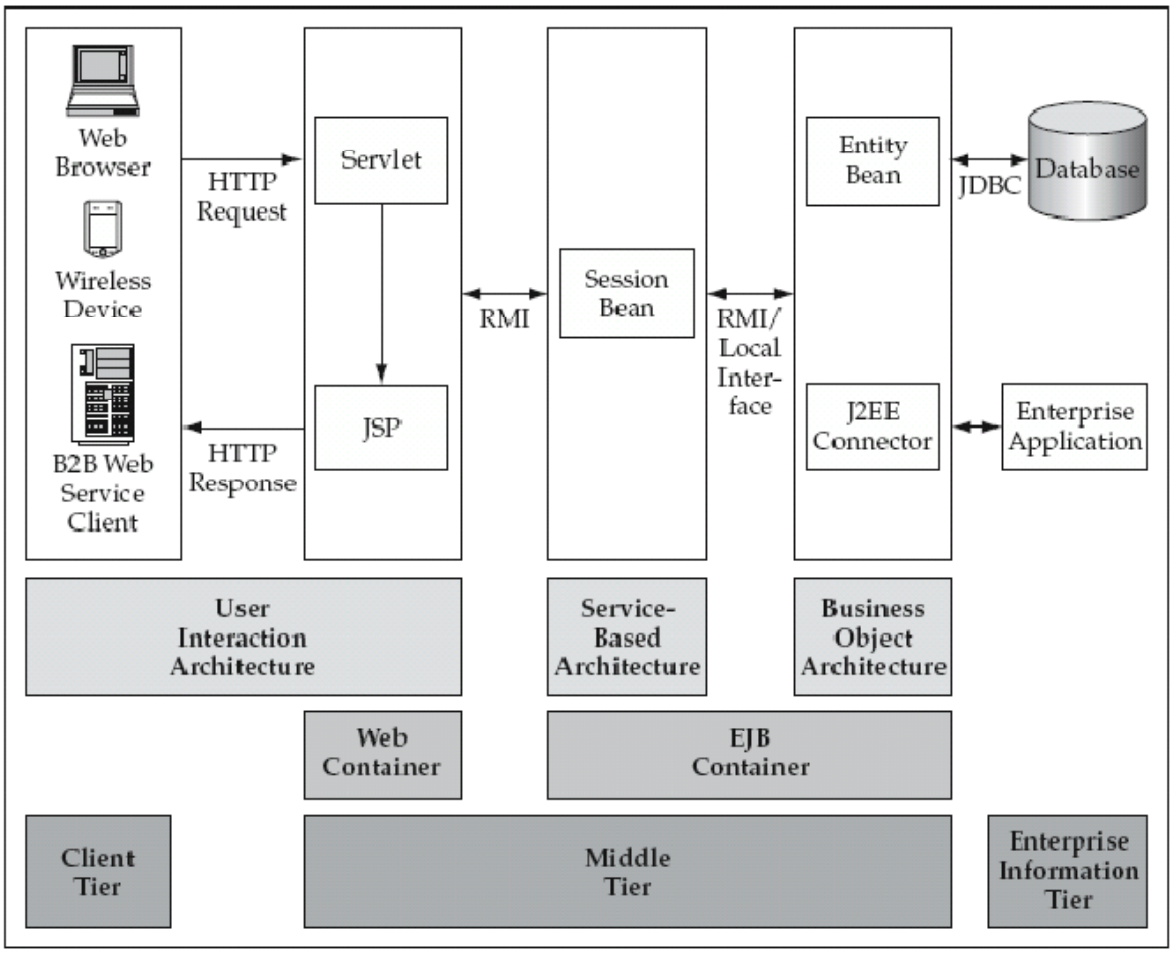
Σχήμα 2.11 Τα APIs της πλατφόρμας J2EE

2.8 Εξυπηρετητές εφαρμογών J2EE

Η προδιαγραφή J2EE ορίζει ένα σύνολο από διεπαφές και κλάσεις. Η κατασκευαστές λογισμικού παρέχουν εφαρμογές για τις παραπάνω διεπαφές, οι οποίες ακολουθούν πιστά τις προδιαγραφές του J2EE . Οι εφαρμογές αυτές ονομάζονται εξυπηρετητές εφαρμογών (application servers) J2EE. Οι εξυπηρετητές εφαρμογών J2EE προσφέρουν την υποδομή για την παροχή υπηρεσιών όπως νημάτωσης (threading), πολιτικής εναλλαγής πόρων (pooling) και διαχείρισης δοσοληψιών (transaction management) , ώστε ο προγραμματιστής, όσο είναι δυνατόν, να μην ασχολείται με την ανάπτυξη διαχειριστικού κώδικα και να ασχολείται κυρίως με την υλοποίηση του επιχειρησιακού κομματιού της εφαρμογής .



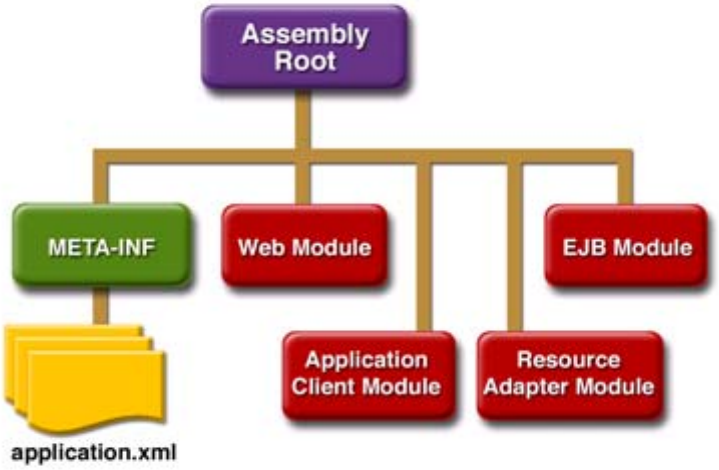
Σχήμα 2.12 Επικοινωνίες εξυπηρετητή J2EE



Σχήμα 2.13 Βασική αρχιτεκτονική J2EE

2.9 Συσκευασία εφαρμογών

Μια εφαρμογή J2EE πρέπει να βρίσκεται σε ένα επιχειρησιακό αρχείο (Enterprise Archive file-EAR) το οποίο έχει την επέκταση .EAR. Όταν δημιουργεί κανείς μια εφαρμογή J2EE , αυτή αποτελείται συνήθως, από μια εφαρμογή ιστού (web) και από μια enterprise bean εφαρμογή (ejb). Κατά τη μεταγλώττιση των παραπάνω εφαρμογών, στον εξυπηρετητή εφαρμογών (application server), δημιουργούνται αυτόματα, ένα αρχείο με επέκταση .WAR (Web Archive) και ένα EJB-JAR (EJB Java Archive) αρχείο τα οποία ενθυλακώνονται σε ένα EAR (Enterprise Archive) αρχείο, το οποίο τοποθετείται εντός του εξυπηρετητή εφαρμογών. Δεν χρειάζεται επιπλέον κώδικας, είναι μονάχα ζήτημα δόμησης (ή συσκευασίας) των διαφόρων προδιαγραφών J2EE μέσα σε EAR αρχεία J2EE.



Σχήμα 2.14 Δομή αρχείων μιας επιχειρησιακής εφαρμογής

Σχήμα 2.15 Δομή αρχείων ενός EAR αρχείου συσκευασίας μιας επιχειρησιακής εφαρμογής

2.10 Σύνοψη

Η πλατφόρμα J2EE προσφέρει μια καλά μελετημένη αρχιτεκτονική για ανάπτυξη σύνθετων επιχειρησιακών εφαρμογών.

Ο συνδυασμός τεχνολογιών J2EE όπως αυτών των EJBs, servlet, και JSP – και των περιληπτικών APIs (JDBC, JNDI, JMS, κτλ.) δίνει στου χρήστες ποικίλα πλεονεκτήματα. Η ανάπτυξη ακολουθώντας την J2EE προδιαγραφή, απλοποιεί τη συνολική ανάπτυξη μεγάλης κλίμακας κατανεμημένων εφαρμογών. Κάποια από τα κρίσιμα σημεία που απλοποιούνται μέσω της προδιαγραφής J2EE περιλαμβάνουν, κατανεμημένες εφαρμογές δια μέσου πολλαπλών διαδικασιών και επεξεργαστών, ασφάλεια, δοσοληψίες, διαχείριση μόνιμων δεδομένων, και εγκατάσταση της εφαρμογής (στον εξυπηρετητή).

3

Enterprise Java Beans

3.1 Εισαγωγή

Ο όρος επιχειρησιακός (enterprise) είναι γενικά ένας από τους πιο πολυχρησιμοποιημένους μέχρι τώρα, στο χώρο της ανάπτυξης λογισμικού. Επίσης κάποιος που ασχολείται με την ανάπτυξη επιχειρησιακών εφαρμογών, τον απασχολεί συνεχώς μια έννοια, αυτή της πληροφορίας. Οι επιχειρησιακές εφαρμογές ορίζονται από την ανάγκη συλλογής, μεταφοράς και ανάκτησης μεγάλου όγκου πληροφορίας. Και φυσικά αυτού του είδους η πληροφορία δεν μπορεί να υπάρχει κάπου, έτσι απλώς γενικά . Σήμερα η αποθήκευση και η ανάκτηση τέτοιας πληροφορίας αποτελεί υπόθεση πολλών εκατομμυρίων ευρώ και οι ανάγκες από τα συνεχώς αυξανόμενα επιχειρησιακά πληροφοριακά συστήματα τα επόμενα χρόνια προβλέπεται ότι θα σημειώσουν έκρηξη.

Αρκετοί τρόποι για οργάνωση-διαχείριση των αποθηκευμένων δεδομένων (persisting data) των βάσεων δεδομένων έχουν έρθει και παρέρθει, και καμιά έννοια τα προηγούμενα χρόνια δεν είχε την ισχύ και τη διάρκεια, όπως αυτής των σχεσιακών βάσεων δεδομένων. Η εισαγωγή αυτού του τρόπου οργάνωσης, είχε ως αποτέλεσμα την αποθήκευση των δεδομένων με αυτόν τον τρόπο από το μεγαλύτερο πλήθος των επιχειρήσεων. Θεωρείται στις μέρες μας το σημείο αρχής για κάθε επιχειρησιακή εφαρμογή με διάρκεια ζωής που θα συνεχίζεται ακόμα και μετά το ξεπέρασμα του σκοπού της ίδιας της εφαρμογής για τον οποίο είχε φτιαχτεί αρχικά.

Έτσι η κατανόηση των σχεσιακών δεδομένων αποτελεί κλειδί για την επιτυχή ανάπτυξη των επιχειρησιακών εφαρμογών. Πρώτιστη μέριμνα από την σκοπιά της ανάπτυξής εφαρμογών είναι οι αναπτυσσόμενες εφαρμογές να δουλεύουν καλά με τα διαφορά συστήματα βάσεων δεδομένων. Ιδιαίτερα για τη γλώσσα Java, μέρος της επιτυχίας της μπορεί να χαρακτηριστεί η ευρέως υιοθέτηση από τη γλώσσα, χαρακτηριστικών για την κατασκευή επιχειρησιακών συστημάτων

δεδομένων. Για τους εμπορικούς ιστοτόπους , οι εφαρμογές Java αποτελούν την καρδία της επιχειρησιακής ανάπτυξης δεδομένων.

Παρόλη την επιτυχία της πλατφόρμας Java στην λειτουργία της με τα συστήματα δεδομένων, παρόλα αυτά υποφέρει από ένα πρόβλημα. Η μετακίνηση των δεδομένων από και προς τα πίσω, σε ένα σύστημα βάσεων δεδομένων, αποτελεί περισσότερη (αρκετά επίπονή) εργασία από αυτό που είναι πραγματικά να γίνει. Οι προγραμματιστές σε γλώσσα Java ξοδεύουν αρκετό χρόνο για τη μετατροπή γραμμών και στηλών , των πινάκων μιας βάσης δεδομένων, σε αντικείμενα, ή στο να βρουν από μόνοι τους τα κατάλληλα frameworks (πλατφόρμες ανάπτυξης, οι οποίες παρέχουν γενικά αρχιτεκτονικά εκμαγεία τα οποία χρησιμοποιούνται για το κτίσιμο , εξειδικευμένου τομέα εφαρμογών) τα οποία θα προσπαθήσουν να κρύψουν τη φυσική υπόσταση της βάσης δεδομένων ώστε να διευκολύνουν αυτόν που θα αναπτύξει στη συνέχεια την εφαρμογή.

Η λύση στο παραπάνω πρόβλημα ήδη υπάρχει. Τα τελευταία χρόνια προτυποποιήθηκε και χρηματοδοτείται διπλά, τόσο για εμπορικούς σκοπούς όσο και για διερεύνηση από την κοινότητα ανοικτού κώδικα. Η απάντηση στο πρόβλημα, είναι το Java Persistence API το οποίο δημιουργήθηκε για να δώσει μια μεγάλη ώθηση στον τρόπο που διαχειριζόμαστε την καταχώριση δεδομένων στις βάσεις δεδομένων (persistence) με την γλώσσα Java. Αν και οι αντικειμενοστραφής αντιστοίχηση είναι το κύριο στοιχείο του API, ωστόσο προσφέρει λύσεις για τις αρχιτεκτονικές δυσκολίες που βρίσκει η ολοκλήρωση της καταχώρισης δεδομένων των βάσεων δεδομένων σε επιχειρησιακές εφαρμογές που συνεχώς επεκτείνονται και μεγαλώνουν. Με τη γέννηση του Java Persistence API ως μέρος του EJB 3.0., για πρώτη φορά οι προγραμματιστές έχουν ένα προτυποποιήμενο τρόπο για τη γεφύρωση του κενού μεταξύ του αντικειμενοστραφούς μοντέλου της γλώσσας, με το σχεσιακό πρότυπο που ακολουθούν τα σχεσιακά συστήματα βάσεων δεδομένων.

3.1 Η γέννηση του EJB

Γύρω στα 1996, που ή Java ενισχύονταν στο χώρο της ανάπτυξής λογισμικού με την εισαγωγή τεχνολογιών (όπως RMI και JTA) οι οποίες κατεύθυναν τις ανάγκες των εφαρμογών μεγάλης κλίμακας, ή ανάγκη για ένα μια πλατφόρμα επιχειρησιακών συστατικών λογισμικού η οποία θα μπορούσε να ενοποιήσει αυτές τις τεχνολογίες και να τις ενσωματώσει κάτω μια προδιαγραφή ανάπτυξης. Το EJB δημιουργήθηκε για να καλύψει αυτή την ανάγκη. Τα 12 επόμενα χρόνια, ενσωμάτωσε αρκετά νέα χαρακτηριστικά (ενώ απέβαλε κάποια άλλα) ενώ ωρίμαζε σε ένα στιβαρό

και προτυποποιήμενο framework για εγκατάσταση και εκτέλεση επιχειρησιακών συστατικών λογισμικού σε ένα κατανεμημένο, πολλαπλών χρηστών περιβάλλον.

3.2 Η προδιαγραφή EJB

Το EJB 3 ορίζεται από την αναφορά JSR 220 ως: Enterprise Java Beans 3. Ενώ οι προηγούμενες εκδόσεις του EJB καταλάμβαναν το χώρο ένος απλού εγγράφου, αυτή η αναφορά JSR καταλαμβάνει τρία έγγραφα. Το πρώτο έγγραφο (ejbcore) παρέχει μια σύνθεση των υψηλού επιπέδου χαρακτηριστικών της νέας έκδοσης, και εστιάζει στο νέο απλοποιημένο πρότυπο ανάπτυξης για την κατασκευή συστατικών λογισμικού EJB. Τα άλλα δύο έγγραφα (simplified και persistence) διευθετούν τις τεχνικές λεπτομέρειες του πυρήνα του επιχειρησιακού bean framework και του persistence μοντέλου, αντίστοιχα. Το περιεχόμενο των τριών αυτών εγγράφων παρατίθεται παρακάτω:

- Το απλοποιημένο EJB 3 API δίνει μια υψηλού επιπέδου θεώρηση του νέου αναπτυξιακού μοντέλου του EJB 3.

- Οι συμβάσεις πυρήνα (core contracts) και οι απαιτήσεις εστιάζουν στα session και στα message-driven beans.

- Το Java Persistence API διευθέτει ζητήματα για τα entities και το persistence framework.

  Αυτό που βγαίνει σαν συμπέρασμα από τα έγγραφα είναι ότι η προδιαγραφή EJB 3 είναι διπλά : μια προδιαγραφή συστατικών λογισμικού και ένα framework.

3.2.1 Η προδιαγραφή συστατικών λογισμικού EJB

Σαν μια προδιαγραφή συστατικών λογισμικού, το EJB ορίζει τρείς τύπους που οι προγραμματιστές μπορούν κατασκευάσουν και να προσαρμόσουν, οι οποίοι είναι οι ακόλουθοι:

- Τα session beans εκτελούν λειτουργίες business υπηρεσιών και διευθύνουν τις δοσοληψίες και τον έλεγχο πρόσβασης.

- Τα message-driven beans (MDBs) καλούνται ασύγχρονα για αποκρίσεις σε εξωτερικά γεγονότα, μέσω της σχέσης τους με μια ουρά μηνυμάτων ή άλλου αντικειμένου.

- Τα entities beans τα οποία είναι αντικείμενα που έχουν μοναδική ταυτότητα και αναπαριστούν τα αποθηκευμένα (persistent) δεδομένα, τα οποία βρίσκονται στη βάση δεδομένων.

3.2.2 Το EJB framework

Το framework των EJB παρέχει το περιβάλλον υποστήριξης μέσα στο οποίο λειτουργούν τα EJB components. Αυτό εμπεριέχει τις συνδιαλλαγές του υποδοχέα (container) και τις υπηρεσίες ασφάλειας, πολιτικής εναλλαγής (pooling) και προσωρινής αποθήκευσης (caching) πόρων, υπηρεσίες κύκλου ζωής των συστατικών λογισμικού, υποστήριξή ταυτοχρονισμού και άλλων. Τα συστατικά λογισμικού EJB προσδιορίζουν τις λεπτομέρειες για το πώς αυτά θα αλληλεπιδράσουν με τον παρεχόμενο υποδοχέα χρησιμοποιώντας ειδικά μεταδεδομένα τα όποια είτε λαμβάνονται από τον υποδοχέα και εφαρμόζονται στη συμπεριφορά των EJB κατά τον χρόνο εκτέλεσης, είτε μεταγλωττίζεται για εκείνη την περίοδο ένα συστατικό λογισμικού EJB που είναι εγκατεστημένο σε έναν υποδοχέα EJB και χρησιμοποιείται για την κατασκευή του προφίλ συμπεριφοράς (wrapping) του.

3.2 Η εξέλιξη της προδιαγραφής EJB

Κάθε φορά που μια καινούρια έκδοση της προδιαγραφής EJB εισαγόταν, έβαζε νέα σημαντικά χαρακτηριστικά προς, την ικανοποίηση των εκάστοτε απαιτήσεων και την υιοθέτηση απ' την πλευρά της των νεοεισερχόμενων τεχνολογιών. Παρακάτω παρατίθεται η εξέλιξη της προδιαγραφής EJB από την στιγμή τη γέννησης της το 1996, με τους πιο σημαντικούς σταθμούς της, από την πρώτη εμπορική της υλοποίηση το 1998 μέχρι τις μέρες μας.

3.2.1 EJB 1.0

Η αρχική έκδοση 1.0, ξεκινά με την υποστήριξή για υπηρεσιών stateful και stateless αντικειμένων, που ονομάζονται session beans, και προαιρετική υποστήριξή για αποθηκευμένα

(persistent) αντικείμενα, τα οποία ονομάζονται entity beans. Επίσης παρείχε υποστήριξη φορητότητας και διαχείριση απομακρυσμένων κλήσεων μέσω απομακρυσμένης διεπαφής για την πρόσβαση στα EJBs, όχι όμως με τρόπο αποδοτικό, αφού υπήρχε αρκετή εξάρτηση από την εκάστοτε απομακρυσμένη υποδομή.

3.2.2 EJB 1.1

Η επόμενη έκδοση , 1.1, επέβαλε την υποστήριξή για entity beans μεταξύ των κατασκευαστών, και εισήγαγε το αρχείο XML περιγραφέα διάταξης (deployment descriptor) το οποίο αντικατέστησε τα μέχρι πρότινος μεταδεδομένα που περιέγραφαν την αρχιτεκτονική της υποδομής μιας EJB εφαρμογής.

3.2.3 EJB 2.0

Το EJB 2.0 εισήγαγε μια τοπική διεπαφή (local interface) για την πρόσβαση στα EJB , για πελάτες που εκτελούν εντός του υποδοχέα J2EE. Επίσης εισήγαγε τα message-driven bean (MDB) για υποστήριξη ασύγχρονης επικοινωνίας και την Enterprise JavaBeans Query Language (EJB QL) για τη δυνατότητα δημιουργίας SQL ερωτημάτων στη βάση δεδομένων από τα στιγμιότυπα των entity beans.

3.2.4 EJB 2.1

Το EJB 2.1 πρόσθεσε υποστήριξη για υπηρεσίες ιστού (web services) και επέκτεινε την EJB QL.

3.2.5 EJB 3.0

Για το EJB 3, η πιο αξιοσημείωτη αλλαγή ήταν ότι τα entity beans έχουν αντικατασταθεί από συμβατικές κλάσεις Java (POJOs - plain old Java objects) οι οποίες μπορούν να εκτελούνται και έξω από τον υποδοχέα EJB, και έτσι να μην απαιτούν ειδικές διεπαφές ή ειδικό EJB κώδικά σε μια entity κλάση. Τα session beans δεν χρειάζονται πλέον home ή συστατικών λογισμικού ειδικές διεπαφές EJB, αν και συνεχίζουν να υποστηρίζουν λογικά απομακρυσμένες και μη

απομακρυσμένες διεπαφές. Επίσης χρησιμοποιούνται οι επισημειώσεις (annotations) JDK 5.0, για την εισαγωγή μέσα στις entities κλάσεις των μεταδεδομένων περιγραφής των entities, απλοποιώντας κατά πολύ την περιγραφή , και κάνοντας προαιρετική τη μέχρι πρότινος χρήση σχημάτων περιγραφής XML (XML deployment descriptors). Για τα entities, η home διεπαφή αντικαταστάθηκε από έναν διαχειριστή οντοτήτων (entity manager) για τη διαχείριση των λειτουργιών του κύκλου ζωής των entities, ο οποίος αναλαμβάνει και την εκτέλεση ερωτημάτων SQL στη βάση δεδομένων.
Όλες αυτές οι αλλαγές, μαζί και με αρκετές άλλες συνέστησαν την καινούργια φιλοσοφία του EJB 3, που επικεντρώθηκε στην απλοποίηση του προτύπου ανάπτυξης.

3.3 Χαρακτηριστικά του πυρήνα της τεχνολογίας EJB

3.3.1 Επεκτασιμότητα

Μεγάλες εφαρμογές απαιτούν την δυνατότητα επέκτασης τους, όταν το φορτίο των πελατών αυξάνεται. Ο εξυπηρετητής EJB επιστρατεύει πολιτική εναλλαγή πόρων για να αυξήσει την επαναχρησιμοποίηση των αντικειμένων, αξιοποιεί μια προσωρινή μνήμη για την αποφυγή επαναλαμβανόμενων ερωτημάτων ή δημιουργίας ίδιων αντικειμένων, και υλοποιεί μια αισιόδοξη πολιτική κλειδώματος στο μεσαίο στρώμα για την μείωση του φορτίου του συστήματος διαχείρισης σχεσιακής βάσης δεδομένων (Relational Database Management System -RDBMS) και την αποφυγή προβλημάτων λόγω ταυτόχρονης πρόσβασης στα δεδομένα.

3.3.2 Διαχείριση δοσοληψιών

Το Java Transaction API (JTA) ορίζει μια προδιαγραφή API για κατανεμημένες δοσοληψίες(transactions), και ο EJB server λειτουργεί ως ένας JTA διαχειριστής δοσοληψιών για τα EJBs, έχοντας συμπεριφορά όπως αυτή ορίζεται από την προδιαγραφή EJB.

3.3.3 Ασφάλεια πολλών χρηστών

Μια μέθοδος ελέγχου του επιπέδου πρόσβασης πρέπει να ορίζεται δηλώνοντας τα κατάλληλα EJBs, επιβάλλοντας το επίπεδο δικαιωμάτων χρήστη (user) και ρόλου χρήστη (role) που έχει οριστεί από τους διαχειριστές του εξυπηρετητή εφαρμογών.

3.3.4 Φορητότητα

Παρόμοιων προδιαγραφών enterprise beans μπορούν να εγκαθίστανται σε κάθε εξυπηρετητή εφαρμογών που υλοποιεί EJB, τουλάχιστον θεωρητικώς. Στην πράξη (και ειδικά για τις εκδόσεις , πριν το EJB 3), οι κατασκευαστές παρείχαν τους δικούς τους ορισμούς μεταδεδομένων στους προγραμματιστές , για την εγκατάσταση των enterprise bean, εξαρτώντας την ίδια στιγμή την εφαρμογή από την εκάστοτε υποδομή του κάθε κατασκευαστή. Μετά την επικράτηση του EJB3, έγινε μεγάλη προσπάθεια να ενσωματωθούν αυτά τα διαφορετικά χαρακτηριστικά, ώστε τα EJB που υλοποιούνται σήμερα να είναι περισσότερο φορητά σε σχέση με αυτά του παρελθόντος.

3.3.5 Δυνατότητα επαναχρησιμοποίησης (Reusability)

Τα συστατικά λογισμικού EJB είναι χαλαρής σύζευξης. Ένα EJB μπορεί να είναι επαναχρησιμοποιήσιμο και να συσκευάζεται για πολλές εφαρμογές, ώστε να μπορεί να επικοινωνήσει , η να έχει πρόσβαση , σε άλλα εξαρτώμενα EJBs.

3.3.6 Δυνατότητα αποθήκευσης (Persistence)

Τα entity bean όπως έχει ειπωθεί παραπάνω, στο EJB 3 έχουν αντικατασταθεί από συμβατικές κλάσεις (plain old Java object - POJO) - επονομαζόμενες απλά ως entity – που απεικονίζουν περιοχές καταχωρημένων αντικειμένων στη βάση δεδομένων τα οποία έχουν μοναδικές ταυτότητες. Μια entity κλάση ανταποκρίνεται σε έναν πίνακα βάσης δεδομένων, και κάθε στιγμιότυπο του entity αντικειμένου σε μια απλή γραμμή στον συγκεκριμένο πίνακα.

3.4 Τύποι των Enterprise Java Bean

3.4.1 Σε τι βοηθάνε τα Enterprise Java Bean

Τα EJBs αρχικά προτάθηκαν σαν μια ανεξάρτητη προδιαγραφή από τη Sun Micro Systems, με σκοπό να απλοποιήσουν την ανάπτυξη εφαρμογών απ’την πλευρά του εξυπηρετητή, μέσω της παροχής ενός πολύ μεγάλου πλήθους υπηρεσιών που δημιουργήθηκαν για να δώσουν λύση σε κρίσιμα ζητήματα που αφορούσαν την ανάπτυξη εφαρμογών. Γι’αυτό το σκοπό αναπτύχθηκαν

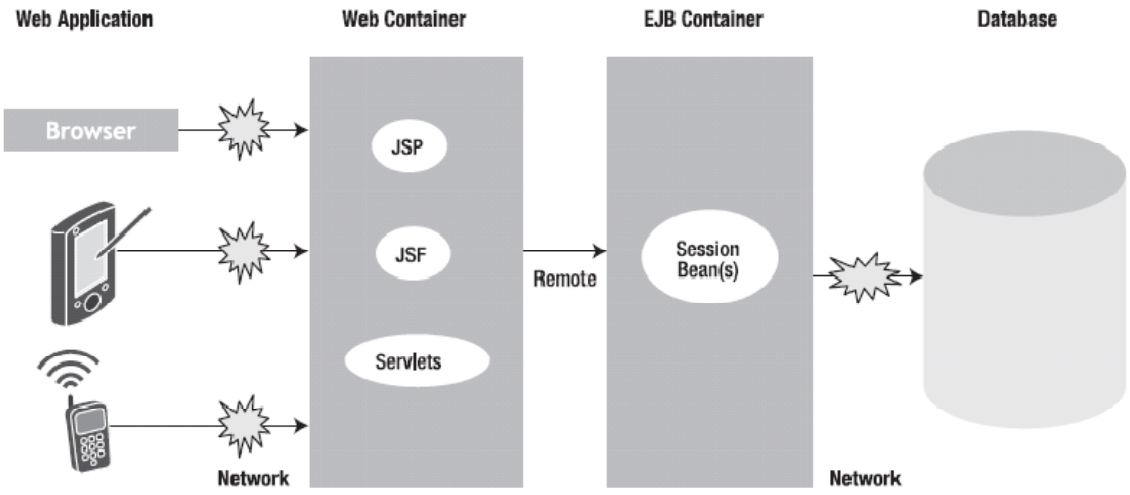
τρείς τύποι EJB όπως θα δούμε παρακάτω, με τον κάθε τύπο να επιτελεί ένα συγκεκριμένο ρόλο στην προδιαγραφή EJB.

3.4.2 Session beans

Τα session bean είναι μια τεχνολογία συστατικών λογισμικού που σχεδιάστηκε για την ενσωμάτωση υπηρεσιών του επιχειρησιακού (business) κομματιού της εφαρμογής. Οι υποστηριζόμενες λειτουργίες ορίζονται με τη χρήση μιας συνηθισμένης διεπαφής Java, που αναφέρεται ως επιχειρησιακή (business) διεπαφή του session bean, την οποία οι πελάτες χρησιμοποιούν για να αλληλεπιδράσουν με το bean. Η υλοποίηση του bean είναι λίγο μεγαλύτερη από μια κλάση Java η οποία υλοποιεί την επιχειρησιακή διεπαφή.

Η ονομασία του session bean έχει να κάνει με τον τρόπο που οι πελάτες έχουν πρόσβαση και αλληλεπίδραση με αυτό. Όταν ένας client αποκτά μια αναφορά σε ένα session bean από τον εξυπηρετητή, ξεκινά μια περίοδος (session) με αυτό το bean και τότε επιτρέπεται η κλήση κάποιας επιχειρησιακής λειτουργίας με αυτό.

Τα session bean είναι συστατικά λογισμικού Java τα οποία εκτελούνται είτε σε αυτόνομους (stand-alone) υποδοχείς EJB ή σε υποδοχείς EJB οι οποίοι είναι μέρος ενός εξυπηρετητή εφαρμογών που ακολουθεί την προδιαγραφή της πλατφόρμας της Java Enterprise Edition. Αυτά τα συστατικά λογισμικού Java τυπικά χρησιμοποιούνται για προτυποποιήσουν μια συγκεκριμένη εργασία του χρήστη ή μια περίπτωση χρήσης, όπως π.χ. εισαγωγή πληροφορίας πελάτη ή την υλοποίηση μιας διαδικασίας η οποία διατηρεί μια κατάσταση «συνομιλίας» με μια εφαρμογή πελάτη. Τα session bean μπορούν να κρατάνε την επιχειρησιακή λογική για αρκετούς τύπους εφαρμογών, όπως αυτές των ανθρωπίνων πόρων, εισαγωγής παραγγελιών και εφαρμογών αναφοράς δαπανών.



Σχήμα 3.1 Ο ρόλος των session beans σε μια εφαρμογή ιστού αρχιτεκτονικής 3 στρωμάτων

Κατά την ολοκλήρωση μιας αίτησης της εφαρμογής ιστού συμβαίνουν τα ακόλουθα:

Μια ενέργεια του χρήστη εγκαθιστά μια σύνδεση με το session bean που εκτελείται μέσα στον υποδοχέα EJB χρησιμοποιώντας πρωτόκολλο RMI (remote method invocation) . Η εφαρμογή του πελάτη καλεί μια ή περισσότερες μεθόδους του session bean. Το session bean αφού δεχτεί αίτηση , την επεξεργάζεται και επικυρώνει τα δεδομένα – αλληλεπιδρώντας με βάσεις δεδομένων, επιχειρησιακές εφαρμογές, συμβατικά συστήματα, κτλ.- για την εκτέλεση μιας επιχειρησιακής λειτουργίας ή εξεργασίας. Τέλος το session bean στέλνει πίσω στην εφαρμογή ιστού μιαν απόκριση είτε μέσω μιας συλλογής δεδομένων ή απλών αντικειμένων (objects) που περιέχει αναγνωρίσιμα μηνύματα. Η εφαρμογή ιστού μορφοποιεί την απόκριση καταλλήλως , και τη στέλνει στην συσκευή που έκανε την αίτηση (browser, PDA, telnet, κτλ.).
Υπάρχουν δύο τύποι session bean, τα stateless τα οποία δεν διατηρούν την κατάσταση τους με την πλευρά του πελάτη της εφαρμογής και τα stateful που διατηρούν την κατάσταση τους με την πλευρά του πελάτη της εφαρμογής π.χ. κάθε ξεχωριστό στιγμιότυπο του session bean συσχετίζεται με μια συγκεκριμένη αίτηση –κλήση του πελάτη.

3.4.2.1 Stateless session beans

Όπως έχει αναφερθεί ένα stateless session bean είναι φτιαγμένο για να φέρει εις πέρας, τους στόχους μιας λειτουργίας εντός του χρόνου μιας απλής μεθόδου. Τα stateless bean επιτρέπεται να υλοποιούν πολλές επιχειρησιακές (business) λειτουργίες, άλλα κάθε μέθοδος δεν μπορεί να θεωρεί ότι κάποια άλλη είχε κληθεί πριν από αυτή. Αυτό μπορεί να φαίνεται ως περιορισμός των stateless bean, άλλα είναι κάτι που συνηθίζεται αρκετά στην υλοποίηση business υπηρεσιών. Διαφορετικά τα stateful session bean, τα οποία είναι καλά για αποθήκευση κατά την διάρκεια μιας συνδιαλλαγής (όπως στην περίπτωση του καλαθιού αγορών σε μια διαδικτυακή εφαρμογή λιανικής πώλησης) , τα stateless session bean είναι σχεδιασμένα νε φέρουν εις πέρας ανεξάρτητες λειτουργίες πολύ πιο αποδοτικά. Ο λόγος είναι ότι σε κάποιες δεδομένες χρονικές στιγμές ο υποδοχέας EJB αποθηκεύει την κατάσταση των stateful session bean σε κάποιο μόνιμο μέσο αποθήκευσης δεδομένων, πράγμα που δεν συμβαίνει στα stateless session bean, όπου δεν αποθηκεύεται ποτέ η κατάσταση τους σε κάποιο μόνιμό μέσο αποθήκευσης. Κατά συνέπεια λόγω της μη αναφοράς των stateless session bean σε κάποιο εξωτερικό μέσο αποθήκευσης και της μικρότερης χρονικής απόκρισης που αυτό συνεπάγεται, παρέχουν καλύτερη απόδοση από τα stateful bean.
Κατά το χρόνο μη κλήσης κάποιας μεθόδου, όλα τα στιγμιότυπα ενός stateless bean είναι ισότιμα επιτρέποντας στον υποδοχέα EJB να αποδίδει ένα στιγμιότυπο σε κάθε ξεχωριστό πελάτη. Έτσι τα

stateless session bean μπορούν να υποστηρίξουν μεγάλο πλήθος πελατών με ελάχιστο κόστος για τους συνολικούς πόρους του εξυπηρετητή.
Μόνο τα stateless session bean μπορούν να υλοποιούν υπηρεσίες ιστού (web services), και όχι οι άλλοι τύποι των EJBs.

Τα stateless session bean αποτελούνται από τα ακόλουθα στοιχεία:

- Τις επιχειρησιακές διεπαφές (business interface), τα οποία περιέχουν τις δηλώσεις των επιχειρησιακών μεθόδων οι οποίες πρέπει να είναι ορατές από τις εφαρμογές πελάτη.

- Μια κλάση bean, η οποία περιέχει την υλοποίηση της επιχειρησιακής μεθόδου που εκτελείται από αυτό.

3.4.2.1.1 Η κλάση bean

Μια κλάση stateless session bean είναι μια κοινή κλάση Java η οποία περιέχει μια επισημείωση (annotation) κλάσης τύπου Stateless: @Stateless. Παράδειγμα κλάσης bean φαίνεται στη λίστα (Λίστα 3.1) του παρακάτω κώδικα.

Λίστα 3.1 : SearchFacadeBean.java

```
import javax.ejb.Stateless;

@Stateless
public class SearchFacadeBean implements SearchFacadeRemote, SearchFacadeLocal {
public SearchFacadeBean() {
                          }
…
}
```

3.4.2.1.2 Η επιχειρησιακή διεπαφή (business interface)

Οι επιχειρησιακές διεπαφές stateless session bean είναι συνηθισμένες διεπαφές η οποίες δεν επέκταση κάποιων άλλων ειδικών διεπαφών EJB.
Οι επιχειρησιακές διεπαφές μπορούν να χρησιμοποιούν επισημειώσεις (annotation) , όπως περιγράφεται παρακάτω:

• Η επισημείωση @Remote χρησιμοποιείται για να επισημειώσει την απομακρυσμένη επιχειρησιακή διεπαφή (remote business interface).

• Η επισημείωση @Local χρησιμοποιείται για να επισημειώσει την τοπική επιχειρησιακή διεπαφή (local business interface).

Στις παρακάτω λίστες (Λίστα 3.2 και 3.3) κώδικα παρουσιάζονται αντιστοίχως, παραδείγματα ορισμού απομακρυσμένης και τοπικής επιχειρησιακής διεπαφής .

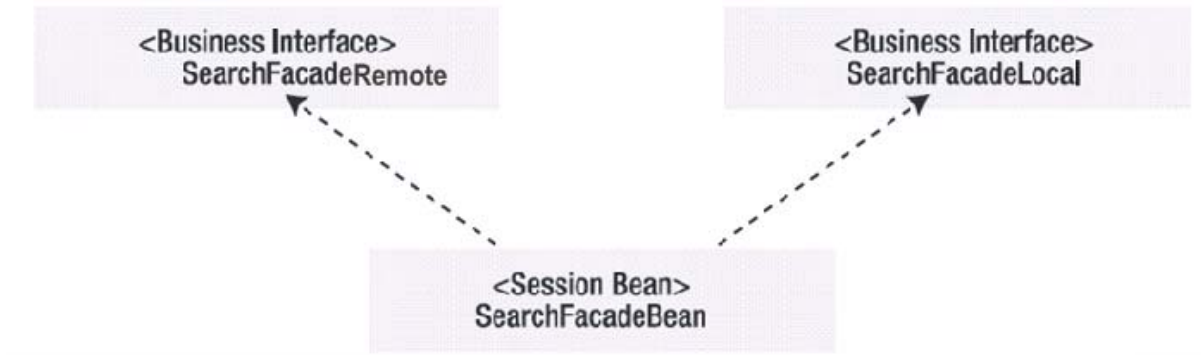
Λίστα 3.2 : SearchFacade.java

```
import java.util.List;
import javax.ejb.Remote;

@Remote
public interface SearchFacadeRemote {
List wineSearch(String wineType);
}
```

Λίστα 3.3 : SearchFacadeLocal.java

```
import java.util.List;
import javax.ejb.Local;

@Local
public interface SearchFacadeLocal {
List wineSearch(String wineType);
}
```

Το SearchFacade session bean έχει διπλά, απομακρυσμένη και τοπική διεπαφή, όπως φαίνεται στο παρακάτω σχήμα (Σχήμα 3.2)



Σχήμα 3.2 Οι επιχειρησιακές διεπαφές του SearchFacade session bean

Οι επιχειρησιακές (business) μέθοδοι

Οι μέθοδοι που υλοποιούνται στην bean κλάση πρέπει να ανταποκρίνονται στις επιχειρησιακές μεθόδους που έχουν δηλωθεί στην απομακρυσμένη και τοπική διεπαφή. Ένα παράδειγμα επιχειρησιακής μεθόδου, για την κλάση SearchFacadeBean, φαίνεται στη λίστα (Λίστα 3.4) του παρακάτω κώδικα.

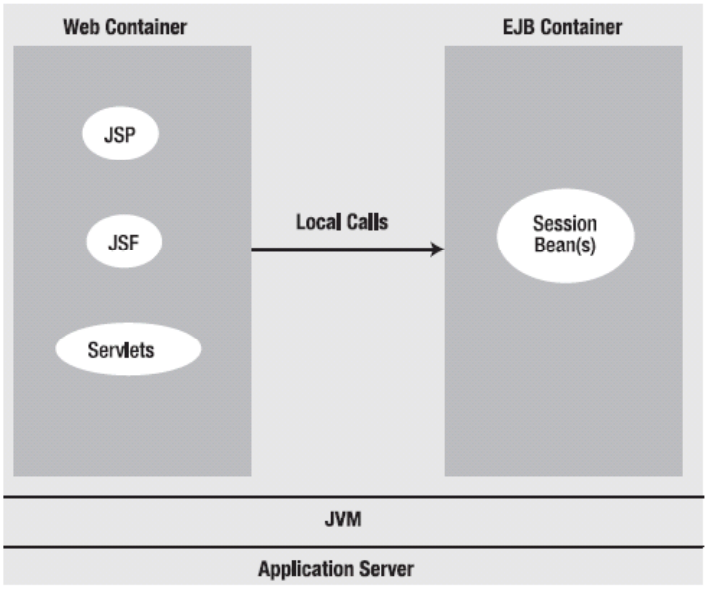
Λίστα 3.4 : SearchFacadeBean.java

```
import java.util.ArrayList;
import java.util.List;
import javax.ejb.Stateless;

@Stateless
public class SearchFacadeBean implements SearchFacadeRemote, SearchFacadeLocal {
public SearchFacadeBean() { }

public List wineSearch(String wineType) {
List wineList = new ArrayList();
if (wineType.equals("Red"))
{
wineList.add("Bordeaux");
wineList.add("Merlot");
wineList.add("Pinot Noir");
}
else if (wineType.equals("White"))
{
wineList.add("Chardonnay");
```

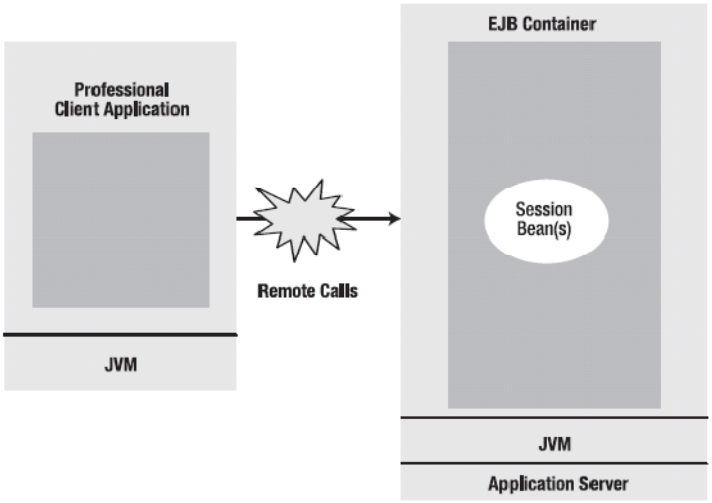
```
}
return wineList;
}
}
```

3.4.2.1.3 Τα Session Beans απ'την πλευρά του πελάτη

Ένα session bean μπορεί να ιδωθεί ως μια λογική επέκταση ενός προγράμματος πελάτη η μιας εφαρμογής, όπου αυτό θα πραγματοποιεί το μεγαλύτερο μέρος της λογικής και της επεξεργασίας δεδομένων για την εφαρμογή. Μια εφαρμογή πελάτη, τυπικά έχει πρόσβαση στο session αντικείμενο διαμέσου των διεπαφών που βλέπει ο πελάτης, οι οποίες είναι οι επιχειρησιακές διεπαφές για τις οποίες αναφερθήκαμε παραπάνω. Αν η αρχιτεκτονική της εφαρμογής απαιτεί ή εφαρμογή πελάτη (π.χ. μια εφαρμογή ιστού) να τρέχει σε άλλη Java Virtual Machine (JVM) από αυτή που εκτελούνται τα session bean μέσα σε έναν υποδοχέα EJB, τότε απαιτείται χρήση απομακρυσμένης διεπαφής. Οι ξεχωριστές JVM μπορεί να βρίσκονται στην ίδια φυσική μηχανή η σε ξεχωριστές. Εάν η αρχιτεκτονική της εφαρμογής πρόκειται να χρησιμοποιήσει την ίδια JVM διπλά, και για την εφαρμογή πελάτη και για τα session bean, τότε απαιτείται χρήση τοπικής διεπαφής. Φυσικά είναι πιθανό η αρχιτεκτονική μιας εφαρμογής να χρειάζεται και τις δύο διεπαφές, απομακρυσμένη και τοπική.



Σχήμα 3.3 Χρήση των τοπικών διεπαφών (local interface) των session beans από έναν πελάτη σε μια εφαρμογή ιστού



Σχήμα 3.4 Χρήση των απομακρυσμένων διεπαφών (remote interface) των session από έναν απομακρυσμένο από έναν πελάτη σε μια εφαρμογή ιστού

Ο παρακάτω πίνακας (Πίνακας 3.1) παρουσιάζει τους λόγους που κάποιος μπορεί να έχει για να επιλέξει μεταξύ τοπικού (local) και απομακρυσμένου (remote) πελάτη.

Πίνακας 3.1 : Λόγοι για επιλογή μεταξύ τοπικού (local) και απομακρυσμένου (remote) πελάτη

Απομακρυσμένος	**Τοπικός**
Χαλαρή ζεύξη μεταξύ bean και πελάτη	Μικρή δαπάνη πόρων κόστος για την πρόσβαση σε ένα συστατικό λογισμικού
Ανεξαρτησία θέσης	Εξάρτηση θέσης
Δαπανηρές απομακρυσμένες κλήσεις	Πρέπει να συνδυάζεται με το bean
Τα αντικείμενα πρέπει να είναι σε σειροποιημένα (serialized) για να μεταδοθούν	Δεν απαιτείται
Τα αντικείμενα προσπελάζονται κατά τιμή	Τα αντικείμενα προσπελάζονται κατά αναφορά

Τρόποι εύρεσης της επιχειρησιακής διεπαφής του session bean

Ένας πελάτης, πριν καλέσει μια μέθοδο ενός session bean , πρέπει να έχει αποκτήσει την επιχειρησιακή διεπαφή του session bean. Η εύρεση του γίνεται με δύο τρόπους, είτε με dependency injection ή μέσω JNDI.

Dependency injection

Χρησιμοποιώντας dependency injection, μπορούμε να πάρουμε την επιχειρησιακή διεπαφή π.χ. απ'το SearchFacade session bean με τον ακόλουθη λίστα (Λίστα 3.5) κώδικα:

Λίστα 3.5 : Λήψη της SearchFacade επιχειρησιακής διεπαφής με dependency injection

```
@EJB SearchFacade searchFacade;
```

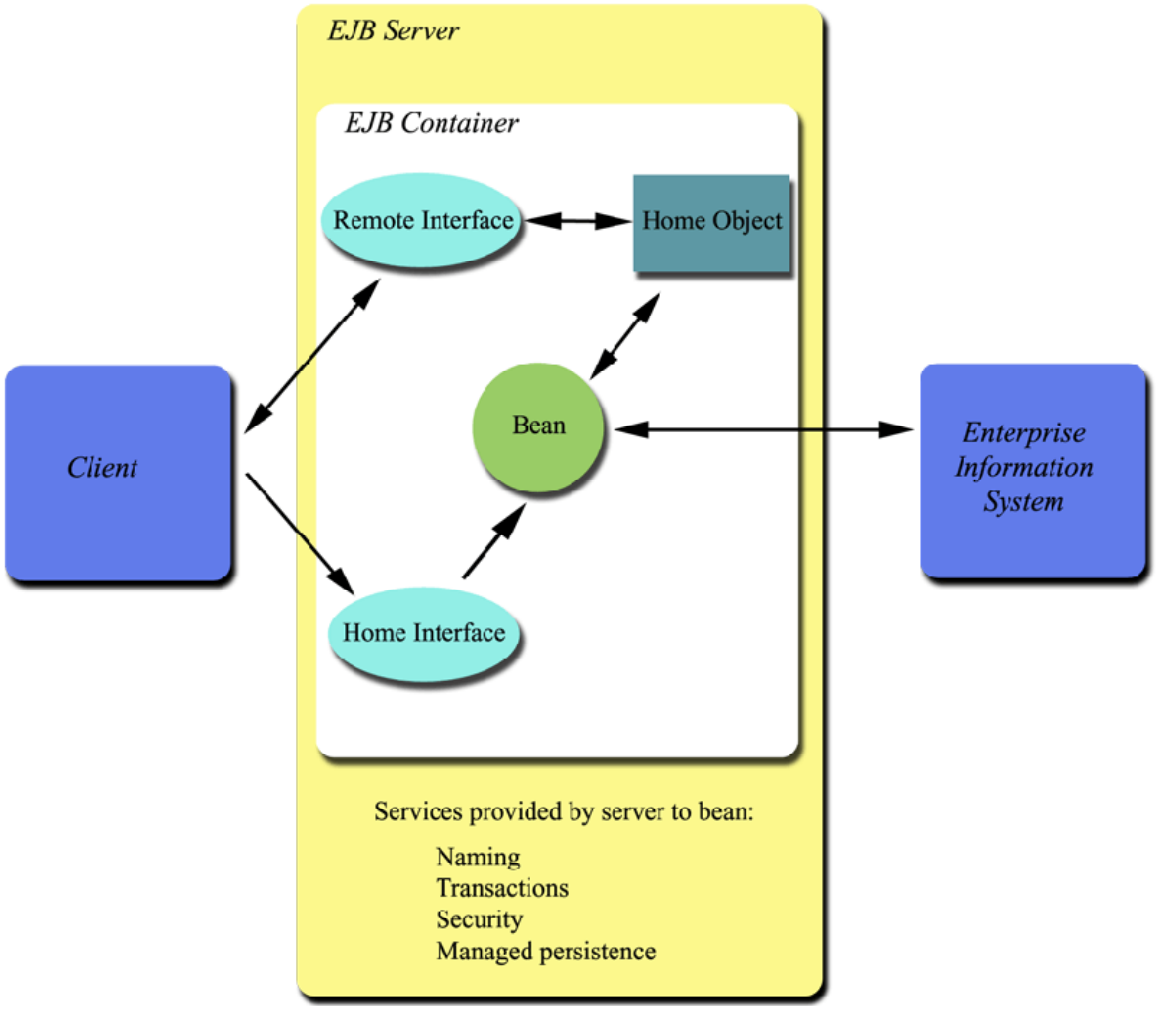
JNDI

Εάν η πρόσβαση στο session bean είναι remote, ο client μπορεί να κάνει χρήση JNDI όταν το περιεχόμενο της διεπαφής έχει ληφθεί με τις σωστές παραμέτρους περιβάλλοντος (environment properties).
Οι τοπικοί πελάτες, επίσης, μπορούν να κάνουν χρήση JNDI, αλλά ο τρόπος του dependency injection έχει πιο απλό κώδικα.
Παράδειγμα (Λίστα 3.6) μεθόδου με JNDI που επιστρέφει την απομακρυσμένη διεπαφή :

Λίστα 3.6 : Λήψη του SearchFacade με μέθοδο που χρησιμοποιεί JNDI

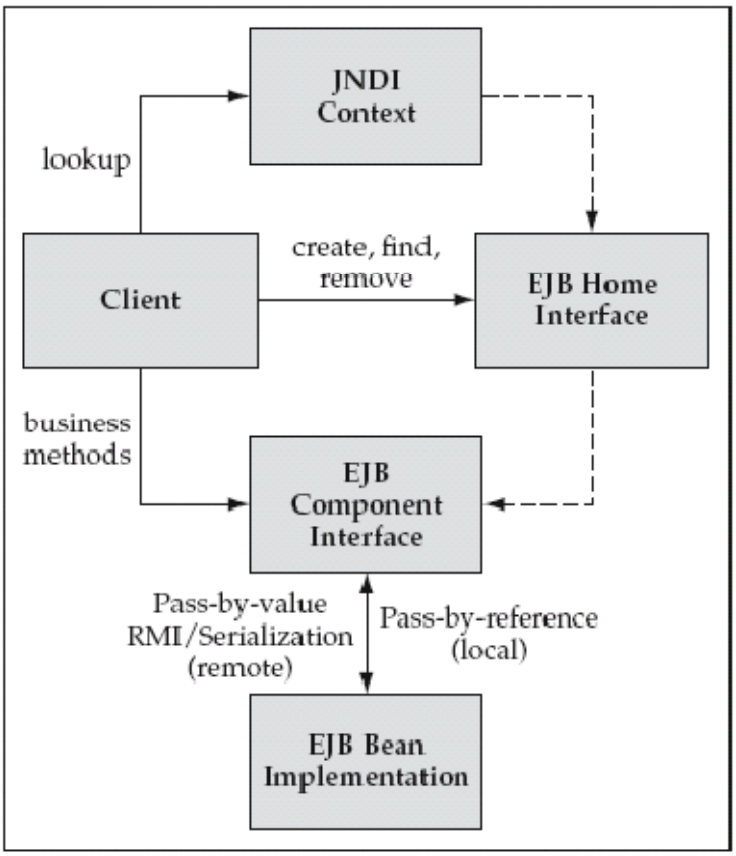
```
SearchFacade  lookup SearchFacade() {

            Context c = new InitialContext();
             return (SearchFacade) c.lookup("URL_of_resource");
          }
```

Παρακάτω (Σχήμα 3.5) παρουσιάζεται το αρχιτεκτονικό διάγραμμα, με το οποίο κανείς μπορεί να δει πως συσχετίζονται τα διάφορα στοιχεία, μιας EJB εφαρμογής.



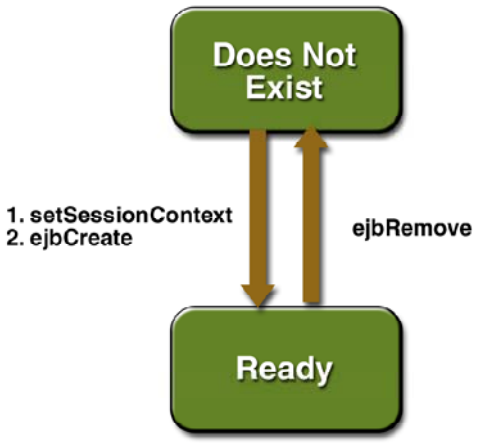
Σχήμα 3.5 Αρχιτεκτονικό διάγραμμα επιχειρησιακού (EJB) στρώματος

Στο δομικό διάγραμμα του πιο κάτω σχήματος (Σχήμα 3.6) παρουσιάζεται η χρήση των JNDI και RMI με τα συστατικά λογισμικού EJB.



Σχήμα 3.6 Χρήση των JNDI και RMI με συστατικά λογισμικού EJB

3.4.2.1.4 Ο κύκλος ζωής ένος Stateless Session Bean



Σχήμα 3.7 Ο κύκλος ζωής ενός stateless session bean

Επειδή το stateless session bean δεν μπορεί παθητικοποιηθεί (κατάσταση στη οποία παύεται η σχέση που είχε δημιουργηθεί με τον πελάτη, ώστε το bean είναι ανενεργό παροδικά), έτσι ο κύκλος

ζωής του έχει μόνο δύο καταστάσεις, είτε να έχει σύνδεση με τον πελάτη (κατάσταση Ready) και να περιμένει να δεχτεί κλήσεις ή να μην υπάρχει καθόλου σχέση (κατάσταση Does not exist)

3.4.2.2 Stateful session beans

Στην εισαγωγή μας για τα session bean περιγράψαμε τις διαφορές μεταξύ stateless και stateful bean οι όποιες βασίζονται στον τρόπο που αλληλεπιδρούν ο πελάτης και ο εξυπηρετητής. Στην περίπτωση των stateless session bean, αυτή η αλληλεπίδραση ξεκινά κα τελειώνει με την κλήση μιας απλής μεθόδου. Κάποιες φορές όμως οι πελάτες έχουν την ανάγκη, για πολλαπλές κλήσεις σε μια υπηρεσία και να έχουν τη δυνατότητα κάθε κλήση τους να μπορεί να έχει πρόσβαση ή να μπορεί να αποκτήσει τα αποτελέσματα προηγούμενων κλήσεων.

Τα stateful session bean έχουν σχεδιαστεί για να χειρίζονται αποδοτικά αυτήν την περίπτωση μέσω της παροχής αφιερωμένων υπηρεσιών στον πελάτη, ο οποίος ξεκινά όταν κάνει μια αναφορά στο bean και τελειώνει μόνο όταν αυτός επιλέξει να διακοπεί η «συνομιλία».

Ένα πολύ επεξηγηματικό παράδειγμα για την χρησιμότητα των stateful session bean αποτελεί η κάρτα αγοράς (shopping cart) μιας on-line εφαρμογής αγορών (e-commerce).

Ο πελάτης αποκτά μια αναφορά στην κάρτα αγοράς, και ξεκινά την «συνομιλία». Κατά την διάρκεια session του χρήστη, ο πελάτης προσθέτει ή αφαιρεί ποσά από την κάρτα αγορών η οποία διατηρεί συνεχώς μια προσαρμοσμένη επικοινωνία με τον πελάτη. Έπειτα όταν η περίοδος επικοινωνίας (session) ολοκληρωθεί, ο πελάτης ολοκληρώνει την αγορά, και έτσι ή κάρτα αγοράς μπορεί να αφαιρεθεί από το μηχάνημα του πελάτη .

Η παραπάνω διαδικασία δεν διαφέρει από αυτή της χρήσης ένος μη διαχειρήσιμου αντικειμένου Java (non-managed Java object) στον κώδικα της εφαρμογής. Δημιουργείται ένα στιγμιότυπο , και καλεί τις λειτουργίες του αντικειμένου , αποθηκεύοντας ταυτόχρονα την κατάσταση τους , και έπειτα τις απομακρύνει από το αντικείμενο όταν κρίνει ότι πλέον δεν χρειάζονται άλλο. Η μόνη διαφορά με τα stateful session bean είναι ότι ο εξυπηρετητής

διαχειρίζεται το ενεργό στιγμιότυπο του αντικειμένου και ο πελάτης αλληλεπιδρά με αυτό το στιγμιότυπο έμμεσα, μέσω της επιχειρησιακής διεπαφής του session bean.

Τα stateful session bean με αυτά που προσφέρουν συνδράμουν κατά ένα μεγάλο μέρος στην λειτουργικότητα των stateless session bean. Τα χαρακτηριστικά τα οποία ισχύουν για τα stateless session beans, όπως η εφαρμογή απομακρυσμένης διεπαφής

τ ισχύουν εξίσου και για τα stateful session bean.

Ομοίως με τα stateless session bean, τα stateful bean αποτελούνται και αυτά από μια κλάση bean και μια επιχειρησιακή διεπαφή.

3.4.2.2.1 Η κλάση bean

Μια κλάση stateful session bean είναι μια κοινή κλάση Java οι οποία περιέχει μια επισημείωση (annotation) τύπου stateful: @Stateful. Παράδειγμα μιας Stateful session bean κλάσης, για μια κάρτα αγοράς παρουσιάζεται η κλάση ShoppingCart session bean στην παρακάτω λίστα κώδικα (Λίστα 3.7).

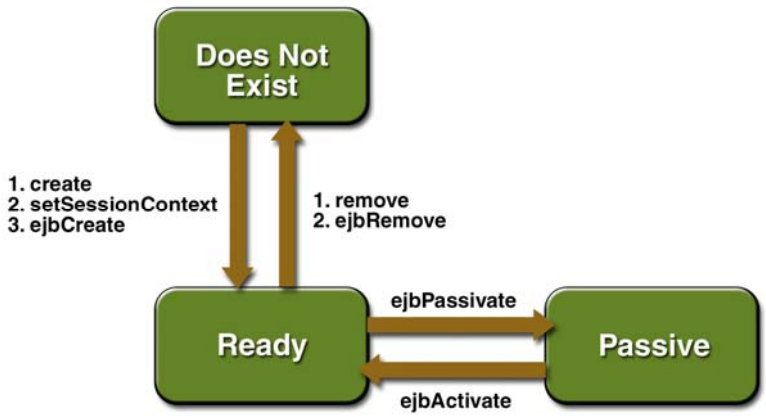
Λίστα 3.7 : ShoppingCartBean.java

```
import javax.ejb.Stateful;
@Stateful(name="ShoppingCart")
public class ShoppingCartBean implements ShoppingCart, ShoppingCartLocal {
public ShoppingCartBean() {
}
}
```

3.4.2.2.2 Η επιχειρησιακή διεπαφή

Οι επιχειρησιακές διεπαφές για τα stateful session bean είναι παρόμοιες με εκείνες των stateless session bean, και επισημαίνονται με τον ίδιο τρόπο, χρησιμοποιώντας επισημειώσεις : @Local και @Remote.

3.4.2.2.3 Ο κύκλος ζωής ένος Stateful Session Bean



Σχήμα 3.7 Ο κύκλος ζωής ένος stateful session bean

Εδώ ο κύκλος ζωής αποτελείται από μια κατάσταση παραπάνω (Passive), στην οποία το stateful session bean είναι ανενεργό (passivate) (ή όχι - passivate, ενεργό - activate), χωρίς όμως να έχει χάσει τελείως τη σχέση που είχε δημιουργήσει με τον πελάτη (κατάσταση Ready), αφού κατά την κλήση κάποιας επιχειρησιακής μεθόδου , ενεργοποιείται και περνάει αμέσως στην κατάσταση έτοιμο (Ready) όταν χρειαστεί να εξυπηρετήσει μια κλήση.

3.4.3 Entity beans

3.4.3.1 Τι είναι ένα entity bean ;

Ένα entity bean αναπαριστά ένα επιχειρησιακό (business) αντικείμενο που βρίσκεται σε ένα μηχανισμό μόνιμής αποθήκευσης δεδομένων. Παραδείγματα επιχειρησιακών αντικειμένων αποτελούν οι πελάτες, οι παραγγελίες, και τα προϊόντα. Απ'την πλευρά του εξυπηρετητή εφαρμογών, μηχανισμός μόνιμής αποθήκευσης δεδομένων αποτελεί μια σχεσιακή βάση δεδομένων.
Τυπικά , κάθε entity bean έχει μια υποδομή πίνακα στη σχεσιακή βάση δεδομένων και κάθε στιγμιότυπο του ανταποκρίνεται σε μια γραμμή αυτού του πίνακα.

3.4.3.2 Τι είναι αυτό που κάνει τα entity bean διαφορετικά από τα session beans;

Τα entity bean διαφέρουν από τα session bean σε πολλά σημεία. Η κύρια διαφορά τους έγκειται στο ότι τα entity beans αντιπροσωπεύουν μόνιμα δεδομένα . Επίσης επιτρέπουν διαμοιραζόμενη πρόσβαση, έχουν πρωτεύοντα κλειδιά, και μπορούν να λάβουν μέρος σε συσχετίσεις με άλλα entity bean.

3.4.3.3 Μονιμότητα δεδομένων (persistence)

Για το λόγο ότι η κατάσταση ενός entity bean αποθηκεύεται σε έναν σε έναν μηχανισμό αποθήκευσης, θεωρείται μόνιμή (persistent). Η έννοια «μόνιμη», σημαίνει ότι η κατάσταση του entity bean συνεχίζει να υπάρχει και μετά το χρόνο ζωής λειτουργίας της εφαρμογής ή του εξυπηρετητή εφαρμογών. Χαρακτηριστικό παράδειγμα αποτελεί ή περίπτωση των βάσεων δεδομένων, όπου τα δεδομένα που είναι αποθηκευμένα σε αυτές είναι καταχωρημένα σε μόνιμη κατάσταση. Αφού συνεχίζουν να υπάρχουν στην ίδια κατάσταση και μετά το κλείσιμο του εξυπηρετητή βάσεων δεδομένων ή της εφαρμογής που τα δίνει ως υπηρεσία.

3.4.3.4 Διαμοιραζόμενη πρόσβαση

Τα entity bean μπορούν να μοιράζονται σε πολλούς πελάτες. Επειδή οι πελάτες θα θέλουν να αλλάζουν τα ίδια δεδομένα, είναι σημαντικό τα entity bean να λειτουργούν με δοσοληψίες (transactions). Γενικά , ο υποδοχέας EJB παρέχει διαχείριση δοσοληψιών από μόνος του. Στην περίπτωση που ο προγραμματιστής θέλει να ορίσει τα δικά του όρια σε ένα bean , αυτό το δηλώνει απλώς στον περιγραφέα εγκατάστασης (deployment descriptor), χωρίς να χρειάζεται ο προγραμματιστής να γράψει κώδικα για της δοσοληψίες του bean.

3.4.3.5 Πρωτεύον κλειδί

Κάθε entity bean έχει ένα μοναδικό αριθμό ταυτότητας αντικειμένου (object identifier). Για παράδειγμα το entity bean ενός πελάτη, μπορεί να προσδιορίζεται από έναν αριθμό πελατών. Ένα μοναδικό αναγνωριστικό ταυτότητας ή ένα πρωτεύον κλειδί, βοηθάει τον πελάτη για την εύρεση του συγκεκριμένου entity bean.

3.4.3.5 Συσχετίσεις μεταξύ entity bean

Όπως ένας πίνακας μιας σχεσιακή βάσης δεδομένων, έτσι και ένα entity bean επιτρέπεται να συσχετίζεται με άλλα entity bean. Για παράδειγμα σε μια εφαρμογή διαχείρισης μαθημάτων μιας σχόλης το StudentBean και το CourseBean θα συσχετίζονται γιατί οι σπουδαστές εγγράφονται σε μαθήματα.
Μετά τη δημιουργία συσχέτισης μέσα σε ένα bean από τον προγραμματιστής, ο υποδοχέας EJB είναι αυτός που αναλαμβάνει την εφαρμογή, τον έλεγχο, και τη διατήρηση της συσχέτισης στη φυσική υποδομή της βάσης δεδομένων.

3.4.3.6 Διαχείριση καταχωρημένων δεδομένων από τον υποδοχέα

Διαχείριση καταχωρημένων δεδομένων από τον υποδοχέα σημαίνει ότι ο υποδοχέας EJB διαχειρίζεται τις προσβάσεις που απαιτούνται στη βάση δεδομένων, που γίνονται μέσω του entity bean. Ο κώδικάς του bean δεν περιέχει κλήσεις στη βάση τύπου SQL. Ο κώδικάς του bean δεν αναφέρεται σε ένα συγκεκριμένο αποθηκευτικό μηχανισμό (βάση δεδομένων). Συνακόλουθο αυτής της ελαστικότητας είναι, ότι εάν κανείς εγκαταστήσει το entity bean σε διαφορετικούς εξυπηρετητές J2EE οι οποίοι χρησιμοποιούν διαφορετικές βάσεις δεδομένων, δεν θα χρειαστεί να τροποποιηθεί ή να ξαναμεταγλωττιστεί ο κώδικας του bean.

3.4.3.7 Αφηρημένο σχήμα περιγραφής (abstract schema)

Μέρος του περιγραφέα εγκατάστασης (deployment descriptor) του entity bean, είναι το αφηρημένο σχήμα περιγραφής το οποίο ορίζει τα πεδία της καταχώρισης των δεδομένων και τις συσχετίσεις του bean. Ο όρος αφηρημένος (abstract) διαχωρίζει τη λογική απεικόνιση από την φυσική απεικόνιση της φυσικής υποδομής του αποθηκευτικού χώρου. Σε μια σχεσιακή βάση δεδομένων, για παράδειγμα, το φυσικό σχήμα είναι φτιαγμένο από δομές όπως πίνακες, γραμμές και στήλες.

3.4.3.8 Πότε γίνεται χρήση των Entity Bean

Η χρήση των entity bean γίνεται στις ακόλουθες περιπτώσεις :

- **Όταν το bean αναπαριστά ένα επιχειρησιακό entity και όχι μια διαδικασία**. Για παράδειγμα, Το CreditCardBean θα ήταν ένα entity bean, αλλά το CreditCardVerifierBean θα ήταν ένα session bean.

- **Όταν η κατάσταση του bean θέλουμε να είναι μόνιμα αποθηκευμένη.** Εάν, το στιγμιότυπό του bean τερματιστεί ή αν ο εξυπηρετητής εφαρμογών σταματήσει τη λειτουργία του, η κατάσταση του bean θα συνεχίζει να υπάρχει στην αποθήκη μόνιμων δεδομένων (βάση δεδομένων).

Παρακάτω (Λίστα 3.8) παρουσιάζεται παράδειγμα κλάσης, ενός απλού Entity Bean: Customer.java

Λίστα 3.8 : Ο κώδικας περιγράφει την κλάση Customer.java Java bean μετά το μετασχηματισμό της σε entity.

```
@Entity
public class Customer {
@Id
private long customerId;
private String name;
public long getCustomerId() { return customerId; }
public void setCustomerId(long customerId) { this.customerId = customerId; }
```

```
public String getName() { return name; }
public void setName(String name) { this.name = name; }
}
```

Τα ελάχιστα μεταδεδομένα που χρειάζονται να προστεθούν σε μια απλή κλάση Java για να γίνει κλάση entity είναι τα οι επισημειώσεις @Entity και @Id. Η επισημείωση @Entity χρειάζεται, κατά την εγκατάσταση του bean στον εξυπηρετητή, για να προσδιορίσει ότι η κλάση είναι κλάση entity. Η επισημείωση @Id τοποθετείται για να προσδιορίσει τη στήλη του πρωτεύοντος κλειδιού στον πίνακα, όπου οι τιμές τις θα πρέπει να είναι μοναδικές. Μετά την εγκατάσταση στον εξυπηρετητή του παραπάνω entity bean μέσω της κλάσης του δημιουργείται ένας πίνακας στη βάση δεδομένων με όνομα Customer και με ονόματα στηλών: customerId (πρωτεύον κλειδί- τύπου long) και name (τύπου String)

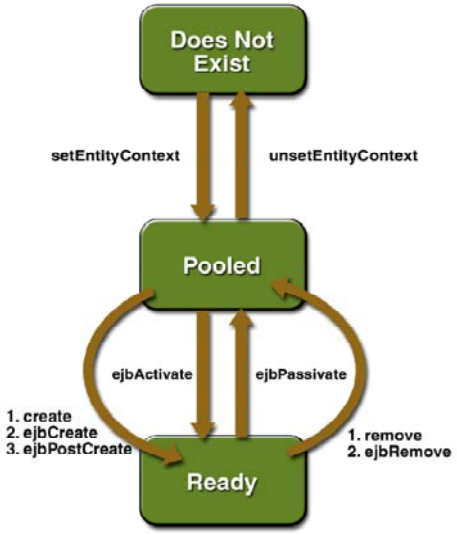
3.4.3.9 Τι χρειάζεται για να δουλέψει ένα enterprise bean;

Ο προγραμματιστής πρέπει καταρχάς να δημιουργήσει για το bean τις ακόλουθες διεπαφές interface και κλάσεις :

- **Την απομακρυσμένη ή τοπική διεπαφή (η και τις δύο).** Στις απομακρυσμένες ή τοπικές διεπαφές ορίζονται οι μέθοδοι που ένας πελάτης χρησιμοποιεί για να, δημιουργήσει (create), να βρει (find), να τροποποιήσει (edit) και να διαγράψει (destroy) στιγμιότυπα ενός enterprise bean καθώς και σε αυτά δηλώνονται και οι επιχειρησιακές μέθοδοι που εκτελεί το bean. Ένας πελάτης προσπελάζει αυτές τις μεθόδους μέσω της απομακρυσμένης ή της τοπικής διεπαφής.

- **Την session bean κλάση που υλοποιεί την επιχειρησιακή λογική για το bean.** Το session bean υλοποιεί τις επιχειρησιακές και τις βασικές (create, edit, destroy) μεθόδους , οι οποίες καλούνται μέσω αυτού από το entity bean.

3.4.3.10 Ο κύκλος ζωής ένος entity bean



Σχήμα 3.8 Ο κύκλος ζωής ένος entity bean

Το παραπάνω διάγραμμα καταστάσεων (Σχήμα 3.8) παρουσιάζει τις διαφορές καταστάσεις απ'τις οποίες περνάει ένα entity bean κατά τη διάρκεια ζωής του.

Αφού ο υποδοχέας EJB δημιουργήσει ένα στιγμιότυπο του bean, καλεί τη μέθοδο setEntityContext η οποία περνά το entity περιεχόμενο (θέση, διαφορές παραμέτρους του bean κτλ.) στο στιγμιότυπο του bean. Στη συνέχεια οδηγείται σε μια δεξαμενή των διαθέσιμων στιγμιοτύπων και περιμένει να εξυπηρετηθεί. Όσο είναι στην κατάσταση pooled , το στιγμιότυπο δεν συσχετίζεται με κάποιο συγκεκριμένο ταυτότητα EJB αντικειμένου. Όλα τα στιγμιότυπα στη δεξαμενή είναι πανομοιότυπα. Ο υποδοχέας EJB αποδίδει έναν αριθμό ταυτότητας σε κάθε στιγμιότυπο όταν τα μετακινεί στην κατάσταση ready. Υπάρχουν δύο ενδεχόμενα για να βρεθεί ένα στιγμιότυπο από την κατάσταση pooled στην ready. Στο πρώτο ενδεχόμενο, ο πελάτης καλεί μια create μέθοδο για να δημιουργήσει την απεικόνιση (γραμμή σε έναν πίνακα της βάσης) του συγκεκριμένου στιγμιότυπου στη βάση. Στο δεύτερο ενδεχόμενο, ο υποδοχέας EJB καλεί μια ejbActivate μέθοδο για να ενεργοποιήσει ένα bean που ήδη υπάρχει στη βάση. Όσο ένα entity bean είναι στην κατάσταση ready μπορεί, μόνο τότε, να καλέσει μια από τις business μεθόδους .

3.4.4 Message-driven beans

Μέχρι τώρα είδαμε συστατικά λογισμικού τα όποια επικοινωνούσαν με σύγχρονο τρόπο. Ο πελάτης καλεί μια μέθοδο μέσω της επιχειρησιακής διεπαφής, και ο εξυπηρετητής ολοκληρώνει την κλήση αυτής της μεθόδου πριν επιστρέψει τον έλεγχο στον πελάτη. Για την πλειονότητα των

υπηρεσιών αυτή είναι η πιο φυσική προσέγγιση. Υπάρχουν όμως παρόλα αυτά περιπτώσεις που αυτό δεν είναι απαραίτητο για τον πελάτη που περιμένει μια απόκριση από τον εξυπηρετητή. Θα ήταν πιο επιθυμητό ο πελάτης να κάνει μια αίτηση και στη συνέχεια να ασχοληθεί με κάτι άλλο-χωρίς να παραμένει δεσμευμένος , αφήνοντας έτσι τον εξυπηρετητή να διαχειριστή την αίτηση ασύγχρονα.

Το message-driven bean (MDB) είναι ένα συστατικό λογισμικού EJB που χρησιμοποιείται για ασύγχρονή ανταλλαγή μηνυμάτων. Οι πελάτες κάνουν αιτήσεις στο MDB χρησιμοποιώντας το Java Message Service (JMS) σύστημα ανταλλαγή μηνυμάτων. Αυτές οι αιτήσεις μπαίνουν σε μια ουρά εξυπηρέτησης και διανέμονται στο MDB μέσω του εξυπηρετητή. Ο εξυπηρετητής καλεί την επιχειρησιακή διεπαφή του MDB όταν αυτός λάβει ένα μήνυμα αποστολής από τον πελάτη. Η διαφορά με τα session bean έγκειται στον ορισμό της σύμβασης (component contract) του, όπου ο «τρόπος συμπεριφοράς» ορίζεται από την δομή των μηνυμάτων που έχει σχεδιαστεί να δέχεται και όχι από το πώς έχει οριστεί η επιχειρησιακή διεπαφή του.

3.4.5 Ο διαχειριστής των entity bean (entity manager)

Τα entity δεν καταχωρούνται στη βάση δεδομένων από μόνα τους, όταν αυτά δημιουργούνται. Ούτε σβήνονται από μόνα τους από τη βάση, όταν είναι να διαγραφούν. Είναι το κομμάτι της επιχειρησιακής λογικής (business logic) της εφαρμογής αυτό που χειρίζεται τα entities για τη διαχείριση των κύκλων ζωής αυτών. Το Java Persistence API παρέχει μια διεπαφή διαχειριστή οντοτήτων για αυτό το σκοπό , ώστε να επιτρέπει στις εφαρμογές να διαχειρίζονται και να ψάχνουν για τα entity σε μια σχεσιακή βάση δεδομένων. Με μια πρώτη ματιά αυτό φαίνεται ως περιορισμός του Java Persistence API. Αν το περιβάλλον εκτέλεσης που είναι υπεύθυνο για την καταχώρηση, γνωρίζει ποια αντικείμενα είναι καταχωρημένα, γιατί η εφαρμογή θα πρέπει να εμπλέκεται σε αυτή τη διαδικασία; Παρόλα αυτά δεν χρειάζεται κανείς να προβληματιστεί για αυτήν τη σχεδιαστική επιλογή είναι διπλά ελεγμένη και μελετημένη και περισσότερο προσοδοφόρα για την εφαρμογή από μια πιο διαφανή λύση αποθήκευσης. Οι εφαρμογές που παρέχουν μόνιμη αποθήκευση είναι ουσιαστικά μια συνεργασία μεταξύ του παροχέα μόνιμης καταχώρησης δεδομένων και της εφαρμογής. Το Java Persistence API φέρει ένα επίπεδο ελέγχου και ελαστικότητας που δεν θα μπορούσε αλλιώς να επιτευχθεί, χωρίς την ενεργή συνδρομή απ'την πλευρά της εφαρμογής.

3.4.5.1 To persistence unit

Το configuration (η αναφορά των ρυθμίσεων) το οποίο περιγράφει το persistence unit ορίζεται σε ένα αρχείο XML το οποίο ονομάζεται persistence.xml. Κάθε persistence unit έχει ένα όνομα, ώστε όταν μια εφαρμογή θέλει να αναφερθεί σε ένα συγκεκριμένο configuration για ένα entity αρκεί μόνο να αναφέρει το όνομα του αντίστοιχου persistence unit το οποίο ορίζει αυτό το configuration. Ένα απλό persistence.xml επιτρέπεται να έχει ένα ή περισσότερα configuration για το persistence unit, αλλά κάθε persistence unit να είναι ξεχωριστώ και διακριτό από τα άλλα, ορίζοντας έτσι έναν λογικό διαχωρισμό σαν βρισκόντουσαν σε διαφορετικά persistence.xml αρχεία.

Ένα παράδειγμα ένος persistence.xml αρχείου παρουσιάζεται στην παρακάτω λίστα κώδικα (Λίστα 3.8), όπου όπως παρουσιάζεται ορίζεται το όνομα του persistence unit που περιλαμβάνει, και εντός του persistence unit ορίζεται ο τύπος του διαχειριστή οντοτήτων
, ή βάση δεδομένων καθώς και οι κλάσεις των entity beans στις οποίες αναφέρεται το συγκεκριμένο configuration.

Λίστα 3.8 : persistence.xml

```
<?xml version="1.0" encoding="windows-1252" ?>
<persistence xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd"
xmlns="http://java.sun.com/xml/ns/persistence">

<persistence-unit name="Chapter03-Unit">
<provider>oracle.toplink.essentials.ejb.cmp3.EntityManagerFactoryProvider
</provider>
<jta-data-source>jdbc/wineappDS</jta-data-source>
<class>com.apress.ejb3.ch03.Address</class>
<class>com.apress.ejb3.ch03.Customer</class>
<class>com.apress.ejb3.ch03.CustomerOrder</class>
</persistence-unit>
</persistence>
```

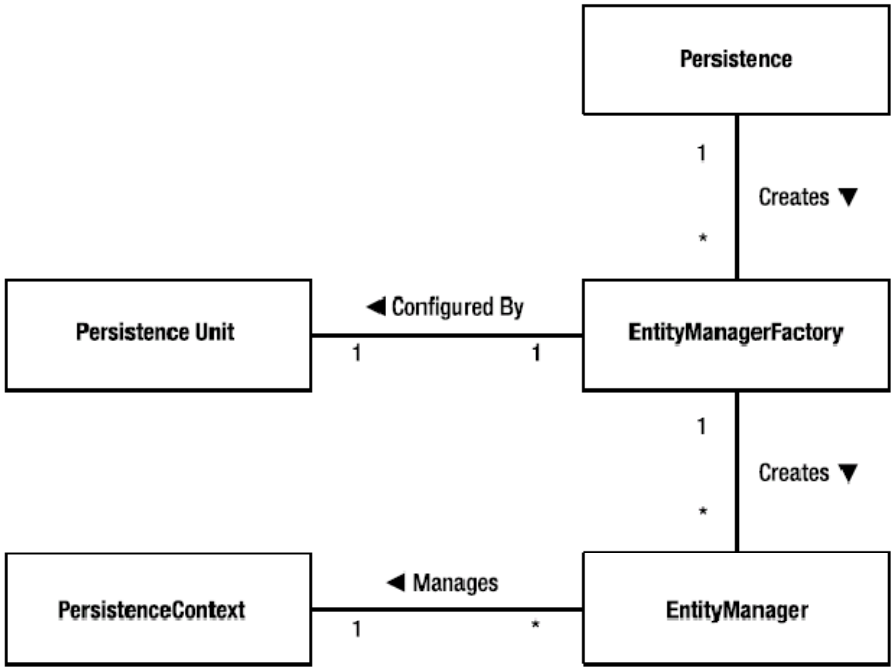
3.4.5.2 Persistence contexts

Όπως είδαμε παραπάνω ως persistence unit ονομάζεται το configuration των entity κλάσεων. Ένα persistence context είναι μια διαχειριζόμενη ομάδα από entity στιγμιότυπα, τα όποία βρίσκονται

εντός του διαχειριστή οντοτήτων και ανά πάσα στιγμή μπορούν να χρησιμοποιηθούν . Κάθε persistence context είναι συσχετιζόμενο με ένα persistence unit (ορίζεται στο percistence.xml) το οποίο περιορίζει τις κλάσεις των διαχειριζόμενων στιγμιότυπων, σε μια ομάδα κλάσεων όπως αυτή ορίζεται στο persistence unit. Λέγοντας ότι ένα entity είναι διαχειριζόμενο σημαίνει ότι αυτό περιέχεται εντός του persistence context και είναι επιτρεπτό να ενεργοποιηθεί από τον διαχειριστή οντοτήτων. Για αυτό το λόγο λέμε ότι ένας διαχειριστής οντοτήτων διαχειρίζεται ένα persistence context.

Η κατανόηση του persistence context αποτελεί το κλειδί για την κατανόηση του entity manager. Εάν ένα entity περιλαμβάνεται ή δεν περιλαμβάνεται στο persistence context αυτό θα ορίσει και τις οποιεσδήποτε persistent λειτουργίες πάνω σε αυτό. Εάν το persistence context παίρνει μέρος σε μια δοσοληψία (transaction), τότε ή κατάσταση των διαχειριζόμενων entity που βρίσκονται στη μνήμη θα συγχρονιστεί με τη βάση δεδομένων. Αν και παίζει σημαντικό ρόλο, το persistence context πότε δεν είναι ορατό στην εφαρμογή.

Πάντα προσπελάζεται εμμέσως δια του διαχειριστή οντοτήτων και να παρουσιάζεται όταν το χρειαστούμε.



Σχήμα 3.9 Οι συσχετίσεις μεταξύ των Java Persistence API εννοιών

Το παραπάνω διάγραμμα (Σχήμα 3.9) δείχνει ότι για κάθε persistence unit υπάρχει ένας EntityManagerFactory και ότι πολλοί διαχειριστές οντοτήτων μπορούν να δημιουργηθούν από έναν απλό EntityManagerFactory. Αυτό το οποίο είναι αξιοσημείωτο είναι ότι πολλοί (ίσως

διαφορετικού σκοπού) διαχειριστές οντοτήτων μπορούν να διαχειρίζονται το ίδιο persistence context.

3.4.5.3 Βασικές λειτουργίες του διαχειριστή οντοτήτων

Εδώ θα παρουσιάσουμε τις βασικές λειτουργίες του διαχειριστή οντοτήτων, μέσα από τις βασικές καταστάσεις , απ'τις οποίες μπορεί να περάσει το στιγμιότυπο ένος entity bean , κατά την διάρκεια του κύκλου ζωής του ως αντικείμενου Java που βρίσκεται στη μνήμη.

3.4.5.3.1 Δημιουργία νέου στιγμιότυπου ενός entity

Ένας πελάτης δημιουργεί ένα νέο στιγμιότυπο με τη χρήση ενός Java κατασκευαστή της κλάσης του entity. Ένας πελάτης μπορεί να βρίσκεται εκτός η εντός ενός υποδοχέα EJB. Απ'την πλευρά την κατασκευαστική , αυτό αποτελεί μια καινούρια κατάσταση όπου δεν έχει αποδοθεί σε αυτή ακόμη κάποια persistent ταυτότητα, επειδή δεν έχει ακόμη συσχετιστεί με το persistence context του διαχειριστή οντοτήτων. Ο πελάτης είναι ελεύθερος να καλεί οποιαδήποτε απ'τις μεθόδους του και να αποδίδει τιμές στα δεδομένα, και όλα όσα ενημερώνουν το entity να παραμένουν τοπικά στην entity κλάση.

3.4.5.3.2 Αποθήκευση στιγμιοτύπου ενός entity

Η διαδικασία της αποθήκευσης ενός entity έχει να κάνει με διαδικασία λήψης ένος προσωρινού στιγμιότυπου ένος entity ή κάποιου που δεν είχε ακόμα κάποια μόνιμη απεικόνιση στη βάση δεδομένων, και να αποθηκεύσει την κατάσταση του έτσι ώστε να μπορεί να ανακτηθεί κάποια άλλη στιγμή. Αυτή είναι και η έννοια του persistence - δημιουργία μιας κατάστασης η οποία ζει περισσότερο από την διαδικασία που την δημιούργησε.
Στο παρακάτω παράδειγμα κώδικα (Λίστα 3.9) φαίνεται ή χρήση του διαχειριστή οντοτήτων
, στη διαδικασία αποθήκευσης ενός στιγμιότυπου της κλάσης entity bean Employee.

Λίστα 3.9 : Κώδικας που αποθηκεύει το στιγμιότυπο της κλάσης entity bean Employee

```
Employee emp = new Employee(158);
em.persist(emp);
```

Για να μετασχηματίσει την κλάση entity σε ένα αποθηκευμένο αντικείμενο, ο πελάτης αποκτά ένα στιγμιότυπο διαχειριστή οντοτήτων και καλεί τη μέθοδο EntityManager.persist().
Στον παρακάτω κώδικα (Λίστα 3.10) ένος session bean παρουσιάζεται ή απόκτηση του διαχειριστή οντοτήτων μέσω injection τρόπου (επισημείωση @PersistenceContext), και στη συνέχεια κάνει persist το entity στιγμιότυπο περνώντας το ως παράμετρο στην persistEntity() μέθοδο.

Λίστα 3.10 : Κώδικας που αποθηκεύει το στιγμιότυπο της κλάσης entity bean Employee

```
@Stateless
public class MySessionEJB implements MySession {
@PersistenceContext("Chapter03-Unit")
private EntityManager em;
public Object persistEntity(Object entity) {
return em.persist(entity);
}
}
```

Όταν ένα entity αποθηκεύεται στη βάση δεδομένων, αυτό προτίθεται στο persistence context ως διαχειριζόμενο στιγμιότυπο.

3.4.5.3.3 Εύρεση ενός entity

Όταν ένα entity είναι αποθηκευμένο σε μια βάση δεδομένων , το επόμενο πράγμα που πρόκειται να του συμβεί είναι η αναζήτηση του.
Ένα παράδειγμα εύρεσης ενός entity αντικειμένου Employee , με πρωτεύον κλειδί τον αριθμό 158 φαίνεται στη παρακάτω λίστα κώδικα (Λίστα 3.11).

Λίστα 3.11 : Κώδικας ο οποίος επιστρέφει ένα entity που έχει πρωτεύον κλειδί τον αριθμό 158

```
Employee emp = em.find(Employee.class, 158);
```

3.4.5.3.4 Διαγραφή ενός entity

Για να αφαιρεθεί ένα αντικείμενο, το όποιο έχει απεικόνιση στη βάση δεδομένων, θα πρέπει να είναι διαχειρίσιμο, δηλαδή με άλλα λόγια να υπάρχει στο persistence context. Ένα entity γίνεται ένα αφαιρούμενο στιγμιότυπο, όταν η remove() μέθοδος κληθεί. Η γραμμή (ή οι γραμμές , εάν αυτό το entity αντιστοιχείται σε πολλούς πίνακες) αναπαριστά την αποθηκευμένη του κατάσταση που θα αφαιρεθεί όταν ή σχετική δοσοληψία (transaction) διαγραφής του ,συμβεί. Ένα παράδειγμα διαγραφής ενός Employee φαίνεται στη παρακάτω λίστα κώδικα (Λίστα 3.12).

Λίστα 3.12 : Κώδικας ο οποίος αφού βρει ένα entity που έχει πρωτεύον κλειδί τον αριθμό 158 στη συνέχεια το διαγράφει από τη βάση δεδομένων στην οποία είναι αποθηκευμένο

```
Employee emp = em.find(Employee.class, 158);
em.remove(emp);
```

3.4.5.2.5 Ενημέρωση – επεξεργασία του αποθηκευμένου περιεχομένου ένος entity

Ένα entity επιτρέπεται να ενημερώνεται με αρκετούς και διαφόρους τρόπους, αλλά εδώ θα παρουσιάσουμε τον πιο συνηθισμένο και στην πιο απλή περίπτωση. Είναι ή περίπτωση που έχουμε ένα διαχειρίσιμο entity και θέλουμε να τροποποιήσουμε το persistence context του. Εάν δεν έχουμε μια αναφορά αυτού θα πρέπει αρχικά να λάβουμε χρησιμοποιώντας την μέθοδο find() και μετά να εκτελέσουμε τις λειτουργίες τροποποίησης στο διαχειριζόμενο entity. Το παράδειγμα της παρακάτω λίστας (Λίστα 3.13) κώδικα προσθέτει 1,000 € στο μισθό (salary) ενός υπαλλήλου (Employee) με πρωτεύον κλειδί 158.

Λίστα 3.13 : Κώδικας που τροποποιεί το περιεχόμενο ενός entity

```
Employee emp = em.find(Employee.class, 158);
emp.setSalary(emp.getSalary() + 1000);
```

3.4.5.2.6 Προσωρινά αποσπασμένο και επανενώσιμο στιγμιότυπο ενός Entity

Αποσπασμένο (detached) είναι ένα entity το οποίο δεν είναι πλέον συσχετισμένο με το persistence context. Παρόλα αυτά, το entity παραμένει σε διαχειρίσιμη κατάσταση για το persistence context , έως ότου αυτό αφαιρεθεί τελειωτικά από τη βάση δεδομένων.

Όταν ένα entity στιγμιότυπο επιστρέφει (επανενώνεται- merging) στο persistence context ύστερα από μια EntityManager.merge() κλήση που το διαχειρίστηκε , οι αλλαγές δεν γίνονται αμέσως στην αποθήκη μόνιμης πληροφορίας (persistent storage) π.χ. στη βάση δεδομένων. Απλώς ενημερώνεται το στιγμιότυπο του entity, χωρίς να αποθηκευτεί στη βάση δεδομένων. Στο παρακάτω παράδειγμα (Λίστα 3.14) ο πελάτης τροποποιεί (θέτει μια τιμή στη στήλη Name) του entity , χωρίς αυτή η αλλαγή να ενημερώνει το περιεχόμενο της βάσης δεδομένων.

Λίστα 3.14 : Κώδικας που τροποποιεί το περιεχόμενο μιας στήλης με όνομα Name

```
entity = mySession.persistEntity(entity);
entity.setName( "foo" );
```

3.4.5.3 Λειτουργίες καταρράκτη (cascading operations)

Εξ ορισμού, κάθε λειτουργία του διαχειριστή οντοτήτων εφαρμόζεται μόνο στο entity το οποίο παρέχεται ως παράμετρος στη μέθοδο που εκτελεί τη συγκεκριμένη λειτουργία . Δηλαδή η λειτουργία δεν περνάει (cascade) σε άλλα entity τα οποία έχουν συσχέτιση με το entity στο οποίο εφαρμόζεται η λειτουργία αυτή. Για κάποιες λειτουργίες όπως η remove(), ή παραπάνω κατάσταση έχει να κάνει με την επιθυμητή συμπεριφορά. Δεν θα θέλαμε ο διαχειριστής οντοτήτων να κάνει λανθασμένες θεωρήσεις σχετικά με το πια entity στιγμιότυπα θα πρέπει να αφαιρεθούν ως παράπλευρή επίδραση από κάποια άλλη λειτουργία. Κάτι τέτοιο δεν έχει να κάνει με την πραγματικότητα λειτουργιών όπως της persist(). Αν παραδείγματος χάρη είχαμε , αλλαγές σε δυο νεοσύστατα entity τα οποία είχαν μια αμφίδρομή συσχέτιση μεταξύ τους , είναι αναγκαίο να αποθηκευτούν και τα δύο ταυτόχρονα.

Για αυτό το λόγο οι λειτουργίες του διαχειριστή οντοτήτων προσδιορίζονται χρησιμοποιώντας τον τύπο CascadeType. Οι σταθερές PERSIST, REFRESH, REMOVE, και MERGE αποδίδουν την αντίστοιχη λειτουργία με το ίδιο όνομα, στον διαχειριστή οντοτήτων

και ή σταθερά ALL δηλώνει στον διαχειριστή οντοτήτων ότι και οι τέσσερεις, παραπάνω λειτουργιές θα πρέπει να περνάνε (cascaded), δηλαδή να λειτουργούν παραπλεύρως στα άλλα beans με τα οποία υπάρχει συσχέτιση.

Ένα χρηστικό παράδειγμα χρήσης του της λειτουργίας καταρράκτη, είναι κατά την διαδικασία διαγραφή ενός αντικειμένου, έτσι έχοντας θέσει τον διαχειριστή οντοτήτων αυτόματα να κάνει cascade remove() αυτό φαινομενικά θα μας διευκόλυνε αρκετά στο να εξαλείψουμε την ανάγκη

διαγραφής πολλαπλών στιγμιότυπων χειροκίνητα ή με κώδικα. Παρόλο που θα μπορούσαμε να εφαρμόσουμε λειτουργίες καταρράκτη για ένα μεγάλο πλήθος περιπτώσεων, η εφαρμογή αυτής της πολιτικής πρέπει να γίνεται μόνο για συγκεκριμένες περιπτώσεις . Έτσι υπάρχουν μόνο δύο περιπτώσεις που εφαρμόζουμε πολιτική καταρράκτη στη remove() λειτουργία , στις συσχετίσεις τύπου: ένα- προς – ένα (one-to-one) και ένα – προς- πολλά (one-to-many) όπου δηλαδή έχουμε καθαρά μια συσχέτιση γονέα -παιδιού. Θα ήταν τυφλή ενέργεια να εφαρμόζαμε την λειτουργία καταρράκτη, αδιακρίτως, σε όλες τις ένα- προς – ένα (one-to-one) και ένα – προς- πολλά (one-to-many) συσχετίσεις, για το λόγο ότι τα entities τα οποία θα επηρέαζε άμεσα, θα μπορούσαν να συμμετέχουν και σε άλλες συσχετίσεις ή απλώς να υφίστανται ως αυτόνομα entity.

3.5 Υπηρεσίες και διασύνδεση των EJB

3.5.1 Ασφάλεια στο EJB στρώμα

Η EJB αρχιτεκτονική και η πλατφόρμα J2EE παρέχουν μια ισχυρή επιχειρησιακή υποδομή ασφάλειας ή οποία επιτρέπει στους διαχειριστές να ελέγχουν ποίος επιτρέπεται να έχει πρόσβαση σε εφαρμογές και δεδομένα. Αυτό το επίπεδο ασφάλειας πετυχαίνεται μέσω:

- **Προστασίας μηνυμάτων**, για την επιβεβαίωση της ορθότητας και της εμπιστευτικότητας των δεδομένων καθώς αυτά μεταδίδονται μέσω του Internet.

- **Πιστοποίησης (authentication)**, για επαλήθευση της ταυτότητας των πελατών των υπηρεσιών ιστού.

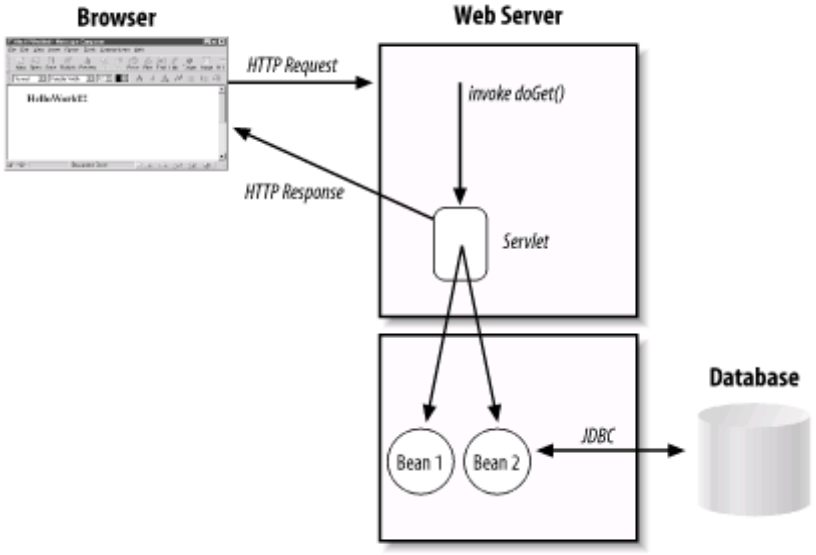
- **Εξουσιοδότησης (authorization)**, για περιορισμένη πρόσβαση, σε λειτουργίες, μόνο από επιτρεπόμενες ομάδες.

3.5.2 Ιστού (web) και επιχειρησιακά (ejb) συστατικά λογισμικού

Μαζί, servlet και JSP παρέχουν μια ισχυρή πλατφόρμα για δημιουργία δυναμικών σελίδων ιστού. Τα servlet και τα JSP αποτελούν συστατικά λογισμικού ιστού, που μπορούν να έχουν πρόσβαση σε πόρους όπως JDBC και enterprise bean. Επειδή τα συστατικά λογισμικού ιστού

μπορούν να προσπελάζουν βάσεις δεδομένων κάνοντας χρήση JDBC, επιτρέπουν σε μια επιχείρηση να διαθέτει τα επιχειρησιακά συστήματα στον Ιστό μέσω μιας διεπαφής HTML. Οι διεπαφές HTML έχουν αρκετά πλεονεκτήματα από τις περισσότερες συμβατικές διεπαφές πελάτη. Οι πιο σημαντικές έχουν να κάνουν με κατανεμημένα περιβάλλοντα και firewalls (τείχη προστασίας). Οι συμβατικοί πελάτες χρειάζεται να είναι εγκατεστημένοι και κατανεμημένοι σε μηχανές πελατών : και χρειάζονται επιπρόσθετη δουλεία εγκατάστασης και συντήρησης. Τα applets, τα οποία κατεβαίνουν στην μηχανή του πελάτη δυναμικά και μπορούν έτσι να περιορίσουν τον πονοκέφαλο της εγκατάστασης, έχουν τους δικούς περιορισμούς - όπως περιορισμούς που έχουν να κάνουν με την ίδια τη φύση τους, στοίβαγμα (όλα σε ένα), και με μεγάλης διάρκειας κατεβάσματα (downloads). Σε αντίθεση, η HTML είναι τρομερά «ελαφριά» , δεν χρειάζεται προετοιμασία για την εγκατάσταση της, και δεν υποφέρει από περιορισμούς ασφάλειας. Επιπροσθέτως, οι διεπαφές HTML μπορούν να τροποποιηθούν και να αναβαθμιστούν χωρίς να χρειάζεται να ενημερωθούν οι πελάτες.

Τα Firewalls παρουσιάζουν ένα αξιοσημείωτο πρόβλημα στο ηλεκτρονικό εμπόριο (e-commerce). Το πρωτόκολλο HTTP με όποιες σελίδες ιστού κληθεί και διανεμηθεί, μπορεί να περάσει μέσα από τα πιο πολλά firewalls χωρίς κάποιο πρόβλημα, όμως αλλά πρωτόκολλά όπως IIOP ή JRMP δεν μπορούν . Αυτός ο περιορισμός είναι πολύ σημαντικός. Σημαίνει ότι ένας πελάτης συνήθως δεν μπορεί να προσπελάζει έναν εξυπηρετητή χρησιμοποιώντας IIOP ή JRMP χωρίς τροποποιήσεις στο firewall. Και συνήθως, το firewall δεν είναι κάτω από τον έλεγχο των ομάδων (groups) που χρειάζονται την εφαρμογή να «τρέχει». Το HTTP δεν υποφέρει από αυτόν τον περιορισμό, έως τώρα όλα τα firewalls επιτρέπουν στο HTTP να περνά ανεμπόδιστα.
Τα προβλήματα με την κατανομή (distribution) και τα firewalls έχουν οδηγήσει τους πιο πολλούς της βιομηχανίας EJB να υιοθετήσουν μια αρχιτεκτονική βασισμένη στα συστατικά λογισμικού ιστού (servlet/JSP) και Enterprise Java Bean. Τα συστατικά λογισμικού ιστού παρέχουν τον μηχανισμό παρουσίασης για τη δημιουργία σελίδων ιστού. Τα EJB παρέχουν ένα ενδιάμεσο στρώμα για το το κομμάτι της επιχειρησιακής λογικής. Τα συστατικά λογισμικού ιστού προσπελάζουν τα enterprise beans χρησιμοποιώντας τα ίδια API όπως οι πελάτες των εφαρμογών. Κάθε τεχνολογία κάνει αυτό που νομίζει ότι είναι καλύτερο: servlet και JSP είναι έξοχα συστατικά λογισμικού για δυναμική HTML, ενώ τα EJB είναι μια έξοχή πλατφόρμα για ανάπτυξη επιχειρησιακής λογικής. Το σχήμα (Σχήμα 3.10) παρουσιάζει πως ή αρχιτεκτονική αυτή λειτουργεί.



Σχήμα 3.10 Χρήση μαζί servlet/JSP και EJB

Αυτή η αρχιτεκτονική συστατικών λογισμικού ιστού και EJB είναι τόσο ευρέως αποδεκτή όπου κανείς θα τολμούσε να κάνει το εξής ερώτημα: "Δεν θα έπρεπε αυτή η αρχιτεκτονική να βρίσκεται σε μια ενιαία πλατφόρμα ;" Η προδιαγραφή J2EE απαντά σε αυτήν την ερώτηση. Το J2EE ορίζει μια απλή πλατφόρμα εξυπηρετητή εφαρμογών η οποία εστιάζει στην αλληλεπίδραση μεταξύ servlet, JSP, and EJB. Το J2EE είναι σημαντικό γιατί παρέχει μια προδιαγραφή για τη διασύνδεση των συστατικών λογισμικού ιστού με τα enterprise beans, δημιουργώντας περισσότερες λύσεις φορητότητας μεταξύ των κατασκευαστών εφαρμογών, από ότι θα παρείχαν και οι δύο προδιαγραφές συστατικών λογισμικού μαζί.

3.6 Πλεονεκτήματα των enterprise Java bean

Η προδιαγραφή enterprise Java bean (EJB) βρίσκεται στην αιχμή της αρχιτεκτονικής για ανάπτυξη, εγκατάστασή, διαχείριση αξιόπιστων επιχειρησιακών εφαρμογών σε περιβάλλοντα παραγωγής. Εδώ θα παρουσιαστούν τα πλεονεκτήματα της χρήσης της αρχιτεκτονικής EJB για επιχειρησιακές εφαρμογές.

3.6.1 Πλεονεκτήματα από την πλευρά των προγραμματιστών

- **Απλότητα.** Είναι πιο εύκολο να αναπτύξεις μια επιχειρησιακή εφαρμογή με την EJB αρχιτεκτονική πάρα χωρίς αυτή. Επειδή ή EJB αρχιτεκτονική βοηθάει τον προγραμματιστή

εφαρμογών να έχει πρόσβαση και χρήση των επιχειρησιακών υπηρεσιών με τον ελάχιστο όγκο εργασίας και στον ελάχιστο χρόνο. Το γράψιμο ενός enterprise bean είναι σχεδόν τόσο απλό σαν έγραφε κανείς μια κλάση Java. Ο προγραμματιστής δεν χρειάζεται να μεριμνά για ζητήματα επιπέδου συστήματος , όπως ασφάλειας, δοσοληψιών, πολυνημάτωσης, πρωτοκόλλων ασφάλειας, κατανεμημένου προγραμματισμού και πολιτικής άντλησης πόρων. Αυτό έχει ως αποτέλεσμα , ο προγραμματιστής να συγκεντρώνει την προσοχή του μόνο στο κομμάτι της επιχειρησιακής λογικής της εφαρμογής.

- **Φορητότητα εφαρμογών**. Μια EJB εφαρμογή μπορεί να εγκαθίστανται σε έναν συμβατό εξυπηρετητή J2EE. Αυτό σημαίνει ότι ο προγραμματιστής της εφαρμογής μπορεί να πουλά την εφαρμογή σε όποιον πελάτη χρησιμοποιεί έναν συμβατό εξυπηρετητή J2EE. Αυτό επίσης σημαίνει ότι οι επιχειρήσεις δεν δεσμεύονται με ένα κατασκευαστή εξυπηρετητή εφαρμογών. Αντιθέτως, μπορούν να επιλέξουν τον καλύτερο για την περίπτωση τους – κάποιον δηλαδή που ικανοποιεί τις δικές τους απαιτήσεις.

- **Επαναχρησιμοποίηση συστατικών λογισμικού.** Μια EJB εφαρμογή αποτελείται από συστατικά λογισμικού enterprise bean. Κάθε enterprise bean αποτελεί ένα επαναχρησιμοποιήσιμο δομικό στοιχείο (building block). Υπάρχουν τρείς βασικοί τρόποι να επαναχρησιμοποιήσει κανείς ένα enterprise bean:

    - Ένα enterprise bean που δεν έχει ακόμη εγκατασταθεί μπορεί να είναι επαναχρησιμοποιήσιμο κατά το χρόνο ανάπτυξης και να συμπεριλαμβάνεται σε διάφορες εφαρμογές. Το bean μπορεί να προσαρμόζεται για κάθε εφαρμογή χωρίς να χρειαστεί να αλλαχτεί ή ακόμα και να υπάρξει πρόσβαση στον πηγαίο κώδικα του.

    - Άλλες εφαρμογές μπορούν να επαναχρησιμοποιήσουν ένα enterprise bean το οποίο έχει εγκατασταθεί στο επιχειρησιακό περιβάλλον του πελάτη, μέσω κλήσεων στις διεπαφές του πελάτη. Πολλές εφαρμογές μπορούν να κάνουν κλήση σε ένα εγκατεστημένο bean.

- Επιπροσθέτως, το κομμάτι της επιχειρησιακής λογικής των enterprise bean μπορεί να επαναχρησιμοποιηθεί δημιουργώντας μια υποκλάση Java της enterprise bean κλάσης.

- **Ικανότητα κτισίματος πολύπλοκών εφαρμογών.** Η αρχιτεκτονική EJB απλοποιεί την πολυπλοκότητα των επιχειρησιακών εφαρμογών. Οι αρχιτεκτονικές αυτές δομούνται από μια ομάδα προγραμματιστών και διαμορφώνονται συνεχώς. Η αρχιτεκτονική βασισμένη σε συστατικά λογισμικού EJB είναι κατάλληλη για την ανάπτυξή και συντήρηση σύνθετων επιχειρησιακών εφαρμογών. Με ένα ξεκάθαρο ορισμό ρολών και καλά ορισμένες διεπαφές, η αρχιτεκτονική EJB υποστηρίζει και προωθεί κατά πολύ την ανάπτυξη από ομάδα, περιορίζοντας ταυτόχρονα τις απαιτήσεις από τους μεμονωμένους προγραμματιστές.

- **Διαχωρισμός της επιχειρησιακής λογικής από τον μηχανισμό παρουσίασης**. Ένα επιχειρησιακό enterprise bean τυπικά ενθυλακώνει μια επιχειρησιακή διαδικασία ή ενός επιχειρησιακού entity (αντικείμενο το οποίο αναπαριστά τα επιχειρησιακά business δεδομένα), κάνοντας τα ανεξάρτητα από τον μηχανισμό παρουσίασης. Ο προγραμματιστής του επιχειρησιακού κομματιού δεν χρειάζεται να ανησυχεί για το πώς θα παρουσιάζεται ή έξοδος της εφαρμογής. Ο σχεδιαστής σελίδων ιστού χρειάζεται να επικεντρώνεται μόνο στα δεδομένα εξόδου που θα περάσει στην ιστοσελίδα. Επιπροσθέτως , αυτός ο διαχωρισμός κάνει δυνατή την ανάπτυξη πολλαπλών πολιτικών παρουσίασης για την ίδια επιχειρησιακή διαδικασία ή της αλλαγής της πολιτικής παρουσίασης μιας επιχειρησιακής διαδικασίας χωρίς να χρειαστεί να τροποποιηθεί ο κώδικας που υλοποιεί την επιχειρησιακή διαδικασία.

- **Εύκολή ανάπτυξη των υπηρεσιών ιστού (web services)**. Οι υπηρεσίες ιστού, χαρακτηριστικό της αρχιτεκτονικής EJB παρέχουν ένα εύκολο τρόπο στους προγραμματιστές Java για ανάπτυξη και πρόσβαση στις υπηρεσίες ιστού. Οι προγραμματιστές σε Java δεν χρειάζονται να σκοτίζονται με την πολυπλοκότητα των λεπτομερειών του μορφοτύπου περιγραφής των υπηρεσιών ιστού και των πρωτοκόλλων σύνδεσης που βασίζονται σε γλώσσα XML αλλά αντί για αυτά μπορούν να προγραμματίζουν στο οικείο επίπεδο των συστατικών λογισμικού enterprise bean, των διεπαφών Java και των τύπων δεδομένων. Τα εργαλεία που παρέχονται από τον υποδοχέα διαχειρίζονται την αντιστοίχηση με την προδιαγραφή των υπηρεσιών ιστού.

- **Εγκατάσταση σε πολλά λειτουργικά περιβάλλοντα**. Ο στόχος ένος κατασκευαστή εφαρμογών λογισμικού είναι να πουλά μια εφαρμογή σε όσο το δυνατόν περισσότερους πελάτες. Επειδή κάθε πελάτης έχει το δικό του μοναδικό περιβάλλον λειτουργίας, η εφαρμογή τυπικά χρειάζεται να προσαρμόζεται κατά τη διάρκεια της εγκατάστασης σε κάθε περιβάλλον λειτουργίας, ώστε να μπορεί να σχετίζεται σε διαφορετικά σχήματα βάσεων δεδομένων.

    - Η αρχιτεκτονική EJB επιτρέπει στον προγραμματιστή να διαχωρίζει την συνηθισμένη επιχειρησιακή λογική της εφαρμογής από την συνηθισμένη λογική κατά την εκτέλεση της εγκατάστασης.

    - Η αρχιτεκτονική EJB σε ένα entity bean μπορεί να συνδέεται με διαφορετικά σχήματα βάσεων δεδομένων. Για'αυτό η διασύνδεση γίνεται κατά την διαδικασία της εγκατάστασης. Ο προγραμματιστής των εφαρμογών μπορεί να γράψει κώδικα ο οποίος δεν περιορίζεται σε έναν απλό τύπο συστήματος διαχείρισης βάσης δεδομένων ή ένος σχήματος βάσης δεδομένων.

    - Η EJB αρχιτεκτονική διευκολύνει την ανάπτυξη μιας εφαρμογής μέσω της επιβολής προδιαγραφών εγκατάστασης, όπως εκείνα για εύρεση πόρων, άλλων εξαρτήσεων εφαρμογών , ρυθμίσεων ασφαλείας κτλ. Οι προδιαγραφές ενεργοποιούν τη χρήση εργαλείων ανάπτυξης. Οι προδιαγραφές και τα εργαλεία κάνουν σχεδόν μηδενική την πιθανότητα παρανόησης στην επικοινωνία μεταξύ προγραμματιστών και αυτού που θα κάνει την εγκατάσταση.

    - Κατανεμημένη εγκατάσταση. Η αρχιτεκτονική EJB κάνει δυνατή την εγκατάσταση εφαρμογών με έναν κατανεμημένο τρόπο μεταξύ πολλαπλών εξυπηρετητών ενός δικτύου. Ο προγραμματιστής bean δεν χρειάζεται να ανησυχεί για την τοπολογία της εγκατάστασης όταν αναπτύσσει ένα enterprise bean αλλά να γράφει τον ίδιο κώδικά είτε ο πελάτης του enterprise bean είναι στην ίδια ή σε διαφορετική μηχανή.

- **Διαλειτουργικότητα εφαρμογής.** Η EJB αρχιτεκτονική κάνει πιο εύκολη την ολοκλήρωση εφαρμογών από διαφορετικούς κατασκευαστές. Η διεπαφή με την οποία ο πελάτης βλέπει το enterprise bean προσφέρεται, ως το σημείο για να γίνει η ολοκλήρωση μεταξύ των εφαρμογών.

- **Ολοκλήρωση με συστήματα που δεν είναι γραμμένα σε Java**. Τα σχετικά API J2EE, όπως η προδιαγραφή J2EE Connector και ή προδιαγραφή Java™ Message Service, και οι J2EE Web services τεχνολογίες , όπως το Java™ API για RPC (JAX-RPC) βασισμένο σε XML, κάνουν δυνατή την ολοκλήρωση enterprise bean εφαρμογών με ποικίλες εφαρμογές που δεν είναι γραμμένες σε Java, όπως τα συστήματα ERP ή εφαρμογών μεγάλων υπολογιστικών μονάδων (mainframe), με ένα προτυποποιήμενο τρόπο.

- **Εκπαιδευτικοί πόροι και εργαλεία ανάπτυξης**. Για το λόγο ότι η EJB αρχιτεκτονική αποτελεί ένα βιομηχανική προδιαγραφή, οι προγραμματιστές EJB εφαρμογών αποκτούν συνεχώς πλεονεκτήματα από την αύξηση της εκπαιδευτικής πληροφορίας που είναι σχετική στο πως θα φτιάχνουν εφαρμογές EJB. Αρκετά σημαντικό είναι επίσης ότι, τα πανίσχυρα εργαλεία ανάπτυξης που διατίθενται από τους κατασκευαστές εργαλείων απλοποιούν την ανάπτυξη και τη συντήρηση των εφαρμογών EJB.

3.6.2 Πλεονεκτήματα από την πλευρά των αγοραστών - πελατών

Η οπτική ενός πελάτη , πάνω στην αρχιτεκτονική EJB είναι διαφορετική από εκείνη του προγραμματιστή εφαρμογών. Η αρχιτεκτονική EJB παρέχει τα ακόλουθα πλεονεκτήματα στον πελάτη: επιλογή του εξυπηρετητή εφαρμογών, διευκόλυνση του διαχειριστή της εφαρμογής, ολοκλήρωση με τις υπάρχουσες εφαρμογές και δεδομένων του πελάτη, ολοκλήρωση με επιχειρησιακές εφαρμογές των πελατών, συνεργατών και προμηθευτών, και ασφάλεια εφαρμογών.

- **Επιλογή του εξυπηρετητή**. Επειδή η αρχιτεκτονική EJB αποτελεί μια ευρέως διαδεδομένη βιομηχανική προδιαγραφή και αποτελεί μέρος της πλατφόρμας J2EE , οι οργανισμοί των πελατών έχουν να επιλέξουν από μια πλατιά γκάμα συμβατών εξυπηρετητώνJ2EE . Οι πελάτες μπορούν να επιλέξουν ένα προϊόν το οποίο συναντά τις δικές τους ανάγκες που συνήθως έχουν να κάνουν με επεκτασιμότητα , δυνατότητες ολοκλήρωσης με άλλα συστήματα, με πρωτόκολλα ασφάλειας, τιμή, κτλ. Οι πελάτες δεν δεσμεύονται με συγκεκριμένα προϊόντα κατασκευαστών. Επειδή θα πρέπει οι ανάγκες τους να αλλάζουν , οι πελάτες θα πρέπει να μπορούν εύκολα να επαναγκαταστήσουν μια εφαρμογή EJB σε έναν εξυπηρετητή άλλου κατασκευαστή.

- **Διευκόλυνση της διαχείρισης των εφαρμογών.** Επειδή η αρχιτεκτονική EJB παρέχει ένα προτυποποιήμενο περιβάλλον, οι κατασκευαστές εξυπηρετητών πρέπει να έχουν τα κίνητρα να αναπτύσσουν εργαλεία διαχείρισης για την βελτίωση των προϊόντων . Ως αποτέλεσμα έχουμε την παροχή εξειδικευμένων εργαλείων διαχείρισης τα οποία παρέχονται με τον υποδοχέα EJB επιτρέποντας το τμήμα Πληροφορικής Τεχνολογίας (Information Technology- IT) του πελάτη να μπορεί να εκτελεί, μεταξύ άλλων λειτουργίες, όπως εκκίνηση και σταμάτημα της εφαρμογής, κατανομή των πόρων συστήματος στην εφαρμογή, και παρακολούθηση παραβιάσεων ασφάλειας.

- **Ολοκλήρωση με τις υπάρχουσες εφαρμογές του πελάτη και των δεδομένων του**. Η EJB αρχιτεκτονική και άλλα σχετικά API J2EE απλοποιούν και προτυποποιούν την ολοκλήρωση των εφαρμογών EJB με μη-Java εφαρμογές και συστήματα στο λειτουργικό περιβάλλον του πελάτη. Για παράδειγμα , ένας πελάτης δεν πρέπει να τροποποιεί , σε καμία περίπτωση το σχήμα της βάσης δεδομένων , για να το προσαρμόσει την εφαρμογή. Αντιθέτως, μια εφαρμογή EJB θα πρέπει να ταιριάζει με το υπάρχον σχήμα της βάσης δεδομένων όταν αυτή εγκαθίστανται.

- **Ολοκλήρωση με επιχειρησιακές εφαρμογές των πελατών, συνεργατών και προμηθευτών**. Η διαλειτουργικότητα και τα χαρακτηριστικά υπηρεσιών ιστού (web services) της αρχιτεκτονικής EJB επιτρέπουν σε εφαρμογές που είναι βασισμένες στην αρχιτεκτονική EJB να διατίθενται ως υπηρεσίες ιστού σε άλλες επιχειρήσεις. Αυτό σημαίνει ότι οι επιχειρήσεις έχουν μια απλή, συνεπή τεχνολογία για ανάπτυξη intranet, Internet, και εφαρμογών ηλεκτρονικών επιχειρήσεων (e-business).

- **Ασφάλεια εφαρμογών**. Η αρχιτεκτονική EJB μεταφέρει το μεγαλύτερο μέρος της ευθύνης για την ασφάλεια των εφαρμογών από τον προγραμματιστή εφαρμογών στον κατασκευαστή των εξυπηρετητών, στον διαχειριστή συστήματος, και σε αυτόν που κάνει την εγκατάσταση των εφαρμογών. Οι άνθρωποι που εκτελούν αυτούς τους ρόλους είναι περισσότερο σχετικοί από τον προγραμματιστή της εφαρμογής για να θέσουν επίπεδα ασφάλειας στην εφαρμογή. Αυτό οδηγεί στην καλύτερη ασφάλεια των επιχειρησιακών εφαρμογών.

3.7 Πότε θα πρέπει κανείς να χρησιμοποιεί EJB και πότε όχι ;

Γενικά είναι καλό να κάνουμε χρήση αυτών. Παρόλα αυτά , για μερικές περιπτώσεις , τα EJB δεν είναι και ή καλύτερη λύση για ένα πρόβλημα. Καλό είναι να δούμε λοιπόν σε ποιες περιπτώσεις μας κάνουν και σε ποίες όχι. Σε αρκετές περιπτώσεις , όπως σε κάποιες συγκεντρωτικές επιχειρησιακές εφαρμογές τα EJB απλά , δεν είναι και ή καλύτερη επιλογή.

3.7.1 Πότε είναι καλό να γίνεται χρήση των EJB

Παρακάτω παρουσιάζονται οι περιπτώσεις που τα EJB αποτελούν σχεδιαστική λύση:

- **Επιχειρησιακές (business) δοσοληψίες απλών και πολλαπλών συστημάτων.** Η ικανότητα της διατήρησης της ακεραιότητας των δοσοληψιών για σύνθετα επιχειρησιακά entity αποτελεί ένα από τα δυνατά σημεία των EJB. Τα EJB δεν είναι μόνα στη παροχή σαφούς ελέγχου δοσοληψιών κατά την αποθήκευση απλών δεδομένων. Παρόλα αυτά , τα EJB μεγαλουργούν όπου υπάρχουν πολλαπλοί πόροι (σχεσιακές βάσεις δεδομένων, συστήματα μηνυμάτων κτλ.) επειδή επιτρέπουν πραγματοποίηση δοσοληψιών με αρκετούς και διαφορετικούς πόρους, και έτσι με αυτόν τον τρόπο οι διάφοροι πόροι υποστηρίζουν κατανεμημένες δοσοληψίες.

- **Κατανεμημένη λειτουργικότητα.** Συχνά οι επιχειρησιακές υπηρεσίες εκτελούνται σε έναν απομακρυσμένο εξυπηρετητή. Για παράδειγμα , μια επιχείρηση που θα έχει αρκετά διαφορετικά συστήματα , με αρκετές διαβαθμίσεις που έχουν να κάνουν με δυσκαμψίες της εφαρμογής καθώς και σε θέματά προστασίας. Ένα από αυτά τα συστήματα μπορεί να έχει πρόσβαση στα άλλα. Τα EJB, τα οποία είναι κατανεμημένα από τη φύση τους , αποτελούν τον πιο απλό τρόπο για κατανεμημένες- απομακρυσμένες υπηρεσίες Τα EJB επίσης επιτρέπουν την παροχή επιχειρησιακών υπηρεσιών σε απομακρυσμένους πελάτες απλούστερα σε σχέση με άλλους εναλλακτικούς. Η απομακρυσμένη πρόσβαση στις βάσεις δεδομένων, δια μέσου συστατικών λογισμικού είναι πιο εύκολη για συντήρηση σε σχέση με την απευθείας, επειδή ο κώδικάς των συστατικών λογισμικού μπορεί να προστατέψει τον πελάτη από αλλαγές στο σχήμα της βάσης δεδομένων.

- **Φορητά συστατικά λογισμικού (όχι κλάσεις)**. Έως τώρα , εάν κάποιος ήθελε να μοιραστεί τις επιχειρησιακές υπηρεσίες του με κάποιον άλλο προγραμματιστή εφαρμογών, θα αναγκαζόταν να μοιραστεί τις κλάσεις ή τουλάχιστον τις συσκευασίες. Η γλώσσα Java δεν επιτρέπει με εύκολο τρόπο τη δημιουργία, επιχειρησιακών συστατικών λογισμικού ή επαναχρησιμοποιήσιμων δομικών κομματιών λογισμικού , που μπορούν να δομηθούν μαζί με άλλα συστατικά λογισμικού και να σχηματίσουν μια εφαρμογή. Τα EJB επιτρέπουν τη συσκευασία κομματιού της επιχειρησιακής λογικής (business logic) σε μια τακτική, κατανεμημένη μονάδα ή οποία μπορεί να διαμοιράζεται με τρόπο χαλαρής σύζευξης. Ο χρήστης των συστατικών λογισμικού χρειάζεται μόνο ένα αρχείο περιγραφής για το περιβάλλον.

- **Εμπιστοσύνη μεταξύ των εφαρμογών στην ασύγχρονή μετάδοση μηνυμάτων**. Τα EJB (ειδικά τα MDB) παρέχουν μια ισχυρή τεχνολογία για χειρισμό ασύγχρονής επικοινωνίας όπως μετάδοση μηνυμάτων βασιζόμενη σε JMS ή σε υπηρεσίες ιστού.

- **Ρόλοι ασφάλειας.** Εάν οι επιχειρησιακές λειτουργίες μπορούν να αντιστοιχηθούν σε εξειδικευμένους επιχειρησιακούς ρόλους στις εφαρμογές, τότε τα EJB ίσως είναι μια καλή επιλογή. Τόσα πολλά έγιναν από την δυνατότητα διαχείρισης τον δοσοληψιών των EJB που τα χαρακτηριστικά ασφαλείας που βασίζονται στον περιγραφέα εγκατάστασης (deployment descriptor) τους πλέον παραβλέπονται. Είναι αυτή η δυνατότητα πολύ ισχυρή; Ναι, εάν στους χρήστες της εφαρμογής έχουν αποδοθεί διακριτοί ρόλοι και οι κανόνες αυτών των ρόλων υπαγορεύουν ποιοι χρήστες μπορούν να γράψουν ποια δεδομένα, τότε τα EJB είναι καλή επιλογή.

- **Διαχείριση persistence context.** Η διαχείριση του κύκλου ζωής από τον διαχειριστή οντοτήτων (entity manager), που έχει δημιουργηθεί από ένα `EntityManagerFactory`, μπορεί να είναι κάπως σύνθετη διαδικασία, αφού εάν κάνεις κάνει εμφωλευμένες κλήσεις αντικειμένων , χρειάζεται πρώτα να περνάει το στιγμιότυπο του διαχειριστή οντοτήτων ως παράμετρο. Όμως ακόμα και αν η εφαρμογή είναι απλή , τα EJB παρέχουν καλή διαχείριση persistence context η οποία θα συρρικνώσει και θα απλοποιήσει τον κώδικά που γράφεται.

3.7.2 Πότε δεν είναι καλό να γίνεται χρήση των EJB

Σε αρκετές περιπτώσεις κτισίματος μιας επιχειρησιακής εφαρμογής λογισμικού, τα EJB ίσως στην πραγματικότητα να αποτελούν εμπόδιο για την ικανοποίηση των επιχειρησιακών στόχων που έχουν μπει. Παρακάτω παρατίθενται οι περιπτώσεις που δεν θα ήταν αποδοτική η χρήση των EJB :

- **Εφαρμογές που χρειάζονται έλεγχο νημάτωσης**. Εάν η σχεδίαση της εφαρμογής απαιτεί εκτεταμένη χρήση νημάτωσης τότε, ή προδιαγραφή EJB πραγματικά αποτρέπει για τη χρήση EJB (αν και κάποιοι κατασκευαστές υποδοχέων EJB επιτρέπουν μη φορητούς τρόπους σε σχέση με αυτόν τον περιορισμό). Τι γίνεται όμως με τα συστήματα διαχείρισης πόρων των υποδοχέων ,δοσοληψίες, ασφάλειας, και άλλων ποιοτικών υπηρεσιών που κάνουν χρήση νημάτωσης; Απλώς τα νήματα που θα χρειαστούν, θα δημιουργηθούν έκτος του έλεγχου του υποδοχέα και πιθανώς να προκαλέσουν σφάλματα συστήματος. Επίσης , οι υποδοχείς EJB επιτρέπεται να κατανέμουν EJB δια πολλαπλών JVMs, αποτρέποντας έτσι, τον συγχρονισμό της νημάτωσης.

- **Απόδοση κατά την εκτέλεση**. Επειδή τα EJB κάνουν πολλά περισσότερα από τις απλές κλάσεις Java, είναι πιο αργά από αυτές. Ο υποδοχέας EJB πρέπει να κάνει αρκετά: διατήρηση της ακεραιότητας των δοσοληψιών, διαχείριση των bean στιγμιότυπων των «δεξαμενών» bean (pools), επιβολή ρόλων ασφάλειας, διαχείριση πόρων , συντονισμού κατανεμημένων λειτουργιών , συγχρονισμού διαμοιραζομένων υπηρεσιών (εάν ο κατασκευαστής προσφέρει clustering δυνατότητες), κ.α. Στις πιο πολλές εφαρμογές ή παραπάνω θεώρηση ίσως μην λαμβάνει τόσο μεγάλη κλίμακα , άλλα αν κανείς εκτελεί αρκετές και δαπανηρές από άποψη υπολογιστικής ισχύος λειτουργίες, η χρήση των EJB ίσως να είναι ακατάλληλη.

- **Επιτήρηση (babysitting) της βάσης δεδομένων**. Οι εφαρμογές που από μόνες τους επιτηρούν μια βάση δεδομένων ίσως να μην χρειάζονται τον έλεγχο persistence context που παρέχουν τα EJB επειδή αυτές κάνουν (συνεχείς και επαναλαμβανόμενες) αναβαθμίσεις ρουτίνας της βάσης δεδομένων ή σειριακή επεξεργασία. Σε αυτή την περίπτωση , τα EJB ίσως να είναι υπερβολή.

Μέρος Β΄

4

Μια μελέτη περίπτωσης: Εφαρμογή διαχείρισης μαθημάτων SchoolApplication

Το κομμάτι της ανάπτυξης είχε να κάνει με τη σχεδίαση και υλοποίηση μιας επιχειρησιακής εφαρμογής διαχείρισης μαθημάτων: SchoolApplication, με επιχειρησιακό (ejb) και στρώμα ιστού (web), χρησιμοποιώντας τεχνολογίες J2EE (servlet, xml, EJB κτλ.). Κύριος σκοπός της ανάπτυξης ήταν η σχεδίαση και η υλοποίηση μιας βάσης δεδομένων πραγματικών απαιτήσεων όσο το δυνατόν περισσοτέρων χαρακτηριστικών (διαφορετικών τύπων : δεδομένων και ιδιοτήτων οντοτήτων , συσχετίσεων οντοτήτων, πρωτευόντων κλειδιών, πολλαπλότητας και κατεύθυνσης συσχετίσεων, κληρονομικότητας, περιορισμών κ.τ.λ.), με τη χρήση της νέας έκδοσης της τεχνολογίας Enterprise Java Bean (EJB3).

Κατά την διαδικασία εισόδου του χρήστη στην εφαρμογή παρέχεται ένα πρωταρχικό επίπεδο ασφάλειας ιστού, για την πιστοποίηση και εξουσιοδότηση των χρηστών από το εξυπηρετητή

. Έτσι απαιτείται από αυτόν η εισαγωγή των προσωπικών στοιχείων (όνομα και κωδικός) του χρήστη σε με μια login σελίδα . Η εφαρμογή έχει τη δυνατότητα εισόδου τριών διαφορετικών κατηγοριών εξουσιοδοτημένων χρηστών: Διαχειριστή (administrator), Σπουδαστή (student) και Καθηγητή (professor), όπου στον καθένα από αυτούς παρέχεται ένα διαφορετικό πλήθος δυνατοτήτων χρήσης, το οποίο έχει να κάνει με τη φυσική σχέση της κάθε κατηγορίας χρηστών με

τη γραμματεία της υποθετικής σχολής , για την οποία προορίζεται υποθετικά η εφαρμογή SchoolApplication. Έτσι σχηματικά : ο Διαχειριστής μέσα από το δικό του μενού μπορεί να εκτελέσει τις συνήθεις λειτουργίες διαχείρισής μιας βάσης δεδομένων, οι οποίες έχουν να κάνουν με : προσθήκη, διαγραφή, επεξεργασία και εύρεση μιας οντότητας (γραμμής πίνακα) καθώς και με προβολές των υπαρχόντων συνόλων οντοτήτων (πίνακας). Ο Καθηγητής μέσα απ'το μενού του μπορεί να δει τα προσωπικά του στοιχεία, τα μαθήματα που διδάσκει, τους σπουδαστές που παρακολουθούν σε κάθε μάθημα του, τις διδασκαλίες που έχει κ.α. Ο Σπουδαστής μέσα απ'το μενού του πέρα ότι μπορεί να δει τα προσωπικά του στοιχεία, τα μαθήματα που διδάσκονται , τις διδασκαλίες που γίνονται για ένα συγκεκριμένο μάθημα, έχει τη δυνατότητα να κάνει δήλωση μαθημάτων (τρία μαθήματα ανά εξάμηνο) για το νέο εξάμηνο με επιπλέον δυνατότητα διόρθωσης τυχόν λανθασμένης επιλογής του. Η σύνδεση και ή διαπροσωπεία των χρηστών με την εφαρμογή γίνεται σε περιβάλλον ιστού μέσω του πρωτοκόλλου HTTP, με δυνατότητα χρήσης της εφαρμογής από πολλούς χρήστες ταυτόχρονα.

5

Ανάλυση απαιτήσεων από την εφαρμογή

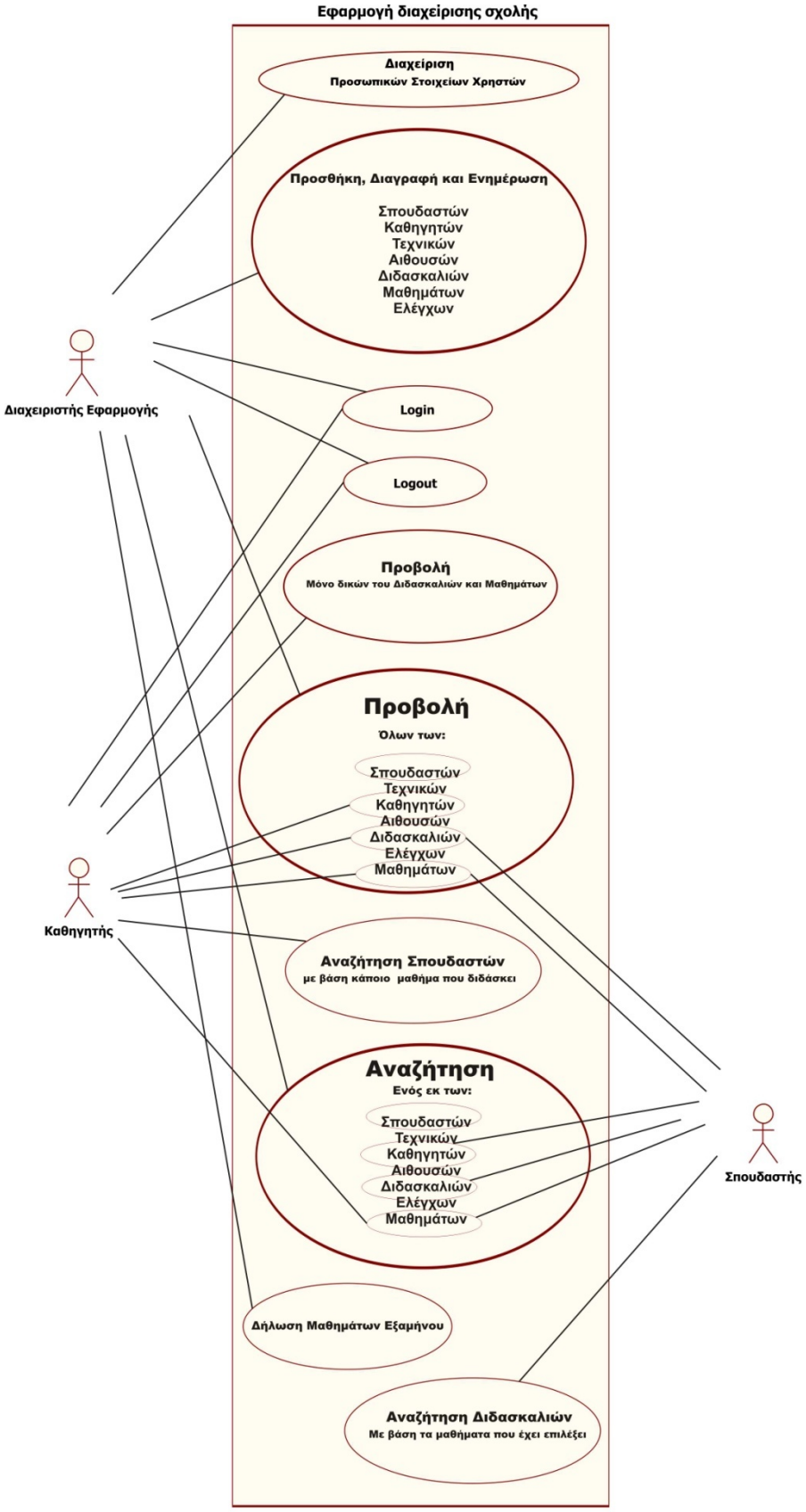
Στο κεφάλαιο αυτό παρουσιάζονται, η ανάλυση απαιτήσεων της εφαρμογής με τη μορφή διαγραμμάτων περιπτώσεων χρήσης, καθώς και το διάγραμμα κλάσεων με στο οποίο ορίζονται οι βασικές οντότητες και οι σχέσεις αυτών βάσει των απαιτήσεων μας.

5.1 Περιπτώσεις χρήσης

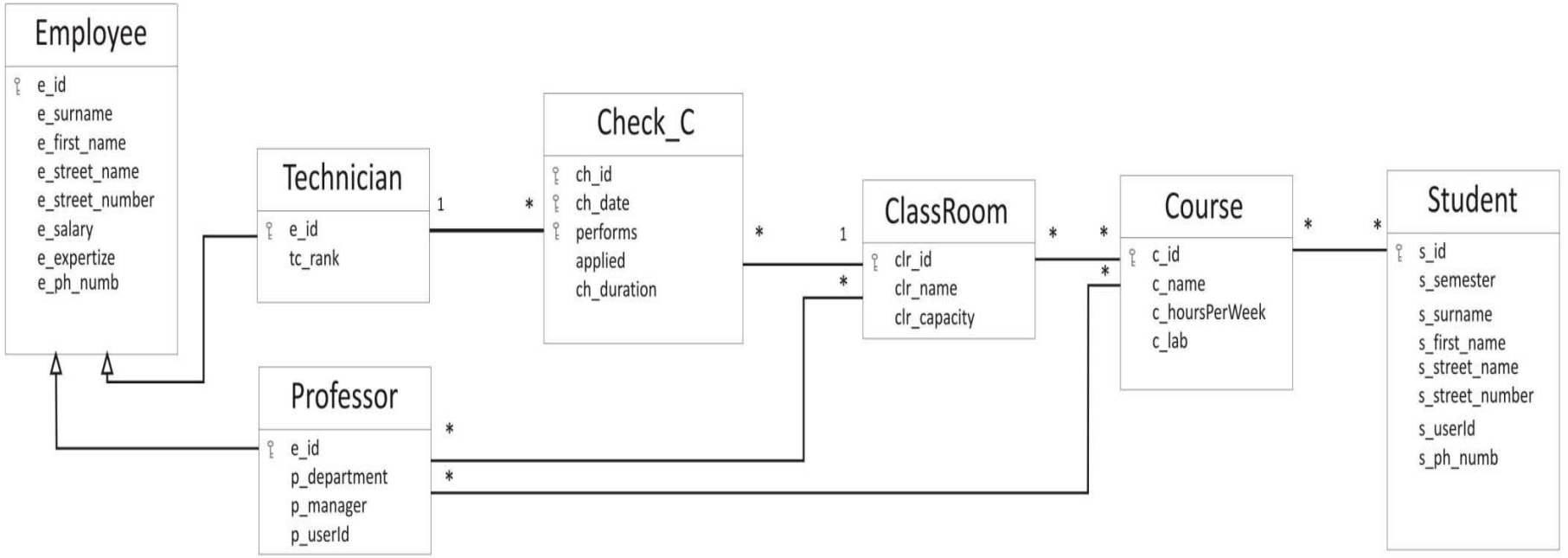
Στην ενότητα αυτή παρουσιάζονται τα σενάρια χρήσης του προγράμματος για τις τρείς κατηγορίες εξουσιοδοτημένων χρηστών , όπως αυτά παρουσιάζονται στο παρακάτω διάγραμμα περιπτώσεων χρήσης (Σχήμα 5.1).

5.2 Η περιγραφή των οντοτήτων

Στην ενότητα αυτή παρουσιάζονται οι περιγραφές για τις βασικές κλάσεις και τις σχέσεις τους σε αφηρημένο επίπεδο (entity model), όπου αποδίδονται με το παρακάτω διάγραμμα οντοτήτων (Σχήμα 5.2).



Σχήμα 5.1 Περιπτώσεις χρήσης της εφαρμογής



Σχήμα 5.2 Διάγραμμα κλάσεων

6

Ανάπτυξη της εφαρμογής

6.1 Σχεδίαση της εφαρμογής και υλοποίηση της σχεδίασης

Εδώ θα παρουσιάζεται η σχεδίαση και η υλοποίηση της σχεδίασης των τριών διαφορετικών στρωμάτων που συνιστούν την εφαρμογή SchoolApplication. Του επιχειρησιακού (ejb) στρώματος το οποίο υλοποιείται μέσω κλάσεων EJB και αρχείων xml, του στρώματος της βάσης δεδομένων (EIS-Enterprise Information System) το οποίο υλοποιείται μέσω του επιχειρησιακού στρώματος από τα entity beans , και του στρώματος ιστού (web) το οποίο υλοποιείται μέσω κλάσεων servlet και αρχείων xml.
Η σχεδίαση της εφαρμογής κάνει χρήση προγραμματιστικών υποδειγμάτων (pattern) για τη διασύνδεση του στρώματος παρουσίασης (web) του πελάτη με το επιχειρησιακό (ejb) στρώμα.

6.1.1 Προγραμματιστικά υποδείγματα (pattern)

6.1.1.1 Τι είναι και τι προσφέρουν τα προγραμματιστικά υποδείγματα ;

Τα προγραμματιστικά υποδείγματα, γενικώς μπορούμε να πούμε ότι, υποδεικνύουν αρχιτεκτονικές για την επίλυση συνήθων προβλημάτων προγραμματιστικής φύσης που συναντώνται από τους προγραμματιστές κατά την ανάπτυξη εφαρμογών.

Κάποια χαρακτηριστικά που προσφέρουν τα προγραμματιστικά υποδείγματα είναι :

- Τα προγραμματιστικά υποδείγματα είναι τυπικά γραμμένα έχοντας δομημένη μορφή.

- Τα προγραμματιστικά υποδείγματα προστατεύουν τους προγραμματιστές από την («ανακάλυψη του τροχού») επίλυση προβλημάτων σχεδίασης που έχουν ήδη παρουσιαστεί στο παρελθόν και έχουν επιλυθεί από άλλους με αποδοτικό τρόπο.
- Τα προγραμματιστικά υποδείγματα υπάρχουν σε διαφορετικά επίπεδα αφαίρεσης.
- Τα προγραμματιστικά υποδείγματα βρίσκονται υπό διαρκή ανάπτυξη.
- Τα προγραμματιστικά υποδείγματα αποτελούν επαναχρησιμοποιήσιμα τεχνουργήματα.
- Τα σχέδια επικοινωνίας των προγραμματιστικών υποδειγμάτων αποτελούν τις πιο αποδοτικές πρακτικές.
- Τα προγραμματιστικά υποδείγματα μπορούν να χρησιμοποιηθούν μαζί για την επίλυση μεγάλης έκτασης προγραμματιστικών προβλημάτων σχεδίασης.

6.1.1.2 Το πρόβλημα της επικοινωνίας των διαφορετικών στρωμάτων σε μια πολυστρωματική εφαρμογή ;

Ένα πολυστρωματικό , κατανεμημένο σύστημα χρειάζεται απομακρυσμένες κλήσεις για τη λήψη δεδομένων μεταξύ των στρωμάτων. Οι πελάτες εκτίθονται μέσω της επαφής τους με κατανεμημένα συστατικά λογισμικού. Το στρώμα παρουσίασης αλληλεπιδρά απευθείας με τις υπηρεσίες του επιχειρησιακού στρώματος. Αυτή, η απευθείας αλληλεπίδραση, εκθέτει τις λεπτομέρειες υλοποίησης της υποδομής, της προγραμματιστικής διεπαφής της εφαρμογής (API) των επιχειρησιακών υπηρεσιών, στο στρώμα παρουσίασης. Ως αποτέλεσμα του παραπάνω, τα συστατικά λογισμικού του στρώματος παρουσίασης είναι ευάλωτα σε αλλαγές, που μπορεί να υποστούν οι υπηρεσίες του επιχειρησιακού στρώματος. Όταν η υλοποίηση των υπηρεσιών του επιχειρησιακού στρώματος αλλάξει, ο εκτεθειμένος κώδικας υλοποίησης στο στρώμα παρουσίασης θα πρέπει να αλλάξει επίσης.

Επιπλέον, ίσως μπορεί να υπάρχει μια επιβάρυνση της απόδοσης του δικτύου, για το λόγο ότι στοιχεία λογισμικού του στρώματος παρουσίασης τα οποία χρησιμοποιούν το API των υπηρεσιών του επιχειρησιακού στρώματος θα κάνουν αρκετές κλήσεις υπηρεσιών μέσω του δικτύου (περίπτωση εφαρμογής με απομακρυσμένη διεπαφή : επιχειρησιακό στρώμα και στρώμα παρουσίασης σε διαφορετικές εικονικές μηχανές Java (JVM)).

Τέλος, εκθέτοντας τα API των υπηρεσιών του επιχειρησιακού στρώματος απευθείας στον πελάτη, επιβάλλουν στον πελάτη να τις μοιραστεί μαζί με τα άλλα χαρακτηριστικά της τεχνολογίας EJB που σχετίζονται με την κατανεμημένη φύση της.

6.1.1.3 Ποιες απαιτήσεις πρέπει να πληροί η σχέση μεταξύ του επιχειρησιακού στρώματος και του στρώματος παρουσίασης και πως ικανοποιούνται αυτές με τη χρήση προγραμματιστικού υποδείγματος.

Οι απαιτήσεις μεταξύ των δύο στρωμάτων είναι :

- Οι πελάτες του στρώματος παρουσίασης χρειάζονται πρόσβαση στις επιχειρησιακές υπηρεσίες.
- Οι διαφορετικοί πελάτες, όπως οι συσκευές , οι πελάτες ιστού κτλ. χρειάζονται πρόσβαση στις υπηρεσίες του επιχειρησιακού στρώματος.
- Τα API των υπηρεσιών του επιχειρησιακού στρώματος πιθανώς να αλλάξουν με τις επιχειρησιακές απαιτήσεις.
- Είναι επιθυμητή η ελαχιστοποίηση της σύζευξης μεταξύ των πελατών του στρώματος παρουσίασης και των υπηρεσιών του επιχειρησιακού στρώματος, με την απόκρυψη των λεπτομερειών της φυσικής υποδομής των υπηρεσιών, όπως επίσης και με την εύρεση και την προσπέλαση αυτών.
- Οι πελάτες μπορεί να χρειάζονται να υλοποιούν μηχανισμούς αποθήκευσης της πληροφορίας των υπηρεσιών του επιχειρηματικού στρώματος.
- Είναι επιθυμητό να μειωθεί η συμφόρηση του δικτύου μεταξύ πελάτη και υπηρεσιών επιχειρησιακού στρώματος.

Η λύση :

Η χρήση προγραμματιστικού υποδείγματος Επιχειρησιακού Εκπροσώπου (Business Delegate) για τη μείωση της σύζευξης μεταξύ πελατών στρώματος παρουσίασης και υπηρεσιών επιχειρησιακού στρώματος. Μια κλάση Επιχειρησιακού Εκπροσώπου κρύβει, τις λεπτομέρειες υλοποίησης των υπηρεσιών του επιχειρησιακού στρώματος και τις λεπτομέρειες των διαδικασιών εύρεσης και πρόσβασης, καθώς επίσης και στοιχεία της αρχιτεκτονικής EJB αυτών. Η κλάση Επιχειρησιακού Εκπροσώπου δρα ως ένα επίπεδο αφαίρεσης του επιχειρηματικού στρώματος από την πλευρά του πελάτη. Παρέχει ένα επίπεδο αφαίρεσης, το οποίο κρύβει από τον πελάτη την υλοποίηση των υπηρεσιών του επιχειρησιακού στρώματος. Κάνοντας χρήση προγραμματιστικού υποδείγματος

Επιχειρησιακού Εκπροσώπου μειώνεται οι ζεύξη του στρώματος παρουσίασης των πελατών και των επιχειρησιακών υπηρεσιών του συστήματος. Εξαρτώμενη από την πολιτική υλοποίησης, η κλάση Επιχειρησιακού Εκπροσώπου προστατεύει του χρήστες από πιθανές αλλαγές στην υλοποίηση του API των υπηρεσιών του επιχειρησιακού στρώματος. Πιθανά, αυτό μειώνει τον αριθμό των αλλαγών οι οποίες πρέπει να γίνουν στο κώδικα του στρώματος παρουσίασης του πελάτη όταν οι των υπηρεσίες του επιχειρησιακού στρώματος ή υλοποίηση της υποδομής τους αλλάξει. Παρόλαυτα, οι μέθοδοι διεπαφής στην κλάση Επιχειρησιακού Εκπροσώπου θα χρειαστούν τροποποίηση εάν η επιχειρησιακή υποδομή αλλάξει. Ομολογουμένως, είναι πιο επιθυμητό αυτές οι αλλαγές να γίνονται στις υπηρεσίες του επιχειρησιακού στρώματος παρά μάλλον στην κλάση Επιχειρησιακού Εκπροσώπου.

Το κύριο πλεονέκτημα χρήσης του προγραμματιστικού υποδείγματος Επιχειρησιακού Εκπροσώπου, όπως ειπώθηκε παραπάνω , είναι η απόκρυψη λεπτομερειών της υλοποίησης των υποδομών των υπηρεσιών του επιχειρησιακού στρώματος. Για παράδειγμα, ο πελάτης μπορεί να είναι διαφανής από τις υπηρεσίες ονομάτων (naming) και εύρεσης (lookup). Η κλάση Επιχειρησιακού Εκπροσώπου μπορεί επίσης να διαχειρίζεται τις εξαιρέσεις (exceptions) των υπηρεσιών του επιχειρησιακού στρώματος, όπως είναι οι εξαιρέσεις του java.rmi.Remote, Java Messages Service (JMS) κτλ. Η κλάση Επιχειρησιακού Εκπροσώπου μπορεί να διακόπτει εξαιρέσεις υπηρεσιών αυτού του επιπέδου και να δημιουργεί αντί για αυτές εξαιρέσεις επίπεδου εφαρμογών. Οι εξαιρέσεις επιπέδου εφαρμογών είναι περισσότερο διαχειρίσιμες από τους πελάτες, και περισσότερο φιλικές προς τον χρήστη της εφαρμογής. Επίσης η κλάση Επιχειρησιακού Εκπροσώπου διαχειρίζεται γεγονότα που έχουν να κάνουν με σφάλματα των υπηρεσιών που εκτελούνται κρύβοντας τα ταυτόχρονα από το χρήστη. Έτσι προβλήματα κατά την εκτέλεση δεν παρουσιάζονται στον χρήστη, μέχρις ότου να επιβεβαιωθεί ότι δεν επιδέχεται περαιτέρω λύση. Ένα άλλο πλεονέκτημα είναι ότι η κλάση Επιχειρησιακού Εκπροσώπου μπορεί να χρησιμοποιηθεί για να αποθηκεύει αποτελέσματα και αναφορές από απομακρυσμένες υπηρεσίες επιχειρησιακού στρώματος. Η αποθήκευση αυτή επιφέρει πολλά στην απόδοση επειδή περιορίζει μη χρειαζούμενές και ενδεχομένως δαπανηρές επαναλαμβανόμενες κλήσεις αντικειμένων κατά το μήκος του δικτύου.

Η κλάση Επιχειρησιακού Εκπροσώπου χρησιμοποιεί ένα συστατικό λογισμικού που ονομάζεται Lookup Service (υπηρεσία εύρεσης). Η υπηρεσία εύρεσης είναι υπεύθυνη για την απόκρυψη των λεπτομερειών του κώδικα υλοποίησης της υποδομής των υπηρεσιών της στο επιχειρησιακό στρώμα. Η υπηρεσία εύρεσης μπορεί να αποτελεί μέρος της κλάσης Επιχειρησιακού Εκπροσώπου, αλλά συνίσταται να υλοποιείται ως ξεχωριστώ συστατικό

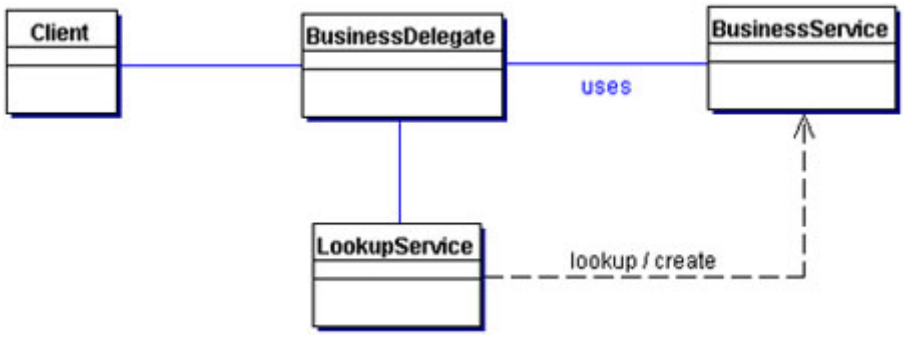
λογισμικού, όπως άλλωστε περιγράφεται στο προγραμματιστικό υπόδειγμα της υπηρεσίας τοποθεσίας (Service Locator).

Όταν το υπόδειγμα Επιχειρησιακού Εκπροσώπου χρησιμοποιείται με το υπόδειγμα Session Facade (εκπροσώπου των μεθόδων των EJBs στο επιχειρησιακό στρώμα), τυπικά υπάρχει μια σχέση ένα - προς - ένα μεταξύ αυτών. Αυτή η σχέση είναι ένα - προς - ένα επειδή η λογική η οποία εμπεριέχεται στην κλάση Επιχειρησιακού Εκπροσώπου σχετίζεται με την αλληλεπίδραση της με πολλές υπηρεσίες επιχειρησιακού στρώματος (δημιουργώντας μια σχέση ένα – προς – πολλά) η όποια συχνά παράγεται πίσω στην κλάση Session Facade.

Τελικώς, θα πρέπει να επισημανθεί ότι αυτό το προγραμματιστικό υπόδειγμα χρησιμοποιείται για να μειώσει τη ζεύξη μεταξύ διαφορετικών στρωμάτων, και όχι απλώς του στρώματος παρουσίασης και του επιχειρησιακού.

6.1.1.4 Η δομή του προγραμματιστικού υποδείγματος Επιχειρησιακού Εκπροσώπου

Το παρακάτω σχήμα παρουσιάζει το διάγραμμα κλάσεων του προγραμματιστικού υποδείγματος Επιχειρησιακού Εκπροσώπου. Ο πελάτης κάνει αίτηση στην κλάση προγραμματιστικού υποδείγματος Επιχειρησιακού Εκπροσώπου (Business Delegate) για την παροχή σε αυτόν πρόσβαση σε υπηρεσία του επιχειρησιακού στρώματος που βρίσκεται κάτω από την διεύθυνση της. Η κλάση Business Delegate χρησιμοποιεί μια υπηρεσία εύρεσης (LookupService) για την εύρεση του αιτούμενου συστατικού λογισμικού της υπηρεσίας του επιχειρησιακού στρώματος (BusinessService).



Σχήμα 6.1 Διάγραμμα κλάσης Επιχειρησιακού Εκπροσώπου (BusinessDelegate)

Λίστα 6.1 Τμήμα κώδικα της κλάσης Επιχειρησιακού Εκπροσώπου της εφαρμογής μας : Delegate.java

```
public class Delegate  {

    public Delegate()   {     }

// Υλοποίηση της μεθόδου lookupSchoolFacade
SchoolFacadeLocal lookupSchoolFacade() {
        try {
            //Παίρνει το Path που βρίσκεται η εφαρμογή
            Context c = new InitialContext();
            //αι επιστρέφει το τοπικό interface της εφαρμογής
            return                                          (SchoolFacadeLocal)
c.lookup("SchoolApp/SchoolFacade/local");
        }
        catch(NamingException ne) {

Logger.getLogger(getClass().getName()).log(Level.SEVERE,"exception   caught"
,ne);
            throw new RuntimeException(ne);
        }
    }
   //------------------- Μέθοδοι που υλοποιούνται οι οποίες  βρίσκονται στο
//EJB στρώμα
 //------------------------------ Ορισμοί Μεθόδων    ----------------------
    //--------Συνήθεις Ορισμοί Μεθόδων για Όλα τα σύνολα Οντοτήτων -------

  public void create(Object entity){

     this.lookupSchoolFacade().create(entity);
}

    public void edit(Object entity){

     this.lookupSchoolFacade().edit(entity);
}

    public void destroy(Object entity){
```

```
     this.lookupSchoolFacade().destroy(entity);
}
        public void refreshEntities(Object entity){

     this.lookupSchoolFacade().refreshEntities(entity);
}

//----Ορισμοί Μεθόδων Εύρεσης  Όλων των Οντοτήτων ενός  Συνόλου Οντοτήτων -

  public List findAllS(){

     return this.lookupSchoolFacade().findAllS();
}
    public List findAllC(){

     return this.lookupSchoolFacade().findAllC();
}
    public List findAllE(){

     return this.lookupSchoolFacade().findAllE();
}
    public List findAllP(){

     return this.lookupSchoolFacade().findAllP();
}
    public List findAllT(){

     return this.lookupSchoolFacade().findAllT();
}
```

6.1.2 Σχεδίαση και υλοποίηση του επιχειρησιακού στρώματος (βάσης δεδομένων) της εφαρμογής

Η σχεδίαση της βάσης δεδομένων (διάγραμμα οντοτήτων συσχετίσεων Σχήμα 6.1) της εφαρμογής έγινε με γνώμονα τη χρήση όσο το δυνατόν περισσοτέρων χαρακτηριστικών (διαφορετικών τύπων : δεδομένων και ιδιοτήτων οντοτήτων , συσχετίσεων οντοτήτων, πρωτευόντων κλειδιών, πολλαπλότητας και κατεύθυνσης συσχετίσεων, κληρονομικότητας, περιορισμών κ.τ.λ.) που συναντά κανείς σε μια σχεσιακή βάση δεδομένων.

Στόχος κατά την υλοποίηση της εφαρμογής ήταν η επαλήθευση της σχεδίασης από την προδιαγραφή EJB, ώστε παράλληλα να αναδειχθούν τα πλεονεκτήματα – μειονεκτήματα που φέρνει η υλοποίηση μιας σχεσιακής βάσης δεδομένων με την τεχνολογία EJB3. Έτσι, κατά την υλοποίηση της βάσης δεδομένων έγινε προσπάθεια υλοποίησης του μοντέλου οντοτήτων συσχετίσεων της βάσης δεδομένων που καταρτίστηκε κατά τη φάση της σχεδίασης της εφαρμογής.

Επιπλέον, ακολουθώντας την προδιαγραφή EJB στο στρώμα αυτό αναπτύσσεται η διεπαφή του για την διασύνδεση του στρώματος αυτού με τον "έξω κόσμο" . Έγινε επιλογή τοπικής διεπαφής (SchoolFacadeLocal), πράγμα που σημαίνει ότι είναι απαραίτητο το στρώμα ιστού (web) να βρίσκεται στην ίδια μηχανή εκτέλεσης της γλώσσας Java (Java Virtual Machine).

6.1.2.1 Σχεδίαση

Έγινε προσπάθεια , όσο το δυνατόν η βάση να ανταποκρίνεται σε πραγματικές καταστάσεις και με πνεύμα να παρουσιαστούν οι περισσότερες λειτουργίες που βρίσκουμε σε μια συνηθισμένη βάση. Προϊόν της σχεδίασης του επιχειρησιακού (ejb) στρώματος αποτελεί το διάγραμμα οντοτήτων- συσχετίσεων

6.1.2.1.1 Σύνολα οντοτήτων

Τα **σύνολα οντοτήτων** που χρησιμοποίησα είναι τα ακόλουθα:

Οι Σπουδαστές (σύνολο οντοτήτων **Student**) : σε αυτό το σύνολο οντοτήτων γίνεται η εγγραφή του σπουδαστή στο εξάμηνο, όπου δηλώνονται , ο αριθμός μητρώου του σπουδαστή , τα προσωπικά στοιχεία (όνομα, διεύθυνση κ.τ.λ) , το εξάμηνο, το τηλέφωνο, το όνομα χρήστη που έχει στο σύστημα και τα μαθήματα που δηλώνει. Μπορεί να δηλώσει, από κανένα(0) έως και τρία(3) και πολλούς τηλεφωνικούς αριθμούς.

Τα Μαθήματα (σύνολο οντοτήτων **Course**) : σε αυτό το σύνολο οντοτήτων δηλώνονται , ο κωδικός αριθμός του μαθήματος, το όνομα, οι εβδομαδιαίες ώρες διδασκαλίας του και το αν έχει εργαστήριο.

Οι Καθημερινές Ώρες Διδασκαλίας (σύνολο οντοτήτων **DailyTeachingTime**) : εδώ έχουμε τις επιτρεπόμενες ώρες, καθημερινά , που μπορούν να διεξαχθούν τα μαθήματα στη

σχολή. Δηλώνεται ο κωδικός αριθμός και οι ώρα έναρξης - λήξης (π.χ. 10:00-12:00) των ωρών διδασκαλίας.

Οι Εβδομαδιαίες Μέρες Διδασκαλίας (σύνολο οντοτήτων WeeklyTeachingDay) : εδώ έχουμε τις επιτρεπόμενες μέρες, εβδομαδιαία, που μπορούν να διεξαχθούν τα μαθήματα στη σχολή. Δηλώνεται ο κωδικός αριθμός και το όνομα της μέρας.

Οι Εργαζόμενοι (σύνολο οντοτήτων Employee) : σε αυτό το σύνολο οντοτήτων δηλώνονται , ο αριθμός μητρώου του εργαζομένου, τα προσωπικά στοιχεία(όνομα, διεύθυνση κ.τ.λ) , ο μισθός , η ειδικότητα και το τηλέφωνο. Ένας εργαζόμενος μπορεί να δηλώσει πολλούς τηλεφωνικούς αριθμούς.

- **Οι Καθηγητές** (σύνολο οντοτήτων Professor) : σε αυτό το σύνολο οντοτήτων δηλώνονται , ο τομέας στον οποίο ανήκει ο καθηγητής, ο αριθμός μητρώου του διευθυντή του τομέα και όλες οι άλλες ιδιότητες κληρονομούνται από το σύνολο οντοτήτων Εργαζόμενοι(Employee)και δηλώνονται μέσω αυτού.

- **Οι Τεχνικοί** (σύνολο οντοτήτων Technician) : σε αυτό το σύνολο οντοτήτων δηλώνονται , η βαθμίδα του τεχνικού και όλες οι άλλες ιδιότητες κληρονομούνται από το σύνολο οντοτήτων Εργαζόμενοι(Employee)και δηλώνονται μέσω αυτού.

Οι Αίθουσες Διδασκαλίας (σύνολο οντοτήτων ClassRoom) : εδώ έχουμε τις επιτρεπόμενες αίθουσες διδασκαλίας που καθημερινά, μπορούν να διεξαχθούν τα μαθήματα στη σχολή. Δηλώνεται ο κωδικός αριθμός, η ονομασία και η χωρητικότητα της αίθουσας.

Οι Πραγματοποιηθέντες Έλεγχοι (σύνολο οντοτήτων Check_C) : που διενεργούνται από τους Τεχνικούς (Technician)της σχολής σε συγκεκριμένες αίθουσες(ClassRoom). Λόγω της εμφανούς εξάρτησης από άλλα ισχυρά σύνολα οντοτήτων (Technician, ClassRoom), οι πραγματοποιηθέντες έλεγχοι μοντελοποιούνται ως αδύναμο σύνολο οντοτήτων. Εκτός από τα γνωρίσματα που αφορούν την ημερομηνία διεξαγωγής του ελέγχου, την διάρκεια εκτέλεσης , χρειάζεται να προσδιοριστεί και ένα διακριτικό γνώρισμα. Αν γινόταν η παραδοχή ότι κάποια αίθουσα δεν μπορεί να ελεγχθεί απ΄ τον ίδιο τεχνικό μέσα στην ίδια ημέρα, τότε η ημερομηνία θα μπορούσε να εξυπηρετήσει ως μερικό κλειδί. Ωστόσο, εδώ υιοθετείται η λογική υπόθεση ότι κάτι τέτοιο δεν πρέπει να αποκλειστεί, οπότε εισάγεται ένα

πρόσθετο γνώρισμα (ch_id) ως μερικό κλειδί (λ.χ. ο αύξων αριθμός του δελτίου ελέγχου του βιβλιαρίου ελέγχου του τεχνικού).

Τα σύνολα **συσχετίσεων** που χρησιμοποίησα είναι τα ακόλουθα:

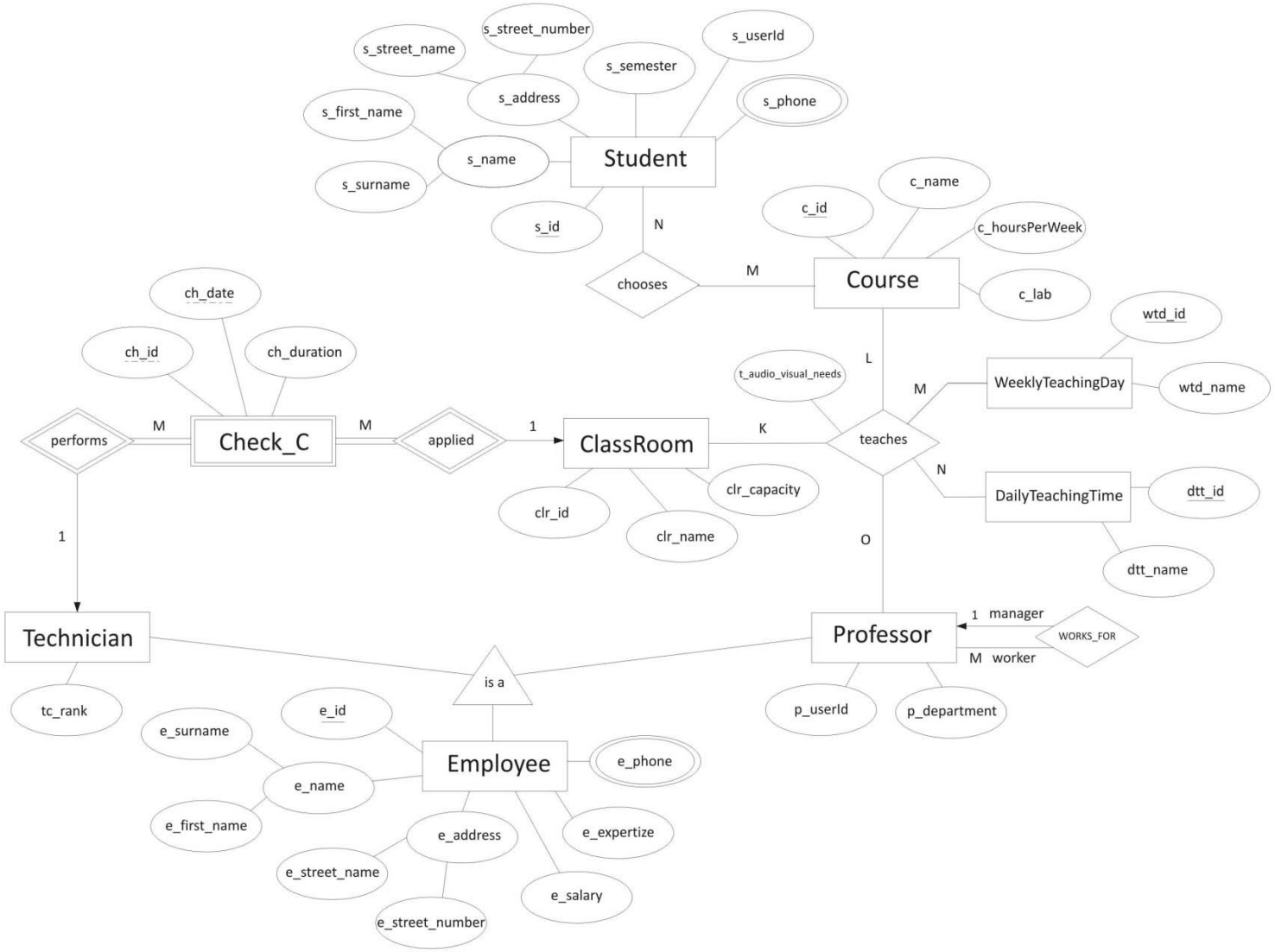
Η συσχέτιση **Επιλέγει (chooses)** διασυνδέει τους Σπουδαστές με τα αντίστοιχα Μαθήματα με βαθμό πληθικότητας N : M χωρίς να απαιτείται ολική συμμετοχή από καμία πλευρά της σχέσης , αφού μπορεί κάποιος σπουδαστής να μην επιλέξει κανένα μάθημα για το εξάμηνο εγράφης(π.χ. δεν χρωστάει μαθήματα αλλά του μένει ή πτυχιακή εργασία) αλλά και ένα μάθημα μπορεί να μη δηλωθεί από κανένα σπουδαστή .

Η συσχέτιση **Διδάσκει (teaches-** εδώ ορίζονται οι διδασκαλίες) , διασυνδέει τους Καθηγητές() με τα αντίστοιχα Μαθήματα, τις **Εβδομαδιαίες Μέρες Διδασκαλίας,** *τις* **Καθημερινές Ώρες Διδασκαλίας** και τις **Αίθουσες Διδασκαλίας** με βαθμό πληθικότητας πολλά προς πολλά (N : M) χωρίς να απαιτείται ολική συμμετοχή από καμία πλευρά της σχέσης. Επιπλέον στη συσχέτιση ορίζεται μια επιπλέον ιδιότητα για το αν χρειάζεται οπτικοακουστικός εξοπλισμός.

Οι δύο τελευταίες συσχετίσεις **Εκτελεί** (**performs**) και Εφαρμόζεται(applied) αφορούν την αδύναμη οντότητα **Πραγματοποιηθέντες Έλεγχοι** (**Check_C**) και αποτελούν τις *προσδιοριστικές* της συσχετίσεις. Πράγματι, κάθε πραγματοποιούμενος έλεγχος προσδιορίζεται βάσει του αντίστοιχου κανόνα της FAA, εφαρμόζεται σε μια συγκεκριμένη Αίθουσα και τον Εκτελεί ένας ορισμένος τεχνικός, γι' αυτό οι **Πραγματοποιηθέντες Έλεγχοι** έχουν *ολική* συμμετοχή και στις δύο συσχετίσεις. Επίσης, εφόσον ο ίδιος(1) Τεχνικός γενικά πραγματοποιεί πολλούς(M) ελέγχους και η κάθε(1) Αίθουσα είναι δυνατόν να υποστεί πολλούς ελέγχους. Καθεμιά από τις δύο συσχετίσεις έχει βαθμό πληθικότητας 1 : M.

6.1.2.1.2 Διάγραμμα οντοτήτων – συσχετίσεων

Στο παρακάτω διάγραμμα (Σχήμα 6.1) αποτυπώνονται οι, οντότητες (ιδιότητες και τύποι αυτών) και οι σχέσεις (συσχετίσεις και τύποι αυτών) για τη δημιουργία της επιθυμητής βάσης δεδομένων της εφαρμογής .



Σχήμα 6.2 Διάγραμμα οντοτήτων- συσχετίσεων

6.1.2.2 Υλοποίηση

Η ενότητα αυτή έχει την έννοια να περιγράψει το πώς υλοποιήθηκε τελικά η βάση δεδομένων στο φυσικό επίπεδο, σε σχέση με τις αρχικές κλάσεις της σχεδίασης.
Η υλοποίηση έγινε υπό το πνεύμα, περισσότερο της παρουσίασης όσο το δυνατόν περισσότερων διαφορετικών χαρακτηριστικών και δυνατοτήτων της προδιαγραφής EJB 3 και λιγότερο με γνώμονα την απόδοση και την όσο δυνατή συμπίεση (όχι επαναλαμβανόμενη πληροφορία) της βάσης π.χ. έχουν υλοποιηθεί δύο σχέσεις για την οντότητα Τηλέφωνο, μια για τον Σπουδαστή και μία για τον Εργαζόμενο (Phone_Std και Phone_Emp). Στις παρακάτω ενότητες γίνεται ανάλυση της υλοποίησης. Παρουσιάζονται λεπτομερώς οι σχέσεις που υλοποιήθηκαν σε αντιστοιχία με τα σύνολα οντοτήτων και των συσχετίσεων αυτών, από το διάγραμμα οντοτήτων – συσχετίσεων της σχεδίασης. Απόρροια της υλοποίησης, αποτελεί το σχεσιακό διάγραμμα της βάσης δεδομένων όπου δίνεται μια εικόνα των σχέσεων και των συσχετίσεων όπως αυτές τελικά διαμορφώθηκαν, μετά το πέρας της υλοποίησης.

6.1.2.2.1 Σχέσεις

Οι Σχέσεις που υλοποιήθηκαν, κατά αντιστοιχία με τα παραπάνω σύνολα οντοτήτων της σχεδίασης είναι οι ακόλουθες:
Α) Για τα σύνολα οντοτήτων:

Τα κλειδιά σημειώνονται με έντονα στοιχεία (bold). Αρχικά, κάθε ισχυρό σύνολο ***οντοτήτων*** μετατρέπεται απευθείας σε σχέση

Η **Student (s_id**,s_semester, s_userID**)** τα σύνθετα γνωρίσματα s_name και s_address έχουν υλοποιηθεί σε ξεχωριστούς πίνακες **std_name** και **std_address** αντίστοιχα. Επιπλέον και για το *πλειότιμο* γνώρισμα **s_phone** δημιουργείται χωριστός πίνακας.

- Η **Std_Name (student_id**, s_surname, s_first_name**)**
- Η **Std_Address (student_id**, s_street_name, s_street_number**)**
- Η **Phone_Std (ph_id**, ph_numb, STUDENT_ID**) –** Μη δόκιμη υλοποίηση(χωρίς χρήση συλλογής).

Η **Course (c_id,** c_name, c_hoursPerWeek, c_lab**) .**

Η **DailyTeachingTime (dtt_id**, dtt_name**) .**

Η **WeeklyTeachingDay (wtd_id**, wtd_name**) .**

Η **Employee (e_id**, e_surname, e_first_name, e_street_name, e_street_number, e_salary, e_expertize**)**. Τα σύνθετα γνωρίσματα e_name και e_address έχουν αντικατασταθεί με τα συστατικά τους. Εν τούτοις, για το πλειότιμο γνώρισμα e_phone δημιουργείται χωριστός πίνακας.

- Η **Phone_Emp (ph_id**, ph_numb**) –** Δόκιμη υλοποίηση (με χρήση συλλογής).

Εξάλλου, οι εξειδικεύσεις διατηρούν το κλειδί της υπερκείμενης κλάσης, μαζί με τα τυχόν πρόσθετα δικά τους γνωρίσματα. Εφόσον η εξειδίκευση είναι μερική, η σχέση *EMPLOYEES* παραμένει και **δεν** υποκαθίσταται από τις υποκλάσεις της:

- Η **Professor (e_id**, p_department, WORKS_FOR, p_userID**)**
- Η **Technician (e_id**, tc_rank**)**
- Η **ClassRoom (clr_id**, clr_name, clr_capacity**)**

Τέλος, η αδύναμη οντότητα πραγματοποιηθέντες έλεγχοι **Check_C** , χρειάζεται το ένα κλειδί (performs)από τη μία απ' τις δύο ισχυρές οντότητες που την προσδιορίζουν προκειμένου να σχηματίσει, μαζί με το δικό της μερικό κλειδί σύνθετο κλειδί , το *σύνθετο κλειδί* της τελικής σχέσης:

Η **Check_C (ch_id**, **ch_date**, **performs**, applied, ch_duration**) .**

Β) Για τα σύνολα συσχετίσεων:

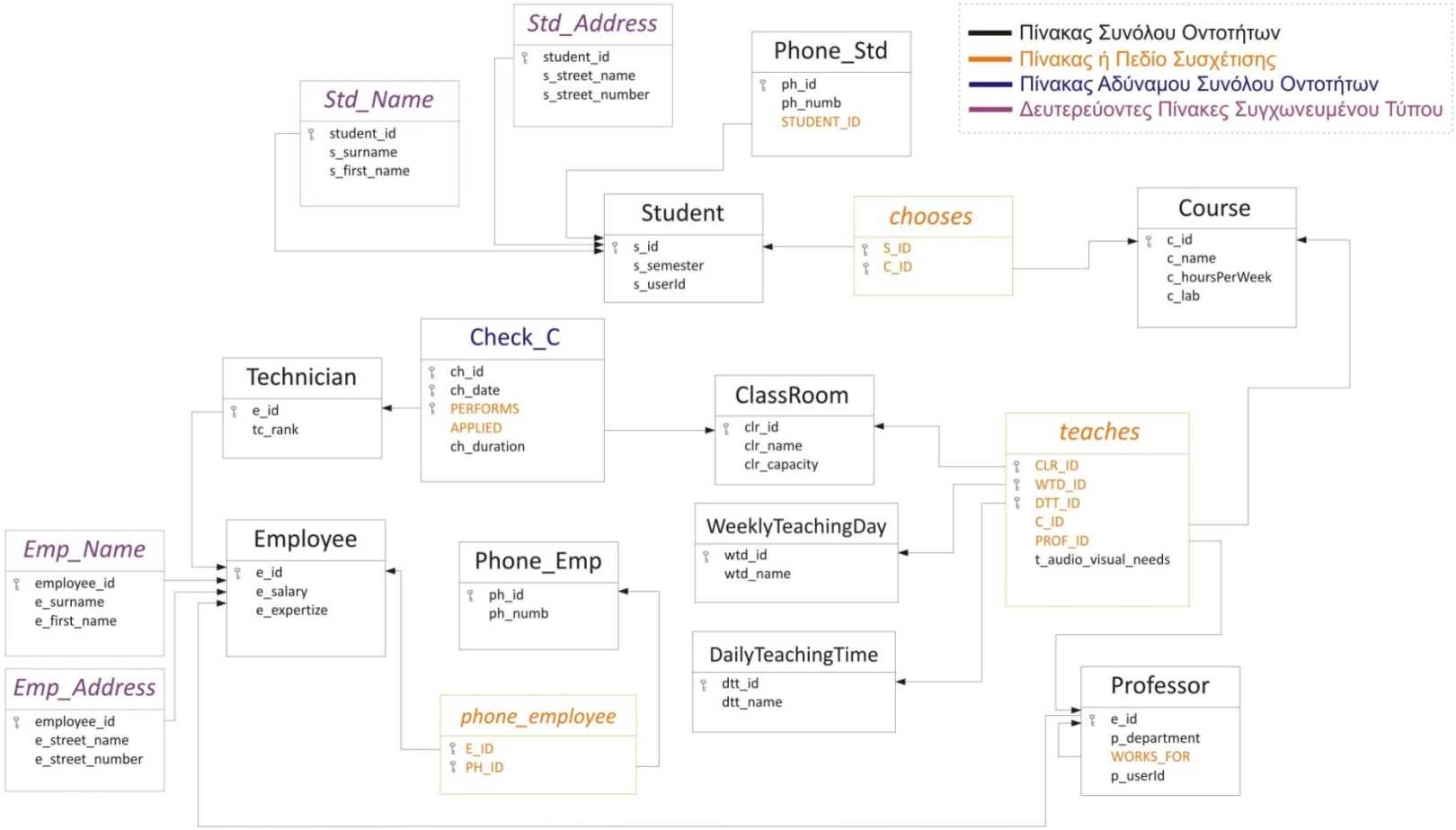
Η **chooses (S_ID**, **C_ID) -** υλοποιείται σαν πίνακας διασύνδεσης (join table)

Η **teaches (CLR_ID**, **WTD_ID**, **DTT_ID**, C_ID, PROF_ID, t_audio_visual_needs**)-** υλοποιείται σαν σύνολο οντοτήτων που αποτελείται από στήλες διασύνδεσης και όχι σαν πίνακας διασύνδεσης(join table)

Η **phone_employee (E_ID**, **PH_ID)** – υλοποιείται σαν πίνακας διασύνδεσης(join table)(δεν φαίνεται στο διάγραμμα οντοτήτων συσχετίσεων)

Οι συσχετίσεις **STUDENT_ID**, **WORKS_FOR**, **PERFORMS**, **APPLIED** υλοποιούνται απλά ως διασυνδέσεις στήλης, χρησιμοποιώντας ως διασύνδεση την ίδια τη στήλη

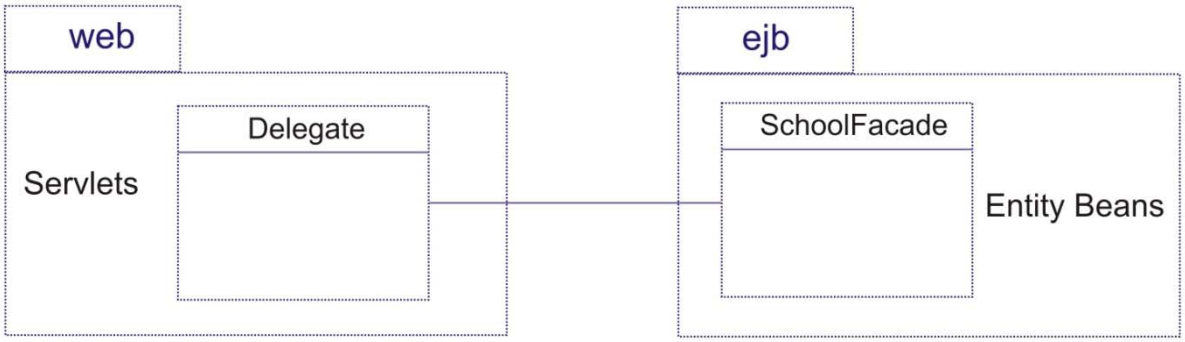
6.1.2.1.2 Διάγραμμα σχεσιακού σχήματος



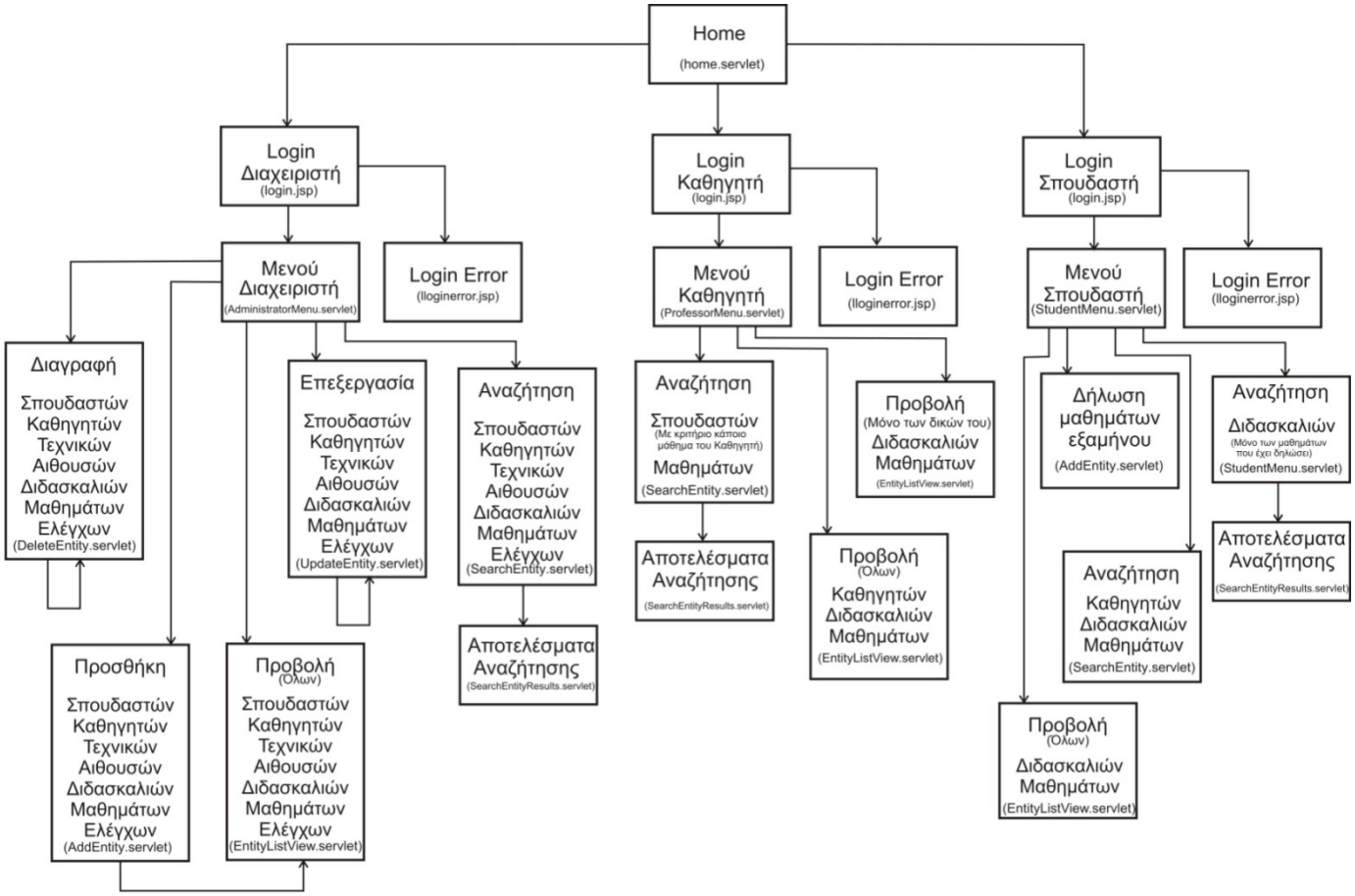
Σχήμα 6.3 Διάγραμμα σχεσιακού σχήματος

6.1.3 Σχεδίαση και υλοποίηση του στρώματος ιστού (web)

Η σχεδίαση του στρώματος ιστού (web) έγινε με δενδροειδή τρόπο, όπως φαίνεται στο παρακάτω διάγραμμα πλοήγησης (Σχήμα 6.4), και υλοποιήθηκε από κλάσεις servlet. Για τη διασύνδεση του στρώματος αυτού με το επιχειρησιακό στρώμα (ejb) χρησιμοποιήθηκε μια κλάση (Delegate) που παίζει το ρόλο του «εκπροσώπου» του στρώματος ιστού με τον "έξω κόσμο" (Σχήμα 6.3). Η κλάση αυτή υλοποιεί όλες τις επιχειρησιακές μεθόδους των servlet , καλώντας τις αντίστοιχες μεθόδους της διεπαφής του επιχειρησιακού στρώματος (ejb).



Σχήμα 6.4 Διάγραμμα διασύνδεσης του επιχειρησιακού στρώματος (ejb) και του στρώματος ιστού (web) της εφαρμογής

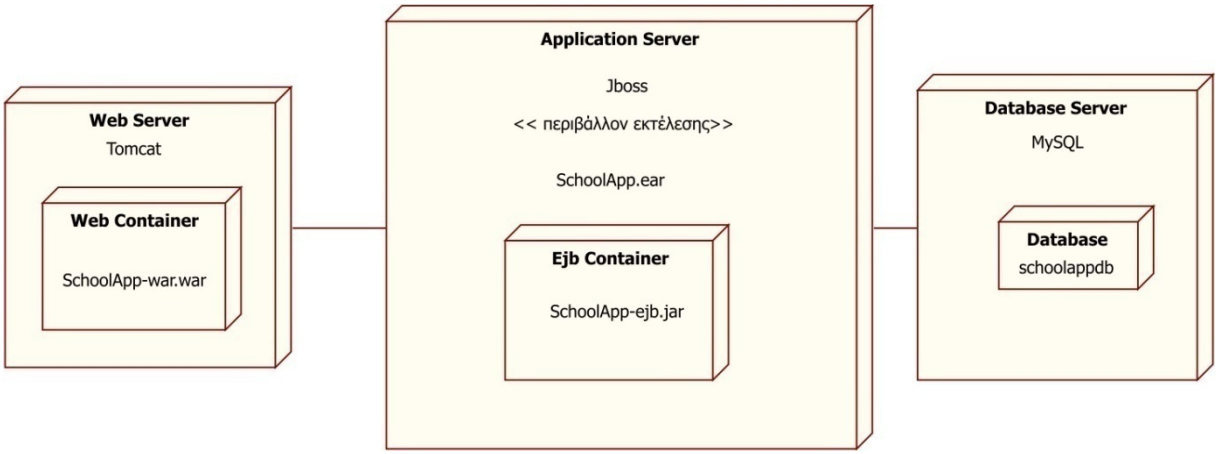


Σχήμα 6.5 Διάγραμμα πλοήγησης της εφαρμογής (γραφηματική διαπροσωπεία του συστήματος)

6.2 Διαγράμματα που περιγράφουν την υλοποίηση και την λειτουργία της εφαρμογής

6.2.1 Παραταξιακό διάγραμμα

Στο παρακάτω παραταξιακό διάγραμμα (Σχήμα 6.5), παρουσιάζονται οι εξυπηρετητές των διαφόρων στρωμάτων της εφαρμογής, ιστού, επιχειρησιακού και πληροφοριακού συστήματος με τα αντίστοιχα αρχεία που εκτελεί ο καθένας από αυτούς.



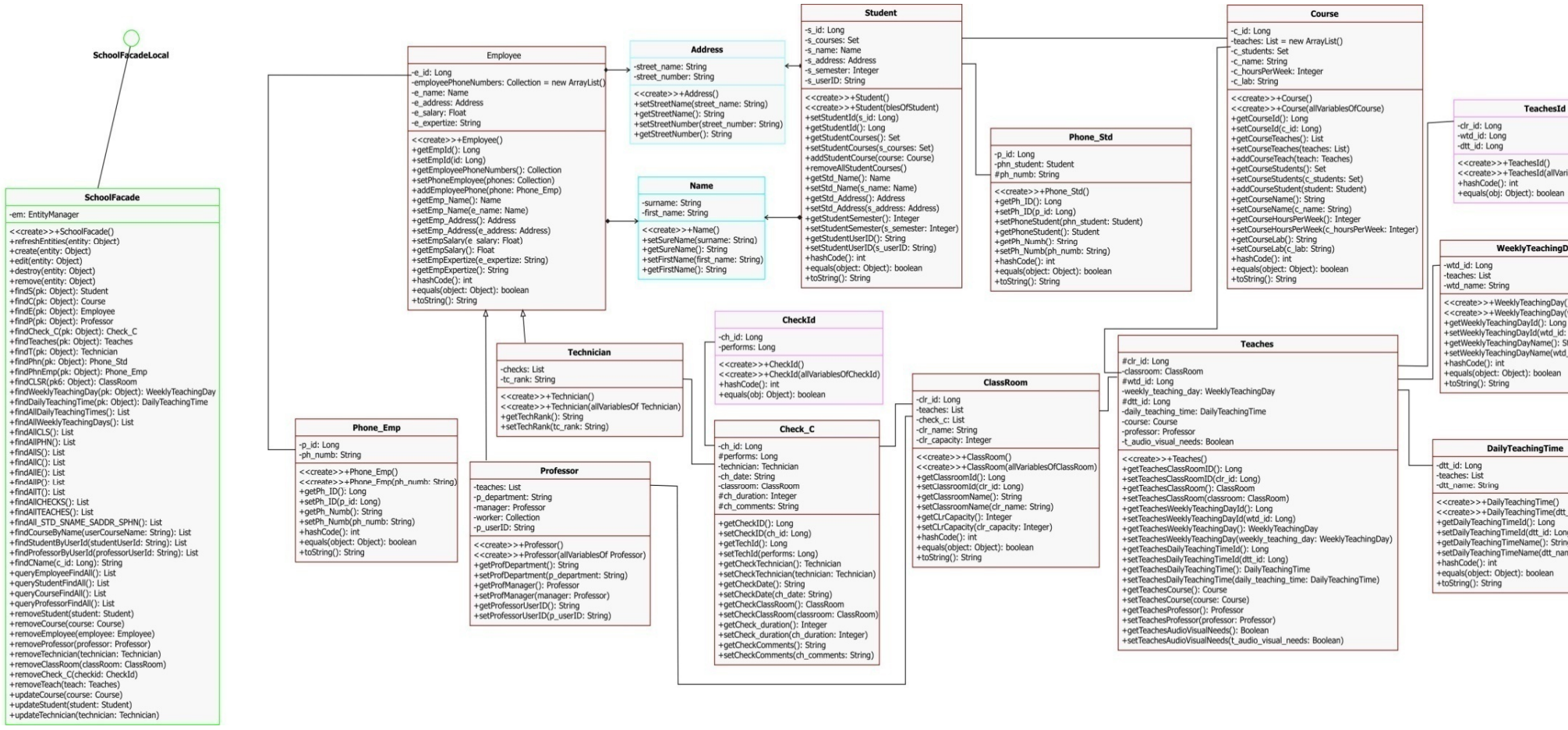
Σχήμα 6.6 Παραταξιακό διάγραμμα

6.2.2 Αναλυτικό διάγραμμα κλάσεων

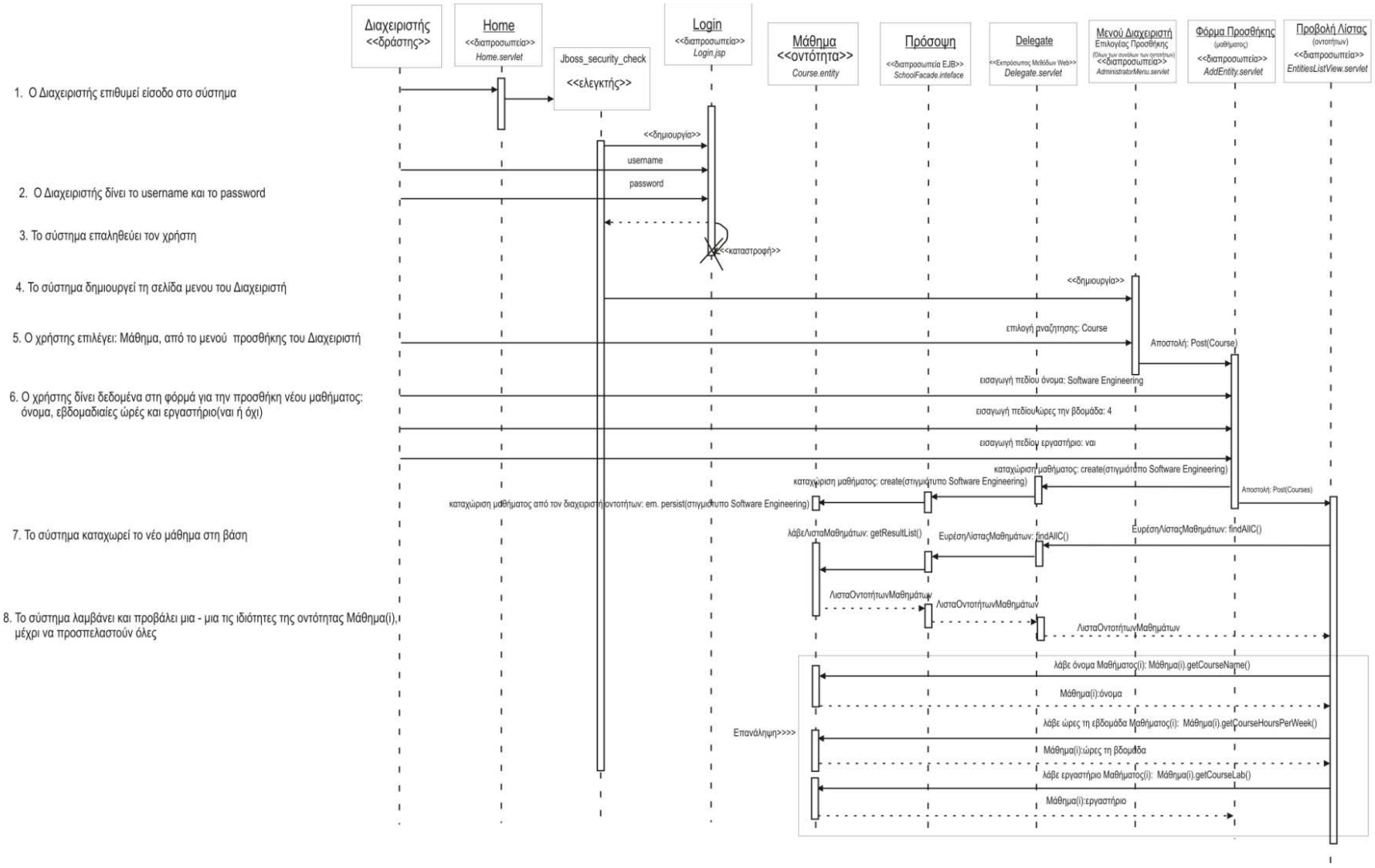
Παρουσιάζεται παρακάτω το αναλυτικό διάγραμμα κλάσεων (Σχήμα 6.5), όπως διαμορφώθηκε μετά από την τελική υλοποίηση του επιχειρησιακού στρώματος της εφαρμογής, όπου παρουσιάζονται αναλυτικά οι σχέσεις , οι ιδιότητες και οι μέθοδοι κάθε κλάσης που χρησιμοποιήθηκε.

6.2.3 Ακολουθιακά διαγράμματα

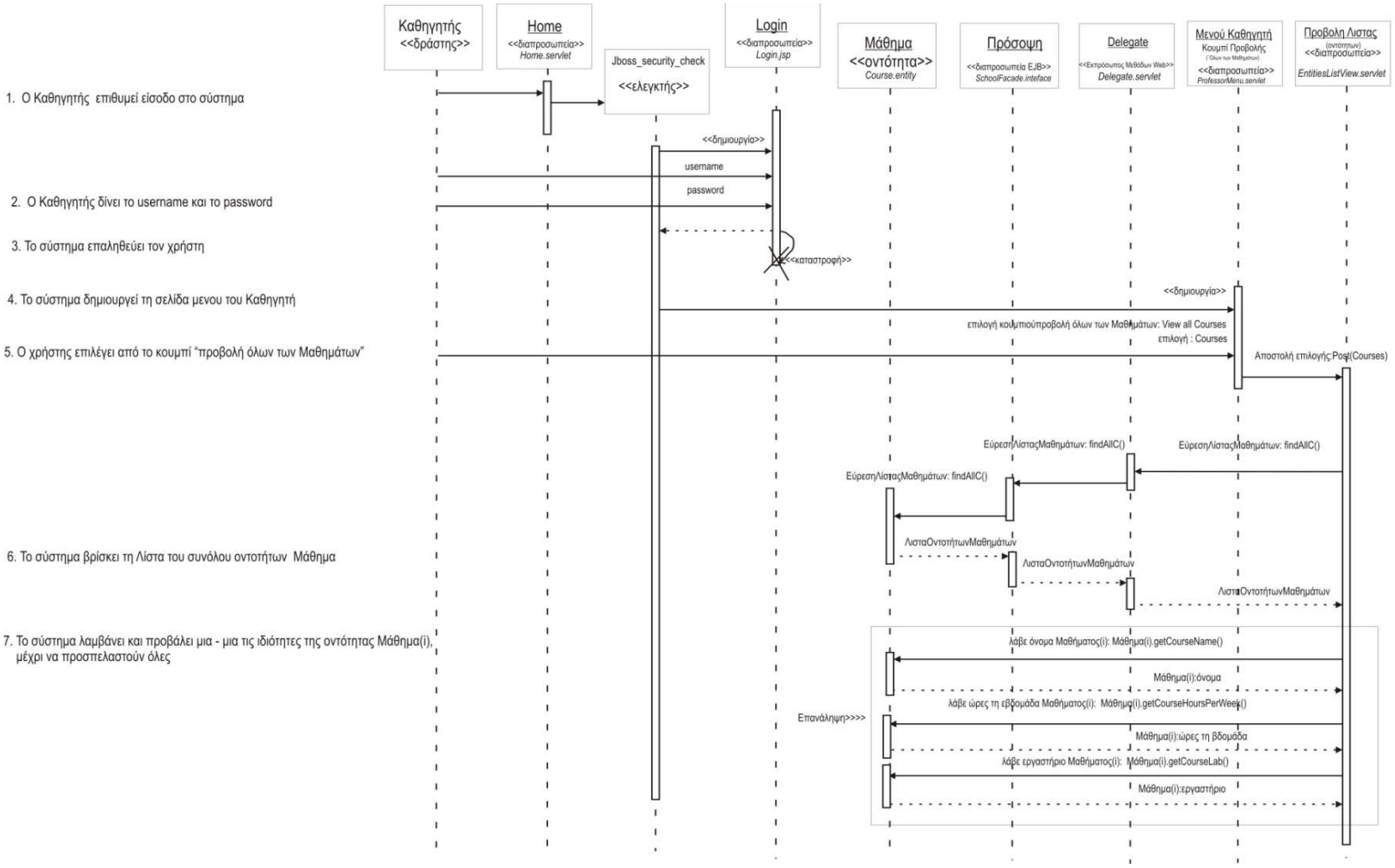
Παρουσιάζονται τα ακολουθιακά διαγράμματα χρήσης (Σχήμα 6.7, 6.8 και 6.9) για τις τρεις κατηγορίες χρηστών : Διαχειριστή, Καθηγητή και Σπουδαστή. Για τον καθένα από αυτούς χρησιμοποιείται ένα ξεχωριστό σενάριο χρήσης της εφαρμογής, που περιγράφει τη λειτουργία της εφαρμογής μέσα από την κλήση των μεθόδων των κλάσεων του στρώματος ιστού (web) και επιχειρησιακού (ejb), όπου αναδεικνύεται και ο τρόπος διασύνδεσης τους.



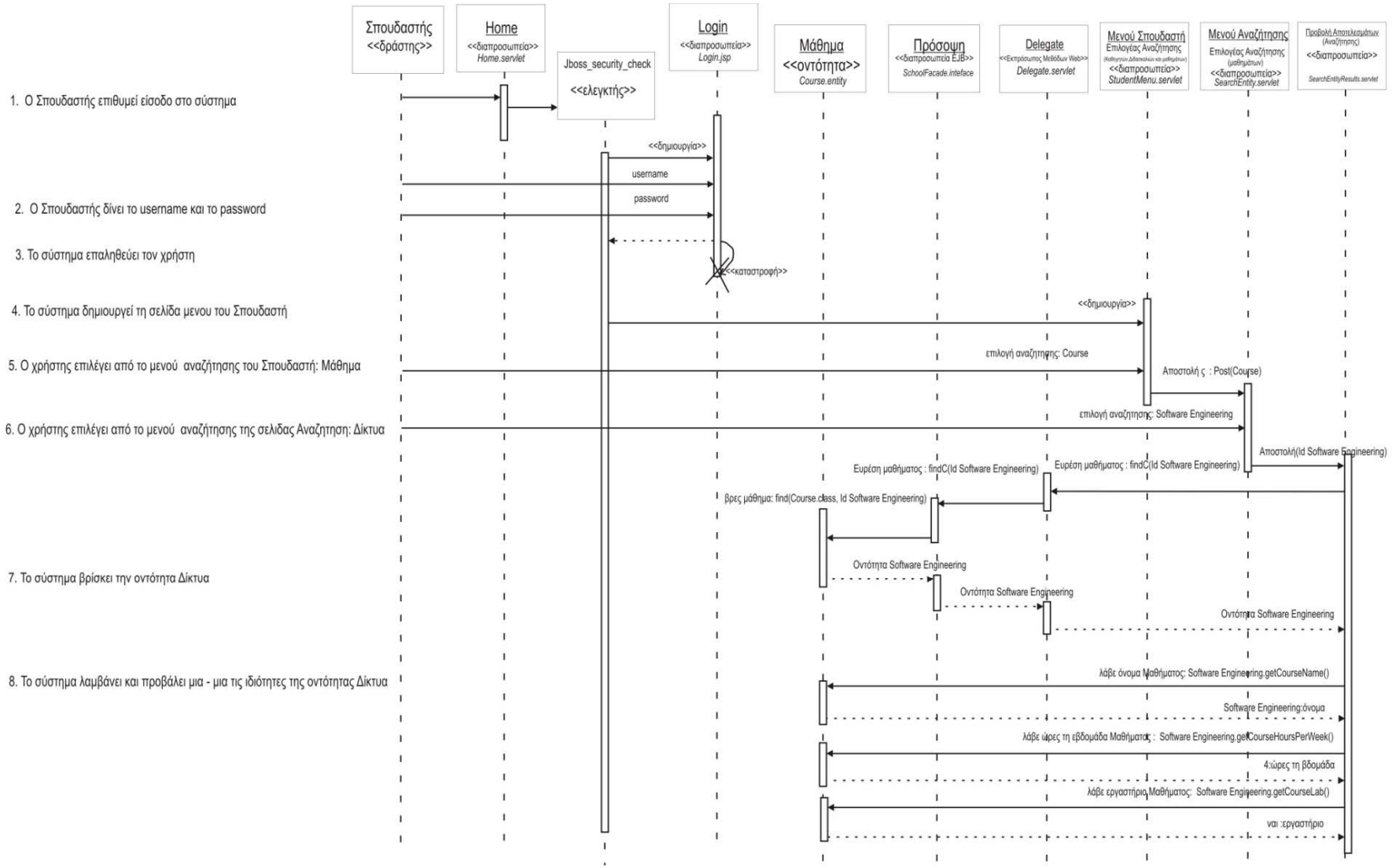
Σχήμα 6.7 Αναλυτικό διάγραμμα κλάσεων



Σχήμα 6.8 Ακολουθιακό διάγραμμα χρήσης Διαχειριστή. Σενάριο προσθήκη μαθήματος



Σχήμα 6.9 Ακολουθιακό διάγραμμα χρήσης Καθηγητή. Σενάριο προβολή όλων των μαθημάτων



Σχήμα 6.10 Ακολουθιακό διάγραμμα χρήσης Σπουδαστή. Σενάριο εύρεση μαθήματος

7

Μια περιγραφή της υλοποίησης

Εδώ παρουσιάζονται οι διάφορες πλατφόρμες που χρησιμοποιήσαμε και ο τρόπος σύνδεσης μεταξύ τους.

7.1 Εγκαταστάσεις εφαρμογών

1. **J2EE** : Πλατφόρμα (JAVA 2) Ανάπτυξης Εφαρμογών Ιστού(υποστηρίζει: server, HTML, Enterprise JavaBean (EJB) , Java Database Connectivity (JDBC), Messaging (Java Message Service), Java servlet *κτλ.)*

   (Αρχείο εγκατάστασης : jdk-1_5_0_03-windows-i586-p)

2. **NETBEANS – JBOSS – APACHE TOMCAT** :

   α) Netbeans 5.5 : Περιβάλλον Ανάπτυξής Εφαρμογών και Διαχείρισης Servers(Web Server, Application Server, Database Server κτλ.)

   β) Jboss 4.0.4 : Application Server

γ) Apache Tomcat 5.5.17 : Web Server

(Αρχείο εγκατάστασης : jboss-4_0_4-nb-5_5-windows να γίνεται τοποθέτηση του Jboss στο C:\jboss-4.0.4.GA)

3. **MySQL** : Database Server

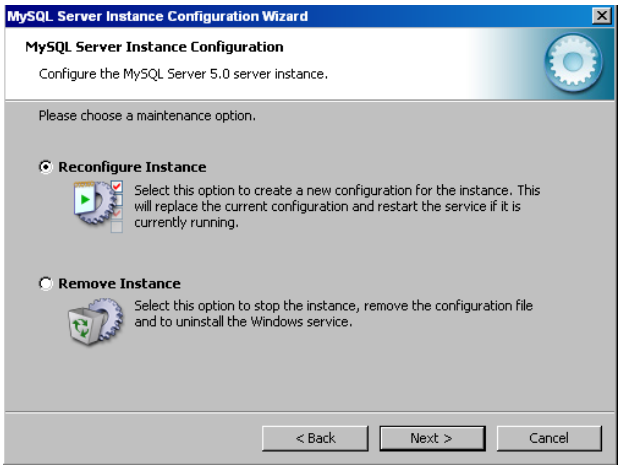
(Αρχείο εγκατάστασης : mysql-essential-5.0.41-win32)

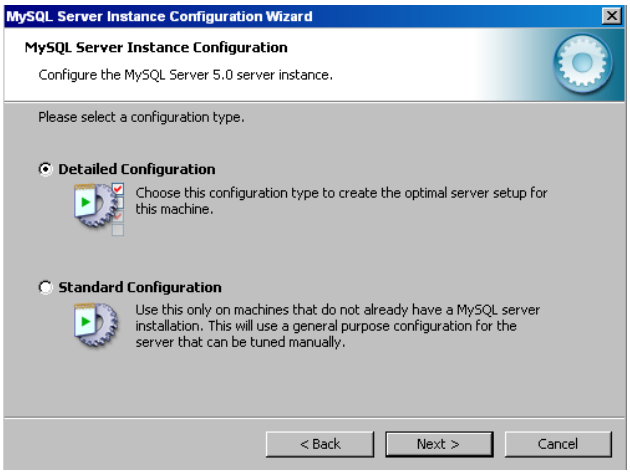
7.1.1 Ρυθμίσεις εγκατάστασης

7.1.1.1 Εξυπηρετητή βάσης δεδομένων MySQL

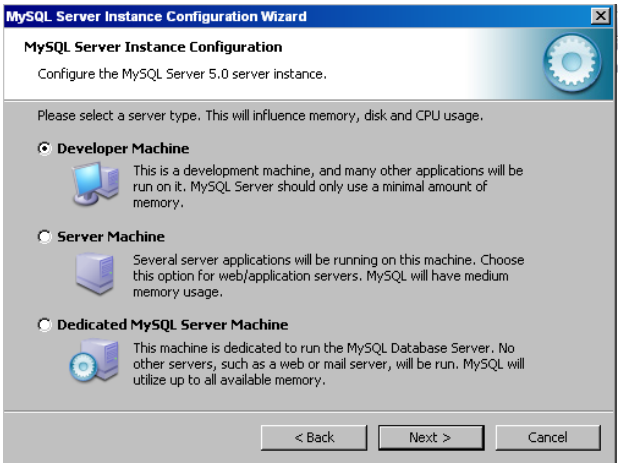
α) Ρυθμίσεις Εγκατάστασης :

1 2

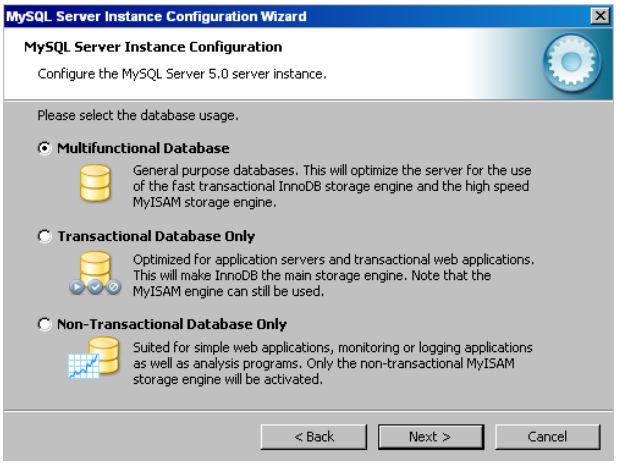




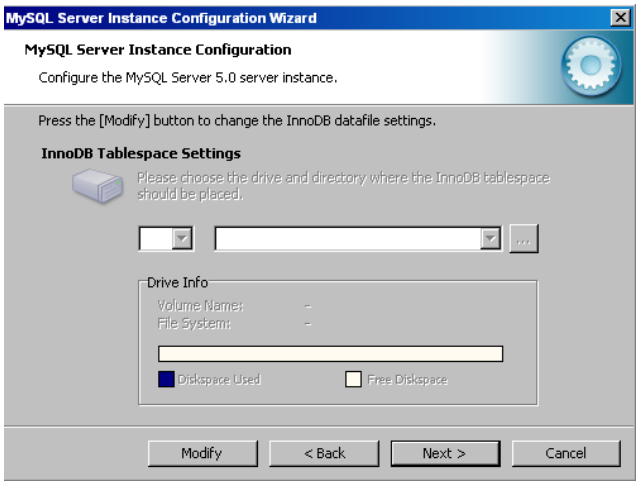
1 2


MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.0 server instance.
Please select a server type. This will influence memory, disk and CPU usage.
Developer Machine
This is a development machine, and many other applications will be run on it. MySQL Server should only use a minimal amount of memory.
Server Machine
Several server applications will be running on this machine. Choose this option for web/application servers. MySQL will have medium memory usage.
Dedicated MySQL Server Machine
This machine is dedicated to run the MySQL Database Server. No other servers, such as a web or mail server, will be run. MySQL will utilize up to all available memory.
< Back
Next >
Cancel

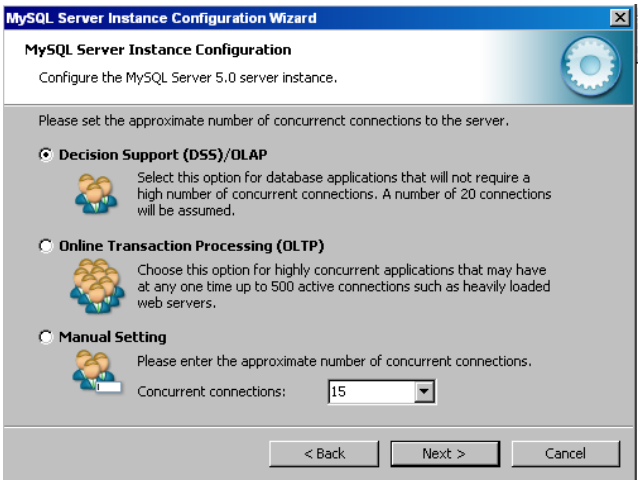
3


MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.0 server instance.
Please select the database usage.
Multifunctional Database
General purpose databases. This will optimize the server for the use of the fast transactional InnoDB storage engine and the high speed MyISAM storage engine.
Transactional Database Only
Optimized for application servers and transactional web applications. This will make InnoDB the main storage engine. Note that the MyISAM engine can still be used.
Non-Transactional Database Only
Suited for simple web applications, monitoring or logging applications as well as analysis programs. Only the non-transactional MyISAM storage engine will be activated.
< Back
Next >
Cancel

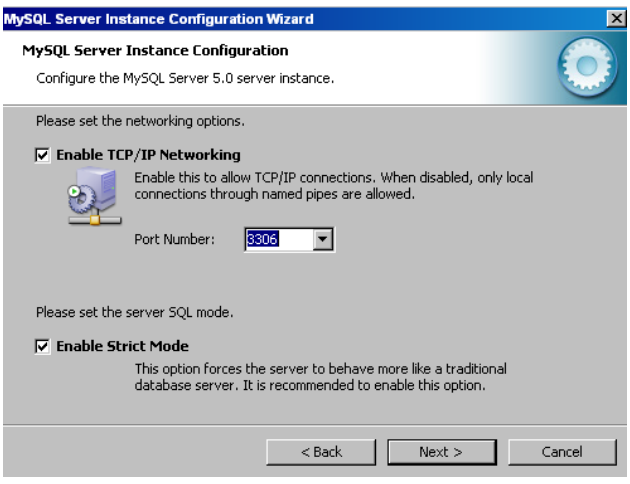
4


MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.0 server instance.
Press the [Modify] button to change the InnoDB datafile settings.
InnoDB Tablespace Settings
Please choose the drive and directory where the InnoDB tablespace should be placed.
Drive Info
Volume Name:
File System:
Diskspace Used
Free Diskspace
Modify
< Back
Next >
Cancel

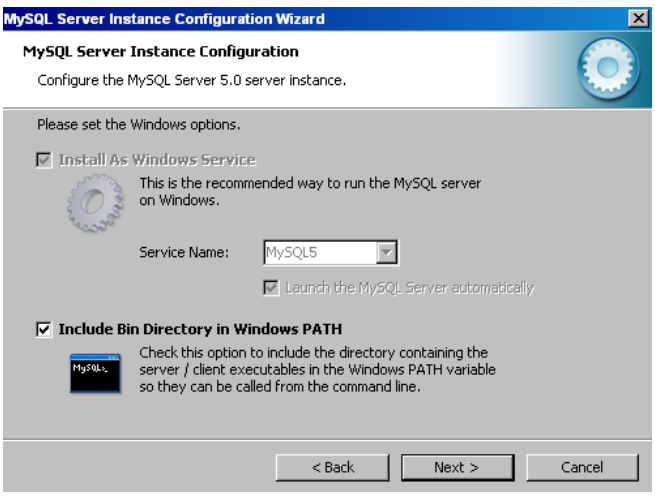
5


MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.0 server instance.
Please set the approximate number of concurrenct connections to the server.
Decision Support (DSS)/OLAP
Select this option for database applications that will not require a high number of concurrent connections. A number of 20 connections will be assumed.
Online Transaction Processing (OLTP)
Choose this option for highly concurrent applications that may have at any one time up to 500 active connections such as heavily loaded web servers.
Manual Setting
Please enter the approximate number of concurrent connections.
Concurrent connections:
15
< Back
Next >
Cancel

6

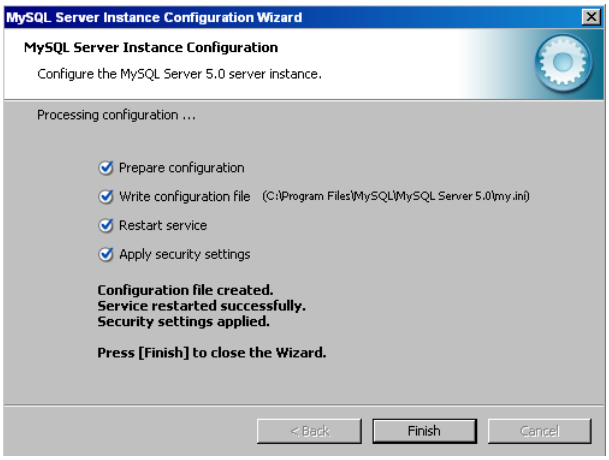

MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.0 server instance.
Please set the networking options.
Enable TCP/IP Networking
Enable this to allow TCP/IP connections. When disabled, only local connections through named pipes are allowed.
Port Number:
3306
Please set the server SQL mode.
Enable Strict Mode
This option forces the server to behave more like a traditional database server. It is recommended to enable this option.
< Back
Next >
Cancel

7


MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.0 server instance.
Please set the Windows options.
Install As Windows Service
This is the recommended way to run the MySQL server on Windows.
Service Name:
MySQL5
Launch the MySQL Server automatically
Include Bin Directory in Windows PATH
Check this option to include the directory containing the server / client executables in the Windows PATH variable so they can be called from the command line.
< Back
Next >
Cancel

8

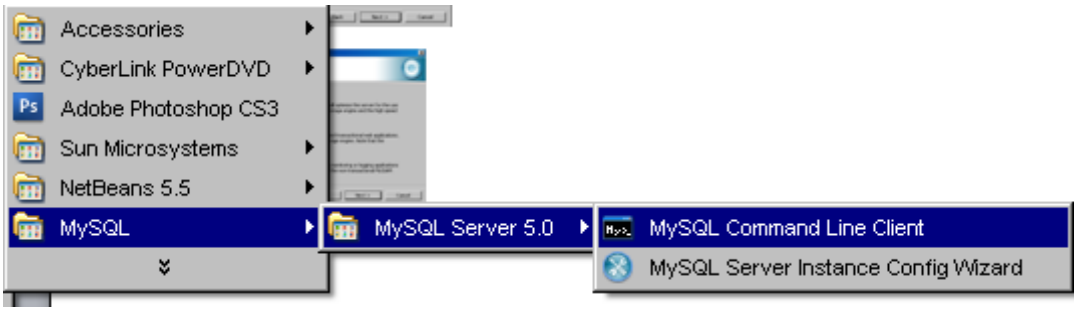
9



10

Σχήμα 7.1 Παρουσίαση εγκατάστασης του MySql server

β) Προετοιμασία της εφαρμογής στον MySQL server (δημιουργία της βάσης schoolappdb)



Σχήμα 7.2 Είσοδος στο περιβάλλον γραμμής εντολών του εξυπερετητή MySql

Enter password: root

mysql>CREATE database schoolappdb;

γ) Διασύνδεση του εξυπηρετητή βάσης δεδομένων (MySQL), με τον Mysql (Connector / J Driver) με τον εξυπηρετητή εφαρμογών (JBOSS) και το περιβάλλον Netbeans.

(Αρχείο εγκατάστασης : mysql-connector-java-5.1.6)

7.1.1.2 Εξυπηρετητή εφαρμογών Jboss

Επίσημες Οδηγίες Εγκατάστασης

Οι Επίσημες Οδηγίες Εγκατάστασης- εναλλακτικά για γρήγορη εγκατάσταση, χρήση των δικών μάς οδηγιών (με χρήση έτοιμων αρχείων) ακολουθούν μετά από αυτές.

Κατέβασμα του οδηγού (driver)

- Πρώτα πηγαίνουμε στο , http://www.mysql.com/products/connector/j/ και κατεβάζουμε την κατάλληλη έκδοση του mySQL server μας. Η δικιά μας είναι η : mysql-connector-java-5.1.6
- Έπειτα ,κάνουμε αποσυμπίεση του αρχείου mysql-connector-java-5.1.6.zip και εξάγουμε το jar αρχείο.

1. Εγκαθιστούμε τον οδηγό MySQL JDBC (mysql-connector-java-5.1.6-bin.jar) στον κατάλογο C:/jboss-4.0.4.GA /server/default/lib..

Ρυθμίσεις του πηγαίου αρχείου της βάσης δεδομένων

2. Διαγράφουμε το αρχείο **hsqldb-ds.xml** από τον κατάλογο $JBOSS_HOME/server/default/deploy/.

3.Αντιγράφουε το αρχείο **mysql-ds.xml** που βρίσκεται στον κατάλογο C:/jboss-4.0.4.GA /docs/examples/jca/ στον κατάλογο C:/jboss-4.0.4.GA /server/default/deploy. Τροποποιούμε το στοιχείο <local-tx-datasource> με τις δικές μας ρυθμίσεις σύνδεσης της βάσης MySQL, όπως φαίνονται παρακάτω:

```
<datasources>
<local-tx-datasource>
  <jndi-name>DefaultDS</jndi-name>
  <connection-url>jdbc:mysql://localhost:3306/schoolappdb</connection-url>
  <driver-class>com.mysql.jdbc.Driver</driver-class>
  <user-name>root</user-name>
  <password>root</password>
```

```
  <exception-sorter-class-
    name>org.jboss.resource.adapter.jdbc.vendor.MySQLExceptionSorter</exc
    eption-sorter-class-name>

<metadata>

  <type-mapping>mySQL</type-mapping>

    </metadata>

    </local-tx-datasource>

    </datasources>
```

4. Στο αρχείο **standardjaws.xml** που βρίσκεται στον κατάλογο $JBOSS_HOME/server/default/conf/, τροποποιούμε το στοιχείο <type-mapping> όπως φαίνεται παρακάτω:

```
  <type-mapping>mySQL</type-mapping>
```

5. Στον ίδιο κατάλογο (C:/jboss-4.0.4.GA /server/default/conf/) τροποποιούμε το αρχείο **standardjbosscmp-jdbc.xml** αλλάζοντας τα μη χρειαζούμενα στοιχεία, όπως φαίνεται παρακάτω:

```
  <datasource>java:/DefaultDS</datasource>
   <datasource-mapping>mySQL</datasource-mapping>
   <fk-constraint>true</fk-constraint>
```

6. Στον ίδιο κατάλογο (C:/jboss-4.0.4.GA /server/default/conf/) στο αρχείο **login-config.xml** προσθέτουμε εντός του στοιχείου <policy> τα ακόλουθα ρυθμίσεις για ασφάλεια (Λιστα 7.1):

Λιστα 7.1: Ρυθμίσεις για ασφάλεια

```
<policy>

    <application-policy name = "MySqlDbRealm?">
   <authentication>
      <login-module                          code                          =
"org.jboss.resource.security.ConfiguredIdentityLoginModule?"     flag     =
"required">
         <module-option name = "principal">root</module-option>
         <module-option name = "userName">root</module-option>
```

```
         <module-option name ="password">root</module-option>
         <module-option                          name                          =
"managedConnectionFactoryName">jboss.jca:service=LocalTxCM?,name=DefaultDS<
/module-option>
      </login-module>
   </authentication>
</application-policy>

    <application-policy name = "client-login">
       <authentication>
          <login-module code = "org.jboss.security.ClientLoginModule"
             flag = "required">
             <!-- Any existing security context will be restored on logout
-->
             <module-option     name="restore-login-identity">true</module-
option>
          </login-module>
       </authentication>
    </application-policy>

    <!-- Security domain for JBossMQ -->
    <application-policy name = "jbossmq">
       <authentication>
          <login-module                          code                          =
"org.jboss.security.auth.spi.DatabaseServerLoginModule"
             flag = "required">
             <module-option name = "unauthenticatedIdentity">guest</module-
option>
             <module-option  name  =  "dsJndiName">java:/DefaultDS</module-
option>
             <module-option  name  =  "principalsQuery">SELECT  PASSWD  FROM
JMS_USERS WHERE USERID=?</module-option>
             <module-option name = "rolesQuery">SELECT ROLEID, 'Roles' FROM
JMS_ROLES WHERE USERID=?</module-option>
          </login-module>
       </authentication>
    </application-policy>


    <application-policy name = "jmx-console">
       <authentication>
```

```
          <login-module
code="org.jboss.security.auth.spi.UsersRolesLoginModule"
             flag = "required">
           <module-option         name="usersProperties">props/jmx-console-
users.properties</module-option>
           <module-option         name="rolesProperties">props/jmx-console-
roles.properties</module-option>
          </login-module>
       </authentication>
    </application-policy>


    <application-policy name = "$webConsoleDomain">
       <authentication>
          <login-module
code="org.jboss.security.auth.spi.UsersRolesLoginModule"
             flag = "required">
             <module-option             name="usersProperties">web-console-
users.properties</module-option>
             <module-option             name="rolesProperties">web-console-
roles.properties</module-option>
          </login-module>
       </authentication>
    </application-policy>


    <application-policy name="JBossWS">
      <authentication>
        <login-module
code="org.jboss.security.auth.spi.UsersRolesLoginModule"
          flag="required">
          <module-option              name="usersProperties">props/jbossws-
users.properties</module-option>
          <module-option              name="rolesProperties">props/jbossws-
roles.properties</module-option>
          <module-option   name="unauthenticatedIdentity">anonymous</module-
option>
        </login-module>
      </authentication>
    </application-policy>
```

```
      <application-policy name = "other">

       <authentication>
          <login-module                          code                          =
"org.jboss.security.auth.spi.UsersRolesLoginModule"
             flag = "required" />
       </authentication>
    </application-policy>

</policy>
```

7. Αντικατάσταση του αρχείου **hsql-jdbc2-service.xml** που βρίσκεται στον κατάλογο C:/jboss-4.0.4.GA /server/default/deploy/jms/ με το αρχείο **mysql-jdbc2-service.xml** που βρίσκεται στον κατάλογο C:/jboss-4.0.4.GA/docs/examples/jms/ .

8. Αλλαγή από MySqlDS σε DefaultDS στο αρχείο mysql-jdbc2-service.xml που βρίσκεται στον κατάλογο C:/jboss-4.0.4.GA/server/default/deploy/jms/, όπως φαίνεται παρακάτω :

```
<mbean code="org.jboss.mq.pm.jdbc2.PersistenceManager"
  name="jboss.mq:service=PersistenceManager">
  <depends optional-attribute-
name="ConnectionManager">jboss.jca:service=DataSourceBinding,name=DefaultDS</depen
ds>
  <attribute name="SqlProperties?">
    . . .
```

9. Μετονομασία του αρχείου **hsqldb-jdbc-state-service.xml** που βρίσκεται στον κατάλογο C:/jboss-4.0.4.GA/server/default/deploy/jms/σε **mysql-jdbc-state-service.xml**.

Οι Δικές μας Οδηγίες Εγκατάστασης : βασίζονται στα αρχεία του φακέλου *Αρχεία για να τη σύνδεση του MySQL server με τον Jboss server* που έχω στείλει.

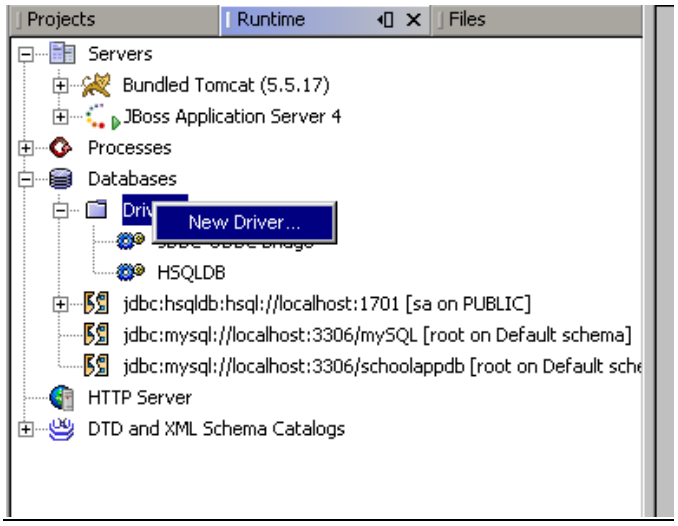
1. Αντιγραφή του αρχείου **mysql-connector-java-5.1.6-bin.jar** απ'το φάκελο …JBOSS Ρυθμίσεις\Αρχεία Εγκατάστασης\1 mysql-connector-java-5.1.6-bin COPY INTO C-jboss-4.0.4.GAserver-defaut-lib στο φάκελο C:\jboss-4.0.4.GA\server\default\lib

2. Αντιγραφή του **mysql-ds.xml** αρχείου απ'το φάκελο …\JBOSS Ρυθμίσεις\Αρχεία Εγκατάστασης\2 mysql-ds.xml COPY INTO C-jboss-4.0.4.GA-server-default-deploy στο φάκελο C:\jboss-4.0.4.GA\server\default\deploy και διαγραφή του **hsqldb-ds.xml**.

3. Αντιγραφή(και επικάλυψη) των **standardjaws.xml** , **standardjbosscmp-jdbc.xml** και **login-config.xml** αρχείων απ'το φάκελο ***…\JBOSS Ρυθμίσεις\Αρχεία Εγκατάστασης\4 standardjaws.xml standardjbosscmp-jdbc.xml login-config.xml COPY INTO C-jboss-4.0.4.GA-server-default-conf*** στο φάκελο C:\jboss-4.0.4.GA\server\default\conf

4. Αντιγραφή των mysql-jdbc2-service.xml και mysql-jdbc-state-service.xml αρχείων απ'το φάκελο ***…\JBOSS Ρυθμίσεις\Αρχεία Εγκατάστασης\7 mysql-jdbc2-service.xml mysql-jdbc-state-service COPY INTO C-jboss-4.0.4.GA-server-default-deploy-jms DELETE hsql-jdbc2-service.xml*** στο φάκελο C:\jboss-4.0.4.GA\server\default\deploy\jms και διαγραφή των hsql-jdbc2-service.xml και hsqldb-jdbc-state-service.xml αρχείων.

Ρυθμίσεις Εγκατάστασης του οδηγού και ορισμός της βάσης (schoolappdb) της εφαρμογής SchoolApp στο περιβάλλον Netbeans :

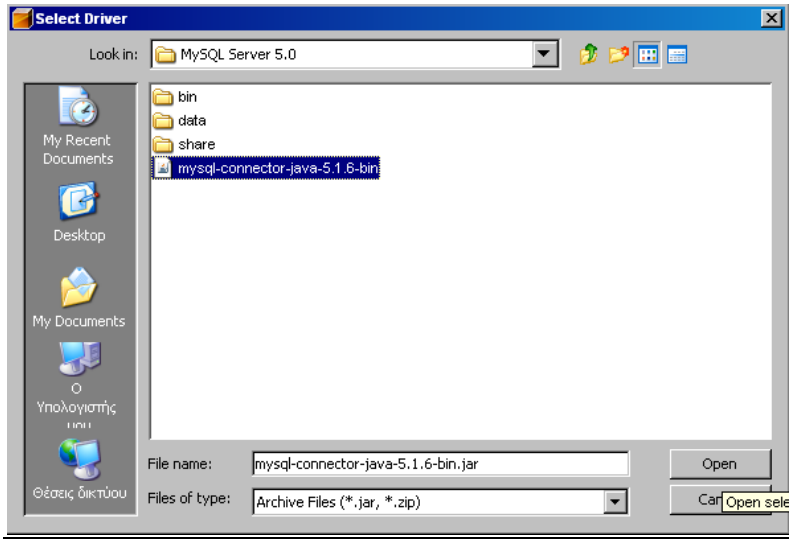


.

.

.

ν

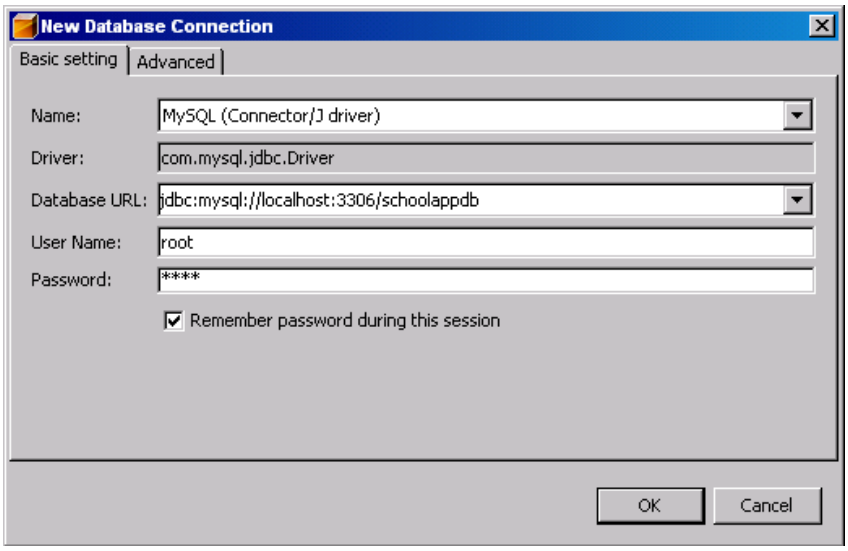


.

.

.

ν



Σχήμα 7.3 Ρυθμίσεις Εγκατάστασης του οδηγού και ορισμός της βάσης (schoolappdb) της εφαρμογής SchoolApp στο περιβάλλον Netbeans

Τα στοιχεία, που χρειάζονται, τα παίρνουμε από το αρχείο mysql-ds.xml που βρίσκεται στο φάκελο C:\jboss-4.0.4.GA\server\default\deploy, και φαίνονται παρακάτω.

mysql-ds.xml

```
<local-tx-datasource>
<jndi-name>DefaultDS</jndi-name>
<connection-url>jdbc:mysql://localhost:3306/schoolappdb</connection-url>
<driver-class>com.mysql.jdbc.Driver</driver-class>
<user-name>root</user-name>
<password>root</password>
```

7.2 Ο εξυπηρετητής εφαρμογών Jboss

Ο JBoss είναι ένας εξυπηρετητής εφαρμογών, ανοικτού κώδικα, που ακολουθεί την προδιαγραφή Java EE, με ενσωματωμένο τον εξυπηρετητή ιστού Apache Tomcat 5.5. Υποστηρίζεται από οποιαδήποτε Java Virtual Machine (JVM) μεταξύ των εκδόσεων 1.4 και 1.5. Ο JBoss εξυπηρετητής εφαρμογών μπορεί να εκτελείται αρκετά λειτουργικά συστήματα όπως Linux , Mac OS X, Microsoft Windows και σε άλλα που παρέχουν την κατάλληλη JVM. Ο JBoss είναι πιστοποιημένος ως εξολοκλήρου συμβατός εξυπηρετητής εφαρμογών με τις τελευταίες εκδόσεις της πλατφόρμας J2EE, οι οποίες παρέχουν μεγαλύτερη αξιοπιστία και πιο εύκολους τρόπους ολοκλήρωσης και υποστηρίζουν βελτιωμένες υπηρεσίες ιστού (web services) , ενημερωμένες δυνατότητες μηνυμάτων Java και ισχυρότερων επιλογών ασφάλειας.
Άλλοι εναλλακτικοί εξυπηρετητές εφαρμογών που ακολουθούν την προδιαγραφή Java EE, και μπορούν εναλλακτικά να εκτελέσουν την εφαρμογή που κατασκευάσαμε είναι οι : Sybase Enterprise Application Server (Sybase Inc), WebLogic Server (BEA), JBoss (Red Hat), WebSphere Application Server and WebSphere Application Server Community Edition (IBM), JRun (Adobe), Apache Geronimo (Apache Software Foundation), Oracle OC4J (Oracle Corporation), Sun Java System Application Server (Sun Microsystems), SAP Netweaver AS (SAP), and Glassfish Application Server (βασισμένος στον εξυπηρετητή εφαρμογών Sun Java System).

8

Παρουσίαση της εφαρμογής

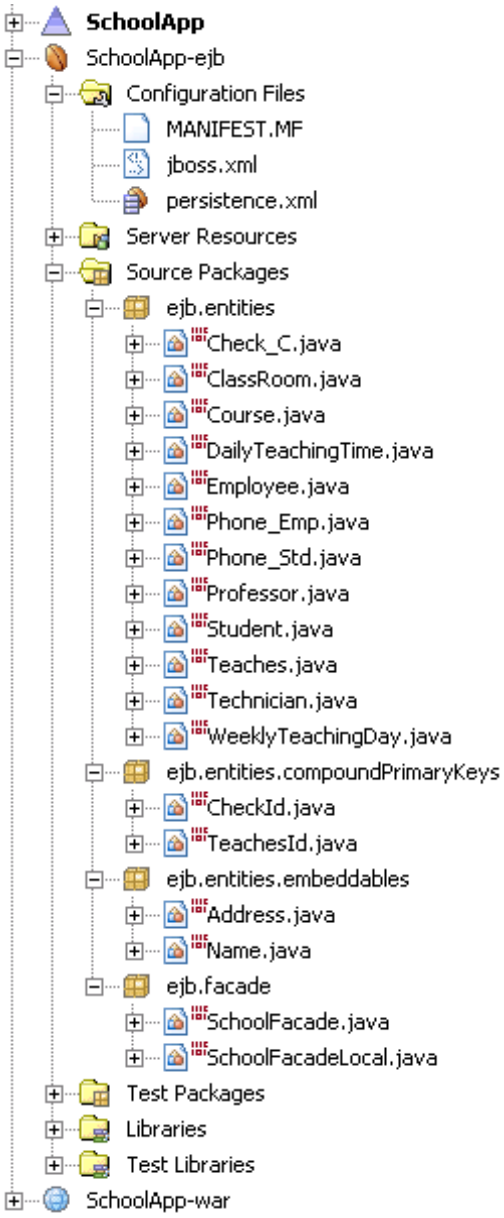
Στο κεφάλαιο αυτό γίνεται η παρουσίαση, με χρήση εικόνων, των δομικών στοιχείων της εφαρμογής και της χρήσης αυτής.

8.1 Περιεχόμενα αρχεία της εφαρμογής

Γενικά η εφαρμογή αποτελείται από δύο στρώματα :

α) το SchoolApp-ejb (επιχειρησιακό στρώμα), το οποίο περιλαμβάνει τα Entity Beans (συνόλα οντοτήτων-Entities κλάσεις) που είναι οι κλάσεις : Student, Phone_Std, Course, Teaches, Check_C, ClassRoom, DailyTeachingTime , WeeklyTeachingDay, Employee, Professor, Technician, Phone_Emp . Τις βοηθητικές κλάσεις των Entity Beans που είναι οι κλάσεις : Address, Name, TeachesId, CheckId. Όλες οι παραπάνω κλάσεις είναι αυτές που δημιουργούν τη βάση δεδομένων. Επίσης στο κομμάτι αυτό περιλαμβάνεται, το interface (SchoolFacadeLocal) και την υλοποίηση (SchoolFacadeLocal) του, όπου παρέχονται οι κατάλληλες μέθοδοι για την επικοινωνία της βάσης με τον έξω κόσμο.

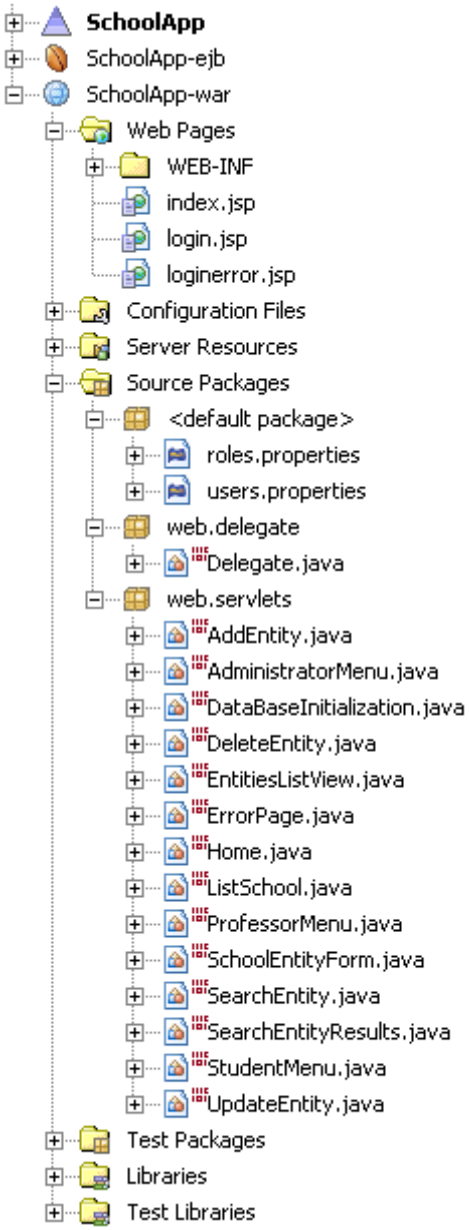
Τα αρχεία της εφαρμογής SchoolApp του επιχειρησιακού στρώματος, όπως φαίνονται από το περιβάλλον Netbeans:



Σχήμα 8.1 Τα αρχεία της εφαρμογής SchoolApp του επιχειρησιακού στρώματος

β) το SchoolApp-war (στρώμα ιστού), το οποίο περιλαμβάνει τα servlet, για την web αλληλεπίδραση του χρήστη με την εφαρμογή. Περιλαμβάνει δύο servlet i) το SchoolEntityForm, με το οποίο υποβάλλονται στη βάση, οι τιμές των πεδίων για κάθε πίνακα και ii) το ListSchool το οποίο παρέχει, τις προβολές (Views) των περιεχομένων των πινάκων της βάσης στο χρήστη.

Τα αρχεία της εφαρμογής SchoolApp του στρώματος ιστού, όπως φαίνονται από το περιβάλλον Netbeans:

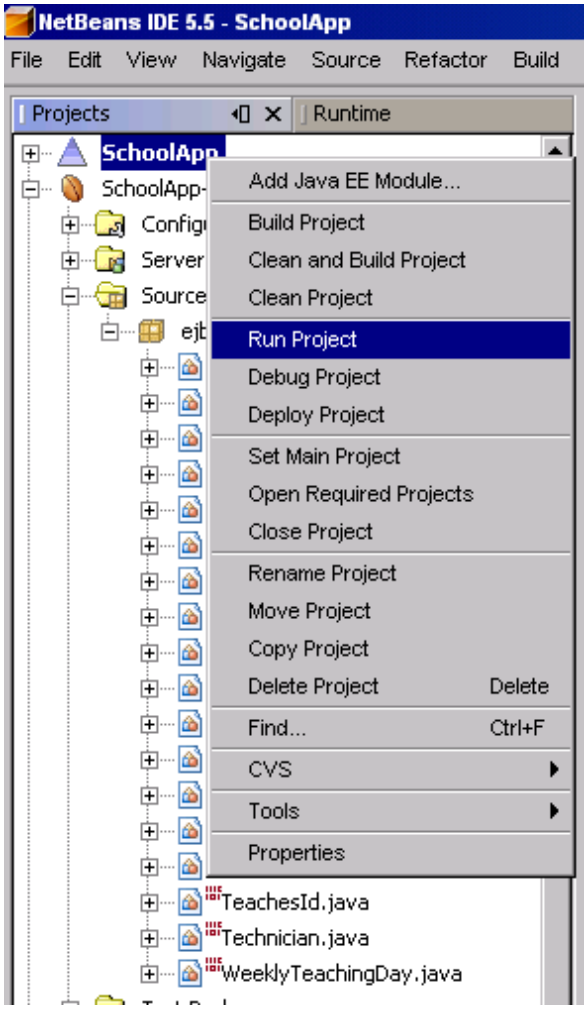


Σχήμα 8.2 Τα αρχεία της εφαρμογής SchoolApp του στρώματος ιστού

8.2 Παραδείγματα χρήσης της εφαρμογής

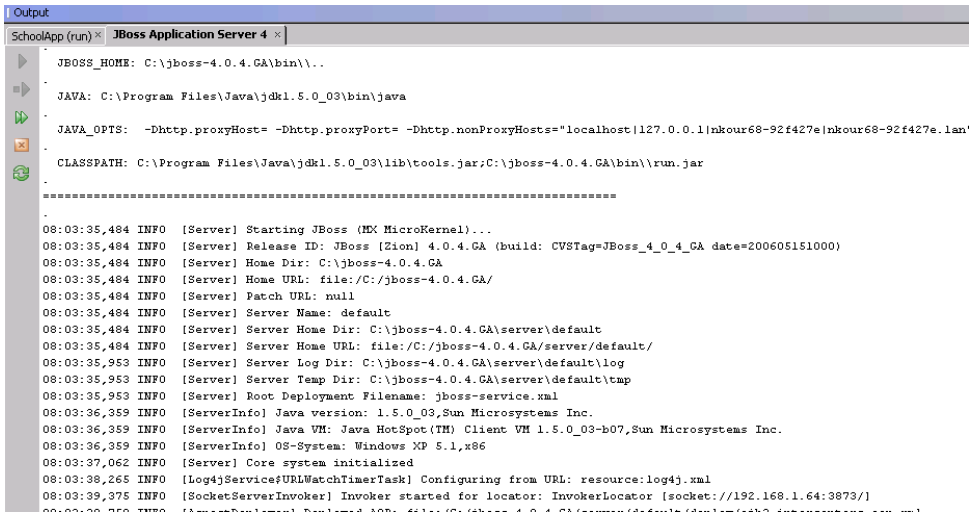
8.2.1 Εκκίνηση της Εφαρμογής

Η εφαρμογή ξεκινάει με την εκτέλεση του servlet Home (Σχήμα 8.3) με ταυτόχρονη εκκίνηση του εξυπηρετητή ιστού tomcat και του



Σχήμα 8.3 Εκκίνηση τη εφαρμογής από το περιβάλλον NetBeans

εξυπηρετητή εφαρμογών Jboss για το κτίσιμο του επιχειρησιακού στρώματος της εφαρμογής (Σχήμα 8.4).



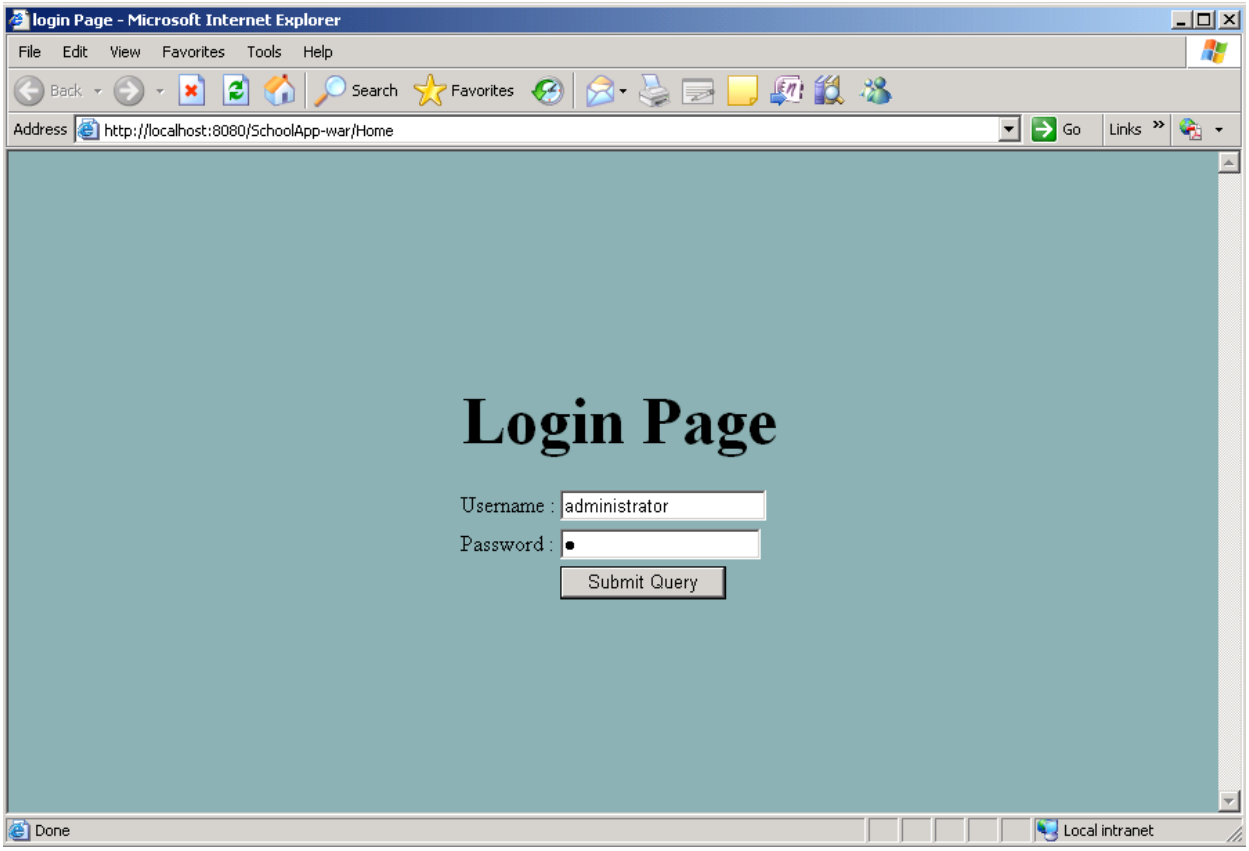
Σχήμα 8.4 Εκκίνηση του εξυπηρετητή εφαρμογών Jboss

Με το τέλος της εκτέλεσης παρουσιάζεται η Login (login.jsp) ιστοσελίδα της εφαρμογής από τον εξυπηρετητή ιστού για την πιστοποίηση και εξουσιοδότηση των χρηστών.

8.2.2 Χρήση της εφαρμογής από τους χρήστες

8.2.2.1 Χρήση της εφαρμογής από τον Διαχειριστή

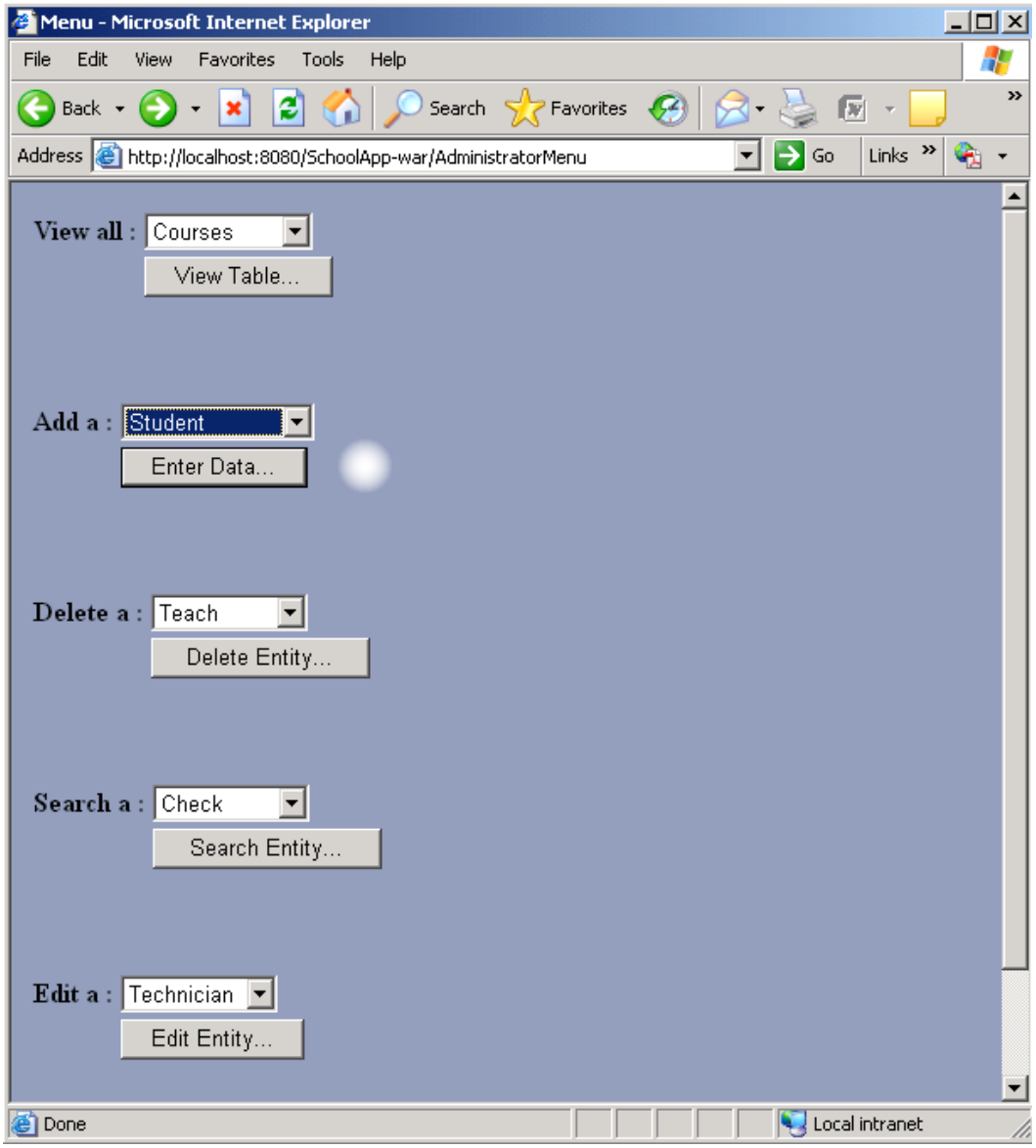
Χρήση της εφαρμογής από τον Διαχειριστή (Administrator). Σενάριο εγγραφή Σπουδαστή (Student).



.

.

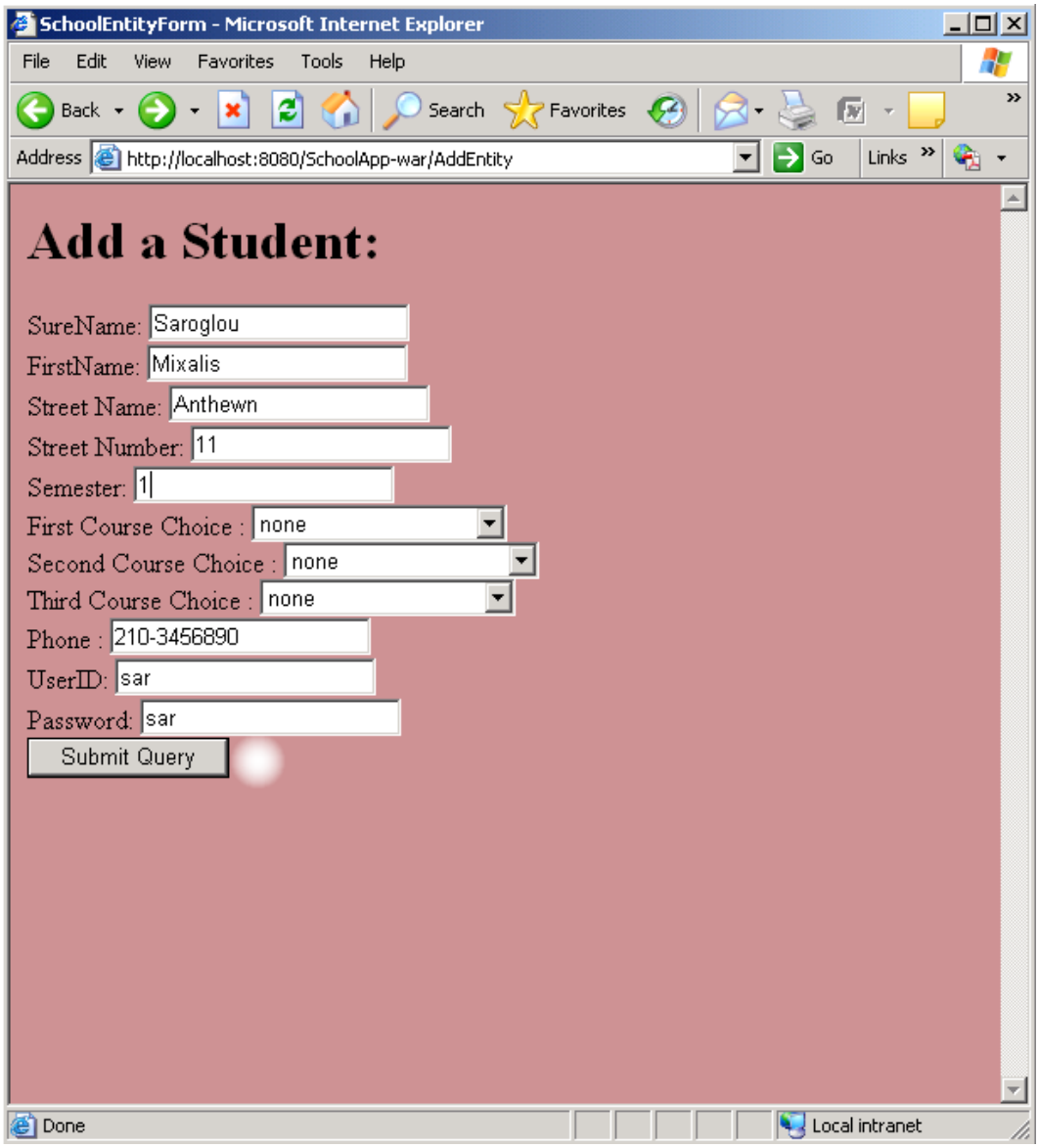
v


Menu - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/SchoolApp-war/AdministratorMenu
Go Links
View all : Courses
View Table...
Add a : Student
Enter Data...
Delete a : Teach
Delete Entity...
Search a : Check
Search Entity...
Edit a : Technician
Edit Entity...
Done
Local intranet

.

.

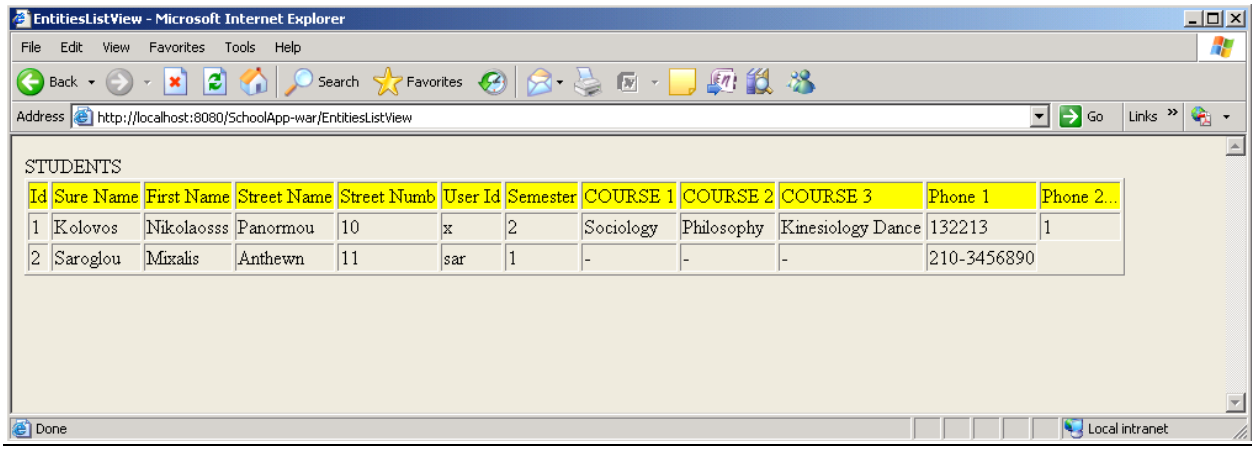
v


SchoolEntityForm - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/SchoolApp-war/AddEntity
Go Links
Add a Student:
SureName: Saroglou
FirstName: Mixalis
Street Name: Anthewn
Street Number: 11
Semester: 1
First Course Choice : none
Second Course Choice : none
Third Course Choice : none
Phone : 210-3456890
UserID: sar
Password: sar
Submit Query
Done
Local intranet

.

.

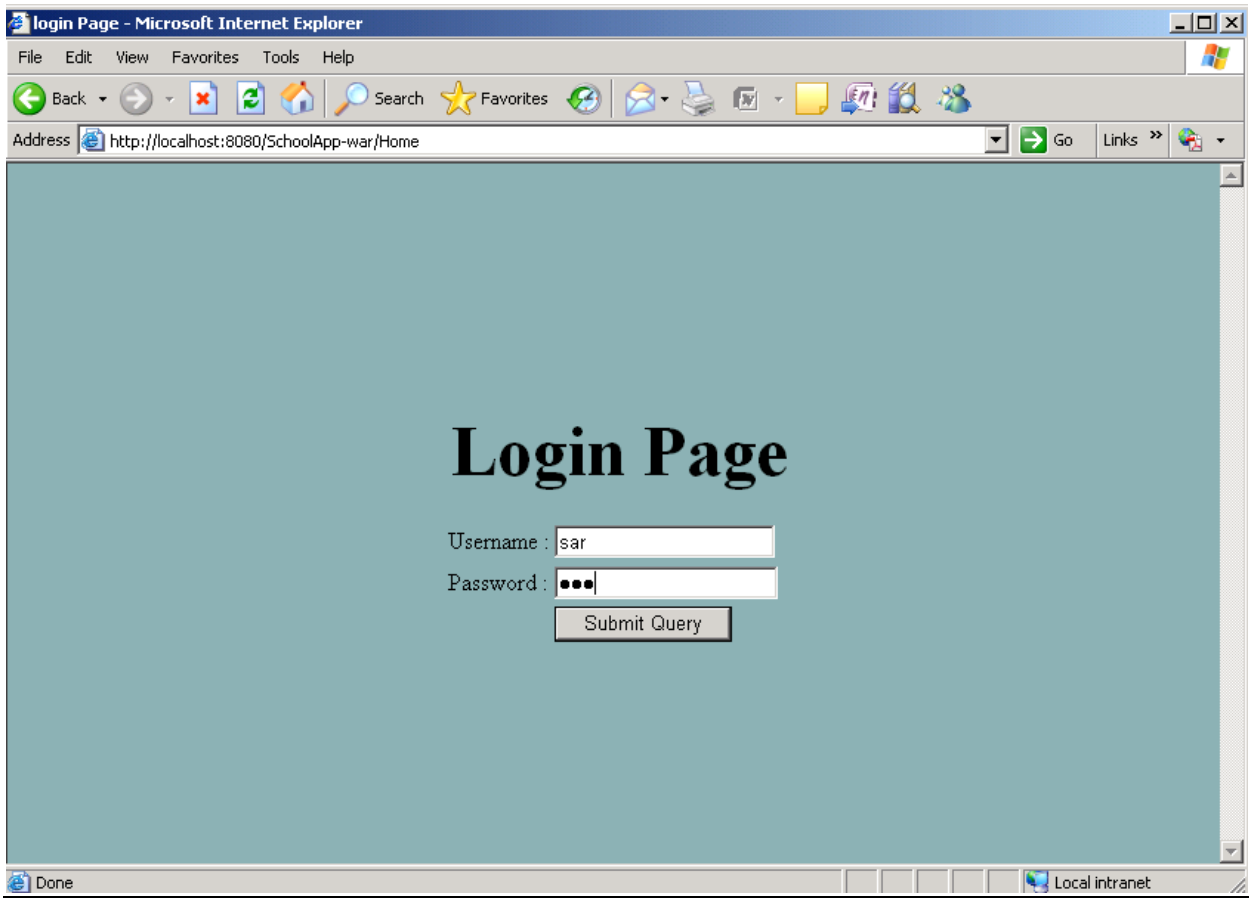
v



Σχήμα 8.5 Χρήση της εφαρμογής από τον Διαχειριστή. Σενάριο εγγραφή Σπουδαστή.

8.2.2.2 Χρήση της εφαρμογής από τον Σπουδαστή

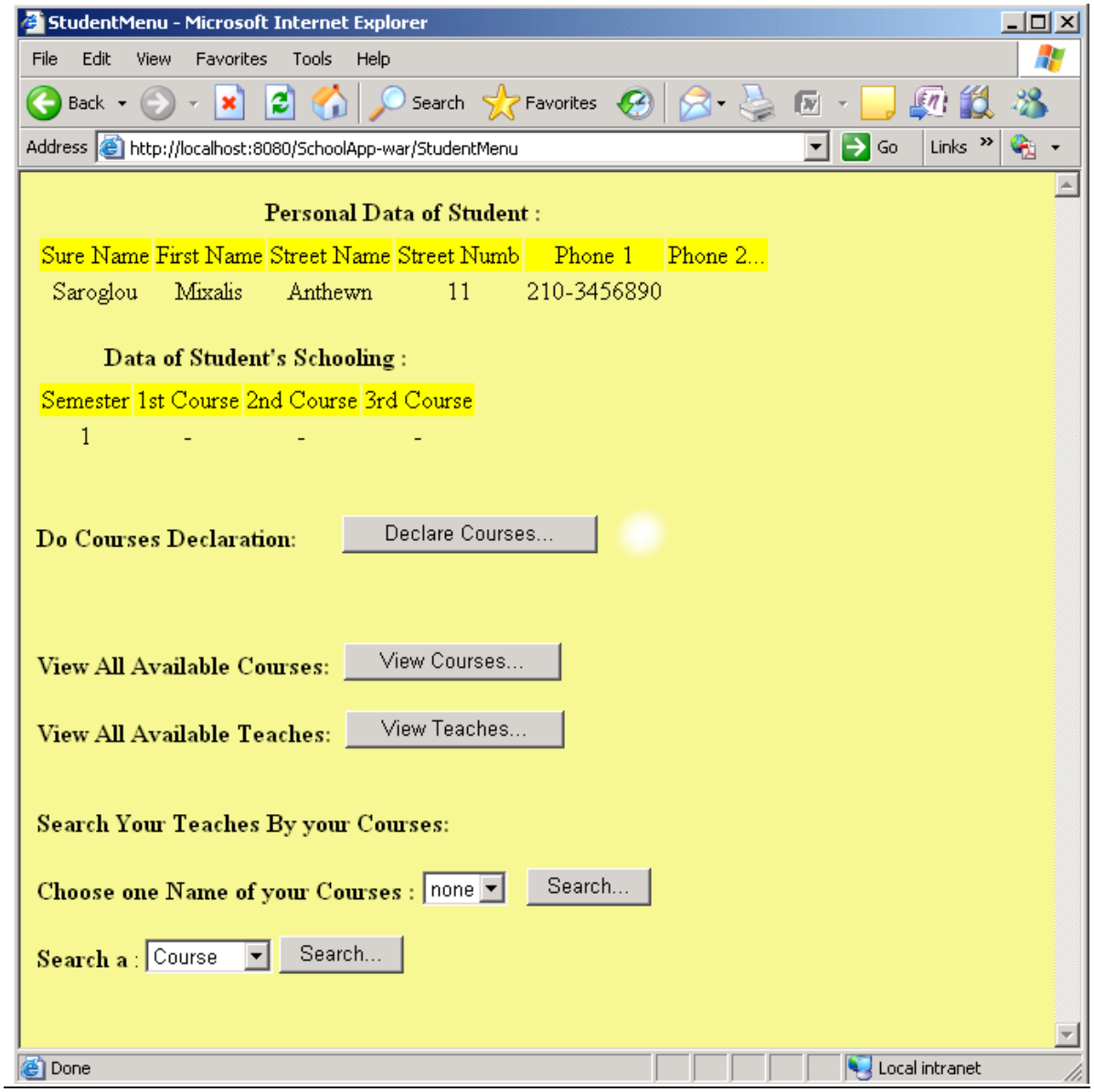
Χρήση της εφαρμογής από τον Σπουδαστή (Student). Σενάριο δήλωση μαθημάτων εξαμήνου.



.

.

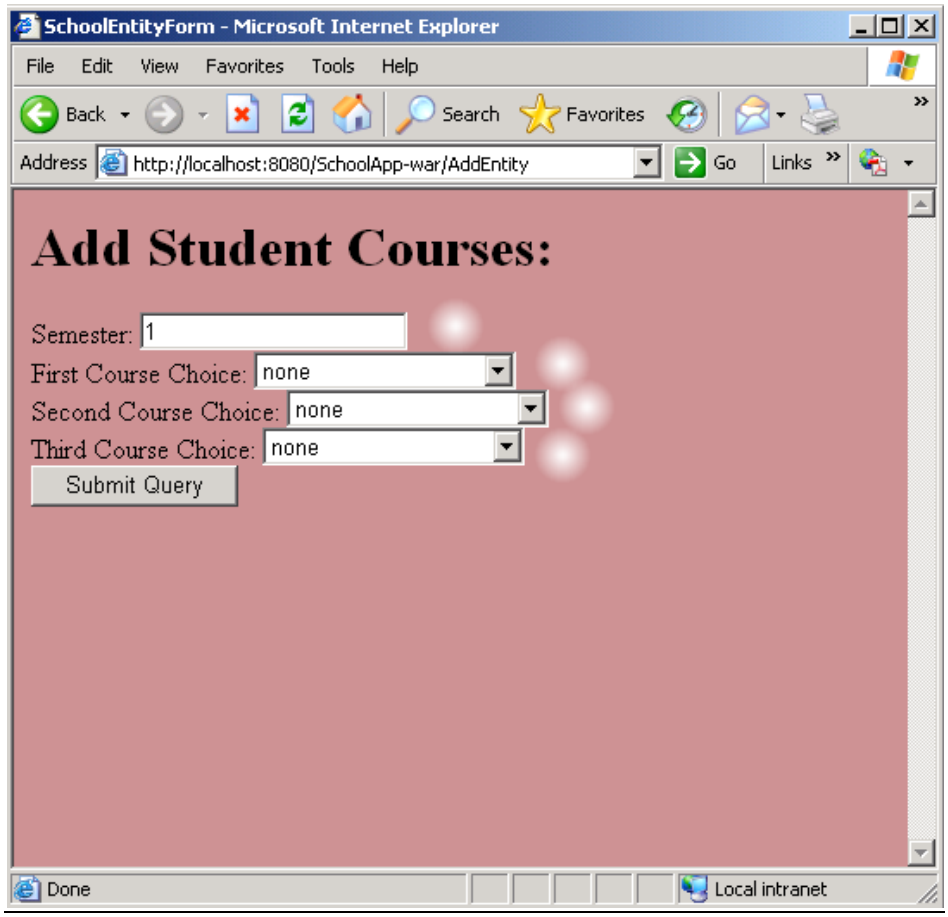
v


StudentMenu - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/SchoolApp-war/StudentMenu
Go Links
Personal Data of Student :
Sure Name First Name Street Name Street Numb Phone 1 Phone 2...
Saroglou Mixalis Anthewn 11 210-3456890
Data of Student's Schooling :
Semester 1st Course 2nd Course 3rd Course
1 - - -
Do Courses Declaration:
Declare Courses...
View All Available Courses:
View Courses...
View All Available Teaches:
View Teaches...
Search Your Teaches By your Courses:
Choose one Name of your Courses :
none
Search...
Search a :
Course
Search...
Done
Local intranet

.

.

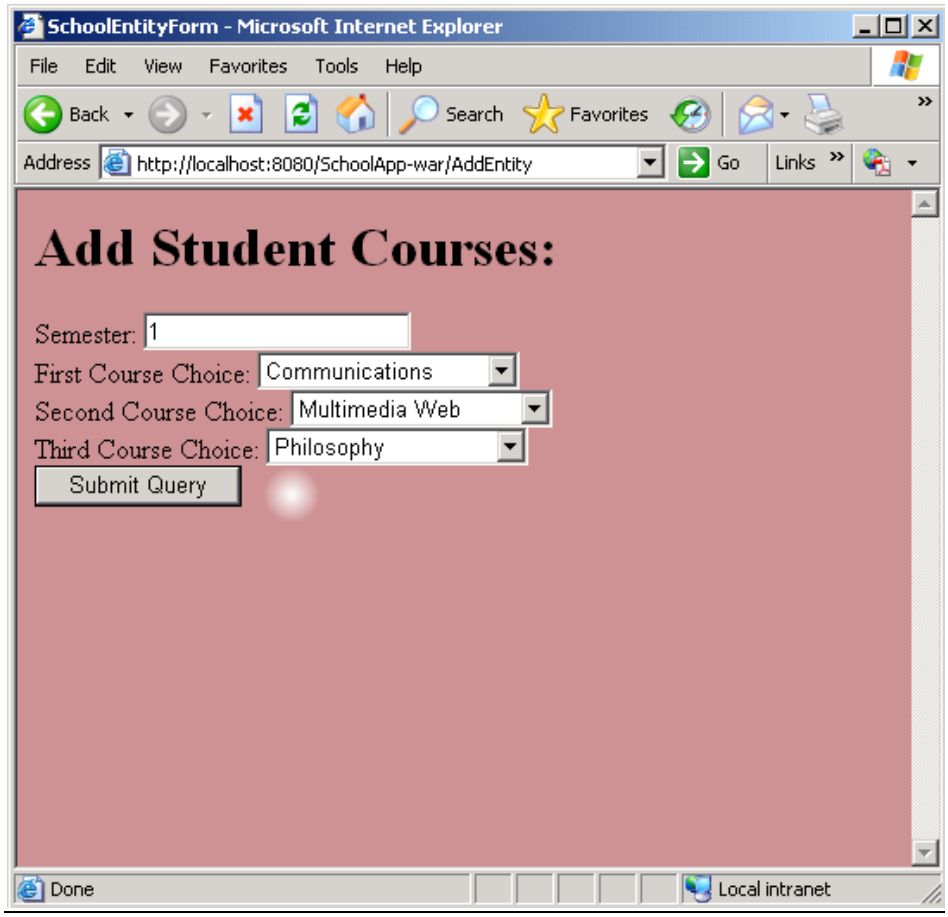
v


SchoolEntityForm - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/SchoolApp-war/AddEntity
Go Links
Add Student Courses:
Semester: 1
First Course Choice: none
Second Course Choice: none
Third Course Choice: none
Submit Query
Done
Local intranet

.

.

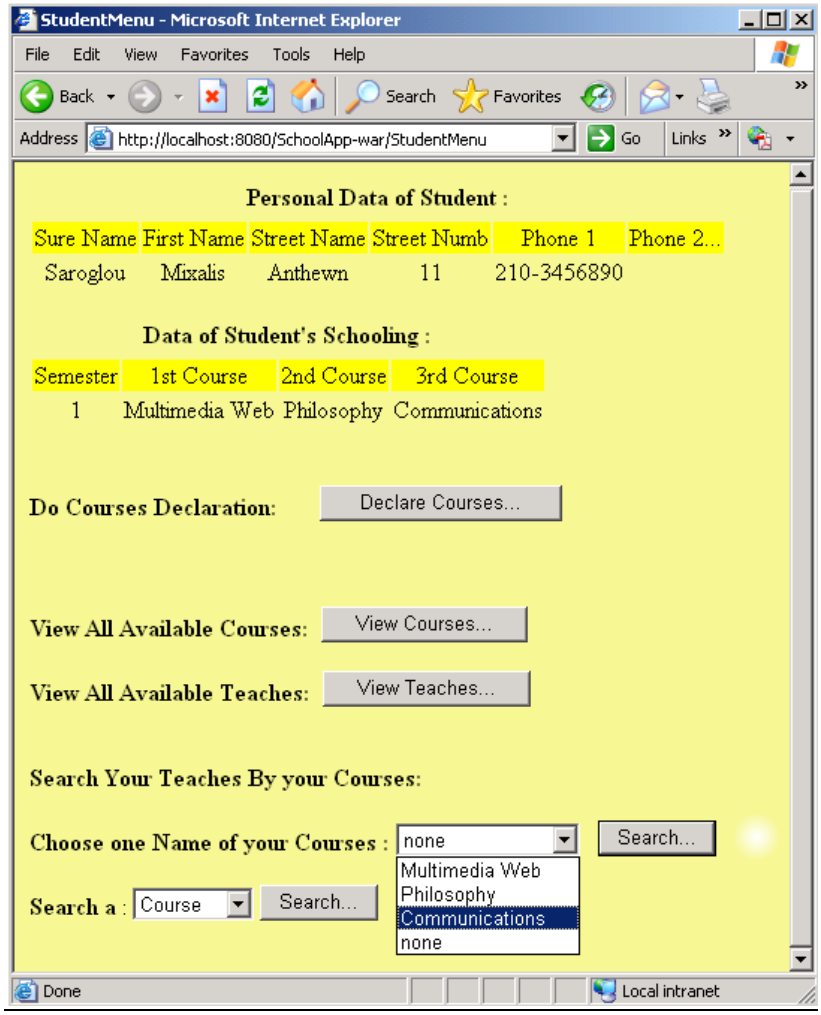
v


SchoolEntityForm - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/SchoolApp-war/AddEntity
Go Links
Add Student Courses:
Semester: 1
First Course Choice: Communications
Second Course Choice: Multimedia Web
Third Course Choice: Philosophy
Submit Query
Done
Local intranet

.

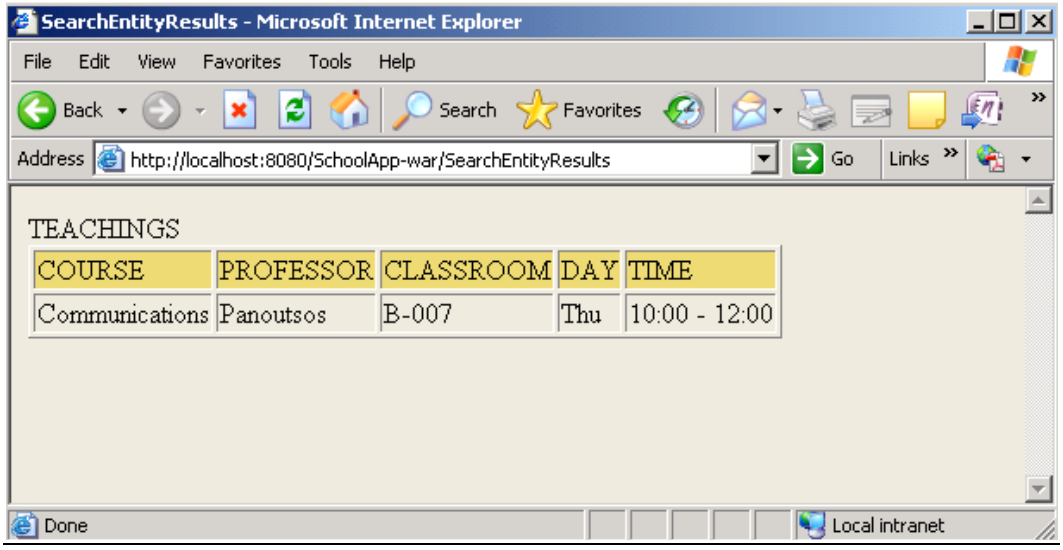
.

v



.

v



Σχήμα 8.6 Χρήση της εφαρμογής από το Σπουδαστή. Σενάριο δήλωση μαθημάτων εξαμήνου.

9

Γενικά συμπεράσματα

Αν θέλαμε να βγάλουμε κάποια συμπεράσματα για την εφαρμογή SchoolApp, αυτά θα περιορίζονταν αναγκαστικά στο πεδίο της ανάπτυξης και λιγότερο στην απόδοση της εφαρμογής. Κάτι διαφορετικό από αυτό, θα απαιτούσε πλήθος μετρήσεων και σύγκριση τους με άλλες τεχνολογίες που υλοποιούν παρόμοιες εφαρμογές απ'την πλευρά του εξυπηρετητή, πράγμα δύσκολο να υλοποιηθεί στα πλαίσια της παρούσας διπλωματικής. Έτσι λοιπόν τα όποια συμπεράσματα έχουν να κάνουν με το κατά πόσο βοηθάει στην ανάπτυξή μια πλατφόρμα ανάπτυξής όπως η J2EE καθώς και με τη χρήση της τεχνολογίας EJB στο γράψιμο εφαρμογών βάσεων δεδομένων.
Ως προς τη χρήση της πλατφόρμας J2EE για ανάπτυξη εφαρμογών, θα συμφωνήσουμε και εμείς με τα προαναφερθέντα πλεονεκτήματα, για απαλλαγή του προγραμματιστή από το γράψιμο κώδικα που έχει να κάνει με συνήθη θέματα π.χ. ασφάλειας , δοσοληψιών κ.τ.λ. της εφαρμογής και αποσαφήνισης των κομματιών της εφαρμογής (επιχειρησιακού και παρουσίασης-ιστού) με αποτέλεσμα πιο ξεκάθαρο κώδικα, πράγμα που βοηθάει την συντήρηση του.
Ως προς την χρήση της τεχνολογίας EJB τα συμπεράσματα είναι πολύ θετικά αφού βοηθάει στη συγγραφή επιχειρησιακών εφαρμογών με σωστό και οργανωμένο τρόπο , γράφοντας (χωρίς τη χρήση εντολών SQL που αναφέρονται στο φυσικό επίπεδο) κώδικα στο επίπεδο των κλάσεων της Java ο οποίος αντιστοιχείται απευθείας στο σχεσιακό μοντέλο της βάσης δεδομένων.

10

Βιβλιογραφία

[1] Keith, M., & Schincariol, M., Pro EJB 3: Java Persistence API, Apress,2006

[2] Raghu, R., Wetherbee, K., Wetherbee, J., & Zadrozny, P., Beginning EJB™ 3 Application Development: From Novice to Professional, Apress,2006

[3] Sriganesh, R., Brose, G. and Silverman, M., Mastering Enterprise JavaBeans™ 3.0, Wiley, 2006

[4] Panda, D., Rahman, R. and Lane, D., EJB 3 in Action, Manning, 2006

[5] Armstrong, A., Ball, J. and Bodoff, S., The J2EE™ 1.4 Tutorial, Sun Microsystems, 2005

[6] Sinan, A., Learning UML, O'Reilly & Associates, 2003

[7] Fowler M., Patterns of Enterprise Appllication Architecture, Addison-Wesley Signature Series, 2003

[8] Alur, D., Crupi, J., & Malks, D. , Core J2EE Patterns: Best Practices and Design Strategies, Upper Saddler River, NJ: Prentice Hall.