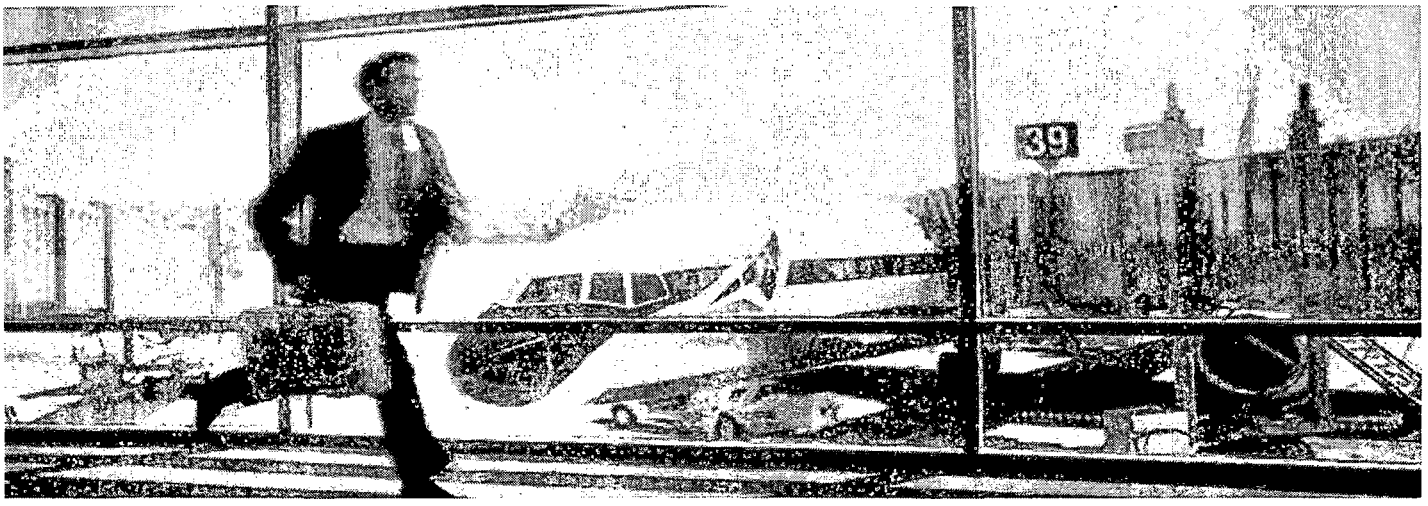
Τ.Ε.Ι. ΜΕΣΟΛΟΓΓΙΟΥ
ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ & ΟΙΚΟΝΟΜΙΑΣ
ΤΜΗΜΑ ΕΦΑΡΜΟΓΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΣΤΗ ΔΙΟΙΚΗΣΗ & ΣΤΗΝ ΟΙΚΟΝΟΜΙΑ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΘΕΜΑ: ***Σχεδιασμός & υλοποίηση συστήματος πώλησης Αεροπορικών Εισιτηρίων μέσο Internet εικονικής εταιρίας.***



Εισηγητής: κ. Φείδας Χρήστος

Σπουδαστής: Μαγκαφάς Γιώργος
Α.Μ. : 10590

ΙΟΥΝΙΟΣ 2007

Περιεχόμενα

Μαγκαφάς Γιώργος
ΑΜ 10590

Μαγκαφάς Γιώργος
ΑΜ 10590

1. Εισαγωγή

Κατά τη διάρκεια του προηγούμενου αιώνα, τα αεροπορικά ταξίδια έχουν "συρρικνώσει" τον πλανήτη, κάνοντας το πέρασμα των μεγάλων αποστάσεων ένα θέμα ωρών παρά ημερών, εβδομάδων ή και μηνών ακόμα.

Ακόμα, και δεδομένου ακόμη ότι τα ταξίδια συνέχισαν να αναπτύσσονται με γρηγορότερους ρυθμούς , η βιομηχανία ταξιδιών παρέμεινε στα παραδοσιακά εισιτήρια εγγράφου και την παραδοσιακή αρχειοθέτηση.

Οι υπολογιστές και τα ηλεκτρονικά δεδομένα είναι, τελικά, πολύ νεώτεροι από τα αεροπλάνα ή τη βιομηχανία ταξιδιωτικών γραφείων, που ήταν ήδη μια επιχείρηση δισεκατομμυρίων δολαρίων 40 έτη πριν, όταν ακόμη τα αεροπορικά ταξίδια ήταν μια πολύ μικρότερη αγορά και το δολάριο είχε μια πολύ μεγαλύτερη αξία.

Μόλις πρόσφατα στα μέσα της δεκαετίας του '90, οι ταξιδιωτικοί πράκτορες συνέχισαν να τυπώνουν τα ταξιδιωτικά δελτία, να γράφουν τις εκθέσεις πωλήσεων με το χέρι και να στέλνουν τα έγγραφα από τα αρχεία πωλήσεων στις οικονομικές επιχειρήσεις για τακτοποίηση με την keypunch μέθοδο (τρύπημα καρτών).

Η διαδικασία αυτή ήταν χρονοβόρα και έφερε την δυνατότητα για, αφθονία από προβλήματα, απάτες, χαμένες πληροφορίες, λάθη εισαγωγής δεδομένων και την ανεπαρκή πρόσβαση στα αρχεία.

Αλλά η βιομηχανία θα κινούταν σύντομα κατά μήκος μιας νέας, όλο και περισσότερο ηλεκτρονικής τροχιάς. Το 1984, ως ένα αποτέλεσμα μίας δεκαετίας άρσης των ελέγχων στη βιομηχανία των αερογραμμών, αυτή η γρήγορα αυξανόμενη οικονομική υπηρεσία τραπεζικών διευθετήσεων ξέφυγε από την ATA (Air Transport Association) για να γίνει Airlines Reporting Corporation (ARC), μια ανεξάρτητη επιχείρηση. Από τότε, η επιχείρηση αυτή έχει οδηγήσει τη βιομηχανία στην ψηφιακή εποχή. Μετά από την εισαγωγή της πρώτης ηλεκτρονικής πιστωτικής κάρτας (ECCB) το 1989, η επιχείρηση εγκατέστησε την πρώτη πλήρως-ηλεκτρονική αγορά εισιτηρίων ακριβώς έξι χρόνια αργότερα.

Το 1997, η ARC εισήγαγε ένα πλήρως-ηλεκτρονικό σύστημα υποβολής εκθέσεων, και το 2001, ανέπτυξε ένα πρόγραμμα ανάκτησης και καταγραφής ηλεκτρονικών εισιτηρίων.

Σήμερα η ARC επεξεργάζεται, εκθέτει και τακτοποιεί 150 εκατομμύρια αεροπορικά εισιτήρια, 90% των οποίων είναι ηλεκτρονικά, κάθε έτος για τις πτήσεις που εξυπηρετούν τις Ηνωμένες Πολιτείες, το Πουέρτο Ρίκο και τα US Virgin Islands.

Μαγκαφάς Γιώργος
ΑΜ 10590

2. Σκοπός της πτυχιακής εργασίας

Το σύστημα πώλησης αεροπορικών εισιτηρίων απευθύνετε σε αεροπορικές εταιρίες για την κάλυψη των αναγκών τους στον τομέα της εξυπηρέτησης πελατών επί 24ωρης βάσης.

Για να υπάρχει μια ολοκληρωμένη εξυπηρέτηση από μέρους της εταιρείας και κυρίως των πελατών της θα δημιουργηθεί μια εφαρμογή που θα προσφέρει πραγματική και άμεση εξυπηρέτηση προς όλους τους πελάτες της που διαθέτουν ένα ηλεκτρονικό υπολογιστή με πρόσβαση στο διαδίκτυο. Έτσι θα υπάρχει ευκολία πρόσβασης χωρίς περιορισμό χώρου και χρόνου.

Με λίγα λόγια τα βασικότερα πλεονεκτήματα της εφαρμογής με βάση το σύνολο των χρηστών είναι:

- Δυνατότητα 24ωρης λειτουργίας
- Αποδέσμευση του προσωπικού της Εταιρίας
- Πρόσβαση μέσω Internet
- Μέγιστη ασφάλεια δεδομένων προσωπικού χαρακτήρα.

Η πτυχιακή αυτή, αφορά τη πώληση εισιτηρίων και δημιουργήθηκε για να καλύψει τις ανάγκες της αεροπορικής εταιρείας. Θα εκμεταλλευτούμε τις νεότερες εξελίξεις στα θέματα της δημιουργίας και διαχείρισης βάσεων δεδομένων, της δημιουργίας και διαχείρισης του γραφικού περιβάλλοντος της φιλικότητας και ασφάλειας των λειτουργικών συστημάτων.

Τη διακρίνει η επεκτασιμότητα, λόγω της ευελιξίας της γίνεται κάθε πιθανή τροποποίηση της για την προσαρμογή της σε κάποιες ειδικές ανάγκες της εταιρίας, που ίσως παρουσιαστούν.

3. Τεχνολογίες Ανάπτυξης Πληροφοριακών Συστημάτων Διαδικτύου

Η ανάπτυξη Πληροφοριακών Συστημάτων στις μέρες μας είναι προκλητικό, αλλά συνάμα δύσκολο έργο και η ανάπτυξη καλών Πληροφοριακών Συστημάτων είναι ακόμη πιο δύσκολη. Τα σύγχρονα Πληροφοριακά Συστήματα είναι συνήθως μεγάλα, σύνθετα και πολύπλοκα. Ως αποτέλεσμα, η διαδικασία ανάπτυξής τους εμπεριέχει παγίδες, σημαντικά ρίσκα και υψηλά κόστη.

Η Ανάλυση και Σχεδίαση Πληροφοριακών Συστημάτων παρέχει μια δομημένη και πειθαρχημένη προσέγγιση στη διαδικασία ανάπτυξης Πληροφοριακών Συστημάτων, με απώτερο στόχο την αντιμετώπιση των δυσκολιών, προβλημάτων και κινδύνων τις οποίες η διαδικασία αυτή εμπερικλείει.

Είναι το μέσο για τη συνεργασία μεταξύ:

- ανθρώπινου δυναμικού
- δεδομένων
- διαδικασιών
- υλικού (δικτύου υπολογιστών, εκτυπωτών κλπ)
- τεχνολογίας της πληροφορικής (IT)

Η συνεργασία αυτή σκοπό έχει την υποστήριξη και βελτίωση των καθημερινών λειτουργιών, καθώς επίσης και για την υποστήριξη για λύσεις προβλημάτων και για τις ανάγκες λήψης αποφάσεων.

4. Δημιουργία Ιστοσελίδων.

Για τη δημιουργία ιστοσελίδων χρησιμοποιούνται ειδικές γλώσσες προγραμματισμού, μερικές από αυτές αναγράφονται παρακάτω.

- Στατικό Περιεχόμενο
  - HTML
- Δυναμικό Περιεχόμενο
  - JSP
  - PHP
  - JavaScripts
  - XML

4.1. HTML

Η *HTML* είναι το ακρωνύμιο των λέξεων ***HyperText Markup Language*** (γλώσσα μορφοποίησης υπερκειμένου) και είναι η βασική γλώσσα δόμησης σελίδων του *World Wide Web*. Είναι μία γλώσσα προγραμματισμού. Επιτρέπει την ενσωμάτωση ήχου και εικόνων στις web σελίδες. Αρχικά είχε κατασκευασθεί με σκοπό μόνο την μορφοποίηση κειμένου, αλλά στη συνέχεια ενσωμάτωσε σχεδιαστικές τεχνικές κ.α. Βασικός στόχος στο σχεδιασμό της, ήταν η απλότητα της γλώσσας, γεγονός που φαίνεται εύκολα και βοήθησε στη σημερινή δημοτικότητα της και στην ανάπτυξη της. Ακόμα και κάποιος που δεν είναι προγραμματιστής μπορεί να φτιάξει μια απλή σελίδα HTML χρησιμοποιώντας ακόμα κι ένα απλό text editor. Η απλότητα στο σχεδιασμό της HTML ήταν ένα βασικό στοιχείο της επιτυχίας του WWW, καθώς ο καθένας μπορούσε χωρίς ιδιαίτερες γνώσεις να φτιάξει μια απλή WWW σελίδα.

Η γλώσσα χρησιμοποιεί έναν αριθμό από *tags* για την μορφοποίηση κειμένου, για την δημιουργία συνδέσμων *(links)* μετάβασης ανάμεσα των σελίδων, για την εισαγωγή εικόνων, ήχου κ.α. Όταν ένας Web Browser ανοίγει ένα αρχείο HTML τα στοιχεία *(tags)* μεταφράζονται σε κατάλληλα χαρακτηριστικά με αποτελέσματα στην εμφάνιση και στην λειτουργικότητα της συγκεκριμένης σελίδας, να έχει ένα απλό αλλά κατανοητό interface και να είναι όσο πιο εύχρηστη γίνεται.

Όμως η εκρηκτική εξέλιξη του WWW είχε σαν αποτέλεσμα την ανάγκη δημιουργίας ιστοσελίδων με δυναμικό περιεχόμενο.

4.2. PHP

Η PHP, όπου τα αρχικά σημαίνουν Hypertext PreProcessor, είναι μια γλώσσα συγγραφής σεναρίων (scripting language) που ενσωματώνεται μέσα στον κώδικα της HTML και εκτελείται στην πλευρά του server (server-side scripting).
Το μεγαλύτερο μέρος της σύνταξής της, η PHP το έχει δανειστεί από την C, την Java και την Perl και διαθέτει και μερικά δικά της μοναδικά χαρακτηριστικά. Ο σκοπός της γλώσσας είναι να δώσει τη δυνατότητα στους web developers να δημιουργούν δυναμικά παραγόμενες ιστοσελίδες.

Αντί να γράψουμε ένα πρόγραμμα με πολλές εντολές για να δημιουργήσουμε κώδικα HTML, γράφουμε ένα HTML script με κάποιον ενσωματωμένο κώδικα για να κάνει ορισμένες ενέργειες που του αναθέσαμε. Ο κώδικας της PHP περικλείεται με ειδικά tags αρχής και τέλους για να μπορούμε να εισερχόμαστε και να εξερχόμαστε από το PHP mode.

Αυτό που την ξεχωρίζει από μια γλώσσα όπως η JavaScript, η οποία εκτελείται στην πλευρά του χρήστη (client-side), είναι ότι ο κώδικάς της εκτελείται στον server. Αν είχαμε σ' έναν server ένα script, ο χρήστης (client) θα λάμβανε το αποτέλεσμα της εκτέλεσης αυτού του script, χωρίς να είναι σε θέση να γνωρίζει ποιος μπορεί να είναι ο αρχικός κώδικας.

Μπορούμε ακόμη να ρυθμίσουμε (configure) τον web server ώστε να επεξεργάζεται όλα τα HTML αρχεία με την PHP και τότε δεν θα υπάρχει πράγματι κανένας τρόπος να μάθουν οι χρήστες τον κώδικά μας.

Στο πιο βασικό επίπεδο, η PHP μπορεί να επεξεργάζεται τα δεδομένα μιας φόρμας, να δημιουργεί δυναμικά περιεχόμενα ιστοσελίδων ή αποστολή και λήψη cookies. Ίσως το δυνατότερο και πιο σημαντικό χαρακτηριστικό της PHP είναι η υποστήριξη που παρέχει σε μια ευρεία γκάμα από βάσεις δεδομένων. Υποστηρίζει τις εξής βάσεις δεδομένων:

Adabas D	dBase	Empress	FilePro	Informix	InterBase	Msql
MySQL	Oracle	PostgreSQL	Solid	Sybase	Velocis	Unix dbm

Η PHP παρέχει επίσης υποστήριξη για συνομιλία με άλλες υπηρεσίες, χρησιμοποιώντας πρωτόκολλα όπως τα IMAP, SNMP, NNTP, POP3 ή και το HTTP. Η PHP είναι μια εξαιρετικά ικανή γλώσσα, με ένα μεγάλο εύρος ενσωματωμένων λειτουργιών (built-in functions) οι οποίες κάνουν σχεδόν τα πάντα ξεκινώντας από το tracking των user sessions μέχρι τη δημιουργία dynamic graphics και αρχείων PDF. Βασικό μειονέκτημα αυτού του συστήματος αποτελεί το γεγονός ότι δεν υπάρχει καμία επίσημη υποστήριξη για οποιοδήποτε πρόβλημα αντιμετωπίζει κάποιος με την PHP.

4.3. SQL

Η SQL (Structured Query Language - Δομημένη γλώσσα ερωτημάτων), είναι μια γλώσσα υπολογιστών που χρησιμοποιείται για εκφράσει διαδικασίες βάσεων δεδομένων για δεδομένα που είναι οργανωμένα σε σχεσιακή φόρμα (π.χ. σε πίνακες). Η SQL υιοθετήθηκε ως πρότυπο της βιομηχανίας το 1986.

Για να εργαστούν μαζί οι άνθρωποι και οι βάσεις δεδομένων πρέπει να μιλούν μια κοινή γλώσσα. Αν και σε γενικές γραμμές κάθε κατασκευαστής βάσεων δεδομένων θα μπορούσε να καθορίσει μια δικιά του τέτοια γλώσσα, αυτό όμως θα προκαλούσε προβλήματα και για το χρήστη και για τους προμηθευτές βάσεων δεδομένων.
Άντ' αυτού, οριστικέ μια σταθερή γλώσσα η SQL ,την οποία υποστηρίζουν όλοι οι προμηθευτές βάσεων δεδομένων.

Η SQL διαφέρει από τις διαδικαστικές γλώσσες στο ότι επικεντρώνεται στον ορισμό και το χειρισμό των. δεδομένων και οι εντολές της είναι πολύ κατανοητές μια και γράφονται σε απλά αγγλικά.

Η SQL επιτρέπει στους χρηστές να έχούν πρόσβαση σε δεδομένα σχεσιακών μοντέλων όπως αυτά της Oracle MySQL Microsoft SQL Serve Access και αλλά επιτρέποντας στους χρηστές να επιλέγουν ποια δεδομένα θέλουν να δουν.

Η SQL επιτρέπει στους χρηστές να ορίσουν τα δεδομένα και να τα χειριστούν. Αυτή εργασία θα χρησιμοποιήσει SQL εντολές για χειριστεί το σύστημα. Η SQL έχει δυο κύρια συστατικά:

- Data Manipulation Language (DML) – Γλώσσα Χειρισμού Δεδομένων συμπεριλαμβάνοντας τις εντολές όπως: SELECT, INSERT.
- Data Definition Language (DDL) – Γλώσσα Καθορισμού Στοιχείων συμπεριλαμβάνοντας τις εντολές όπως: CREATE, DROP, ALTER.

4.4. JDBC

Πριν μερικά χρόνια, για να αντιμετωπίσει η Microsoft το πρόβλημα των διαφορετικών τύπων βάσεων δεδομένων ανέπτυξε το πρότυπο ODBC(Open Data Base Connectivity), ένα πρότυπο για την προσπέλαση βάσεων δεδομένων. Το ODBC καθορίζει ένα συγκεκριμένο τύπο συμπεριφοράς των εφαρμογών στην περίπτωση που αυτές καλούν βάσεις δεδομένων, και έναν άλλον όταν μια συγκεκριμένη βάση δεδομένων ανταποκρίνεται σε κλήσεις τύπου ODBC από μια εφαρμογή.

Η εταιρεία Sun συμπεριέλαβε στην Java τη διασύνδεση JDBC η οποία καθορίζει τον τρόπο επικοινωνίας μιας εφαρμογής με Σχεσιακές Βάσεις Δεδομένων. Η γλώσσα με αυτόν τον τρόπο κατορθώνει να υποστηρίζει σχεσιακές βάσεις δεδομένων που δημιουργούνται από διαφορετικά DBMS όπως κατορθώνει να υποστηρίζει και τα διαφορετικά λειτουργικά συστήματα.

Το JDBC επιτρέπει στον προγραμματιστή να γράψει εφαρμογές βάσεων δεδομένων με τη χρήση αποκλειστικά και μόνο της γλώσσας Java. Παρότι η ίδια η εταιρεία Sun δηλώνει ότι το JDBC δεν είναι ακρωνύμιο πολλοί θεωρούν ότι είναι το ακρωνύμιο των λέξεων Java Data Base Connectivity. Όπως και το ODBC της Microsoft έτσι και το JDBC καθορίζει το πώς οι εφαρμογές θα επικοινωνούν με τις βάσεις δεδομένων και ταυτόχρονα το πώς θα κατασκευάζονται οι Β.Δ. ώστε να μπορούν να ανταποκρίνονται στις κλήσεις των εφαρμογών. Η διάφορα είναι ότι το ODBC είναι γραμμένο σε γλώσσα **C**, ενώ το JDBC είναι αποτελείται αποκλειστικά από κλάσεις και διασυνδέσεις αποκλειστικά γραμμένες σε **Java**.

Το JDBC δίνει στον προγραμματιστή τις εξής δυνατότητες:

- Να συνδέσει μια εφαρμογή με μια υπάρχουσα βάση δεδομένων.
- Να δημιουργήσει εντολές SQL και να τις περάσει στην βάση δεδομένων προκείμενου να εκτελέσει διάφορες εργασίες .
- Να πάρει τα αποτελέσματα των επεξεργασιών των εντολών SQL από τη βάση δεδομένων .
- Να επεξεργαστεί περαιτέρω τα αποτελέσματα που έλαβε.

Η JSP, η κύρια γλώσσα που θα χρησιμοποιήσουμε σε αυτήν την εργασία, χρησιμοποιεί τις βιβλιοθήκες JDBC για να επικοινωνεί με την βάση δεδομένων έτσι ώστε τα δεδομένα της βάσης να μπορούν να ενημερωθούν, να ανακτηθούν, ή να διαγραφούν χρησιμοποιώντας JSP φόρμες.

Ένα πρόγραμμα που κάνει χρήση του JDBC, για να ανοίξει και να χρησιμοποιήσει μια βάση δεδομένων ακολουθεί τα παρακάτω βήματα:

- Εισάγει τις απαραίτητες κλάσεις
- Φορτώνει τον κατάλληλο οδηγό JDBC
- Προσδιορίζει την πηγή των δεδομένων
- Δημιουργεί ένα αντικείμενο Connection
- Δημιουργεί ένα αντικείμενο Statement
- Κάνει ερώτηση (query) στη Β.Δ. με την βοήθεια του αντικειμένου Statement
- Παίρνει δεδομένα από το αντικείμενο ResultSet που επιστρέφεται
- Κλείνει το αντικείμενο ResultSet
- Κλείνει το αντικείμενο Statement
- Κλείνει το αντικείμενο Connection

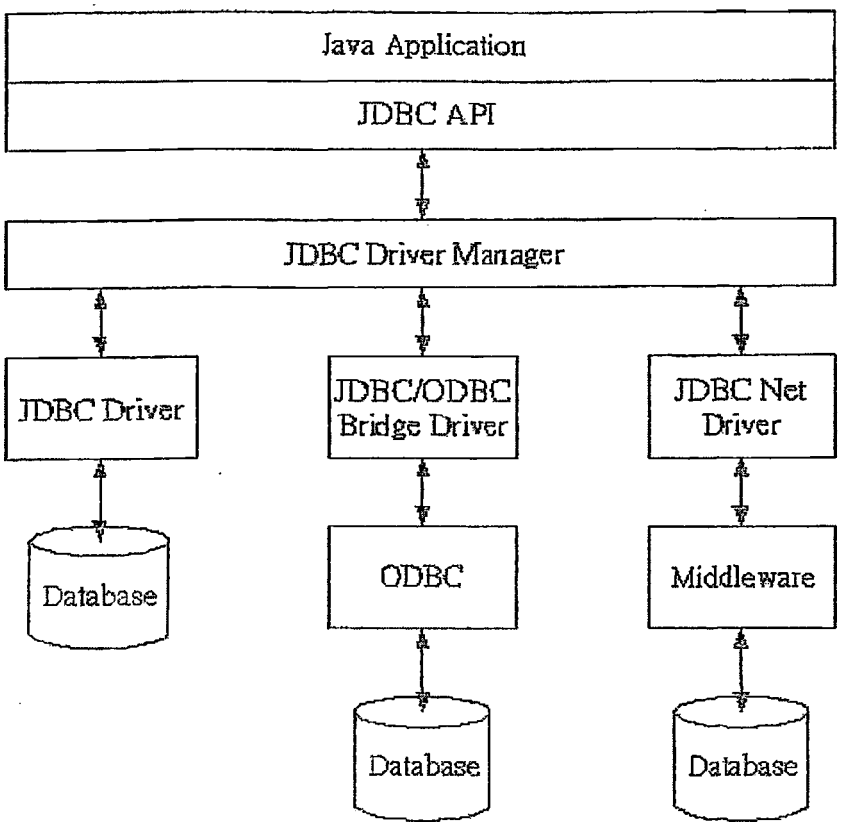
Η όλη διεργασία ξεκινάει με την κλάση Driver Manager. Αυτός συνδέει τον κατάλληλο οδηγό JDBC (που αντιστοιχεί στη βάση δεδομένων με την οποία θέλουμε να επικοινωνήσουμε) με την εφαρμογή μας. Μόλις αποκατασταθεί η σύνδεση, ο Driver Manager διεκπεραιώνει την επικοινωνία μεταξύ της εφαρμογής και του οδηγού.

Στη συνέχεια η σύνδεση γίνεται με τρεις τρόπους:

1. Με τον JDBC Driver και μετά την βάση δεδομένων
2. Τη γέφυρα JDBC-ODBC .Η γέφυρα επιτρέπει σε μια εφαρμογή συμβατή με JDBC να επικοινωνεί με βάση δεδομένων συμβατή με ODBC. Η γέφυρα μεταφέρει εντολές και αποτελέσματα μέσω του οδηγού ODBC.
3. Και τον JDBC NET Driver.

Οι JDBC Drivers είναι κατηγοριοποιημένοι σε τέσσερις τύπους:

1. Τύπος 1 – Η γέφυρα JDBC-ODBC
2. Τύπος 2 – Εγγενής δυαδικός κώδικας. Παρόμοιος με την γέφυρα JDBC-ODBC μόνο που μπαίνει μέσα σε κώδικα Java.
3. Τύπος 3 – Καθαρή Java στο middleware (υλικολογισμικό) της βάσης δεδομένων. Οι οδηγοί τύπου 3 επικοινωνούν χρησιμοποιώντας πρωτόκολλα δικτύων με middleware servers, οι οποίοι επικοινωνούν με ένα η παραπάνω Σύστημα Διαχείρισης Βάσεων Δεδομένων.(JDBC NET Driver)
4. Τύπος 4 – Καθαρή Java απ' ευθείας στη βάση δεδομένων. Αυτός ο τύπος οδηγού μπαίνει κατευθείαν στο εγγενές πρωτόκολλο που χρησιμοποιείται από σύστημα διαχείρισης βάσεων δεδομένων.



Η ΔΟΜΗ ΤΗΣ ΒΙΒΛΙΟΘΗΚΗΣ JDBC

Και τύπος 1 και ο τύπος 2 απαιτούν κώδικα να εγκατασταθεί και συντεθεί ανάλογα στα συστήματα των πελατών.

Ο τύπος 3 χρησιμοποιεί ένα πρωτόκολλο δικτύων που επικοινωνεί μεταξύ του κεντρικού υπολογιστή και του πελάτη.

Οι οδηγοί τύπου 4 μπορεί να μην είναι κατάλληλοι για συστήματα διαχείρισης βάσεων δεδομένων που είναι πίσω από firewall αλλά είναι αρκετά κατάλληλα για java servlets και JSP(java server page) που είναι συνδεδεμένα σε μια database server που είναι στο ίδιο hardware η στο ίδιο τοπικό δίκτυο.

5. Servlets, JavaServer Pages και JavaBeans

5.1. Servlet

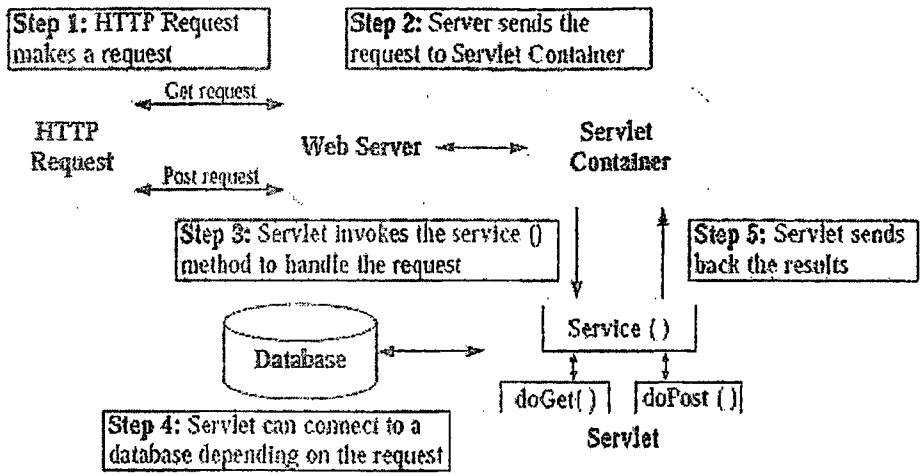
5.1.1. Τι είναι ένα Servlet

Με τον όρο servlet (μικροεφαρμογή διακομιστή) εννοούμε μια εφαρμογή ή ένα σενάριο (script) γραμμένο σε γλώσσα Java το οποίο εκτελείται σε έναν server (π.χ Ιστού) σε αντίθεση με τα Προγράμματα applets που εκτελούνται σε clients pc.

Servlet είναι μια δυναμική κλάση που εξυπηρετεί αιτήματα από το internet ή από έναν server. Τα Servlet είναι σημείο κλειδί για την ανάπτυξη της java ως προς τους server. Ένα Servlet είναι μια μικρή επέκταση σε ένα server που εμπλουτίζει την λειτουργικότητα του server. Τα Servlet επιτρέπουν στους προγραμματιστές να επεκτείνουν η να τροποποιήσουν τύπο server (web server, mail server η οποιονδήποτε άλλο server) που είναι βασισμένος στη java με ένα άγνωστο μέχρι τώρα βαθμό ευκαμψίας, μεταφερσιμότητας και ευκολίας.

Επειδή τα Servlet «τρέχουν» σε server , δεν εξαρτώνται από την συμβατότητα με τους browser.

Η εικόνα δείχνει την ροή ενός μηνύματος ενός http αιτήματος σε ένα servlet , το οποίο ξεκινάει όταν ο χρήστης μπαίνει στο site.



Ροή μηνύματος σε ένα web server με servlet container

Μαγκαφάς Γιώργος
ΑΜ 10590

5.1.2. Κύκλος Ζωής ενός Servlet.

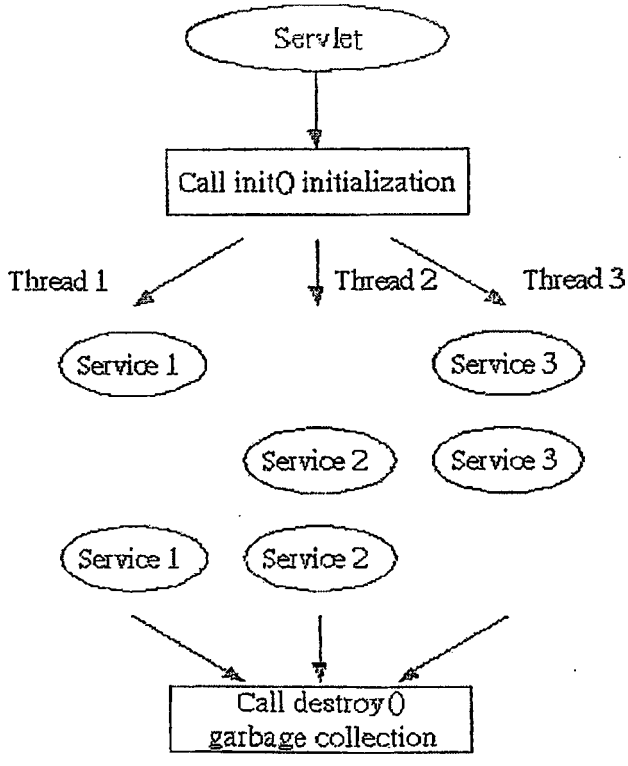
Κάθε servlet container έχει έναν ορισμένο επακριβώς κύκλο ζωής.
Γεννιέται είτε όταν servlet container ξεκινάει, στο πρώτο αίτημα, είτε πάνω στο πρώτο αίτημα μετά από κάποια αλλαγή στην κλάση (αρχείο) του Servlet.

Όλα τα servlets υπάρχουν για να διαχειρίζονται αιτήματα και λήγουν όταν βρεθούν εξαιρέσεις (σφάλματα), πάνω στην διαδικασία φόρτωσης από το servlet container, ή όταν το servlet container σταματήσει να λειτουργεί.

Για την επεξεργασία του αιτήματος, κάθε servlet πρέπει να εκτελεί τρεις «στάνταρ» μεθόδους: την init ( ), service ( ) και την destroy ( ).

Το container ενός servlet κάνει τα παρακάτω:

- Ο server φορτώνει το servlet όταν αρχικά ζητηθεί
- Ο server εκτελεί την μέθοδο init ( ) του servlet
- Το servlet χρησιμοποιεί πολύ-επεξεργασία για να διαχειριστή τα αιτήματα καλώντας την μέθοδο service ( ).
- Όταν τελειώνει η διαδικασία καλείται η μέθοδος destroy ( ) του κάθε servlet για «καταστρέψει» το servlet.



Ο κύκλος ζωής ενός servlet

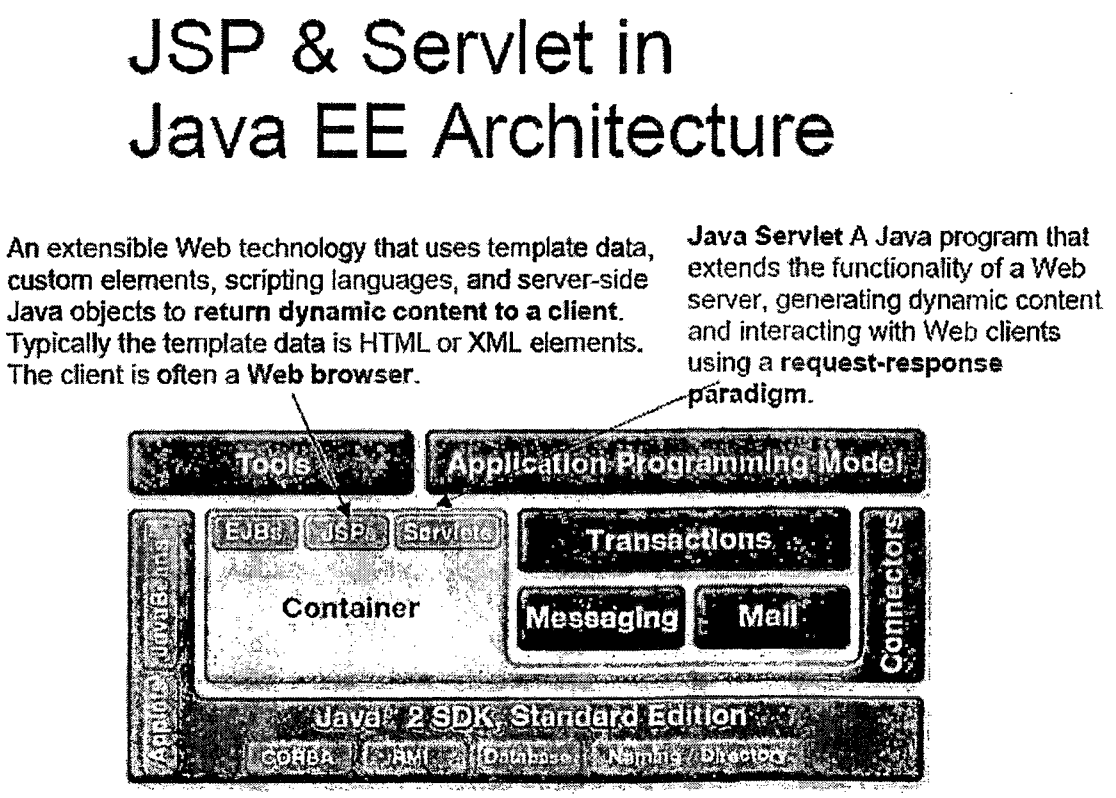
5.1.3. Τα πλεονεκτήματα των servlets.

Τα servlet προσφέρουν έναν αριθμό πλεονεκτημάτων, περιλαμβάνοντας: μεταφερσιμότητα, δύναμη, αποδοτικότητα, ασφάλεια, ενσωμάτωσης και ευελιξίας.

Τα κυριότερα πλεονεκτήματα πάνω στα περιβάλλοντα προγραμματισμού για server :

- Έχουν καλύτερη απόδοση επειδή μπορούν να παραμείνουν εσωτερικά και να διαχειριστούν πολλαπλά νήματα ταυτόχρονα.
- Απλότητα επειδή δεν απαιτούν κάποια εγκατάσταση λογισμικού στους χρήστες εκτός από έναν web browser.
- Αν και οι http servers δεν έχουν κάποια δυνατότητα για να θυμούνται λεπτομέρειες κάποιου προηγούμενου αιτήματος από τον ίδιο χρήστη, τα servlet χορηγούν ένα μέσο που παρακολουθεί την API που ξεπερνά αυτήν την οριοθέτηση .

Η πρόσβαση στη τεχνολογία της JAVA περιλαμβάνοντας τα νήματα, την δουλειά πάνω στα δίκτυα και την επικοινωνία με τις βάσεις δεδομένων



5.2.Java Server Page (JSP)

Η Java, έχει εξελιχθεί πολύ από την πρώτη της παρουσίαση από την Sun. Μετά τα applets που αποτέλεσαν τη βασική μέθοδο αξιοποίησης των (πολλών) δυνατοτήτων της γλώσσας, ακολούθησε το server side scripting με την τεχνολογία Java Server Pages και σήμερα η πλατφόρμα ωριμάζει, και πλέον προσφέρει ένα ολόκληρο περιβάλλον για την ανάπτυξη Enterprise εφαρμογών που αξιοποιούν πλήρως τις δυνατότητες που προσφέρει το Internet, την Java 2 Enterprise Edition (J2EE).

Η Sun παρουσίασε την τεχνολογία JSP για ν' ανταγωνιστεί την Microsoft στο χώρο των Server-side applications. Μια Java Server Page (JSP) είναι ένα πρότυπο για ιστοσελίδες που χρησιμοποιεί τον κώδικα της Java για να παράγει ένα έγγραφο HTML δυναμικά. Η JSP τρέχει σε ένα server-side component γνωστό ως JSP container, το οποίο μεταφράζει τις σελίδες σε Java Servlets. Την πρώτη φορά που έχουμε πρόσβαση σε ένα JSP μπορούμε να παρατηρήσουμε ότι παίρνει λίγο χρόνο για να ανταποκριθεί.

5.2.1. Πλεονεκτήματα - μειονεκτήματα

Η JSP χρησιμοποιεί component objects, tags που επιτρέπουν την ενσωμάτωση server side κώδικα στην HTML σελίδα, session tracking και διάδραση με βάσεις δεδομένων. Επίσης η JSP χρησιμοποιεί JavaBeans για να υλοποιήσει την component αρχιτεκτονική.

Η τεχνολογία JSP δεν "δένει" αναγκαστικά με κάποιο συγκεκριμένο Web server ή λειτουργικό, και εκτός Microsoft είναι ήδη ένα ευρέως αποδεκτό standard.

Ένας παράγοντας δυσκολίας είναι η εξοικείωση πολλών developers με τη χρήση των RAD εργαλείων της Microsoft (Visual Studio/InterDev), αλλά πλέον στην αγορά κυκλοφορούν καλύτερα περιβάλλοντα για Java /JSP/J2EE όπως είναι το NetBeans και το Sun StudioCreator.

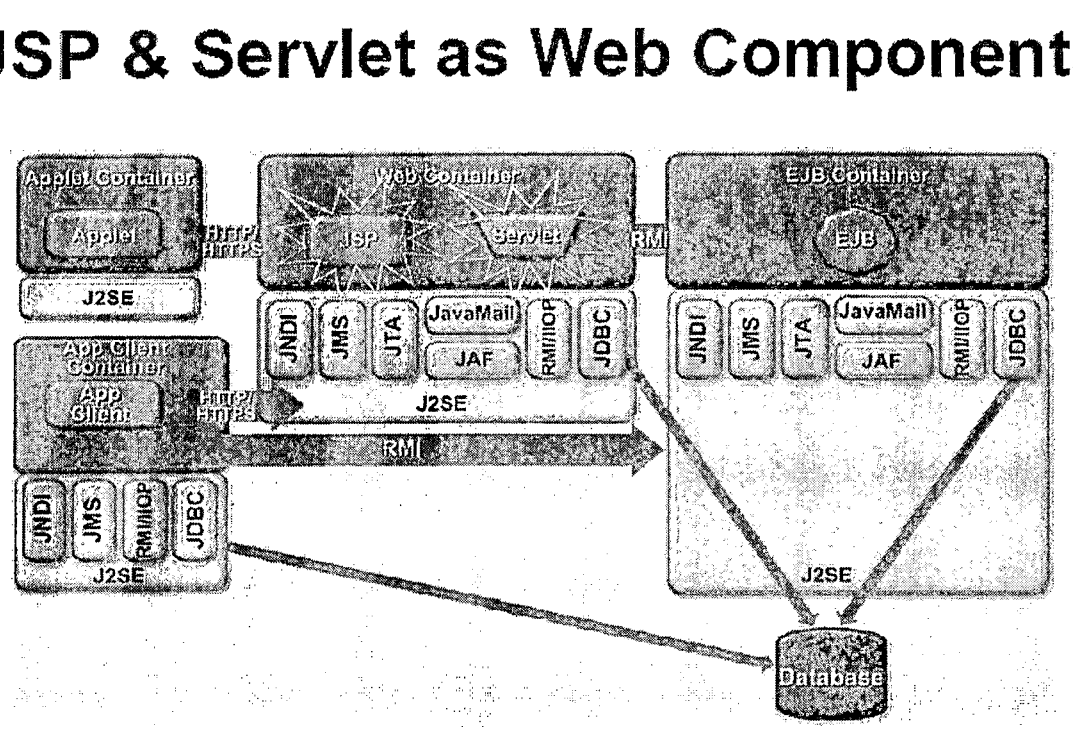
Ακόμα η αντικειμενοστραφής φύση της Java, προσθέτει ακόμα περισσότερο στο learning curve, για τους developers που δεν είναι εξοικειωμένοι με τον object oriented προγραμματισμό.

Η Java είναι μια γλώσσα με πολύ 'περισσότερες δυνατότητες από την Visual Basic/VBScript. Ένα ακόμα σημαντικό πλεονέκτημα της τεχνολογίας JSP είναι τα extensible tags. Με αυτό τον μηχανισμό, ο developer μπορεί να δημιουργήσει custom tags επιτρέποντας με αυτό τον τρόπο την επέκταση της tag γλώσσας των JSP.

5.2.2. Επέκταση / δυνατότητες

Η τεχνολογία JSP, είναι στην ουσία μια υλοποίηση του Servlet API της Sun. Ακριβώς γι αυτό το λόγο, αποτελεί κομμάτι της J2EE αρχιτεκτονικής, πράγμα που της επιτρέπει να χρησιμοποιεί όλα τα J2EE components όπως JavaBeans, Enterprise Java Beans components & Java servlets. Στην ουσία οι JSP σελίδες γίνονται compiled σαν servlets, οπότε έχουν όλα τα χαρακτηριστικά μιας Java server side εφαρμογής.
Επίσης μέσα από JSP σελίδες, έχουμε πρόσβαση σε όλα τα services της J2EE όπως JDBC για πρόσβαση σε βάσεις δεδομένων, JavaMail, Java Message Service (JMS) κ.α.

Επίσης οι JSP σελίδες μπορούν να επικοινωνούν με enterprise applications χρησιμοποιώντας CORBA-compliant τεχνολογίες όπως: Java IOL και RMI-POP.



5.2.3. Πως λειτουργεί

Ο κώδικας για μία JSP σελίδα, είναι στην ουσία HTML που περιέχει εκτός από τα γνωστά html tags και τα ειδικά JSP tags ή / και Java κώδικα σε JSP tags. Η κατάληξη του αρχείου είναι .jsp και αυτό ενημερώνει τον server ότι αυτή η σελίδα απαιτεί "ειδική μεταχείριση".

Η "ειδική" αυτή μεταχείριση, που επιτυγχάνετε μέσα από server extension ή plug-in (ανάλογα με τον server) και αποτελείται από 4 στάδια:

1. Η JSP engine κάνει parse τη σελίδα και δημιουργεί ένα Java source file.
2. Κάνει compile το java αρχείο σε java class. Το class αυτό είναι ένα Java servlet και από αυτό το σημείο η servlet engine το αντιμετωπίζει σαν τέτοιο
3. Η servlet engine φορτώνει το servlet προς εκτέλεση
4. Το servlet εκτελείται και επιστρέφει τι αποτελέσματα του στον web server που τα παρουσιάζει με τη μορφή απλού html κώδικα στον browser του χρήστη.

Παρότι η διαδικασία αυτή μπορεί να ακούγεται χρονοβόρα, είναι πιο αποτελεσματική απ' ότι φαίνεται. Τα δύο πρώτα στάδια εκτελούνται μόνο μία φορά, όταν κάνουμε deploy ή update τις σελίδες. Το 3° στάδιο εκτελείται μόνο στο πρώτο request για το servlet, από τη στιγμή που ξεκίνησε ο server. Μετά απ' αυτό ο class loader φορτώνει την class μια φορά και είναι διαθέσιμο για όλη τη "ζωή" της Java Virtual Machine. Επίσης, πολλοί application servers παρέχουν page caching, που μπορεί να βελτιώσει ακόμα περισσότερο την αποτελεσματικότητα. Με page caching, ακόμα και το 4° στάδιο μπορεί να εκτελεστεί μόνο μια φορά, πράγμα που εξαρτάται φυσικά από το πόσο δυναμικά είναι τα data που περιέχει η σελίδα.

5.2.4. Βασικά JSP tags

Οι JSP σελίδες μπορούν να περιέχουν πολλών ειδών tags, ακόμα και κάποια που ορίζει ο developer, όμως 4 είναι τα κύρια tags που αποτελούν το βασικό "οπλοστάσιο" για να ξεκινήσει κανείς την ανάπτυξη σελίδων JSP:

- Declaration tag (<%! %>) Το tag αυτό χρησιμοποιείται για την δήλωση μεταβλητών ή μεθόδων π.χ.: **<%! int k = o; %>**
- Specification tag (<% %>) Το tag αυτό χρησιμοποιείται για ενσωμάτωση Java κώδικα. π.χ. **<% for (í= O ; k < user_array.count; k++) { out.println(user_array[i]) ; }%>**
- Expression tag (<%= %>) Το tag αυτό χρησιμοποιείται για να εμφανίσουμε μία Java expression, κι είναι shorthand για την μέθοδο out.println()· π.χ **<%= user_array[2] %>**
- JavaBean tags (<jsp:useBean>, <jsp:setProperty>) Για την ενσωμάτωση JavaBeans. Το tag εντοπίζει τα beans μέσω των "scope" και "class" attributes που περιέχονται σε αυτό. Εάν δεν εντοπιστεί το bean, τότε δημιουργείται ένα νέο από την class που έχουμε ορίσει στο tag.

5.2.5. Εν κατακλείδι

Με αυτή την παρουσίαση των JSP απλά αγγίξαμε την επιφάνεια αυτής της τεχνολογίας. Οι σελίδες JSP είναι πολύ περισσότερα από απλές HTML με ενσωματωμένη Java. Αποτελούν ένα μόνο component της J2EE enterprise server αρχιτεκτονικής. Αξιοποιώντας αυτή τη τεχνολογία που περιλαμβάνει JSP, servlets, EJBs, JNDI, XMI, κ.α. γίνετε αντιληπτό πως αποτελεί μια πολύ ανταγωνιστική αρχιτεκτονική για τη δημιουργία distributed applications και σίγουρα αποτελεί μια καλή εναλλακτική λύση γι αυτούς που δεν θέλουν να περιοριστούν στις αρχιτεκτονικές της Microsoft.

5.3.Τι Είναι Ένα Java Bean

Ένα java bean είναι 100% συστατικό της java που δουλεύει σε κάθε java virtual machine. Με τη χρήση αυτής της τεχνολογίας ο προγραμματιστής μπορεί να γράψει έναν «κόκκο» (Bean), που είναι ένα κομμάτι επαναχρησιμοποιημένου κώδικα (reusable code), το οποίο σε άλλες περιπτώσεις έχει εξωτερική μορφή γραφικών και σε άλλες όχι.

Με την τεχνολογία Java Beans η Sun θέλησε να δημιουργήσει μια αρχιτεκτονική βασισμένη σε συστατικά (component base) για τη δημιουργία, οπτικών η μη, τμημάτων επαναχρησιμοποιημένου κώδικα για την κατασκευή εφαρμογών.

Όταν προσθέτουμε και άλλο java κώδικα σε μια jsp σελίδα γίνεται όλο και ποιο δύσκολο για να διαχειριστεί κάποιος την σελίδα. Είναι ακόμα ποιο δύσκολο όταν ο κώδικας java μπερδευτεί με την html. ο τρόπος με τον οποίο μπορούμε να κρατήσουμε την java ξεχωριστή είναι να χρησιμοποιήσουμε java bean.
Τα παρακάτω είναι οι ελάχιστες απαιτήσεις που ένα συστατικό σε java bean :

1. Πρέπει οι constructor να έχουν μηδενική αντιδικία(conflict).
2. Πρέπει να χρησιμοποιεί getters και setters για να εκθέτει τις ιδιότητες.
3. Πρέπει να υποστηρίζει το JDK

Με αυτούς τους περιορισμούς, οι πολλές κλάσεις είναι ήδη «κόκκοι» (beans) η μπορούν με λίγη προσπάθεια να μετατραπούν σε bean . Δεν χρειάζεται ένα ξεχωριστό περιβάλλον για να τρέχουν τα bean εκτός από κάποια Java Virtual Machine.

Για να χρησιμοποιήσουμε ένα Java Bean , πρώτα, πρέπει να καταλάβουμε τις ιδιότητες του. Οι πιο πολλοί κόκκοι έχουν ιδιότητες.
Ένα Bean περιέχει :

- Γεγονότα
- Ιδιότητες
- Μεθόδους

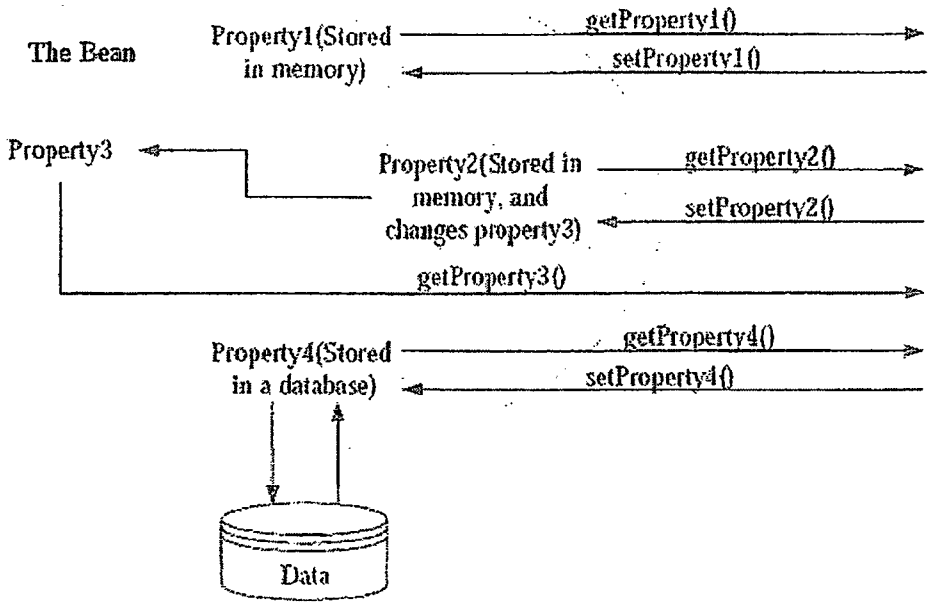
Τα γεγονότα (events) είναι αντικείμενα που χρησιμοποιούνται για να δηλώσουν μια αλλαγή στην κατάσταση ενός αντικειμένου-πηγής του γεγονότος.

Όταν συμβεί ένα γεγονός σε έναν bean, παράγεται ένα αντικείμενο γεγονότος το οποίο ειδοποιεί μέσω της κατάλληλης μεθόδου τον ακροατή που έχει καταχωρηθεί για να παρακολουθεί το γεγονός.

Οι ιδιότητες που καθορίζουν τα χαρακτηριστικά ενός bean είναι κάτι ανάλογο με τις ιδιότητες ενός τυπικού συστατικού Java π.χ. ενός button . Ορισμένες από τις ιδιότητες αυτές αντιπροσωπεύουν ορατά του κόκκου, ενώ άλλες μη ορατά.

Για να δημιουργηθεί ή να ανακτηθεί μια ισότητα χρησιμοποιούνται μέθοδοι getters και setters όπως οι παρακάτω:
Public void **set** < όνομα_ιδιότητας> (<τύπος> < τιμή>);
Public < τύπος> **get** < ονομα_ιδιοτητας > ( ) ;
Μια ιδιότητα μπορεί να είναι μόνο για εγγραφή (λείπει η μέθοδος get), μόνο για ανάγνωση (λείπει η set) , ή για εγγραφή και ανάγνωση (υπάρχουν και οι δυο μέθοδοι).

Ένα μοντέλο Java Bean φαίνεται στο παρακάτω σχήμα.



Μοντέλο ενός Java Bean

5.3.1. Πλεονεκτήματα των Java Beans

Ένα πλεονέκτημα των Java Bean είναι ότι δεν χρειάζεται να ξέρεις κάτι από πριν (εκτός από Java) για να τα χρησιμοποιήσεις.

Άλλο ένα είναι ότι δεν χρειάζεται να κάνεις κάτι για να αναζητήσεις η να αλλάξεις μια ιδιότητα, απλά να ψάξεις κάποιες πληροφορίες σε μια βάση δεδομένων.

Επίσης σύμφωνα με την γνώμη πολλών ειδικών, η τεχνολογία JavaBeans υπερτερεί μια και οι ανάλογες τεχνολογίες της Microsoft λειτουργούν *κυρίως* στα WINDOWS ή *μόνο* στα WIINDOWS.

Μαγκαφάς Γιώργος
ΑΜ 10590

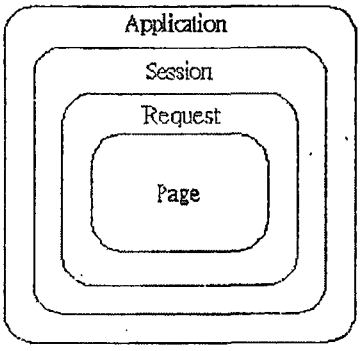
5.4.Java Server Page και Java Beans

Για να χρησιμοποιήσεις JavaBeans μέσα σε JSP, οι προδιαγραφές του JSP απαιτούν τρία tags: ένα είναι να εντοπίσει η να δημιουργήσει ένα Java Bean , άλλο ένα είναι να θέσει(set) μια ιδιότητα σε κάποιο Java Bean και τέλος να πάρει (get) μια ιδιότητα από ένα Java Bean .

Purpose	Syntax
To load a JavaBean	<jsp:useBean id="name" scope="page|request|session|application" class="className" />
To set a property	<jsp:setProperty name="beanName" prop_expr />
To get a property	<jsp:getProperty name="beanName" property="propertyName" />

Σύνταξη για τη χρησιμοποίηση ενός JavaBean σε JSP

Οι ιδιότητες πεδίου για τη φόρτωση ενός JavaBean καθορίζει το αντικείμενο με το οποίο εμείς συνδέουμε το bean. Αυτό το πεδίο του JavaBean φαίνεται στο σχήμα



Το πεδίο ενός JavaBean

1. Page: Έγκυρη όσο διαρκεί αυτή η σελίδα JSP. Αυτό είναι το πεδίο προεπιλογής.
2. Request: Έγκυρη για το υπόλοιπο της σελίδας JSP και για οποιουσδήποτε άλλους πόρους συντήρησης αυτού του αιτήματος.
3. Session: Έγκυρη κατά τη διάρκεια της εκτέλεσης οποιασδήποτε σελίδας JSP ή Java Servletσε αυτή το HTTP session.
4. Application: Έγκυρη σε οποιαδήποτε σελίδα JSP ή Java Servlet σε αυτήν την WEB εφαρμογή.

Μαγκαφάς Γιώργος
ΑΜ 10590

6. SERVERS

Ένας server χρησιμοποιείται για να εξυπηρετήσει τα έγγραφα στους clients. Μια μελέτη που πραγματοποιήθηκε από την NetCraft (http://www.netcraft.com/survey/) δείχνει ότι οι τρεις δημοφιλέστεροι servers δικτύου στο διαδίκτυο είναι ο Apache Tomcat, ο οποίος είναι ο νούμερο ένα , ο Microsoft IIS και ο iPlanet Netscape.

6.1. IPlanet

Ο IPlanet χρησιμοποιεί την Sun Java 2 Platform, Enterprise Edition (J2EE), σαν βάση για τα JSP/EJB based server. Είναι σχεδιασμένος να είναι ιδιαίτερα αυξομειώσιμος – εξελικτικός , προσφέρει υποστήριξη για διαδικασίες υψηλής αξιοπιστίας και μπορεί να ενσωματωθεί με πολλές εφαρμογές .

Σύμφωνα με την Sun το πλεονέκτημά του απέναντι στον Apache Tomcat είναι ότι τρέχει πραγματικά γρήγορα και δεν συντρίβει (crash).

6.2. Windows IIS Server

Ο Microsoft's IIS server είναι το βασικό component για τη δημιουργία μιας εφαρμογής διαδικτύου ή ενδοδικτύου για τα Windows NT Server, Windows NT Workstation and Windows 95. Ο IIS υποστηρίζει τυποποιημένα πρωτόκολλα πληροφοριών, και είναι επεκτάσιμος σε μεγάλο βαθμό μέσω του Common Gateway Interface (CGI).

Ο IIS είναι πλήρως ενσωματωμένος με τα Windows NT Server 4.0. Με αυτή τη σφιχτή ενσωμάτωση του Internet Information Server και των Windows NT Server, μπορείς να εκμεταλλευτείς την ισχυρή ασφάλεια που προσφέρουν (built in security) τα Windows NT Server και το Windows NT File Systems (NTFS).

Μπορείτε να χρησιμοποιήσετε τις πρόσθετες τεχνολογίες Διαδικτύου που παρέχονται από τον IIS για να εμπλουτίσετε τη Microsoft BackOffice. Η Microsoft BackOffice περιλαμβάνει:

- Microsoft SQL server client/server database management system.
- Microsoft Exchange Server client/server messaging system.
- Microsoft Proxy Server.
- Microsoft SNA Server connectivity for IBM enterprise networks.
- Microsoft System Management Server centralized management for distributed systems.
- Microsoft Commercial Internet Server (MCIS).

Με τον IIS μπορείτε να αναπτύξετε εφαρμογές server για να φιλοξενήσετε την πιο πρόσφατη ύλη του Web. Ο IIS υποστηρίζει πλήρως το Microsoft Visual Basic programming system, VB Script, Microsoft Jscript development software, και τα Java components. Υποστηρίζει επίσης τα CGI applications Web based προγράμματα , και

ISAPI επεκτάσεις και φίλτρα (Microsoft Press, 1998). Είναι καλύτερο να χρησιμοποιήσετε την ASP με τον IIS καθώς ο IIS έρχεται με την πλήρη υποστήριξη για την ASP. Η PHP θα λειτουργήσει καλά σε IIS αλλά μερικές από τις πιο προηγμένες αλλά λιγότερο χρησιμοποιούμενες λειτουργίες μπορούν να είναι απρόβλεπτες ή μπορούν να μην δουλέψουν καθόλου.

6.3. Apache Tomcat

Ο Tomcat είναι μέρος του Apache Software Foundations Jakarta Project. Δημιουργήθηκε το 1995 όταν μια ομάδα εθελοντών ανέπτυξε ένα ευρέως supported web server. Είναι μια πρωτοβουλία ανοικτού κώδικα μέσα από το Linux, επιτρέποντας στον καθένα να συμβάλει στο τελικό προϊόν εάν θέλει.

Ο Apache Tomcat δεν έχει GUI κονσόλα διαχείρισης και αυτό τον κάνει λιγάκι πιο δύσκολο να τον χρησιμοποιήσεις από άλλους servers. Ολόκληρη η διαμόρφωση γίνεται μέσω μιας σειράς αρχείων διαμόρφωσης (configuration files).

Το πλεονέκτημα του Tomcat είναι ότι θα τρέξει σχεδόν με οτιδήποτε νέα έκδοση του java development kit (JDK) και μια σύνδεση δικτύου μέχρι και Multiprocessor Solaris Servers. Αυτό οφείλετε στη φιλοσοφία της java "write once". Αυτό επιτρέπει να διαμορφώσετε την πλατφόρμα σας όσο αυξάνονται οι ανάγκες.

6.4. Συμπέρασμα

Για την δημιουργία της εφαρμογής έχει επιλεχθεί ο Apache Tomcat για το λόγο ότι είναι σχεδιασμένος με τεχνολογία ανοιχτού κώδικα και κατ'επέκταση είναι δωρεάν και το ότι έρχεται ενσωματωμένος με το NetBeans IDE 5.5 Visual Web.
Ο Apache Tomcat είναι επίσης συμβατός με PHP, JSP και MYSQL έτσι και για αυτούς τους λόγους αποφασίστηκε η χρησιμοποίηση του.

Μαγκαφάς Γιώργος
ΑΜ 10590

7. DATABASES

Μια βάση δεδομένων είναι μια συλλογή από στοιχεία που οργανώνονται έτσι ώστε το περιεχόμενό τους να είναι προσβάσημο να μπορεί εύκολα να ρυθμιστεί, και να ενημερωθεί. Ο επικρατέστερος τύπος βάσης δεδομένων είναι η σχεσιακή βάση δεδομένων, μια συνοπτική βάση δεδομένων στην οποία τα στοιχεία καθορίζονται έτσι ώστε να μπορούν να αναδιοργανωθούν και να προσεγγιστούν με διάφορους τρόπους.

7.1. Microsoft Access

Από την εισαγωγή της MS – Access στην αγορά των βάσεων δεδομένων τον Οκτώβριο του 1992, εκατομμύρια χρηστών παγκοσμίως έχουν αγκαλιάσει αυτό το εύκαμπτο και εύχρηστο προϊόν, και το έχουν κάνει ένα από τα εξέχων συστήματα διαχείρισης βάσεων δεδομένων.

Τα πλεονεκτήματα της Access είναι ότι είναι εύκολο να την μάθεις να την χρησιμοποιήσεις. Είναι γρήγορη στο να εξετάσει, να διορθώσει και να εκσφαλματώσει (debug) και έχει ένα απλό GUI, το οποίο το καθιστά εύκολο να δημιουργήσει τους πίνακες.

Ένα μειονέκτημα της Access είναι ότι δεν παρέχει πολύ καλή υποστήριξη για μεγάλους αριθμούς χρηστών.

7.2. Oracle

Η εταιρία Oracle είναι ο παγκόσμιος μεγαλύτερος προμηθευτής λογισμικού. Η Oracle ανέπτυξε το πρώτο εμπορικό σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων SQL το 1979. Το 1997 κυκλοφόρησε το Oracle 8 και περιέλαβε βελτιωμένη εξελιξιμότητα, απόδοση και αντικειμενοστραφείς ικανότητες. Στην Oracle 8i του 1999 εισήχθηκαν χαρακτηριστικά γνωρίσματα για μια πλήρη Διαδικτυακή βάση δεδομένων.

Η Oracle είναι μια ασφαλής, αξιόπιστη και εύκαμπτη βάση δεδομένων. Η Oracle αποδίδει πολύ καλά στις διάφορες πλατφόρμες. Αυτό επιτρέπει την ενοποίηση μεταξύ των διάφορων βάσεων δεδομένων. Ο Oracle server για το σύστημα διαχείρισης βάσης δεδομένων της είναι διαθέσιμο για τους περισσότερους τύπους υπολογιστών.
Τα μειονεκτήματα της Oracle είναι ότι αναπτύχθηκε για επαγγελματική χρήση ως εκ τούτου η καμπύλη εκμάθησης είναι απότομη. Η Oracle είναι εξαιρετικά ακριβή και γενικά οι περισσότεροι χρήστες (home users) δεν θα είχαν πρόσβαση σε έναν server Oracle.

7.3. MYSQL

Πολλές σελίδες δεν χρειάζονται την δύναμη και την τιμή που έχει η Oracle. Η MYSQL είναι μια βάση δεδομένων SQL ανοικτού κώδικα διαθέσιμη για τον καθένα να την χρησιμοποιήσει, με πολλά από τα χαρακτηριστικά γνωρίσματα της Oracle, είναι το πιο γνωστό σύστημα διαχείρισης βάσεων δεδομένων (ανοιχτού κώδικα), αναπτύχθηκε, διανέμεται και υποστηρίζεται από την MySQL AB.

Η MYSQL είναι διαθέσιμη για οποιοδήποτε υπολογιστή που έχει μια αξιοπρεπή δύναμη και είναι πολύ εύκολο να εγκατασταθεί (Turner 2002). Μπορείτε να κατεβάσετε το MYSQL GUI TOOLS που είναι ένα πολύ χρήσιμο εργαλείο κατά τη δημιουργία των πινάκων και την εργασία μαζί τους. Η MYSQL και ο Apache Tomcat πηγαίνουν σχεδόν μαζί που είναι καλό, δεδομένου ότι θα χρησιμοποιούμε Tomcat.

Ξεκινώντας υπήρχε η πρόθεση να χρησιμοποιηθεί το σύστημα βάσεων δεδομένων **msql** για να ενωθούν πίνακες χρησιμοποιώντας κάποιες ρουτίνες . Όμως μετά από κάποιο πειραματισμό; καταλήξανε στο συμπέρασμα ότι η **msql** δεν ήταν γρήγορη αρκετά ούτε ευέλικτη αρκετά για της ανάγκες που υπήρχαν.

Ο προσδιορισμός του ονόματος MySQL δεν είναι ξεκάθαρος. Τα βασικά αρχεία και πολλές βιβλιοθήκες που υπάρχουν έχουν το πρόθεμα my για πάνω από 10 χρόνια. Όμως η κόρη του συν ιδρυτή Monty Widenius ονομάζεται και αυτή MY. Τι από τα δυο έδωσε το όνομα της στη MySQL παραμένει ακόμα ένα μυστήριο.

7.3.1. Χαρακτηριστικά

Οι παρακάτω παραθέτουμε κάποια από τα πιο σημαντικά χαρακτηριστικά της MySQL :

- Είναι γραμμένη σε C και C++
- Είναι δοκιμασμένη με μια μεγάλη γκάμα διαφορετικών compilers
- Δουλεύει σε πολλές διάφορες πλατφόρμες
- Χρησιμοποιεί GNU Automake, Autoconf, και Libtool για τη μεταφερσιμότητα οι Πλήρως πολύ-επεξεργαστική χρησιμοποιώντας πυρήνα νημάτων
- Πολύ γρήγορο σύστημα μνήμης
- Εύκολο να προστεθούν άλλες συσκευές αποθήκευσης
- Ο κώδικας της MySQL είναι δοκιμασμένος σε πολλά εργαλεία

7.3.2. Πλεονεκτήματα

Τα πλεονεκτήματα της MYSQL είναι:

- Ισχυρή και αποδοτική
- Υλοποιεί Τυποποιημένα χαρακτηριστικά γνωρίσματα SQL
- Ελεύθερα διαθέσιμη, ανοικτού κώδικα
- Ευρεία, ενθαρρυντική κοινότητα χρηστών(user community)
- Ασφάλεια
- Συνδεσιμότητα
- MySQL Database Server είναι πολύ γρήγορος, αξιόπιστος και εύκολος στη χρήση
- Ο MySQL Server δουλεύει σε συστήματα που βασίζονται στο μοντέλο πελάτη / εξυπηρετητή (Client/Server)
- Πολλά προγράμματα σχετικά με την MySQL είναι διαθέσιμα.

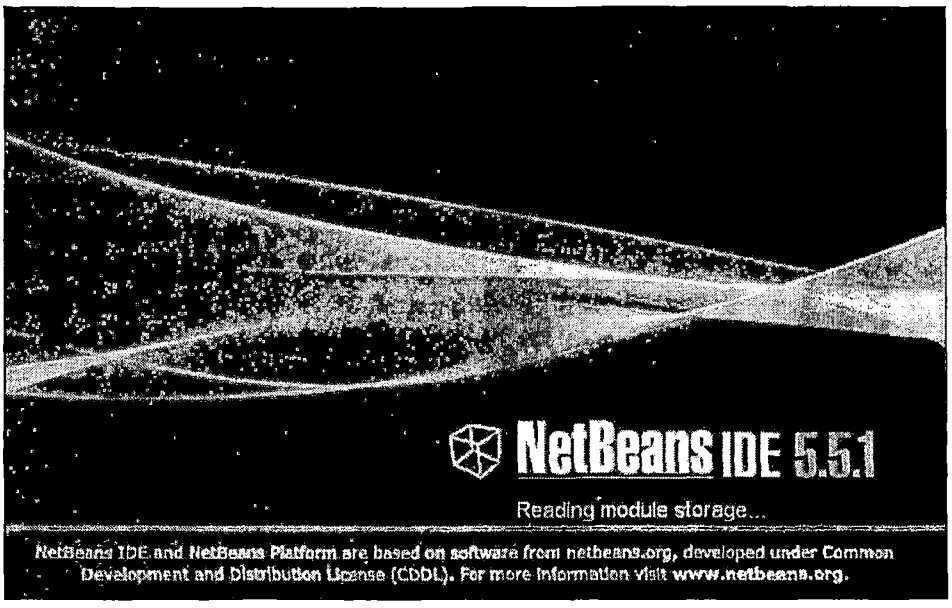
Τα μειονεκτήματα MYSQL είναι ότι δεν έχει της λειτουργίες Rollback και Commit . Η Rollback επιτρέπει σε σας να θέσετε ένα ασφαλές σημείο στη βάση δεδομένων πριν αρχίσετε να κάνετε μια σειρά από συναλλαγές, και να είστε σε θέση είτε να επιστρέψετε στην αρχική κατάσταση (Rollback) είτε να δεσμεύσετε τις αλλαγές στο τέλος (Commit).

7.4. Συμπέρασμα αξιολόγησης Database

Ιδανικά θα επιθυμούσαμε να χρησιμοποιήσουμε Oracle, έχει πολύ περισσότερα χαρακτηριστικά γνωρίσματα. Το πρόβλημα είναι ότι λόγω του ότι είναι πολύ ακριβή είναι σχεδόν αδύνατο να την αποκτήσουμε. Έτσι γι αυτό τον λόγο έχουμε αποφασίσει να μην τη χρησιμοποιήσουμε. Η Access αφ' ετέρου είναι ευρέως διαθέσιμη αλλά δυστυχώς δεν θα ταίριαζε σε αυτό το πρόγραμμα επειδή Access προκαλεί πολλά προβλήματα όταν προσπαθεί ένας μεγάλος αριθμός χρηστών να έχει πρόσβαση συγχρόνως. Έτσι όλα αυτά μας οδηγούν στη MYSQL. Θεωρούμε ότι αυτή είναι η καλύτερη επιλογή μας δεδομένου ότι είναι τεχνολογία ανοικτού κώδικα και είναι εύκολο να εγκατασταθεί. Έχει επίσης αρκετά χαρακτηριστικά γνωρίσματα για τη χρήση της.

8. Περιβάλλον Σχεδίασης και Υλοποίησης Εφαρμογής

8.1. NetBeans IDE 5.5.1

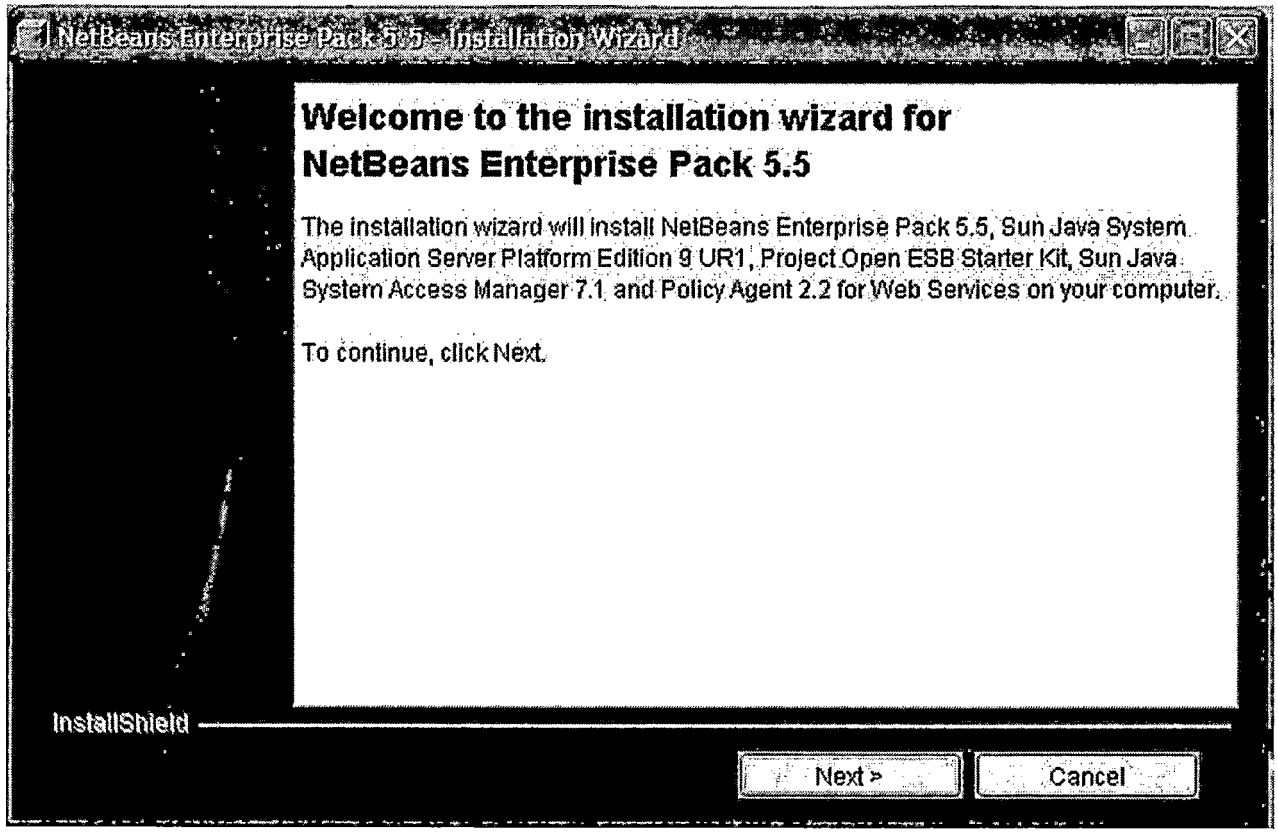


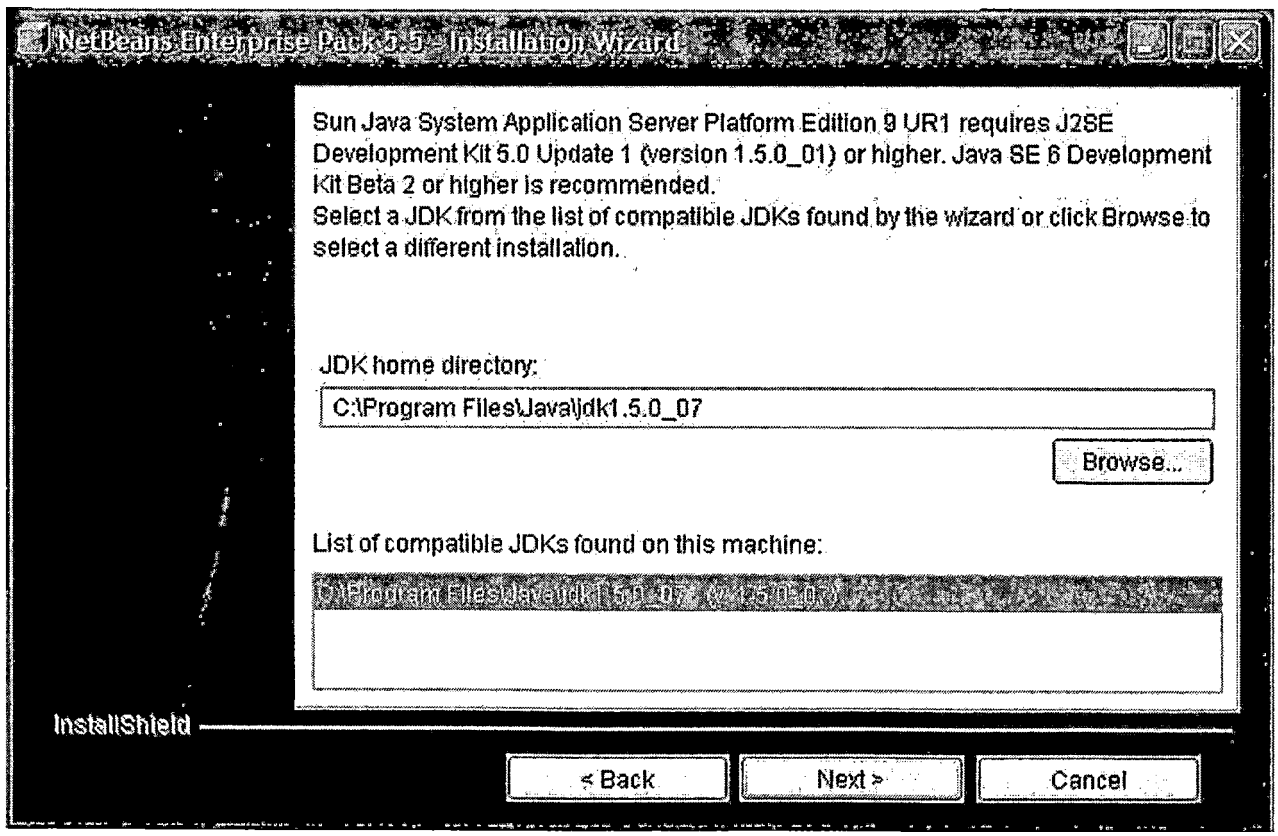
Το **NetBeans IDE 5.5.1** είναι ένα από τα πιο σύγχρονα περιβάλλοντα ανάπτυξης λογισμικού και δημιουργίας εφαρμογών. Είναι ελευθέρου κώδικα και το υποστηρίζουν μια γκάμα από πολλές εταιρίες δημιουργώντας μια σειρά από δικά τους components που το κάνουν ακόμα πιο ανταγωνιστικό. Με την μεταφερσιμότητα του λειτουργεί και σε άλλες πλατφόρμες όπως Linux, Unix, Mac OS

Η στάνταρ έκδοση του NetBeans έρχεται χωρίς το σχεδιαστικό περιβάλλον το οποίο πρέπει να εγκατασταθεί ξεχωριστά και μετά την εγκατάσταση της στάνταρ έκδοσης.

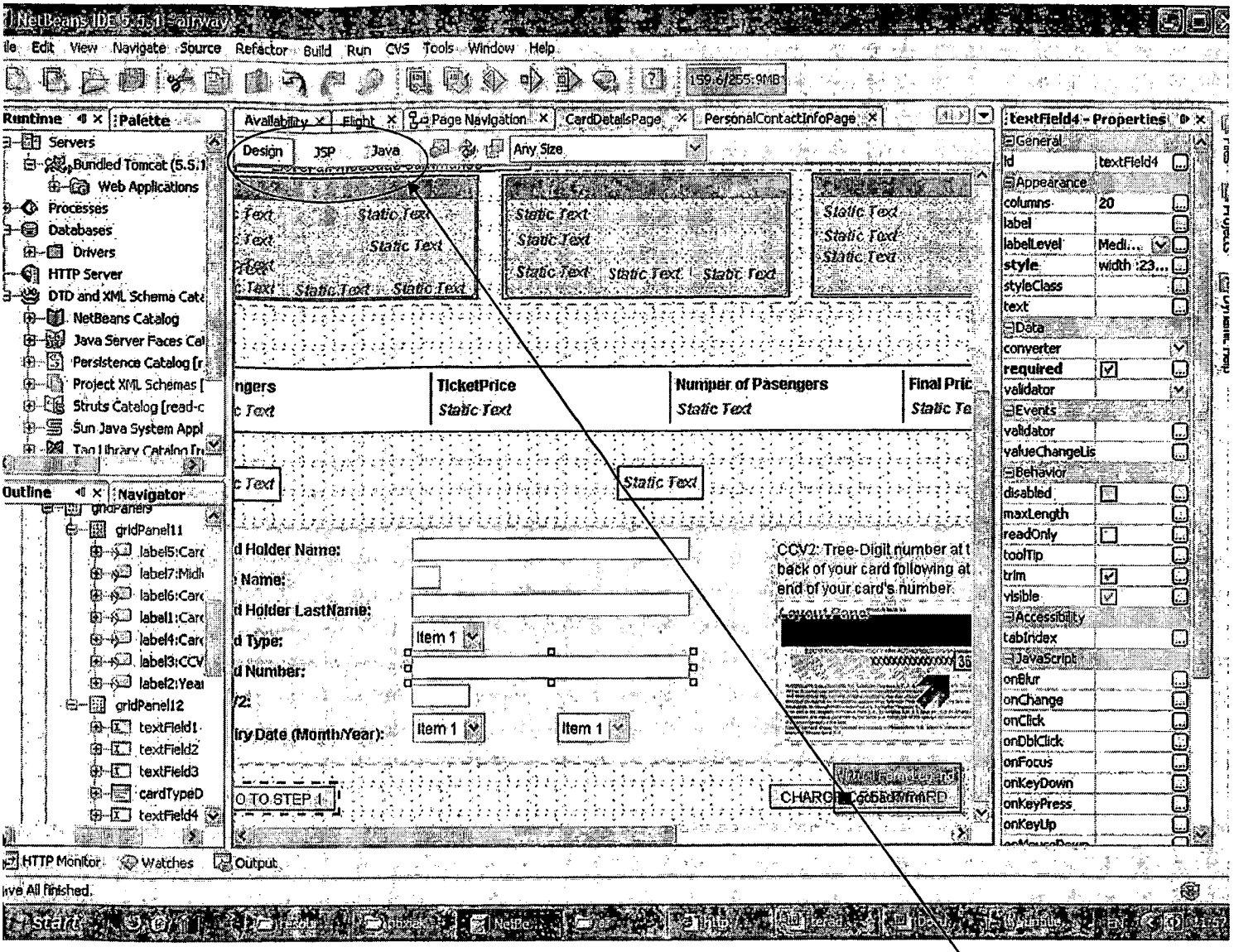
Για την εγκατάσταση του χρειάζεται να έχουμε ήδη εγκαταστήσει στο σύστημά μας τα Java Development Kit(JDK) και Java Runtime Environment(JRE).

Το J2SE Runtime Environment περιέχει την Java virtual machine, runtime class libraries, και το Java application launcher τα οποία είναι αναγκαία για έχουμε την δυνατότητα να τρέχουμε προγράμματα τα οποία είναι γραμμένα σε Java. Από την άλλη το J2SE Development Kit περιέχει development tools όπως compilers και debuggers αναγκαία κατά την δημιουργία της εφαρμογής μας.





Μαγκαφάς Γιώργος
ΑΜ 10590

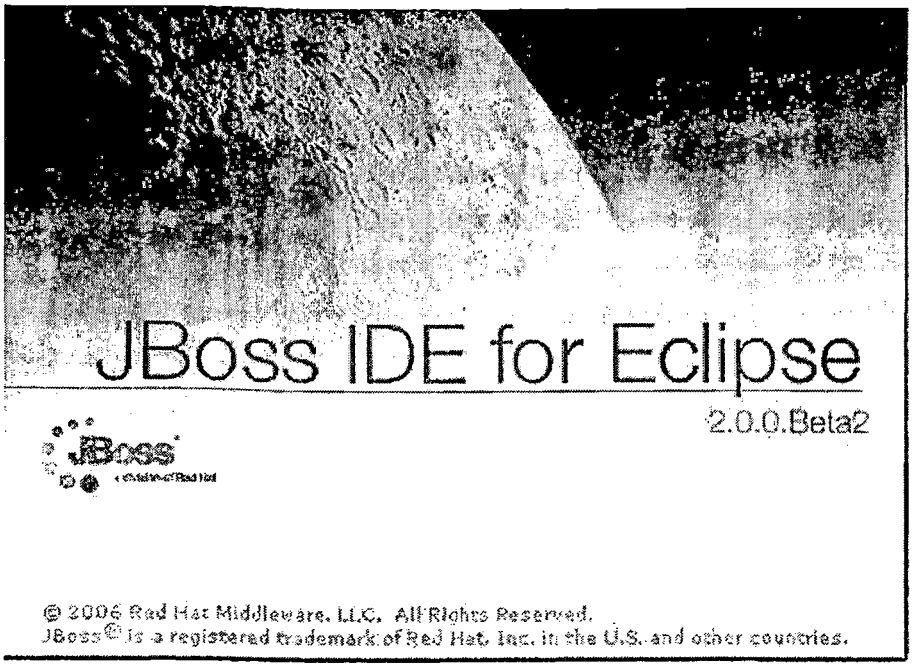


Το περιβάλλον σχεδίασης του NetBeans 5.5.1 ένα αρκετά φιλικό και εύκολο περιβάλλον προς τον χρήστη.

Με κουμπιά άμεσης πρόσβασης προς τους κώδικες Java και JSP αλλά και του Design

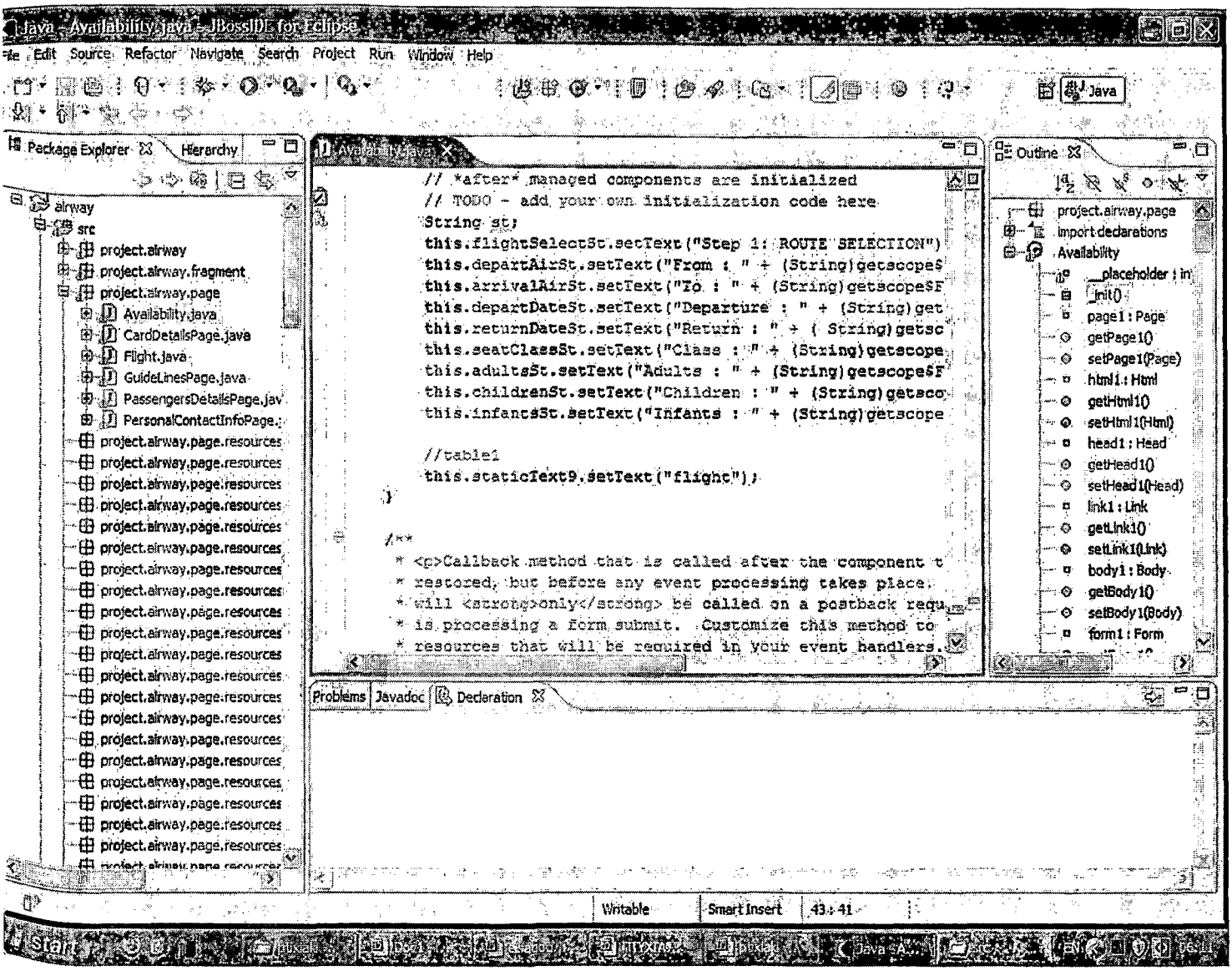
Μαγκαφάς Γιώργος
ΑΜ 10590

8.2. Eclipse με JBoss.



Το Eclipse είναι και αυτό περιβάλλον ανάπτυξης εφαρμογών χωρίς όμως το σχεδιαστικό περιβάλλον που έχει το NetBeans.

Η κύρια χρησιμοποίηση του κατά την διάρκεια αυτής της πτυχιακής εργασίας ήταν για την μετατροπή των πινάκων της Βάσης Δεδομένων του project σε οντότητες (class) και των πεδίων των πινάκων σε objects μέσο του Hibernate mapping.



Το περιβάλλον εγγραφής κώδικα του Eclipse.

Μαγκαφάς Γιώργος
ΑΜ 10590

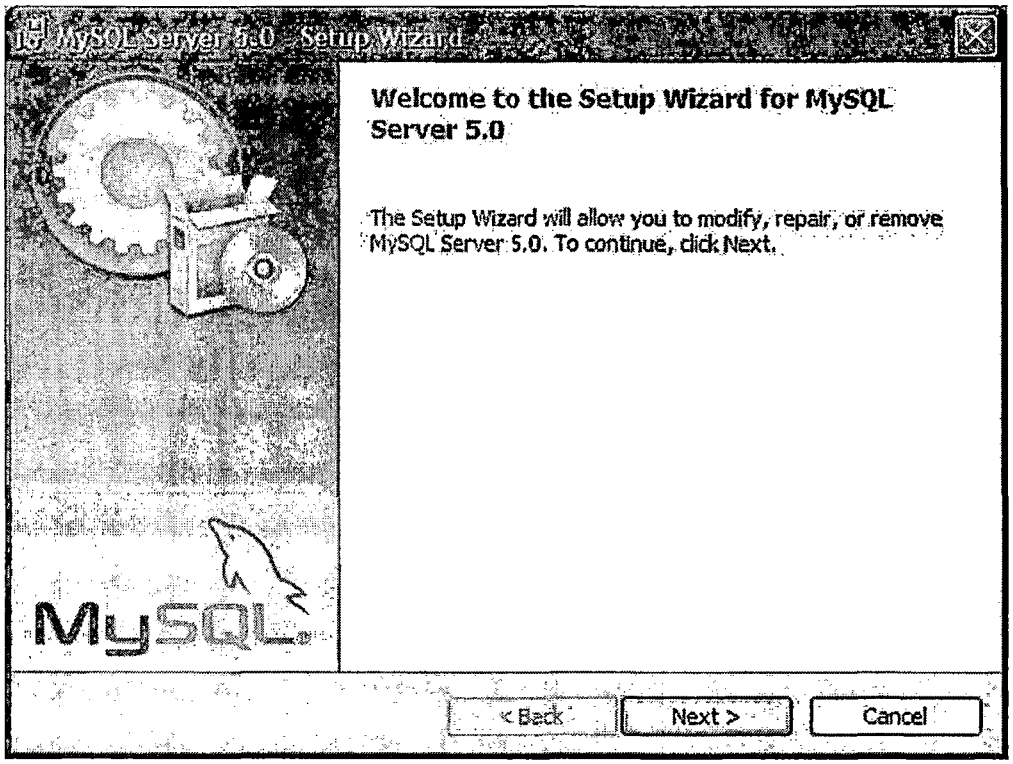
9. Βάση Δεδομένων

Η βάση δεδομένων αρχικά δημιουργήθηκε σε Microsoft Access αλλά λόγω των γνωστών προβλημάτων που παρουσιάζει αποφασίστηκε να μετατραπεί σε MySQL.

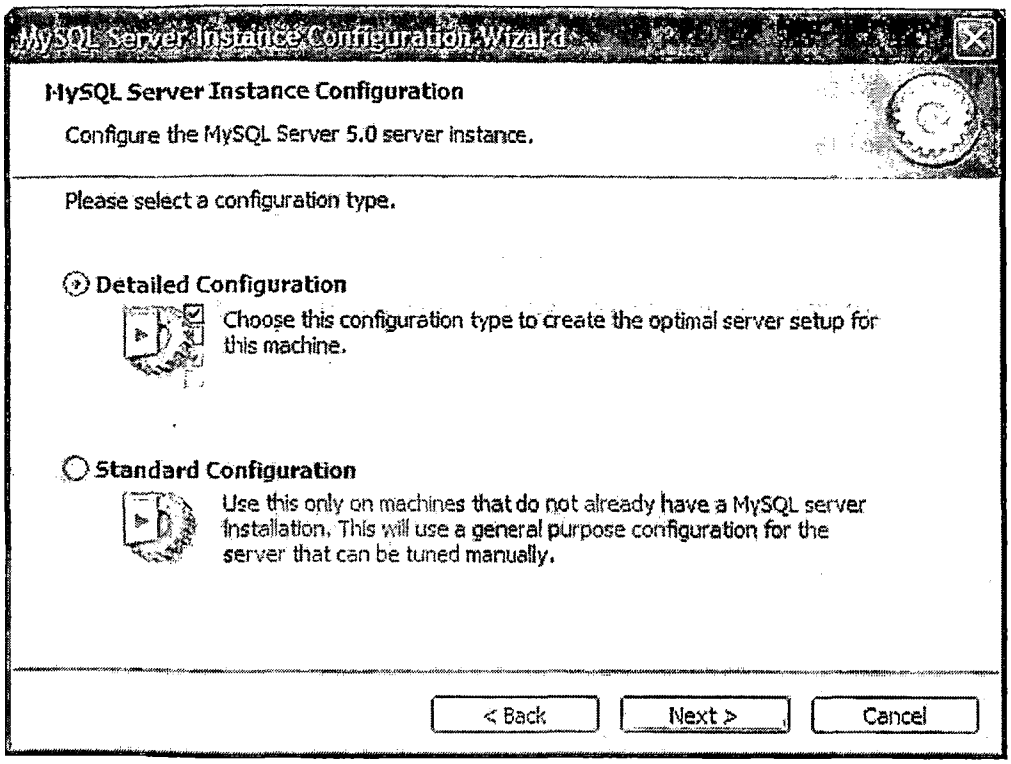
Η διαδικασία της μετανάστευσης (migration) έγινε με ένα από τα πολλά εργαλεία που υποστηρίζει η MySQL το MySQL Migration Tool.

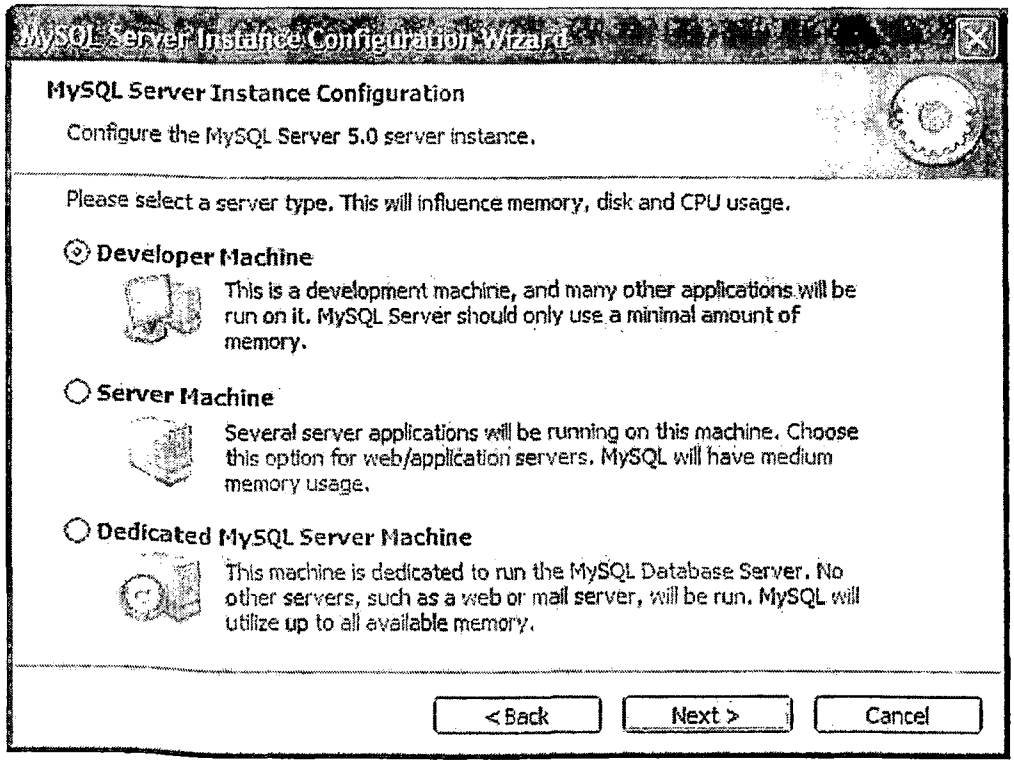
Πρώτα απ' όλα πρέπει να εγκαταστήσουμε την MySQL στο σύστημά μας.

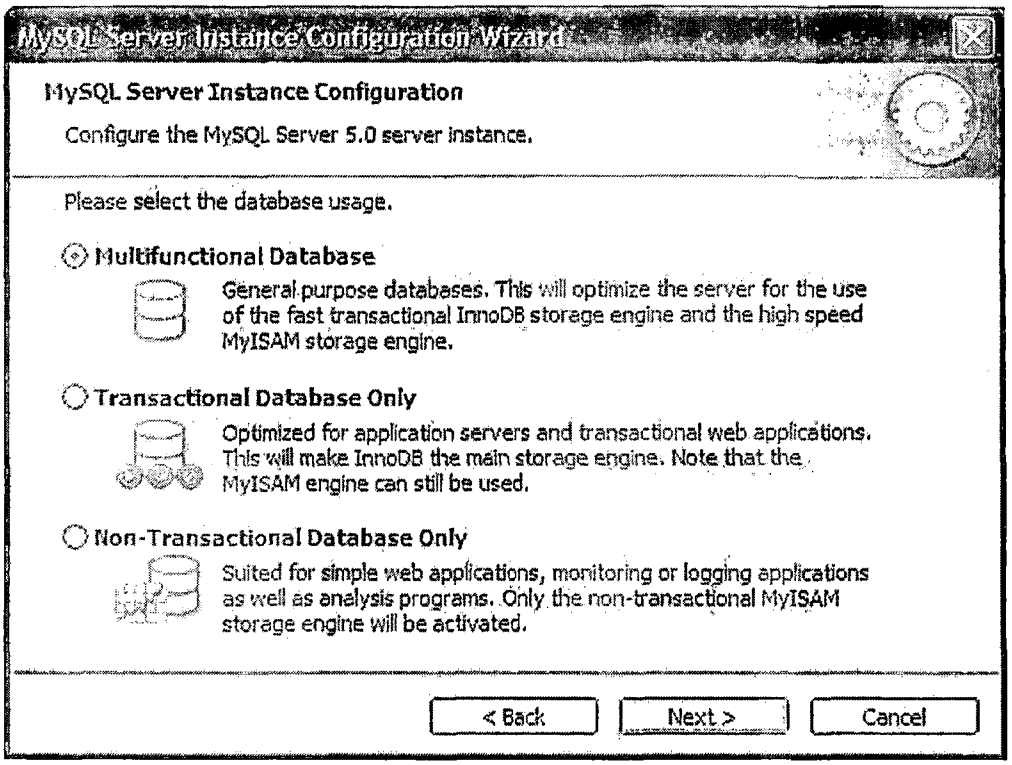
9.1. Εγκατάσταση MySQL.

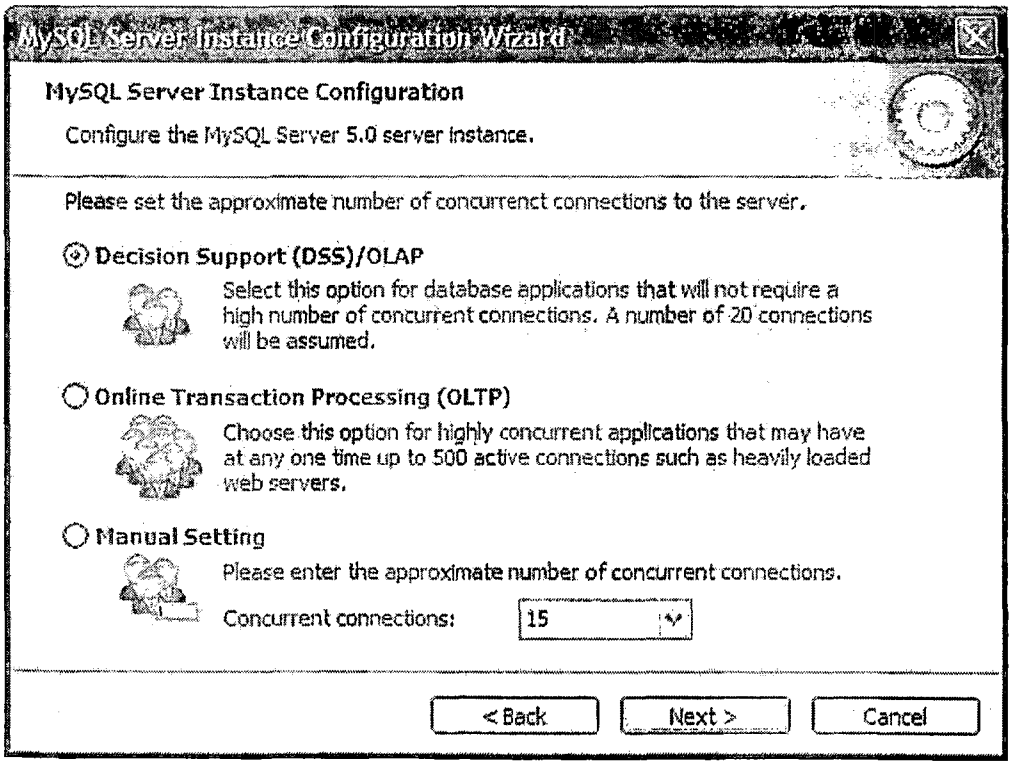


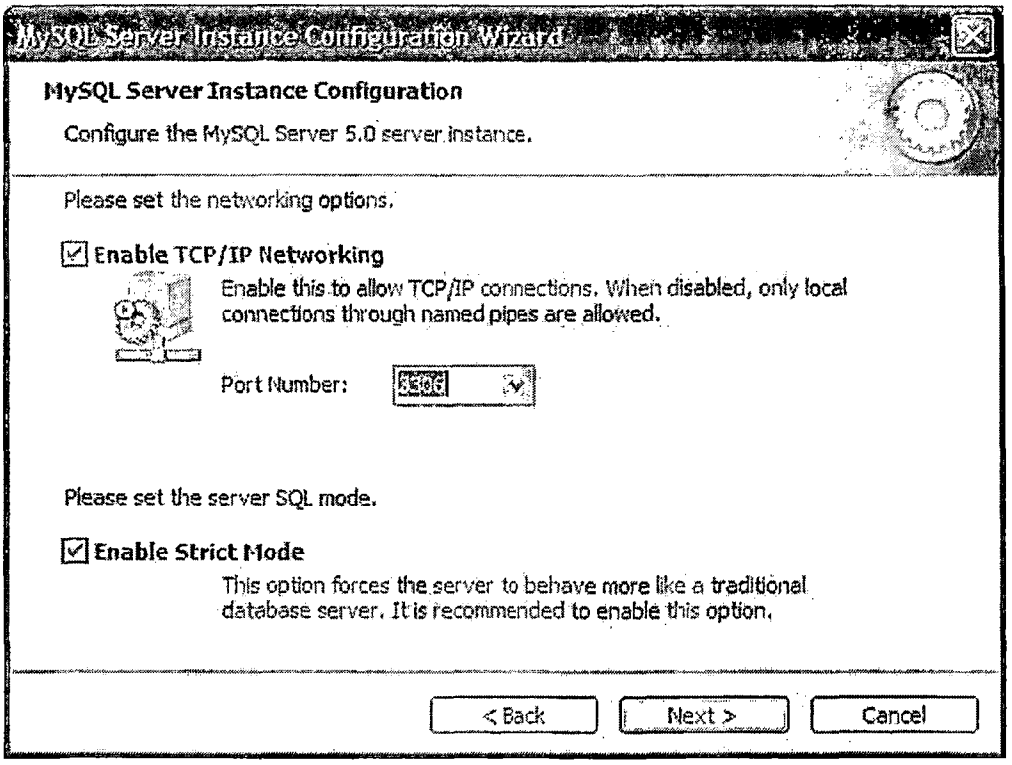
Το ίδιο το πρόγραμμα θα σας καθοδηγήσει για την σωστή εγκατάσταση της MySQL. Ακολούθως θα πρέπει να γίνουν οι απαραίτητες ρυθμίσεις για τη σωστή λειτουργία της MySQL σύμφωνα με τις ανάγκες που έχει η εφαρμογή που θα δημιουργήσουμε.

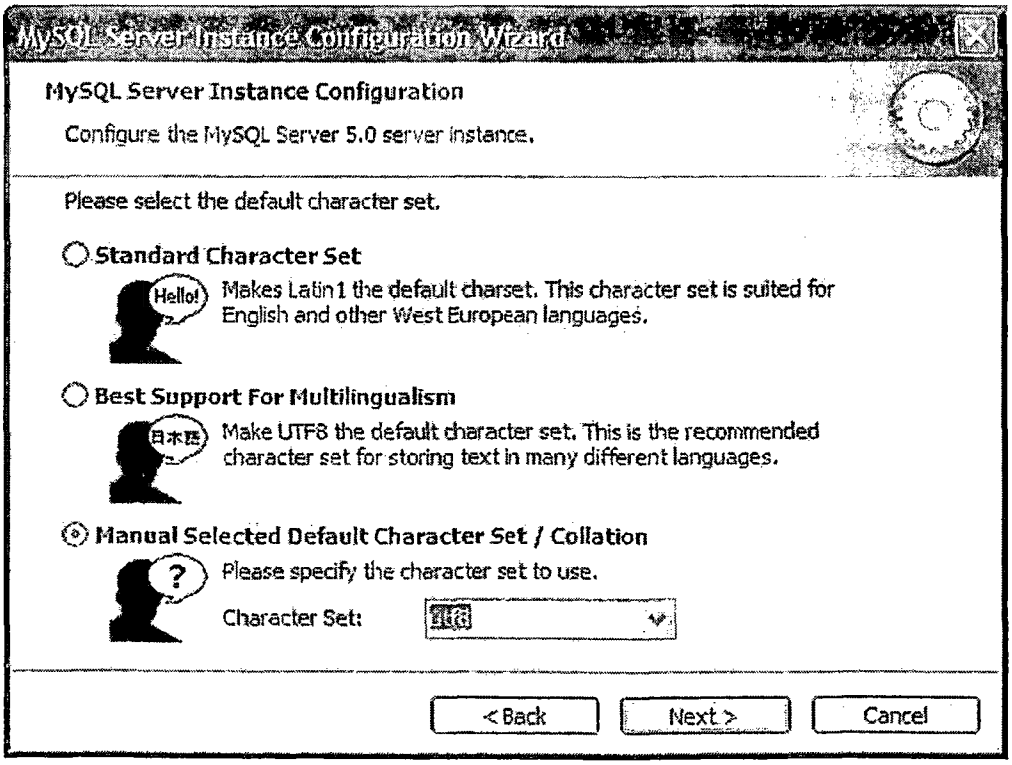

MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.0 server instance.
Please select a configuration type.
Detailed Configuration
Choose this configuration type to create the optimal server setup for this machine.
Standard Configuration
Use this only on machines that do not already have a MySQL server installation. This will use a general purpose configuration for the server that can be tuned manually.
< Back
Next >
Cancel

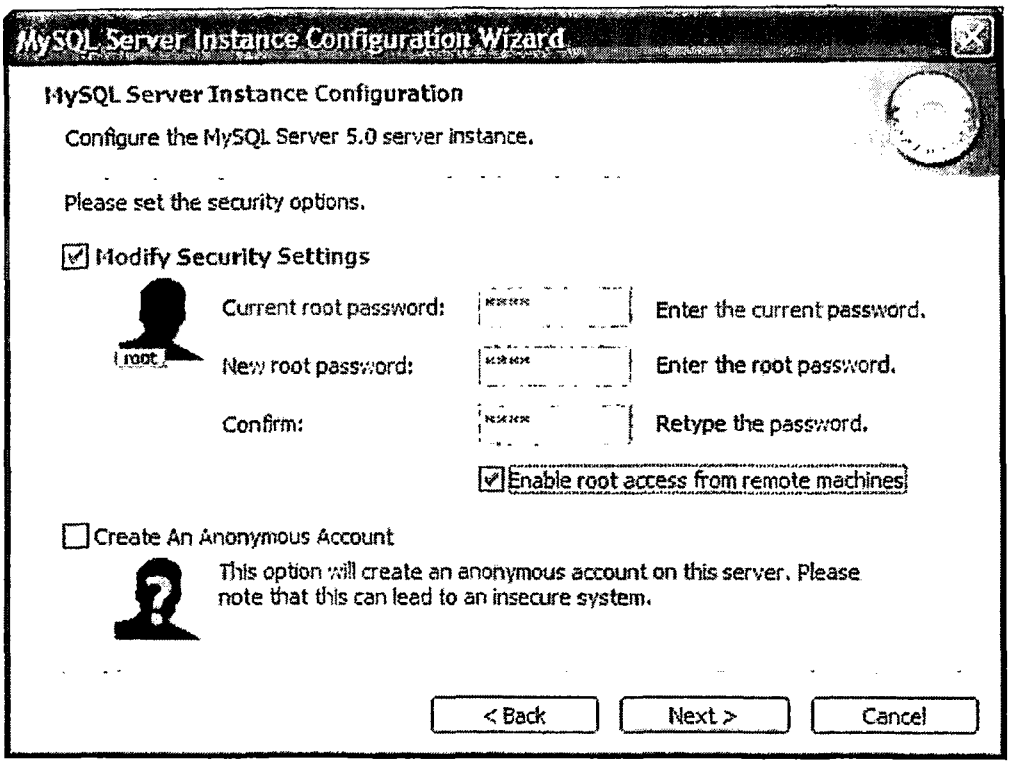

MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.0 server instance.
Please select a server type. This will influence memory, disk and CPU usage.
Developer Machine
This is a development machine, and many other applications will be run on it. MySQL 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

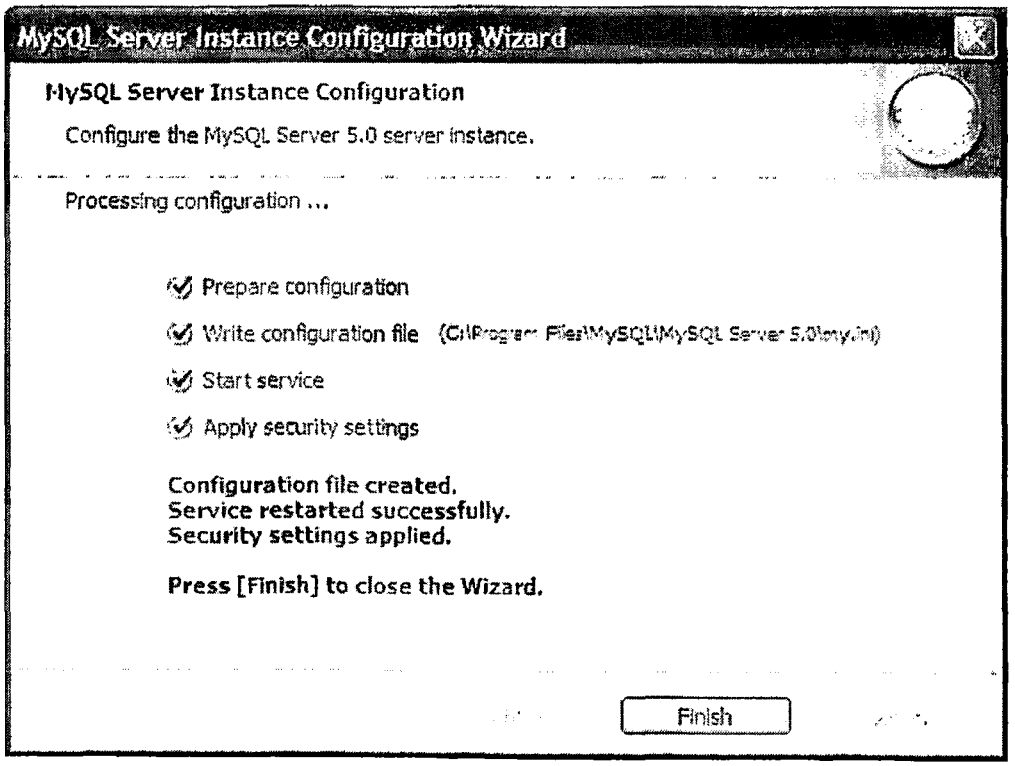

MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.0 server instance.
Please select the database usage.
Multifunctional Database
General purpose databases. This will optimize the server for the use of the fast transactional InnoDB storage engine 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


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


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

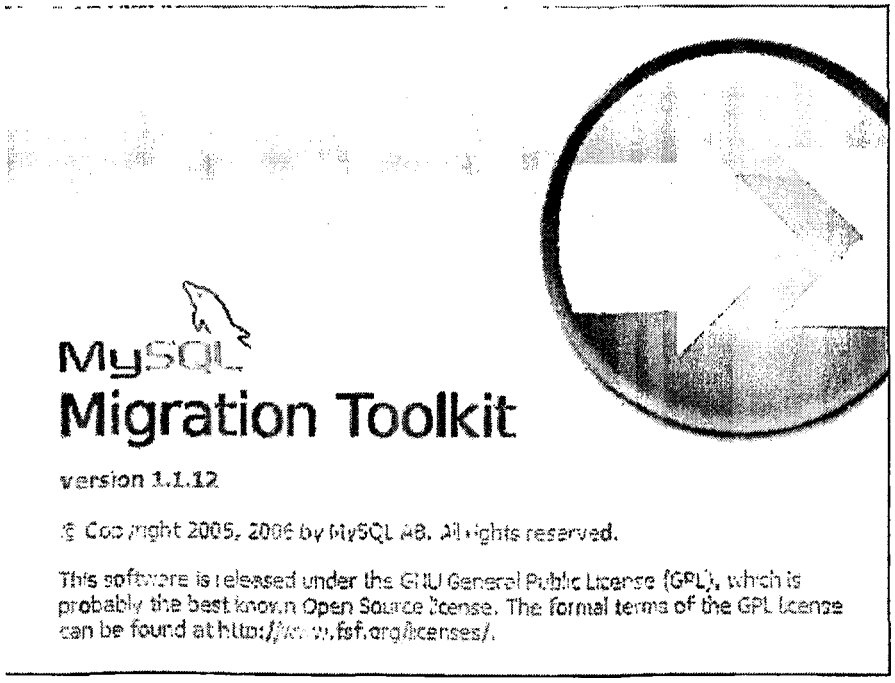

MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.0 server instance.
Please select the default character set.
Standard Character Set
Hello!
Makes Latin1 the default charset. This character set is suited for English and other West European languages.
Best Support For Multilingualism
日本語
Make UTF8 the default character set. This is the recommended character set for storing text in many different languages.
Manual Selected Default Character Set / Collation
?
Please specify the character set to use.
Character Set:
< Back
Next >
Cancel

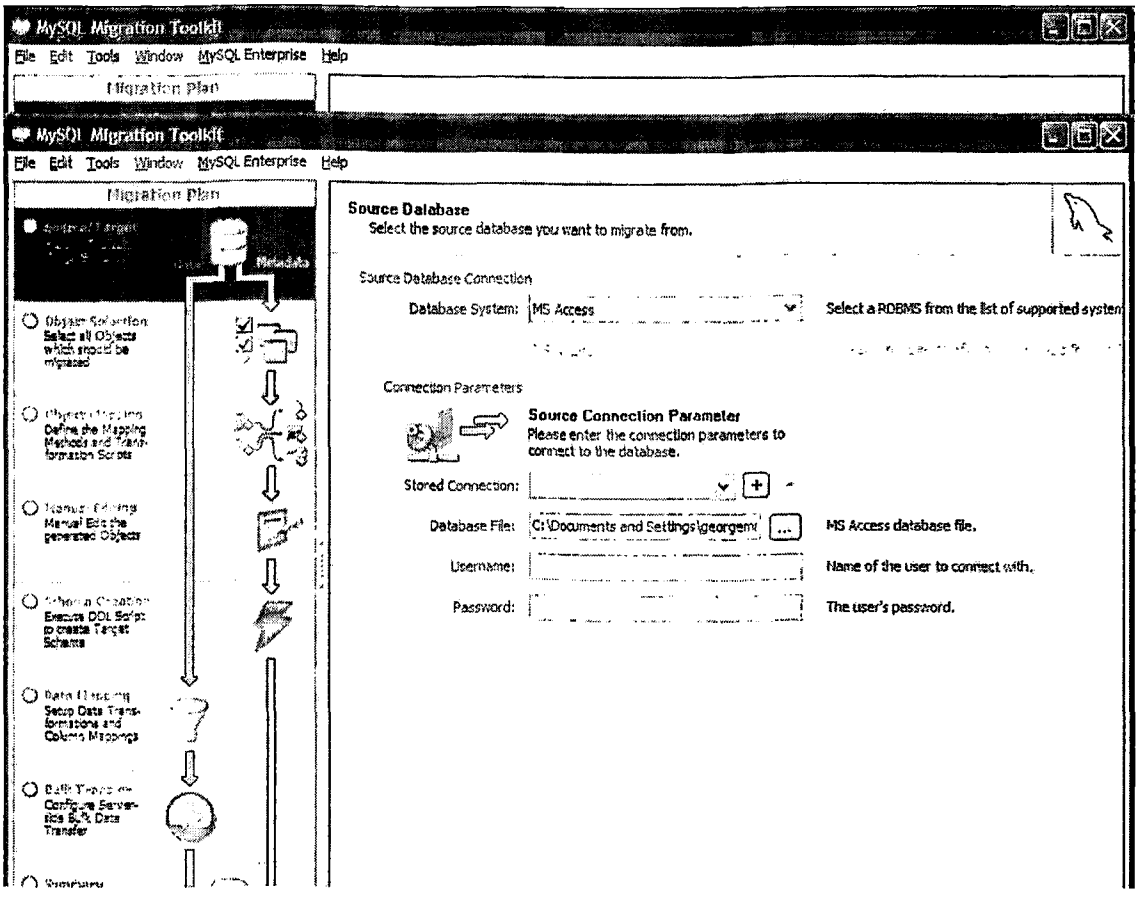

MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.0 server instance.
Please set the security options.
Modify Security Settings
root
Current root password:
Enter the current password.
New root password:
Enter the root password.
Confirm:
Retype the password.
Enable root access from remote machines
Create An Anonymous Account
This option will create an anonymous account on this server. Please note that this can lead to an insecure system.
< Back
Next >
Cancel

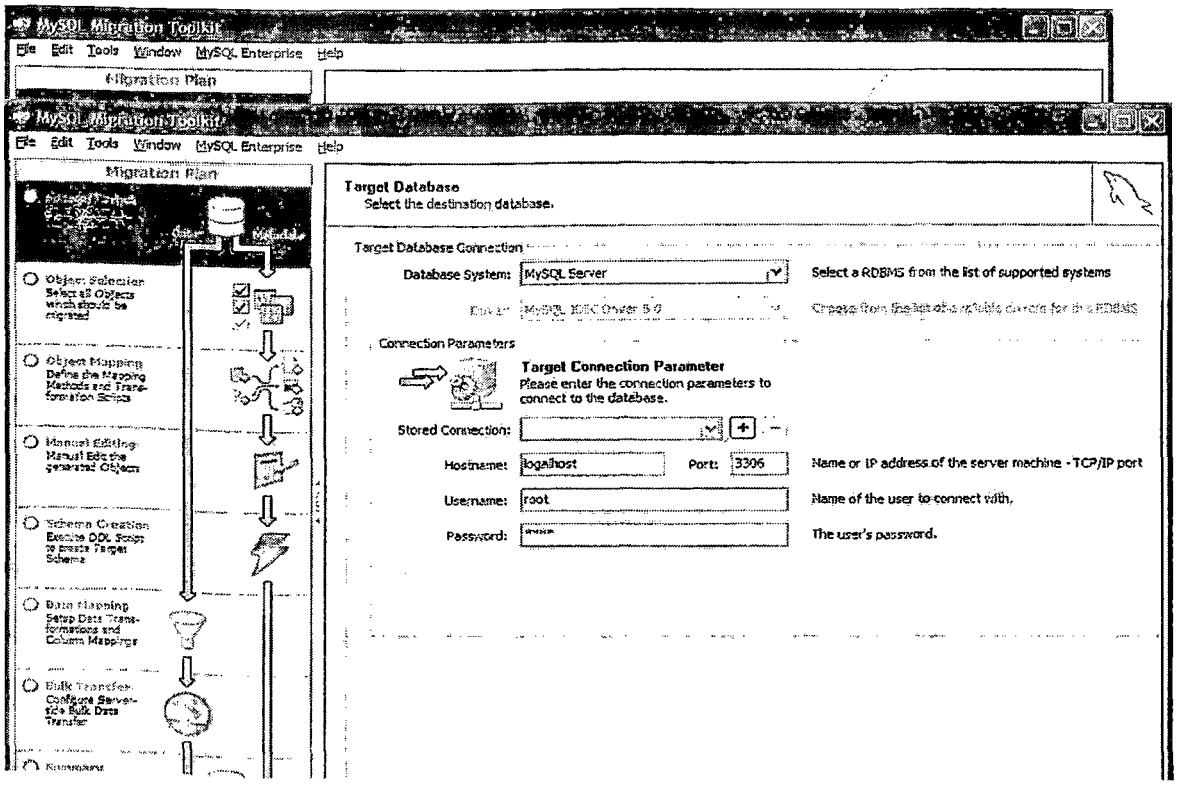

MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.0 server instance.
Processing configuration ...
Prepare configuration
Write configuration file
Start service
Apply security settings
Configuration file created.
Service restarted successfully.
Security settings applied.
Press [Finish] to close the Wizard.
Finish

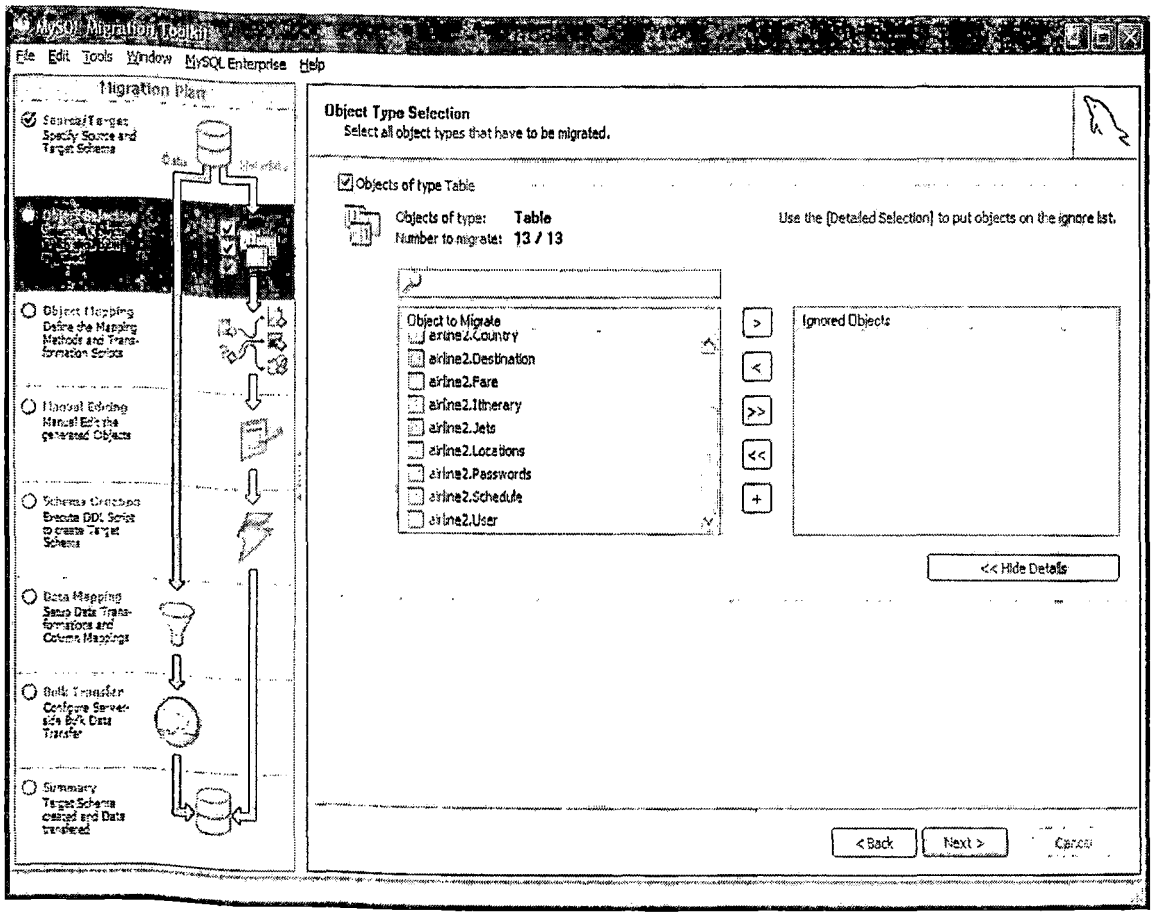
Στη συνέχεια πρέπει να γίνει η μετανάστευση (migration) της βάσης από την Access σε MySQL και για την ενέργεια αυτή θα χρησιμοποιήσουμε το MySQL Migration Tool.

9.2. MySQL Migration Tool

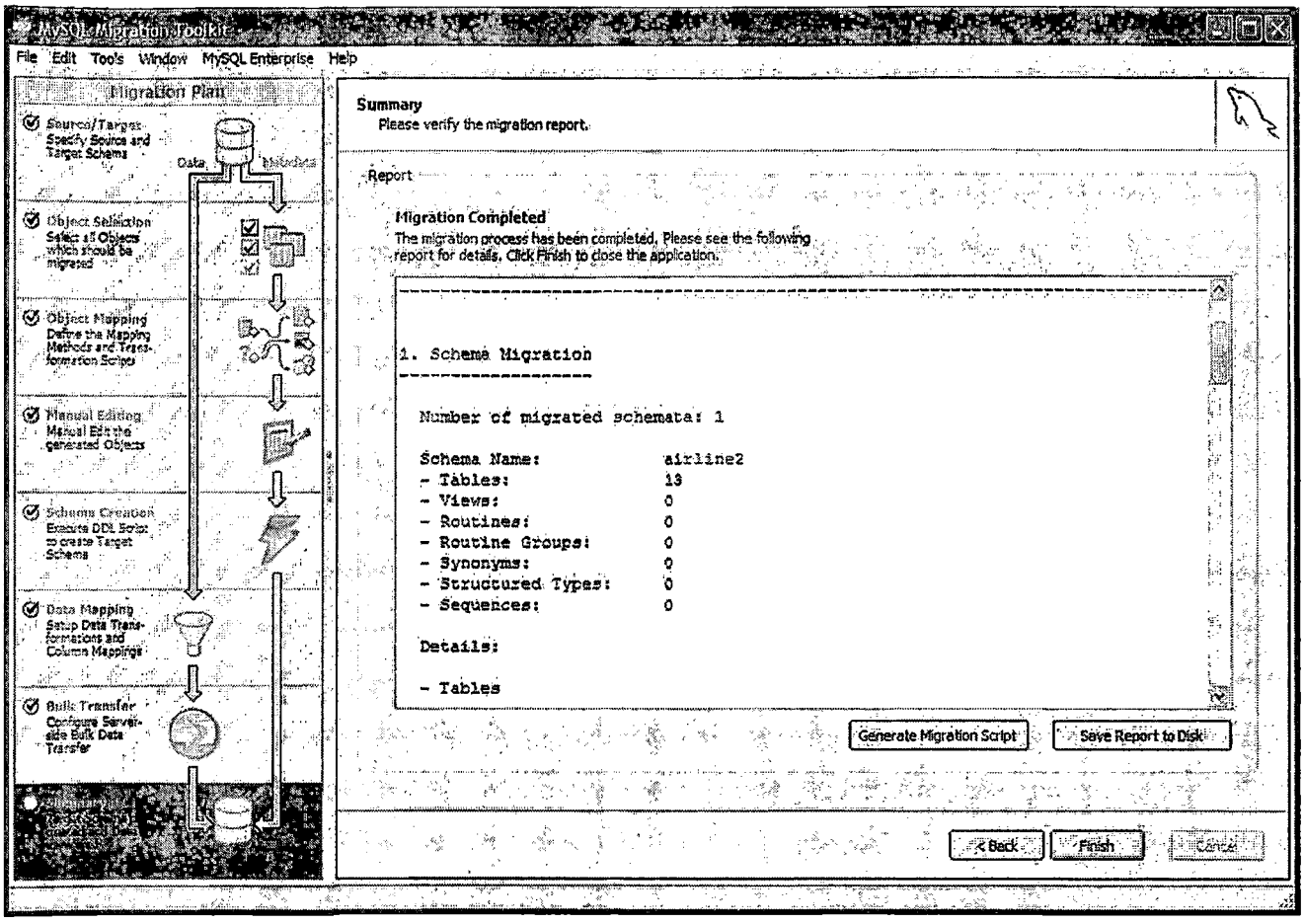








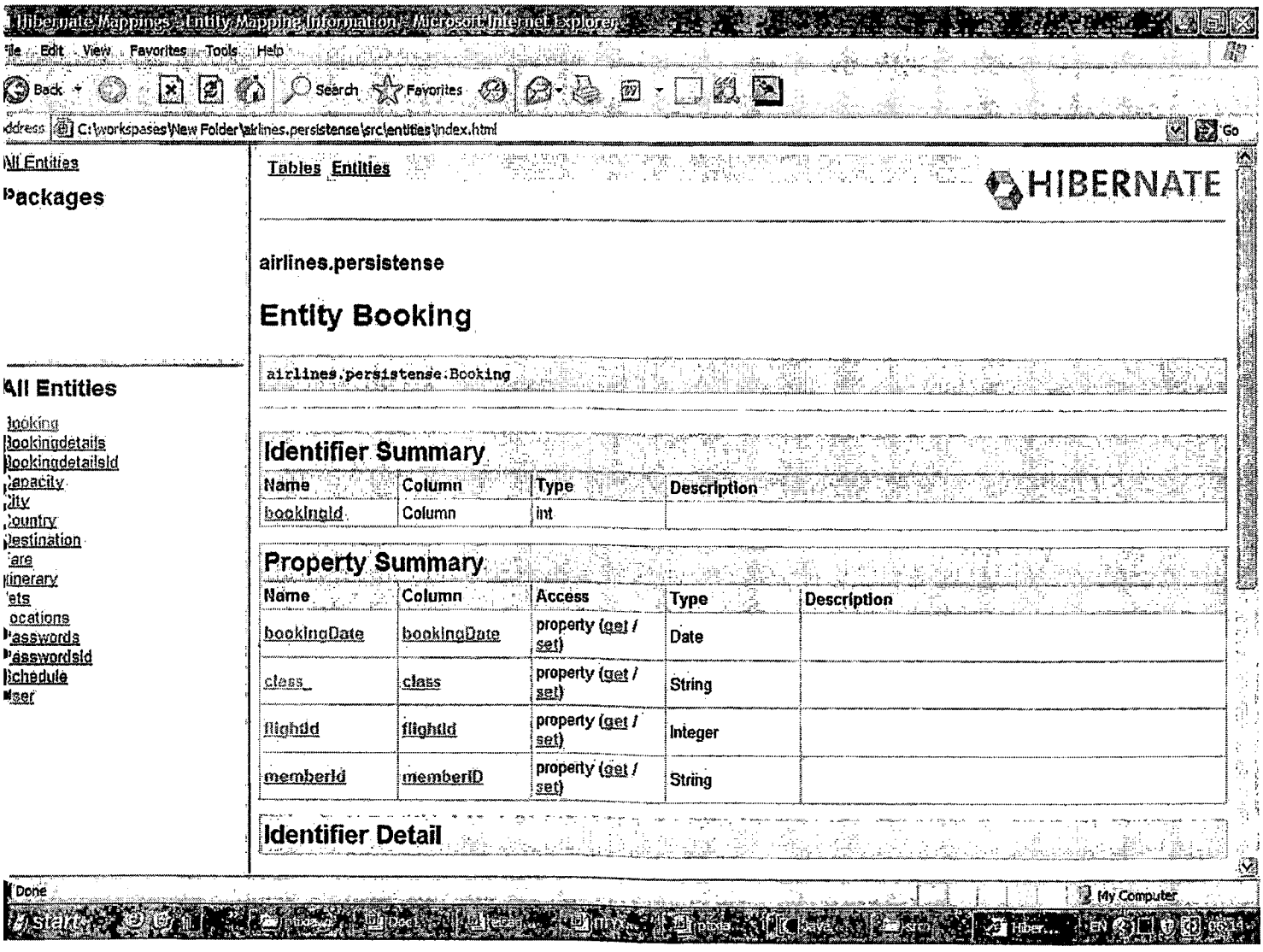
Μαγκαφάς Γιώργος
ΑΜ 10590



Μετά την επιτυχημένη διαδικασία μετανάστευσης στη MySQL έχει σειρά η διαδικασία του Hibernate Mapping μέσο του Eclipse. Σημαντικό στοιχείο είναι ότι αυτή η έκδοση του Eclipse έρχεται με ενσωματωμένο τον JBoss Server όπου αυτό μας απαλλάσσει από επιπλέον ρυθμίσεις και χρονοβόρες διαδικασίες για το Reverse Engineer.

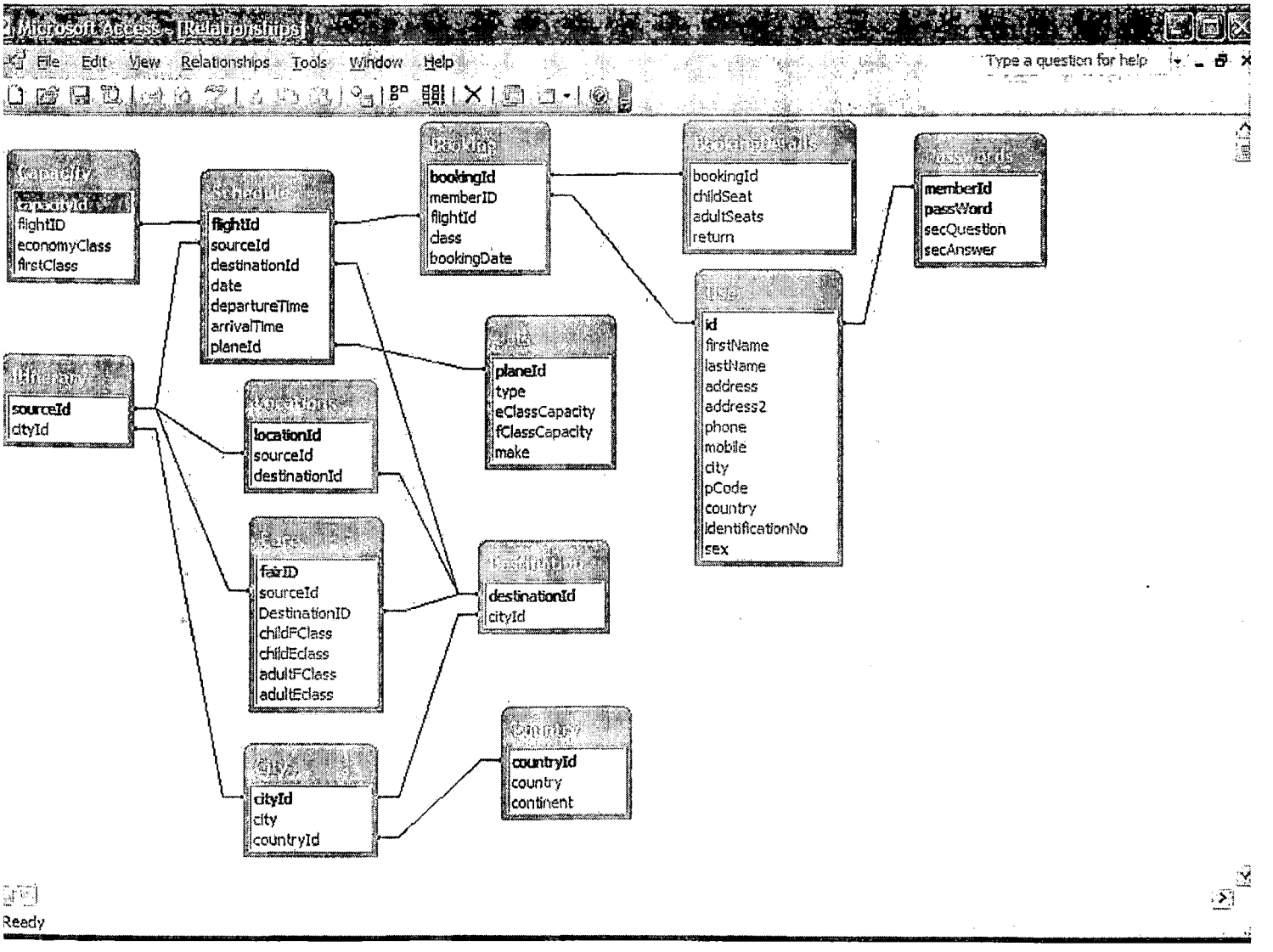
10.Hibernate Mapping

Στην παρακάτω οθόνη φαίνονται χαρακτηριστικά οι πίνακες που έχουμε στη βάση δεδομένων σε μορφή Mapping για γρήγορη και εύκολη διαχείρισή τους.



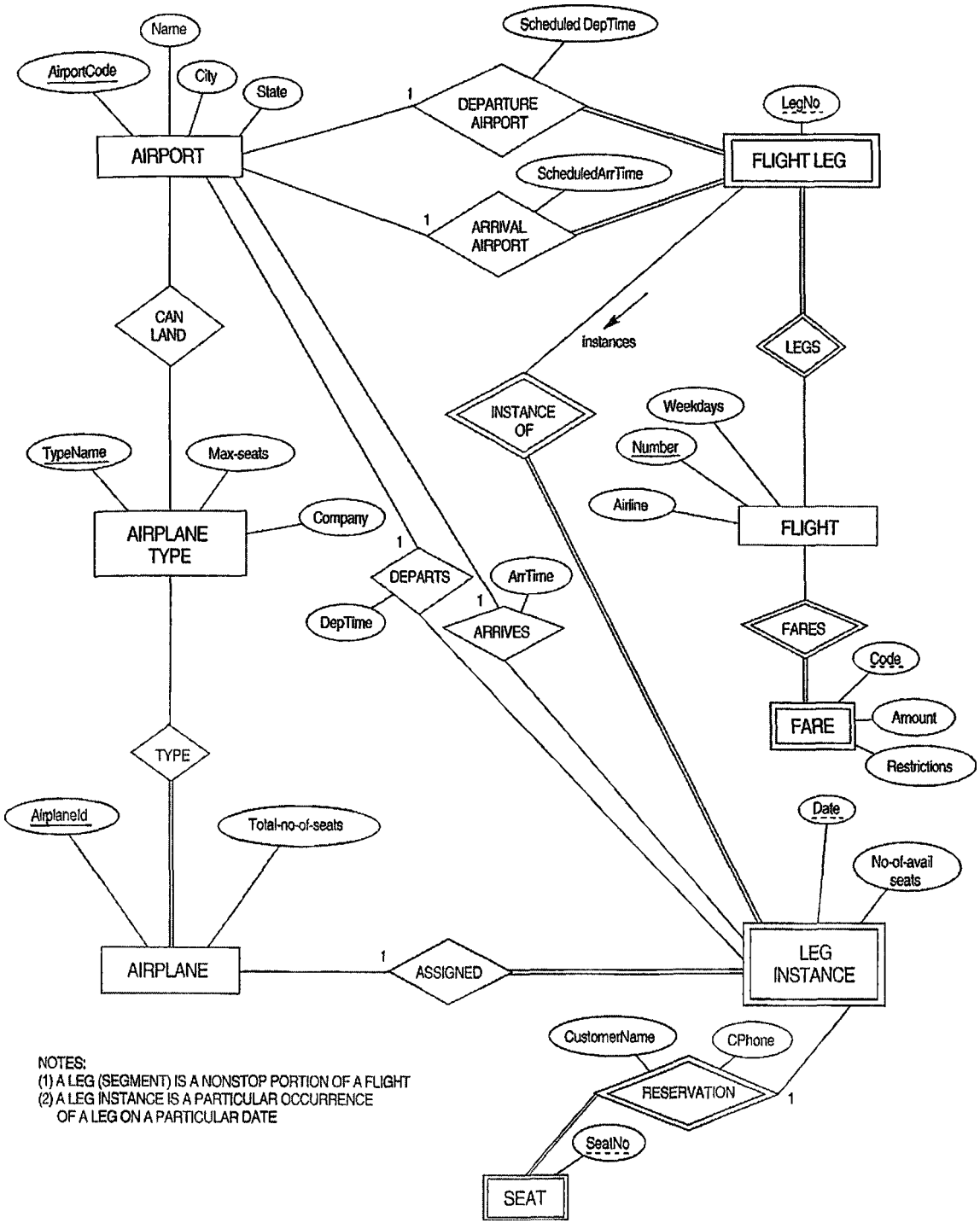
Η Βάση Δεδομένων του Συστήματος όπως αναφέραμε και πιο πάνω είχε αρχικά δημιουργηθεί σε Access και στη συνέχεια μέσο του Migration tool την μετατρέψαμε σε MySQL.

Στη παρακάτω εικόνα παρουσιάζονται οι σχέσεις μεταξύ των πινάκων (relationships) όπως φαίνονται μέσα από την Access.



Μαγκαφάς Γιώργος
ΑΜ 10590

11. Το E–R διάγραμμα της εφαρμογής.



Μαγκαφάς Γιώργος
ΑΜ 10590

11.1. Ανάλυση της Βάσης Δεδομένων και του E–R.

Η Βάση μας αποτελείτε από 13 πίνακες συνδεδεμένους μεταξύ τους παρεχοντας πληροφορίες και δεδομένα στους χρήστες του συστήματος .

11.1.1. Πίνακας Booking.

Είναι ο πιο βασικός πίνακας στη ΒΔ γιατί αυτό είναι και το ζητούμενο από τους χρήστες η κράτηση εισιτηρίων μέσω του internet.
Περιέχει τα πεδία bookingid (Primary key),meberid,Flightid,class,bookingdate.

11.1.2. Πίνακας schedule

Ο πίνακας αυτός περιγράφει τις προγραμματισμένες πράξεις για μια πτήση και έχει τα πεδία flightid(Primary Key),sourceid,destinationid,date,departuretime,arrivaltime,planeid

11.1.3. Πίνακας capacity

Περιέχει τα πεδία capacityid(PK) flightid, economyclass,firstclass και μας δείχνει το είδος του εισιτηρίου economy και business .

11.1.4. Πίνακας itinerary

Ο πίνακας αυτός έχει τον ρόλο της ένωσης των πινάκων schedule location fare city και έχει τα πεδία sourceid(PK),cityid.

11.1.5. Πίνακας locations

Ο πίνακας αυτός μας υποδεικνύει το προορισμό της πτήσης και έχει ως πεδία το locationid(PK),sourceid, destinationid.

11.1.6. Πίνακας fare

Έχει τα πεδία fareid(PK), sourceid, destinationid, childfclass, childeclass, adultfclass, adulteclass και αναφέρεται στο είδος του εισιτηρίου.

11.1.7. Πίνακας destination

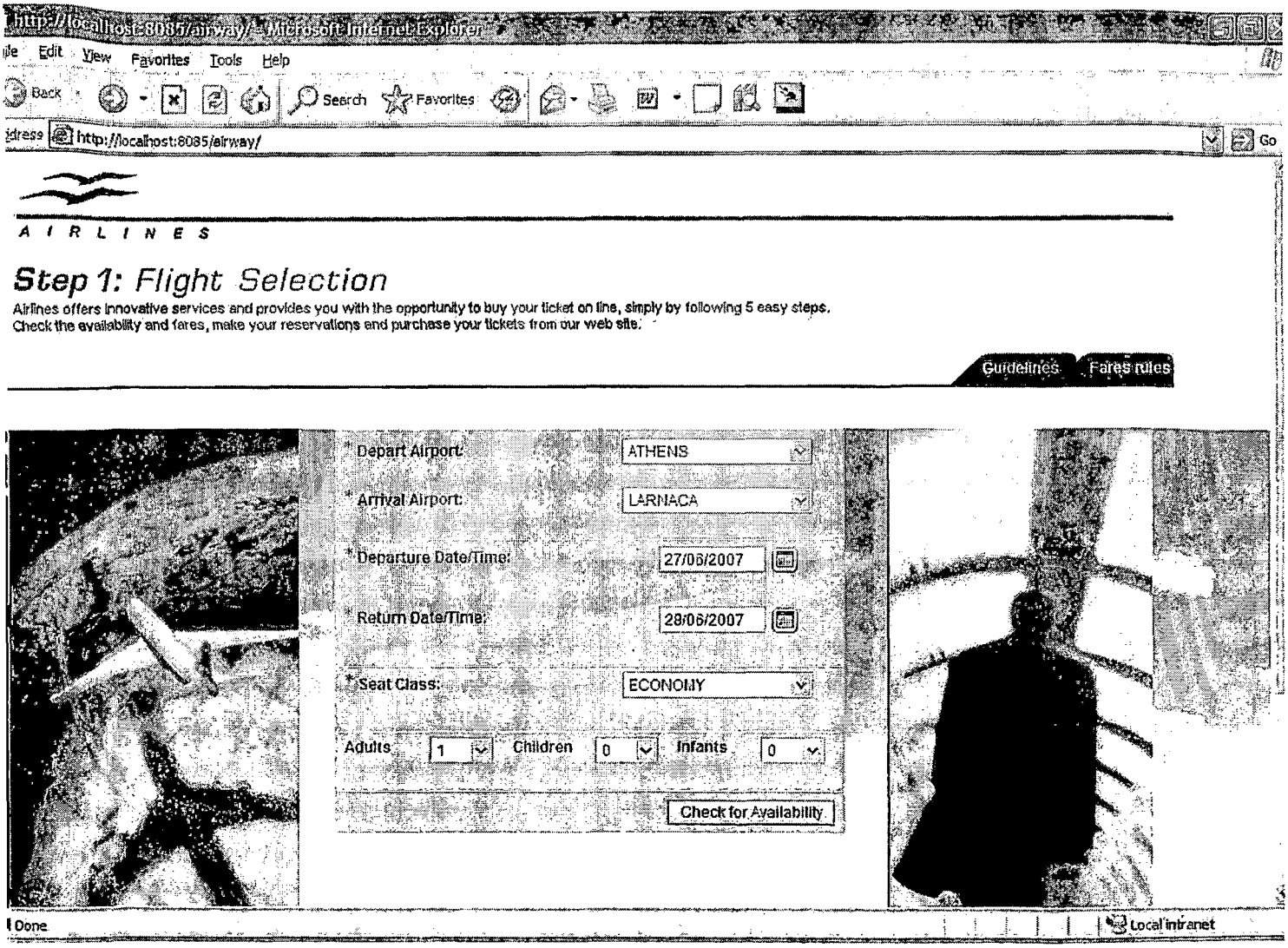
Ενώνεται με τους πίνακες schedule, locations, fare, city και δηλώνει τον προορισμό της πτήσης .

11.1.8. Πίνακας User

Αυτός ο πίνακας περιέχει τα στοιχεία του χρήστη που κάνει την κράτηση του εισιτηρίου. Ενώνεται άμεσα με τον πίνακα booking και έχει id(PK), firstname, lastname, address, address2, phone, mobile ,city ,pcode, country, identificationNo, sex

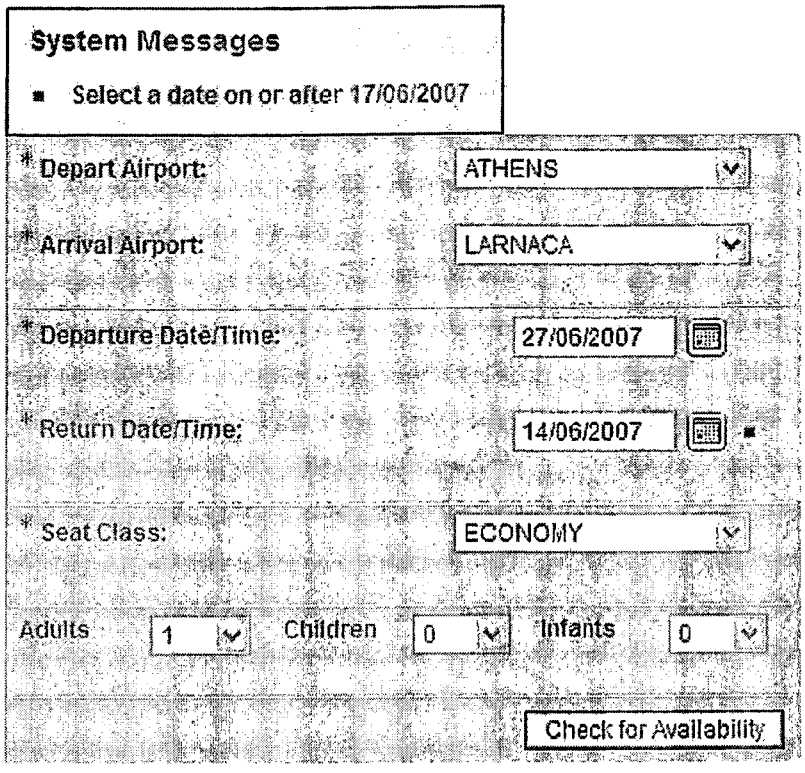
12. Υλοποίηση Συστήματος

Παρακάτω βλέπουμε την κεντρική σελίδα της εφαρμογής όπου ο χρήστης θα επιλέξει τον προορισμό του την ημέρα αναχώρησης και επιστροφής του την θέση στην οποία επιθυμεί να καθίσει καθώς και τυχών άλλα άτομα (αριθμό και κατηγορία) που θα ταξιδέψουν μαζί του.

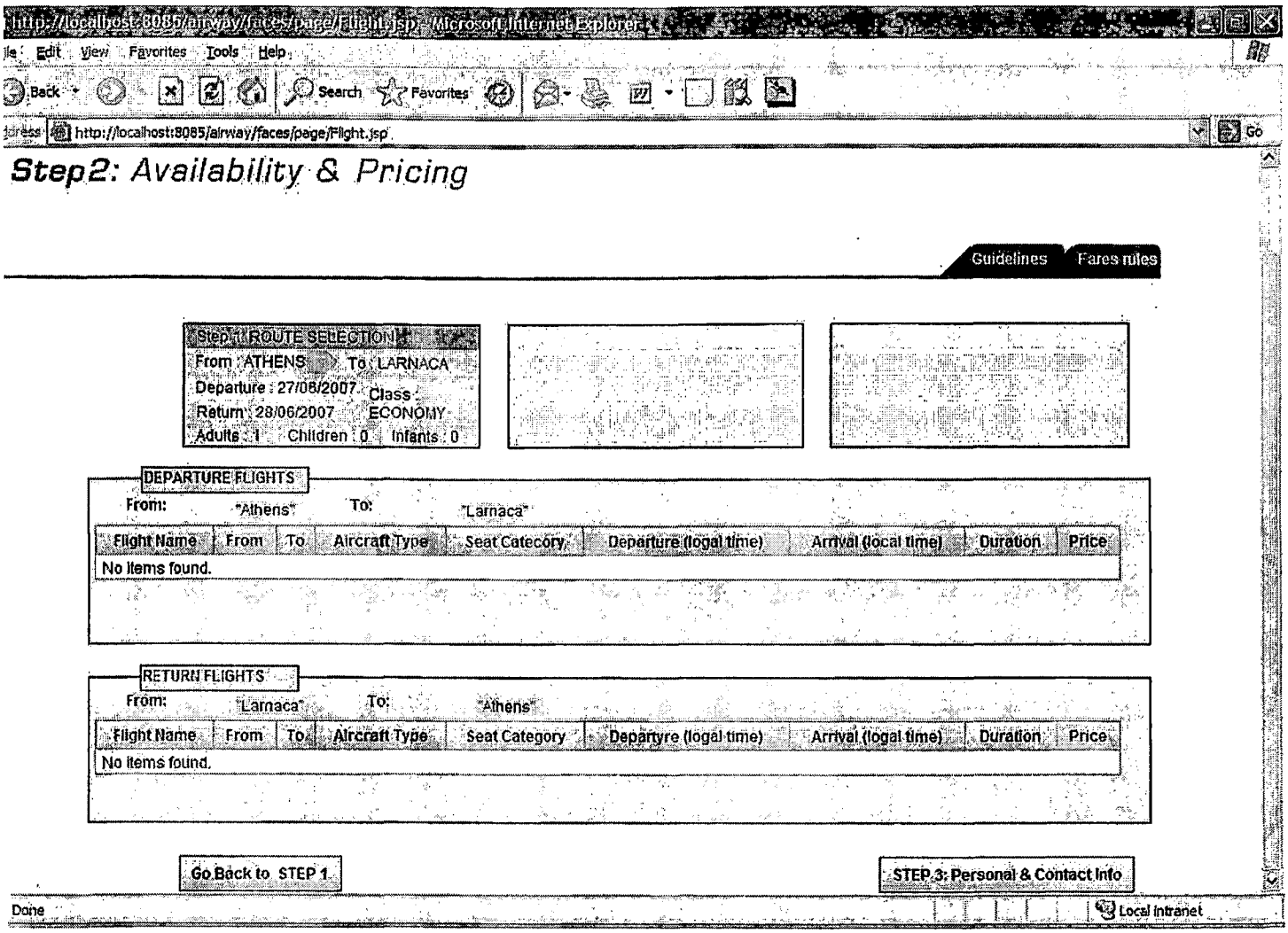


Ακολούθως πατώντας το κουμπί "check for availability" το σύστημα θα ελέγξει την αίτηση του χρήστη και αν όλα τα παιδία είναι συμπληρωμένα σωστά θα προχωρήσει στο επόμενο στάδιο, την επιλογή πτήσης.

Αν όχι τότε το σύστημα θα εμφανίσει ένα μήνυμα λάθους και δεν θα αφήσει το χρήστη να προχωρήσει αν δεν το διορθώσει. Όπως φαίνετε στην παρακάτω εικόνα.



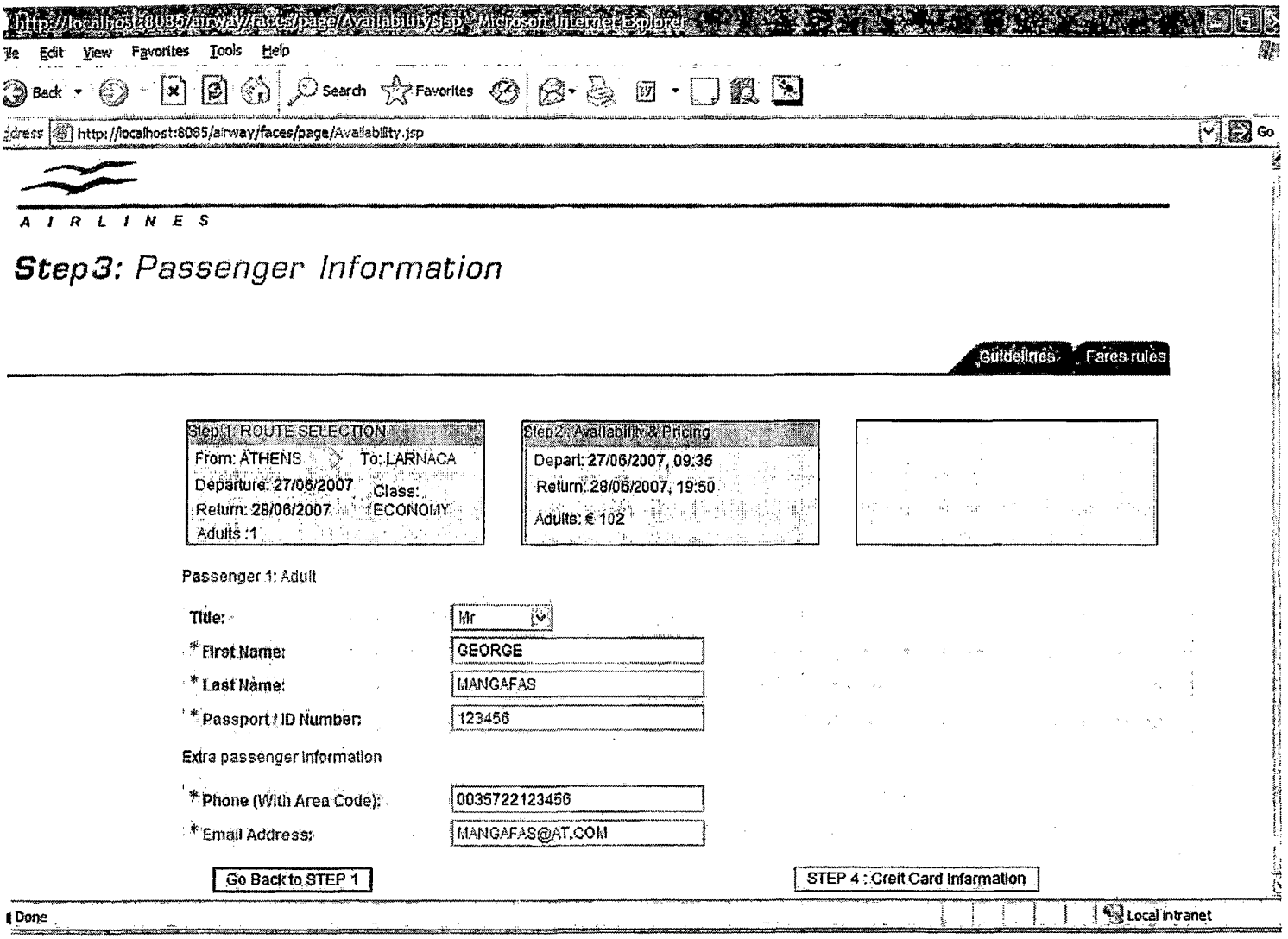
Σε αυτή την περίπτωση το σύστημα εντόπισε λάθος κατά την εισαγωγή της ημερομηνίας. Συγκεκριμένα ο χρήστης έβαλε ημερομηνία πριν από την ημερομηνία του συστήματος που σε αυτή την περίπτωση είναι η 17/06/2007.



Μετά την εισαγωγή των σωστών παραμέτρων προχωράμε στο βήμα 2 που είναι η επιλογή της ώρας και της πτήσης που μας βολεύει.

Το σύστημα μας εμφανίζει επίσης τον τύπο του αεροσκάφους για τις συγκεκριμένες πτήσεις, την διάρκεια της πτήσης καθώς και την τιμή βάσει των διαθέσιμων θέσεων στη συγκεκριμένη πτήση.

Αφού επιλέξουμε πτήση προχωράμε στο επόμενο βήμα που είναι η εισαγωγή των στοιχείων για τους επιβάτες.

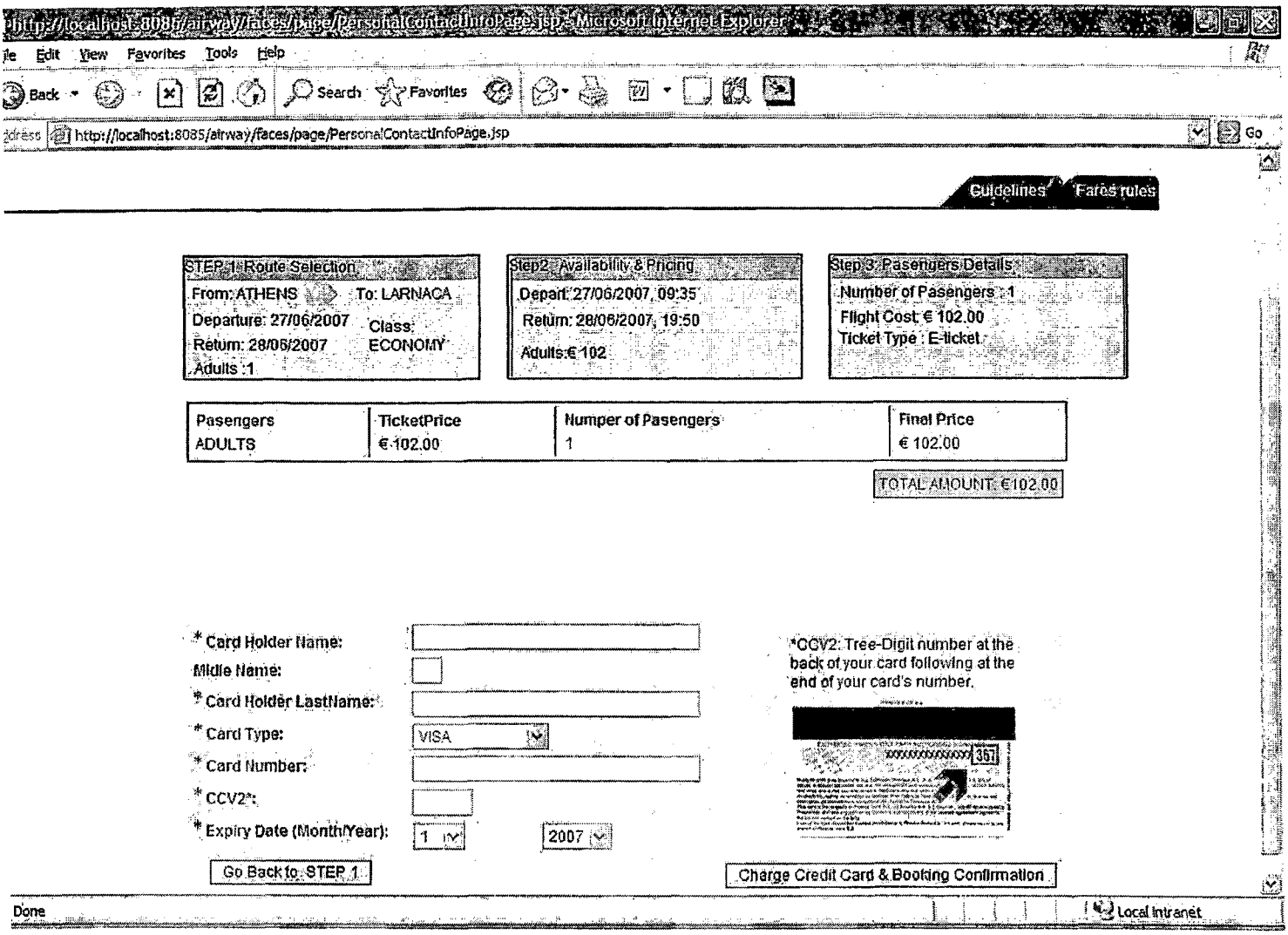


Σε αυτό το στάδιο ο χρήστης εισάγει τα προσωπικά του στοιχεία που είναι απαραίτητα για την δημιουργία της κράτησης .

Είναι απαραίτητη η εισαγωγή όλων των στοιχείων που είναι μαρκαρισμένα με * σε αυτό το στάδιο.

Αλλιώς το σύστημα θα εμφανίσει μήνυμα λάθους.

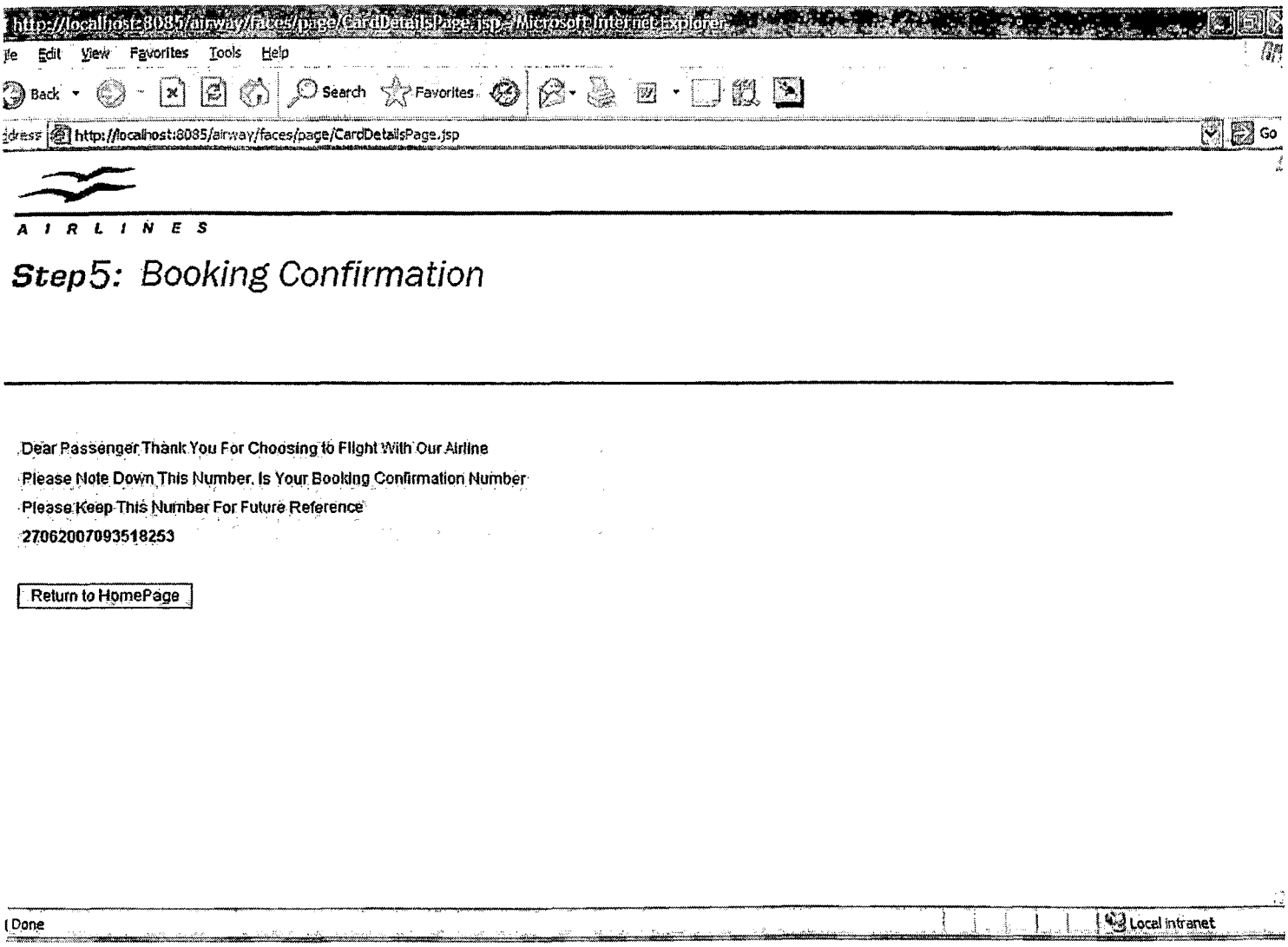
Αν όλα έχουν συμπληρωθεί σωστά τότε το σύστημα θα μας μεταφέρει στην επόμενη σελίδα (Βήμα 4) που είναι η εισαγωγή των στοιχείων της πιστωτικής κάρτας.



Στη Σελίδα αυτή βλέπουμε συγκεντρωμένες όλες τις πληροφορίες που χρειάζονται για να έχουμε μία σωστή διαδικασία κράτησης .

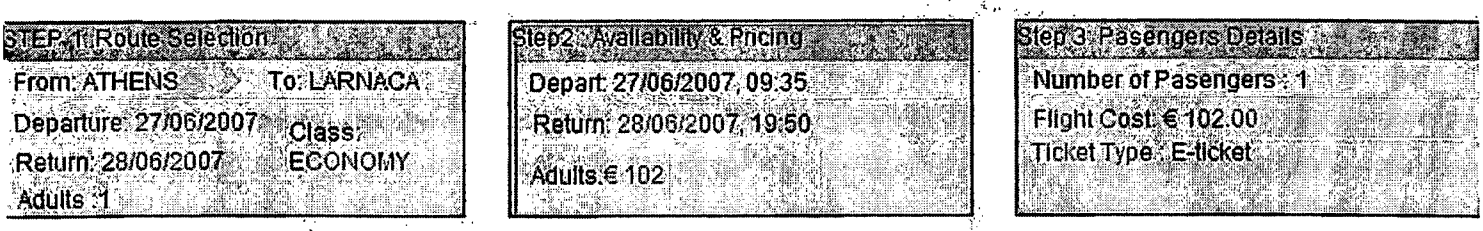
Ο χρήστης θα χρειαστεί να εισάγει τις πληροφορίες για την πιστωτική του κάρτα και ακολούθως το σύστημα θα ελέγξει την εγκυρότητα της.

Αν οι πληροφορίες που έδωσε είναι σωστές τότε το σύστημα θα μας μεταφέρει στην τελική οθόνη όπου θα μας επιστρέψει έναν αριθμό επιβεβαίωσης.



Ο αριθμός επιβεβαίωσης βγαίνει από την ημερομηνία της πτήσης την ώρα και ένα τυχαίο αριθμό από το 00001 μέχρι το 50000.

Σε όλη την διάρκεια της διαδικασίας κράτησης το σύστημα μας κρατά ενήμερους μέσο ενός info panel για τις εισαγωγές που κάναμε στο σύστημα κατά την περιήγηση μας σε αυτό.



Σε όλη την διάρκεια της διαδικασίας κράτησης αν ο χρήστης επιλέξει λάθος πτήση ή εισάγει λάθος στοιχεία και θέλει να τα επαναφέρει πρέπει αναγκαστικά να αρχίσει από την αρχή την διαδικασία.

Μαγκαφάς Γιώργος
ΑΜ 10590

13.ΣΥΜΠΕΡΑΣΜΑΤΑ

Η ευχρηστία αποτελεί πολύ σημαντικό στοιχείο για την αποδοχή του συστήματος. Βασικά εκφράζεται μέσω της ακόλουθης ερώτησης:

"Πότε το σύστημα είναι αρκετά καλό για να ικανοποιήσει όλες τις ανάγκες και τις απαιτήσεις των χρηστών;"

Γενικά, η ολική αποδοχή (acceptability) του συστήματος είναι ο συνδυασμός τόσο της κοινωνικής αποδοχής (social acceptability) όσο και της πρακτικής αποδοχής (practical acceptability) του.

Αν θεωρήσουμε ως δεδομένο ότι το σύστημα είναι κοινωνικά αποδεκτό, μπορούμε και αναλύουμε την πρακτική αποδοχή του σε διάφορες κατηγορίες. Περιλαμβάνουμε τόσο τις παραδοσιακές κατηγορίες, όπως η αξιοπιστία, η συμβατότητα με τα υπάρχοντα συστήματα κλπ., όσο και την κατηγορία της χρησιμότητας (usefulness).

Χρησιμότητα είναι η έννοια εκείνη που σχετίζεται με το αν ένα σύστημα μπορεί να χρησιμοποιηθεί για να επιτύχει κάποιους επιθυμητούς στόχους. Μπορεί επιπρόσθετα να διαιρεθεί σε άλλες δύο κατηγορίες:

- *Χρηστικότητα*: αναφέρεται στην ερώτηση πότε η λειτουργικότητα ενός συστήματος ικανοποιεί τις απαιτήσεις.
- *Ευχρηστία*: αναφέρεται στην ερώτηση σε πόσο καλό βαθμό μπορούν οι χρήστες να χρησιμοποιήσουν τις λειτουργίες του συστήματος.

Η ευχρηστία εφαρμόζεται σε όλα τα τμήματα του συστήματος με το οποίο ένας άνθρωπος μπορεί να αλληλεπιδρά περιλαμβάνοντας και τα στάδια της εγκατάστασης και συντήρησής του. Είναι πολύ σημαντικό να γίνει κατανοητό ότι η ευχρηστία δεν είναι ένα απλό μονοδιάστατο χαρακτηριστικό της διεπιφάνειας χρήστη. Αποτελείται από διάφορα στοιχεία, τα οποία συνοψίζονται στα ακόλουθα πέντε βασικά χαρακτηριστικά:

- **Ευκολία στη μάθηση (learnability):** το σύστημα πρέπει να είναι εύκολο στην εκμάθηση, ώστε ο χρήστης να μπορεί να ολοκληρώνει γρήγορα την εργασία του.
- **Αποδοτικότητα χρήσης (efficiency):** το σύστημα πρέπει να είναι αποδοτικό στη χρήση, έτσι ώστε όταν ο χρήστης το μάθει να μπορεί να αποδώσει σε υψηλό βαθμό.
- **Ευκολία απομνημόνευσης (memorability):** το σύστημα θα πρέπει να το θυμούνται εύκολα οι χρήστες. Έτσι ένας μη τακτικός χρήστης θα είναι ικανός να γυρίσει στο σύστημα ακόμα και μετά από αρκετό καιρό απουσίας, χωρίς να απαιτείται να τα ξαναμάθει όλα από την αρχή.
- **Χαμηλή συχνότητα σφαλμάτων (few errors):** το σύστημα θα πρέπει να έχει πάρα πολύ χαμηλή συχνότητα σφαλμάτων. Με αυτό τον τρόπο οι χρήστες θα κάνουν μικρό αριθμό λαθών κατά τη διάρκεια χρήσης του. Ακόμα όμως και αν κάνουν λάθη θα πρέπει να μπορούν με εύκολο τρόπο να τα διορθώσουν. Επιπλέον, καταστροφικά λάθη δεν θα πρέπει να συμβαίνουν ποτέ.
- **Υποκειμενική ικανοποίηση χρήστη (subjective satisfaction):** το σύστημα θα πρέπει να είναι ευχάριστο στη χρήση του και οι χρήστες να είναι υποκειμενικά ικανοποιημένοι από αυτό.

Η ευχρηστία τυπικά μετριέται με τη βοήθεια χρηστών (επιλέγονται να είναι όσο το δυνατόν πιο αντιπροσωπευτικοί των τελικών χρηστών), που χρησιμοποιούν το σύστημα για να εκτελέσουν ένα σύνολο από προκαθορισμένες ενέργειες. Επίσης, μπορεί να μετρηθεί από πραγματικούς χρήστες, οι οποίοι εκτελούν όποιες εργασίες επιθυμούν. Σε κάθε περίπτωση πάντως το σημαντικό είναι ότι η ευχρηστία είναι μετρήσιμο μέγεθος (εκτέλεση συγκεκριμένων ενεργειών από συγκεκριμένους χρήστες). Ενδεχομένως, το ίδιο σύστημα, όταν μετρηθεί, να εμφανίσει διαφορετικά χαρακτηριστικά ευχρηστίας, εάν χρησιμοποιηθούν διαφορετικοί χρήστες για διαφορετικές εργασίες.

Η ευκολία μάθησης είναι ίσως από μια άποψη το πιο σημαντικό χαρακτηριστικό της ευχρηστίας. Άλλωστε, η πρώτη εμπειρία που οι περισσότεροι χρήστες έχουν με ένα σύστημα είναι κατά τη διαδικασία μάθησης της χρήσης του. Βέβαια στο σύστημά μας το περιβάλλον διεπαφής δεν είναι πολύπλοκο και ούτε απαιτείται ειδική εκπαίδευση των χρηστών.

Ο υψηλός δείκτης ευκολίας μάθησης του συστήματος έχει μια απότομη κλίση στο πρώτο τμήμα της καμπύλης μάθησης και επιτρέπουν στους χρήστες να φτάσουν σε ένα λογικό επίπεδο επιτυχίας χρήσης σε σύντομο χρονικό διάστημα.

Όλες οι εφαρμογές και οι υπηρεσίες παρουσιάζουν μια καμπή μάθησης. Αν και υποστηρίζεται από πολλούς, στην πραγματικότητα η ευχρηστία και η ευκολία μάθησης δεν είναι έννοιες αμοιβαία αποκλειόμενες. Με άλλα λόγια, για να μπορέσουμε να σχεδιάσουμε ένα εύχρηστο περιβάλλον διεπαφής δεν χρειάζεται κατ' ανάγκη να καταφύγουμε σε μια λύση εξαιρετικά δύσκολη στην εκμάθηση, που θα εξυπηρετεί μακροπρόθεσμα τον εξαιρετικά έμπειρο χρήστη της εφαρμογής μας. Η ανάγκη για εύχρηστα περιβάλλοντα διεπαφής είναι εξίσου επιτακτική με την ανάγκη για συστήματα και εφαρμογές που δεν παρουσιάζουν εκτενή καμπύλη μάθησης.#

Η αποδοτικότητα αναφέρεται στο σταθερής κατάστασης επίπεδο απόδοσης των έμπειρων χρηστών, τη στιγμή δηλαδή που η καμπύλη μάθησης ευθυγραμμίζεται. Φυσικά, οι χρήστες θα καθυστερήσουν να φτάσουν σ' αυτό το τελικό επίπεδο απόδοσης. Επίσης, κάποιοι χρήστες πιθανόν να συνεχίσουν να μαθαίνουν συνεχώς, αν και οι περισσότεροι χρήστες φαίνεται ότι σταματούν όταν έχουν μάθει «αρκετά». Δυστυχώς, αυτό το σταθερής κατάστασης επίπεδο απόδοσης δεν είναι ιδανικό για χρήστες, οι οποίοι, μαθαίνοντας πρόσθετα προχωρημένα χαρακτηριστικά, μερικές φορές γλιτώνουν περισσότερο χρόνο κατά τη χρήση του συστήματος παρά όταν μαθαίνουν τη λειτουργία τους.

Οι ευκαιριακοί χρήστες είναι η τρίτη μεγάλη κατηγορία μετά τους αρχάριους και τους έμπειρους χρήστες. Οι ευκαιριακοί χρήστες χρησιμοποιούν το σύστημα περιστασιακά σε σχέση με τους έμπειρους χρήστες, για τους οποίους υποθέτουμε μεγάλη συχνότητα χρήσης. Εν τούτοις, σε αντίθεση με τους αρχάριους χρήστες, οι ευκαιριακοί χρήστες έχουν χρησιμοποιήσει το σύστημα κάποια στιγμή, οπότε δεν απαιτείται να το μάθουν από την αρχή, απλώς απαιτείται να θυμηθούν πώς να το χρησιμοποιήσουν βασιζόμενοι στην προηγούμενη εμπειρία.

Η ύπαρξη μιας διεπαφής χρήστη που είναι εύκολο να τη θυμηθεί κανείς είναι επίσης σημαντικό στοιχείο για τους χρήστες που επιστρέφουν σε αυτήν μετά από διακοπές ή που είχαν κάποιο άλλο λόγο και σταμάτησαν περιοδικά τη χρήση του προγράμματος.

Τα περισσότερα περιβάλλοντα διεπαφής χρήστη είναι κτισμένα με βάση την αρχή να γίνουν όσο το δυνατόν πιο ορατά στους χρήστες. Οι χρήστες αυτών των συστημάτων δεν χρειάζεται να είναι ενεργά ικανοί για να θυμηθούν τι είναι διαθέσιμο, αφού το σύστημα θα τους το θυμίσει όταν είναι απαραίτητο.

Οι χρήστες θα πρέπει να κάνουν όσο το δυνατόν λιγότερα λάθη, όταν χρησιμοποιούν το σύστημα. Τυπικά, ένα λάθος ορίζεται ως μια ενέργεια η οποία δεν εκπληρώνει τον επιθυμητό στόχο. Ο ρυθμός των λαθών του συστήματος καθορίζεται από τη μέτρηση του αριθμού τέτοιων ενεργειών που γίνονται από χρήστες όταν εκτελούν κάποιες καθορισμένες εργασίες. Ο ρυθμός των λαθών μπορεί έτσι να χρησιμοποιηθεί ως μέρος πειράματος που μετράει άλλα χαρακτηριστικά της ευχρηστίας.

Εάν απλώς ορίσουμε το λάθος ως η κάθε μη σωστή ενέργεια του χρήστη, δεν λαμβάνουμε υπόψη την επίδραση των διαφορετικών λαθών. Μερικά λάθη διορθώνονται αμέσως από το χρήστη και δεν έχουν καμία επίδραση παρά μόνο να ελαττώσουν το ρυθμό ενεργειών του. Τέτοια λάθη δεν χρειάζεται να μετριούνται ξεχωριστά, αφού η επίδρασή τους περικλείεται στην αποδοτικότητα χρήσης, εάν μετράμε τον κανονικό τρόπο σε σχέση με το χρόνο αλληλεπιδράσεων των χρηστών.

Άλλα λάθη είναι περισσότερο καταστροφικά από τη φύση τους, είτε επειδή δεν ανακαλύπτονται από το χρήστη, οδηγώντας σε ένα λάθος προϊόν εργασίας, είτε εξαιτίας της καταστροφής της εργασίας του χρήστη, κάνοντας δύσκολη την ανάκτησή της. Τέτοια καταστροφικά λάθη πρέπει να μετριούνται ξεχωριστά από τα μη σημαντικά λάθη και ειδική προσπάθεια πρέπει να δίνεται για την ελαχιστοποίηση της συχνότητάς τους.

Το τελευταίο χαρακτηριστικό ευχρηστίας είναι η υποκειμενική ικανοποίηση, που αναφέρεται στο κατά πόσο είναι ευχάριστο το σύστημα. Μπορεί να μετρηθεί απλώς ρωτώντας τους χρήστες την υποκειμενική τους γνώμη. Από την προοπτική ενός μόνο χρήστη, οι απαντήσεις σε τέτοιες ερωτήσεις είναι υποκειμενικές, αλλά όταν πολλές απαντήσεις από πολλούς χρήστες χρησιμοποιηθούν και υπολογισθούν κατά μέσο όρο, τότε το αποτέλεσμα είναι μια αντικειμενική μέτρηση της ευχαρίστησης του χρήστη. Μια και ο στόχος είναι να δούμε αν το σύστημα ικανοποιεί το χρήστη, φαίνεται να είναι κατάλληλο να μετρήσουμε την υποκειμενική ικανοποίηση ρωτώντας τους χρήστες, και αυτό γίνεται στις περισσότερες μελέτες ευχρηστίας.

Η έννοια του χρήστη (user) θα πρέπει να περιλαμβάνει οποιονδήποτε του οποίου η εργασία επηρεάζεται από το σύστημα με κάποιο τρόπο.

14.Βιβλιογραφία

1) <<Αλληλεπίδραση Ανθρώπου Υπολογιστή>>
Σπύρος Συρμακέσης
Εκδόσεις: Ελληνικά Γράμματα
Α.Ε. 2003

2) <<Θεμελιώδεις Αρχές Συστημάτων Βάσεων Δεδομένων>> Τόμος Α' & Β'
R. Elmasri – S.B. Navathe
Μετάφραση Μιχάλης Χατζόπουλος
Εκδόσεις: Δίαυλος
Α.Ε. 2001

Links

http://www.sun.com/

http://www.eclipse.org/

http://www.netbeans.com/

http://www.mysql.com/

http://www.apache.com/

http://www.teradata.com/

http://www.sabre.com/

http://www.aegean.gr/

http://www.cyprusairways.com.cy/

http://www.ba.com/

http://www.getthere.com/

http://www.travelocity.com/

Παράρτημα Α

Δείγμα Κώδικα JAVA

```
public void init() {
    // Perform initializations inherited from our superclass
    super.init();
    // Perform application initialization that must complete
    // *before* managed components are initialized
    // TODO - add your own initialiation code here

    airportsOptions.setOptions(new com.sun.rave.web.ui.model.Option[] {
      new com.sun.rave.web.ui.model.Option("", "Select Airport"),
      new com.sun.rave.web.ui.model.Option("AMSTERDAM", "AMSTERDAM"),
      new com.sun.rave.web.ui.model.Option("ATHENS", "ATHENS"),
      new com.sun.rave.web.ui.model.Option("BRUSSELS", "BRUSSELS"),
      new com.sun.rave.web.ui.model.Option("FRANKFURT", "FRANKFURT"),
      new com.sun.rave.web.ui.model.Option("HERACLION", "HERACLION"),
      new com.sun.rave.web.ui.model.Option("LARNACA", "LARNACA"),
      new com.sun.rave.web.ui.model.Option("LONDON", "LONDON HHEATHROW"),
      new com.sun.rave.web.ui.model.Option("MANCHESTER", "MANCHESTER"),
      new com.sun.rave.web.ui.model.Option("MILAN", "MILAN"),
      new com.sun.rave.web.ui.model.Option("MOSCOW", "MOSCOW"),
      new com.sun.rave.web.ui.model.Option("PAFOS", "PAFOS"),
      new com.sun.rave.web.ui.model.Option("PARIS", "PARIS"),
      new com.sun.rave.web.ui.model.Option("ROME", "ROME"),
      new com.sun.rave.web.ui.model.Option("SOFIA", "SOFIA"),
      new com.sun.rave.web.ui.model.Option("THESSALONIKI", "THESSALONIKI"),
      new com.sun.rave.web.ui.model.Option("VIENNA", "VIENNA"),
      new com.sun.rave.web.ui.model.Option("ZURICH", "ZURICH")
    });

    numberOfPasengerOptions.setOptions(new com.sun.rave.web.ui.model.Option[] {
      new com.sun.rave.web.ui.model.Option("0", "0"),
      new com.sun.rave.web.ui.model.Option("1", "1"),
      new com.sun.rave.web.ui.model.Option("2", "2"),
      new com.sun.rave.web.ui.model.Option("3", "3"),
      new com.sun.rave.web.ui.model.Option("4", "4"),
      new com.sun.rave.web.ui.model.Option("5", "5")
    });

    seatClassOptions.setOptions(new com.sun.rave.web.ui.model.Option[] {
      new com.sun.rave.web.ui.model.Option("ECONOMY", "ECONOMY"),
      new com.sun.rave.web.ui.model.Option("BUSINESS", "BUSINESS"),
    });


    // <editor-fold defaultstate="collapsed" desc="Managed Component Initialization">
    // Initialize automatically managed components
```


Μαγκαφάς Γιώργος
ΑΜ 10590

```
    // *Note* - this logic should NOT be modified
    try {
        _init();
    } catch (Exception e) {
        log("Flight Initialization Failure", e);
        throw e instanceof FacesException ? (FacesException) e: new FacesException(e);
    }

    // </editor-fold>
    // Perform application initialization that must complete
    // *after* managed components are initialized
    // TODO - add your own initialization code here
}
public void preprocess() {
}

/**
 * <p>Callback method that is called just before rendering takes place.
 * This method will <strong>only</strong> be called for the page that
 * will actually be rendered (and not, for example, on a page that
 * handled a postback and then navigated to a different page).  Customize
 * this method to allocate resources that will be required for rendering
 * this page.</p>
 */
public void prerender() {
}

/**
 * <p>Callback method that is called after rendering is completed for
 * this request, if <code>init()</code> was called (regardless of whether
 * or not this was the page that was actually rendered).  Customize this
 * method to release resources acquired in the <code>init()</code>,
 * <code>preprocess()</code>, or <code>prerender()</code> methods (or
 * acquired during execution of an event handler).</p>
 */
public void destroy() {
}

public Date getMinDate(){
    return new Date();

}

public Date getMaxDate(){
    java.util.Calendar currentDateCal = new GregorianCalendar();
    java.util.Calendar minDateCal = new GregorianCalendar(currentDateCal.get(java.util.Calendar.YEAR)
0, java.util.Calendar.DECEMBER, 31);

    return minDateCal.getTime();
}
```


Μαγκαφάς Γιώργος
ΑΜ 10590

```
/**
 * <p>Return a reference to the scoped data bean.</p>
 */
protected ApplicationBean1 getApplicationBean1() {
    return (ApplicationBean1)getBean("ApplicationBean1");
}

/**
 * <p>Return a reference to the scoped data bean.</p>
 */
protected RequestBean1 getRequestBean1() {
    return (RequestBean1)getBean("RequestBean1");
}

/**
 * <p>Return a reference to the scoped data bean.</p>
 */
protected SessionBean1 getSessionBean1() {
    return (SessionBean1)getBean("SessionBean1");
}

/**
 * <p>Return a reference to the scoped data bean.</p>
 */
protected project.airway.scope.AvailabilitySessionBean getscope$AvailabilitySessionBean() {
    return (project.airway.scope.AvailabilitySessionBean)getBean("scope$AvailabilitySessionBean");
}

/**
 * <p>Return a reference to the scoped data bean.</p>
 */
protected project.airway.scope.SessionBean getscope$SessionBean() {
    return (project.airway.scope.SessionBean)getBean("scope$SessionBean");
}

/**
 * <p>Return a reference to the scoped data bean.</p>
 */
protected project.airway.scope.RequestBean getscope$RequestBean() {
    return (project.airway.scope.RequestBean)getBean("scope$RequestBean");
}

/**
 * <p>Return a reference to the scoped data bean.</p>
 */
protected project.airway.scope.ApplicationBean getscope$ApplicationBean() {
    return (project.airway.scope.ApplicationBean)getBean("scope$ApplicationBean");
}
```

```
  /**
   * <p>Return a reference to the scoped data bean.</p>
   */
  protected FlightSessionBean getscope$FlightSessionBean() {
     return (FlightSessionBean)getBean("scope$FlightSessionBean");
  }

/   public ObjectDataProvider getCustomerOdp(){
/      return getScope$CustomerPageSessionBean().getCustomerOdp();
/   }

  public String searchBtn_action() {

     String departAir =  (String)this.departAirDd.getSelected();
     String arriveAir =  (String)this.arriveAirDd.getSelected ();
     String departDate = (String)this.departCalendary.getSelectedDate().toString();
     String returnDate = (String)this.returnCalentary.getSelectedDate().toString();
     String seatClass = (String)this.seatClassDd.getSelected();
     String adults = (String)this.adultsDd.getSelected();
     String children = (String)this.childrenDd.getSelected();
     String infants = (String)this.infantsDd.getSelected();


;etscope$FlightSessionBean().setFlightValues(departAir,arriveAir,departDate,returnDate,seatClass,adults,chi
dren,infants);

     return "availability";
  }
```


Μαγκαφάς Γιώργος
ΑΜ 10590

```
*
* Availability.java
*
* Created on May 18, 2007, 1:00 AM
* Copyright mangafas
*/
ackage project.airway.page;

mport com.sun.data.provider.impl.ObjectListDataProvider;
mport com.sun.rave.web.ui.appbase.AbstractPageBean;
mport com.sun.rave.web.ui.component.Body;
mport com.sun.rave.web.ui.component.Button;
mport com.sun.rave.web.ui.component.Form;
mport com.sun.rave.web.ui.component.Head;
mport com.sun.rave.web.ui.component.Html;
mport com.sun.rave.web.ui.component.Label;
mport com.sun.rave.web.ui.component.Link;
mport com.sun.rave.web.ui.component.Page;
mport com.sun.rave.web.ui.component.PanelLayout;
mport com.sun.rave.web.ui.component.RadioButton;
mport com.sun.rave.web.ui.component.StaticText;
mport com.sun.rave.web.ui.component.Table;
mport com.sun.rave.web.ui.component.TableColumn;
mport com.sun.rave.web.ui.component.TableRowGroup;
mport javax.faces.FacesException;
mport javax.faces.component.html.HtmlPanelGrid;
mport project.airway.ApplicationBean1;
mport project.airway.RequestBean1;
mport project.airway.SessionBean1;

 public Availability() {
 }

 /**
  * <p>Callback method that is called whenever a page is navigated to,
  * either directly via a URL, or indirectly via page navigation.
  * Customize this method to acquire resources that will be needed
  * for event handlers and lifecycle methods, whether or not this
  * page is performing post back processing.</p>
  *
  * <p>Note that, if the current request is a postback, the property
  * values of the components do <strong>not</strong> represent any
  * values submitted with this request.  Instead, they represent the
  * property values that were saved for this view when it was rendered.</p>
  */
 public void init() {
    // Perform initializations inherited from our superclass
    super.init();
```

```
    // Perform application initialization that must complete
    // *before* managed components are initialized
    // TODO - add your own initialiation code here

    // <editor-fold defaultstate="collapsed" desc="Managed Component Initialization">
    // Initialize automatically managed components
    // *Note* - this logic should NOT be modified
    try {
      _init();
    } catch (Exception e) {
      log("Availability Initialization Failure", e);
      throw e instanceof FacesException ? (FacesException) e: new FacesException(e);
    }

    // </editor-fold>
    // Perform application initialization that must complete
    // *after* managed components are initialized
    // TODO - add your own initialization code here
    String st;
    this.flightSelectSt.setText("Step 1: ROUTE SELECTION");
    this.departAirSt.setText("From : " + (String)getscope$FlightSessionBean().getDepartAirport());
    this.arrivalAirSt.setText("To : " + (String)getscope$FlightSessionBean().getArrivalAirport());
    this.departDateSt.setText("Departure : 27/06/2007");// +
String)getscope$FlightSessionBean().getDepartureDate());
    this.returnDateSt.setText("Return    : 28/06/2007");//+ (
tring)getscope$FlightSessionBean().getReturnDate());
    this.seatClassSt.setText("Class : " + (String)getscope$FlightSessionBean().getSeatClass());
    this.adultsSt.setText("Adults : " + (String)getscope$FlightSessionBean().getAdults());
    this.childrenSt.setText("Children : " + (String)getscope$FlightSessionBean().getChildrens());
    this.infantsSt.setText("Infants : " + (String)getscope$FlightSessionBean().getInfants());

    //table1
    this.staticText9.setText("flight");
  }
ublic void preprocess() {
  }

  /**
   * <p>Callback method that is called just before rendering takes place.
   * This method will <strong>only</strong> be called for the page that
   * will actually be rendered (and not, for example, on a page that
   * handled a postback and then navigated to a different page).  Customize
   * this method to allocate resources that will be required for rendering
   * this page.</p>
   */
  public void prerender() {
  }

  /**
   * <p>Callback method that is called after rendering is completed for
```


Μαγκαφάς Γιώργος
ΑΜ 10590

```
 * this request, if <code>init()</code> was called (regardless of whether
 * or not this was the page that was actually rendered). Customize this
 * method to release resources acquired in the <code>init()</code>,
 * <code>preprocess()</code>, or <code>prerender()</code> methods (or
 * acquired during execution of an event handler).</p>
 */
public void destroy() {
}

public ObjectListDataProvider getDepartureFlightsOldp(){
    return getscope$AvailabilitySessionBean().getDepartureFlightsOldp();
}

public ObjectListDataProvider getReturnFlightsOldp(){
    return getscope$AvailabilitySessionBean().getReturnFlightsOldp();
}

public String goBackBtn_action() {
    // TODO: Process the button click action. Return value is a navigation
    // case name where null will return to the same page.

    return "goback";
}
```


Μαγκαφάς Γιώργος
ΑΜ 10590

Παράρτημα Β

Δείγμα Κώδικα JSP

```
xml version="1.0" encoding="UTF-8"?>
;p:root version="1.2" xmlns:f="http://java.sun.com/jsf/core" xmlns:h="http://java.sun.com/jsf/html"
ılns:ig="http://www.infragistics.com/faces/netadvantage"
xmlns:jsp="http://java.sun.com/JSP/Page" xmlns:ui="http://www.sun.com/web/ui">
<jsp:directive.page contentType="text/html;charset=UTF-8" pageEncoding="UTF-8"/>
<f:view>
  <ui:page binding="#{page$Availability.page1}" id="page1">
    <ui:html binding="#{page$Availability.html1}" id="html1">
      <ui:head binding="#{page$Availability.head1}" id="head1">
        <ui:link binding="#{page$Availability.link1}" id="link1" url="/resources/stylesheet.css"/>
      </ui:head>
      <ui:body binding="#{page$Availability.body1}" id="body1" style="-rave-layout: grid">
        <jsp:directive.include file="../fragment/HeaderFragment2.jspf"/>
        <ui:form binding="#{page$Availability.form1}" id="form1">
          <ui:panelLayout binding="#{page$Availability.layoutPanel1}" id="layoutPanel1"
            style="height: 456px; position: relative; -rave-layout: grid" styleClass="pageGridPanel">
            <ui:panelLayout binding="#{page$Availability.inform1Lp}" id="inform1Lp"
ıelLayout="flow" style="height: 111px; left: 0px; top: 0px; position: absolute; width: 769px; -rave-layout:
ıd">
              <h:panelGrid binding="#{page$Availability.gridPanel4}" columns="3" id="gridPanel4"
le="height: 96px; left: 0px; top: 0px; position: absolute; width: 100%">
                <ui:panelLayout binding="#{page$Availability.informLp}" id="informLp" style="border-
dth: 1px; border-style: solid; background-image: url(../resources/inform.gif); height: 100px; width: 233px">
                  <ui:staticText binding="#{page$Availability.flightSelectSt}" id="flightSelectSt"
ıle="left: 10px; top: 2px; position: absolute"/>
                  <ui:staticText binding="#{page$Availability.departAirSt}" id="departAirSt" style="left:
ж; top: 22px; position: absolute"/>
                  <ui:staticText binding="#{page$Availability.arrivalAirSt}" id="arrivalAirSt" style="left:
3px; top: 23px; position: absolute"/>
                  <ui:staticText binding="#{page$Availability.seatClassSt}" id="seatClassSt" style="left:
4px; top: 48px; position: absolute"/>
                  <ui:staticText binding="#{page$Availability.departDateSt}" id="departDateSt"
ıle="left: 9px; top: 43px; position: absolute"/>
                  <ui:staticText binding="#{page$Availability.returnDateSt}" id="returnDateSt"
ıle="left: 10px; top: 63px; position: absolute"/>
                  <ui:staticText binding="#{page$Availability.adultsSt}" id="adultsSt" style="left: 10px;
ı: 82px; position: absolute"/>
                  <ui:staticText binding="#{page$Availability.childrenSt}" id="childrenSt" style="left:
ıx; top: 82px; position: absolute"/>
                  <ui:staticText binding="#{page$Availability.infantsSt}" id="infantsSt" style="left:
2px; top: 82px; position: absolute"/>
                </ui:panelLayout>
                <ui:panelLayout binding="#{page$Availability.inform2Lp}" id="inform2Lp"
ıelLayout="flow" style="border-width: 1px; border-style: solid; background-image:
```

```
l(../resources/informDesable.gif); height: 100px; width: 235px"/>
                    <ui:panelLayout binding="#{page$Availability.inform3Lp}" id="inform3Lp"
nelLayout="flow" style="border-width: 1px; border-style: solid; background-image:
l(../resources/informDesable.gif); height: 100px; width: 235px"/>
                  </h:panelGrid>
                </ui:panelLayout>
                <ui:panelLayout binding="#{page$Availability.layoutPanel2}" id="layoutPanel2"
nelLayout="flow" style="border-width: 1px; border-style: solid; background-color: rgb(230, 230, 230); height:
4px; line-height: normal; left: -72px; top: 128px; position: absolute; text-indent: 1px; width: 838px; -rave-layout:
id">
                  <ui:label binding="#{page$Availability.label1}" id="label1"
                    style="border-width: 1px; border-style: solid; background-color: rgb(200, 220, 220); height:
px; left: 41px; top: -10px; position: absolute; width: 132px" text=" DEPARTURE FLIGHTS"/>
                  <h:panelGrid binding="#{page$Availability.gridPanel1}" columns="4" id="gridPanel1"
                    style="height: 39px; left: 26px; top: 5px; position: absolute" width="367">
                    <ui:staticText binding="#{page$Availability.staticText1}" id="staticText1"
                      style="font-size: 12px; font-weight: bold; height: 23px; width: 45px" text="From:"/>
                    <ui:staticText binding="#{page$Availability.staticText2}" id="staticText2"
yle="background-color: #dddfdf" text="&amp;quot;Athens&amp;quot;"/>
                    <ui:staticText binding="#{page$Availability.staticText3}" id="staticText3"
                      style="font-size: 12px; font-weight: bold; height: 23px; width: 45px" text="To:"/>
                    <ui:staticText binding="#{page$Availability.staticText4}" id="staticText4"
yle="background-color: rgb(221, 223, 223)" text="&amp;quot;Larnaca&amp;quot;"/>
                  </h:panelGrid>
                  <h:panelGrid binding="#{page$Availability.tableGp}" id="tableGp" style="left: 1px; top:
4px; position: absolute" width="815">
                    <ui:table augmentTitle="false" binding="#{page$Availability.table1}" id="table1"
te="true" style="" width="100%">
                      <ui:tableRowGroup binding="#{page$Availability.tableRowGroup1}"
="tableRowGroup1"
                        sourceData="#{page$Availability.departureFlightsOldp}" sourceVar="currentRow">
                        <ui:tableColumn align="center" binding="#{page$Availability.tableColumn1}"
="tableColumn1" width="40">
                          <ui:radioButton binding="#{page$Availability.radioButton1}" id="radioButton1"
me="radioButton-group-tableColumn1"/>
                        </ui:tableColumn>
                        <ui:tableColumn align="center" binding="#{page$Availability.tableColumn2}"
aderText="Flight Name"
                          id="tableColumn2" style="width: 50px">
                          <ui:staticText binding="#{page$Availability.staticText9}" id="staticText9"/>
                        </ui:tableColumn>
                        <ui:tableColumn align="center" binding="#{page$Availability.tableColumn3}"
aderText="From" id="tableColumn3">
                          <ui:staticText binding="#{page$Availability.staticText11}" id="staticText11"/>
                        </ui:tableColumn>
                        <ui:tableColumn align="center" binding="#{page$Availability.tableColumn4}"
aderText="To" id="tableColumn4">
                          <ui:staticText binding="#{page$Availability.staticText10}" id="staticText10"/>
                        </ui:tableColumn>
                        <ui:tableColumn align="center" binding="#{page$Availability.tableColumn5}"
```

```
aderText="Aircraft Type" id="tableColumn5">
                        <ui:staticText binding="#{page$Availability.staticText22}" id="staticText22"/>
                    </ui:tableColumn>
                    <ui:tableColumn align="center" binding="#{page$Availability.tableColumn6}"
aderText="Seat Catecory" id="tableColumn6">
                        <ui:staticText binding="#{page$Availability.staticText23}" id="staticText23"/>
                    </ui:tableColumn>
                    <ui:tableColumn align="center" binding="#{page$Availability.tableColumn7}"
aderText="Departure (logal time)" id="tableColumn7">
                        <ui:staticText binding="#{page$Availability.staticText24}" id="staticText24"/>
                    </ui:tableColumn>
                    <ui:tableColumn align="center" binding="#{page$Availability.tableColumn8}"
aderText="Arrival (local time)" id="tableColumn8">
                        <ui:staticText binding="#{page$Availability.staticText25}" id="staticText25"/>
                    </ui:tableColumn>
                    <ui:tableColumn align="center" binding="#{page$Availability.tableColumn9}"
aderText="Duration" id="tableColumn9">
                        <ui:staticText binding="#{page$Availability.staticText26}" id="staticText26"/>
                    </ui:tableColumn>
                    <ui:tableColumn align="center" binding="#{page$Availability.tableColumn10}"
aderText="Price" id="tableColumn10">
                        <ui:staticText binding="#{page$Availability.staticText27}" id="staticText27"/>
                    </ui:tableColumn>
                </ui:tableRowGroup>
            </ui:table>
        </h:panelGrid>
    </ui:panelLayout>
    <ui:panelLayout binding="#{page$Availability.layoutPanel3}" id="layoutPanel3"
anelLayout="flow" style="border-width: 1px; border-style: solid; background-color: rgb(230, 230, 230); height:
18px; line-height: normal; left: -72px; top: 288px; position: absolute; text-indent: 1px; width: 838px; -rave-layout:
rid">
        <h:panelGrid binding="#{page$Availability.gridPanel2}" columns="4" id="gridPanel2"
            style="height: 39px; left: 26px; top: 3px; position: absolute" width="367">
            <ui:staticText binding="#{page$Availability.staticText5}" id="staticText5"
                style="font-size: 12px; font-weight: bold; height: 23px; width: 45px" text="From:"/>
            <ui:staticText binding="#{page$Availability.staticText6}" id="staticText6"
le="background-color: rgb(221, 223, 223)" text="&amp;quot;Larnaca&amp;quot;"/>
            <ui:staticText binding="#{page$Availability.staticText7}" id="staticText7"
                style="font-size: 12px; font-weight: bold; height: 23px; width: 45px" text="To:"/>
            <ui:staticText binding="#{page$Availability.staticText8}" id="staticText8"
le="background-color: #dddfdf" text="&amp;quot;Athens&amp;quot;"/>
        </h:panelGrid>
        <ui:label binding="#{page$Availability.label2}" id="label2"
            style="border-width: 1px; border-style: solid; background-color: rgb(200, 220, 220); height:
px; left: 40px; top: -10px; position: absolute; width: 125px" text="RETURN FLIGHTS"/>
        <h:panelGrid binding="#{page$Availability.gridPanel3}" id="gridPanel3" style="left: 1px;
: 31px; position: absolute" width="815">
            <ui:table augmentTitle="false" binding="#{page$Availability.table2}" id="table2"
="true" width="1189">
                <ui:tableRowGroup binding="#{page$Availability.tableRowGroup2}"
```

```
="tableRowGroup2" rows="10"
                        sourceData="#{page$Availability.returnFlightsOldp}" sourceVar="currentRow">
                        <ui:tableColumn align="center" binding="#{page$Availability.tableColumn11}"
="tableColumn11" width="40">
                          <ui:staticText binding="#{page$Availability.staticText12}" id="staticText12"
xt="#{currentRow.value['column1']}"/>
                        </ui:tableColumn>
                        <ui:tableColumn align="center" binding="#{page$Availability.tableColumn12}"
aderText="Flight Name" id="tableColumn12">
                          <ui:staticText binding="#{page$Availability.staticText13}" id="staticText13"
xt="#{currentRow.value['column2']}"/>
                        </ui:tableColumn>
                        <ui:tableColumn align="center" binding="#{page$Availability.tableColumn20}"
aderText="From" id="tableColumn20">
                          <ui:staticText binding="#{page$Availability.staticText21}" id="staticText21"
xt="Static Text"/>
                        </ui:tableColumn>
                        <ui:tableColumn align="center" binding="#{page$Availability.tableColumn13}"
eaderText="To" id="tableColumn13">
                          <ui:staticText binding="#{page$Availability.staticText14}" id="staticText14"
xt="#{currentRow.value['column3']}"/>
                        </ui:tableColumn>
                        <ui:tableColumn align="center" binding="#{page$Availability.tableColumn14}"
eaderText="Aircraft Type" id="tableColumn14">
                          <ui:staticText binding="#{page$Availability.staticText15}" id="staticText15"
xt="Static Text"/>
                        </ui:tableColumn>
                        <ui:tableColumn align="center" binding="#{page$Availability.tableColumn15}"
eaderText="Seat Category" id="tableColumn15">
                          <ui:staticText binding="#{page$Availability.staticText16}" id="staticText16"
xt="Static Text"/>
                        </ui:tableColumn>
                        <ui:tableColumn align="center" binding="#{page$Availability.tableColumn16}"
eaderText="Departyre (logal time)" id="tableColumn16">
                          <ui:staticText binding="#{page$Availability.staticText17}" id="staticText17"
xt="Static Text"/>
                        </ui:tableColumn>
                        <ui:tableColumn align="center" binding="#{page$Availability.tableColumn17}"
aderText="Arrival (logal time)"
                          id="tableColumn17" width="99">
                          <ui:staticText binding="#{page$Availability.staticText18}" id="staticText18"
t="Static Text"/>
                        </ui:tableColumn>
                        <ui:tableColumn align="center" binding="#{page$Availability.tableColumn18}"
aderText="Duration" id="tableColumn18" width="56">
                          <ui:staticText binding="#{page$Availability.staticText19}" id="staticText19"
t="Static Text"/>
                        </ui:tableColumn>
                        <ui:tableColumn align="center" binding="#{page$Availability.tableColumn19}"
aderText="Price" id="tableColumn19" width="42">
```

```
                                <ui:staticText binding="#{page$Availability.staticText20}" id="staticText20"
xt="Static Text"/>
                            </ui:tableColumn>
                        </ui:tableRowGroup>
                    </ui:table>
                </h:panelGrid>
            </ui:panelLayout>
            <ui:button action="step3" binding="#{page$Availability.step3Btn}" id="step3Btn"
              style="font-weight: bold; height: 29px; left: 551px; top: 432px; position: absolute" text="STEP
: Personal &amp;amp; Contact Info"/>
            <ui:button action="#{page$Availability.goBackBtn_action}"
inding="#{page$Availability.goBackBtn}" id="goBackBtn"
              style="font-weight: bold; height: 29px; left: -1px; top: 432px; position: absolute; width: 119px"
ext="Go Back to  STEP 1"/>
        </ui:panelLayout>
    </ui:form>
  </ui:body>
 </ui:html>
</ui:page>
</f:view>
</jsp:root>
```