Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών

Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων

Πτυχιακή εργασία

Τίτλος: Σύστημα Παροχής Πιστωτικών Υπηρεσιών

Επίθετο: Καλογεράκης

Όνομα: Μανώλης

Α.Μ:3491

Επιβλέπων καθηγήτης: Νίκος Παπαδάκης

Επιτροπή Αξιολόγησης:

Ημερομηνία παρουσίασης:

Abstract

An integral part of the IT system today is the creation and management of databases where data is stored. Each company and enterprise in any competitive range and placed should be able to keep the information on data relating to:

- customers,
- suppliers,
- employees,
- revenue - expense reports, and more.

This document deals with the description and analysis of dissertation thesis on the construction of an integrated data storage system for the provision of credit services for the needs of a company. It is an informational database system that adopts modern web programming techniques and full automated database management.

The following chapters develop the technologies and tools chosen to complete the dissertation. Below is a full report about the basic necessary theories for internet programming and the system that has been implemented.

Σύνοψη

Αναπόσπαστο κομμάτι στο τομέα των πληροφοριακών συστημάτων σήμερα είναι η δημιουργία και η διαχείριση βάσεων όπου αποθηκεύονται δεδομένα. Κάθε εταιρία και επιχείρηση σε όποιο εύρος ανταγωνιστικότητας και να τοποθετείται πρέπει να είναι σε θέση να διατηρεί τις πληροφορίες για δεδομένα που αφορούν :

- πελάτες,
- προμηθευτές,
- υπαλλήλους,
- αναφορές εσόδων – εξόδων και άλλα.

Το παρών έγγραφο αφορά την περιγραφή και την ανάλυση πτυχιακής εργασίας με θέμα την κατασκευή ολοκληρωμένου συστήματος αποθήκευσης δεδομένων για την παροχή πιστωτικών υπηρεσιών για τις ανάγκες μιας εταιρίας. Πρόκειται για ένα πληροφοριακό σύστημα βάσης δεδομένων που υιοθετεί τις σύγχρονες τεχνικές προγραμματισμού στο διαδίκτυο και την πλήρη διαχείριση αυτοματοποιημένης βάσης.

Στα παρακάτω κεφάλαια αναπτύσσονται οι τεχνολογίες και τα εργαλεία που επιλέχθηκαν για την περάτωση της πτυχιακής εργασίας, οι βασικές πλην απαραίτητες θεωρίες για τον προγραμματισμό στο διαδίκτυο και γίνεται πλήρη αναφορά στο σύστημα που υλοποιήθηκε.

Table of Contents

Εικόνες

Πίνακας πινάκων

1 Εισαγωγή

Στα παρακάτω κεφάλαια παραθέτονται και αναλύονται οι τρόποι που επιλύουν το θέμα που πραγματεύεται η παρούσα πτυχιακή εργασία. Πρόκειται για ένα σύστημα παροχής πιστωτικών υπηρεσιών σε μια εταιρία. Οι τεχνολογίες που επιλέχθηκαν είναι συνοπτικά οι παρακάτω:

- HTML, για την κατασκευή της διεπαφής ώστε να μπορεί ο χρήστης να έχει πρόσβαση στην εμφάνιση και την επεξεργασία των πληροφοριών
- Javascript, για κάποιες εκάστοτε λειτουργίες
- PHP, για την διάδραση βάσης και ιστοσελίδας
- MySQL, για την δημιουργία και την διαχείριση της βάσης δεδομένων.
- CSS, για την μορφοποίηση των βασικών περιεχομένων της ιστοσελίδα

1.1 Περίληψη

Συνοπτικά για τις ανάγκες της εργασίας απαιτείται να διατηρούνται πληροφορίες στη βάση:

1. Για κάθε πελάτη
   i. Όνομα και επώνυμο
   ii. Διεύθυνση
   iii. Ηλικία
   iv. Τηλέφωνο
   v. Ημερομηνία απόκτησης διπλώματος
   vi. Αριθμός ταυτότητας
2. Για κάθε ασφαλιστήριο
   i. Κωδικός
   ii. Ημερομηνία δημιουργίας
   iii. Μηνιαίο ποσό
   iv. Κατηγορία (ανάλογα το ποσό)
3. Για κάθε δάνειο
   i. Κωδικός

ii. Ημερομηνία δημιουργίας

iii. Ποσό δανείου

iv. Επιτόκιο

Θα δίνεται η δυνατότητα στον χρήστη του συστήματος να ενημερώνεται για τις πληροφορίες που τον ενδιαφέρουν με σχετικές αναφορές όπως:

- Πελάτες που έχουν καθυστερήσει τη δόση
- Πελάτες που έχουν οφειλές
- Πελάτες που είναι συνεχώς συνεπείς
- Πελάτες που έχουν περισσότερα από κάποιο αριθμό ασφαλιστηρίων
- Αριθμός των πελατών ανά ασφαλιστήριο
- Συγκεντρωτική αναφορά για κάθε μήνα για τα δάνεια

1.2 Σκοπός και Στόχοι Εργασίας

Σκοπός της εργασίας είναι η υλοποίηση του συστήματος με τις παραπάνω πληροφορίες σε συνδυασμό με ένα φιλικό προς τον χρήστη περιβάλλον.

Βασικός στόχος της εργασίας είναι η επαφή με τις σύγχρονες μεθόδους δημιουργίας και διαχείρισης βάσεων δεδομένων. Η τριβή με την κατασκευή διεπαφής που θα συνδέει τον χρήστη με την βάση δεδομένων ώστε να μπορεί να διαχειριστεί τις επιθυμητές πληροφορίες.

1.3 Δομή Εργασίας

Στο κεφάλαιο 2 γίνεται μια εκτενής αναφορά στη γλώσσα προγραμματισμού java που αποτελεί τον κορμό του συστήματος που υλοποιήθηκε. Στο 3ο κεφάλαιο παραθέτονται οι βασικές τεχνικές λειτουργίες που απαιτούνται για να εγκατασταθεί το σύστημα τοπικά στον υπολογιστή. Στο κεφάλαιο 4 παρουσιάζεται η δομή και ο τρόπος χρήσης της βάσης δεδομένων. Παραθέτονται οι σελίδες που πλοηγείται ο χρήστης και κάποιοι κώδικες που τις υλοποιούν. Στο τελευταίο κεφάλαιο αναφέρονται οι πηγές από όπου αντλήθηκαν οι περισσότερες πληροφορίες που περιέχει η συγκεκριμένη αναφορά της πτυχιακής εργασίας.

2 Μεθοδολογία Υλοποίησης

Στο συγκεκριμένο κεφάλαιο αναλύονται βασικές θεωρίες για την ανάπτυξη μιας βάσης δεδομένων και το διαδίκτυο. Αναφέρονται ιστορικά στοιχεία για την πορεία που ακολούθησε η επιστήμη του προγραμματισμού στο διαδίκτυο. Ακόμη θα γνωρίσουμε τον τρόπο που επιλέγουν οι σύγχρονοι προγραμματιστές να αναπτύσσουν ιστοσελίδες και βάσεις αποθήκευσης δεδομένων.

2.1 Μέθοδος Ανάλυσης και Ανάπτυξης της Πτυχιακής

Στον παρακάτω πίνακα συνοψίζονται οι τεχνολογίες που επιλέχθηκαν για την περάτωση της πτυχιακής εργασίας και ο εκάστοτε στόχος που υλοποιούν.

Στόχος	**Τεχνολογία**
Κατασκευή ιστοσελίδας	HTML
Μορφοποίηση του περιεχομένου	CSS(Cascading Style Sheets)
Βάση δεδομένων	MySQL
Διασύνδεση βάσης και ιστοσελίδας.	Java

Table 1 Τεχνολογία για κάθε στόχο

2.2 Η προέλευση της Java

Η Java είναι μια γενική γλώσσα προγραμματισμού ηλεκτρονικών υπολογιστών που είναι

- concurrent
- βασισμένη σε κλάσεις,
- προσανατολισμένη σε αντικείμενα και
- έχει σχεδιαστεί ειδικά για να έχει όσο το δυνατόν λιγότερες εξαρτήσεις εφαρμογής.

Σκοπός είναι να επιτρέψει στους προγραμματιστές εφαρμογών να "γράψουν μία φορά για να τρέξει οπουδήποτε" (WORA), που σημαίνει ότι ο μεταγλωττισμένος κώδικας Java μπορεί να τρέξει σε όλες τις πλατφόρμες που υποστηρίζουν την Java χωρίς την ανάγκη αναδιπλασιασμού. Οι εφαρμογές Java συνήθως μεταγλωττίζονται σε bytecode που μπορούν να τρέξουν σε οποιαδήποτε εικονική μηχανή Java (JVM) ανεξάρτητα από την αρχιτεκτονική του υπολογιστή. Από το 2016, η Java είναι μια από τις πιο δημοφιλείς γλώσσες προγραμματισμού

που χρησιμοποιούνται, ειδικά για web εφαρμογές client-server, με 9 εκατομμύρια προγραμματιστές. Η Java αναπτύχθηκε αρχικά από τον James Gosling στη Sun Microsystems (η οποία από τότε έχει αποκτηθεί από την Oracle Corporation) και κυκλοφόρησε το 1995 ως βασική συνιστώσα της πλατφόρμας Java της Sun Microsystems. Η γλώσσα αποδίδει μεγάλο μέρος της σύνταξής της στη C και τη C ++, αλλά έχει λιγότερες εγκαταστάσεις χαμηλού επιπέδου από ότι και οι δύο.

Οι πρωτότυπες μεταγλωττιστές Java, οι εικονικές μηχανές και οι βιβλιοθήκες τάξεων της αρχικής εφαρμογής και της αναφοράς κυκλοφόρησαν αρχικά από την Sun υπό άδειες ιδιοκτησίας. Από τον Μάιο του 2007, σύμφωνα με τις προδιαγραφές της Κοινοτικής Διαδικασίας Java, η Sun επανεξέτασε τις περισσότερες από τις τεχνολογίες Java της βάσει της Γενικής Δημόσιας Άδειας GNU. Άλλοι έχουν επίσης αναπτύξει εναλλακτικές εφαρμογές αυτών των τεχνολογιών Sun, όπως το GNU Compiler για Java (μεταγλωττιστής bytecode), το GNU Classpath (τυπικές βιβλιοθήκες) και το IcedTea-Web (plugin του προγράμματος περιήγησης για μικροεφαρμογές).

Η τελευταία έκδοση είναι η Java 9, η οποία κυκλοφόρησε στις 21 Σεπτεμβρίου 2017 και είναι μία από τις δύο εκδόσεις που υποστηρίζονται δωρεάν από την Oracle. Οι εκδόσεις νωρίτερα από την Java 8 υποστηρίζονται τόσο από την Oracle όσο και από άλλες εταιρείες σε εμπορική βάση.

Illustration 1: Java logo

Ιστορικά

Ο James Gosling, ο Mike Sheridan και ο Patrick Naughton ξεκίνησαν το πλάνο γλώσσας προγραμματισμού Java τον Ιούνιο του 1991. Η Java σχεδιάστηκε αρχικά για διαδραστική τηλεόραση, αλλά ήταν πολύ προηγμένη για τη βιομηχανία ψηφιακής καλωδιακής τηλεόρασης την εποχή εκείνη . Η γλώσσα αρχικά ονομάζεται Oak από μια βελανιδιά που βρισκόταν έξω από το γραφείο του Gosling. Αργότερα το έργο πήγε με το όνομα Green και τελικά μετονομάστηκε σε Java, από coffee Java. Ο Gosling σχεδίασε Java με μια σύνταξη τύπου C / C ++ που θα έβρισκαν οικεία οι προγραμματιστές του συστήματος και των εφαρμογών.

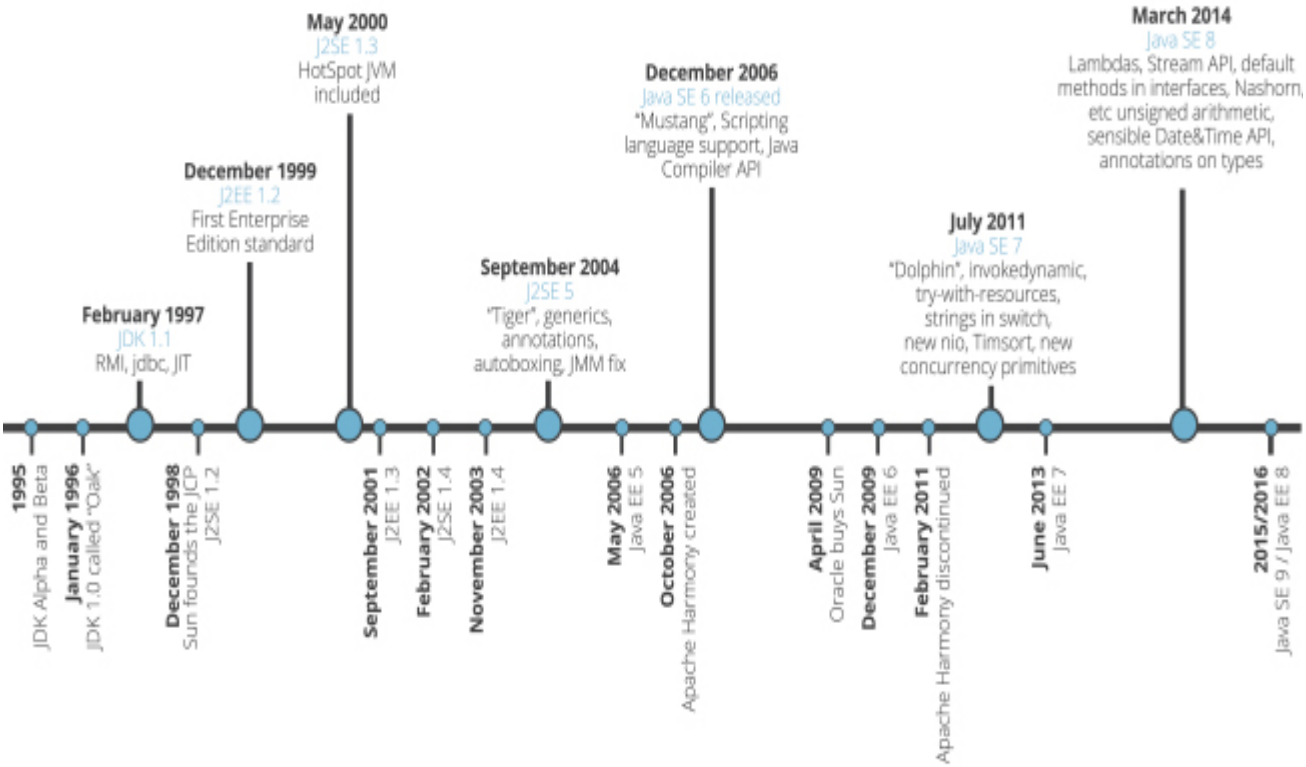


Illustration 2: Java History

Η Sun Microsystems κυκλοφόρησε την πρώτη δημόσια εφαρμογή ως Java 1.0 το 1995. Υποσχέθηκε ότι θα ικανοποιεί το "γράψτε μία φορά, τρέξτε οπουδήποτε" (WORA), παρέχοντας δωρεάν χρόνο εκτέλεσης σε δημοφιλείς πλατφόρμες. Αρκετά ασφαλής και με ρυθμιζόμενη ασφάλεια, επέτρεψε περιορισμούς πρόσβασης σε δίκτυο και αρχεία. Τα μεγάλα προγράμματα περιήγησης ιστού σύντομα ενσωμάτωσαν τη δυνατότητα εκτέλεσης μικροεφαρμογών Java μέσα σε ιστοσελίδες και η Java γρήγορα έγινε δημοφιλής. Ο μεταγλωττιστής Java 1.0 ανατυπώθηκε στην Java από τον Arthur van Hoff για να συμμορφωθεί αυστηρά με τις προδιαγραφές γλώσσας Java 1.0. Με την εμφάνιση της Java 2 (που κυκλοφόρησε αρχικά ως J2SE 1.2 το Δεκέμβριο 1998 - 1999), οι νέες εκδόσεις είχαν πολλαπλές διαμορφώσεις που δημιουργήθηκαν για διαφορετικούς τύπους πλατφορμών. Το J2EE περιελάμβανε τεχνολογίες και API για επιχειρηματικές εφαρμογές που συνήθως εκτελούνται σε περιβάλλοντα διακομιστών, ενώ τα J2ME διαθέτουν API βελτιστοποιημένα για κινητές εφαρμογές. Η έκδοση για υπολογιστές μετονομάστηκε σε J2SE. Το 2006, για εμπορικούς σκοπούς, η Sun μετονομάστηκε σε νέες εκδόσεις J2 όπως Java EE, Java ME και Java SE, αντίστοιχα.

Το 1997, η Sun Microsystems προσέγγισε τον οργανισμό προτύπων ISO / IEC JTC 1 και αργότερα την Ecma International για να επισημοποιήσει την Java, αλλά σύντομα αποσύρθηκε από τη διαδικασία. Η Java παραμένει ένα de facto πρότυπο, που ελέγχεται μέσω της κοινοτικής διαδικασίας Java. Κάποτε, η Sun έκανε τις περισσότερες από τις υλοποιήσεις της Java διαθέσιμες χωρίς χρέωση, παρά την κατάσταση του ιδιόκτητου λογισμικού τους. Η Sun πραγματοποίησε έσοδα από την Java μέσω της πώλησης αδειών για εξειδικευμένα προϊόντα όπως το Java Enterprise System.

Στις 13 Νοεμβρίου 2006, η Sun απελευθέρωσε μεγάλο μέρος της εικονικής μηχανής Java (JVM) ως ελεύθερο λογισμικό ανοικτού κώδικα (FOSS), υπό τους όρους της Γενικής Δημόσιας Άδειας GNU (GPL). Στις 8 Μαΐου 2007, η Sun ολοκλήρωσε τη διαδικασία, κάνοντας όλο τον βασικό κώδικα του JVM διαθέσιμο υπό όρους ελεύθερου λογισμικού / ανοικτού πηγαίου κώδικα, εκτός από ένα μικρό τμήμα του κώδικα στον οποίο ο Sun δεν κατοικούσε τα πνευματικά δικαιώματα.

Ο αντιπρόεδρος της Sun, Rich Green, δήλωσε ότι ο ιδανικός ρόλος της Sun σε σχέση με την Java ήταν ως «ευαγγελιστής» . Μετά την εξαγορά της Sun Microsystems από την Oracle Corporation το 2009-10, η Oracle χαρακτηρίστηκε ως ο "διαχειριστής της τεχνολογίας Java με μια αμείλικτη δέσμευση για την προώθηση μιας κοινότητας συμμετοχής και διαφάνειας". Αυτό δεν εμπόδισε την Oracle να ασκήσει αγωγή εναντίον της Google σύντομα μετά από αυτήν για

τη χρήση της Java μέσα στο Android SDK. Το λογισμικό Java λειτουργεί σε όλα, από φορητούς υπολογιστές έως κέντρα δεδομένων, κονσόλες παιχνιδιών έως επιστημονικούς υπερυπολογιστές. Στις 2 Απριλίου 2010, ο James Gosling παραιτήθηκε από την Oracle.

Αρχές

Υπήρχαν πέντε πρωταρχικοί στόχοι στη δημιουργία της γλώσσας Java:

1. Πρέπει να είναι "απλή, αντικειμενοστρεφής και οικεία".
2. Πρέπει να είναι "ανθεκτικό και ασφαλές".
3. Πρέπει να είναι "ουδέτερο στην αρχιτεκτονική και φορητό".
4. Πρέπει να εκτελείται με "υψηλή απόδοση".
5. Πρέπει να "ερμηνευτεί, να σπειρωθεί και να γίνει δυναμική".

Εκδόσεις

Από το 2017, και οι δύο Java 8 και 9 υποστηρίζονται επίσημα. Κυριότερες εκδόσεις της Java, μαζί με τις ημερομηνίες κυκλοφορίας τους:

- JDK 1.0 (23 Ιανουαρίου 1996) [39]
- JDK 1.1 (19 Φεβρουαρίου 1997)
- J2SE 1.2 (8 Δεκεμβρίου 1998)
- J2SE 1.3 (8 Μαΐου 2000)
- J2SE 1.4 (6 Φεβρουαρίου 2002)
- J2SE 5.0 (30 Σεπτεμβρίου 2004)
- Java SE 6 (11 Δεκεμβρίου 2006)
- Java SE 7 (28 Ιουλίου 2011)
- Java SE 8 (18 Μαρτίου 2014)

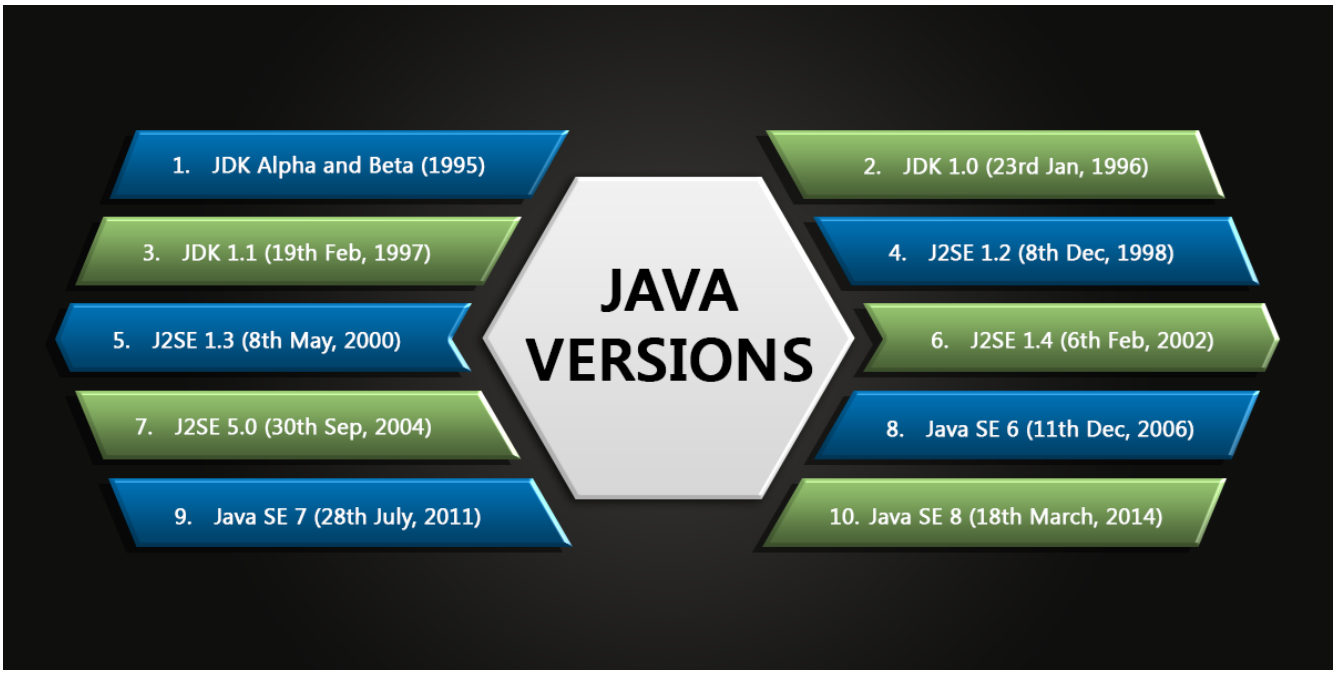


Illustration 3: Java versions

- Java SE 9 (21 Σεπτεμβρίου 2017)

Πλατφόρμα

Η πλατφόρμα Java είναι μια σειρά προγραμμάτων που διευκολύνουν την ανάπτυξη και εκτέλεση προγραμμάτων γραμμένα στη γλώσσα προγραμματισμού Java. Μια πλατφόρμα Java περιλαμβάνει:

- μια μηχανή εκτέλεσης (που ονομάζεται εικονική μηχανή),
- έναν μεταγλωττιστή
- ένα σύνολο βιβλιοθηκών
- ενδέχεται να υπάρχουν επιπλέον διακομιστές και
- εναλλακτικές βιβλιοθήκες που εξαρτώνται από τις απαιτήσεις.

Η Java δεν είναι συγκεκριμένη σε κανένα επεξεργαστή ή λειτουργικό σύστημα, καθώς οι πλατφόρμες Java έχουν εφαρμοστεί για ένα ευρύ φάσμα υλικού και λειτουργικών συστημάτων με σκοπό να επιτρέψουν στα προγράμματα Java να εκτελούνται ταυτόσημα σε όλα αυτά. Οι διαφορετικές πλατφόρμες στοχεύουν σε διαφορετικές κατηγορίες συσκευών και τομέων εφαρμογής:

- Κάρτα Java: Μια τεχνολογία που επιτρέπει στις μικρές εφαρμογές Java (applets) να τρέχουν με ασφάλεια σε έξυπνες κάρτες και παρόμοιες συσκευές μικρής μνήμης.

- Java ME (Micro Edition): Καθορίζει πολλά διαφορετικά σύνολα βιβλιοθηκών (γνωστά ως προφίλ) για συσκευές με περιορισμένη χωρητικότητα αποθήκευσης, οθόνης και ισχύος. Συχνά χρησιμοποιείται για την ανάπτυξη εφαρμογών για φορητές συσκευές, PDA, αποκωδικοποιητές τηλεόρασης και εκτυπωτές.
- Java SE (Standard Edition): Για γενική χρήση σε επιτραπέζιους υπολογιστές, διακομιστές και παρόμοιες συσκευές.
- Java EE (Enterprise Edition): Java SE και διάφορα API που είναι χρήσιμα για επιχειρηματικές εφαρμογές πολλαπλών επιπέδων client-server.

Η πλατφόρμα Java αποτελείται από πολλά προγράμματα, καθένα από τα οποία παρέχει ένα τμήμα των συνολικών δυνατοτήτων του. Για παράδειγμα, ο Java compiler, ο οποίος μετατρέπει τον πηγαίο κώδικα Java σε Java bytecode (μια ενδιάμεση γλώσσα για το JVM), παρέχεται ως μέρος του Java Development Kit (JDK). Το Java Runtime Environment (JRE), το οποίο συμπληρώνει το JVM με τον μεταγλωττιστή Just-in-Time (JIT), μετατρέπει τον ενδιάμεσο bytecode σε κώδικα εγγενούς μηχανής. Η πλατφόρμα Java περιλαμβάνει επίσης ένα εκτεταμένο σύνολο βιβλιοθηκών.

Τα βασικά συστατικά στοιχεία της πλατφόρμας είναι:
- ο μεταγλωττιστής γλώσσας Java,
- οι βιβλιοθήκες και
- το περιβάλλον χρόνου εκτέλεσης στο οποίο εκτελείται ο ενδιάμεσος bytecode σύμφωνα με τους κανόνες που ορίζονται στην προδιαγραφή εικονικής μηχανής.

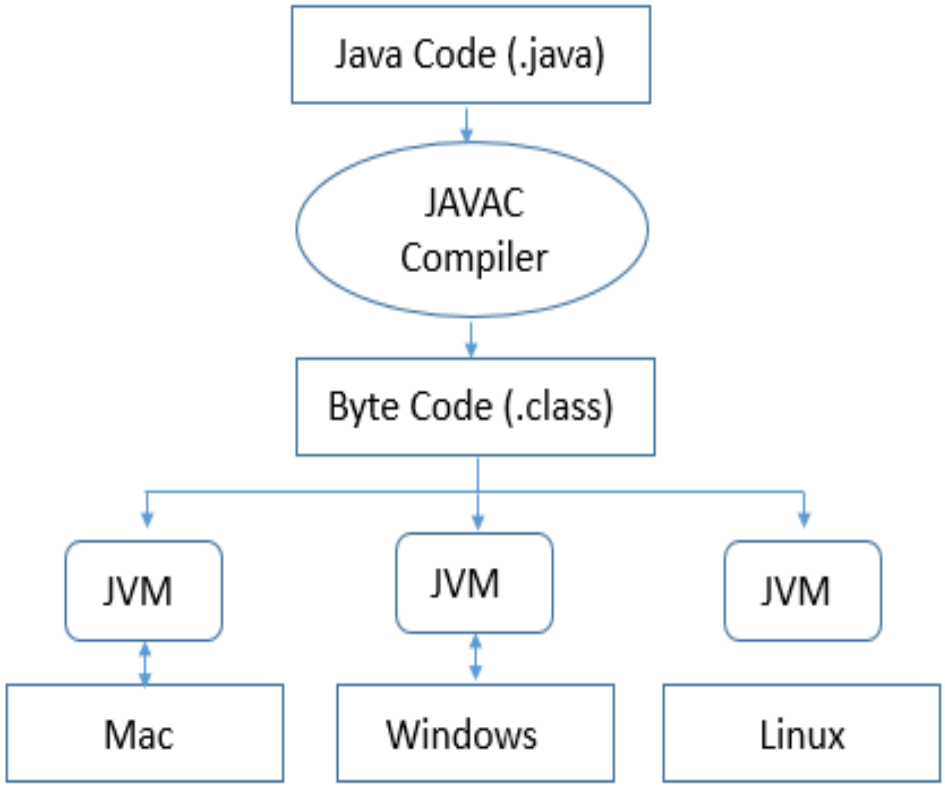


Illustration 4: Java platform

Αυτόματη διαχείριση μνήμης

Η Java χρησιμοποιεί έναν αυτόματο συλλέκτη σκουπιδιών για τη διαχείριση μνήμης στον κύκλο ζωής του αντικειμένου. Ο προγραμματιστής καθορίζει πότε δημιουργούνται τα αντικείμενα και ο χρόνος εκτέλεσης Java είναι υπεύθυνος για την ανάκτηση της μνήμης μόλις τα αντικείμενα δεν χρησιμοποιούνται πλέον. Αφού δεν παραμείνουν αναφορές σε ένα αντικείμενο, η μη πληρέστερη μνήμη καθίσταται επιλέξιμη για να ελευθερωθεί αυτόματα από τον συλλέκτη απορριμμάτων. Κάτι παρόμοιο με διαρροή μνήμης μπορεί να συμβεί αν ο κώδικας ενός προγραμματιστή κρατά μια αναφορά σε ένα αντικείμενο που δεν χρειάζεται πια, συνήθως όταν αντικείμενα που δεν χρειάζονται πλέον αποθηκεύονται σε δοχεία που είναι ακόμα σε χρήση. Εάν ονομάζονται μέθοδοι για ένα ανύπαρκτο αντικείμενο, ρίχνεται μια "εξαίρεση μηδενικού δείκτη".

Μια από τις ιδέες πίσω από το μοντέλο αυτόματης διαχείρισης μνήμης της Java είναι ότι οι προγραμματιστές μπορούν να απαλλαγούν από την επιβάρυνση που απαιτείται για τη χειρωνακτική διαχείριση μνήμης. Σε ορισμένες γλώσσες, η μνήμη για τη δημιουργία αντικειμένων κατανέμεται σιωπηρά στη στοίβα ή εκχωρείται ρητά και απομετροποιείται από τον σωρό. Στην τελευταία περίπτωση, η ευθύνη της διαχείρισης μνήμης βρίσκεται με τον

προγραμματιστή. Εάν το πρόγραμμα δεν απομακρύνει ένα αντικείμενο, εμφανίζεται διαρροή μνήμης. Εάν το πρόγραμμα προσπαθεί να αποκτήσει πρόσβαση ή να απομακρύνει μνήμη που έχει ήδη απομετρηθεί, το αποτέλεσμα είναι απροσδιόριστο και δύσκολο να προβλεφθεί και το πρόγραμμα είναι πιθανό να γίνει ασταθές ή να καταρρεύσει. Αυτό μπορεί να αντιμετωπιστεί εν μέρει με τη χρήση έξυπνων δεικτών, αλλά αυτά προσθέτουν επιβάρυνση και πολυπλοκότητα. Σημειώστε ότι η συλλογή απορριμμάτων δεν εμποδίζει διαρροές μνήμης "λογικής", δηλαδή εκείνες όπου η μνήμη εξακολουθεί να αναφέρεται αλλά ποτέ δεν χρησιμοποιείται.

Η συλλογή σκουπιδιών μπορεί να συμβεί ανά πάσα στιγμή. Στην ιδανική περίπτωση, θα συμβεί όταν ένα πρόγραμμα είναι αδρανές. Είναι εγγυημένη η ενεργοποίησή του εάν δεν υπάρχει αρκετή ελεύθερη μνήμη στον σωρό για να διαθέσει ένα νέο αντικείμενο. αυτό μπορεί να προκαλέσει προσωρινή διακοπή ενός προγράμματος. Δεν είναι δυνατή η διαχείριση ρητής μνήμης στην Java.

Η Java δεν υποστηρίζει την αριθμητική ένδειξη δείκτη C / C ++, όπου οι διευθύνσεις αντικειμένων και οι μη υπογεγραμμένοι ακέραιοι (συνήθως μεγάλοι ακέραιοι) μπορούν να χρησιμοποιηθούν εναλλακτικά. Αυτό επιτρέπει στον συλλέκτη απορριμμάτων να μεταφέρει τα αναφερόμενα αντικείμενα και εξασφαλίζει ασφάλεια και ασφάλεια τύπου.

Όπως και στην C ++ και σε ορισμένες άλλες αντικειμενοστρεφείς γλώσσες, οι μεταβλητές των πρωτότυπων τύπων δεδομένων της Java είτε αποθηκεύονται απευθείας σε πεδία (για αντικείμενα) είτε στη στοίβα (για μεθόδους) και όχι στο σωρό, όπως ισχύει και για μη πρωτόγονα δεδομένα τύπους (αλλά βλέπε ανάλυση διαφυγής). Αυτή ήταν μια συνειδητή απόφαση των σχεδιαστών της Java για λόγους απόδοσης.

Η Java περιέχει πολλούς τύπους συλλεκτών σκουπιδιών. Από προεπιλογή, το HotSpot χρησιμοποιεί τον παράλληλο συλλέκτη απορριμμάτων. Ωστόσο, υπάρχουν και αρκετοί άλλοι συλλέκτες σκουπιδιών που μπορούν να χρησιμοποιηθούν για τη διαχείριση του σωρού. Για το 90% των εφαρμογών στην Java, επαρκεί ο συλλέκτης απορριμμάτων Concurrent Mark-Sweep (CMS). Η Oracle σκοπεύει να αντικαταστήσει το CMS με τον συλλέκτη Garbage-First (G1).

2.3 Το συντακτικό της Java

Η σύνταξη της γλώσσας προγραμματισμού Java είναι το σύνολο κανόνων που καθορίζουν τον τρόπο γραφής και ερμηνείας ενός προγράμματος Java.

Η σύνταξη προέρχεται κυρίως από τη C και τη C ++. Σε αντίθεση με τη C ++, στην Java δεν υπάρχουν καθολικές λειτουργίες ή μεταβλητές, αλλά υπάρχουν στοιχεία δεδομένων τα οποία επίσης θεωρούνται ως μεταβλητές σε παγκόσμια κλίμακα. Όλος ο κώδικας ανήκει σε

τάξεις και όλες οι τιμές είναι αντικείμενα. Η μόνη εξαίρεση είναι οι πρωταρχικοί τύποι, οι οποίοι δεν αντιπροσωπεύονται από μια κλάση για λόγους απόδοσης (αν και μπορούν να μετατραπούν αυτόματα σε αντικείμενα και αντίστροφα μέσω autoboxing). Ορισμένες λειτουργίες όπως ο υπερφορτωμένος τελεστής ή οι μη υπογεγραμμένοι τύποι ακέραιων αριθμών παραλείπονται για απλοποίηση της γλώσσας και για αποφυγή πιθανών λαθών προγραμματισμού.

Η σύνταξη της Java έχει επεκταθεί σταδιακά κατά τη διάρκεια των οκτώ μεγάλων δυνατοτήτων υποστήριξης της έκδοσης JDK, όπως γενικά προγραμματισμός και λυχνίες λειτουργίας (αποκαλούμενες εκφράσεις lambda στη Java).

Αναγνωριστικά

Ένα αναγνωριστικό είναι το όνομα ενός στοιχείου στον κώδικα. Υπάρχουν ορισμένες συνήθεις συμβάσεις ονομασίας που πρέπει να ακολουθήσετε κατά την επιλογή ονομάτων για στοιχεία. Τα αναγνωριστικά της Java είναι διακριτικά πεζών-κεφαλαίων.

Ένα αναγνωριστικό μπορεί να περιέχει:

- Οποιοδήποτε χαρακτήρα Unicode που είναι ένα γράμμα (συμπεριλαμβανομένων αριθμητικών γραμμάτων όπως ρωμαϊκά ψηφία) ή ψηφίων.
- Υπογραφή νομίσματος ($).
- Χαρακτήρα στίξης (όπως _).

Ένα αναγνωριστικό δεν μπορεί:

- Να ξεκινάει με ένα ψηφίο.
- Να είναι ίδιο με μια αποκλειστική λέξη-κλειδί, μηδενική κυριολεκτική ή boolean literal.

Λέξεις κλειδιά

- abstract
- continue
- for
- new
- switch
- assert
- default
- goto
- package
- synchronized
- boolean

- do
- if
- private
- this
- break
- double
- implements
- protected
- throw
- byte
- else
- import
- public
- throws
- case
- enum
- instanceof
- return
- transient
- catch
- extends
- int
- short
- try
- char
- final
- interface
- static
- void
- class
- finally
- long
- strictfp
- volatile
- const
- float
- native

```
/**
 * This is a documentation comment.
 *
 * @author John Doe
 */
```

- super
- while

Σχόλια

Η Java έχει τρία είδη σχολίων:

- τα παραδοσιακά σχόλια,
- τα σχόλια στο τέλος της γραμμής και
- τα σχόλια τεκμηρίωσης.

Τα παραδοσιακά σχόλια, γνωστά και ως μπλοκ σχόλια, ξεκινούν με / * και τελειώνουν με * /, μπορούν να εκτεθούν σε πολλαπλές γραμμές. Αυτός ο τύπος σχολίου προέρχεται από τις C και C ++.

Τα σχόλια στα τέλη της γραμμής αρχίζουν με // και επεκτείνονται στο τέλος της τρέχουσας γραμμής. Αυτός ο τύπος σχολίου υπάρχει επίσης στη C ++ και στη σύγχρονη C.

Τα σχόλια τεκμηρίωσης στα αρχεία προέλευσης επεξεργάζονται από το εργαλείο Javadoc για τη δημιουργία τεκμηρίωσης. Αυτός ο τύπος σχόλιου είναι πανομοιότυπος με τα παραδοσιακά σχόλια, εκτός από το ότι αρχίζει με / ** και ακολουθεί τις συμβάσεις που ορίζονται από το εργαλείο Javadoc. Από τεχνική άποψη, αυτά τα σχόλια είναι ένα ιδιαίτερο είδος παραδοσιακών σχολίων και δεν καθορίζονται συγκεκριμένα στις προδιαγραφές της γλώσσας.

Δομή του προγράμματος

Οι εφαρμογές Java αποτελούνται από συλλογές κλάσεων. Οι κλάσεις υπάρχουν σε πακέτα αλλά μπορούν επίσης να ενσωματωθούν σε άλλες κατηγορίες.

Main method

```
package myapplication.mylibrary;

public class MyClass {
}
```

Κάθε εφαρμογή Java πρέπει να έχει ένα σημείο εισόδου. Αυτό ισχύει τόσο για εφαρμογές γραφικών διεπαφών όσο και για εφαρμογές κονσόλας. Το σημείο εισόδου είναι η κύρια μέθοδος. Μπορούν να υπάρχουν περισσότερες από μία κλάσεις με μια κύρια μέθοδο, αλλά η κύρια κλάση ορίζεται πάντα εξωτερικά (για παράδειγμα, σε ένα πρόδηλο αρχείο). Η μέθοδος πρέπει να είναι στατική και να περάσει τα επιχειρήματα της γραμμής εντολών ως μια σειρά από χορδές. Σε αντίθεση με τη C ++ ή τη C #, δεν επιστρέφει ποτέ μια τιμή και πρέπει να επιστρέψει void.

Πακέτα

Τα πακέτα αποτελούν μέρος ενός ονόματος κλάσης και χρησιμοποιούνται για την ομαδοποίηση και τη διάκριση ονομασμένων οντοτήτων από άλλες κλάσεις. Ένας άλλος σκοπός των πακέτων είναι η ρύθμιση της πρόσβασης σε κώδικα μαζί με τροποποιητές πρόσβασης. Για παράδειγμα, το java.io.InputStream είναι ένα πλήρες όνομα κατηγορίας για το class InputStream που βρίσκεται στο java.io του πακέτου.

Ένα πακέτο δηλώνεται στην αρχή του αρχείου με τη δήλωση:

Οι κλάσεις με τον δημόσιο τροποποιητή πρέπει να τοποθετούνται στα αρχεία με το ίδιο όνομα και την επέκταση java και να τοποθετούνται σε ένθετους φακέλους που αντιστοιχούν στο όνομα του πακέτου. Η παραπάνω κλάση myapplication.mylibrary.MyClass θα έχει την ακόλουθη διαδρομή:
"myapplication / mylibrary / MyClass.java".

Τελεστές

13	`||`	Logical conditional-OR	
14	`c ? t : f`	Ternary conditional (see ?:)	Right-to-left

15	`=`
	`+=` `-=`
	`*=` `/=` `%=`
	`<<=` `>>=` `>>>=`
	`&=` `^=` `|=`
		Simple assignment
		Assignment by sum and difference
		Assignment by product, quotient, and remainder
		Assignment by bitwise left shift, signed right shift and unsigned right shift
		Assignment by bitwise AND, XOR, and OR	

Illustration 6: Java Τελεστές 2

	`new`	Class instance or array creation	
4	`*` `/` `%`	Multiplication, division, and modulus (remainder)	Left-to-right
5	`+` `-` `+`	Addition and subtraction String concatenation	
6	`<<` `>>` `>>>`	Bitwise left shift, signed right shift and unsigned right shift	

7	`<` `<=`
	`>` `>=`
	`instanceof`
		Relational "less than" and "less than or equal to"
		Relational "greater than" and "greater than or equal to"
		Type comparison	

8	`==` `!=`	Relational "equal to" and "not equal to"	
9	`&`	Bitwise and logical AND	
10	`^`	Bitwise and logical XOR (exclusive or)	
11	`|`	Bitwise and logical OR (inclusive or)	
12	`&&`	Logical conditional-AND	

Illustration 5: Java Τελεστές

Οι χειριστές της Java είναι παρόμοιοι με αυτούς της C ++. Ωστόσο, δεν υπάρχει διαχειριστής διαγραφής λόγω μηχανισμών συλλογής απορριμμάτων στην Java και δεν υπάρχουν λειτουργίες σε δείκτες, αφού η Java δεν τις υποστηρίζει. Μια άλλη διαφορά είναι ότι η Java έχει έναν χειριστή ο οποίος δεν έχει υπογράψει τη σωστή μετατόπιση (>>>), ενώ η υπογραφή του χειριστή της δεξιάς μετατόπισης της C εξαρτάται από τον τύπο. Οι χειριστές της Java δεν μπορούν να υπερφορτωθούν.

Δομές ελέγχου

```
int a = 1;
int b = 2;
int minVal = (a < b) ? a : b;
```

```
} else {
    doSomethingDifferent();
}
```

Οι if δηλώσεις στην Java είναι παρόμοιες με εκείνες της C και χρησιμοποιούν την ίδια σύνταξη:

Η if δήλωση μπορεί να περιλαμβάνει προαιρετικό else block, οπότε γίνεται μια εντολή if-then-else:

Όπως στη C, η else-if κατασκευή δεν περιλαμβάνει καμία ειδική λέξη-κλειδί, σχηματίζεται ως ακολουθία χωριστών statements if-then-else:

Επίσης, σημειώστε ότι ο χειριστής ?: μπορεί να χρησιμοποιηθεί στη θέση απλής if-else δήλωσης, για παράδειγμα:

Switch statement

```
switch (ch) {
    case 'A':
        doSomething(); // Triggered if ch == 'A'
        break;
    case 'B':
    case 'C':
        doSomethingElse(); // Triggered if ch == 'B' or ch == 'C'
        break;
    default:
        doSomethingDifferent(); // Triggered in any other case
        break;
}
```

Οι εντολές επιλογής σε Java μπορούν να χρησιμοποιούν τύπους δεδομένων byte, short, char και int (όχι long) ή τους αντίστοιχους τύπους περιτύλιξης. Ξεκινώντας με το J2SE 5.0, είναι δυνατή η χρήση τύπων enum. Ξεκινώντας με το Java SE 7, είναι δυνατή η χρήση των χορδών. Άλλοι τύποι αναφοράς δεν μπορούν να χρησιμοποιηθούν στις δηλώσεις μεταγωγής.

Πιθανές τιμές εμφανίζονται χρησιμοποιώντας ετικέτες περιπτώσεων. Αυτές οι ετικέτες στην Java μπορεί να περιέχουν μόνο σταθερές (συμπεριλαμβανομένων των σταθερών enum και των σταθερών συμβολοσειράς). Η εκτέλεση θα ξεκινήσει μετά την ετικέτα που αντιστοιχεί στην έκφραση μέσα στις αγκύλες. Μπορεί να υπάρχει μια προαιρετική προεπιλεγμένη ετικέτα για να δηλώσει ότι ο κώδικας που ακολουθεί θα εκτελεστεί αν καμία από τις ετικέτες των περιπτώσεων δεν αντιστοιχεί στην έκφραση.

Ο κωδικός για κάθε ετικέτα τελειώνει με τη λέξη-κλειδί διακοπής. Είναι πιθανό να παραλείψετε να προκαλέσει την εκτέλεση να προχωρήσει στην επόμενη ετικέτα, ωστόσο, συνήθως θα αναγράφεται μια προειδοποίηση κατά τη διάρκεια της σύνταξης.

```
for (int i = 0; i < 10; i++) {
    doSomething();
}

// A more complex loop using two variables
for (int i = 0, j = 9; i < 10; i++, j -= 3) {
    doSomething();
}
```

Δομές Επανάληψης

Οι δηλώσεις επανάληψης είναι δηλώσεις που εκτελούνται επανειλημμένα όταν μια δεδομένη συνθήκη αξιολογείται ως αληθής. Από το J2SE 5.0, η Java έχει τέσσερις μορφές τέτοιων δηλώσεων.

while

Στο βρόχο while, η δοκιμή γίνεται πριν από κάθε επανάληψη.

do..while

Στο do while βρόχο, η δοκιμή γίνεται μετά από κάθε επανάληψη. Κατά συνέπεια, ο κώδικας εκτελείται πάντα τουλάχιστον μία φορά.

for

For βρόχος στη Java περιλαμβάνει ένα initializer, μια συνθήκη και μια αντίθετη έκφραση. Είναι δυνατό να συμπεριληφθούν αρκετές εκφράσεις του ίδιου είδους με χρήση κόμμα ως οριοθέτη. Ωστόσο, σε αντίθεση με τη C, το κόμμα είναι απλώς ένας οριοθέτης και όχι ένας τελεστής.

```
start:
someMethod();
```

Όπως στη C, και οι τρεις εκφράσεις είναι προαιρετικές. Ο ακόλουθος βρόχος είναι άπειρος:

Enhanced for loop

Enhanced for loop ήταν διαθέσιμες από το J2SE 5.0. Αυτός ο τύπος βρόχου χρησιμοποιεί ενσωματωμένους iterators πάνω σε συστοιχίες και συλλογές για να επιστρέψει κάθε στοιχείο στη δεδομένη συλλογή. Κάθε στοιχείο επιστρέφεται και είναι προσβάσιμο στο πλαίσιο του μπλοκ κώδικα. Όταν εκτελείται το μπλοκ, το επόμενο στοιχείο επιστρέφεται έως ότου δεν υπάρχουν υπόλοιπα στοιχεία. Σε αντίθεση με τη C #, αυτό το είδος βρόχου δεν περιλαμβάνει ειδική λέξη-κλειδί, αλλά χρησιμοποιεί διαφορετικό στυλ συμβολισμού.

Δομές Jump

Labels

Στις ετικέτες δίνονται σημεία στον κώδικα που χρησιμοποιούνται από τις δηλώσεις διακοπής και συνέχισης. Σημειώστε ότι η λέξη κλειδί Java goto δεν μπορεί να χρησιμοποιηθεί για να μεταβείτε σε συγκεκριμένα σημεία του κώδικα.

Primitive Types					
Type Name	Wrapper class	Value	Range	Size	Default Value
`byte`	`java.lang.Byte`	integer	−128 through +127	8-bit (1-byte)	`0`
`short`	`java.lang.Short`	integer	−32,768 through +32,767	16-bit (2-byte)	`0`
`int`	`java.lang.Integer`	integer	−2,147,483,648 through +2,147,483,647	32-bit (4-byte)	`0`
`long`	`java.lang.Long`	integer	−9,223,372,036,854,775,808 through +9,223,372,036,854,775,807	64-bit (8-byte)	`0`
`float`	`java.lang.Float`	floating point number	±1.401298E−45 through ±3.402823E+38	32-bit (4-byte)	`0.0f` [3]
`double`	`java.lang.Double`	floating point number	±4.94065645841246E−324 through ±1.79769313486232E+308	64-bit (8-byte)	`0.0`
`boolean`	`java.lang.Boolean`	Boolean	`true` or `false`	1-bit (1-bit)	`false`
`char`	`java.lang.Character`	UTF-16 code unit (BMP character or a part of a surrogate pair)	`'\u0000'` through `'\uFFFF'`	16-bit (2-byte)	`'\u0000'`

Illustration 7: Java πρωταρχικοί τύποι

break

Η εντολή break βγαίνει από τη δήλωση πλησιέστερου βρόχου. Η εκτέλεση συνεχίζεται στη εντολή μετά τη λήξη της δήλωσης, εάν υπάρχει.

Πρωτογενείς τύποι

Οι πρωταρχικοί τύποι της Java περιλαμβάνουν τύπους ακέραιων αριθμών, αριθμούς κινητής υποδιαστολής, μονάδες κώδικα UTF-16 και τύπο boolean. Δεν υπάρχουν τύποι που δεν έχουν υπογραφεί στη Java, εκτός από τον τύπο char, ο οποίος χρησιμοποιείται για την αναπαραγωγή μονάδων κώδικα UTF-16. Η έλλειψη μη υπογεγραμμένων τύπων αντισταθμίζεται με την εισαγωγή μη σωστής δεξιάς μετατόπισης (>>>), η οποία δεν υπάρχει στη C ++.

Classes

Οι κλάσεις αποτελούν βασικά στοιχεία μιας αντικειμενοστρεφής γλώσσας όπως η Java. Περιέχουν μέλη που αποθηκεύουν και χειρίζονται δεδομένα. Οι κλάσεις χωρίζονται σε

κορυφαίο και ένθετο. Οι κλάσεις που είναι ενσωματωμένες είναι κλάσεις τοποθετημένες σε μια άλλη τάξη που μπορεί να έχουν πρόσβαση στα ιδιωτικά μέλη της κατηγορίας που περιβάλλει. Οι κατηγοριοποιημένες τάξεις περιλαμβάνουν τις κλάσεις μελών (οι οποίες μπορούν να οριστούν με τον στατικό τροποποιητή για απλή εμφάνιση ή χωρίς αυτό για τις εσωτερικές τάξεις), τις τοπικές τάξεις και τις ανώνυμες τάξεις.

Top-level class	`class Foo {`
	`  // Class members`
	`}`

Inner class	`class Foo { // Top-level class`
	`  class Bar { // Inner class`
	`  }`
	`}`

Nested class	`class Foo { // Top-level class`
	`  static class Bar { // Nested class`
	`  }`
	`}`

Local class	`class Foo {`
	`  void bar() {`
	`  class Foobar {// Local class within a method`
	`  }`
	`  }`
	`}`

Illustration 8: Java classes 1

Anonymous class	`class Foo {`
	`  void bar() {`
	`  new Object() {// Creation of a new anonymous class extending Object`
	`  };`
	`  }`
	`}`

Illustration 9: Java classes 2

2.4 Java Web Development

Η ανάπτυξη ιστού είναι ένας ευρύς όρος για την εργασία που σχετίζεται με την ανάπτυξη μιας ιστοσελίδας για το Διαδίκτυο (World Wide Web) ή ενός ενδοδικτύου (ιδιωτικό δίκτυο). Η ανάπτυξη ιστού μπορεί να κυμαίνεται από την ανάπτυξη της απλούστερης στατικής ενιαίας σελίδας απλού κειμένου στις πιο σύνθετες εφαρμογές διαδικτύου (ή απλώς «web apps») και στις υπηρεσίες κοινωνικών δικτύων. Μια πιο ολοκληρωμένη λίστα καθηκόντων στις οποίες αναφέρεται συνήθως η ανάπτυξη ιστού, μπορεί να περιλαμβάνει:

- την τεχνική ιστού,
- το σχεδιασμό ιστοσελίδων,
- την ανάπτυξη περιεχομένου ιστού,
- τη σύνδεση με πελάτες,
- το scripting από πλευράς πελάτη / διακομιστή,
- τη διαμόρφωση εξυπηρετητή ιστού και δικτύου και
- την ανάπτυξη ηλεκτρονικού εμπορίου.

Μεταξύ των επαγγελματιών του διαδικτύου, η "ανάπτυξη ιστού" αναφέρεται συνήθως στις κύριες πτυχές του σχεδιασμού ιστοσελίδων:

- γραφή και
- κωδικοποίηση.

Πιο πρόσφατα, η ανάπτυξη ιστού έχει συμβεί στη δημιουργία συστημάτων διαχείρισης περιεχομένου ή CMS. Αυτά τα CMS μπορούν να γίνουν από το μηδέν, ιδιόκτητο ή ανοικτού κώδικα. Σε γενικές γραμμές, το CMS λειτουργεί ως ενδιάμεσο λογισμικό μεταξύ της βάσης δεδομένων και του χρήστη μέσω του προγράμματος περιήγησης. Ένα βασικό όφελος ενός CMS είναι ότι επιτρέπει στους μη τεχνικούς ανθρώπους να κάνουν αλλαγές στον ιστότοπό τους χωρίς να έχουν τεχνικές γνώσεις.

Για μεγαλύτερους οργανισμούς και επιχειρήσεις, οι ομάδες ανάπτυξης ιστού μπορούν να αποτελούνται από εκατοντάδες άτομα (προγραμματιστές ιστού) και να ακολουθούν τις συνήθεις μεθόδους όπως οι μεθοδολογίες της Agile κατά την ανάπτυξη ιστοτόπων. Μικρότερες οργανώσεις μπορεί να απαιτούν μόνο έναν μόνιμο ή συμβαλλόμενο προγραμματιστή ή δευτερεύουσα ανάθεση σε σχετικές θέσεις εργασίας, όπως ένας γραφίστας ή ένας τεχνικός συστημάτων πληροφοριών. Η ανάπτυξη ιστού μπορεί να είναι μια προσπάθεια συνεργασίας μεταξύ των τμημάτων και όχι του τομέα ενός καθορισμένου τμήματος. Υπάρχουν τρία είδη εξειδίκευσης του web developer: προγραμματιστής front-end, back-end προγραμματιστής και full-stack developer. Οι προγραμματιστές του front-end ασχολούνται με τη διάταξη και τα

οπτικά στοιχεία ενός ιστότοπου, ενώ οι back-end προγραμματιστές ασχολούνται με τη λειτουργικότητα μιας ιστοσελίδας. Οι back-end προγραμματιστές θα προγραμματίσουν τις λειτουργίες μιας ιστοσελίδας που θα συλλέγει δεδομένα.

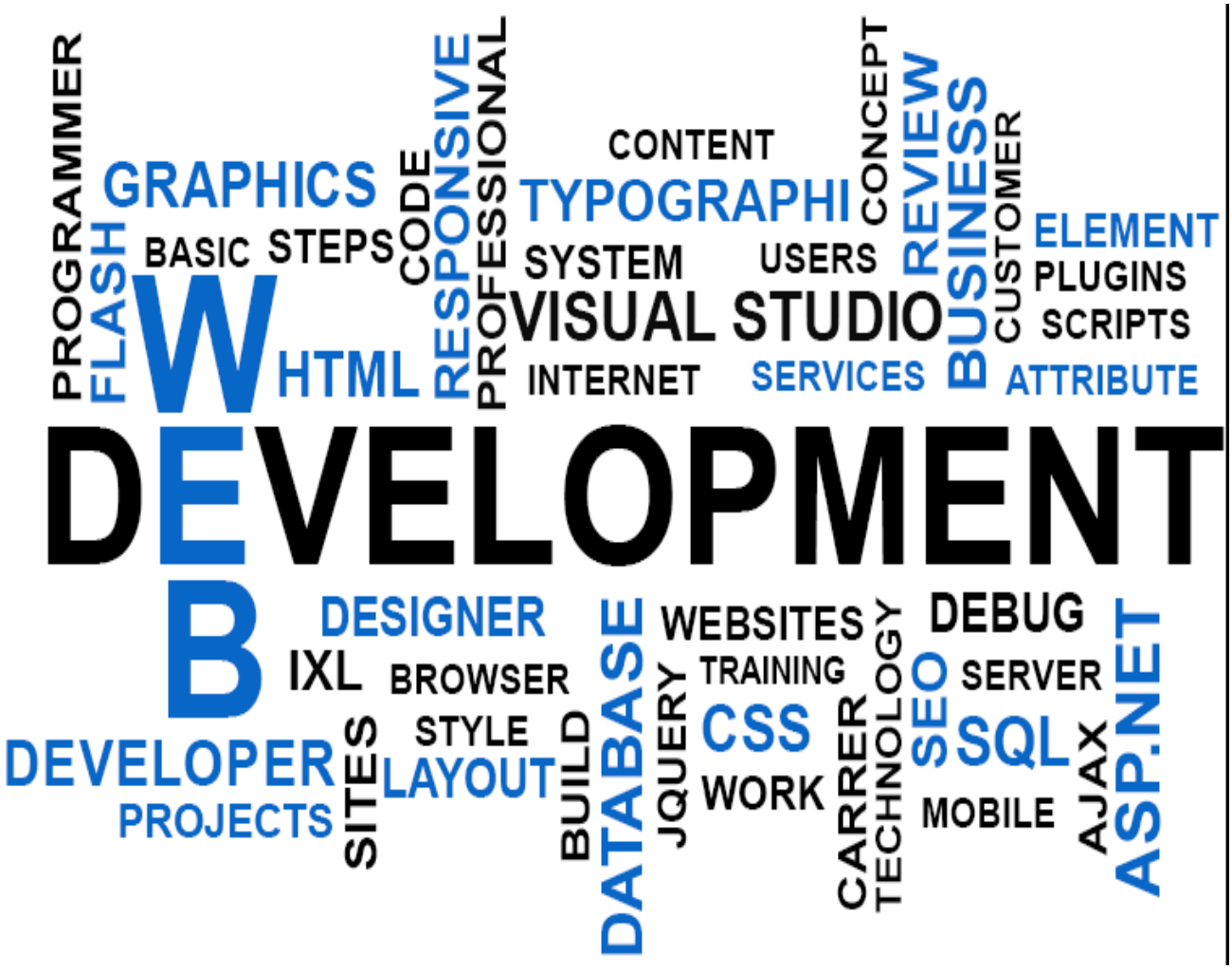


Illustration 10: Web development

Από την εμπορευματοποίηση του διαδικτύου, η ανάπτυξη ιστού υπήρξε μια αναπτυσσόμενη βιομηχανία. Η ανάπτυξη αυτού του κλάδου καθοδηγείται από επιχειρήσεις που επιθυμούν να χρησιμοποιήσουν την ιστοσελίδα τους για να πουλήσουν προϊόντα και υπηρεσίες στους πελάτες.

Υπάρχει λογισμικό ανοιχτού κώδικα για την ανάπτυξη ιστού όπως οι:

- BerkeleyDB,
- GlassFish,
- LAMP (Linux, Apache, MySQL, PHP) και
- Perl / Plack.

Αυτό έχει κρατήσει το κόστος της εκμάθησης ανάπτυξης ιστοσελίδων στο ελάχιστο. Ένας άλλος παράγοντας που συνέβαλε στην ανάπτυξη του κλάδου ήταν η άνοδος του εύχρηστου λογισμικού ανάπτυξης WYSIWYG, όπως:

- το Adobe Dreamweaver,
- το BlueGriffon και
- το Microsoft Visual Studio.

Η γνώση της γλώσσας HTML (HyperText Markup Language) ή των γλωσσών προγραμματισμού εξακολουθεί να απαιτείται για τη χρήση αυτού του λογισμικού, αλλά τα βασικά μπορούν να αποκτηθούν και να εφαρμοστούν γρήγορα με τη βοήθεια αρχείων βοήθειας, τεχνικών βιβλίων, διαδικτυακών σεμιναρίων ή εκπαίδευσης πρόσωπο με πρόσωπο.

Ένα συνεχώς αυξανόμενο σύνολο εργαλείων και τεχνολογιών έχει βοηθήσει τους προγραμματιστές να δημιουργήσουν πιο δυναμικές και διαδραστικές ιστοσελίδες. Επιπλέον, οι προγραμματιστές ιστού βοηθούν τώρα να παρέχουν εφαρμογές ως υπηρεσίες ιστού οι οποίες κατά παράδοση ήταν διαθέσιμες μόνο ως εφαρμογές σε υπολογιστή γραφείου. Αυτό επέτρεψε πολλές ευκαιρίες για αποκέντρωση της ενημέρωσης και της διανομής των μέσων ενημέρωσης. Παραδείγματα μπορούν να παρατηρηθούν με την άνοδο των υπηρεσιών cloud, όπως το Adobe Creative Cloud, το Dropbox και τα Έγγραφα Google. Αυτές οι υπηρεσίες web επιτρέπουν στους χρήστες να αλληλεπιδρούν με εφαρμογές από πολλές τοποθεσίες, αντί να συνδέονται με ένα συγκεκριμένο σταθμό εργασίας για το περιβάλλον εφαρμογής τους.

Παραδείγματα δραματικών μετασχηματισμών στην επικοινωνία και το εμπόριο που οδηγούνται από την ανάπτυξη ιστού περιλαμβάνουν το ηλεκτρονικό εμπόριο. Οι ιστοσελίδες ηλεκτρονικής δημοπρασίας όπως το eBay έχουν αλλάξει τον τρόπο με τον οποίο οι καταναλωτές βρίσκουν και αγοράζουν αγαθά και υπηρεσίες. Οι διαδικτυακοί λιανοπωλητές όπως το Amazon.com και το Buy.com (μεταξύ πολλών άλλων) έχουν μεταμορφώσει την εμπειρία αγορών και ευκαιριών κυνηγιού για πολλούς καταναλωτές. Ένα άλλο καλό παράδειγμα μετασχηματιστικής επικοινωνίας που οδηγείται από την ανάπτυξη ιστού είναι το blog. Οι εφαρμογές Web, όπως το WordPress και το Movable Type, έχουν δημιουργήσει εύκολα εφαρμοσμένα περιβάλλοντα blog για μεμονωμένους ιστότοπους. Η δημοτικότητα των συστημάτων διαχείρισης περιεχομένου ανοιχτού κώδικα, όπως τα συστήματα:

- Joomla
- Drupal
- XOOPS
- TYPO3

καθώς και τα συστήματα διαχείρισης εταιρικού περιεχομένου όπως το Alfresco και η πλατφόρμα eXo, έχουν επεκτείνει τον αντίκτυπο της ανάπτυξης του ιστού στην ηλεκτρονική αλληλεπίδραση και επικοινωνία.

Η ανάπτυξη του ιστού έχει επίσης επηρεάσει την προσωπική δικτύωση και το μάρκετινγκ. Οι ιστότοποι δεν είναι πλέον απλά εργαλεία για εργασία ή για εμπόριο, αλλά εξυπηρετούν ευρύτερα για επικοινωνία και κοινωνική δικτύωση. Ιστοσελίδες όπως το Facebook και το Twitter παρέχουν στους χρήστες μια πλατφόρμα για να επικοινωνούν και οργανώσεις με έναν πιο προσωπικό και διαδραστικό τρόπο για να προσελκύσουν το κοινό.

2.5 Java Web Development

Οι εφαρμογές Web είναι εξ ορισμού κατανεμημένες εφαρμογές, δηλαδή προγράμματα που τρέχουν σε περισσότερους από έναν υπολογιστές και επικοινωνούν μέσω δικτύου ή διακομιστή. Συγκεκριμένα, οι εφαρμογές web προσεγγίζονται με ένα πρόγραμμα περιήγησης ιστού και είναι δημοφιλείς λόγω της ευκολίας χρήσης του προγράμματος περιήγησης από τον πελάτη-χρήστη. Για την επιχείρηση, η δυνατότητα ενημέρωσης και συντήρησης εφαρμογών ιστού χωρίς ανάπτυξη και εγκατάσταση λογισμικού σε δυνητικά χιλιάδες υπολογιστές-πελάτες αποτελεί βασικό λόγο για τη δημοτικότητά τους. Οι εφαρμογές Web χρησιμοποιούνται για μηνύματα ηλεκτρονικού ταχυδρομείου, ηλεκτρονικές πωλήσεις λιανικής, πίνακες συζήτησης, ιστολόγια, ηλεκτρονικές τραπεζικές υπηρεσίες και πολλά άλλα. Μια εφαρμογή ιστού μπορεί να προσεγγιστεί και να χρησιμοποιηθεί από εκατομμύρια ανθρώπους.

Όπως οι εφαρμογές γραφείου, οι εφαρμογές ιστού αποτελούνται από πολλά μέρη και περιέχουν συχνά μίνι προγράμματα, μερικά από τα οποία έχουν διεπαφές χρήστη και μερικές από τις οποίες δεν απαιτούν καθόλου γραφικό περιβάλλον χρήστη (GUI). Επιπλέον, οι εφαρμογές ιστού απαιτούν συχνά επιπλέον γλώσσα σήμανσης ή δέσμης ενεργειών, όπως γλώσσα προγραμματισμού HTML, CSS ή JavaScript. Επίσης, πολλές εφαρμογές χρησιμοποιούν μόνο τη γλώσσα προγραμματισμού Java, η οποία είναι ιδανική λόγω της ευελιξίας της.

Μια εφαρμογή Ιστού μπορεί να είναι τόσο απλή όσο μια σελίδα που δείχνει την τρέχουσα ημερομηνία και ώρα ή είναι τόσο περίπλοκη όσο ένα σύνολο σελίδων στο οποίο μπορείτε να αναζητήσετε και να κάνετε κράτηση για τις πιο βολικές πτήσεις, ξενοδοχεία και ενοικιάσεις αυτοκινήτων για τις επόμενες διακοπές σας.

Οι τεχνολογίες Java που χρησιμοποιείτε για τη δημιουργία εφαρμογών ιστού αποτελούν τμήμα της πλατφόρμας Java EE, εκτός από πολλές κλάσεις και πακέτα Java Platform, Standard Edition (Java SE). Προκειμένου πολλές από αυτές τις τεχνολογίες να λειτουργούν σε ένα διακομιστή, ο διακομιστής πρέπει να έχει εγκατεστημένο ένα δοχείο ή διακομιστή ιστού που να αναγνωρίζει και να εκτελεί τις κατηγορίες που δημιουργείτε.

Τεχνολογίες Java για χρήση σε εφαρμογές Web

Υπάρχουν υπερβολικά πολλές τεχνολογίες Java για να αναφερθούν σε ένα άρθρο, οπότε αυτό το άρθρο θα περιγράφει μόνο τις πιο συχνά χρησιμοποιούμενες. Ο αριθμός των τεχνολογιών που παρατίθενται εδώ μπορεί να φαίνεται συντριπτικός. Λάβετε υπόψη ότι δεν θα χρειαστεί να τα χρησιμοποιήσετε όλα. Στην πραγματικότητα, μια εφαρμογή web συχνά αποτελείται από τίποτα περισσότερο από μία σελίδα που δημιουργήθηκε με την τεχνολογία JavaServer Pages (JSP). Μερικές φορές θα συνδυάσετε τρεις ή περισσότερες τέτοιες τεχνολογίες. Ανεξάρτητα από το πόσα καταλήγετε, είναι καλό να ξέρετε τι είναι διαθέσιμο σε εσάς και πώς μπορείτε να χρησιμοποιήσετε το καθένα σε μια εφαρμογή στο διαδίκτυο.

Illustration 11: Java web

Java Servlet

Ένα Java servlet είναι ένα πρόγραμμα Java που επεκτείνει τις δυνατότητες ενός διακομιστή. Παρόλο που οι εξυπηρετητές μπορούν να ανταποκριθούν σε οποιουδήποτε τύπου αιτημάτων, συνήθως εφαρμόζουν εφαρμογές που φιλοξενούνται σε διακομιστές Web. Τέτοιοι

εξυπηρετητές ιστού είναι το αντίστοιχο της Java με άλλες δυναμικές τεχνολογίες περιεχομένου ιστού όπως το PHP και το ASP.NET.

Ένα Java servlet επεξεργάζεται ή αποθηκεύει μια κλάση Java σε Java EE που συμμορφώνεται με το API Java Servlet, ένα πρότυπο για την εφαρμογή των τάξεων Java που ανταποκρίνονται στα αιτήματα. Τα Servlets θα μπορούσαν καταρχήν να επικοινωνούν μέσω οποιουδήποτε πρωτοκόλλου πελάτη-διακομιστή, αλλά χρησιμοποιούνται συχνότερα με το πρωτόκολλο HTTP. Έτσι, το "servlet" χρησιμοποιείται συχνά ως συντομογραφία για το "servlet HTTP". Έτσι, ένας προγραμματιστής λογισμικού μπορεί να χρησιμοποιήσει ένα servlet για να προσθέσει δυναμικό περιεχόμενο σε έναν web server χρησιμοποιώντας την πλατφόρμα Java. Το περιεχόμενο που δημιουργείται συνήθως είναι HTML, αλλά μπορεί να είναι άλλα δεδομένα όπως η XML. Τα Servlets μπορούν να διατηρούν μεταβλητές κατάστασης σε συνόδους σε πολλές συναλλαγές διακομιστή χρησιμοποιώντας cookies HTTP ή επανασύνδεση διεύθυνσης URL.

Για να αναπτύξετε και να εκτελέσετε ένα servlet, πρέπει να χρησιμοποιηθεί ένα δοχείο ιστού. Ένα δοχείο ιστού (επίσης γνωστό ως δοχείο servlet) είναι ουσιαστικά το στοιχείο ενός διακομιστή ιστού που αλληλεπιδρά με τους εξυπηρετητές. Ο περιέκτης διαδικτύου είναι υπεύθυνος για τη διαχείριση του κύκλου ζωής των εξυπηρετητών, χαρτογράφηση μιας διεύθυνσης URL σε ένα συγκεκριμένο σέρβερ και διασφάλιση ότι ο αιτών URL έχει τα σωστά δικαιώματα πρόσβασης.

Το API Servlet, που περιέχεται στην ιεραρχία πακέτων Java javax.servlet, ορίζει τις αναμενόμενες αλληλεπιδράσεις του δοχείου διαδικτύου και ενός εξυπηρετητή.

Ένα Servlet είναι ένα αντικείμενο που λαμβάνει ένα αίτημα και παράγει μια απάντηση βασισμένη σε αυτό το αίτημα. Το βασικό πακέτο Servlet ορίζει αντικείμενα Java που αντιπροσωπεύουν αιτήματα και απαντήσεις σε servlet, καθώς και αντικείμενα που αντικατοπτρίζουν τις παραμέτρους διαμόρφωσης και περιβάλλον εκτέλεσης του servlet. Το πακέτο javax.servlet.http ορίζει ειδικές υποκλάσεις HTTP των γενικών στοιχείων του εξυπηρετητή, συμπεριλαμβανομένων των αντικειμένων διαχείρισης σύνδεσης που παρακολουθούν πολλαπλά αιτήματα και απαντήσεις μεταξύ του διακομιστή ιστού και ενός πελάτη. Τα Servlets μπορεί να συσκευάζονται σε αρχείο WAR ως εφαρμογή web.

Τα Servlets μπορούν να δημιουργηθούν αυτόματα από τις σελίδες JavaServer (JSP) από τον μεταγλωττιστή των σελίδων JavaServer. Η διαφορά μεταξύ servlets και JSP είναι ότι οι servlets συνήθως ενσωματώνουν HTML μέσα στον Java κώδικα, ενώ οι JSPs ενσωματώνουν κώδικα Java σε HTML. Ενώ η άμεση χρήση των εξυπηρετητών για τη δημιουργία HTML (όπως φαίνεται στο παρακάτω παράδειγμα) έχει γίνει σπάνια, το πλαίσιο ιστού MVC στο Java EE (JSF) εξακολουθεί να χρησιμοποιεί ρητά την τεχνολογία servlet για το χειρισμό χαμηλού επιπέδου ζήτησης / απάντησης μέσω του FacesServlet . Μια κάπως παλαιότερη χρήση είναι η

χρήση servlets σε συνδυασμό με JSPs σε ένα μοτίβο που ονομάζεται "Model 2", το οποίο είναι μια γεύση του model-view-controller.

Τα πλεονεκτήματα της χρήσης servlets είναι η γρήγορη απόδοση και η ευκολία χρήσης σε συνδυασμό με μεγαλύτερη ισχύ σε σχέση με την παραδοσιακή CGI (Common Gateway Interface). Τα παραδοσιακά σενάρια CGI γραμμένα σε Java έχουν πολλά μειονεκτήματα απόδοσης:

- Όταν γίνεται ένα αίτημα HTTP, δημιουργείται μια νέα διαδικασία κάθε φορά που καλείται το σενάριο CGI. Η επιβάρυνση που σχετίζεται με τη δημιουργία διαδικασίας μπορεί να κυριαρχήσει στο φόρτο εργασίας, ειδικά όταν το σενάριο κάνει σχετικά γρήγορες λειτουργίες. Έτσι, η δημιουργία της διαδικασίας θα πάρει περισσότερο χρόνο για την εκτέλεση δέσμης ενεργειών CGI. Αντίθετα, για τους εξυπηρετητές, κάθε ζήτημα αντιμετωπίζεται από ένα ξεχωριστό νήμα Java μέσα στη διαδικασία διακομιστή ιστού, αποφεύγοντας έτσι τα γενικά έξοδα που σχετίζονται με τις διεργασίες forking στο πλαίσιο του δαίμονα HTTP.
- Οι ταυτόχρονες αιτήσεις CGI θα φορτώσουν το σενάριο CGI που θα αντιγραφεί στη μνήμη μία φορά ανά αίτηση. Με τους εξυπηρετητές, υπάρχει μόνο ένα αντίγραφο που διαρκεί μεταξύ των αιτημάτων και μοιράζεται μεταξύ των νημάτων.
- Μόνο μία περίπτωση απαντά ταυτόχρονα σε όλα τα αιτήματα. Αυτό μειώνει τη χρήση της μνήμης και διευκολύνει τη διαχείριση των επίμονων δεδομένων.
- Ένα servlet μπορεί να τρέξει από ένα δοχείο σέρβετ σε ένα περιοριστικό περιβάλλον, που ονομάζεται sandbox. Αυτό είναι παρόμοιο με ένα applet που τρέχει στο sandbox του web browser. Αυτό επιτρέπει την περιορισμένη χρήση δυνητικά επιβλαβών servlet. Τα προγράμματα CGI μπορούν φυσικά να είναι και τα sandbox, αφού είναι απλά διαδικασίες OS.

Τεχνολογίες όπως το FastCGI και τα παράγωγά του (συμπεριλαμβανομένου του SCGI, AJP) δεν παρουσιάζουν τα μειονεκτήματα απόδοσης του CGI, που προκύπτουν από τη συνεχή αναπαραγωγή της διαδικασίας. Ωστόσο, είναι σχεδόν τόσο απλά όσο το CGI. Συνεπώς, βρίσκονται επίσης σε αντίθεση με τους σπονδύλους που είναι ουσιαστικά πιο πολύπλοκοι.

Τρεις μέθοδοι είναι κεντρικές στον κύκλο ζωής ενός εξυπηρετητή. Αυτές είναι

- init ()
- service () και
- destroy ()

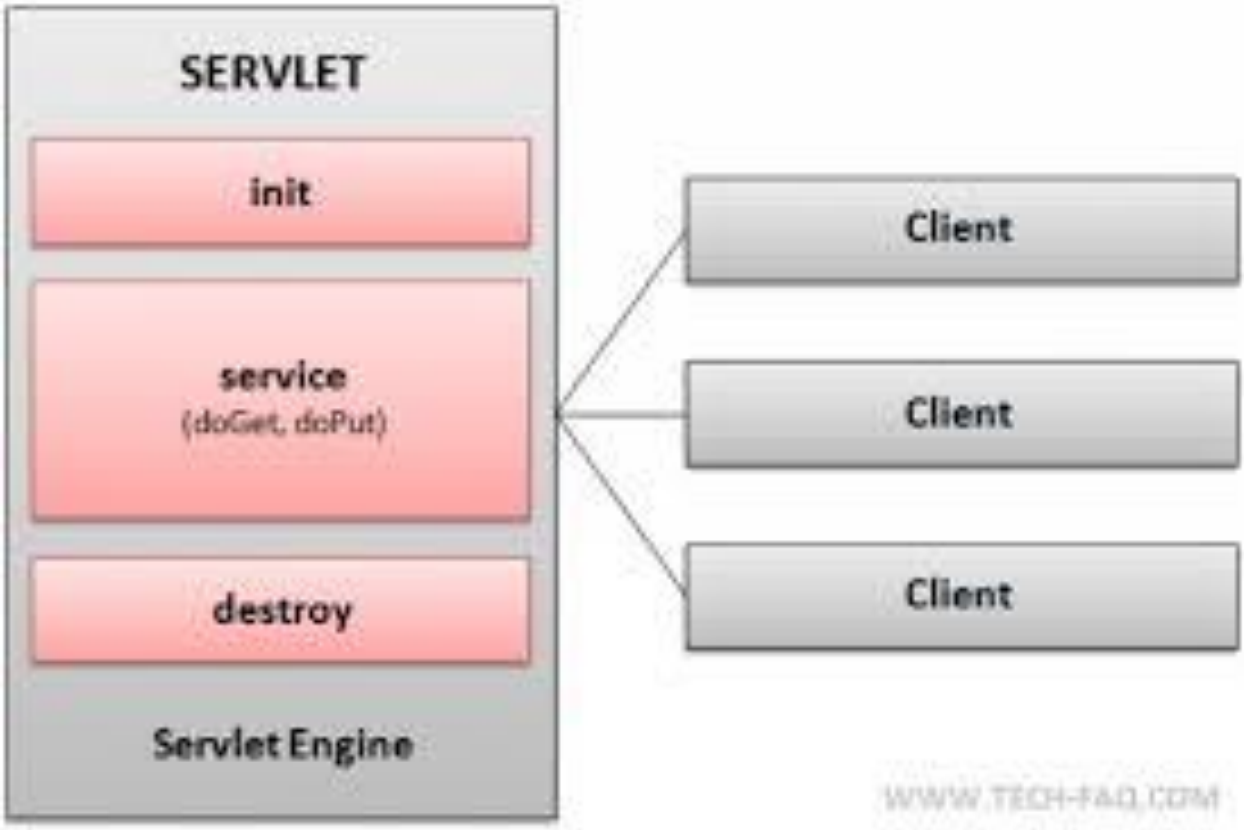


Illustration 12: Servlet

Εφαρμόζονται από κάθε servlet και χρησιμοποιούνται σε συγκεκριμένες χρονικές στιγμές από το διακομιστή. Κατά τη διάρκεια του σταδίου αρχικοποίησης του κύκλου ζωής του εξυπηρετητή, ο διαδικτυακός περιέκτης προετοιμάζει την κατάσταση servlet καλώντας τη μέθοδο init(), περνώντας ένα αντικείμενο που υλοποιεί τη διεπαφή javax.servlet.ServletConfig. Αυτό το αντικείμενο ρύθμισης παραμέτρων επιτρέπει στο διακομιστή να αποκτήσει πρόσβαση στις παραμέτρους αρχικοποίησης ονομαστικής τιμής από την εφαρμογή Ιστού.

Μετά την αρχικοποίηση, η υπηρεσία servlet μπορεί να εξυπηρετήσει αιτήματα πελατών. Κάθε αίτημα εξυπηρετείται στο δικό του ξεχωριστό νήμα. Ο διαδικτυακός περιέκτης καλεί τη μέθοδο service () του servlet για κάθε αίτημα. Η μέθοδος service () καθορίζει το είδος της αίτησης που γίνεται και την αποστέλλει σε μια κατάλληλη μέθοδο για να χειριστεί το αίτημα. Η διαφορά μεταξύ servlets και JSP είναι ότι οι servlets συνήθως ενσωματώνουν HTML μέσα στον Java κώδικα, ενώ οι JSPs ενσωματώνουν κώδικα Java σε HTML. Ο προγραμματιστής του servlet πρέπει να παρέχει μια εφαρμογή για αυτές τις μεθόδους. Εάν ζητηθεί μια μέθοδος που δεν υλοποιείται από το servlet, ονομάζεται η μέθοδος της γονικής κλάσης, με αποτέλεσμα συνήθως να επιστρέφεται σφάλμα στον αιτούντα.

Τέλος, το δοχείο διαδικτύου καλεί τη μέθοδο καταστροφής () που εξαλείφει το servlet. Η μέθοδος καταστροφής (), όπως και το init (), ονομάζεται μόνο μία φορά στον κύκλο ζωής ενός εξυπηρετητή.

Τα παρακάτω είναι ένα τυπικό σενάριο χρήστη αυτών των μεθόδων.

1. Ας υποθέσουμε ότι ένας χρήστης ζητά να επισκεφτεί μια διεύθυνση URL.

- Το πρόγραμμα περιήγησης στη συνέχεια παράγει ένα αίτημα HTTP για αυτήν τη διεύθυνση URL.
- Αυτό το αίτημα αποστέλλεται στον κατάλληλο διακομιστή.

2. Το αίτημα HTTP λαμβάνεται από τον εξυπηρετητή ιστού και προωθείται στο δοχείο servlet.

- Ο περιέκτης χαρτογραφεί αυτό το αίτημα σε ένα συγκεκριμένο σέρβερ.
- Ο servlet ανασύρεται δυναμικά και φορτώνεται στο χώρο διευθύνσεων του δοχείου.

3. Το δοχείο επικαλείται τη μέθοδο init () του servlet.

- Αυτή η μέθοδος ενεργοποιείται μόνο όταν αρχικά φορτώνεται στη μνήμη το servlet.
- Είναι δυνατή η παράδοση παραμέτρων αρχικοποίησης στο servlet, ώστε να μπορεί να διαμορφωθεί.

4. Ο περιέκτης ενεργοποιεί τη μέθοδο service () του εξυπηρετητή.

- Αυτή η μέθοδος καλείται να επεξεργαστεί το αίτημα HTTP.

- Το σέρβετ μπορεί να διαβάσει δεδομένα που έχουν παρασχεθεί στο αίτημα HTTP.
- Ο εξυπηρετητής μπορεί επίσης να διατυπώσει μια απόκριση HTTP για τον πελάτη.

5. Το σέρβερ παραμένει στον χώρο διευθύνσεων του κοντέινερ και είναι διαθέσιμο για να επεξεργαστεί τυχόν άλλες αιτήσεις HTTP που λαμβάνονται από πελάτες.

- Η μέθοδος service () καλείται για κάθε αίτημα HTTP.

6. Ο περιέκτης μπορεί, σε κάποιο σημείο, να αποφασίσει την εκφόρτωση του σέρβετ από τη μνήμη του.

- Οι αλγόριθμοι με τους οποίους γίνεται αυτή η απόφαση είναι συγκεκριμένοι για κάθε δοχείο.

7. Το κοντέινερ καλεί τη μέθοδο καταστροφής () του σέρβερ για να παραιτηθεί από τους πόρους, όπως οι λαβές αρχείων που διατίθενται για το servlet. Σημαντικά δεδομένα μπορεί να αποθηκευτούν σε ένα επίμονο κατάστημα.

8. Η μνήμη που διατίθεται για το servlet και τα αντικείμενα του μπορεί στη συνέχεια να συλλεχθεί τα σκουπίδια.

Παράδειγμα

Το ακόλουθο παράδειγμα servlet εκτυπώνει πόσες φορές ονομάστηκε η υπηρεσία service ().

Σημειώστε ότι το HttpServlet είναι μια υποκατηγορία του GenericServlet, μια εφαρμογή της διασύνδεσης Servlet.

Η μέθοδος service () της κλάσης HttpServlet αποστέλλει αιτήσεις στις μεθόδους doGet (), doPost (), doPut (), doDelete () και ούτω καθεξής. σύμφωνα με το αίτημα HTTP. Στο παρακάτω παράδειγμα η service () παρακάμπτεται και δεν διακρίνει ποια μέθοδος αιτήματος HTTP εξυπηρετεί.

```java
import java.io.IOException;

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ServletLifeCycleExample extends HttpServlet {

    private int count;

    @Override
    public void init(final ServletConfig config) throws ServletException {
        super.init(config);
        getServletContext().log("init() called");
        count = 0;
    }

    @Override
    protected void service(final HttpServletRequest request, final HttpServletResponse response) throws ServletException, IOException {
        getServletContext().log("service() called");
        count++;
        response.getWriter().write("Incrementing the count to " + count);
    }

    @Override
    public void destroy() {
        getServletContext().log("destroy() called");
    }
}
```

Illustration 13: Java servlet example

3 Εγκατάσταση της εφαρμογής

Για να γίνει η εκκίνηση του συστήματος είναι απαραίτητη η εκτέλεση κάποιων βημάτων. Στο συγκεκριμένο κεφάλαιο παρουσιάζονται τα κατάλληλα εργαλεία που επιλέχθηκαν για την περάτωση της πτυχιακής. Ακόμη αναφέρονται οι ενέργειες που απαιτούνται για την έναρξη της εφαρμογής.

3.1 NetBeans

Το NetBeans είναι μια πλατφόρμα ανάπτυξης λογισμικού γραμμένη σε Java. Η πλατφόρμα NetBeans επιτρέπει την ανάπτυξη εφαρμογών από ένα σύνολο αρθρωτών στοιχείων λογισμικού που ονομάζονται ενότητες. Οι εφαρμογές που βασίζονται στην πλατφόρμα NetBeans, συμπεριλαμβανομένου του ολοκληρωμένου περιβάλλοντος ανάπτυξης NetBeans (IDE), μπορούν να επεκταθούν από τρίτους προγραμματιστές.

Το NetBeans IDE προορίζεται κυρίως για ανάπτυξη σε Java, αλλά υποστηρίζει και άλλες γλώσσες, ιδίως

- PHP,
- C / C ++ και
- HTML5

Το NetBeans είναι cross-platform και λειτουργεί σε Microsoft Windows, Mac OS X, Linux, Solaris και σε άλλες πλατφόρμες που υποστηρίζουν συμβατό JVM.

Ο επεξεργαστής υποστηρίζει πολλές γλώσσες από Java, C / C ++, XML και HTML, σε PHP, Groovy, Javadoc, JavaScript και JSP. Επειδή ο επεξεργαστής είναι επεκτάσιμος, μπορείτε να συνδέσετε υποστήριξη για πολλές άλλες γλώσσες.

Illustration 14: NetBeans logo

Η ομάδα NetBeans υποστηρίζει ενεργά το προϊόν και αναζητά προτάσεις από την ευρύτερη κοινότητα. Για κάθε απελευθέρωση έχει προηγηθεί χρονικό διάστημα για κοινοτικές δοκιμές και ανατροφοδότηση. Πάνω από 18 εκατομμύρια λήψεις του NetBeans IDE μέχρι σήμερα, και πάνω από 800.000 συμμετέχοντες προγραμματιστές, το έργο NetBeans αναπτύσσεται και συνεχίζει να αυξάνεται.

Μια νέα έκδοση κυκλοφόρησε στις 8.2 / Οκτώβριος 3,2016. Το IDE του NetBeans είναι το επίσημο IDE για την Java 8. Με τους εκδότες, τους αναλυτές κώδικα και τους μετατροπείς, μπορείτε να αναβαθμίσετε γρήγορα και ομαλά τις εφαρμογές σας για να χρησιμοποιήσετε νέες κατασκευές γλώσσας Java 8.

3.1.1 Εγκατάσταση

1. Αρχικά πρέπει να κατέβει το πακέτο που περιέχει όλες τις πληροφορίες για το πρόγραμμα NetBeans:

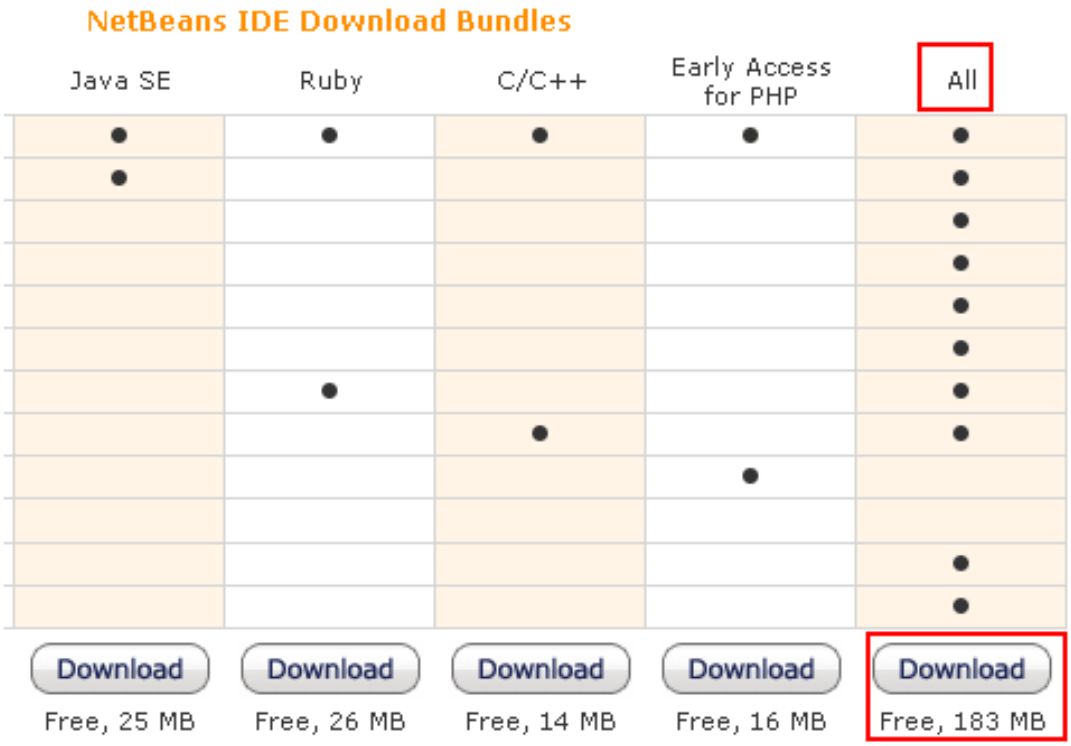


Illustration 15: Install Netbeans 1

2. Έπειτα ανοίγουμε το αρχείο:

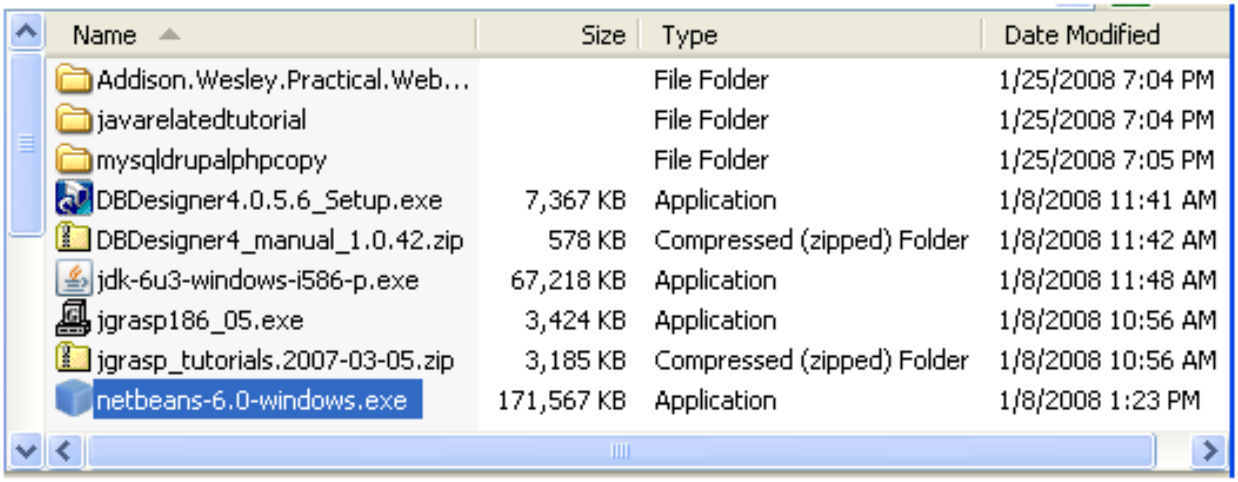


Illustration 16: Install NetBeans 2

3. Εμφανίζεται η πρόοδος της εγκατάστασης:

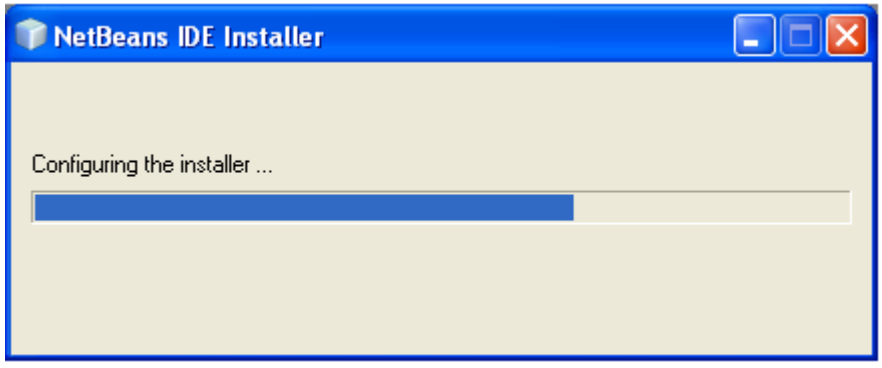
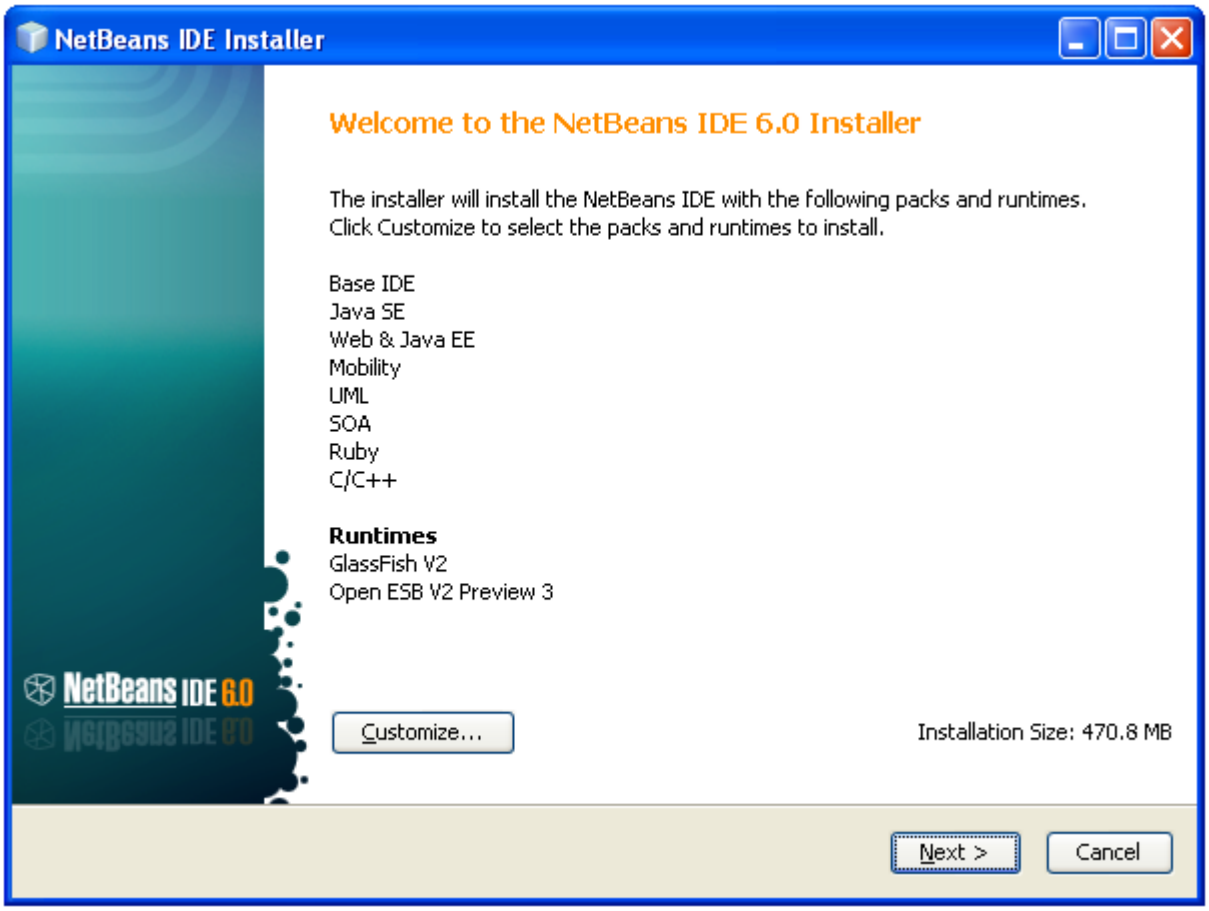
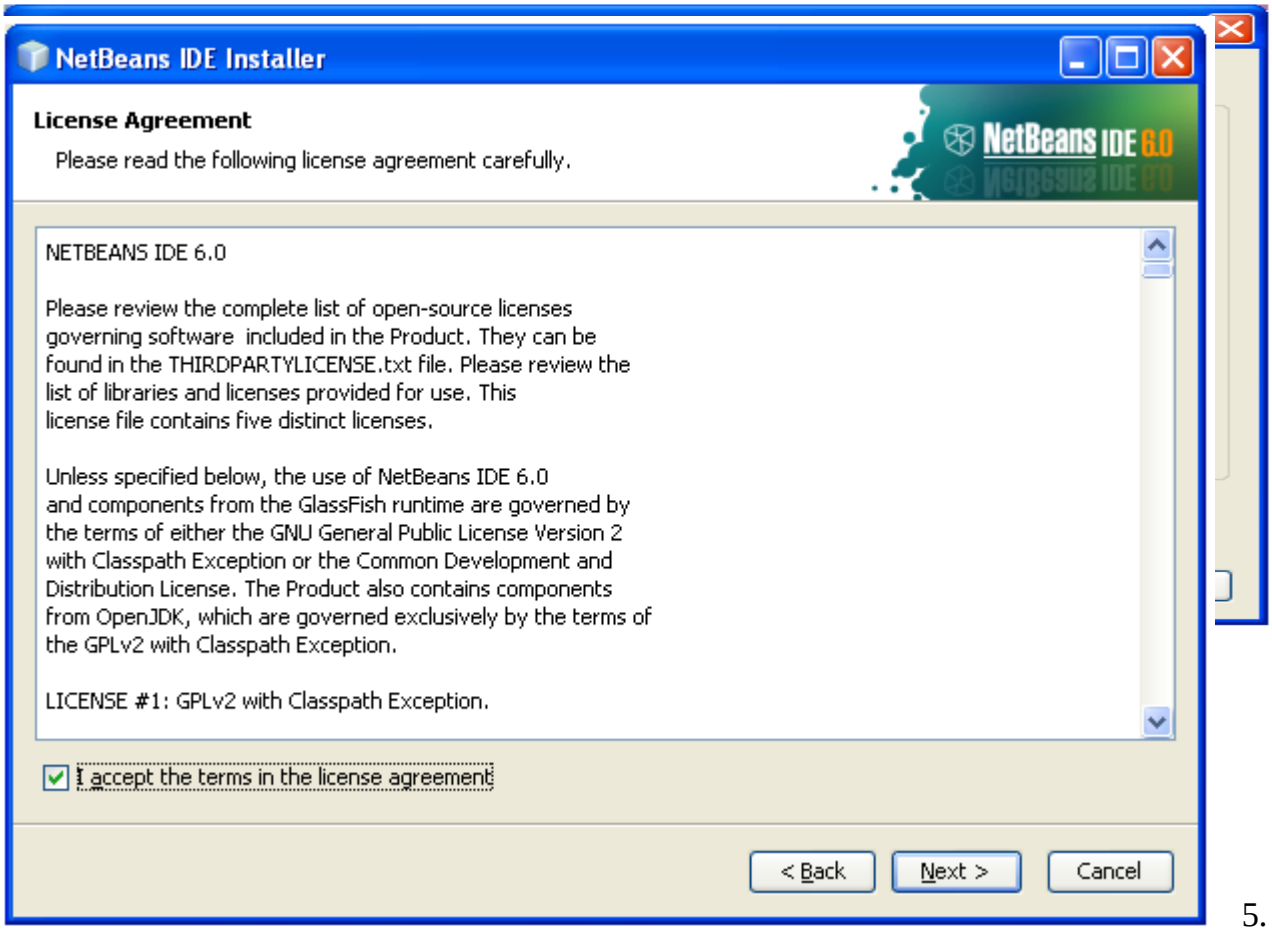


Illustration 17: Install NetBeans 3



4. Έπειτα εμφανίζεται το παράθυρο εγκατάστασης όπου επιλέγουμε το κουμπί Customize:

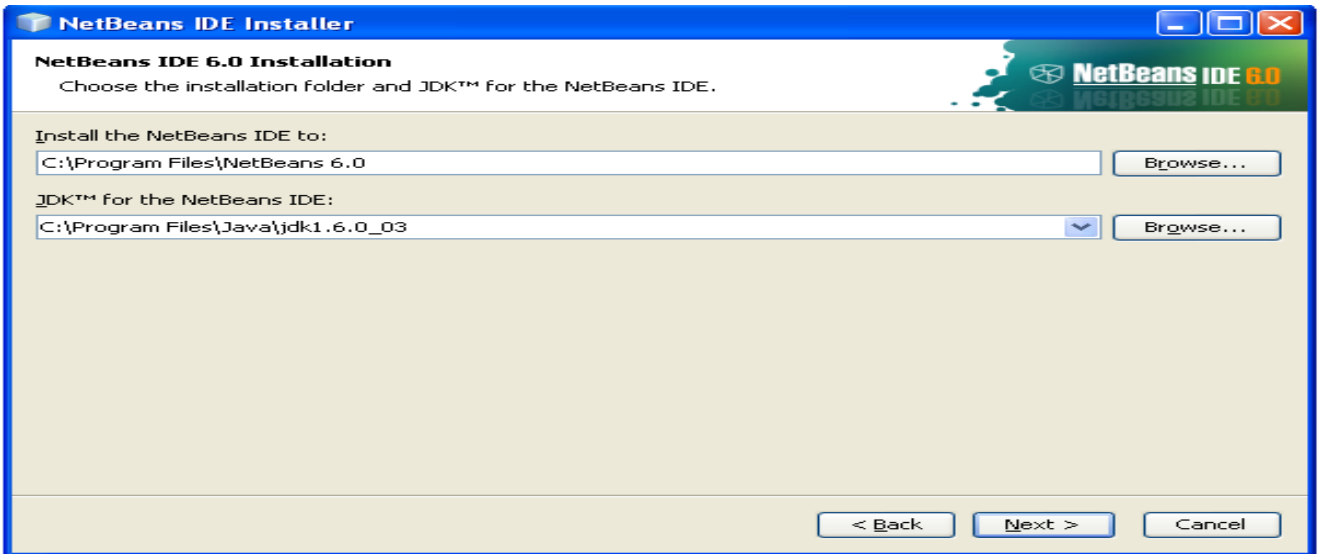


5.

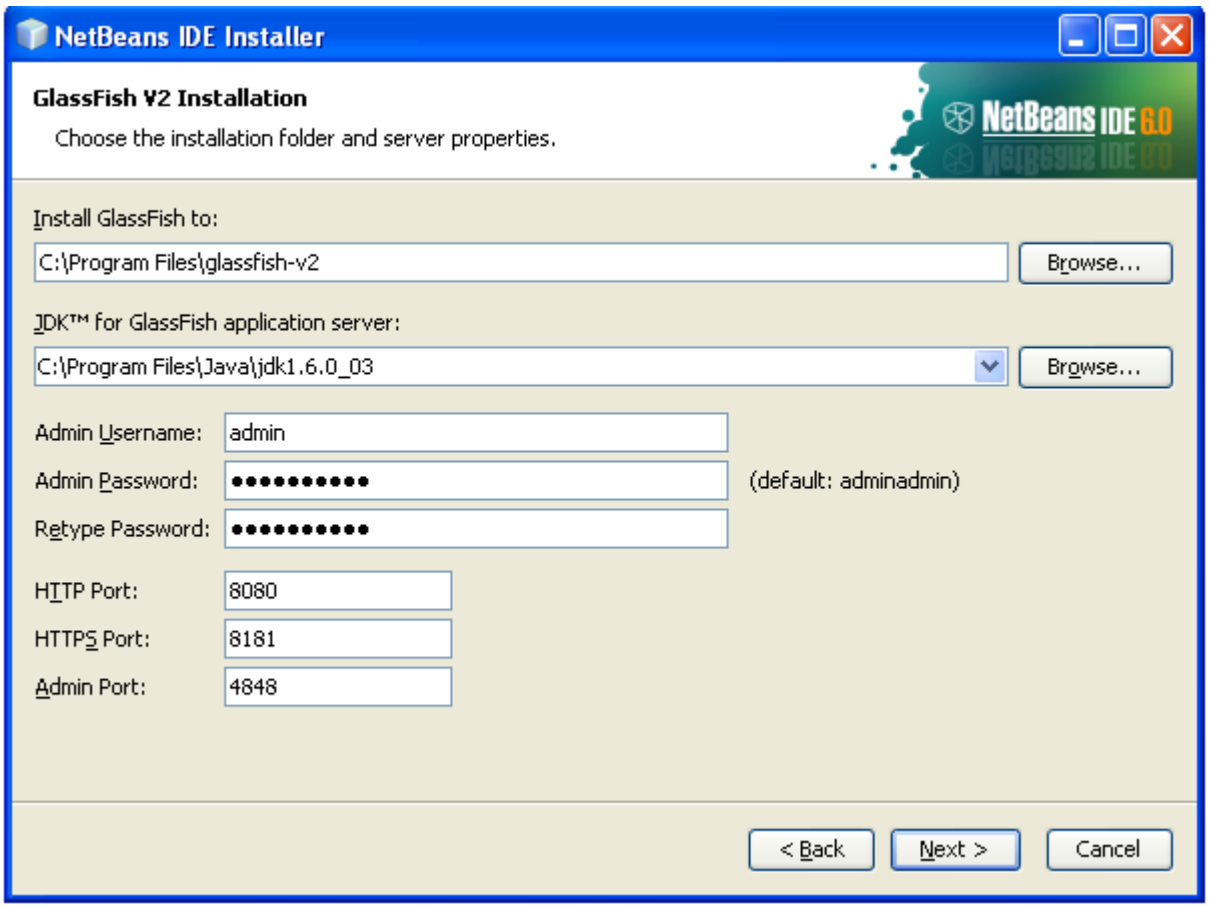
Illustration 18: Install Netbeans 4

Στο επόμενο παράθυρο επιλέγουμε όλα τα κουτιά:

6. Κάνουμε αποδοχή των όρων χρήσης:

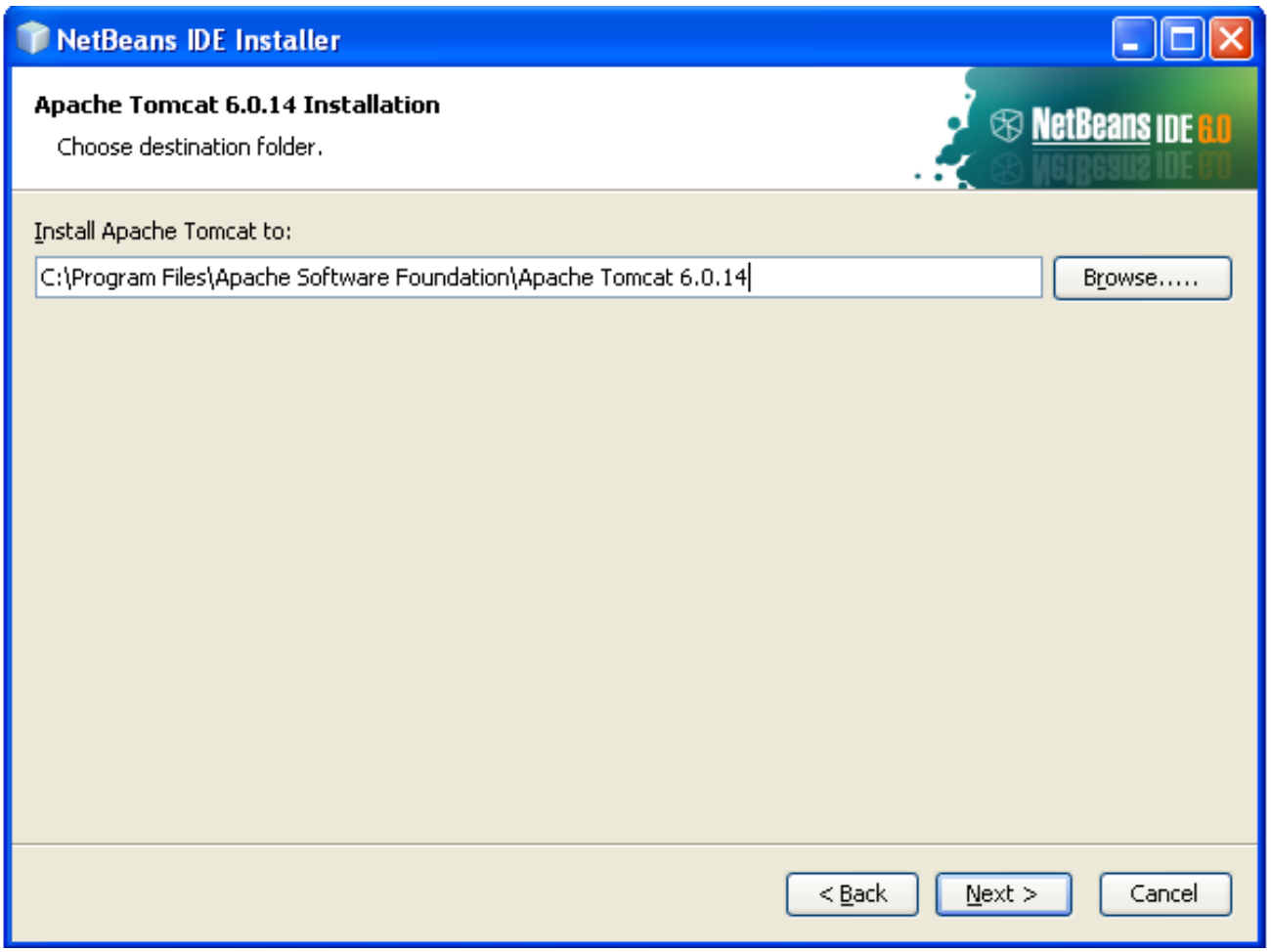


7. Επιλέγουμε το φάκελο όπου θα γίνει η εγκατάσταση:

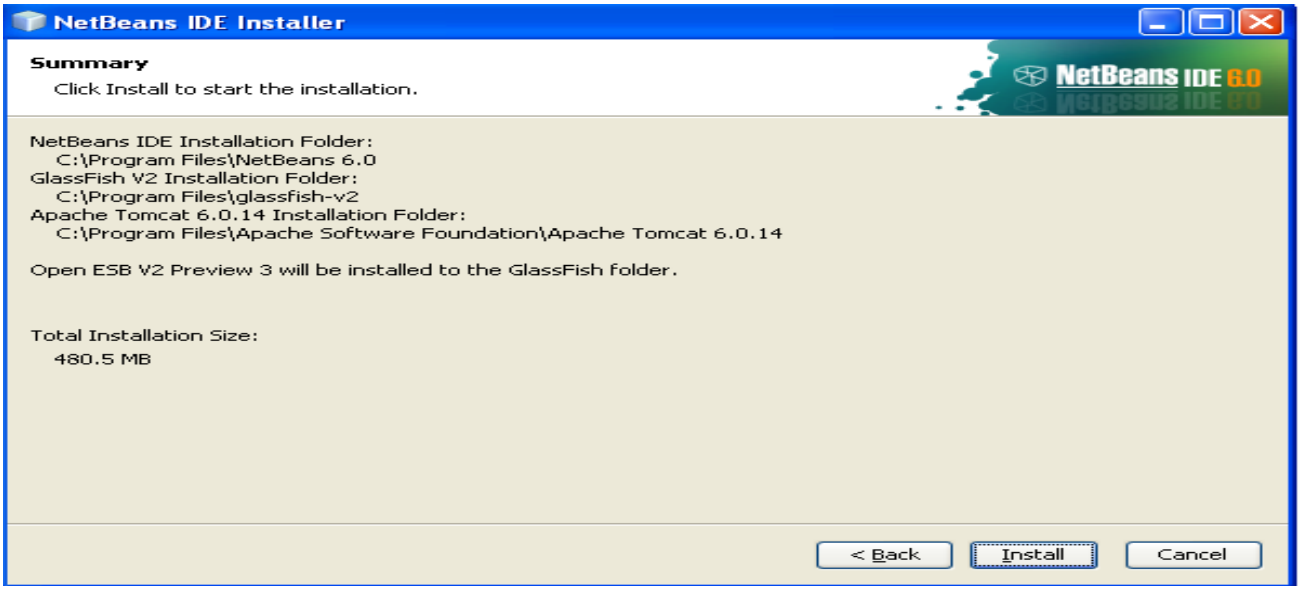


8. Δίνουμε άδεια διαχειριστή:

9. Επόμενο βήμα είναι να δημιουργήσουμε τον GlassFish web server:



10. Έπειτα θέτουμε το μονοπάτι για τον Apache:



11. Τέλος επιλέγουμε το Install και ξεκινά η εγκατάσταση:

3.2 Δημιουργία της βάσης δεδομένων

Αφού έχει εγκατασταθεί το NetBeans ανοίγουμε το project επιλέγοντας File και μετά Open Project. Εντοπίζουμε το αρχείο στον δίσκο και το NetBeans κάνει εκκίνηση.

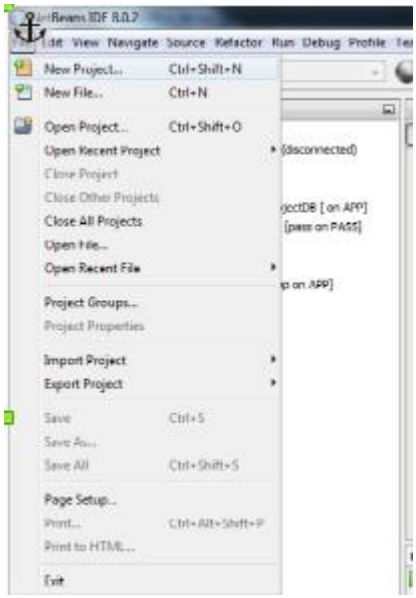


Illustration 19: Εκκίνηση εφαρμογής 1

Έπειτα πρέπει να δημιουργηθεί η βάση όπου θα αποθηκεύονται όλα τα δεδομένα του συστήματος. Επιλέγουμε την καρτέλα Services:

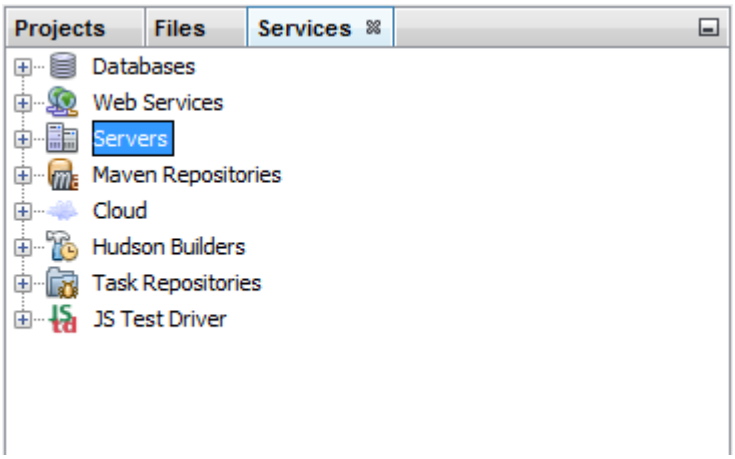


Illustration 20: Εκκίνηση εφαρμογής 2

Επιλέγουμε Databases και μετά JavaDB. Εκεί κάνουμε δεξί κλικ και Create Database.

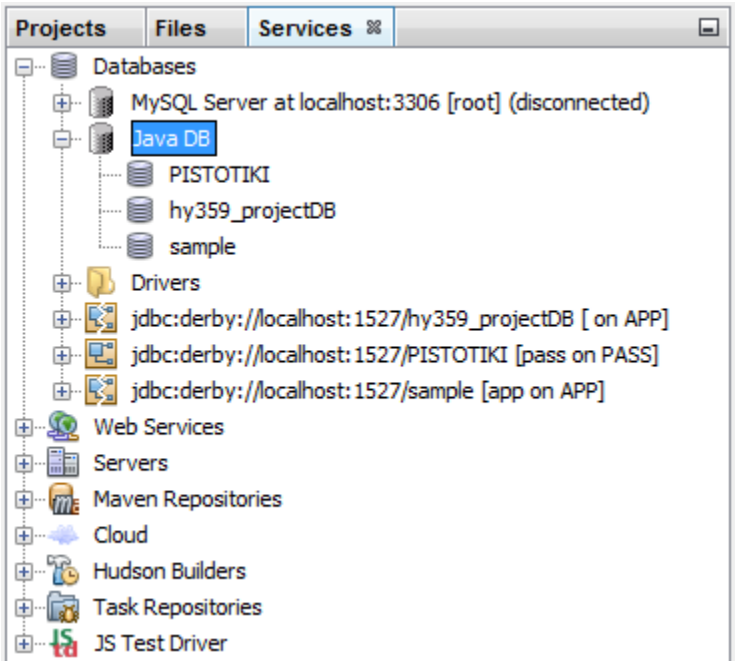


Illustration 21: Εκκίνηση εφαρμογής 3

Δημιουργούμε μια καινούργια βάση με το όνομα PISTOTIKI και δίνουμε κωδικούς πρόσβασης.

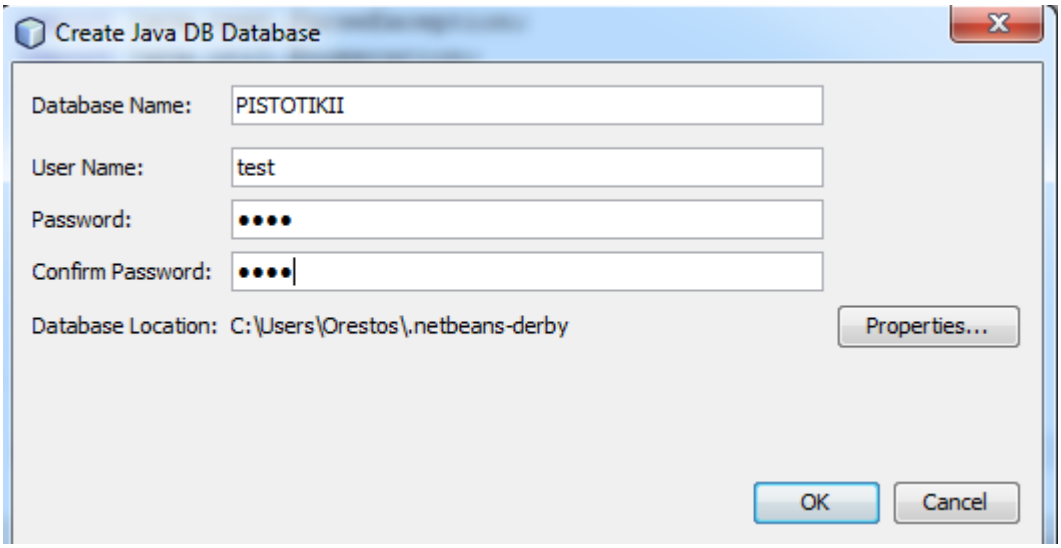


Illustration 22: Εκκίνηση εφαρμογής 4

Αφού έχει δημιουργηθεί η βάση πρέπει να επιλέξουμε το Connect για να συνδεθεί με τον server. Αναγκαία επίσης είναι η εκκίνηση του server.

3.3 Αρχικοποίηση της Βάσης Δεδομένων

Για τη σωστή λειτουργία του συστήματος πρέπει να εκτελεστούν κάποιες ενέργειες την πρώτη φορά που θα εκκινηθεί η εφαρμογή. Επιλέγουμε το Tables από τη βάση και κάνουμε Execute Command. Στο αρχείο που εμφανίζεται πρέπει να εισάγουμε τα ακόλουθα queries για να δημιουργηθούν οι πίνακες της βάσης:

1. Για να δημιουργηθεί ο πίνακας Customers:

```
CREATE TABLE Customers (Cust_name VARCHAR(100), Address VARCHAR(200),
Age INTEGER,   Telephone_number INTEGER, Id_number VARCHAR(50))
```

2. Για να δημιουργηθεί ο πίνακας Life_insurance:

```
CREATE TABLE life_insurance(Insurance_id INTEGER NOT NULL PRIMARY KEY
GENERATED ALWAYS AS IDENTITY(START WITH 1, INCREMENT BY 1),
Contract_date VARCHAR(20), Cust_name VARCHAR(100), Cost_monthly
INTEGER, Instalments_not_paid INTEGER)
```

3. Για να δημιουργηθεί ο πίνακας Loans:

```
CREATE TABLE loans(Loan_id INTEGER NOT NULL PRIMARY KEY GENERATED
ALWAYS AS IDENTITY(START WITH 1, INCREMENT BY 1), Loan_date
VARCHAR(20), Cust_name VARCHAR(100), Loan_money INTEGER,
Cust_monthly_pay INTEGER, Interest_type VARCHAR(30), Interest_rate
INTEGER , Repayment_date VARCHAR(30), Instalments_paid INTEGER,
Instalments_not_paid INTEGER, Total_paid INTEGER)
```

4. Για να δημιουργηθεί ο πίνακας Bonus:

```
CREATE TABLE bonus (Category VARCHAR(30), Id INTEGER, Reduction
VARCHAR(10), Increment VARCHAR(10))
```

5. Για να δημιουργηθεί ο πίνακας Customers_not_paying:

```
CREATE TABLE Customers_not_paying (Cust_name VARCHAR(100))
```

3.4 Πρόσθετα αρχεία

Το σύστημα χρειάζεται μια βιβλιοθήκη για τον Apache Derby Client. Για να εισάγουμε την βιβλιοθήκη πρέπει αρχικά να την κατεβάσουμε από τον σύνδεσμο:

http://mvnrepository.com/artifact/org.apache.derby/derbyclient/10.13.1.1

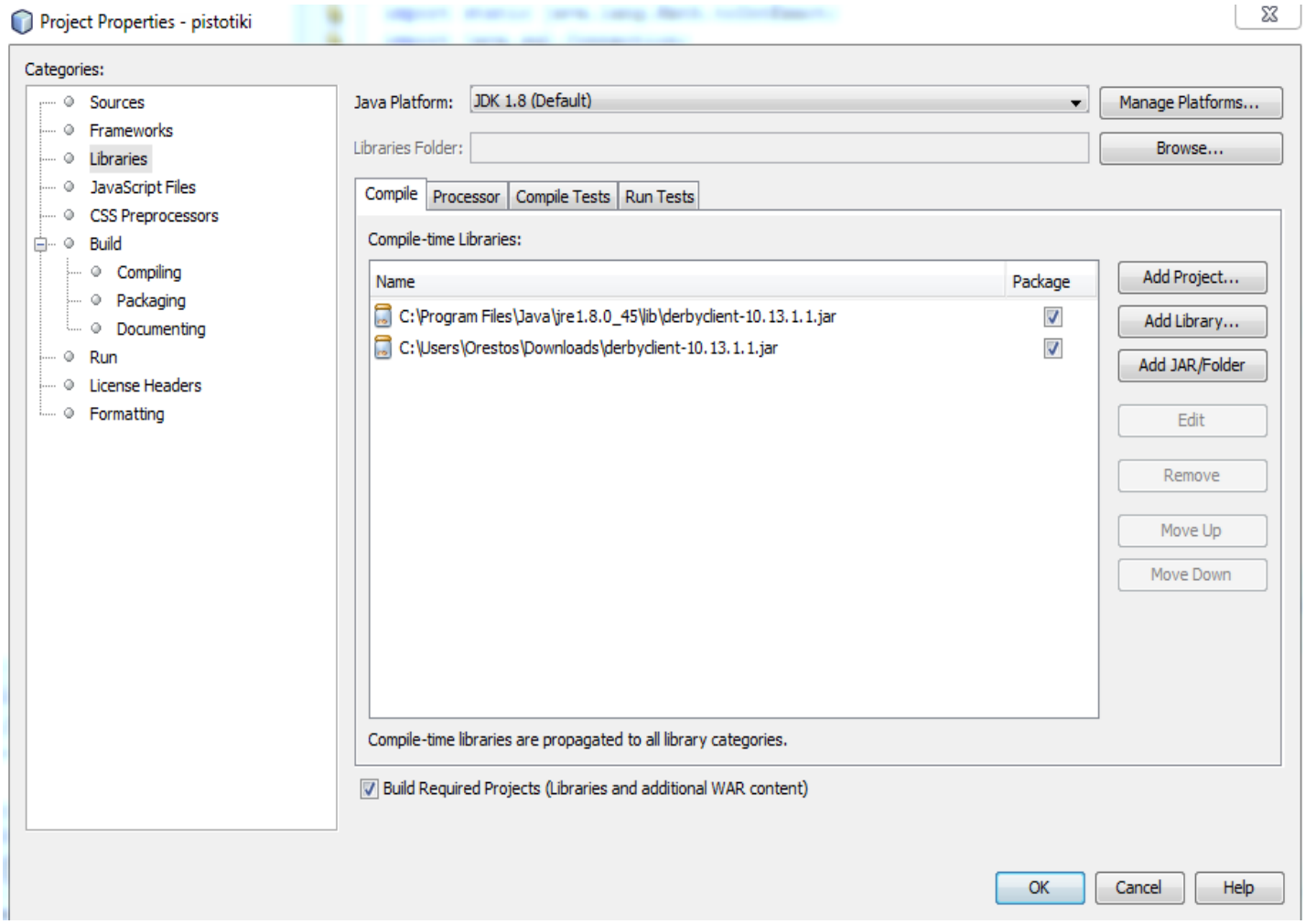


Illustration 24: Βιβλιοθήκη 2

Αφού κατέβει το jar αρχείο κάνουμε δεξί κλικ στο όνομα του project στο NetBeans και έπειτα επιλέγουμε τις ιδιότητες του project. Εκεί πάμε στις βιβλιοθήκες και προσθέτουμε το jar αρχείο που πήραμε από τον παραπάνω σύνδεσμο.

Στο σημείο αυτό έχουμε ολοκληρώσει όλες τις απαραίτητες ενέργειες και το σύστημα μπορεί να εκτελεστεί με επιτυχία με το Run του NetBeans. Για καλύτερη εμπειρία είναι προτεινόμενη η NetBeans Connector επέκταση στο browser.

4 Ανάλυση της δομής και του κώδικα του συστήματος

Στο παρών κεφάλαιο γίνεται η παρουσίαση του πυρήνα του συστήματος που υλοποιήθηκε για τις ανάγκες της πτυχιακή εργασίας. Παρακάτω παραθέτονται:

- Ο τρόπος που λειτουργεί το σύστημα
- Η δομή των καταλόγων
- Οι κώδικες που υλοποιούν την εφαρμογή
- Οι σελίδες όπου πλοηγείται ο χρήστης

4.1 Η δομή του συστήματος

Για τη σωστή οργάνωση του έργου είναι απαραίτητη η αρχειοθέτηση και η ομαδοποίηση των στοιχείων του. Η δομή των καταλόγων έχει ως εξής:

- Αρχεία jsp
  - Εισαγωγή
  - Επεξεργασία
  - Διαγραφή
  - Αναφορές
- Αρχεία css
- Αρχεία java
  - Κλάσεις

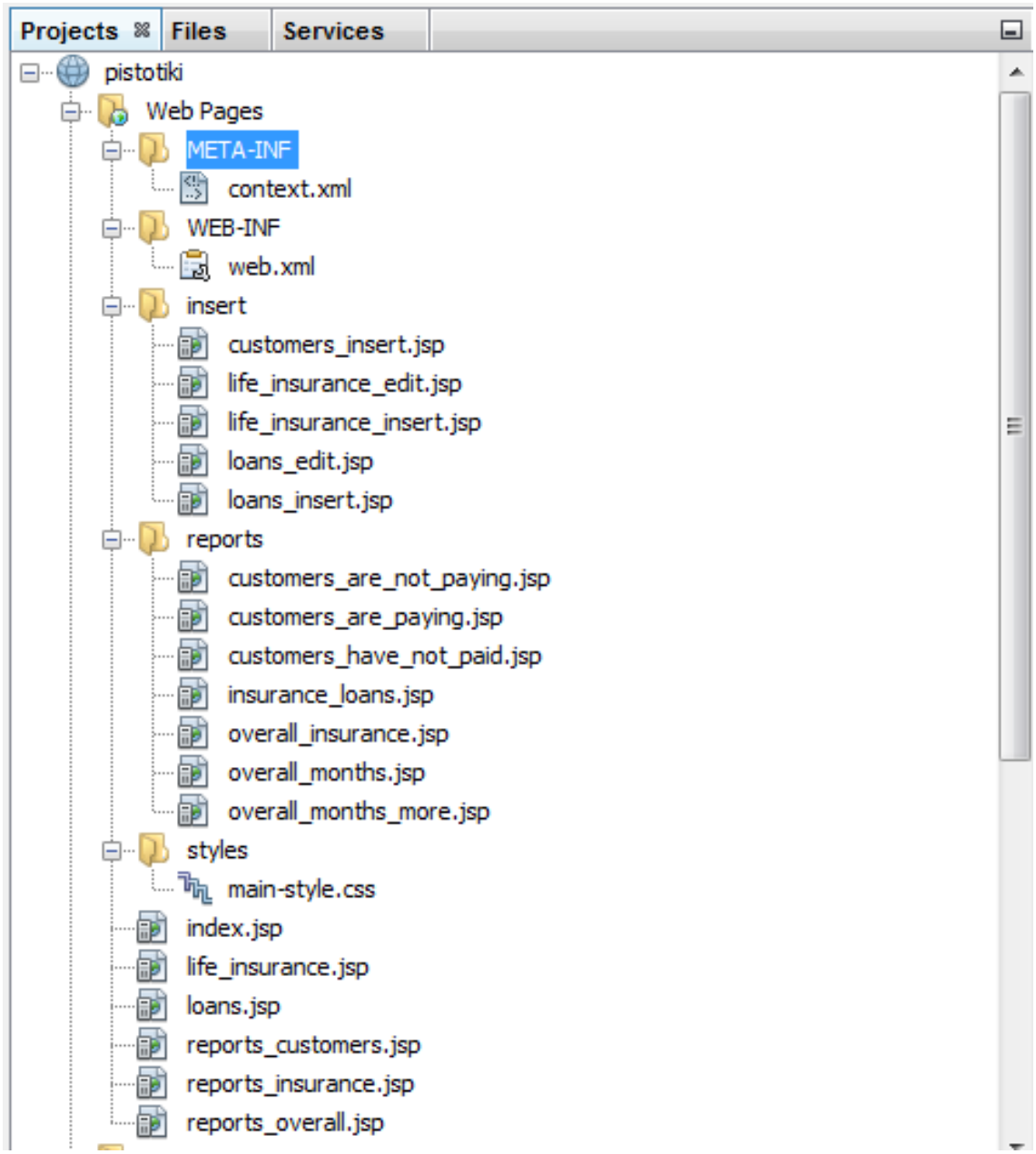


Illustration 25: Δομή καταλόγων 1

↘ Servlets

Τα αρχεία που υλοποιούν την ίδια εργασία τοποθετούνται σε κοινό φάκελο. Η δομή όπως φαίνεται στους καταλόγους του NetBeans εμφανίζεται παρακάτω.

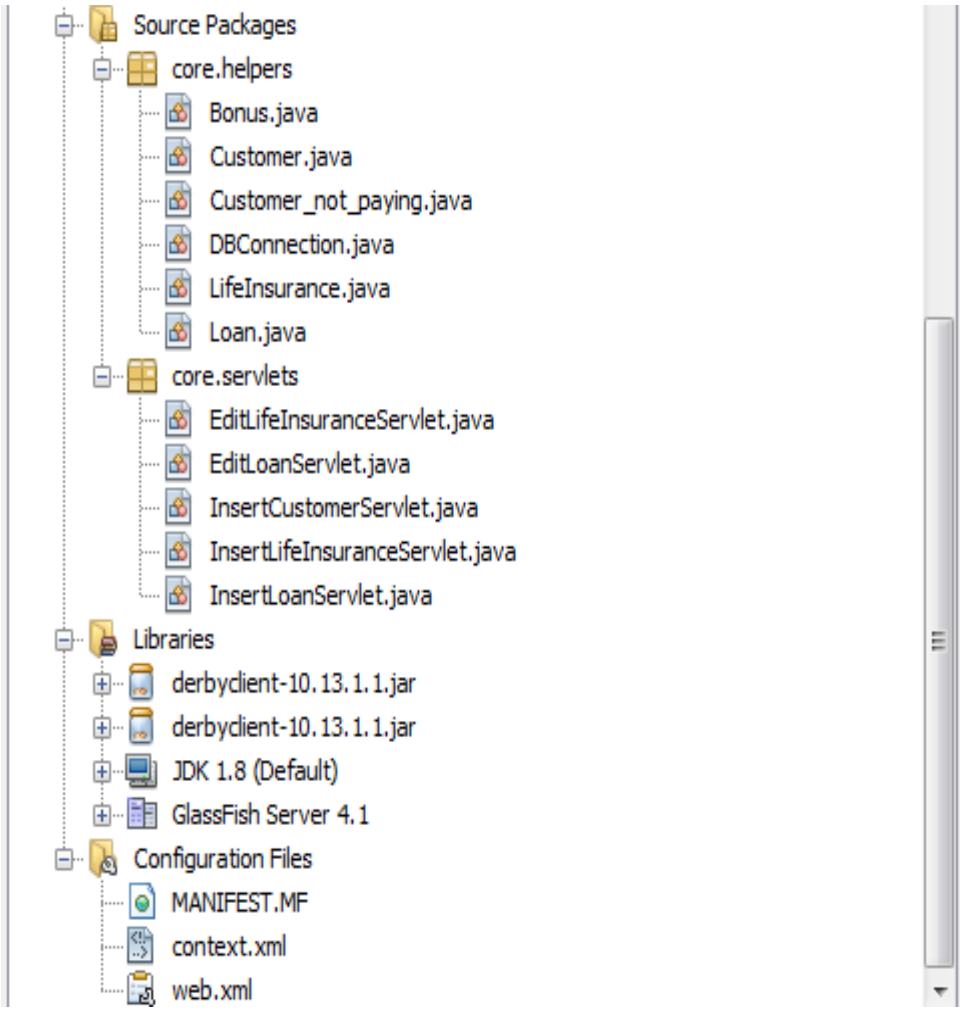


Illustration 26: Δομή καταλόγων 2

4.2 Διεπαφή

Στο συγκεκριμένο κεφάλαιο παρουσιάζονται οι σελίδες του συστήματος και οι κώδικες που τις υλοποιούν. Για όλες τις υποσελίδες έχει υλοποιηθεί ένα κοινό αρχείο css ώστε να υπάρχει κοινή μορφοποίηση των γραμματοσειρών και των άλλων στοιχείων της σελίδας.

main.css

```
body{
    font-family: 'Segoe UI', Helvetica, Arial, Sans-Serif;
    font-size: 14px;
    letter-spacing: normal;
    background-color: #e9eaed;
    margin: 0px;
}

ul {
    list-style-type: none;
```

```
    margin: 0;
    padding: 0;
    overflow: hidden;
    background-color: lightgray;
}

li {
    float: left;
}

li a, .dropbtn {
    display: inline-block;
    color: black;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
}

li a:hover, .dropdown:hover .dropbtn {
    background-color: #666666;
    color: white;
}

a.active {
    background-color: #666666;
    color: white !important;
}

li.dropdown {
    display: inline-block;
}

.dropdown-content {
    display: none;
    position: absolute;
    background-color: #f9f9f9;
    min-width: 160px;
```

```
    box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
    z-index: 1;
}

.dropdown-content a {
    color: black;
    padding: 12px 16px;
    text-decoration: none;
    display: block;
    text-align: left;
}

.dropdown-content a:hover {
    background-color: #f1f1f1;
    color: black;
}

.dropdown:hover .dropdown-content {
    display: block;
}

h1{
    font-weight: lighter;
    font-size: 25px;
    color: lightgray;
    padding:10px;
    background: #666666;
    margin: 0px;
}

h2{
    font-weight: 500;
    margin: 5px 0px 5px 0px;
}

h3{
    margin:5px 0px;
```

```
    font-weight: 500;
}


#main{
    width: 45%;
    margin: 20px auto 20px auto;
}

#details {
    width: 60%;
    margin-left: 20%;
}

table{
    background-color: #F2F3F4;
    border: 1px solid white;
    border-radius: 3px;
    border-collapse: collapse;
    width: 100%
}

tr{
    border: 2px solid white;
}

td{
    border: 2px solid white;
    width: 1%;
    padding: 5px;
}

#name{
    display: inline-block;
    margin: 2px 0px;
    font-style: italic;
}
```

```
td.desc{
    font-weight: 500;
}

#mainContent{
    margin: 20px;
}

#customer-form{
    border: 1px solid #c1c1c1;
    border-radius: 5px;
    background-color: white;
}

input.input{
    //width: 60%;
    height: 25px;
    padding: 0 10px;
    border:1px solid lightgray;
    border-radius: 3px;
}

.customButton {
  border-radius: 3px;
  border-style: solid;
  border-width: 0;
  cursor: pointer;
  font-family: "Helvetica Neue", "Helvetica", Helvetica, Arial, sans-
serif !important;
  position: relative;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  padding: 10px 12px;
  font-size: 14px;
  background-color: #739DB8;
```

```
  border-color: #007095;
  color: #FFFFFF;
  transition: background-color 200ms ease-out;
  margin: 5px 0px;
}

.customButton:hover{
    background-color: #007095;
}

input[type=number]{
    width: 190px;
    height: 25px;
}

input[type=date]{
    width: 190px;
    height: 25px;
}

.reportsButton {
  border-radius: 3px;
  border-style: solid;
  font-family: "Helvetica Neue", "Helvetica", Helvetica, Arial, sans-
serif !important;
  position: relative;
  text-align: center;
  display: inline-block;
  padding: 10px 12px;
  font-size: 14px;
  background-color: #739DB8;
  border-color: #007095;
  color: #FFFFFF;
  transition: background-color 200ms ease-out;
  margin: 5px 0px;
  width: 300px;
  margin-top: 10px;
```

```
}

.reportsButton:hover{
    background-color: #007095;
}
```

Κοινές για όλες τις σελίδες της βάσης είναι και οι κλάσεις. Παράδειγμα:

Bonus.java

```
package core.helpers;

import java.io.FileNotFoundException;
import static java.lang.Math.toIntExact;
import java.sql.Connection;
import java.sql.Date;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.concurrent.TimeUnit;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.http.HttpServletRequest;

public class Bonus {

    private String category;    // insurance or loan
    private int id;
    private String reduction;
    private String increment;

    //default constructor
    public Bonus(){ }

    //constructor with required fields only
    public Bonus(String category,
```

```
                int id,
                String reduction,
                String increment)
{
    this.category       = category;
    this.id             = id;
    this.reduction      = reduction;
    this.increment      = increment;
}

/* Getters and Setters */

public String getCategory() {
    return category;
}

public void setCategory(String category) {
    this.category = category;
}

public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public String getReduction() {
    return reduction;
}

public void setReduction(String reduction) {
    this.reduction = reduction;
}

public String getIncrement() {
```

```
        return increment;
    }

    public void setIncrement(String increment) {
        this.increment = increment;
    }

}
```

Ακόμη στο core package του project είναι τα αρχεία που υλοποιούν τα servlets του συστήματος για την σύνδεση και την επικοινωνία με τη βάση δεδομένων. Παράδειγμα ενός αρχείου java για servlet:

EditLifeInsuranceServlet.java

```
package core.servlets;

import core.helpers.LifeInsurance;
import java.io.IOException;
import java.sql.SQLException;
import java.text.ParseException;
import java.util.Enumeration;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebServlet("/EditLifeInsuranceServlet")

public class EditLifeInsuranceServlet extends HttpServlet {

    //Processes requests for both HTTP GET and POST methods.
```

```
    protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException,
InterruptedException, SQLException, ClassNotFoundException,
ParseException {

        response.setContentType("text/html;charset=UTF-8");
        request.setCharacterEncoding("UTF-8");

        LifeInsurance newLifeInsurance = parseParams(request);
        RequestDispatcher dispatcher;
        newLifeInsurance.updateDatabase(request);
        dispatcher                                                =
request.getRequestDispatcher("life_insurance.jsp");
        dispatcher.forward(request, response);
    }

    /* παίρνει τα values από τη φόρμα της html και τα βάζει σε ένα
insurance object και δίνει αυτό το insurance για update  */
    private LifeInsurance parseParams(HttpServletRequest request)
throws IOException, ServletException{

        Enumeration paramNames = request.getParameterNames();
        LifeInsurance insurance = new LifeInsurance();
        while (paramNames.hasMoreElements()) {
            String paramName = (String) paramNames.nextElement();
            String[]                  paramValues                 =
request.getParameterValues(paramName);
            String paramValue = paramValues[0];

            if (paramValue.length() == 0) {
                paramValue = null;
            }

            switch (paramName) {
                case "cust_name":
                    insurance.setCust_name(paramValue);
```

```
                    break;
                case "insurance_id":
                    insurance.setId(Integer.parseInt(paramValue));
                    break;
                case "instalments_not_paid":

insurance.setInstalments_not_paid(Integer.parseInt(paramValue));
                    break;
            }
        }

        return insurance;
    }

    //Handles the HTTP GET method.
    @Override
    protected       void       doGet(HttpServletRequest       request,
HttpServletResponse response)
            throws ServletException, IOException {
        try {
            processRequest(request, response);
        }    catch    (InterruptedException    |    SQLException    |
ClassNotFoundException ex) {

Logger.getLogger(InsertLifeInsuranceServlet.class.getName()).log(Leve
l.SEVERE, null, ex);
        } catch (ParseException ex) {

Logger.getLogger(InsertLifeInsuranceServlet.class.getName()).log(Leve
l.SEVERE, null, ex);
        }
    }

    //Handles the HTTP POST method.
    @Override
    protected      void      doPost(HttpServletRequest      request,
HttpServletResponse response)
```

```
            throws ServletException, IOException {

        try {
            processRequest(request, response);
        }    catch    (InterruptedException    |    SQLException    |
ClassNotFoundException ex) {

Logger.getLogger(InsertLifeInsuranceServlet.class.getName()).log(Leve
l.SEVERE, null, ex);
        } catch (ParseException ex) {

Logger.getLogger(InsertLifeInsuranceServlet.class.getName()).log(Leve
l.SEVERE, null, ex);
        }

    }
}
```

4.2.1 Σελίδα για τους πελάτες

Αφού γίνει η εκκίνηση του συστήματος στον περιηγητή εμφανίζεται η αρχική σελίδα της βάσης δεδομένων. Η αρχική σελίδα αφορά τους πελάτες της εταιρίας και είναι η παρακάτω.

Illustration 27: Αρχική 1

Για να γίνει μια εισαγωγή πελάτη πρέπει ο χρήστης να επιλέξει το κουπί Εισαγωγή Πελάτη που εμφανίζει την σελίδα εισαγωγής. Εκεί εισάγουμε τα δεδομένα και κάνουμε αποθήκευση. Ο νέος πελάτης έχει εισαχθεί στη βάση δεδομένων. Ο κώδικας που υλοποιεί την παραπάνω σελίδα είναι ο εξής:

```
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.Connection"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="core.helpers.DBConnection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>


</head>
    <title>Πελάτες</title>
```

```
    <link href="styles/main-style.css" rel="stylesheet"
type="text/css">
    <body>
        <ul>
            <li><a class="active" href="index.jsp">Πελάτες</a></li>
            <li><a href="life_insurance.jsp">Ασφάλειες Ζωής</a></li>
            <li><a href="loans.jsp">Δάνεια</a></li>
            <li class="dropdown">
                <a href="javascript:void(0)"
class="dropbtn">Αναφορές</a>
                <div class="dropdown-content">
                    <a href="reports_customers.jsp">Πελάτες</a>
                    <a
href="reports_insurance.jsp">Ασφαλιστήρια/Δάνεια</a>
                    <a href="reports_overall.jsp">Συγκεντρωτικές</a>
                </div>
            </li>
        </ul>

        <div id="mainContent">
            <div id="details">
                <h3>Πελάτες</h3><hr>
                <table>
                    <tr>
                        <td class="desc">
                            <label>Όνομα</label>
                        </td>
                         <td class="desc">
```

```
                        <label>Διεύθυνση</label>
                    </td>
                    <td class="desc">
                        <label>Ηλικία</label>
                    </td>
                    <td class="desc">
                        <label>Τηλέφωνο</label>
                    </td>
                    <td class="desc">
                        <label>Αριθμός Ταυτότητας</label>
                    </td>
                </tr>
                <%

                    String query = "SELECT * FROM CUSTOMERS";
                    Connection conn =
DBConnection.getConnection();
                    Statement stmt = conn.createStatement();
                    ResultSet rs = stmt.executeQuery(query);
                    while(rs.next()){

                %>
                <tr>
                    <td><%=rs.getString("CUST_NAME") %></td>
                    <td><%=rs.getString("ADDRESS") %></td>
                    <td><%=rs.getInt("AGE") %></td>
                    <td><%=rs.getInt("TELEPHONE_NUMBER") %></td>
```

```
                        <td><%=rs.getString("ID_NUMBER") %></td>
                    </tr>
                    <%
                        }
                        rs.close();
                        stmt.close();
                        conn.close();

                    %>
                    <tr>
                        <td><input type="button"
onclick="location.href='insert/customers_insert.jsp';"
class="customButton" name="submitButton" value="Εισαγωγή
Πελάτη"></td>
                    </tr>
                </table>
            </div>
            <hr style="width: 60%;">
        </div>

    </body>
</html>
```

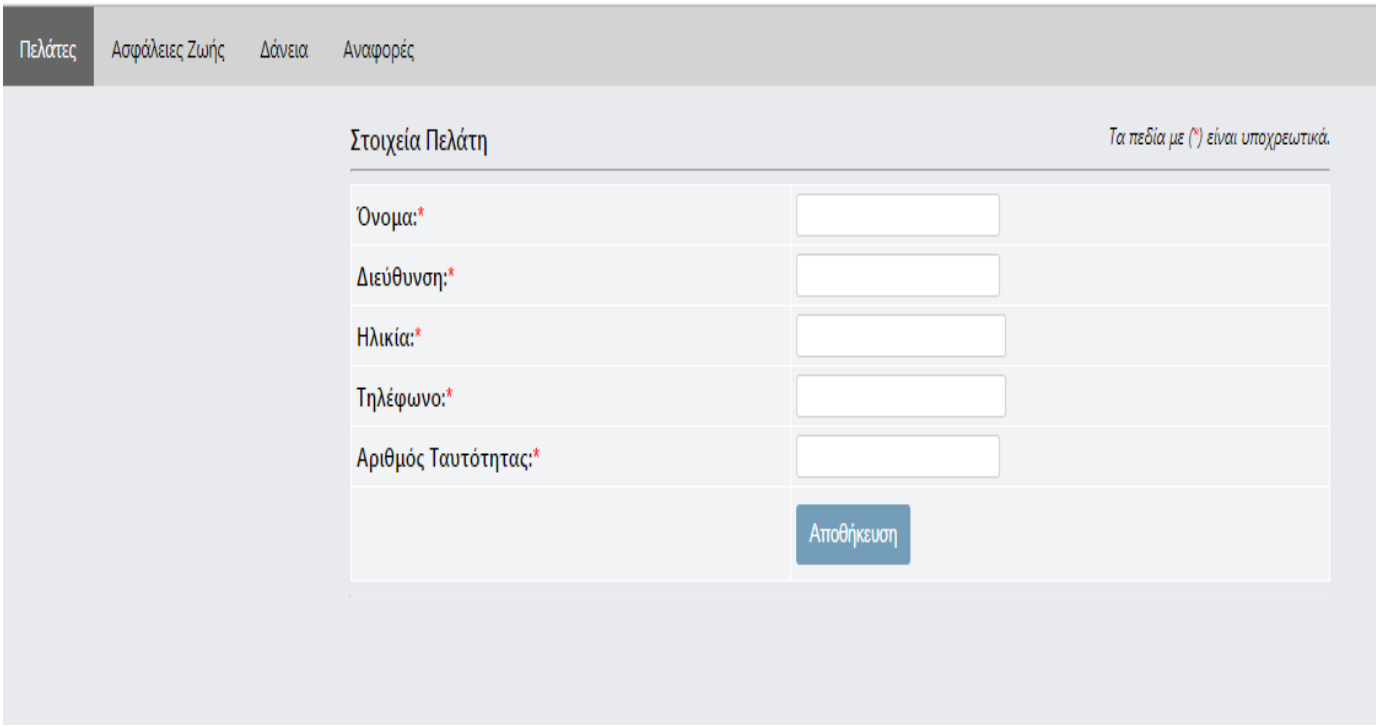


Illustration 28: Αρχική 2

Το αρχείο που υλοποιεί το εισαγωγή πελάτη που εμφανίζεται από πάνω είναι το customers_insert.jsp.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>


</head>
    <title>Πελάτες - Εισαγωγή</title>
    <link href="../styles/main-style.css" rel="stylesheet"
type="text/css">
    <body>
        <ul>
            <li><a class="active"
href="../index.jsp">Πελάτες</a></li>
            <li><a href="../life_insurance.jsp">Ασφάλειες
Ζωής</a></li>
```

```
            <li><a href="../loans.jsp">Δάνεια</a></li>
            <li class="dropdown">
                <a href="javascript:void(0)"
class="dropbtn">Αναφορές</a>
                <div class="dropdown-content">
                    <a href="../reports_customers.jsp">Πελάτες</a>
                    <a
href="../reports_insurance.jsp">Ασφαλιστήρια/Δάνεια</a>
                    <a
href="../reports_overall.jsp">Συγκεντρωτικές</a>
                </div>
            </li>
        </ul>

        <form id="register" action="../InsertCustomerServlet"
method="post" autocomplete="on">
            <div id="mainContent">
                <div id="details">
                    <h3>Στοιχεία Πελάτη<span style="float:right;
font-size: 13px; vertical-align: bottom;"><i>Τα πεδία με (<span
style="color: red;">*</span>) είναι υποχρεωτικά.</i></span></h3><hr>
                    <table>
                        <tr>
                            <td class="desc">
                                <label>Όνομα:<span style="color:
red;">*</span></label>
                            </td>
                            <td>
```

```
                                <input type="text" name="name"
class="input" required>
                            </td>
                        </tr>
                        <tr>
                            <td class="desc">
                                <label>Διεύθυνση:<span style="color:
red;">*</span></label>
                            </td>
                            <td>
                                <input type="text" name="address"
class="input" required>
                            </td>
                        </tr>
                        <tr>
                            <td class="desc">
                                <label>Ηλικία:<span style="color:
red;">*</span></label>
                            </td>
                            <td>
                                <input type="number" name="age"
class="input" min="0" max="100" required>
                            </td>
                        </tr>
                        <tr>
                            <td class="desc">
                                <label>Τηλέφωνο:<span style="color:
red;">*</span></label>
```

```
                            </td>
                            <td>
                                <input type="number"
name="tel_number" class="input" required>
                            </td>
                        </tr>
                        <tr>
                            <td class="desc">
                                <label>Αριθμός Ταυτότητας:<span
style="color: red;">*</span></label>
                            </td>
                            <td>
                                <input type="text" name="id_number"
class="input" required>
                            </td>
                        </tr>
                        <tr>
                            <td></td>
                            <td>
                                <input type="submit"
class="customButton" name="submitButton" value="Αποθήκευση">
                            </td>
                        </tr>
                    </table>
                </div>
                <hr style="width: 60%;">
            </div>
        </form>
```

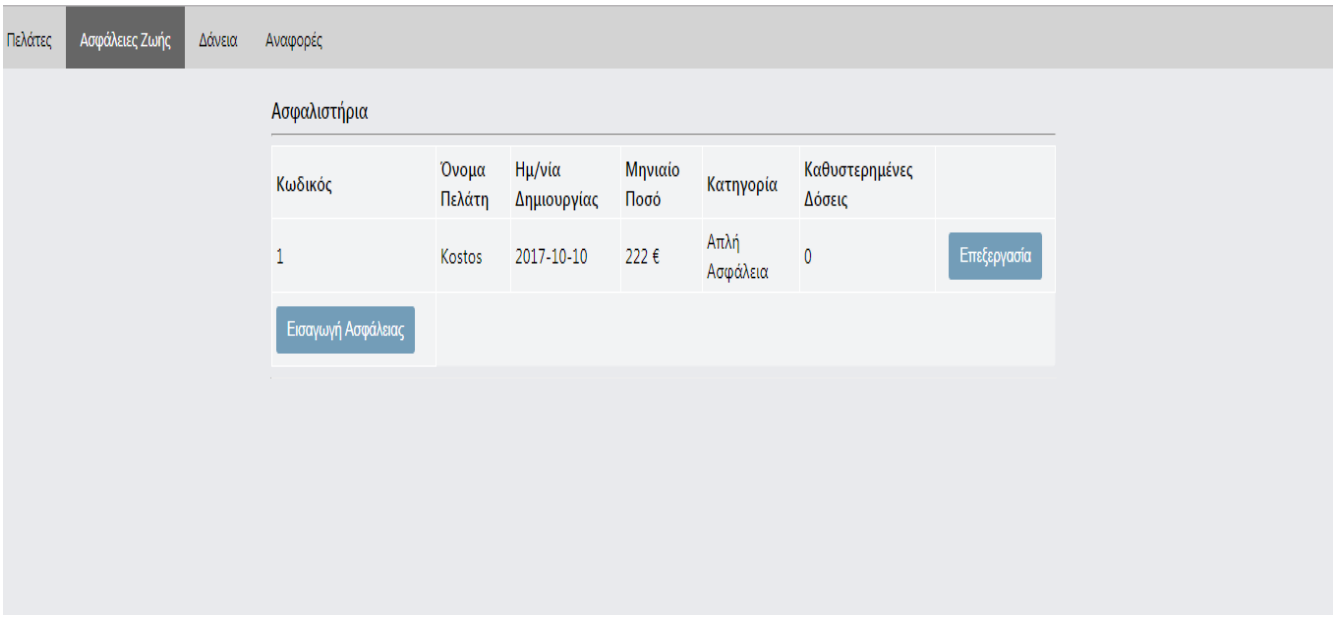


Illustration 29: Ασφάλειες 1

```
    </body>
</html>
```

4.2.2 Σελίδα για τις ασφάλειες της εταιρίας

Το σύστημα παρέχει τη δυνατότητα να γίνει εισαγωγή μιας ασφάλειας ζωής για τους πελάτες και έπειτα επεξεργασία αυτής.

Ο κώδικας που χρειάζεται για την παραπάνω σελίδας είναι ο εξής:

```
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.Connection"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="core.helpers.DBConnection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>

</head>
    <title>Ασφάλειες Ζωής</title>
```

```
    <link href="styles/main-style.css" rel="stylesheet"
type="text/css">
    <body>
        <ul>
            <li><a href="index.jsp">Πελάτες</a></li>
            <li><a class="active" href="life_insurance.jsp">Ασφάλειες
Ζωής</a></li>
            <li><a href="loans.jsp">Δάνεια</a></li>
            <li class="dropdown">
                <a href="javascript:void(0)"
class="dropbtn">Αναφορές</a>
                <div class="dropdown-content">
                    <a href="reports_customers.jsp">Πελάτες</a>
                    <a
href="reports_insurance.jsp">Ασφαλιστήρια/Δάνεια</a>
                    <a href="reports_overall.jsp">Συγκεντρωτικές</a>
                </div>
            </li>
        </ul>

        <div id="mainContent">
            <div id="details">
                <h3>Ασφαλιστήρια</h3><hr>
                <table>
                    <tr>
                        <td class="desc">
                            <label>Κωδικός</label>
                        </td>
```

```
                        <td class="desc">
                            <label>Όνομα Πελάτη</label>
                        </td>
                         <td class="desc">
                            <label>Ημ/νία Δημιουργίας</label>
                        </td>
                        <td class="desc">
                            <label>Μηνιαίο Ποσό</label>
                        </td>
                        <td class="desc">
                            <label>Κατηγορία</label>
                        </td>
                        <td class="desc">
                            <label>Καθυστερημένες Δόσεις</label>
                        </td>
                        <td></td>
                    </tr>
                    <%

                        String query = "SELECT * FROM
LIFE_INSURANCE";
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt = conn.createStatement();
                        ResultSet rs = stmt.executeQuery(query);
                        while(rs.next()){
```

```
                    %>
                    <tr>
                        <td><%=rs.getString("INSURANCE_ID") %></td>
                        <td><%=rs.getString("CUST_NAME") %></td>
                        <td><%=rs.getString("CONTRACT_DATE") %></td>
                        <td><%=rs.getInt("COST_MONTHLY") %> €</td>
                        <td>
                            <%
                                if (rs.getInt("COST_MONTHLY")>=0 &&
rs.getInt("COST_MONTHLY")<=80){
                                %>
                                    Ατυχήματος
                                <% }
                                else if (rs.getInt("COST_MONTHLY")>80
&& rs.getInt("COST_MONTHLY")<=130){
                                %>
                                    Μειούμενου Κεφαλαίου
                                <% }
                                else{
                                %>
                                    Απλή Ασφάλεια
                                <%
                                }
                            %>
                        </td>
                        <td><%=rs.getInt("INSTALMENTS_NOT_PAID")
%></td>
                        <td align="center">
```

```
                            <input type="button"
onclick="location.href='insert/life_insurance_edit.jsp?insurance_id=<
%=rs.getInt("INSURANCE_ID") %>';" class="customButton"
name="editButton" value="Επεξεργασία">
                        </td>
                    </tr>
                    <%
                        }
                        rs.close();
                        stmt.close();
                        conn.close();


                    %>
                    <tr>
                        <td><input type="button"
onclick="location.href='insert/life_insurance_insert.jsp';"
class="customButton" name="submitButton" value="Εισαγωγή
Ασφάλειας"></td>
                    </tr>
                </table>
            </div>
            <hr style="width: 60%;">
        </div>


    </body>
</html>
```

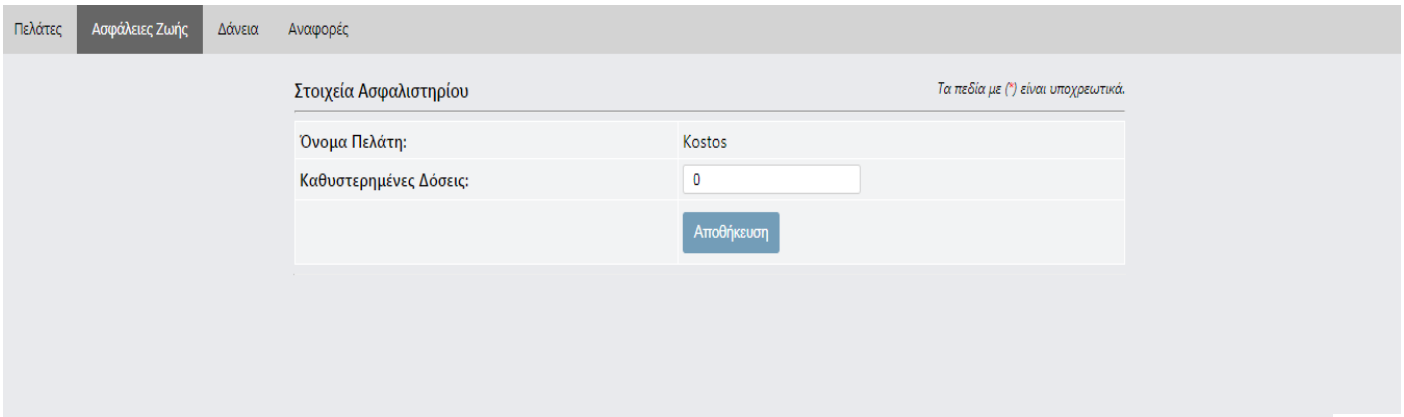


Illustration 31: Ασφάλειες 3

Illustration 30: Ασφάλειες 2

Ο κώδικας που υλοποιεί την επεξεργασία μιας ασφάλειας είναι ο παρακάτω:

```
<%@page import="java.sql.ResultSet"%>
<%@page import="core.helpers.DBConnection"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.Connection"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>


</head>
    <title>Ασφάλειες Ζωής - Επεξεργασία</title>
    <link       href="../styles/main-style.css"       rel="stylesheet"
type="text/css">
```

```
    <body>
        <ul>
            <li><a href="../index.jsp">Πελάτες</a></li>
            <li><a                                    class="active"
href="../life_insurance.jsp">Ασφάλειες Ζωής</a></li>
            <li><a href="../loans.jsp">Δάνεια</a></li>
            <li class="dropdown">
                <a                            href="javascript:void(0)"
class="dropbtn">Αναφορές</a>
                <div class="dropdown-content">
                    <a href="../reports_customers.jsp">Πελάτες</a>
                    <a
href="../reports_insurance.jsp">Ασφαλιστήρια/Δάνεια</a>
                    <a
href="../reports_overall.jsp">Συγκεντρωτικές</a>
                </div>
            </li>
        </ul>

        <form    id="register"    action="../EditLifeInsuranceServlet"
method="post" autocomplete="on">
            <div id="mainContent">
                <div id="details">
                    <h3>Στοιχεία                    Ασφαλιστηρίου<span
style="float:right;  font-size:  13px;  vertical-align:  bottom;"><i>Τα
πεδία     με     (<span     style="color:     red;">*</span>)     είναι
υποχρεωτικά.</i></span></h3><hr>
                    <table>
                        <%
```

```
                            String query = "SELECT * FROM
LIFE_INSURANCE WHERE INSURANCE_ID=" +
Integer.parseInt(request.getParameter("insurance_id")) + "";
                            Connection conn =
DBConnection.getConnection();
                            Statement stmt = conn.createStatement();
                            ResultSet rs = stmt.executeQuery(query);
                            while(rs.next()){
                        %>
                        <tr>
                            <td class="desc">
                                <label>Όνομα Πελάτη:</label>
                            </td>
                            <td>
                                <%=rs.getString("CUST_NAME") %>
                            </td>
                        </tr>
                        <tr>
                            <td class="desc">
                                <label>Καθυστερημένες Δόσεις:</label>
                            </td>
                            <td>
                                <input type="number" min="0"
name="instalments_not_paid"
value="<%=rs.getInt("INSTALMENTS_NOT_PAID") %>" class="input">
                            </td>
                        </tr>
```

```
                        <input    type="hidden"    name="insurance_id"
value="<%=rs.getInt("INSURANCE_ID") %>">
                        <input     type="hidden"     name="cust_name"
value="<%=rs.getString("CUST_NAME") %>">                              <%
}
                            rs.close();
                            stmt.close();
                            conn.close();
                        %>
                        <tr>
                            <td></td>
                            <td>
                                <input                    type="submit"
class="customButton" name="submitButton" value="Αποθήκευση">
                            </td>
                        </tr>
                    </table>
                </div>
                <hr style="width: 60%;">
            </div>
        </form>
    </body>
</html>
```

4.2.3 Σελίδα για τα δάνεια της εταιρίας

Το σύστημα πρέπει να υποστηρίζει την χορήγηση δανείων στους πελάτες της εταιρίας. Αρχικά η σελίδα θα είναι κενή:

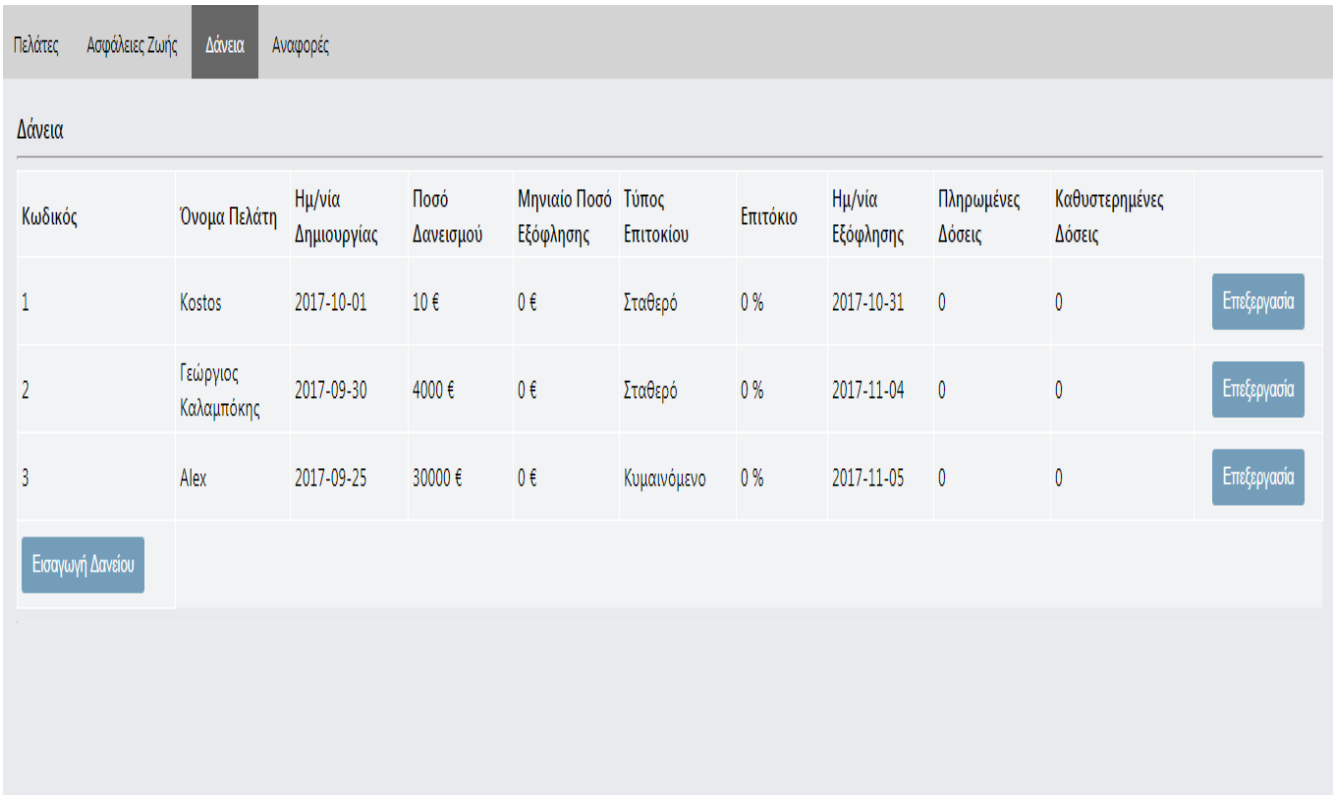


Illustration 34: Δάνεια 3

Επιλέγοντας την Εισαγωγή Δανείου μπορεί ο χρήστης ή ο διαχειριστής να εισάγει ένα νέο δάνειο για κάποιον πελάτη της εταιρίας.

Έτσι η σελίδα με τα δάνεια γεμίζει ως εξής:

Ο κώδικας που υλοποιεί την σελίδα για τα δάνεια είναι ο παρακάτω:

```
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.Connection"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="core.helpers.DBConnection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
```

```
</head>
    <title>Δάνεια</title>
    <link        href="styles/main-style.css"        rel="stylesheet"
type="text/css">
    <body>
        <ul>
            <li><a href="index.jsp">Πελάτες</a></li>
            <li><a href="life_insurance.jsp">Ασφάλειες Ζωής</a></li>
            <li><a class="active" href="loans.jsp">Δάνεια</a></li>
            <li class="dropdown">
                <a                          href="javascript:void(0)"
class="dropbtn">Αναφορές</a>
                <div class="dropdown-content">
                    <a href="reports_customers.jsp">Πελάτες</a>
                    <a
href="reports_insurance.jsp">Ασφαλιστήρια/Δάνεια</a>
                    <a href="reports_overall.jsp">Συγκεντρωτικές</a>
                </div>
            </li>
        </ul>

        <div id="mainContent">
            <div id="details" style="width: 100%; margin-left:0;">
                <h3>Δάνεια</h3><hr>
                <table>
                    <tr>
                        <td class="desc">
```

```
    <label>Κωδικός</label>
</td>
<td class="desc">
    <label>Όνομα Πελάτη</label>
</td>
<td class="desc">
    <label>Ημ/νία Δημιουργίας</label>
</td>

<td class="desc">
    <label>Ποσό Δανεισμού</label>
</td>
<td class="desc">
    <label>Μηνιαίο Ποσό Εξόφλησης</label>
</td>
<td class="desc">
    <label>Τύπος Επιτοκίου</label>
</td>
<td class="desc">
    <label>Επιτόκιο</label>
</td>
<td class="desc">
    <label>Ημ/νία Εξόφλησης</label>
</td>
<td class="desc">
    <label>Πληρωμένες Δόσεις</label>
</td>
```

```
                        <td class="desc">
                            <label>Καθυστερημένες Δόσεις</label>
                        </td>
                        <td class="desc">
                        </td>
                    </tr>
                    <%

                        String query = "SELECT * FROM LOANS";
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt = conn.createStatement();
                        ResultSet rs = stmt.executeQuery(query);
                        while(rs.next()){

                    %>
                    <tr>
                        <td><%=rs.getString("LOAN_ID") %></td>
                        <td><%=rs.getString("CUST_NAME") %></td>
                        <td><%=rs.getString("LOAN_DATE") %></td>
                        <td><%=rs.getInt("LOAN_MONEY") %> €</td>
                        <td><%=rs.getString("CUST_MONTHLY_PAY")    %>
€</td>
                        <td>
                            <%
                                if
(rs.getString("INTEREST_TYPE").equals("fixed")){%>
                                    Σταθερό
```

```
                                <%
                                }
                                else { %>
                                    Κυμαινόμενο
                                <% } %>
                            </td>
                            <td><%=rs.getString("INTEREST_RATE")       %>
%</td>
                            <td><%=rs.getString("REPAYMENT_DATE") %></td>
                            <td><%=rs.getInt("INSTALMENTS_PAID") %></td>
                            <td><%=rs.getInt("INSTALMENTS_NOT_PAID")
%></td>
                            <td align="center">
                                <input                          type="button"
onclick="location.href='insert/loans_edit.jsp?loan_id=<%=rs.getString
("LOAN_ID")     %>';"     class="customButton"     name="editButton"
value="Επεξεργασία">
                            </td>
                        </tr>
                        <%
                            }
                            rs.close();
                            stmt.close();
                            conn.close();

                        %>
                        <tr>
```

```
                        <td><input                          type="button"
onclick="location.href='insert/loans_insert.jsp';"
class="customButton"        name="submitButton"        value="Εισαγωγή
Δανείου"></td>
                    </tr>
                </table>
            </div>
            <hr style="width: 100%;">
        </div>


    </body>
</html>
```

4.2.4 Αναφορές

Για τις ανάγκες της πτυχιακής πρέπει να είναι σε θέση το σύστημα να εμφανίσει αναφορές σχετικά με:

- πελάτες που έχουν καθυστερήσει δόσεις
- πελάτες που είναι συνεπείς
- πελάτες που χρωστάνε δόσεις
- ασφαλιστήρια και δάνεια που έχουν δοθεί από την εταιρία
- συγκεντρωτικές αναφορές σχετικά που αφορούν:
  - Αριθμό πελατών ανα ασφαλιστήριο
  - Ασφαλιστήρια και δάνεια ανα μήνα

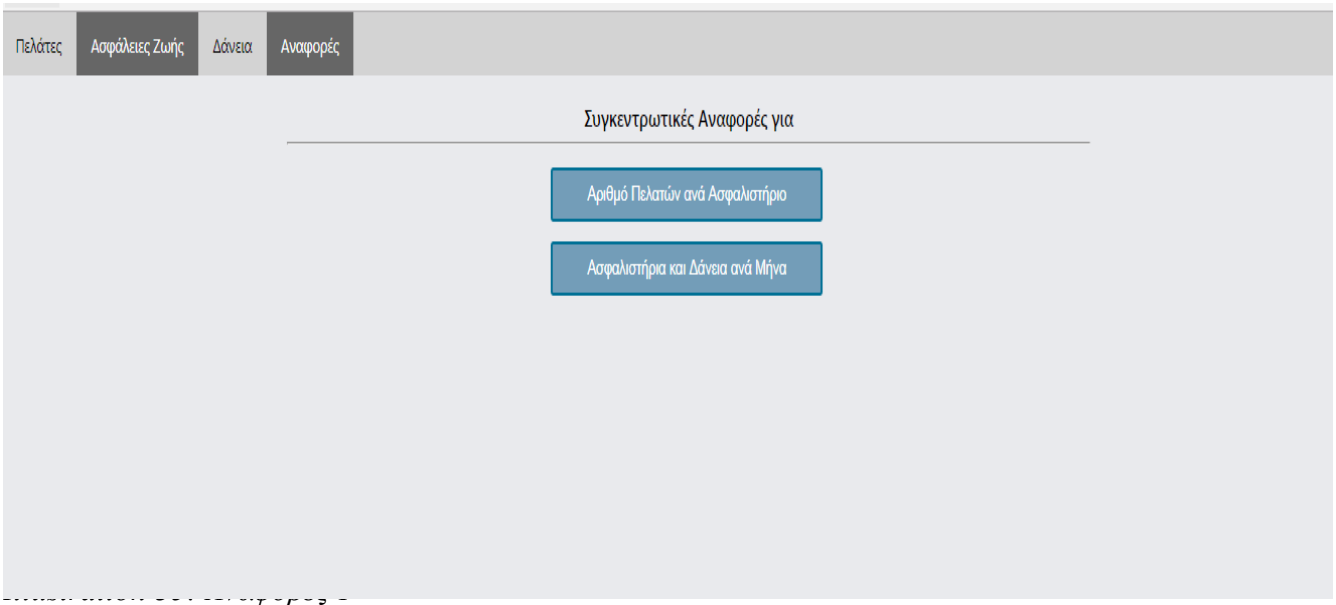


Illustration 37: Αναφορές 3

Παράδειγμα κώδικα που υλοποιεί τη σελίδα για τις αναφορές είναι ο παρακάτω:

```
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.Connection"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="core.helpers.DBConnection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>


</head>
    <title>Αναφορές - Πελάτες</title>
    <link      href="../styles/main-style.css"      rel="stylesheet"
type="text/css">
    <body>
```

```
        <ul>
            <li><a href="../index.jsp">Πελάτες</a></li>
            <li><a              href="../life_insurance.jsp">Ασφάλειες
Ζωής</a></li>
            <li><a href="../loans.jsp">Δάνεια</a></li>
            <li class="dropdown">
                <a    class="active"        href="javascript:void(0)"
class="dropbtn">Αναφορές</a>
                <div class="dropdown-content">
                    <a href="../reports_customers.jsp">Πελάτες</a>
                    <a
href="../reports_insurance.jsp">Ασφαλιστήρια/Δάνεια</a>
                    <a
href="../reports_overall.jsp">Συγκεντρωτικές</a>
                </div>
            </li>
        </ul>

        <div id="mainContent">
            <div id="details">
                <h3>Πελάτες που χρωστάνε δόσεις</h3><hr>
                <table>
                    <tr>
                        <td class="desc">
                            <label>Όνομα Πελάτη</label>
                        </td>
                    </tr>
                    <%
```

```
                        String query = "SELECT DISTINCT
LOANS.CUST_NAME FROM LOANS LEFT JOIN LIFE_INSURANCE ON
LOANS.CUST_NAME=LIFE_INSURANCE.CUST_NAME WHERE
LOANS.INSTALMENTS_NOT_PAID>0 OR
LIFE_INSURANCE.INSTALMENTS_NOT_PAID>0";
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt = conn.createStatement();
                        ResultSet rs = stmt.executeQuery(query);
                        while(rs.next()){

                    %>
                    <tr>
                        <td><%=rs.getString("CUST_NAME") %></td>
                    </tr>
                    <%
                        }
                        rs.close();
                        stmt.close();
                        conn.close();
                    %>
                </table>
            </div>
            <hr style="width: 60%;">
        </div>

    </body>
```

```
</html>
```

5 Βιβλιογραφία

(1) http://www.javaguicodexample.com/installusejavanetbeans1.html

(2) http://www.vogella.com/tutorials/JavaWebTerminology/article.html

(3) http://www.oracle.com/technetwork/articles/java/webapps-1-138794.html

(4) https://en.wikipedia.org/wiki/Web_development

(5) https://en.wikipedia.org/wiki/Java_servlet

(6) http://iiti.ac.in/people/~tanimad/JavaTheCompleteReference.pdf

(7) https://www.w3schools.com/sql/

(8) https://el.wikipedia.org/wiki/SQL

(9) https://en.wikipedia.org/wiki/NetBeans