ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή Εργασία

ΑΝΑΖΗΤΗΣΗ ΕΠΙΣΤΙΜΟΝΗΚΩΝ ΕΓΓΡΑΦΩΝ ΜΕ INTERFACE JAVA ΚΑ Ι PROLOG

Της φοιτήτριας **Επιβλέπων Καθηγητής**

Λεμονή-Σκαρπαλέτζου Βασιλικη **Βοζαλής Εμμανουήλ**

Αρ.Μητρώου: 05/2760

1 ΠΡΟΛΟΓΟΣ

Στην εποχή μας με την έκρηξη της πληροφορίας αλλά και την ανάπτηξη της πληροφορικής σαν επιστήμη πλέων έχουμε οργανώσει τη πληροφορία με τέτοιο τρόπο που καθημερινά για να καλύψουμε τις ανάγκες μας για πληροφορίες,στις περισσότερες περιπτώσεις μπαίνουμε σε μια από τις μεγάλες μηχανές αναζήτησης, πληκτρολογούμε την λέξη ή την φράση προς αναζήτηση και πατάμε το κουμπί αναζήτηση .Μετά από κάποια δευτερόλεπτα η μηχανή αναζήτησης θα μας επιστρέψει μια λίστα με links που σχετίζονται με την λέξη ή φράση που αναζητούσαμε .Βέβαια οι μηχανές αναζήτησης δεν υπήρχαν πάντα.

Όταν πρώτο ξεκίνησε το internet δεν είχε καμία σχέση με το internet που γνωρίζουμε τώρα. Δεν υπήρχε αυτή η πληθώρα των ιστοσελίδων ούτε θεωρούνταν μια από τις μεγαλύτερες επιχειρηματικές δραστηριότητες. Στην αρχή ήταν απλά ένας αριθμός από Ftp (**File transfer protocol**) sites που χρήστες μπορούσαν να κατεβάσουν ή να ανεβάσουν αρχεία. Ακόμα και η εύρεση αυτών των αρχείων ήταν δύσκολη αφού έπρεπε να γνωρίζεις την ακριβή διεύθυνση που βρίσκονταν.

Αυτή η διαδικασία εύρεσης των αρχείων ήταν πολύ δύσκολη διαδικασία, χρονοβόρα και έπρεπε να έχεις αρκετή υπομονή. Αυτό γινόταν πριν ο μαθητής, Alan Emtage 1990, δημιουργήσει το πρώτο εργαλείο αναζήτησης (**Search Engine Tool**). Αυτό που έφτιαξε ήταν ένας κατάλογος από τα αρχεία που υπήρχαν στο internet και ονομάστηκε **Archie** (Archieve).

Το 1991 ένας άλλος μαθητής ο Mark McCahill συνειδητοποίησε ότι αφού μπορείς να ψάχνεις τα αρχεία στο internet τότε μπορείς να ψάχνεις και τα κείμενα. Έτσι δημιούργησε το **Gopher**, ένα πρόγραμμα που κατηγοριοποιούσε το απλό κείμενο των αρχείων που αργότερα έγιναν οι πρώτες ιστοσελίδες.

Με την δημιουργία του Gopher, προέκυψε η ανάγκη για προγράμματα που μπορούσαν να εντοπίσουν πληροφορίες μέσα από τους καταλόγους του Gopher. Έτσι δημιουργήθηκε το πρόγραμμα **Veronica** (Very Easy Rodent-Oriented Net-wide Index to Computerized Archives) και το **jughead** (Jonzy's Universal Gopher Hierarchy Excavation and Display) για να αναζητούν αρχεία και κείμενα που είχαν αποθηκευτεί στο Gopher σύστημα.

Και τα δυο αυτά προγράμματα λειτουργούσαν με τον ίδιο τρόπο, επιτρέποντας στους χρήστες να ψάξουν τους καταλόγους με τις πληροφορίες δίνοντας μια λέξη ή φράση (Keywords). Η πρώτη **μηχανή αναζήτησης (Search Engine)** με την μορφή που γνωρίζουμε τις μηχανές σήμερα, δημιουργήθηκε το 1993 από τον Mathew Gray και ονομάστηκε Wandex. Ήταν το πρώτο πρόγραμμα που δημιουργούσε και καταλόγους αλλά και έψαχνε τις ιστοσελίδες στο internet. Από το 1993 μέχρι το 1998 δημιουργήθηκαν όλες οι μεγάλες μηχανές αναζήτησης (**Search Engines**) που γνωρίζουμε και σήμερα.

- Excite 1993
- Yahoo 1994
- WebCrawler 1994
- Lycos 1994

- Infoseek 1995
- AltaVista 1995
- Iktomi 1996
- Askjeeves 1997
- Google 1997
- MSN Search 1998

Σήμερα,οι **μηχανές αναζήτησης (Search Engines)** είναι πολύπλοκα προγράμματα που μας επιτρέπουν να αναζητήσουμε οποιαδήποτε πληροφορία από αρχεία, έγγραφα, εικόνες, video μέχρι λέξεις ή φράσεις που χρησιμοποιούμε στην καθημερινότητα μας.

2 ΠΕΡΙΛΗΨΗ

Η παρακάτω πτυχιακή πραγματεύεται την δημιουργία μιας μηχανής αναζήτησης επιστημονικών εγγράφων από τη βάση δεδομένων της amazon. Για την υλοποίηση της χρησιμοποιήθηκαν η java η java swing και η prolog. Όλες αυτές οι γλώσσες προγραμματισμού και οι ιδιαιτερότητες της κάθε μιας αναλύονται στα επόμενα κεφάλαια καθώς και τα μέσα της διασύνδεσης τους, μαζί με αυτά γίνεται αναφορά και σε δύο τεχνολογίες που διευκολύνουν το προγραμματιστικό μας έργο (get, maven).Τέλος περιγράφεται ο τρόπος λειτουργίας του project και επισημαίνονται κάποια βασικά του στοιχεία.

Λέξεις Κλειδιά: γλώσσες προγραμματισμού, μηχανή αναζήτησης

3 ABSTRACT

This following graduation project it's about the creation of a search engine scientific documents from database Amazon. For the realization of it, it's been used java, swing java and prolog. All these programming languages and specific characteristics of each analyzed in the next chapters and the means of interconnection, along with those is made a reference by two technologies that facilitate our developers work (get, maven). In the end, the mode project and identifies some key elements are also described.

Key words: programming languages, search engine

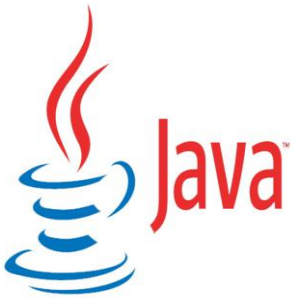
4 ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΕΥΡΕΤΗΡΙΟ ΕΙΚΟΝΩΝ

5 Η JAVA ΣΑΝ ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Σε αυτό το κεφάλαιο θα αναφερθούμε γενικά στη java θα δούμε διεξοδικά τους λόγους που συντέλεσαν στη δημιουργία αυτής της γλώσσας προγραμματισμού και από ποιόν φυσικά δημιουργήθηκε. Στη συνέχεια μιλάμε για την ιστορία της και τα στάδια εξέλιξης της στον χρόνο. Από εκεί και πέρα αναφερόμαστε στα διάφορα εργαλεία της γλώσσας καθώς και διευκρινίζουμε τις διαφορές ανάμεσα στα applets και τις applications και τέλος προβάλουμε την συμβολή της στο Internet που είναι πολύ σημαντική.

5.1 ΠΟΙΑ ΑΝΑΓΚΗ ΜΑΣ ΟΔΗΓΗΣΕ ΣΤΗ JAVA



Στις αρχές του 1991, η *A* αναζητούσε το κατάλληλο εργαλείο για να αποτελέσει την πλατφόρμα ανάπτυξης λογισμικού σε μικρό-συσκευές (έξυπνες οικιακές συσκευές έως πολύπλοκα συστήματα παραγωγής γραφικών). Τα εργαλεία της εποχής ήταν γλώσσες όπως η C++ και η C. Μετά από διάφορους πειραματισμούς προέκυψε το συμπέρασμα ότι οι υπάρχουσες γλώσσες δεν μπορούσαν να καλύψουν τις ανάγκες τους. Ο "πατέρας" της Java, James Gosling, που εργαζόταν εκείνη την εποχή για την Sun, έκανε ήδη πειραματισμούς πάνω στη C++ και είχε παρουσιάσει κατά καιρούς κάποιες πειραματικές γλώσσες (C++ ++) ως πρότυπα για το νέο εργαλείο που αναζητούσαν στην *Sun*. Τελικά μετά από λίγο καιρό κατέληξαν με μια πρόταση για το επιτελείο της εταιρίας, η οποία ήταν η γλώσσα *Oak*. Το όνομά της το πήρε από το ομώνυμο δένδρο (βελανιδιά) το οποίο ο Gosling είχε έξω από το γραφείο του και έβλεπε κάθε μέρα.

5.2 ΙΣΤΟΡΙΑ ΤΗΣ JAVA

Η γλώσσα της Java αρχικά αναπτύχθηκε από την Sun Microsystems υπό την αιγίδα των James Gosling και Bill Joy, σαν μέρος ενός ερευνητικού έργου ανάπτυξης λογισμικού για ηλεκτρονικές συσκευές καταναλωτικού επιπέδου (π.χ video,τηλεόραση). Ο τρόπος αυτός χρησιμοποίησης της, την μετέτρεψε σε μία ιδανική γλώσσα για την διανομή εκτελέσιμων προγραμμάτων μέσω του WWW

(Παγκόσμιου Ιστού) καθώς επίσης σε μία γενικού σκοπού γλώσσα προγραμματισμού για την ανάπτυξη προγραμμάτων τα οποία θα είναι εύκολα στην χρήση και θα μπορούν να μεταφέρονται σε διαφορετικά λειτουργικά συστήματα.

Αν και χρησιμοποιήθηκε από την Sun σε πολλές εφαρμογές (με το όνομα Οακ) με σκοπό την δημιουργία προϊόντων για την ηλεκτρονική αγορά, ενδιαφέρον προκάλεσε στο κοινό μόνο όταν συνδυάστηκε με το πρόγραμμα ανάγνωσης ιστοσελίδων (browser) της HotJava. Αυτό είχε ως αποτέλεσμα, η γλώσσα αυτή να συνδεθεί στενά με την ανάπτυξη μικρό - εφαρμογών στο διαδίκτυο και συγκεκριμένα στον Παγκόσμιο Ιστό (WWW). Η πραγματική όμως απογείωση της δημοτικότητας της Java ξεκίνησε όταν η Netscape ενσωμάτωσε την δυνατότητα της HotJava να τρέχει μικρό- εφαρμογές μέσα στο δικό της πρόγραμμα ανάγνωσης ιστοσελίδων (browser).

Το γεγονός, ότι τα τελευταία χρόνια η Java χρησιμοποιείται κυρίως για την ανάπτυξη μικρό - εφαρμογών δεν σημαίνει ότι η γλώσσα αυτή δεν είναι κατάλληλη για την δημιουργία ολοκληρωμένων εφαρμογών. Εξάλλου αυτό αποδεικνύεται από το γεγονός ότι τα περισσότερα εργαλεία της έχουν γραφτεί με την Java. Μάλιστα, σύμφωνα με την θεωρεία ανάπτυξης μεταγλωττιστών, μία γλώσσα έχει "ενηλικιωθεί" όταν ο μεταγλωττιστής της μπορεί να γραφτεί από την ίδια την γλώσσα. Συνεπώς η Java ως γλώσσα προγραμματισμού έχει ενηλικιωθεί.

Αυτό που θα πρέπει κανείς να θυμάται για την Java είναι ότι πρόκειται για μία καλά οργανωμένη και καλά εκφρασμένη γλώσσα προγραμματισμού που έχει δανειστεί πολλά θετικά χαρακτηριστικά από άλλες γλώσσες όπως τη C++, τη SmallTalk και τη Lisp, αφαίρεσε όμως όλα εκείνα τα στοιχεία αυτών των γλωσσών που ίσως οδηγούσαν σε σύγχυση τους χρήστες.

5.3 ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΗΣ JAVA

Η Java χαρακτηρίζεται από τα εξής: απλή, αντικειμενοστραφής, συμβατή με δικτυακά πρωτόκολλα, ουδέτερη της υποκείμενης αρχιτεκτονικής, φορητή ασφαλής, υψηλής απόδοσης, δυναμική, σταθερή, interpreted και multithreaded. Στις ακόλουθες παραγράφους θα αναλύσουμε καθεμία από αυτές τις έννοιες.

5.3.1 ΑΠΛΗ

Στόχος της ομάδας της Sun που ανέπτυξε την Java, ήταν μια γλώσσα εύκολη στην χρήση, που δεν απαιτεί πολλή εξάσκηση και εκπαίδευση. Οι περισσότεροι προγραμματιστές στις μέρες μας δουλεύουν είτε με την C είτε με την C++. Έτσι, μολονότι η C++ δεν ήταν η κατάλληλη για το αρχικό σχέδιο, η Java σχεδιάστηκε βάσει της C++, με σκοπό να γίνει όσο το δυνατόν περισσότερο κατανοητή.

Η Java παραλείπει πολλά από τα σπανίως χρησιμοποιούμενα και δυσκολονόητα χαρακτηριστικά της C++, που δεν ωφελούν και πολύ την ευελιξία της γλώσσας. Προστέθηκαν διεργασίες, όπως η αυτόματη συλλογή των "σκουπιδιών" *(automatic garbage collection)*, διευκολύνοντας τον προγραμματισμό σε Java. Μια κοινή πηγή πολυπλοκότητας της C++ και της C είναι η διαχείριση της μνήμης. Με την καινούργια διεργασία της αυτόματης συλλογής "σκουπιδιών", που συνιστάται από την περιοδική αποδέσμευση της μνήμης που δεν χρησιμοποιείται, μεγάλο μέρος από την δουλεία των προγραμματιστών αυτοματοποιείται και μειώνονται τα bugs.

Ένα πλεονέκτημα της Java που οφείλεται στην απλότητα της είναι ότι το μέγεθος των απαραίτητων εργαλείων. Ο *Java interpreter* και οι βασικές βιβλιοθήκες είναι μικρές και ο κώδικάς της Java είναι τόσο περιορισμένος σε μέγεθος που μπορεί άνετα να τρέξει σε οποιαδήποτε μικρή μηχανή και να κατέβει από το δίκτυο.

5.3.2 ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ

Λέγοντας ότι μία γλώσσα προγραμματισμού είναι αντικειμενοστραφής, εννοούμε η τεχνική σχεδιασμού ενός προγράμματος συγκεντρώνεται σε αντικείμενα. Ένα αντικείμενο είναι ο συνδυασμός δεδομένων, διαδικασιών και λειτουργιών με βασική ιδιότητα την απόκρυψη του συνδυασμού αυτού. Το κάθε αντικείμενο, δηλαδή, αντιμετωπίζεται σαν ένα "μαύρο κουτί". Τα αντικείμενα δεν

είναι ανεξάρτητα μεταξύ τους, αλλά βρίσκονται σε σχέση αλληλεξάρτησης με τα υπόλοιπα. Υπάρχει η έννοια της κληρονομικότητας μεταξύ των αντικειμένων, δηλαδή ένα αντικείμενο μπορεί να κληρονομήσει δεδομένα από άλλα αντικείμενα.

Οι γλώσσες αντικειμενοστραφή προγραμματισμού είναι γλώσσες υψηλού επιπέδου, αφαιρετικές, αποτελεσματικές, γρήγορες και χρησιμοποιούνται για την δημιουργία μεγάλων και σημαντικών εφαρμογών. Οι αντικειμενοστραφής ευκολίες της Java είναι ίδιες με αυτές της C++, με επεκτάσεις από την Objectine C.

5.3.3 ΣΥΜΒΑΤΗ ΜΕ ΔΙΚΤΥΑ

Η Java έχει μια μεγάλη βιβλιοθήκη από ρουτίνες για την επιτυχημένη συνεργασία με τα πρωτόκολλα HTTP και FTP. Κατ 'αυτόν τον τρόπο, οι δικτυακές συνδέσεις δημιουργούνται ευκολότερα από ότι με την C ή την C++. Τα προγράμματα σε Java μπορούν να έχουν πρόσβαση μέσω δικτύου σε αντικείμενα , με την ίδια άνεση που ένας χρήστης προσπελάζει ένα τοπικό σύστημα αρχείων.

5.3.4 ΣΤΑΘΕΡΗ

Η Java προορίζεται για την σύνταξη προγραμμάτων που θα είναι αξιόπιστα από όλες τις πλευρές. Δίνεται έμφαση στον από νωρίς έλεγχο για πιθανά προβλήματα και στον έλεγχο σε πραγματικό χρόνο και στην εξάλειψη καταστάσεων που προκαλούν λάθη.

Η μεγαλύτερη διαφορά μεταξύ Java και C/C++ είναι το γεγονός ότι η Java έχει ένα μοντέλο δεικτών που εξαφανίζει την πιθανότητα της επαναχρησιμοποίησης της μνήμης και την καταστροφή των δεδομένων. Αντί για αριθμητικούς δείκτες *(pointer arithmetic)*, η Java έχει πραγματικούς πίνακες *(true arrays)*. Οι προγραμματιστές της Java δεν έχουν να φοβηθούν την ακούσια (ή μη) τροποποίηση της μνήμης, γιατί δεν υπάρχουν δείκτες *(pointers)*. Εξάλλου, τα προγράμματα σε Java δεν μπορούν να αποκοτήσουν μη εγκεκριμένη πρόσβαση στην μνήμη.

5.3.5 ΑΣΦΑΛΗΣ

Η Java προορίζεται για χρήση σε ανοικτά, δικτυωμένα περιβάλλοντα. Γι' αυτό το λόγο, ιδιαίτερη προσοχή έχει δοθεί στην ασφάλεια που παρέχει η γλώσσα. Η Java επιτρέπει την κατασκευή προγραμμάτων ελεύθερων από ιούς και η τροποποίηση τους είναι αδύνατη. Οι τεχνικές πιστοποίησης ταυτότητας βασίζονται στην ασύμμετρη κρυπτογραφία.

Υπάρχει μεγάλη σχέση μεταξύ του τρόπου διαχείρισης της μνήμης και της παρεχόμενης ασφάλειας. Αλλαγές στην σημασιολογία των δεικτών της μνήμης κάνουν αδύνατη την μη έγκυρη πρόσβαση στα δεδομένα της μνήμης ή της πρόσβασης των δεδομένων των αντικειμένων. Με αυτόν τον τρόπο καταπολεμούνται οι περισσότεροι ιοί.

Κάθε φορά που κατεβάζετε ένα «κανονικό» πρόγραμμα από το Internet στον υπολογιστή σας είστε πλέον υποψιασμένοι ότι κινδυνεύετε να μολυνθείτε από κάποιον ιό . Πριν από τη java οι περισσότεροι χρήστες δεν κατέβαζαν συχνά εκτελέσιμα προγράμματα . Όσοι όμως έκαναν τέτοιες ενέργειες συχνά σαρώνανε τα προγράμματά τους με ειδικά εργαλεία – προγράμματα αντιμετώπισης των ιών πριν από την εκτέλεση των εν λόγω προγραμμάτων . Ακόμη και σήμερα ,οι περισσότεροι χρήστες εξακολουθούν να ανησυχούν για το ενδεχόμενο μόλυνσης των συστημάτων τους από κάποιον ιό ή από το ενδεχόμενο εκτέλεσης ενός ύποπτου προγράμματος που μπορεί να προκαλέσει ακόμα και την καταστροφή των συστημάτων τους .(Υπάρχουν επίσης προγράμματα το οποία μπορούν να συλλέξουν πληροφορίες άκρος εμπιστευτικές όπως για παράδειγμα αριθμούς πιστωτικών καρτών λογαριασμούς ταμιευτηρίου στις τράπεζες και κωδικούς πρόσβασης .Τα προγράμματα αυτά μπορούν να πετύχουν τους στόχους τους αναζητώντας τα περιεχόμενα του τοπικού συστήματος αρχείων του υπολογιστή σας . Η java δίνει λύση στα προβλήματα αυτά παρέχοντας ένα «Αντιπυρικό τοίχος»(firewall) .το οποίο τίθεται μεταξύ της δικτυωμένης εφαρμογής και του υπολογιστή σας.

Όταν χρησιμοποιείτε έναν web browser (πρόγραμμα περιήγησης ή ανάγνωσης σελίδων) συμβατό με java , είναι εφικτό να κατεβάσετε με ασφάλεια τις mini εφαρμογές (applets) της java χωρίς να έχετε τον φόβο μόλυνσης από ιούς . Ο τρόπος με τον οποίον η java επιτυγχάνει αυτή την ασφάλεια , οφείλετε σε ένα

πρόγραμμα της java προς το περιβάλλον εκτέλεσης της java το οποίο δεν επιτρέπει την πρόσβαση σε άλλα μέρη του υπολογιστή σας . Για την ακρίβεια , η δυνατότητα φόρτωσης (κατεβάσματος των applets με πλήρη εμπιστοσύνη ,ώστε να μην υπάρχει ενδεχόμενο καταστροφής ενός υπολογιστή client (πελάτη), σίγουρα αποτελεί το ποιο σημαντικό χαρακτηριστικό της Java.

5.3.6 ΟΥΔΕΤΕΡΗ ΤΗΣ ΥΠΟΚΕΙΜΕΝΗΣ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ

Η Java έχει σχεδιαστεί για να υποστηρίζει δικτυακές εφαρμογές. Ένα δίκτυο, όμως, αποτελείται από ποικιλία διαφορετικών συστημάτων, με διαφορετικές CPU και λειτουργικά συστήματα. Για να μπορούν οι Java εφαρμογές να εκτελούνται παντού στο δίκτυο, το πρόγραμμα Java πρέπει να περάσει από δύο διαδικασίες ώστε να καταλήξει σε εκτελέσιμη μορφή. Πρώτα ο μεταγλωττιστής, μετατρέπει τον πηγαίο κώδικα του προγράμματος σε μία ενδιάμεση γλώσσα που καλείται *Java bytecodes*. Τα Java bytecodes είναι ανεξάρτητα της πλατφόρμας και με χρήση του ερμηνευτή *(interpreter)* κάθε bytecode εντολή μετατρέπεται σε κατάλληλη δυαδική μορφή για να τρέξει στον εκάστοτε υπολογιστή. Η μεταγλώττιση *(compilation)* συμβαίνει μόνο μια φορά για κάθε Java πρόγραμμα, η ερμηνεία *(interpretation)* γίνεται κάθε φορά που το πρόγραμμα εκτελείται. Το παρακάτω σχήμα επιδεικνύει πως λειτουργεί αυτή η φιλοσοφία.

Τα Java bytecodes μπορούμε να τα φανταστούμε σαν την γλώσσα μηχανής για την *Java Virtual Machine (JVM)*. Κάθε Java ερμηνευτής (π.χ. ένας Web browser που μπορεί να τρέχει applets) είναι μια λογισμική εφαρμογή του της Java Virtual Machine. Η JVM αναλαμβάνει να μετατρέψει τα bytecodes σε κατάλληλη εκτελέσιμη μορφή, ανάλογα με το υποκείμενο software και hardware.

Η τεχνική που περιγράφηκε παραπάνω καλείται *"write once, run anywhere"*. Το Java πρόγραμμα μεταγλωττίζεται μία φορά σε Java bytecodes με τον μεταγλωττιστή της Java. Έπειτα, τα bytecodes μπορούν να τρέξουν σε οποιαδήποτε μηχανή που έχει μία εφαρμοσμένη JVM (ο ερμηνευτής).

5.3.7 ΦΟΡΗΤΗ

Το γεγονός ότι είναι ανεξάρτητη της υποκείμενης πλατφόρμας αποτελεί μεγάλο μέρος του ότι είναι φορητή, άλλα υπάρχουν και άλλα σημεία που χαρακτηρίζουν την φορητότητα της.

Σε αντίθεση με την C/C++ δεν υπάρχουν καθόλου χαρακτηριστικά που εξαρτούνται από την CPU του υπολογιστή. Έτσι, τα μεγέθη των πρωταρχικών τύπων δεδομένων είναι καθορισμένα και η συμπεριφορά τους είναι παντού η ίδια. Για παράδειγμα, *"int"* σημαίνει πάντα έναν 32 bit ακέραιο και *"float"* πάντα αντιπροσωπεύει έναν 32 bit floating αριθμό.

5.3.8 INTERPRETED

Τα Java bytecodes μεταφράζονται σε πραγματικό χρόνο σε εντολές μηχανής που εξαρτώνται από την εκάστοτε πλατφόρμα, και δεν αποθηκεύονται πουθενά. Η διαδικασία είναι γρήγορη και πιο αποτελεσματική. Μαζί με τα bytecodes μεταφέρονται πληροφορίες που μπορούν να χρησιμοποιηθούν κατά την εκτέλεση και παρέχουν την βάση για τους ελέγχους που πραγματοποιεί ο συνδέτης *(linker)*. Επίσης τα προγράμματα γίνονται πιο επιδεκτικά σε debugging διαδικασίες.

5.3.9 ΥΨΗΛΗΣ ΑΠΟΔΟΣΗΣ

Η διαδικασία παραγωγής των εντολών μηχανής είναι απλή και γρήγορη. Ο κώδικας που προκύπτει είναι αποτελεσματικός. Ο μεταγλωττιστής από την μεριά του εφαρμόζει αυτόματη κατανομή των καταχωρητών *(automatic register allocation)* όταν παράγει τα bytecodes. Η τελική μορφή του κώδικα (εκτελέσιμη δυαδική μορφή) είναι μικρή σε μέγεθος και ταχύτατη στην εκτέλεση.

5.3.10 MULTITHREADED

Τα προγράμματα σε Java έχουν την δυνατότητα να αντιμετωπίζουν πολλές καταστάσεις διαδικασίες ταυτόχρονα. Σε αντίθεση, η C και C++ είναι *single-threaded* γλώσσες. Τα πλεονεκτήματα του *multithreading* είναι η καλύτερη πραγματικού χρόνου συμπεριφορά και η καλύτερη αλληλεπιδραστική ανταπόκριση.

5.3.11 ΔΥΝΑΜΙΚΗ

Η Java είναι πιο δυναμική γλώσσα από την C ή C++. Έχει αναπτυχθεί για να προσαρμοστεί σε ένα εξελισσόμενο περιβάλλον. Οι βιβλιοθήκες εργαλείων αναπτύσσονται ελεύθερα με την πρόσθεση νέων μεθόδων και μεταβλητών, χωρίς να επηρεάζονται οι ήδη υπάρχουσες εφαρμογές.

5.4 APPLETS ΚΑΙ APPLICATIONS

Η java μπορεί να χρησιμοποιηθεί για τη δημιουργία δυο τύπων προγραμμάτων:

Εφαρμογές (application) και mini εφαρμογές (applet).Μία εφαρμογή (application) είναι ένα πρόγραμμα το οποίο «τρέχει» πάνω στον υπολογιστή σας κάτω από το λειτουργικό σύστημα που υπάρχει στον υπολογιστή σας .Μια εφαρμογή ,η οποία δημιουργείται στην java είναι λιγο πολύ μια συνηθισμένη εφαρμογή, την οποία μπορείτε να δημιουργήσετε χρησιμοποιώντας οποιονδήποτε άλλο τύπο γλώσσας προγραμματισμού ,όπως visual basic ή C++. Όταν η java χρησιμοποιείται για τη δημιουργία εφαρμογών δεν διαφέρει και τόσο πολύ από τις άλλες γλώσσες προγραμματισμού . Αυτό όμως που την κάνει να ξεχωρίζει από τις άλλες γλώσσες προγραμματισμού είναι ότι η java έχει τη δυνατότητα να δημιουργεί τις λεγόμενες εφαρμογές applets .Μία applet είναι απλός μια εφαρμογή , σχεδιασμένη για να μεταδίδεται μέσω του Internet και να εκτελείται από ένα web browser που να είναι συμβατός με java . αν και για τη δημιουργια μιας mini εφαρμογής (applet) . Αυτό οφείλεται στο γεγονός πως ότι η java δίνει λύσεις σε δύο πολύ σημαντικά προβλήματα που έχουν σχέση με τις applets : θέματα ασφάλειας και θέματα μεταφοράς.

5.5 ΕΡΓΑΛΕΙΑ ΤΗΣ JAVA

Ακολούθως παρουσιάζονται εν συντομία όλα τα εργαλεία που έρχονται με το Java

Java 2 Platform Standard Edition (J2SE), της Sun microsystems. Η παρούσα έκδοση της

Java και του J2SE είναι η Java 2 έκδοση 5.0.

• **javac** Είναι ο compiler της Java. Η χρήση του στο command-line είναι : *javac*

<όνομα αρχείου>. Εδώ να σημειώσουμε ότι το javac δεν παράγει ένα αρχείο με όλον τον κώδικα, αλλά χωριστό αρχείο για κάθε κλάση. Τα αρχεία των κλάσεων ονομάζονται :*<όνομα κλάσης>.class.*

• **java** Είναι ο interpreter της Java. Η χρήση του είναι η εξής : *java <κλάση>, πχ java myClass και όχι java myClass.class*.

• **javaw** (MONO στα Windows 95/NT) Είναι παρόμοιο με το java με μόνη την διαφορά ότι δεν χρειάζεται shell για να τρέξει.

• **jdb** Είναι ο Java debuger.

• **javah** Κατασκευάζει C files και stub files για κάποια κλάση. Αυτά τα αρχεία είναι απαραίτητα όταν θέλουμε να υλοποιήσουμε κάποιες από τις μεθόδους της κλάσης σε C,πράγμα πολύ σπάνιο.

• **javap** Είναι ο Java disassembler.

• **javadoc** Είναι ένα πρόγραμμα για αυτόματη κατασκευή documentation. Είναι αρκετά χρήσιμο στην κατασκευή βοηθημάτων και τεχνικών αναφορών για εφαρμογές οποιουδήποτε μεγέθους.

• **appletviewer** Είναι ένα πρόγραμμα το οποίο μας επιτρέπει να τρέχουμε και να χρησιμοποιούμε τα διάφορα applets σε Java. Οι stand-alone εφαρμογές, ωστόσο, δεν τρέχουν στον appletviewer αλλά κατευθείαν στον java ή javaw.

5.6 ΣΥΜΒΟΛΗ ΤΗΣ JAVA ΣΤΟ INTERNET

Το Internet συνέβαλε στην προβολή της java στο προσκήνιο του προγραμματισμού και η java με τη σειρά της είχε εξασκήσει μια πολύ σημαντική επίδραση στο Internet. Ο λόγος είναι πολύ απλός : η java διευρύνει όλα τα αντικείμενα τα οποία μπορούν να κυκλοφορούν όλα ελεύθερα στον κυβερνοχώρο .Σε ένα δίκτυο ,υπάρχουν δύο μεγάλες κατηγορίες αντικειμένων που μεταδίδονται μεταξύ ενός server και του προσωπικού σας υπολογιστή: Παθητικές πληροφορίες και δυναμικά ,ενεργά προγράμματα .Για παράδειγμα , όταν διαβάζετε την ηλεκτρονική σας αλληλογραφία (email) απλώς βλέπετε παθητικά δεδομένα μέχρι στιγμή που θα εκτελέσετε . Ωστόσο , ένας δεύτερος τύπος αντικειμένου μπορεί να μεταδοθεί προς τον υπολογιστή σας . Πρόκειται για ένα δυναμικό και αυτοεκτελούμενο πρόγραμμα . Ένα τέτοιο πρόγραμμα μπορεί να θεωρηθεί σαν έναν ενεργό πράκτορα του υπολογιστή client (πελάτη) , ακόμα και αν η ενεργοποίηση και η εκκίνηση του γίνεται από τον server . Για παράδειγμα ο server θα μπορούσε να σας δώσει ένα πρόγραμμα προκειμένου να εμφανίζει καθαρά τα δεδομένα τα οποία σας αποστέλλει .

Όσο ποιο πολλή ζήτηση υπάρχει για δυναμικά και δικτυωμένα προγράμματα , άλλο τόσο είναι και τα σοβαρά προβλήματα που προκύπτουν σε θέματα ασφαλείας και δυνατότητα μεταφοράς .Πριν από τη Java , ο κυβερνοχώρος ήταν κλειστός για περίπου τα μισά από αυτά τα αντικείμενα , που τώρα πλέον « κυκλοφορούν και ζουν » μέσα στο internet . Όπως θα δείτε σύντομα στο βιβλίο αυτό , η java έχει δώσει λύσεις σε πολλά προβλήματα παρέχοντας μια νέα μορφή προγράμματος : τις μίνι εφαρμογές , οι οποίες ανά τον κόσμο ονομάζονται applets.

Σε αυτό το κεφάλαιο αναφερθήκαμε σε όσα στοιχεία για την java θεωρήθηκαν σημαντικά στο επόμενο κεφάλαιο θα αναφερθούμε σε ένα παρακλάδι της Java την java swing μέσω της οποίας δίνετε η δυνατότητα στον προγραμματιστή να δημιουργήσει εύκολα και γρήγορα interfaces .

6 JAVA SWING

Σε αυτό το κεφάλαιο ουσιαστικά επεκτείνουμε την λειτουργικότητα της java στην java swing περιγράφουμε ,τι ακριβώς είναι σαν γλώσσα, τι επιπλέων εφαρμογές έχει , τις τροποποιήσεις της μέσα στον χρόνο , μαζί με όλα αυτά αναπτύσσουμε αρκετά την αρχιτεκτονική της καθώς και τις σχέσεις εξαρτίσεις της με την AWT.

6.1 ΤΙ ΕΙΝΑΙ Η JAVA SWING;

Swing είναι το κύριο GUI widget toolkit της java . Είναι μέρος της Oracle Java Foundation Classes (JFC) - ένα API για να παρέχει μια γραφική διεπαφή στον χρήστη (GUI) για τα προγράμματα Java.

Η Swing αναπτύχθηκε για να παρέχει ένα πιο εξελιγμένο σύνολο GUI στοιχείων από την προηγούμενη Abstract Window Toolkit (AWT) .Η Swing παρέχει μια φυσική εμφάνιση και αίσθηση που μιμείται την εμφάνιση και την αίσθηση των διαφόρων πλατφορμών, και υποστηρίζει επίσης pluggable εμφάνιση και αίσθηση που επιτρέπει στις εφαρμογές να έχουν μια εμφάνιση και αίσθηση που δεν σχετίζονται με την υποκείμενη πλατφόρμα. Έχει πιο ισχυρά και ευέλικτα εξαρτήματα από AWT. Εκτός από γνωστά στοιχεία, όπως κουμπιά, πλαίσια ελέγχου και ετικέτες, Swing παρέχει πολλά προηγμένα συστατικά όπως καρτέλες πάνελ δέντρων, πίνακες και λίστες.

Σε αντίθεση με συστατικά AWT, που δεν είναι συστατικά Swing υλοποιείται από την πλατφόρμα- με ειδικό κωδικό. Αντ 'αυτού, είναι γραμμένο εξ ολοκλήρου σε Java και ως εκ τούτου είναι ανεξάρτητα από την πλατφόρμα. Ο όρος «ελαφρύ» χρησιμοποιείται για να περιγράψει ένα τέτοιο στοιχείο.

6.2 ΙΣΤΟΡΙΑ

Το Internet Foundation Classes (IFC) ήταν μια βιβλιοθήκη γραφικών για Java που αναπτύχθηκε αρχικά από την Netscape Communications Corporation και κυκλοφόρησε για πρώτη φορά στις 16 Δεκεμβρίου 1996. Στις 2 Απριλίου 1997, η Sun Microsystems και η Netscape Communications Corporation ανακοίνωσαν την πρόθεσή τους να ενσωματώσουν IFC με άλλες τεχνολογίες για να σχηματίσουν τις Java Foundation Classes . Η "Java Foundation Classes" αργότερα μετονομάστηκε σε "Swing".

Η Swing εισήγαγε ένα μηχανισμό που επέτρεψε την εμφάνιση και την αίσθηση του κάθε στοιχείου σε μια εφαρμογή να τροποποιηθεί χωρίς να χρειάζεται να γίνουν ουσιαστικές αλλαγές στον κώδικα εφαρμογής. Στη java swing μπορεί ένα στοιχείο να εισέρθει και να υποστηρίξει τον κώδικα και να επιτρέπει σε Swing συστατικά να μιμηθούν την εμφάνιση με τα φυσικά συστατικά, διατηρώντας τα πλεονεκτήματα της ανεξάρτητης πλατφόρμας. Αρχικά διανέμονται ως ξεχωριστά στοιχεία σε μία downloadable βιβλιοθήκη, η Swing έχει συμπεριληφθεί ως μέρος της Java Standard Edition από την έκδοση 1.2. Οι κατηγορίες Swing και τα συστατικά της περιέχονται στο javax.swing πακέτο ιεραρχίας.

6.3 ΑΡΧΙΤΕΚΤΟΝΙΚΗ

Swing είναι μια πλατφόρμα-ανεξάρτητη, *Model-View-Controller* GUI πλαισίου για την Java, η οποία ακολουθεί ένα single- threaded μοντέλο προγραμματισμού. Επιπλέον, το πλαίσιο αυτό παρέχει μια διεπαφή μεταξύ της δομής του κώδικα και γραφική παρουσίαση ενός Swing-based GUI.

Η Swing είναι ανεξάρτητη από την πλατφόρμα, επειδή είναι εξολοκλήρου γραμμένη σε Java. Πλήρης τεκμηρίωση για όλες τις κατηγορίες Swing μπορεί να βρεθεί στο Java API Guide .

6.3.1 EXTENSIBLE

Η Swing είναι μία εξαιρετική αρθρωτή αρχιτεκτονική με βάση, την οποία επιτρέπετε το "plugging" των διαφόρων custom implementations συγκεκριμένων διεπαφών πλαισίου. Οι χρήστες μπορούν να παρέχουν τη δική τους προσαρμοσμένη εφαρμογή με αυτά τα συστατικά και να παρακάμψουν τις εφαρμογές προεπιλογής χρησιμοποιώντας τον μηχανισμό κληρονομικότητας της Java.

Η Swing είναι ένα στοιχείο που βασίζεται στο πλαίσιο του οποίου τα στοιχεία είναι όλα προερχόμενα από την τάξη javax.swing.JComponent. Τα Αντικείμενα της Swing είναι ασύγχρονα γεγονότα , συνδέονται με ιδιότητες, και δίνουν απάντηση σε ένα τεκμηριωμένο σύνολο από μεθόδους ειδικά για το στοιχείο.

Τα Swing συστατικά είναι Java Beans συστατικά, συμβατά με τις προδιαγραφές του Java Beans Component Architecture.

6.3.2 ΠΡΟΣΑΡΜΟΣΤΙΚΟΤΗΤΑ

Λαμβάνοντας υπόψη το προγραμματιστικό μοντέλο απόδοσης του πλαισίου της Swing, το κόστος ελέγχου πέρα από τις λεπτομέρειες της παροχής ενός στοιχείου είναι δυνατό. Ως ένα γενικό πρότυπο, η οπτική αναπαράσταση ενός συστατικού Swing είναι μια σύνθεση από ένα τυποποιημένο σύνολο στοιχείων, όπως ένα περίγραμμα, ένθετο, διακοσμήσεις, και άλλες ιδιότητες. Εύκολα , οι χρήστες μπορούν να προσαρμόσουν μέσω του προγραμματισμού σε κάθε component Swing (όπως JTable) αναθέτοντας συγκεκριμένες σύνορα, τα χρώματα, τα υπόβαθρα, σκιάσεις, κλπ. Το βασικό στοιχείο που θα χρησιμοποιήσει στη συνέχεια είναι αυτές οι ιδιότητες για να κάνει τη αναπαράσταση . Ωστόσο, είναι επίσης απολύτως δυνατό να δημιουργήσει μοναδικά GUI ελέγχου με άκρως προσαρμοσμένη οπτική αναπαράσταση.

6.3.3 ΔΙΑΜΟΡΦΩΣΙΜΗ

Η Μεγάλη εξάρτηση της Java Swing με runtime μηχανισμούς και έμμεσες μορφές σύνθεσης που της επιτρέπει να ανταποκρίνεται κατά το χρόνο εκτέλεσης σε θεμελιώδεις αλλαγές στις ρυθμίσεις της. Για παράδειγμα, μια Swing-based εφαρμογή είναι ικανή για hot swapping της διεπαφής χρήστη κατά τη διάρκεια του χρόνου εκτέλεσης . Επιπλέον, οι χρήστες μπορούν να παρέχουν τη δική τους εμφάνιση και την αίσθηση της εφαρμογής, η οποία επιτρέπει την ομοιόμορφη αλλαγή στην εμφάνιση και την αίσθηση των υφιστάμενων εφαρμογών των Swing χωρίς καμία προγραμματική αλλαγή στον κώδικα της εφαρμογής.

6.3.4 LIGHTWEIGHT UI

Το υψηλό επίπεδο της Swing στης ευελιξία αντικατοπτρίζεται από την εγγενή ικανότητά της να υπερισχύσει της μητρικής υποδοχής δηλαδή του λειτουργικού συστήματος (OS) με τους ελέγχους GUI για να παρουσιάζει τις δικές της ρυθμίσεις . Τα Swing χαρακτηριστικά των ελέγχων χρησιμοποιούν το Java 2D APIs, αντί να ζητούν ένα εγγενές κουτί εργαλείων διεπαφής χρήστη. Έτσι, ένα συστατικό Swing δεν έχει αντίστοιχη μητρική συνιστώσα OS GUI, και είναι

ελεύθερο να καταστήσει με κάθε δυνατό τρόπο με τα υποκείμενα γραφικά του GUIs.

Ωστόσο, στον πυρήνα του, κάθε στοιχείο της Swing βασίζεται σε ένα AWT container, δεδομένου ότι η (Swing) JComponent extends (AWT) Container. Αυτό επιτρέπει στη Swing να συνδεθεί σε GUI framework διαχείρισης της υποδοχής του OS, συμπεριλαμβανομένων των κρίσιμων device/screen αντιστοιχίσεων και αλληλεπιδράσεων των χρηστών, όπως πατήματα των πλήκτρων ή κινήσεις του ποντικιού. Η Swing απλά να "μεταφέρει" τα δικά της στοιχεία πάνω από τα υποκείμενα (OS) συστατικά. Έτσι, για παράδειγμα, κάθε στοιχείο Swing ζωγραφίζει στη γραφική συσκευή σε απάντηση σε μια κλήση για component.paint (), η οποία ορίζεται στο (AWT) container . Αλλά σε αντίθεση με AWT συστατικά, τα οποία ανέθεσαν στη ζωγραφική στην OS-μητρική,τα Swing συστατικά είναι υπεύθυνα για τη δική τους απόδοση.

Η μεταφορά και η αποσύνδεση δεν είναι μόνο οπτική, και επεκτείνεται με τη διαχείριση και την εφαρμογή της Swing για το δικό της ανεξάρτητο από λειτουργικό σύστημα τρόπο η σημασιολογίας της για γεγονότα που λειτουργούν μέσα σε ιεραρχίες στοιχείων του περιορισμού της. Σε γενικές γραμμές, τα μέρη που αποτελούν την αρχιτεκτονική της Swing το έργο της χαρτογράφησης των διαφόρων εκδοχών για την OS σημασιολογία του GUI σε ένα απλό, αλλά γενικευμένο, πρότυπο για το AWT container. Με βάση την εν λόγω γενικευμένη πλατφόρμα, που θεσπίζει τη δική της πλούσια και πολύπλοκη GUI σημασιολογία της με τη μορφή του JComponent μοντέλου.

6.4 ΧΑΛΑΡΑ ΣΥΝΔΕΔΕΜΕΝΕΣ ΚΑΙ MVC

Η βιβλιοθήκη της Swing κάνει βαριά χρήση του Model / View / Controller λογισμικού προτύπου σχεδιασμού , η οποία αποσυνδέει εννοιολογικά τα δεδομένα που προβάλλονται από τη διεπαφή χρήστη και ελέγχει μέσω των οποίων προβάλλεται. Εξαιτίας αυτού, τα περισσότερα Swing συστατικά έχουν συνδέσει *μοντέλα* (τα οποία καθορίζονται από τα χαρακτηριστικά του Java interfaces), και οι προγραμματιστές μπορούν να τα χρησιμοποιήσουν σε διάφορες εφαρμογές προεπιλογής ή να παρέχουν τα δικά τους. Το πλαίσιο αυτό παρέχει εφαρμογές προεπιλογής των διεπαφών προτύπων για όλα τα συγκεκριμένα συστατικά τους. Η τυπική χρήση του πλαισίου Swing δεν απαιτεί

τη δημιουργία custom μοντέλων, όπως το πλαίσιο που παρέχει μια σειρά από εφαρμογές προεπιλογής που είναι διαφανής, από προεπιλογή, που συνδέονται με την αντίστοιχη JComponent κλάση-παιδί στη βιβλιοθήκη Swing. Σε γενικές γραμμές, μόνο σύνθετα συστατικά, όπως πίνακες, τα δέντρα και μερικές φορές τους καταλόγους, μπορούν να απαιτούν από τις προσαρμοσμένες εφαρμογές μοντέλων γύρω από την εφαρμογή ειδικών δομών δεδομένων. Για να πάρετε μια καλή αίσθηση των δυνατοτήτων που η αρχιτεκτονική της Swing καθιστά δυνατό να εξεταστούν οι υποθετικές καταστάσεις όπου προσαρμοσμένα μοντέλα οι πίνακες και οι λίστες ή wrappers πάνω από DAO ή / και EJB υπηρεσίες.

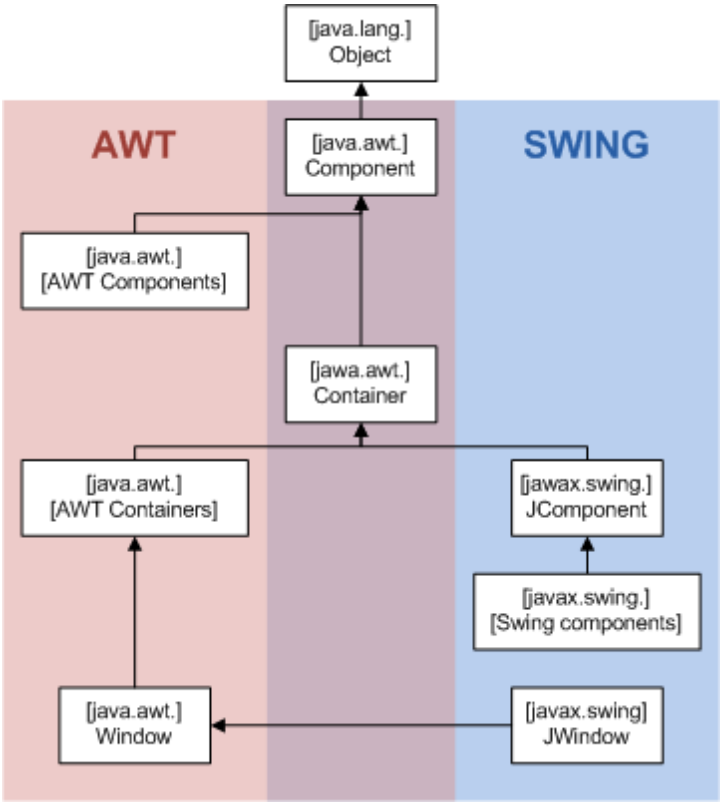
Συνήθως, τα αντικείμενα του μοντέλου Swing είναι υπεύθυνα για την παροχή του συνοπτικού interface που ορίζει τα γεγονότα που λειτουργούν, και προσιτές ιδιότητες για το (εννοιολογικό) μοντέλο δεδομένων για χρήση από τα συνδεδεμένα JComponent. Δεδομένου ότι το συνολικό σχέδιο MVC είναι ένα χαλαρά συνδεδεμένο και συνεργατικό πρότυπο σχέσης αντικειμένου, το μοντέλο προβλέπει τα προγραμματιστικά του μέσα για την προσάρτηση του στο μοντέλο αντικειμένων δεδομένων. Συνήθως, αυτά τα γεγονότα είναι model centric και αντιστοιχίζονται από την JComponent εξειδίκευση σε ένα σημαντικό event για το GUI component.

Για παράδειγμα, η JTable έχει ένα μοντέλο που ονομάζεται TableModel που περιγράφει μια διασύνδεση για το πώς ένας πίνακας θα έχει πρόσβαση στα δεδομένων των πινάκων. Μια προεπιλεγμένη εφαρμογή του παρόντος λειτουργεί σε ένα δισδιάστατο πίνακα .

Ένα component εμφάνισης ενός Swing JComponent είναι το αντικείμενο που χρησιμοποιείται για τη γραφική αναπαράσταση του εννοιολογικού ελέγχου GUI. Μια διάκριση της Swing, ως πλαίσιο GUI, είναι ότι στην εξάρτησή του προγραμματισμού καθιστά τους GUI ελέγχους (σε αντίθεση με τη χρήση των ελέγχων GUI της υποδοχής του λειτουργικού συστήματος). Πριν από την Java 6 Update 10 , η διάκριση αυτή ήταν μια πηγή των επιπλοκών που προέκυπταν κατά τη μίξη των ελέγχων AWT, τα οποία χρησιμοποιούν τα native controls, με τους ελέγχους Swing σε ένα GUI.

Τέλος, όσον αφορά την οπτική σύνθεση και τη διαχείριση, η Swing ευνοεί σχετικές διατάξεις (οι οποίες καθορίζουν τις σχετικές θέσεις μεταξύ των συστατικών), σε αντίθεση με απόλυτα σχεδιαγράμματα (τα οποία καθορίζουν την ακριβή θέση και το μέγεθος των συστατικών). Αυτή η προκατάληψη προς τη «ρευστή» «οπτική τακτοποίηση οφείλεται στις ρίζες της στη applet operating environment που πλαισιώνεται από το σχεδιασμό και την ανάπτυξη του αρχικού Java GUI toolkit. (Θεωρητικά, αυτή η άποψη της διαχείρισης διάταξης είναι αρκετά παρόμοια με αυτή που ενημερώνει την απόδοση του περιεχομένου HTML σε προγράμματα περιήγησης .)

6.5 ΣΧΕΣΗ ΜΕ AWT



Εικόνα 1 AWT και Swing ιεραρχία κλάσεων

Από τις αρχές των εκδόσεων της Java, ένα τμήμα του Abstract Window Toolkit (AWT) έχει εξελιχτεί ανεξάρτητα από την πλατφόρμα APIs για τα στοιχεία της διεπαφής του χρήστη. Σε AWT, κάθε συστατικό καθίσταται και ελέγχεται από ένα φυσικό συστατικό των ειδικών για το υποκείμενο σύστημα παραθύρων.

Αντίθετα, τα συστατικά της Swing συχνά περιγράφεται ως *ελαφρύ,* επειδή δεν απαιτούν την κατανομή των φυσικών πόρων στην εργαλειοθήκη παραθύρων του λειτουργικού συστήματος. Τα AWT συστατικά αναφέρονται ως *βαρέων στοιχείων.*

Μεγάλο μέρος της API Swing είναι γενικά μια συμπληρωματική επέκταση του AWT παρά μια άμεση αντικατάσταση. Στην πραγματικότητα, κάθε Swing ελαφρύ interface υπάρχει τελικά μέσα σε ένα βαρέων βαρών AWT συστατικό επειδή όλα τα top-level συστατικά Swing (JApplet , JDialog , JFrame και JWindow) επεκτείνουν μια AWT top-level container. Πριν από την Java 6 Update 10 , η χρήση των διαφόρων συστατικών μέσα στο ίδιο παράθυρο ήταν γενικά αποθαρρύντική λόγω στις Z-order ασυμβατότητες. Ωστόσο, στις νεότερες εκδόσεις της Java έχουν καθοριστεί τα θέματα αυτά, και τα δύο συστατικά Swing και AWT μπορούν τώρα να χρησιμοποιηθούν σε ένα GUI χωρίς Z-order θέματα.

Αφού πλέων έχουμε αναπτύξει αρκετά ότι έχει να κάνει με τη java παρακάτω θα αναφερθούμε στον τρόπο που μπορούμε να συνδέσουμε δύο γλώσσες προγραμματισμού και συγκεκριμένα την java και την prolog ώστε να μπορούν να συνυπάρχουν σε ένα project.

7 ΔΙΑΣΥΝΔΕΣΗ JAVA ΜΕ PROLOG

Σε αυτό το κεφάλαιο αναφέρουμε τι είναι η JPL βιβλιοθήκη πως λειτουργεί ,πως γίνετε η προετοιμασία της prolog για να μπορέσει να κλιθεί μέσω της jpl φυσικά αναφερόμαστε και στις εξαιρέσεις που μπορεί να προκύψουν και πως διαχειρίζονται, μετά δείχνουμε πως εκτελείτε το debugging . γίνετε επίσης διαχωρισμός του Low-level interface από το high level interface και για το κάθε ένα αναφέρουμε τους υποστηριζόμενους τύπους την ιεραρχία των κλάσεων τους υποστηριζόμενους τύπους δεδομένων και την jpl.fli.prolog .

7.1 ΤΙ ΕΙΝΑΙ Η ΒΙΒΛΙΟΘΗΚΗ JPL ΚΑΙ ΠΩΣ ΛΕΙΤΟΥΡΓΕΙ;

JPL είναι ένα σύνολο από κλάσεις Java και C, λειτουργεί παρέχοντας μια διεπαφή μεταξύ Java και Prolog. JPL χρησιμοποιεί το Java Native Interface (JNI) για να συνδεθείτε σε μια μηχανή Prolog μέσω της Prolog Εξωτερικών Interface Language (FLI), η οποία είναι περισσότερο ή λιγότερο στην διαδικασία του να τυποποιηθεί σε διάφορες εφαρμογές της Prolog. JPL δεν είναι μια

καθαρή εφαρμογή Java της Prolog.Κάνει εκτεταμένη χρήση των ιθαγενών εφαρμογές της Prolog στις υποστηριζόμενες πλατφόρμες. Η τρέχουσα έκδοση του JPL λειτουργεί μόνο με SWI-Prolog.

Επί του παρόντος, JPL υποστηρίζει μόνο την ενσωμάτωση μιας μηχανής Prolog εντός της Java VM. Μελλοντικές εκδόσεις μπορεί να υποστηρίξουν την ενσωμάτωση της Java VM σε Prolog, έτσι ώστε, για παράδειγμα, θα μπορούσε κανείς να επωφεληθεί από την πλούσια ταξική δομή του περιβάλλοντος Java μέσα από Prolog.

Το JPL είναι σχεδιασμένο σε δύο στρώματα, το χαμηλό επίπεδο διασύνδεσης με την Prolog FLI και μια υψηλού επιπέδου διεπαφή Java για τον προγραμματιστή της Java που δεν ασχολείται με τις λεπτομέρειες της Prolog FLI. Η χαμηλού επιπέδου διεπαφή παρέχεται για C προγραμματιστές που επιθυμούν να μεταφέρουν C υλοποιήσεις τους που χρησιμοποιούν το FLI σε Java με ελάχιστο κόπο.

7.2 ΠΡΟΕΤΟΙΜΑΣΊΑ PROLOG

Η **jpl.JPL** κλάση προετοιμάζει την Prolog VM (π.χ. libpl.dll σε Win32), εάν είναι αναγκαίο, όταν η πρώτη **ερώτηση** είναι ενεργοποιημένη, με τις τιμές των παραμέτρων με προεπιλογή. Πριν από την εκκίνηση λαμβάνουν χώρα, οι προεπιλεγμένες τιμές που μπορεί να διαβαστούν, και να μεταβληθούν.

public String [] getDefaultInitArgs ()?
public void setDefaultInitArgs (String [] args)?

Μετά την προετοιμασία, οι τιμές των παραμέτρων που χρησιμοποιήθηκαν πραγματικά ,μπορεί να διαβαστούν.

public String [] getActualInitArgs ()?

(Αυτή η μέθοδος επιστρέφει null αν η αρχικοποίηση δεν έχει συμβεί, και ως εκ τούτου μπορεί να χρησιμοποιηθεί ως μια δοκιμή.) Αυτό επιτρέπει στις Java κλάσεις βιβλιοθηκών να απασχολούν ***JPL*** χωρίς τοποθέτηση οποιουδήποτε βάρους κατά την προετοιμασία από τις εφαρμογές που τα χρησιμοποιούν. Μπορεί επίσης να εξασφαλίσει ότι το VM Prolog ξεκινά μόνο όταν και εφόσον αυτό είναι αναγκαίο.

Ρητή αρχικοποίηση υποστηρίζεται όπως στο ***JPL 1.0.1:***

Public void init()?
  public void init (String args [])?

Ο Κώδικας Java που απαιτεί ένα VM Prolog να προετοιμαστεί με ένα συγκεκριμένο τρόπο μπορεί και να ελέγξει κατά πόσον η προετοιμασία έχει ήδη συμβεί: αν όχι, μπορείτε να καθορίσετε τις παραμέτρους και να υλοποιηθεί έτσι Αν ναι, μπορεί να ανακτήσει και να ελέγξετε τις παραμέτρους εκκίνησης που πράγματι χρησιμοποιήθηκαν, για να καθοριστεί αν η προετοιμασία τους ανταποκρίνεται στις απαιτήσεις του. Αυτή η έκδοση του ***JPL*** δεν υποστηρίζει Επαναρχικοποίηση σε ένα VM Prolog.

Για λεπτομέρειες σχετικά με τις νομικές τιμές των παραμέτρων, μπορούμε να ελέγξουμε στην τοπική τεκμηρίωση της Prolog . Οι περισσότεροι χρήστες θα βασίζονται στην αυτόματη εκκίνηση.

7.3 ΑΠΑΙΤΗΣΕΙΣ

JPL απαιτεί τώρα SWI-Prolog έκδοση 3.1.0 ή νεότερη έκδοση, η οποία είναι διαθέσιμη στην ακόλουθη διεύθυνση URL:

http://www.swi-prolog.org/

SWI-Prolog πληροφορίες άδειας χρήσης είναι διαθέσιμο εδώ:

http://www.swi-prolog.org/license.html

Θα χρειαστείτε επίσης ένα περιβάλλον ανάπτυξης Java. Η Java ιστοσελίδα της Sun είναι ένα καλό μέρος για να ξεκινήσετε:

http://java.sun.com/

JPL 2.0.2 αναπτύχθηκε και δοκιμάστηκε σε Windows NT4, και δεν έχει ακόμη συνταχθεί σε οποιαδήποτε μη πλατφόρμα των Windows JPL1.0.1 γράφτηκε και δοκιμάστηκε σε πυρήνα Linux 2.1.24. Θα πρέπει να

καταρτίζουν σε οποιοδήποτε άλλο σύστημα UNIX με μια πλήρη σουίτα εργαλείων του GNU.

7.4 ΕΠΑΛΗΘΕΥΣΗ ΤΗΣ ΕΓΚΑΤΑΣΤΑΣΗΣ

Για να επιβεβαιώσετε ότι το ***JPL*** και SWI-Prolog είναι ουσιαστικά σε θέση να εργαστούν από κοινού, ανοίξτε ένα παράθυρο κονσόλας και να πάει σε αυτόν τον κατάλογο:

JPL / examples /

Διαβάζει το αρχείο README.txt, και εκτελεί τα διάφορα παραδείγματα που περιγράφει.

Κάθε παράδειγμα Java θα τρέξει ως εφαρμογή (δηλαδή ο καθένας έχει μια μέθοδο main ()), ασκεί SWI-Prolog, και γράφει κάτι για να **System.out** ή **System.err.**

Αν δείτε κάποια εύλογη απόδοση, χωρίς σοβαρά μηνύματα λάθους, τότε όλα μπορεί να είναι καλά.

7.5 ΕΞΑΙΡΕΣΕΙΣ

Το πακέτο ***JPL*** παρέχει αρκετά αργό χειρισμό εξαιρέσεων . Η βασική κλάση για όλες τις εξαιρέσεις ***JPL*** είναι η **JPLException,** η οποία είναι ένα **java.lang.RuntimeException** (και ως εκ τούτου δεν χρειάζεται να δηλωθεί), και τα οποία θα ριχτούν στην απουσία κάθε άλλου είδους εξαίρεσης που μπορεί να πεταχτεί, συνήθως ως αποτέλεσμα κάποιου σφάλματος προγραμματισμού. Μετατρέποντας την εξαίρεση ενός **java.lang.String** πρέπει να παρέχει κάποιες περιγραφικές πληροφορίες σχετικά με την αιτία του σφάλματος. Όλες οι άλλες κατηγορίες excpetion ***JPL*** επεκτείνουν αυτή την κατηγορία. Επί του παρόντος, υπάρχουν δύο, η κατηγορία **QueryInProgressException** και η κατηγορία **PrologException.**

Μια **QueryInProgressException** ρίχνεται όταν ένα **ερώτημα** που έχει ανοίξει, ενώ ένα άλλο είναι σε εξέλιξη. Η εξαίρεση αυτή μπορεί να αλιευθεί σε multi-threaded καταστάσεις.

Μια **PrologException** ρίχνεται είτε κατά τη διάρκεια εκτέλεσης ενός Prolog ενσωματωμένου κατηγόρηματος ή με ρητή κλήση, με κώδικα εφαρμογής Prolog, το κατηγόρημα της Prolog *throw/1.*

Σήμερα δεν υπάρχει μέσο χάρη στο οποίο να υπάρχει αντιμετώπιση των εξαιρέσεων που προκαλούνται από ακατάλληλες παραμέτρους (π.χ., απροσδιόριστα κατηγορήματα) που περνούν από την διεπαφή υψηλού επιπέδου για τον κινητήρα της Prolog.

7.6 DEBUGGING

Κάθε τύπος **Term** (μαζί με το **ερώτημα** τάξης) υποστηρίζει την εφαρμογή της **toString ()** η οποία επιστρέφει μια περισσότερο ή λιγότερο εξοικειωμένη αναπαράσταση κειμένου της Prolog ,του **όρου** ή του **ερωτήματος.**

Μερικές φορές, όμως, η πληροφορία αυτή δεν είναι επαρκής, έτσι ώστε να έχουν παράσχει **debugString** μέθοδο **(),** η οποία παρέχει μια πιο αναλυτική και ρητή αναπαράσταση, συμπεριλαμβανομένων των τύπων (άτομο, ακέραιος κλπ) για κάθε όρο και subterm.

Σε γενικές γραμμές, οι **Term** και **Query** περιπτώσεις αντιπροσωπεύονται στη μορφή *(τύπος δεδομένων),* όπου ο *τύπος* είναι το όνομα του τύπου (π.χ., **άτομο, Ενωση, πλειάδα,** κλπ.), και *τα δεδομένα* είναι μια αναπαράσταση του περιεχομένου του **όρου.** Για παράδειγμα, Εάν ο **όρος** είναι ένα **άτομο,** τα δεδομένα είναι το **άτομο** . Τα επιχειρήματα των **ενώσεων που** αντιπροσωπεύονται από λίστες χωρισμένες με κόμματα μέσα σε αγκύλες ('[' ']').

Επισκόπηση της δομής ενός **όρου** ή **ερώτημα** μπορεί να είναι χρήσιμη για τον προσδιορισμό αν το σφάλμα έγκειται στην Prolog ή Java πλευρά των εφαρμογών JPL σας.

Ίσως ακόμη καλύτερα, **Term** εργαλεία (σε ένα βασικό, αλλά επαρκή τρόπο) η διασύνδεση **javax.swing.TreeModel,** και την **απεικόνιση** του **()** μέθοδος δημιουργεί ένα **JFrame** περιέχει browseable εκπροσώπηση **JTree** του όρου.

7.7 ΧΡΗΣΙΜΟΠΟΙΩΝΤΑΣ ΤΟ LOW-LEVEL INTERFACE

Το ***JPL*** στη χαμηλού επιπέδου διεπαφή υλοποιείται από το Java πακέτο **jpl.fli.** Αυτό το πακέτο περιέχει μια σειρά από μαθήματα που αντανακλούν τους τύπους δεδομένων της SWI-Prolog *Εξωτερικών Interface Language* (FLI), μαζί με μια τάξη **jpl.fli.Prolog,** το οποίο περιέχει στατικές μεταβλητές και στατική εναλλακτικών μεθόδων οι οποίες αντανακλούν τις σταθερές λειτουργίες και η FLI. Το πακέτο έχει σχεδιαστεί για να χρησιμεύσει ως μια άμεση μετάφραση της Prolog FLI και γενικά δεν προορίζεται για το μέσο χρήστη. Ο κύριος σκοπός του είναι να στηρίξει την υψηλού επιπέδου διεπαφή , η χρήση των οποίων είναι προτιμότερη στις περισσότερες εφαρμογές.

7.7.1 ΥΠΟΣΤΗΡΙΖΟΜΕΝΟΙ ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ

Η χαμηλού επιπέδου διεπαφή παρέχει ορισμούς και υποστήριξη για τις ακόλουθες κατηγορίες, οι οποίες είναι ουσιαστικά "holder" κλάσεων για τους αντίστοιχους τύπους δεδομένων στην FLI:

LongHolder

|

+ --- Term_t

|

+ - Atom_t

|

+ - Functor_t

|

+ - Qid_t

|

+ - Fid_t

PointerHolder

|

+ - Predicate_t

|

+ - Module_t

Με εξαίρεση τις **predicate_t** και **module_t,** αυτές οι τάξεις κατέχουν Java long (signed 64-bit) values, που αντιστοιχούν στους τύπους C στο FLI με το ίδιο

όνομα (unsigned long τιμές). Η **module_t** και **predicate_t** κλάσεις κατέχουν επίσης long values, αλλά οι τιμές τους είναι κατανοητές σε C pointers (void)

Η χαμηλού επιπέδου διεπαφή παρέχει επίσης τις ακόλουθες κατηγορίες που με ευκολία χρησιμοποιούνται για να πάρουν τις πληροφορίες πίσω στο JavaVM από Prolog:

IntHolder

LongHolder

DoubleHolder

StringHolder

PointerHolder

Αυτές οι κατηγορίες είναι για χρήση όπου η SWI-Prolog λειτουργία FLI παίρνει τροποποιήσεις (με αναφορά) παραμέτρων.

7.7.2 JPL.FLI.PROLOG

Η κλάση **jpl.fli.Prolog** περιέχει ένα σύνολο από Java (static final) and static native method declarations. Οι δηλώσεις αυτές περισσότερο ή λιγότερο είναι αντίστοιχες με εκείνα στα αρχεία κεφαλίδας για το FLI (σε SWI-Prolog, *SWI-Prolog.h),* και όλα μπορούν να βρεθούν στο *jpl_fli_Prolog.c* αρχείο προέλευσης της C.

Ο γενικός κανόνας είναι ως εξής:

Όλες οι σταθερές και οι μέθοδοι (με λίγες αξιοσημείωτες εξαιρέσεις) είναι τα ίδια όπως εκείνα της FLI, με την Prolog απομακρύνθηκε το ειδικό πρόθεμα της εφαρμογής (στην περίπτωση της SWI-Prolog, PL_) . Για παράδειγμα, PL_VARIABLE στο FLI γίνεται μόνο VARIABLE και η function PL_new_term_ref FLI () γίνεται μόνο new_term_ref (). Μια αξιοσημείωτη εξαίρεση είναι η throw FLI function, η οποία μετονομάζεται σε _**throw** στο FLI. Το throw είναι μια δεσμευμένη λέξη στη Java.

Όλες οι σταθερές τιμές είναι η ίδιες με τα χαμηλού επιπέδου interface που είναι στο FLI.

Οι υπογραφές του FLI λειτουργιών (με μερικές αξιοσημείωτες εξαιρέσεις) διατηρούνται σε χαμηλού επιπέδου interface. Η χαμηλού επιπέδου διεπαφή παρέχει τα παραπάνω είδη για το σκοπό αυτό.

Οι παράμετροι των πρωτόγονων τύπων Java (π.χ., **int, float, μακρύ,** κλπ.) διατηρούνται.

Τροποποίηση των παραμέτρων των πρωτόγονων τύπων Java σε κλάσεις Holder (π.χ., **IntHolder, DoubleHolder, LongHolder,** κλπ.), στην οποία είναι γραμμένες οι τιμές, αντί για τους δείκτες σε αυτούς τους τύπους.

Παράμετροι των άλλων τύπων που διαβάζονται και οδηγούν στην τροποποίηση των παραμέτρων του FLI ώστε να τις πάρουν οι δομές (π.χ., **jpl.fli.term_t)** ως επιχειρήματα στην χαμηλού επιπέδου interface. Αυτό διατηρεί την υπογραφή των μεθόδων αυτών όσο το δυνατόν περισσότερο. Μια αξιοσημείωτη εξαίρεση είναι η λειτουργία strip_module FLI, η οποία λαμβάνει module_t * ως παράμετρος στα Low-Level interface, η μέθοδος **strip_module** παίρνει μια **module_t,** όχι Holder για αυτό το είδος.

7.7.3 ΧΡΗΣΙΜΟΠΟΙΩΝΤΑΣ ΤΟ LOW-LEVEL INTERFACE

Οι προγραμματιστές που ήδη είναι εξοικειωμένοι με το FLI δε θα έχουν καμία έκπληξη. Για παράδειγμα, για να δημιουργήσετε ένα **term_t** σε Java, θα κάνουμε το ίδιο όπως θα κάναμε στη C:

```
term_t t = Prolog.new_term_ref();
```

Η διαφορά είναι ότι η μέθοδος Java είναι τώρα μια στατική αυτόχθονα μέθοδος της κατηγορίας Prolog, έτσι ώστε η σύνταξη είναι ελαφρώς διαφορετική από την αντίστοιχη κλήση σε C. Επιπλέον, πολλοί από τους ίδιους κανόνες στην FLI ισχύουν για τη χαμηλού επιπέδου διεπαφή. Να κάνει ένα όρο αναφοράς ο οποίος περιέχει ένα άτομο, για παράδειγμα, κάποιος πρέπει πρώτα να

δημιουργήσετε το **term_t,** στη συνέχεια, ένα **atom_t,** και στη συνέχεια να τεθεί το άτομο μέσα στον όρο, όπως στην FLI:

```
term_t term = Prolog.new_term_ref();

atom_t atom = Prolog.new_atom( "foo" );

Prolog.put_atom( term, atom );
```

7.7.4 ΠΡΟΕΙΔΟΠΟΙΗΣΕΙΣ

Δεν υπάρχει τίποτα το ιδιαίτερο για την χαμηλού επιπέδου interface. Είναι πραγματικά ακριβώς μια ευθεία αντιστοίχιση Java του FLI, και οι προγραμματιστές της C θα έχουν μόνο μια μικρή δυσκολία στη χρήση. Από την άλλη πλευρά, η μετάφραση του FLI για την Java θέτει κάποιες ιδιαιτερότητες.

7.8 ΧΡΗΣΙΜΟΠΟΙΩΝΤΑΣ ΤΟ HIGH -LEVEL INTERFACE

Το ***JPL 3.0.1 Java-calls-Prolog API*** παρέχει μια σειρά από classes που κρύβουν σχεδόν το σύνολο των επιμέρους λεπτομερειών στο Low-Level Interface . Είναι λιγότερο ευέλικτη από το χαμηλού επιπέδου Interface, αλλά έχει επίσης λιγότερο από μια καμπύλη μάθησης,από την χαμηλού επιπέδου Interface.

Στη Java το πακέτο **JPL** περιέχει όλα τις classes σε αυτό το περιβάλλον. Κάθε μια από τις classes αντιστοιχούν με οποιονδήποτε από τους τύπους δεδομένων στην Prolog Foreign Language Interface (FLI).

7.8.1 Η ΙΕΡΑΡΧΙΑ CLASS

Το ***API*** αποτελείται από την ακόλουθη ιεραρχία κατηγορίας:

Term
|
+--- Variable
|
+--- Compound
| |
| +--- Atom
|
+--- Integer
|
+--- Float

Query

JPLException
|
+-- PrologException

Όρος(term) είναι μια αφηρημένη κατηγορία: μόνο τις υποκατηγορίες της μπορεί να αρχικοποιείται.

Κάθε περίπτωση **ερώτημα(query)** περιέχει έναν **όρο** (term)(που υποδηλώνει το στόχο που είναι να αποδειχθεί), και πολλά άλλα.

Κάθε εμφάνιση της **Ενωσης(Compound)** έχει (java.lang.String) το όνομα και μια σειρά από **(Term)** επιχειρήματα (θα πρέπει να έχει τουλάχιστον μία).

Atom είναι μια εξειδίκευση της **Ενώσεως** (**Compound**)με μηδενική επιχειρήματα.

Στη συνέχεια θα αναφερθούμε στη γλώσσα prolog που ουσιαστικά είναι ο λόγος ύπαρξης της JPL.

8 PROLOG

Εδώ θα αναπτύξουμε διάφορα στοιχεία για τη γλώσσα prolog.ξεκινάμε από το τι είναι ,στη συνέχεια μιλάμε για την προέλευση της καθώς και δίνουμε κάποια ιστορικά στοιχεία αναφερόμαστε στη σύνταξη και τη σημασιολογία της ,στους τύπους των δεδομένων της ,τους κανόνες και τα γεγονότα που την απαρτίζουν , μετά αναφέρουμε τις εφαρμογές της (iso prolog, συλλογές ,αναδρομή ουράς,term ευρετηρίασης κατάθεση και την εφαρμογή της στο hardware).Μετα από όλα αυτά δίνουμε μία κριτική για τη γλώσσα τις επεκτάσεις τους περιορισμούς της και τέλος αναφερόμαστε στις διασυνδέσεις της με άλλες γλώσσες .

8.1 ΤΙ ΕΝΑΙ Η PROLOG ;

Η Prolog είναι μια γενικής χρήσης γλώσσα λογικού προγραμματισμού που σχετίζεται με την τεχνητή νοημοσύνη και την υπολογιστική γλωσσολογία .

Prolog έχει τις ρίζες της στην λογική πρώτης τάξης , η τυπική λογική , και σε αντίθεση με πολλές άλλες γλώσσες προγραμματισμού ,η Prolog είναι δηλωτική : η λογική του προγράμματος εκφράζεται σε όρους σχέσεων, εκπροσωπώντας τους ως γεγονότα και κανόνες . Ένας υπολογισμός ξεκίνα εκτελώντας ένα *ερώτημα μέσω* αυτών των σχέσεων.

8.2 Η ΠΡΟΕΛΕΥΣΗ ΤΗΣ

Το όνομα της *Prolog* επιλέχθηκε από τον Philippe Roussel ως συντομογραφία για *programmation en logique* (γαλλικά για *τον προγραμματισμό στη λογική*). Όταν δημιουργήθηκε περίπου 1972 από τον Alain Colmerauer και με τον Philippe Roussel, με βάση την διαδικαστική ερμηνεία των ρητρών Horn του Robert Kowalski 's . Η προσπάθεια αυτή ξεκίνησε εν μέρει από την επιθυμία να συνδυαστεί η χρήση της λογικής ως δηλωτικής γλώσσας αναπαράστασης γνώσης με τη διαδικαστική αναπαράσταση της γνώσης που ήταν δημοφιλής στη Βόρειο Αμερική στα τέλη της δεκαετίας του 1960 και στις αρχές του 1970. Σύμφωνα με τον Robert Kowalski , το πρώτο σύστημα Prolog αναπτύχθηκε το 1972 από τον Alain Colmerauer και Phillipe Roussel. Οι πρώτες υλοποιήσεις της Prolog ήταν “interpreters”, ωστόσο, ο David HD Warren δημιούργησε το Warren Abstract Machine , η έγκαιρη και επιρροή του compiler της Prolog που

ήρθε να καθορίσει την "Edinburgh Prolog" διάλεκτο η οποία χρησίμευσε ως βάση για τη σύνταξη των πιο σύγχρονων εφαρμογών.

Μεγάλο μέρος της σύγχρονης ανάπτυξης της Prolog προήλθε από την ώθηση της Πέμπτης γενιάς Υπολογιστικών Συστημάτων του έργου (FGCS), η οποία ανέπτυξε μια παραλλαγή της Prolog που ονομάζεται *Γλώσσα Kernel* .

Prolog ήταν μία από τις πρώτες γλώσσες προγραμματισμού λογικής, και παραμένει η πιο δημοφιλής μεταξύ των εν λόγω γλωσσών σήμερα, με πολλές ελεύθερες και εμπορικές εφαρμογές που διατίθενται. Ενώ αρχικά δημιουργήθηκε με στόχο την επεξεργασία της φυσικής γλώσσας , η γλώσσα έχει έκτοτε επεκταθεί σε πολλούς άλλους τομείς, όπως αποδείξεις θεωρημάτων ,έμπειρα συστήματα , παιχνίδια, αυτοματοποιημένα συστήματα τηλεφωνητή, οντολογίες και εξελιγμένα συστήματα ελέγχου . Σύγχρονα περιβάλλοντα Prolog υποστηρίζουν τη δημιουργία γραφικών διεπαφών χρήστη , καθώς και των διοικητικών και δικτυακών εφαρμογών.

8.3 ΣΎΝΤΑΞΗ ΚΑΙ ΣΗΜΑΣΙΟΛΟΓΊΑ

Στην Prolog, λογική του προγράμματος εκφράζεται από την άποψη των σχέσεων, και ένας υπολογισμός ξεκινά εκτελώντας ένα *ερώτημα μέσω* αυτών των σχέσεων. Οι σχέσεις και τα ερωτήματα κατασκευάζονται χρησιμοποιώντας μόνο τον τύπο δεδομένων της Prolog, τον *όρο.* Οι σχέσεις καθορίζονται από *τις ρήτρες.* Λαμβάνοντας υπόψη ένα ερώτημα, ο κινητήρας της Prolog προσπαθεί να βρει ένα ψήφισμα διάψευσης για τον εκμηδενισμό του ερωτήματος..\ Το γεγονός αυτό καθιστά την Prolog (και άλλες γλώσσες προγραμματισμού της λογικής) ιδιαίτερα χρήσιμη για τις βάσεις δεδομένων, συμβολικά μαθηματικά, και εφαρμογές ανάλυσης γλώσσας. Επειδή η Prolog επιτρέπει ακαθόριστα κατηγορήματα , ελέγχοντας την τιμή αλήθειας ορισμένα ειδικά κατηγορήματα μπορεί να έχουν κάποια σκόπιμη ενέργεια, όπως το να εκτυπωθεί μια τιμή στην οθόνη. Εξαιτίας αυτού, έχει επιτραπεί στον προγραμματιστής να χρησιμοποιεί κάποιο ποσό των συμβατικών επιτακτικών αναγκών του προγραμματισμού , όταν η λογική για παράδειγμα είναι άβολη. Έχει μια καθαρή λογική υποσύνολου, που ονομάζεται "καθαρή Prolog", καθώς και μια σειρά από extralogical χαρακτηριστικά.

8.4 ΟΙ ΤΎΠΟΙ ΔΕΔΟΜΈΝΩΝ

Στην Prolog ο μόνος τύπος δεδομένων είναι ο *όρος*. Όροι είτε είναι *άτομα, αριθμοί, μεταβλητές* ή *σύνθετοι όροι.*

Ένα **άτομο** είναι ένα γενικού σκοπού όνομα χωρίς εγγενή έννοια. Παραδείγματα ατόμων περιλαμβάνουν x , blue , 'Burrito' , και 'some atom' .

Οι αριθμοί μπορεί να είναι πλωτήρες ή ακέραιοι .

Οι **μεταβλητές** συμβολίζονται από μια σειρά που αποτελείται από γράμματα, αριθμούς και χαρακτήρες υπογράμμισης, και αρχίζοντας με ένα κεφαλαίο γράμμα ή underscore. Οι μεταβλητές μοιάζουν μεταβλητές στη λογική με την έννοια ότι είναι χαρακτήρες κράτησης θέσης για αυθαίρετους όρους.

Ένας **όρος ένωση** αποτελείται από ένα άτομο που ονομάζεται "συναρτητής" και μια σειρά από «επιχειρήματα», που είναι και πάλι όρος. Οι σύνθετοι όροι συνήθως γράφονται ως συναρτήσεις που ακολουθούνται από μια λίστα διαχωρισμένη με κόμματα των όρων επιχείρημα, τα οποία περιέχονται σε παρένθεση. Ο αριθμός των επιχειρημάτων του όρου που ονομάζεται arity . Ένα άτομο μπορεί να θεωρηθεί ως σύνθετος όρος με μηδενική arity.

Παραδείγματα σύνθετων όρων είναι truck_year('Mazda', 1986) και 'Person_Friends'(zelda,[tom,jim]) .

Ειδικές περιπτώσεις των σύνθετων όρων:

Μια *λίστα* είναι μια διατεταγμένη συλλογή των όρων. Είναι συμβολισμένη με αγκύλες και με τους όρους της να διαχωρίζονται με κόμματα ή στην περίπτωση του κενού καταλόγου περιγράφεται έτσι [] . Για παράδειγμα, [1,2,3] ή [red,green,blue] .

Strings: Μια ακολουθία χαρακτήρων που περιβάλλεται από εισαγωγικά είναι ισοδύναμη με μια λίστα (αριθμητικά) κώδικες χαρακτήρων, κατά κανόνα στην τοπική κωδικοποίηση χαρακτήρων , ή Unicode , εάν το σύστημα υποστηρίζει Unicode. Για παράδειγμα, "to be, or not to be" .

8.5 ΚΑΝΟΝΕΣ ΚΑΙ ΓΕΓΟΝΟΤΑ

Προγράμματα Prolog περιγράφουν σχέσεις, ορίζονται μέσω των ρητρών. Η καθαρή Prolog περιορίζεται σε ρήτρες Horn . Υπάρχουν δύο τύποι των ρητρών: τα γεγονότα και τους κανόνες. Ένας κανόνας είναι της μορφής

Head :- Body.

και διαβάζεται ως " Head είναι αλήθεια αν το σώμα είναι αλήθεια». Το σώμα ενός κανόνα αποτελείται από κλήσεις προς κατηγορήματα, τα οποία ονομάζονται **στόχοι** του κανόνα. Το ενσωματωμένο κατηγόρημα ,/2 (που σημαίνει 2-arity φορέα με το όνομα ,) υποδηλώνει συνδυασμό των στόχων, και ;/2 δηλώνει διάζευξη . Κλίσεις και διαχωρισμούς μπορούν να εμφανίζονται μόνο στο σώμα, όχι στο κεφάλι ενός κανόνα.

Οι ρήτρες με κενό οργανισμούς που ονομάζονται **γεγονότα.** Ένα παράδειγμα ενός γεγονότος είναι:

cat (Τομ).

η οποία είναι ισοδύναμη με τον κανόνα:

cat (Τομ): - αλήθεια.

Το ενσωματωμένο κατηγόρημα true/0 είναι πάντα αλήθεια.

Λαμβάνοντας υπόψη τα παραπάνω πραγματικότητα, μπορεί κανείς να ζητήσει:

είναι tom μια γάτα;

; - Cat (Τομ).

Ναί

ποια πράγματα είναι οι γάτες;

; - Cat (X).

X = tom

Οι όροι με τους οργανισμούς που ονομάζονται **κανόνες.** Ένα παράδειγμα ενός κανόνα είναι:

animal(X):- cat(X)

Αν προσθέσουμε τον κανόνα και να ρωτήσω *τι πράγματα είναι ζώα;*

?- animal(X).
X = tom

Λόγω της φύσης των σχεσιακών πολλών ενσωματωμένων κατηγορημάτων, μπορούν τυπικά να χρησιμοποιηθούν σε διάφορες κατευθύνσεις. Για παράδειγμα, το length/2 μπορεί να χρησιμοποιηθεί για να καθορίσει το μήκος ενός καταλόγου (length(List, L) , δίνεται ένας κατάλογος List), καθώς και να δημιουργήσει έναν σκελετό του κατάλογου σε ένα δεδομένο μήκος (length(X, 5)), καθώς επίσης και να δημιουργήσει δύο σκελετούς λίστας και τα μήκη τους μαζί (length(X, L)). Ομοίως, append/3 μπορεί να χρησιμοποιηθεί τόσο για την προσάρτηση δύο λίστων (append(ListA, ListB, X) δίνονται οι λίστες ListA και ListB), καθώς και για να χωρίσει μια συγκεκριμένη λίστα σε τμήματα (append(X, Y, List) , δίνεται μία Λίστα List). Για το λόγο αυτό, ένα συγκριτικά μικρό σύνολο της βιβλιοθήκης με κατηγορήματα αρκεί για πολλά προγράμματα Prolog.

Ως γενική γλώσσα σκοπού,η Prolog παρέχει επίσης διάφορα ενσωματωμένα κατηγορήματα για να εκτελέσει συνήθεις δραστηριότητες, όπως της εισόδου και της εξόδου , χρησιμοποιώντας γραφικά και άλλα επικοινωνεί με το λειτουργικό σύστημα. Αυτά τα κατηγορήματα δεν είναι δεδομένα είναι χρήσιμα μόνο για τις ενέργειες που παρουσιάζουν στο σύστημα. Για παράδειγμα, με το κατηγόρημα write/1 εμφανίζει ένας όρος στην οθόνη.

8.6 ΥΨΗΛΟΤΕΡΗΣ ΤΆΞΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Εξ ορισμού, η λογική πρώτης τάξης δεν επιτρέπει την ποσοτικοποίηση πάνω σε *κατηγορήματα.* (Όσον αφορά τις μεταβλητές, οι μεταβλητές στην κεφαλί σιωπηρά και καθολικά ποσοτικοποιημένες και αυτά μόνο στο σώμα σιωπηρά.) υψηλότερης τάξης κατηγόρημα είναι ένα κατηγόρημα που διαρκεί σένα ή περισσότερα άλλα κατηγορήματα ως επιχειρήματα. Η Prolog έχει ήδη κάποια ενσωματωμένα ανώτερης τάξης κατηγορήματα, όπως η call/1 , call/2 , call/3 , findall/3 , setof/3 , και bagof/3 . Επιπλέον, από αυθαίρετους στόχους η Prolog μπορεί να κατασκευαστεί και να αξιολογείται κατά το χρόνο εκτέλεσης, είναι

εύκολο να γράψει κανείς ανώτερης τάξης κατηγορήματα όπως maplist/2 , το οποίο εφαρμόζεται σε ένα αυθαίρετο κατηγόρημα σε κάθε μέλος μιας συγκεκριμένης λίστας και sublist/3 , το οποίο φιλτράρει τα στοιχεία που ικανοποιούν ένα δεδομένο κατηγόρημα .

Για να μετατρέψετε λύσεις από χρονική αναπαράσταση (αντικαταστάσεις απάντηση για υπαναχωρήσεις) με χωρική αναπαράσταση (άποψη), η Prolog έχει διάφορες λύσεις ,κατηγορήματα που συλλέγουν όλες τις αντικαταστάσεις ,απάντηση ενός συγκεκριμένου ερωτήματος σε μια λίστα. Αυτό μπορεί να χρησιμοποιηθεί για την κατανόηση της λίστας .

8.7 ΕΦΑΡΜΟΓΉ

8.7.1 ISO PROLOG

Το ISO Prolog πρότυπο αποτελείται από δύο μέρη. ISO / IEC 13211 με 1, που δημοσιεύθηκε το 1995, αποσκοπεί στην τυποποίηση των υφιστάμενων πρακτικών από τις πολλές εφαρμογές από τα βασικά στοιχεία της Prolog. Έχει διευκρινίσει πτυχές της γλώσσας που προηγουμένως ήταν ασαφής και οδηγεί σε φορητά προγράμματα. Υπάρχουν δύο διορθώσεις : Cor.1: 2007 και Cor.2: 2012. ISO / IEC 13211-2, που δημοσιεύθηκαν το 2000, προστέθηκε και υποστήριξη για τις ενότητες με το πρότυπο. Το πρότυπο διατηρείται από τον ISO / IEC JTC1 / SC22 / WG17 της ομάδας εργασίας. ANSI X3J17 είναι από της ΗΠΑ η τεχνικοί συμβουλευτική ομάδα για το πρότυπο.

8.7.2 ΣΥΛΛΟΓΕΣ

Για μεγαλύτερη αποτελεσματικότητα ,ο κώδικας Prolog είναι συνήθως συγκεντρωμένος στην abstract machine code, που συχνά επηρεάζεται από το μητρώο, με βάση τα Warren Abstract Machine (WAM) σύνολο εντολών. Μερικές εφαρμογές χρησιμοποιούν αφηρημένη ερμηνεία για την αντλήσει τύπου και πληροφοριών λειτουργίας των κατηγορημάτων κατά τη μεταγλώττιση, ή συγκεντρώνουν σε πραγματικό κώδικα μηχανής για υψηλή απόδοση. Η επινόηση αποτελεσματικής μεθόδου εφαρμογής για τον κωδικό Prolog είναι ένας τομέας της ενεργού έρευνας στην κοινότητα λογικού προγραμματισμού, και οι διάφορες άλλες μέθοδοι εκτέλεσης που χρησιμοποιούνται σε ορισμένες

εφαρμογές. Αυτές περιλαμβάνουν binarization ρήτρα και stack-based virtual machines .

8.7.3 ΑΝΑΔΡΟΜΗ ΟΥΡΑΣ

Στα συστήματα της Prolog η εφαρμογή έχει συνήθως τη γνωστή μέθοδο βελτιστοποίησης που ονομάζεται ουρά βελτιστοποίηση κλήσης (TCO) για τα αιτιοκρατικά κατηγορήματα που εμφανίζουν αναδρομή ουράς ή, γενικότερα, οι κλήσεις της ουράς: η στοίβα ή το πλαίσιο μιας ρήτρας απορρίπτεται πριν από την εκτέλεση μιας κλήσης σε θέση ουράς. Ως εκ τούτου, είναι ντετερμινιστικά ουρά-αναδρομικά κατηγορήματα που εκτελέστηκαν με σταθερή στοίβα χώρου, όπως οι βρόχοι σε άλλες γλώσσες.

8.7.4 TERM ΕΥΡΕΤΗΡΙΑΣΗΣ

Η εύρεση ρήτρας που είναι ενοποιητέα με έναν όρο σε ένα ερώτημα είναι γραμμική στον αριθμό των ρητρών. Η Term ευρετηρίαση χρησιμοποιεί μια δομή δεδομένων που επιτρέπει sublinear χρόνο αναζητήσεις. Η Indexing επηρεάζει μόνο τις επιδόσεις του προγράμματος, δεν επηρεάζει τη σημασιολογία.

8.7.5 ΚΑΤΑΘΕΣΗ

Μερικά συστήματα Prolog, (BProlog , XSB και Yap), εφαρμόζουν μία memoization μέθοδο που ονομάζεται *κατάθεση,* η οποία απελευθερώνει το χρήστη από την χειροκίνητη αποθήκευση των ενδιάμεσων αποτελεσμάτων.

8.7.6 ΕΦΑΡΜΟΓΗ ΣΕ HARDWARE

Κατά τη διάρκεια του πέμπτου υπολογιστικού συστήματος του έργου (FGCS) , υπήρξαν προσπάθειες για την εφαρμογή της Prolog σε hardware με στόχο την επίτευξη ταχύτερης εκτέλεσης με ειδικές αρχιτεκτονικές. Επιπλέον, η Prolog έχει μια σειρά από ιδιότητες που μπορούν να επιτρέψουν την μέγιστη ταχύτητα μέσω παράλληλης εκτέλεσης. Μια πιο πρόσφατη προσέγγιση είναι να συγκεντρώσει περιορισμένα προγράμματα Prolog σε ένα πεδίο προγραμματιζόμενης συστοιχίας πύλης . Ωστόσο, η ταχεία πρόοδος σε γενικού σκοπού hardware έχει ξεπεράσει σταθερά τις πιο εξειδικευμένες αρχιτεκτονικές.

8.8 ΚΡΙΤΙΚΗ

Αν και η Prolog χρησιμοποιείται ευρέως στον τομέα της έρευνας και της εκπαίδευσης, Η Prolog όπως και άλλες γλώσσες λογικού προγραμματισμού δεν είχαν σημαντικό αντίκτυπο στον κλάδο της πληροφορικής εν γένει. Οι περισσότερες εφαρμογές είναι μικρές σε σχέση με τα βιομηχανικά πρότυπα, με λίγες άνω των 100.000 γραμμές κώδικα. Ο προγραμματισμός σε μεγάλα προγράμματα θεωρείται πως είναι περίπλοκος, διότι δεν μπορούν όλες οι μονάδες να έχουν compilers υποστήριξης Prolog, και υπάρχουν προβλήματα συμβατότητας μεταξύ των συστημάτων που ενοποιούν τους μεγάλους compilers Prolog. Η φορητότητα του κώδικα Prolog σε υλοποιήσεις έχει επίσης ένα πρόβλημα , αλλά οι εξελίξεις από το 2007 ήταν προσανατολισμένες στο να διορθωθεί το θέμα της φορητότητας.

Το λογισμικό που αναπτύχθηκε σε Prolog έχει επικριθεί για το γεγονός ότι ένα υψηλό κόστος σε απόδοση σε σύγκριση με τις συμβατικές γλώσσες προγραμματισμού. Ωστόσο, οι πρόοδοι στις μεθόδους εφαρμογής μείωσαν το κόστος σε ποσοστό 25% -50% για ορισμένες εφαρμογές.

Η Prolog δεν είναι απλώς δηλωτική: παίζει ρόλο η σειρά των ρητρών σε ένα πρόγραμμα Prolog είναι σημαντική διότι αλλάζει το νόημα εντελώς . Άλλες γλώσσες λογικού προγραμματισμού, όπως Datalog , είναι πραγματικά δηλωτική: ρήτρες μπορούν να δοθούν με οποιαδήποτε σειρά.

8.9 ΕΠΕΚΤΆΣΕΙΣ

Διάφορες εφαρμογές έχουν αναπτυχθεί από την Prolog να επεκτείνουν τις δυνατότητες του λογικού προγραμματισμού σε πολλές κατευθύνσεις. Αυτά περιλαμβάνουν τύπους , μοντέλα, Περιορισμούς Λογικός Προγραμματισμός (CLP), object-oriented προγραμματισμό λογικής (OOLP), συγχρονισμός, γραμμική λογική (LLP), λειτουργικές και ανώτερης τάξης λογική δυνατότητες προγραμματισμού, καθώς και η διαλειτουργικότητα με βάσεις γνώσης:

8.10 ΤΎΠΟΙ

Η Prolog είναι μια untyped γλώσσα. Προσπάθειες για την εισαγωγή των τύπων χρονολογούνται από τη δεκαετία του 1980, και από το 2008 υπάρχουν ακόμη προσπάθειες να επεκταθεί η Prolog με τους τύπους. Πληροφορίες του τύπου

είναι χρήσιμη όχι μόνο για την ασφάλεια τύπου , αλλά και για την αιτιολογία σχετικά με τα προγράμματα Prolog.

8.11 MODES

Η σύνταξη της Prolog δεν διευκρινίζει ποια κατηγόρηματα είναι οι συντελεστές και τα οποία είναι αποτελέσματα. Ωστόσο, η πληροφορία αυτή είναι σημαντική και συνιστάται να συμπεριληφθεί στα σχόλια. Η λειτουργίες παρέχουν πολύτιμες πληροφορίες όταν η συλλογιστική σχετικά με τα προγράμματα Prolog μπορεί επίσης να χρησιμοποιηθεί και για την επιτάχυνση της εκτέλεσης.

8.12 ΠΕΡΙΟΡΙΣΜΟΙ

Οι περιορισμοί του λογικού προγραμματισμού Prolog επεκτείνεται για να συμπεριλάβει ιδέες από την ικανοποίηση των περιορισμών . Σε ένα πρόγραμμα λογικός περιορισμός επιτρέπει περιορισμούς στο σώμα των ρητρών, όπως: A(X,Y) :- X+Y>0. Είναι κατάλληλη σε μεγάλης κλίμακας συνδυαστικής βελτιστοποίησης στα προβλήματα. Και έτσι είναι χρήσιμο για εφαρμογές σε βιομηχανικές ρυθμίσεις, όπως η αυτοματοποίηση χρονοδιαγραμμάτων και στον προγραμματισμό της παραγωγής .

8.13 ΔΙΑΣΥΝΔΕΣΕΙΣ ΣΕ ΑΛΛΕΣ ΓΛΩΣΣΕΣ

Υπάρχουν στοιχεία τα οποία μπορούν να γεφυρώσουν το χάσμα μεταξύ Prolog και σε άλλες γλώσσες:

Ο διακομιστής Intelligence LPA επιτρέπει την ενσωμάτωση της LPA Prolog με τη C, C #, C + +, Java, VB, Delphi,. Net, Lua, Python και άλλες γλώσσες. Εκμεταλλεύεται την ειδική σειρά δεδομένων τύπου και τους παρέχει LPA Prolog

Ο API διακομιστής Logic επιτρέπει τόσο την επέκταση και την ενσωμάτωση της Prolog σε C, C + +, Java, VB, Delphi,. NET και οποιαδήποτε γλώσσα / περιβάλλον, το οποίο μπορεί να καλέσει έναν. Dll . Η σύμβαση αυτή εφαρμόζεται για Amzi! Prolog Amzi! Amzi! Prolog + Logic Server , αλλά η προδιαγραφή API μπορεί να διατεθεί για οποιαδήποτε εφαρμογή.

Η JPL είναι μια αμφίδρομη γέφυρα Prolog Java, η οποία αποστέλλεται με SWI-Prolog από προεπιλογή, επιτρέποντας Java και Prolog να καλούν ο ένας τον άλλο (αναδρομικά). Είναι γνωστό ότι έχουν καλή υποστήριξη ταυτοχρονισμού και είναι υπό ενεργό ανάπτυξη.

InterProlog , μια γέφυρα μεταξύ της βιβλιοθήκης προγραμματισμού Java και Prolog, εφαρμογή αμφίδρομης σχέσης κατηγόρημα / μέθοδου καλώντας μεταξύ των δύο γλωσσών. Τα Java αντικείμενα μπορούν να αντιστοιχιστούν με όρους Prolog και αντίστροφα. Επιτρέπει την ανάπτυξη των GUIs και άλλων λειτουργιών σε Java, αφήνοντας τη λογική επεξεργασία στο στρώμα της Prolog. Υποστηρίζει την XSB , με παράλληλη υποστήριξη για SWI-Prolog και YAP προγραμματισμό για το 2013.

Η " prova " παρέχει εγγενή ενοποίηση με τη σύνταξη της Java, τα μηνύματα παράγοντα και κανόνες αντίδρασης. Η " prova " τοποθετεί τον εαυτό της κατά κανόνα με βάση το scripting (RBS) middleware σύστημα. Η γλώσσα ανοίγει νέους ορίζοντες στο συνδυασμό επιτακτικού και δηλωτικού προγραμματισμού .

Prol μία μηχανη ενσωμάτωση για Prolog και για την Java. Περιλαμβάνει ένα μικρό IDE και μερικές βιβλιοθήκες.

GNU Prolog για Java είναι μια εφαρμογή του ISO Prolog ως βιβλιοθήκη της java (gnu.prolog)

Ciao παρέχει διεπαφές με C, C + +, Java, και σχεσιακές βάσεις δεδομένων.

C #-Prolog είναι ένας διερμηνέας Prolog γραμμένος σε (managed) C# .Μπορούν εύκολα να ενσωματωθούν σε προγράμματα C #. Χαρακτηριστικά: αξιόπιστη και αρκετά γρήγορη διερμηνεία, περιβάλλον γραμμής εντολών, Windows-interface, ενσωματωμένα DCG, XML-κατηγορήματα, SQL, κατηγορήματα, παράτασης. Ο πλήρης πηγαίος κώδικας είναι διαθέσιμος, συμπεριλαμβανομένου μιας parser generator που μπορεί να χρησιμοποιηθεί για την προσθήκη ειδικού σκοπού επεκτάσεις.

Jekejeke Prolog API παρέχει στενή σύνδεδεση ταυτόχρονη κλήση-in για να καλέσει τις διευκολύνσεις μεταξύ προλόγου και Java ή Android, με την αξιοσημείωτη δυνατότητα δημιουργίας μεμονωμένων αντικειμένων μιας βάσης γνώσεων. Μπορεί να χρησιμοποιηθεί για να ενσωματώσετε το διερμηνέα Prolog ISO σε standalones, applets, servlets, apks, κλπ..

Μια Warren Abstract Machine για την PHP Ένας μεταγλωττιστής Prolog και διερμηνέα σε PHP 5.3. Μια βιβλιοθήκη που μπορεί να χρησιμοποιηθεί αυτόνομα ή εντός Symfony2.1 πλαίσιο

Μετά την ολοκλήρωση των σχολιασμών των γλωσσών προγραμματισμού θα προχωρήσω στην ανάλυση των βοηθητικών τεχνολογιών που δίνουν κάποιες επιπλέων δυνατότητες στον προγραμματιστή

9 GIT

Σε αυτό το κεφάλαιο περιγράφουμε την τεχνολογια Git .Αρχικά αναφέρουμε το κύριο χαρακτηριστικό του GIT που το κάνει να ξεχωρίζει μετά από αυτό τη σχέση του git με την ταχύτητα μετά αναλύουμε τα χαρακτηριστικά που διαφοροποιούν το git από τα scm, αναφερόμαστε στη δημιουργία πολλαπλών αντιγράφων ασφαλείας ,στο πως μπορούν πολύ προγραμματιστές να αλληλεπιδρούν ταυτόχρονα σε κοινό αποθετήριο ,μετά αναπτύσσουμε την ιστορία του git πως γίνετε η εγκατάσταση και τέλος τι γινόταν πρίν τη χρήση του git .

9.1 ΠΟΙΟ ΤΟ ΚΥΡΙΟ ΧΑΡΑΚΤΗΡΙΣΤΙΚΟ ΤΟΥ GIT ΠΟΥ ΤΟ ΚΑΝΕΙ ΝΑ ΞΕΧΩΡΙΖΕΙ;

Το χαρακτηριστικό του Git που πραγματικά το κάνει να ξεχωρίζει από σχεδόν κάθε άλλο SCM είναι η διακλάδωση του μοντέλο. Το Git επιτρέπει και ενθαρρύνει να έχετε πολλαπλά τοπικά branches που μπορεί να είναι εντελώς ανεξάρτητα μεταξύ τους. Η δημιουργία, συγχώνευση, και τη διαγραφή αυτών των γραμμών της ανάπτυξης διαρκεί δευτερόλεπτα.

Αυτό σημαίνει ότι μπορείτε να κάνετε πράγματα όπως:

9.1.1 ΑΛΛΑΓΗ ΠΕΡΙΕΧΟΜΕΝΟΥ.

Δημιουργήστε ένα branche για να δοκιμάσετε μια ιδέα, μεταβείτε πίσω στο σημείο όπου θα επεκταθεί και συγχωνεύστε το αυτό . Έχετε ένα κλάδο που πάντα περιέχει μόνο ό, τι έχει τελειοποιηθεί , ένα άλλο που συγχωνεύονται οι εργασίες επιχειρήσεις για την καθημερινή εργασία ,για τη δοκιμή, και πολλές άλλες μικρότερες εργασίες.

Χαρακτηριστικό βάση της ροής εργασίας.

Δημιουργία νέων branches για κάθε νέο χαρακτηριστικό στο οποίο εργάζεστε, έτσι ώστε να μπορούν απρόσκοπτα να εναλλάσσονται μεταξύ τους, στη συνέχεια, διαγράψτε κάθε κλάδο, όταν αυτό το χαρακτηριστικό πλέον έχει τελειοποιηθεί και μπορει να συγχωνευτεί στη κύρια γραμμή σας.

9.1.2 ΠΕΙΡΑΜΑΤΙΣΜΟΣ .

Μπορείτε να δημιουργήστε ένα branche για να πειραματιστείτε ,και μόλις αποφασίσετε πως δεν θα σας καλύψει τις ανάγκες μπορείτε να το διαγράψετε χωρίς να αφήσει ίχνη .

9.2 GIT ΚΑΙ ΤΑΧΥΤΗΤΑ

Το Git είναι γρήγορο. Με το Git, σχεδόν όλες οι εργασίες εκτελούνται σε τοπικό επίπεδο και αυτό δίνει ένα τεράστιο πλεονέκτημα ταχύτητας σε κεντρικά συστήματα που συνεχώς πρέπει να επικοινωνούν με ένα διακομιστή κάπου.

Το Git χτίστηκε για να εργαστεί στον πυρήνα του Linux, πράγμα που σημαίνει ότι έπρεπε να χειριστεί αποτελεσματικά μεγάλα αποθετήρια από την πρώτη μέρα. Το Git είναι γραμμένο σε C, μειώνοντας την επιβάρυνση του χρόνου που χρειάζεται για την σύνδεση του με υψηλότερου επιπέδου γλώσσες. Η ταχύτητα και η απόδοση ήταν ο πρωταρχικός στόχος του σχεδιασμού του Git.

9.3 ΧΑΡΑΤΗΡΙΣΤΙΚΑ ΤΟΥ GIT ΠΟΥ ΤΟ ΔΙΑΦΟΡΟΠΟΙΟΥΝ ΑΠΟ ΤΑ SCM

9.3.1 ΚΑΤΑΝΕΜΗΜΕΝΟ

Ένα από τα ωραιότερα χαρακτηριστικά του του Git σε σχέση με οποιοδήποτε SCM, είναι οτι περιλαμβάνει , ότι έχει διανεμηθεί. Αυτό σημαίνει ότι αντί να κάνει ένα " checkout " της τρέχουσας άκρης του πηγαίου κώδικα, θα δημηουργίσει έναν "κλώνο" του συνόλου του αποθετηρίου.

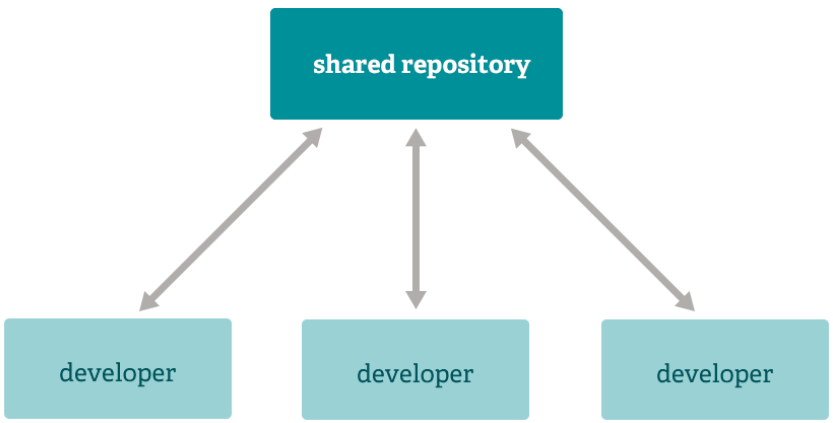
9.3.2 ΠΟΛΛΑΠΛΑ ΑΝΤΙΓΡΑΦΑ ΑΣΦΑΛΕΙΑΣ

Αυτό σημαίνει ότι ακόμα και αν χρησιμοποιείτε μια κεντρική ροή εργασίας, κάθε χρήστης έχει ουσιαστικά ένα πλήρες αντίγραφο ασφαλείας του κεντρικού διακομιστή. Κάθε ένα από αυτά τα αντίγραφα θα μπορούσε να χρησιμοποιηθεί για να αντικαταστήσει τον κύριο διακομιστή σε περίπτωση σύγκρουσης ή διαφθοράς.

Κάθε workflow . Λόγω της κατανεμημένης φύσης του Git και το υπέροχο σύστημα διακλάδωσης, μια σχεδόν ατελείωτη σειρά των ροών εργασίας μπορεί να υλοποιηθεί με σχετική ευκολία.

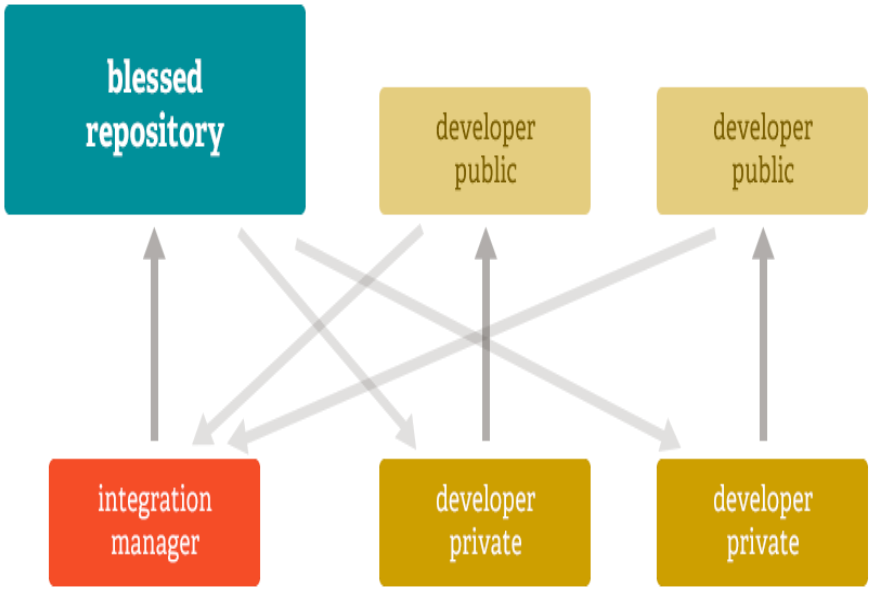
9.3.3 SUBVERSION -STYLEWORKFLOW

Μια κεντρική ροή εργασίας είναι πολύ κοινή, ειδικά για τους ανθρώπους που προέρχονται από ένα σύστημα που αλληλεπιδρούν πολλοί ταυτόχρονα. Το Git δεν θα σας επιτρέψει να κάνετε κάποια τροποποίηση αν για κάποιο λόγο αυτό δεν μπορεί να υλοποιηθεί έτσι, ένα σύστημα που αλληλεπιδρούν πολλοί ταυτόχρονα, όπου όλοι οι προγραμματιστές αλληλεπιδρούν πάνω στον ίδιο διακομιστή λειτουργεί μια χαρά.



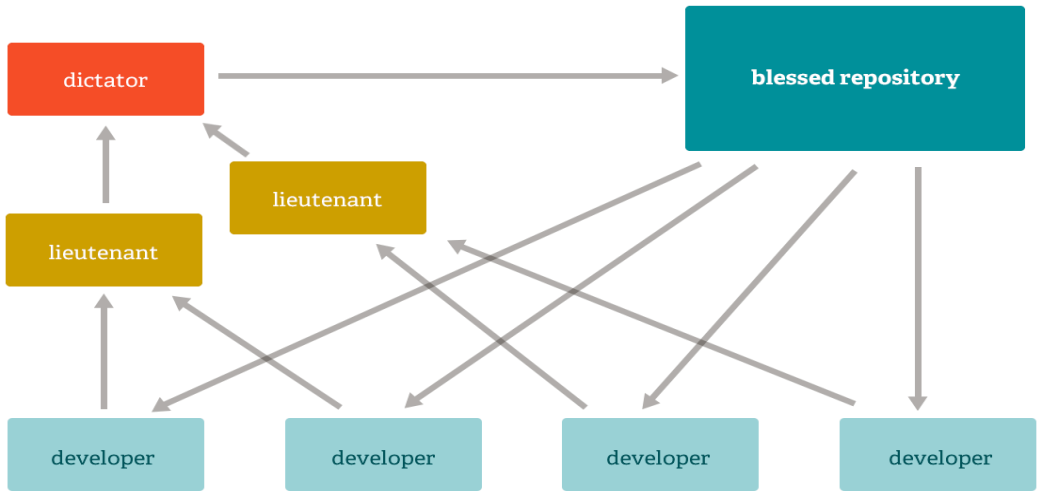
9.3.4 INTEGRATION MANAGER WORKFLOW

Μια άλλη κοινή ροή εργασίας Git περιλαμβάνει έναν διαχειριστή (integration manager) - ένα άτομο που υλοποιεί την αποθήκευση ('blessed' repository). Ένα πλήθος από προγραμματιστές στη συνέχεια που εργάζονται πάνω στους κλώνους από αυτό το αρχείο, ώστε να υλοποιήσουν το δικό του ανεξάρτητο αρχείο , και να ζητήσουν από τον διαχειριστή (integration manager) να ενσωματώσει τις αλλαγές τους. Αυτό είναι το είδος του μοντέλου ανάπτυξης συχνά με τον ανοιχτό κώδικα ή GitHub repositories.



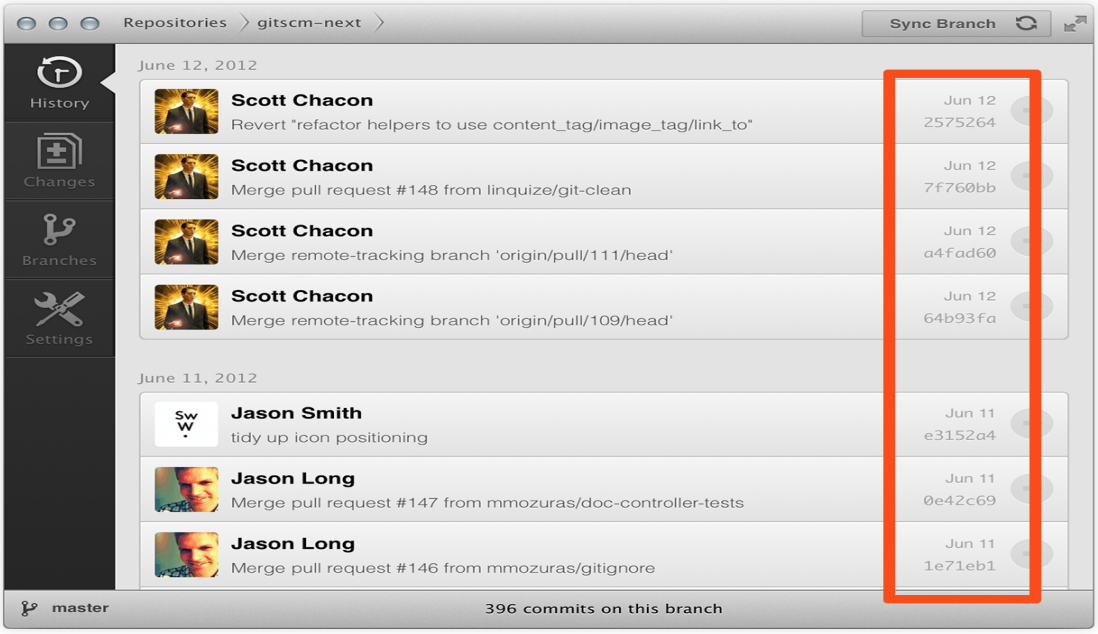
9.3.5 DICTATOR AND LIEUTENANTS WORKFLOW

Για τα περισσότερα τεράστια έργα, η ανάπτυξη της ροής εργασίας, όπως αυτή του πυρήνα του Linux (Linux kernel) είναι συχνά αποτελεσματική. Σε αυτό το μοντέλο, κάποιοι προγραμματιστές ('lieutenants')είναι υπεύθυνοι για ένα συγκεκριμένο υποσύστημα του έργου και συγχωνεύουν σε όλες τις αλλαγές που σχετίζονται με το υποσύστημα. Μια άλλη υλοποίηση («Dictator») μπορεί να τραβήξει τις αλλαγές μόνο από τους Lieutenants και στη συνέχεια να τις αποθηκεύσει ('blessed' repository) στο αρχείο και ο καθένας στη συνέχεια δουλεύει πάνω στους κλώνους του αρχείου.



9.3.6 ΔΙΑΣΦΑΛΙΣΗ ΣΤΑ ΔΕΔΟΜΕΝΑ

Το μοντέλο δεδομένων που χρησιμοποιεί Git εξασφαλίζει την κρυπτογραφική ακεραιότητα για το κάθε κομμάτι του έργου σας. Στο κάθε αρχείο η δέσμευση είναι checksummed και ανακτάτε από το checksum του όταν ελέγχεται . Είναι αδύνατο να πάρει κάποιος κάτι από Git εκτός από τα ακριβή κομμάτια που γίνονται input.

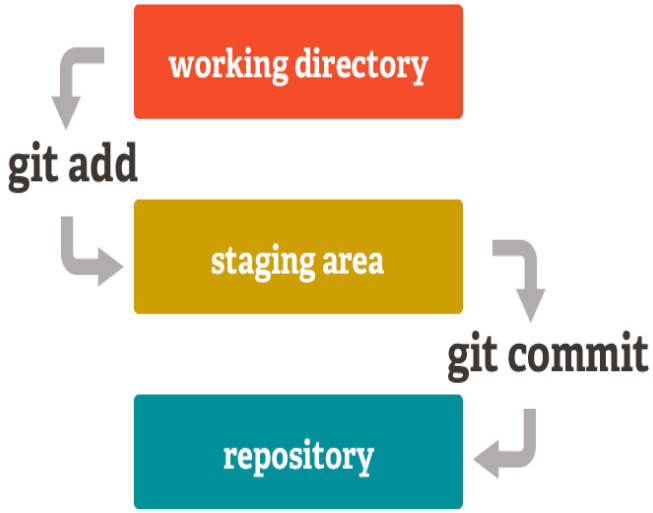


Είναι επίσης αδύνατο να αλλάξει σε οποιοδήποτε αρχείο, ημερομηνία, commit μήνυμα, ή οποιαδήποτε άλλα δεδομένα σε ένα αρχείο Git χωρίς να αλλάζει τις ταυτότητες των αρχείων μετά από αυτό. Αυτό σημαίνει ότι αν έχετε ένα commit ID, μπορείτε να είστε βέβαιοι όχι μόνο ότι το έργο σας είναι ακριβώς το ίδιο όπως όταν διεπράχθη, αλλά ότι τίποτα στην ιστορία του δεν άλλαξε.

9.3.7 STAGING AREA

Σε αντίθεση με τα άλλα συστήματα, το Git έχει κάτι που ονομάζεται "περιοχή ανασυγκρότησης» ή «δείκτη». Αυτό είναι ένα ενδιάμεσος χώρος, όπου δεσμεύεται να διαμορφωθεί και να αναθεωρηθεί πριν από την ολοκλήρωση του commit.

Ένα πράγμα που θέτει το Git σε καλύτερη θέση από τα άλλα εργαλεία είναι ότι υπερτερή στη γρήγορη οργανώση μερικών από τα αρχεία σας και να τα δεσμεύσει χωρίς να επηρεάσει όλα τα άλλα αρχεία στον κατάλογο εργασίας σας και να τα τροποποιήσει ή να χρειάζεται να τα απαριθμήσει στη γραμμή εντολών κατά την διάρκεια της commit.



Αυτό σας επιτρέπει να οργανώσετε μόνο τα τμήματα ενός τροποποιημένου αρχείου.

9.4 ΙΣΤΟΡΙΑ ΤΟΥ GIT

Όπως και με πολλά σπουδαία πράγματα στη ζωή, το Git ξεκίνησε από ένα λάθος και εγινε η αφορμή για πολλές συζητήσεις και αντιπαράθεσης . Ο πυρήνας του Linux είναι ένα έργο ανοικτού λογισμικού πηγής αρκετά μεγάλου πεδίου εφαρμογών. Για το μεγαλύτερο μέρος της ζωής του Linux kernel έκδοσης (1991-2002), οι αλλαγές στο λογισμικό του πέρασαν γύρω από τα patches και τα αρχειοθετημένα αρχεία. Το 2002 Linux kernel άρχισε να χρησιμοποιεί ένα ιδιόκτητο σύστημα DVCS ονομάζεται bitkeeper.

Το 2005, η σχέση μεταξύ της κοινότητας που ανέπτυξε τον πυρήνα του Linux και την εμπορική εταιρεία που ανέπτυξε το bitkeeper χάλασε, καθώς και η δωρεάν χωρίς χρέωση επιδιόρθωσης του εργαλείου έχει ακυρωθεί. Αυτό ώθησε την κοινότητα του Linux στην ανάπτυξη (και ιδίως τον Linus Torvalds, ο δημιουργός του Linux) για να αναπτύξουν το δικό τους εργαλείο με βάση ορισμένα από τα διδάγματα που αντλήθηκαν κατά τη χρήση bitkeeper. Μερικοί από τους στόχους του νέου συστήματος ήταν ως ακολούθως:

- Ταχύτητα
- Απλός σχεδιασμός
- Ισχυρή στήριξη για μη-γραμμική ανάπτυξη (χιλιάδες παράλληλους κλάδους)
- Πλήρως κατανεμημένα
- Ικανή να διαχειριστεί μεγάλα έργα, όπως το Linux kernel αποτελεσματικά (ταχύτητα και το μέγεθος των δεδομένων)

Από τη γέννησή του το 2005, το Git έχει εξελιχθεί και ωριμάσει για να είναι εύκολο στη χρήση και διατηρεί ακόμα τις αρχικές του ιδιότητες. Είναι απίστευτα γρήγορο, είναι πολύ αποτελεσματικό με τα μεγάλα έργα, και έχει ένα απίστευτο σύστημα διακλάδωσης για τη μη-γραμμική ανάπτυξη .

9.5 ΕΓΚΑΤΑΣΤΑΣΗ ΤΟΥ GIT

Εάν μπορείτε, είναι γενικά χρήσιμο να εγκαταστήσετε το Git από την πηγή, γιατί θα έχετε την πιο πρόσφατη έκδοση. Κάθε έκδοση του Git τείνει να περιλαμβάνει χρήσιμες βελτιώσεις , έτσι ώστε να πάρει την πιο πρόσφατη έκδοση η οποία είναι συχνά η καλύτερη δυνατή, οπότε είναι καλύτερα να το κατεβάσετε το λογισμικό κατευθείαν από την πηγή. Είναι επίσης αλήθεια ότι πολλές διανομές του Linux περιέχουν πολύ παλιά πακέτα. Για να εγκαταστήσετε το Git, θα πρέπει να έχετε τις ακόλουθες βιβλιοθήκες από τις οποίες εξαρτάται το Gitcurl, zlib, openssl, expat, and libiconv. Για παράδειγμα, αν είστε σε ένα σύστημα που έχει yum (όπως Fedora) ή το apt-get (όπως Debian based σύστημα), μπορείτε να χρησιμοποιήσετε μία από αυτές τις εντολές για να εγκαταστήσετε όλες τις εξαρτήσεις:

```
$ Apt-get install libcurl4-GnuTLS-dev libexpat1-dev gettext \
  libz-dev libssl-dev
```

Όταν έχετε όλες τις απαραίτητες εξαρτήσεις, μπορείτε να προχωρήσετε και να εκτελέσετε την τελευταία στιγμιαία εικόνα από την ιστοσελίδα Git:

http://git-scm.com/download

Στη συνέχεια, μεταγλωττίσετε και να εγκαταστήσετε:

$ Tar-zxf git-1.7.2.2.tar.gz
$ Cd git-1.7.2.2
$ Κάνουν prefix = / usr / local όλα
$ Sudo make prefix = / usr / local install

9.5.1 ΕΓΚΑΤΑΣΤΑΣΗ ΣΕ LINUX

Εάν θέλετε να εγκαταστήσετε το Git σε Linux μέσω μιας δυαδικής εγκατάστασης, μπορείτε γενικά να το πράξετε μέσω του βασικού εργαλείου πακέτου διαχείρισης που έρχεται με τη διανομή σας. Αν είστε σε Fedora, μπορείτε να χρησιμοποιήσετε το yum:

$ Yum install git-core

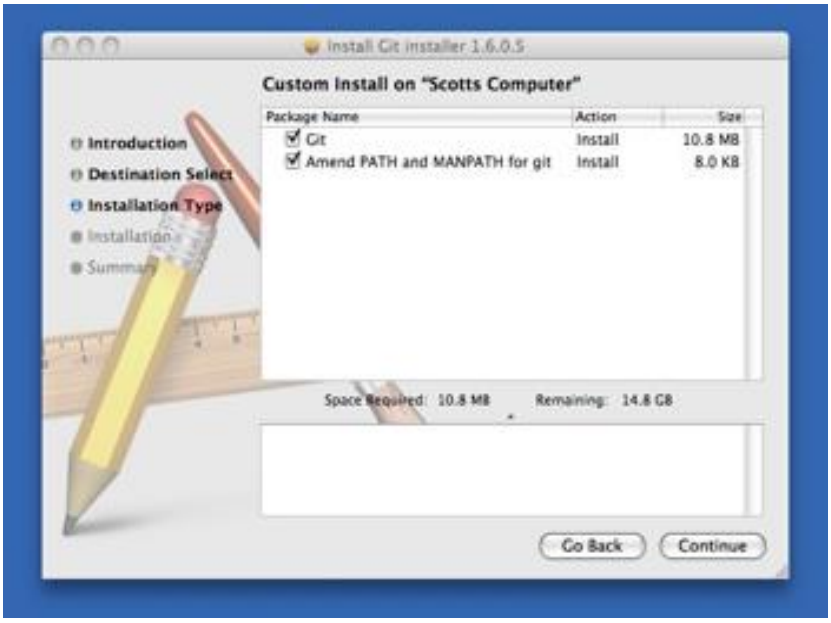
Ή αν είστε σε ένα σύστημα Debian-based διανομή όπως το Ubuntu, δοκιμάστε το apt-get:

$ Apt-get install git

9.5.2 ΕΓΚΑΤΑΣΤΑΣΗ ΣΕ MAC

Υπάρχουν δύο εύκολοι τρόποι για να εγκαταστήσετε Git σε έναν υπολογιστή Mac. Το πιο εύκολο είναι να χρησιμοποιήσετε τον γραφικό εγκαταστάτη Git, τον οποίο μπορείτε να κατεβάσετε από την σελίδα του Google Code :

http://code.google.com/p/git-osx-installer



Εικόνα 2 Git OS X installer

9.5.3 ΕΓΚΑΤΑΣΤΑΣΗ ΣΕ WINDOWS

Εγκατάσταση Git για Windows είναι πολύ εύκολη. Το έργο msysGit διαθέτει μια από τις ευκολότερες διαδικασίες εγκατάστασης. Απλώς κατεβάστε το αρχείο εγκατάστασης exe από τη σελίδα GitHub, και να το εκτελέσετε:

http://msysgit.github.com/

Αφού εγκατασταθεί, έχετε μια έκδοση γραμμής εντολής(συμπεριλαμβανομένου ενός πελάτη SSH, που θα έρθει σε πρακτικό αργότερα) και το πρότυπο GUI.

Σημείωση σχετικά με τη χρήση των Windows: θα πρέπει να χρησιμοποιήσετε Git με το παρεχόμενο msysGit κέλυφος (στυλ Unix). Αν χρειαστεί, για κάποιο λόγο, να χρησιμοποιηθούν το native Windows shell / command κονσόλα της γραμμής, θα πρέπει να χρησιμοποιήσετε διπλά εισαγωγικά αντί για τα απλά εισαγωγικά (για τις παραμέτρους με κενά σε αυτά) και θα πρέπει να αναφερθούν οι παράμετροι που τελειώνουν με την προφορά της περισπωμένη (^) αν είναι το τελευταίο στη γραμμή, όπως είναι ένα σύμβολο συνέχεια στα Windows.

9.6 ΠΡΙΝ ΤΗ ΧΡΗΣΗ ΤΟΥ GIT

Τώρα που έχετε Git στο σύστημά σας, θα θελήσετε να κάνετε μερικά πράγματα για να προσαρμόσετε το περιβάλλον του Git σας. Θα πρέπει να κάνετε αυτά τα πράγματα μόνο μια φορά. θα μείνει έτσι ακόμα και μετά από τις αναβαθμίσεις. Μπορείτε να τα αλλάξετε , επίσης, ανά πάσα στιγμή, με το τρέξιμο των εντολών αυτών και πάλι. Το Git έρχεται με ένα εργαλείο που ονομάζεται git config που σας επιτρέπει να πάρετε και να ρυθμίσετε τις μεταβλητές που ελέγχουν όλες τις πτυχές του τρόπου με τον οποίο το Git θα μοιάζει και θα λειτουργεί. Αυτές οι μεταβλητές μπορούν να αποθηκευτούν σε τρεις διαφορετικές θέσεις:

• / etc / gitconfig αρχείου:

Περιέχει τιμές για κάθε χρήστη στο σύστημα και όλα τα αποθετήρια τους. Αν περάσετε την επιλογή - σύστημα για να git config, διαβάζει και γράφει από αυτό το αρχείο συγκεκριμένα.

• ~ / gitconfig αρχείου:

Ειδικά για το χρήστη. Μπορείτε να κάνετε το Git να διαβάσει και να γράψει σε αυτό το αποθετήριο ειδικά με το πέρασμα του - με γενική επιλογή.

• αρχείο ρυθμίσεων στον κατάλογο του git (δηλαδή, .git / config) ποιο είναι το αποθετήριο που χρησιμοποιείτε αυτήν τη στιγμή: Ειδικά για το ενιαίο αποθετήριο. Κάθε επίπεδο υπερισχύει σε τιμές στο προηγούμενο επίπεδο, έτσι ώστε οι τιμές στην .git / config να έχουν ατού σε σχέση με αυτές στο αρχείο / etc / gitconfig.

9.6.1 Η ΤΑΥΤΟΤΗΤΑ ΣΑΣ

Το πρώτο πράγμα που πρέπει να κάνετε κατά την εγκατάσταση του Git είναι να ορίσετε το όνομα χρήστη και διεύθυνση ηλεκτρονικού ταχυδρομείου. Αυτό είναι σημαντικό επειδή κάθε commit Git χρησιμοποιεί αυτές τις πληροφορίες, και είναι αμετάβλητες :

$ git config --global user.name "John Doe" $ git config --global user.email johndoe@example.com

Και πάλι, θα πρέπει να το κάνετε αυτό μόνο μία φορά, αν περάσει το --global επιλογή, γιατί τότε το Git θα χρησιμοποιεί πάντα τις πληροφορίες για ό, τι κάνετε σε αυτό το σύστημα. Αν θέλετε να τροποποιήσετε αυτό με διαφορετικό όνομα ή διεύθυνση e-mail για τα συγκεκριμένα έργα, μπορείτε να εκτελέσετε την εντολή χωρίς το --global επιλογή όταν είστε σε αυτό το σχέδιο.

9.6.2 ΕΚΔΟΤΗΣ ΣΑΣ

Τώρα που η ταυτότητά σας έχει δημιουργηθεί, μπορείτε να ρυθμίσετε το προεπιλεγμένο επεξεργαστή κειμένου που θα χρησιμοποιηθεί όταν το Git χρειάζεται να πληκτρολογήσετε ένα μήνυμα. Από προεπιλογή, το Git χρησιμοποιεί το προεπιλεγμένο πρόγραμμα επεξεργασίας του συστήματός σας, το οποίο είναι γενικά Vi ή Vim. Αν θέλετε να χρησιμοποιήσετε ένα διαφορετικό πρόγραμμα επεξεργασίας κειμένου, όπως το Emacs, μπορείτε να κάνετε τα εξής:

$ git config --global core.editor emacs

9.6.3 ΤΟ DIFF ΕΡΓΑΛΕΙΟ ΣΑΣ

Μια άλλη χρήσιμη επιλογή μπορεί να θέλετε να ρυθμίσετε τις παραμέτρους είναι η προεπιλεγμένη diff εργαλείο που θα χρησιμοποιηθεί για την επίλυση των συγκρούσεων συγχώνευσης. Ας πούμε ότι θέλετε να χρησιμοποιήσετε vimdiff:

$ git config --global merge.tool vimdiff

Το Git δέχεται KDiff3, tkdiff, xxdiff,vimdiff, gvimdiff, ECMerge και opendiff ως έγκυρα εργαλεία συγχώνευσης. Μπορείτε επίσης να ορίσετε ένα προσαρμοσμένο εργαλείο.

9.6.4 ΄ΕΛΕΓΧΟΣ ΤΩΝ ΡΥΘΜΙΣΕΩΝ

Αν θέλετε να ελέγξετε τις ρυθμίσεις σας, μπορείτε να χρησιμοποιήσετε το git config --list που είναι η εντολή για να καταγραφούν όλες οι ρυθμίσεις του Git που μπορούν να βρεθούν σε αυτό το σημείο:

$ git config --list user.name=Scott Chacon user.email=schacon@gmail.com color.status=auto color.branch=auto color.interactive=auto color.diff=auto ...

9.6.5 ΛΗΨΗ ΒΟΗΘΕΙΑΣ

Αν ποτέ χρειαστείτε βοήθεια κατά τη χρήση του Git, υπάρχουν τρεις τρόποι για να πάρετε την σελίδα βοηθείας (man σελίδες) βοηθούν για οποιαδήποτε από τις εντολές Git:

$ git help <verb> $ git <verb> --help $ man git-<verb>

Για παράδειγμα, μπορείτε να πάρετε τη βοήθεια σελίδα man για την εντολή config με την εντολή

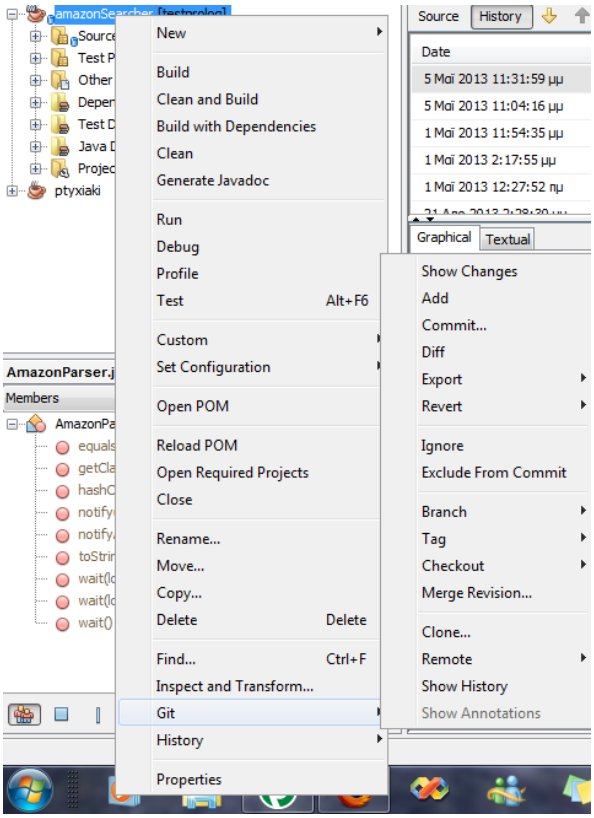
$ git help config

Οι εντολές αυτές είναι χρήσιμες γιατί μπορείτε να έχετε πρόσβαση σε οπουδήποτε, ακόμα και εκτός σύνδεσης. Αν οι manpages και αυτό το βιβλίο δεν είναι αρκετά και θα πρέπει να έχετε βοήθεια από κάποιο άτομο , μπορείτε να δοκιμάσετε το #git ή #github που είναι κανάλι στο διακομιστή Freenode IRC

(irc.freenode.net). Αυτά τα κανάλια συχνά είναι γεμάτα με εκατοντάδες ανθρώπους που είναι πολύ καλοί γνώστες του Git και συχνά είναι πρόθυμοι να βοηθήσουν.

Μετά το τέλος της ανάλυσης του κεφαλαίου για το Git σαν τεχνολογία την εξέλιξης της και τη χρησιμότητα της ακολουθεί μια άλλη τεχνολογία το Maven .

9.7 ΟΙ ΒΑΣΙΚΟΤΕΡΕΣ ΕΝΤΟΛΕΣ ΤΟΥ GIT



Εδώ βλέπουμε τις εντολές που μας δίνει το git .Οι κυριότερες από αυτές είναι η push, η pull,η tag,η clone, και η commit).

Η Push είναι μία εντολή που χρησιμοποιείτε όταν δουλεύουμε σε μία ομάδα προγραμματιστών και θέλουμε να κοινοποιήσουμε τις αλλαγές του κώδικα μας .

Η Pull είναι μία εντολή που ενσωματώνει τις αλλαγές από την υπόλοιπη ομάδα .

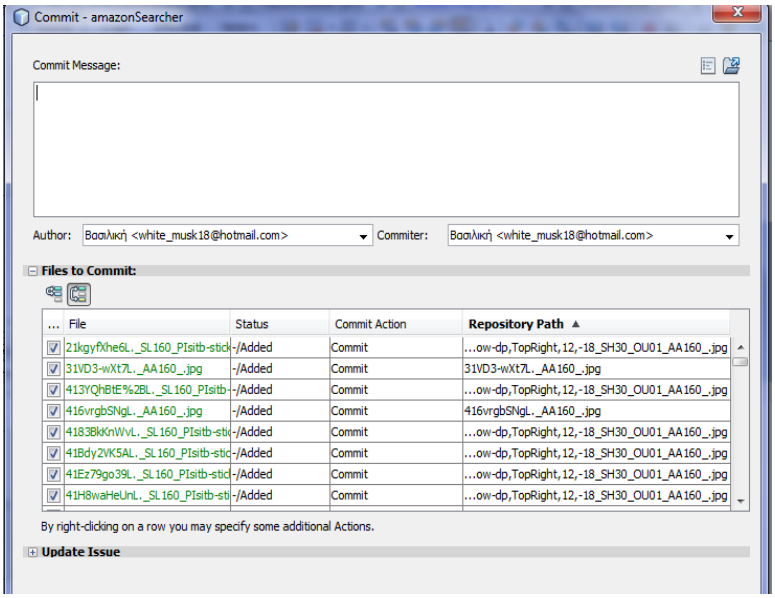
Η clone είναι η εντολή που δημιουργεί ένα κλώνο του αποθετηρίου ανεξάρτητο που τροποποιείται χωρίς να επιφέρει καμία αλλαγή το πρωτότυπο αποθετήριο.

Η Tag είναι η εντολή που δημιουργεί μια λίστα ή τη διαγράφει ή ακόμα μπορεί να ελέγχει την ετικέτα κάποιου αντικειμένου.

Και τέλος η Commit αποθηκεύει το τρέχον περιεχόμενο μαζί με ένα μήνυμα καταγραφής από τον χρήστη που περιγράφει τις αλλαγές.

Τώρα θα αναφέρουμε και από ένα παράδειγμα για την κάθε εντολή ώστε να μπορέσουμε να δείξουμε την απλότητα της λειτουργίας του git.

9.7.1 ΓΙΑ ΤΗΝ COMMIT

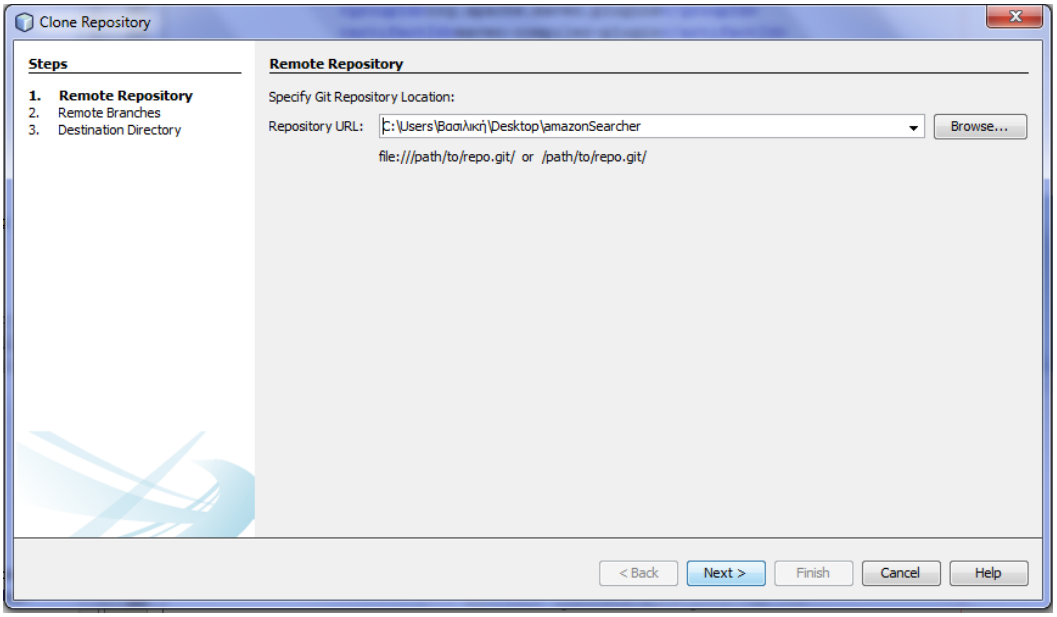


Εδώ βλέπουμε τα πεδία που συμπληρώνει ο προγραμματιστής τα στοιχειά του και το σχόλιο για την φάση που βρίσκετε η εφαρμογή.

Εικόνα 3 COMMIT

9.7.2 ΓΙΑ ΤΗΝ CLONE

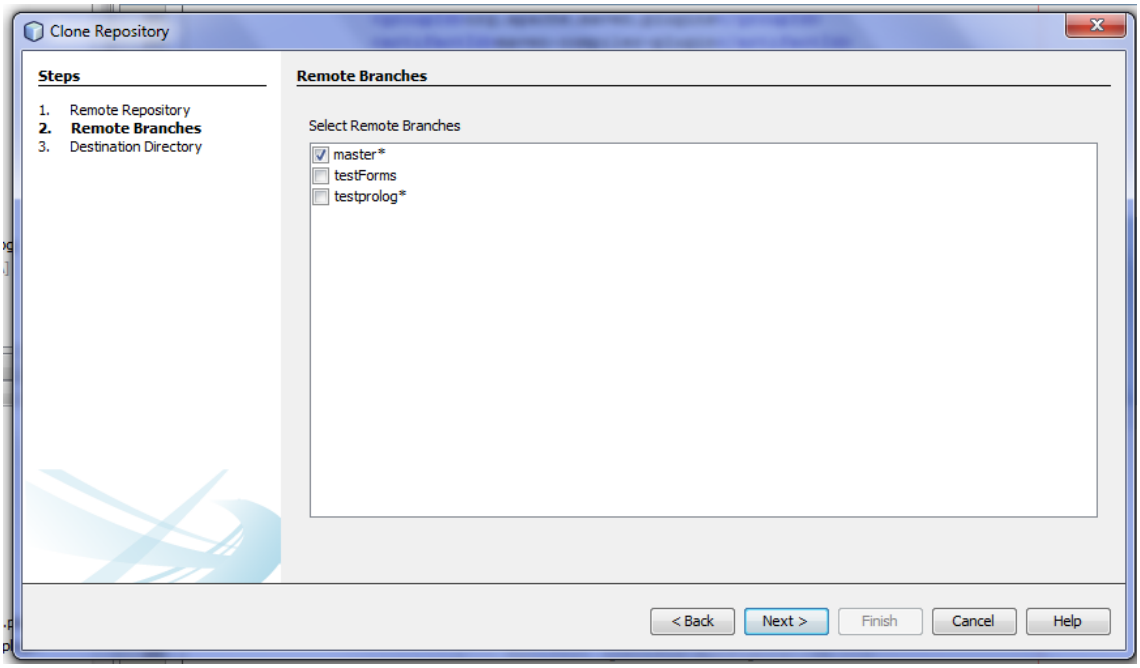
Το πρώτο βήμα…..



Εικόνα 4 REMOTE REPOSITORY

Εδώ αρχικά επιλέγουμε την διεύθυνση του αποθετηρίου που θέλουμε να κοινοποιήσουμε.

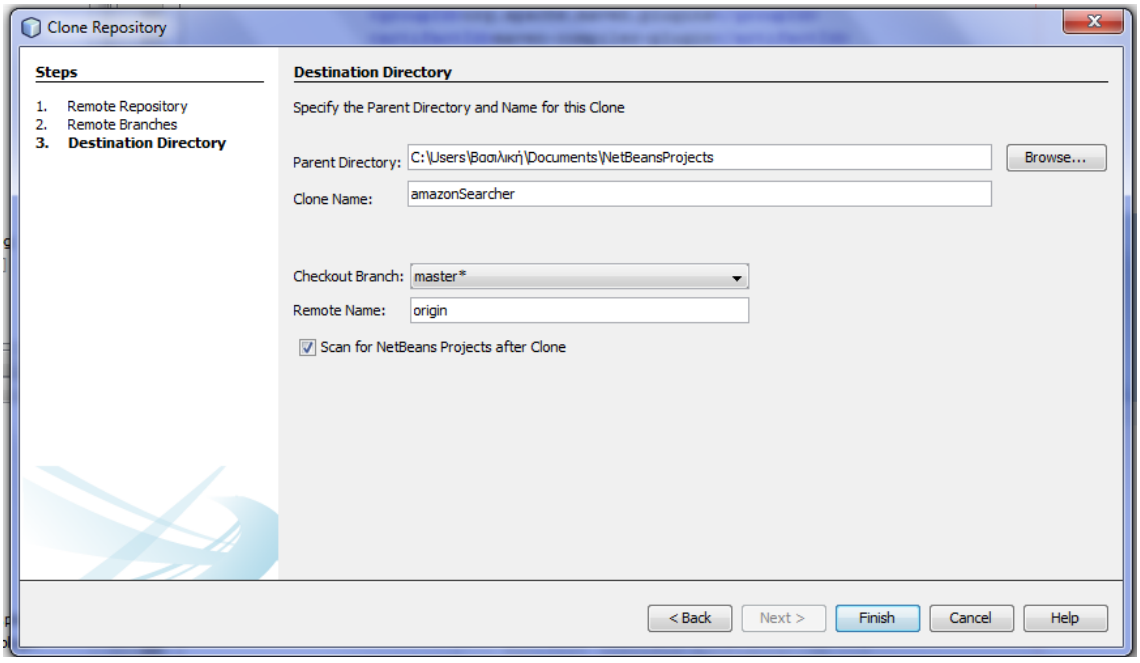
Το δεύτερο βήμα…



Εικόνα 5 REMOTE BRANCHES

Εδώ επιλέγουμε αυτά που μας ενδιαφέρουν από το αποθετήριο.

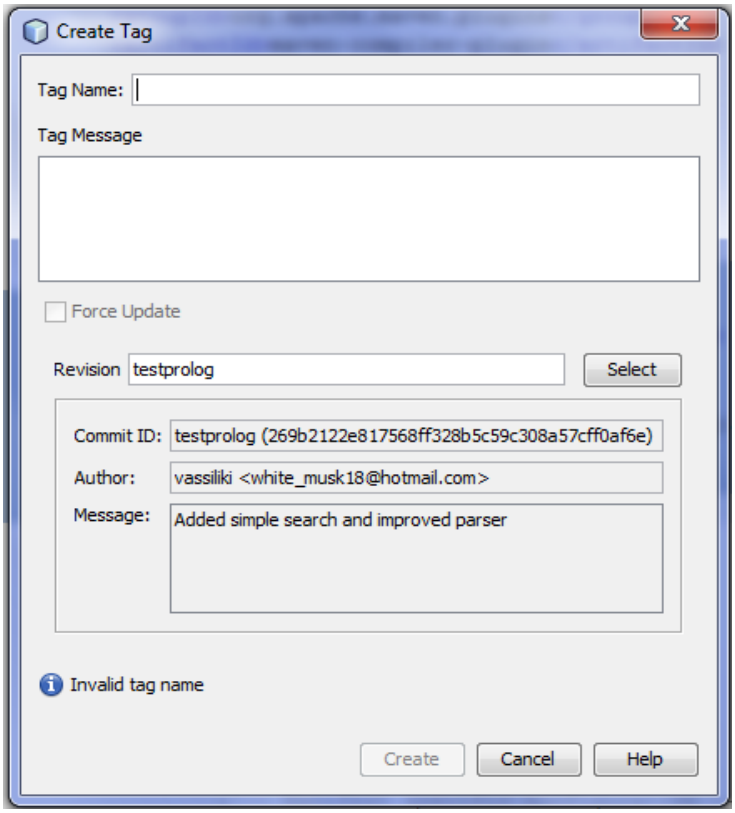
Και τρίτο βήμα…



Εικόνα 6 DESTINATION DIRECTORY

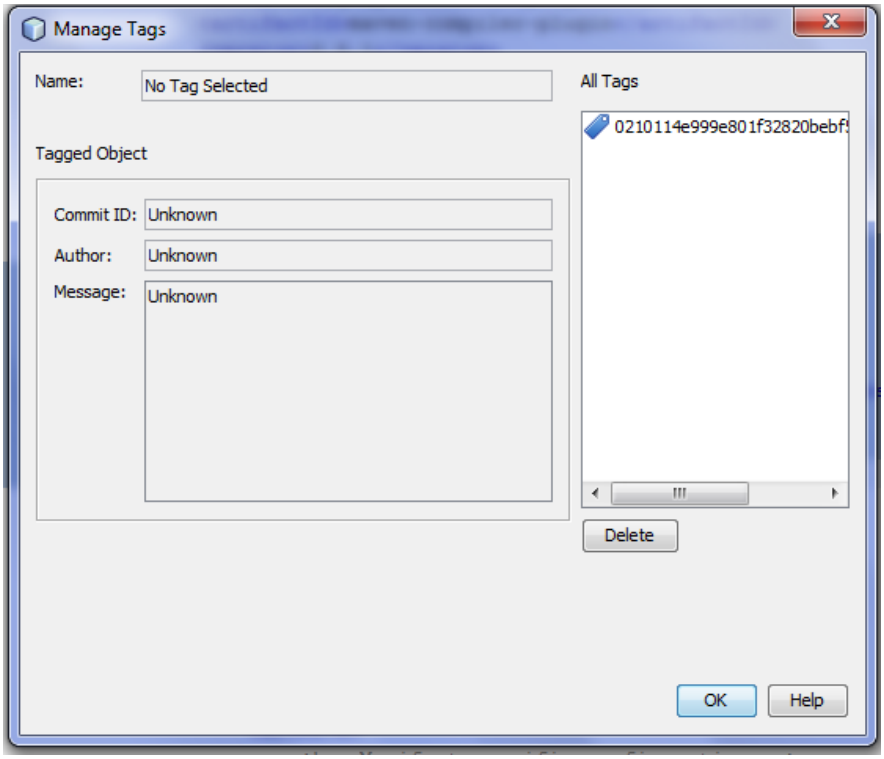
Εδώ δίνουμε που θέλουμε να αποθηκευτεί ο κλώνος και με τι όνομα.

9.7.3 ΓΙΑ ΤΗΝ ΕΝΤΟΛΗ TAG



Εδώ δημιουργούμε μια λίστα και καταγράφουμε τα χαρακτηριστικά της

Εικόνα 7 CREATE TAG



Εδώ κάνουμε διαχείριση από τις λίστες μας .

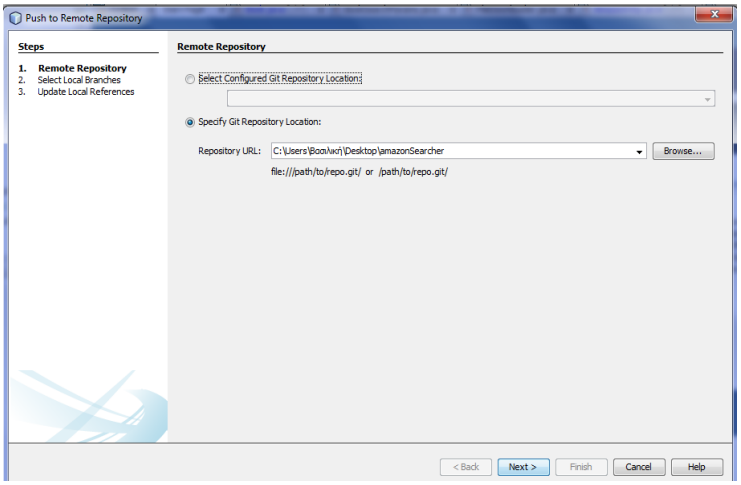
Εικόνα 8 MANAGE TAGS

9.7.4 ΓΙΑ ΤΗΝ ΕΝΤΟΛΗ PUSH

Που ουσιαστικά μας δίνει τον τρόπο το αποθετήριο μας να δηλωθεί και να χρησιμοποιηθεί από το κεντρικό αποθετήριο

Αρχικά κάνουμε δεξί click στο πακέτο της εφαρμογής μας και επιλέγουμε > Git > Remote > Push.

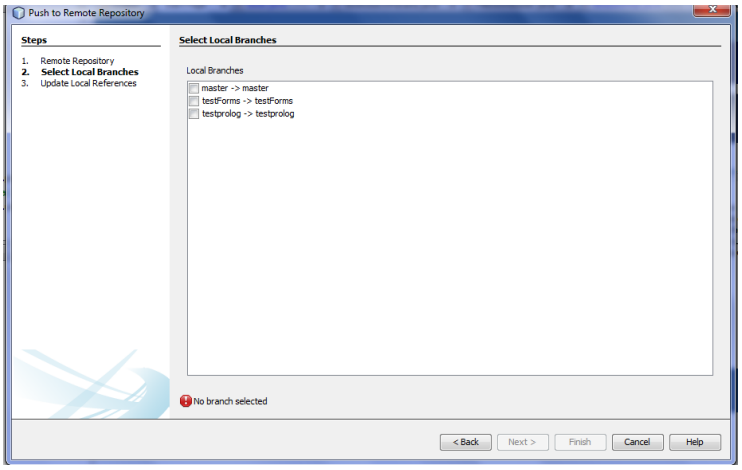
Το πρώτο βήμα ..



Εδώ επιλέγουμε από ποιο διεύθυνση θέλουμε να δημοσιεύσουμε το αποθετήριο μας.

Εικόνα 9 REMOTE REPOSITORY_PUSH

Το δεύτερο βήμα…



Σε αυτό το σημείο επιλέγουμε από ποιο επιμέρους αποθετήριο θέλουμε να δημοσιεύσουμε.

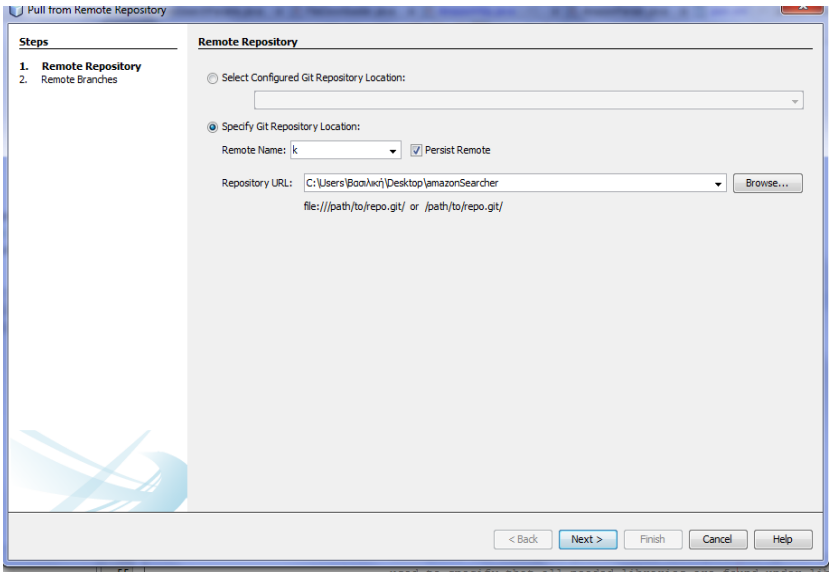
Εικόνα 10 REMOTE BRANCHES_PUSH

Τέλος πατάμε Finish

9.7.5 ΓΙΑ ΤΗΝ ΕΝΤΟΛΗ PULL

Αρχικά κάνουμε δεξί click στο πακέτο της εφαρμογής μας και επιλέγουμε > Git > Remote > Pull

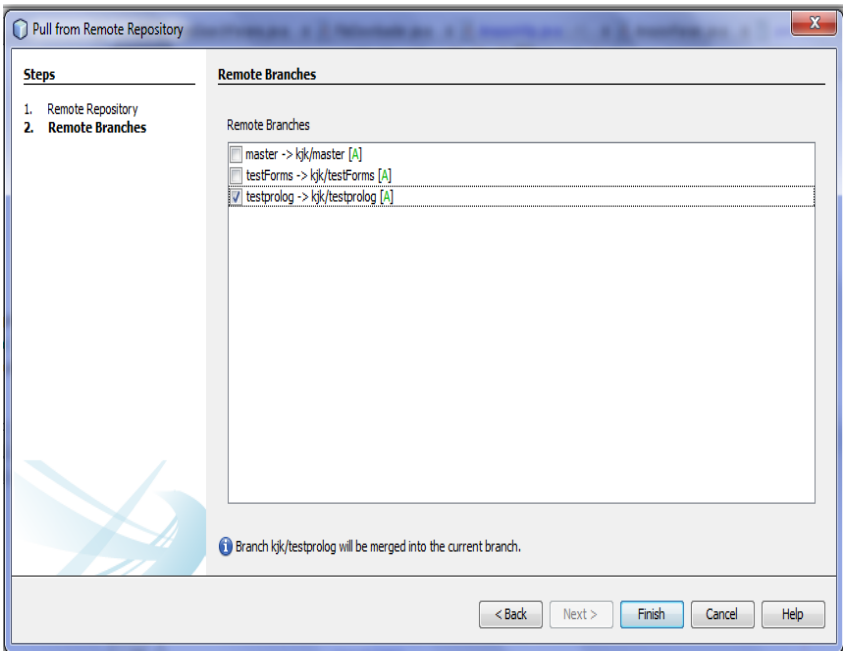
Πρώτο βήμα..



Εδώ δίνουμε το όνομα του αποθετηρίου που μας ενδιαφέρει.

Εικόνα 11 REMOTE REPOSOTORY_PULL

Βήμα δεύτερο…



Εδώ καθορίζουμε το τμήμα του αποθετηρίου που θα ενημερωθεί.

Στο Τέλος πατάμε Finish.

Εικόνα 12 REMOTE BRANCHES_PULL

10 MAVEN

Σε αυτό το κεφάλαιο αναφερόμαστε στο Maven ,κανουμε μία περιγραφή του εργαλείου τι ακριβώς είναι τι χρησιμότητα έχει , την ιστορία του πως ξεκίνησε που έφτασε, κάνουμε μια σύγκριση με το προγενέστερό του ant ,αναφέρουμε τα plugins του ,τον κύκλο ζωης του και τέλος μιλάμε για τις εξαρτίσεις του.

10.1 ΤΙ ΕΙΝΑΙ ΤΟ MAVEN;

Το **Maven** είναι ένα εργαλείο αυτοματισμού κατασκευής που χρησιμοποιείται κυρίως για Java έργα. Το Maven εξυπηρετεί έναν παρόμοιο σκοπό με το Apache Ant εργαλείο, αλλά βασίζεται σε διαφορετικές έννοιες και λειτουργεί με διαφορετικό τρόπο. Όπως Ant μπορεί επίσης να χρησιμοποιηθεί για την κατασκευή και διαχείριση έργων γραμμένων σε C # , Ruby , Scala , και σε άλλες γλώσσες. Το έργο Maven φιλοξενείται από το Apache Software Foundation , όπου ήταν στο παρελθόν μέρος του έργου Τζακάρτα .

10.2 ΤΙ ΚΑΝΕΙ ΤΟ MAVEN;

Το Maven χρησιμοποιεί ένα XML αρχείο για να περιγράψει το έργο λογισμικού που χτίζετε , τις εξαρτήσεις του σε άλλες εξωτερικές μονάδες ,τα συστατικά του, τη σειρά αναβαθμίσεων, του καταλόγους, και τα απαιτούμενα plug-ins . Έρχεται με προκαθορισμένους στόχους για την εκτέλεση ορισμένων σαφώς καθορισμένων καθηκόντων , όπως η κατάρτιση κώδικα και του packaging.

Maven εκτελεί δυναμικά λήψεις βιβλιοθήκων Java και Maven plug-ins από ένα ή περισσότερα αποθετήρια όπως το Maven 2 Repository Central, και τα αποθηκεύει σε μια τοπική μνήμη cache. Αυτή η τοπική μνήμη cache του κατεβάσει αντικείμενα μπορεί επίσης να ενημερωθεί με έργα της που δημιουργούνται από local projects. Επίσης μπορούν να ενημερώνονται από δημόσια αποθετήρια(Public repositories).

Το Maven είναι χτισμένο με μια plugin-based αρχιτεκτονική που του επιτρέπει να κάνει χρήση οποιασδήποτε εφαρμογής ελεγχθεί μέσω standard input. Θεωρητικά, αυτό θα επιτρέπει σε κανέναν να γράψει plugins για τη διασύνδεση με την κατασκευή εργαλείων (μεταγλωττιστές, εργαλεία δοκιμής μονάδα, κλπ) για οποιαδήποτε άλλη γλώσσα. Στην πραγματικότητα, η υποστήριξη και η χρήση για άλλες γλώσσες εκτός από Java ήταν ελάχιστη.

10.3 ΙΣΤΟΡΙΑ

Το Maven, δημιουργήθηκε από Jason van Zyl Sonatype , είχε ξεκίνησει ως υποέργο του Apache Turbine το 2002. Το 2003, ψηφίστηκε και έγινε αποδεκτό ως κορυφαίο επίπεδο Apache Software Foundation έργου. Τον Ιούλιο του 2004, η έκδοση του κρισίμου πρώτου οροσήμου του Maven ήταν το v1.0. Maven 2 μετά αναβαθμίστηκε σε v2.0 τον Οκτώβριο του 2005 μετά από περίπου έξι μήνες σε beta κύκλους. Το Maven 3.0 κυκλοφόρησε τον Οκτώβριο του 2010 είναι ως επί το πλείστον συμβατό με Maven 2.

10.4 MAVEN ΣΕ ΣΥΓΚΡΙΣΗ ΜΕ ΤΟ ANT

Η θεμελιώδης διαφορά μεταξύ Maven και του Ant είναι ότι ο σχεδιασμός του Maven αφορά όλα τα έργα που έχουν μια συγκεκριμένη δομή και ένα σύνολο υποστηριζόμενων εργασίων σε ροές (π.χ., να πάρει τους πόρους από τον έλεγχο πηγή, την κατάρτιση του έργου, τον έλεγχο μονάδας, κ.λπ.). Ενώ τα περισσότερα έργα λογισμικού σε ισχύ με την υποστήριξη αυτών των επιχειρημάτων και στην πραγματικότητα έχουν μια καλά καθορισμένη δομή, το Maven προϋποθέτει ότι αυτή η δομή και οι λεπτομέρειες εφαρμογής της λειτουργίας πρέπει να οριστεί στο αρχείο POM. Έτσι, το Maven βασίζεται σε μια σύμβαση σχετικά με το πώς θα καθορίσει τα έργα και τον κατάλογο στις ροές εργασιών που υποστηρίζονται γενικά σε όλα τα έργα.

Αυτός ο περιορισμός του σχεδίου μοιάζει με τον τρόπο που ένα IDE χειρίζεται ένα έργο, και παρέχει πολλά οφέλη, όπως είναι ο ορισμός συνοπτικά του έργου, και τη δυνατότητα αυτόματης ολοκλήρωσης του έργου του Maven με άλλα εργαλεία ανάπτυξης, όπως IDEs, να χτίσουν servers, κλπ.

Αλλά ένα μειονέκτημα αυτής της προσέγγισης είναι ότι το Maven απαιτεί από τον χρήστη να κατανοήσει πρώτα τι είναι ένα έργο από το την πλευρά του Maven , και πώς το Maven λειτουργεί με τα έργα, διότι αυτό που συμβαίνει όταν κάποιος εκτελεί μια φάση στο Maven δεν είναι άμεσα εμφανές μόνο από την εξέταση του φάκελου του έργου του Maven. Σε πολλές περιπτώσεις, η απαιτούμενη δομή είναι επίσης ένα σημαντικό εμπόδιο στο θέμα της μετάβασης

σε ένα ώριμο έργο για το Maven, επειδή είναι συνήθως δύσκολο να προσαρμοστούν από άλλες προσεγγίσεις.

Το Αντ λειτουργεί με XML σενάρια κατασκευής ορίζεται σε ένα ή περισσότερα αρχεία. Επεξεργάζεται στόχους από αυτά τα αρχεία και κάθε στόχος εκτελεί καθήκοντα. Κάθε εργασία εκτελεί μια τεχνική λειτουργία, όπως η λειτουργία ενός compiler ή αντιγραφή αρχείων γύρω της . Οι στόχοι εκτελούνται κατά κύριο λόγο με τη σειρά που ορίζεται από την εξάρτησή τους σε άλλους στόχους. Έτσι, το Ant είναι ένα εργαλείο που στοχεύει σε αλυσίδες που συσχετίζονται και η εκτέλεση τους βασίζεται σε αλληλεξαρτήσεις και άλλους Boolean όρους.

Τα οφέλη που παρέχονται από το Ant είναι επίσης πολλά. Έχει μία γλώσσα XML που βελτιστοποιείτε για να γίνετε σαφέστερος ορισμός του τι κάνει κάθε έργο και από τι εξαρτάται. Επίσης, όλες οι πληροφορίες σχετικά με το τι θα πρέπει να εκτελείτε από ένα Ant target μπορεί να βρεθεί στο Ant script .

Ένας προγραμματιστής για να εξοικειωθεί με το Ant θα πρέπει να είναι σε θέση να προσδιορίσει τι είναι ένα απλό σενάριο Ant και τι κάνει ακριβώς με την εξέταση του σεναρίου. Αυτό συνήθως δεν είναι αληθές για το Maven.

Ωστόσο, ακόμη και ένας έμπειρος προγραμματιστής που είναι νέος σε ένα έργο με τη χρήση Ant δεν μπορεί να συμπεράνει ποια είναι η μεγαλύτερη δομή του επίπεδο ενός script Ant είναι και τι κάνει χωρίς να εξετάζει το σενάριο λεπτομερώς. Ανάλογα με την πολυπλοκότητα του σεναρίου, αυτό μπορεί γρήγορα να γίνει μια τεράστια πρόκληση. Με το Maven, ένας προγραμματιστής ο οποίος στο παρελθόν εργάστηκε με άλλα έργα Maven μπορεί να εξετάσει τάχιστα τη δομή ενός έργου μόλις το δει και κάνει την εκτέλεση των κατ 'αποκοπή εργασιών Maven- work-flows από αυτό, ενώ ήδη ξέρει τι να περιμένει ως αποτέλεσμα.

Είναι δυνατή η χρήση των Ant scripts που έχουν οριστεί και που συμπεριφέρονται με ενιαίο τρόπο για όλα τα έργα σε μια ομάδα εργασίας ή ενός οργανισμού. Ωστόσο, όταν ο αριθμός και η πολυπλοκότητα των σχεδίων αυξάνεται, είναι επίσης πολύ εύκολο να απομακρυνθούν από την αρχικώς επιθυμητή ομοιομορφία. Με το Maven αυτό είναι μικρότερο πρόβλημα, επειδή

το εργαλείο επιβάλλει πάντα ένα συγκεκριμένο τρόπο αντιμετώπισης των πραγμάτων.

Σημειώστε ότι είναι επίσης δυνατόν να επεκταθεί και να ρυθμίσετε το Maven κατά τρόπο που αποκλίνει από το αρχικές ρυθμίσεις του Maven . Αυτό είναι ιδιαίτερα σύνηθες για το Maven 2 και νεότερες εκδόσεις, όπως Mojos ή πιο επίσημα, plugins και προσαρμοσμένες δομές καταλόγου του έργου.

10.5 ΕΝΝΟΙΕΣ

10.5.1 ΕΡΓΟ OBJECT MODEL

Ένα μοντέλο αντικειμένου του έργου (POM) παρέχει όλες τις ρυθμίσεις για ένα μόνο σχέδιο. Γενική ρύθμιση καλύπτει το όνομα του έργου, τον ιδιοκτήτη του και τις εξαρτήσεις του σε άλλα έργα. Κάποιος μπορεί επίσης να ρυθμίσει επιμέρους φάσεις της διαδικασίας κατασκευής, τα οποία εφαρμόζονται ως plugins . Για παράδειγμα, μπορεί κανείς να διαμορφώσει τον compiler-plugin και να χρησιμοποιήσει την Java έκδοση 1.5 για την κατάρτιση, ή να καθορίσει το πακέτο του έργου, ακόμη και αν κάποια δοκιμή αποτύχει.

Τα μεγαλύτερα έργα θα πρέπει να χωρίζεται σε διάφορες ενότητες, ή υποέργα, καθένα με το δικό του POM. Κάποιος μπορεί να γράψει στη συνέχεια μια ρίζα POM, μέσω του οποίου μπορεί κανείς να συγκεντρώσει όλες τις ενότητες με μία μόνο εντολή. Το Pon μπορεί να κληρονομήσει επίσης την διαμόρφωση από άλλα pon . Όλα τα pon κληρονομούν από την Super POM από προεπιλογή. Η Σούπερ POM παρέχει προεπιλεγμένες ρυθμίσεις, όπως κατάλογοι προεπιλεγμένες πηγες, plugins προεπιλογής, και ούτω καθεξής.

10.5.2 PLUGINS

Τα περισσότερα στη λειτουργικότητα των Maven είναι τα plugins . Ένα plugin παρέχει ένα σύνολο στόχων που μπορούν να εκτελεστούν χρησιμοποιώντας την ακόλουθη σύνταξη:

mvn [plugin-name]: [στόχος-name]

Για παράδειγμα, ένα πρόγραμμα Java μπορεί να μεταγλωττίζεται με τη μεταγλώττιση του compiler-plugin του

εκτελώντας mvn compiler:compile .

Υπάρχουν Maven plugins για την κατασκευή, τον έλεγχο, την διαχείριση και τον έλεγχο της πηγής, τρέχουν σε ένα web server, δημιουργώντας σε Eclipse αρχεία του έργου, και πολλά άλλα . Τα Plugins εισάγονται και ρυθμίζονται σε ένα <plugins>-τμήμα ενός pom.xml αρχείου. Μερικά βασικά plugins περιλαμβάνονται σε κάθε έργο από προεπιλογή, και έχουν λογικές προεπιλεγμένες ρυθμίσεις.

Ωστόσο, θα ήταν άβολο αν η κατά την δημιουργία και το χτίσιμο της εφαρμογής δεν γινόταν έλεγχος στο πακέτο του έργου του λογισμικού που απαιτείται για τη λειτουργία του κάθε αντίστοιχου στόχου με μη αυτόματο τρόπο:

mvn compiler:compile

mvn surefire:test

mvn jar:jar

Plug-ins είναι ο πρωταρχικός τρόπος για την επέκταση Maven. Η ανάπτυξη ενός Maven plug-in μπορεί να γίνει με την επέκταση της κατηγορίας org.apache.maven.plugin.AbstractMojo.

10.5.3 ΚΥΚΛΟΣ ΖΩΗΣ

Ο κύκλου ζωής είναι μια λίστα από ονομαστικές φάσεις που θα προηγηθούν ώστε να υλοποιηθεί το λογισμικό . Ένα από τα πρότυπα κύκλου ζωής του Maven είναι ο προεπιλεγμένος *κύκλος ζωής,* ο οποίος περιλαμβάνει τις ακόλουθες φάσεις, με αυτή τη σειρά:

1. process-resources

2. compile

3. process-test-resources

4. test-compile

5. test

6. package

7. install

8. deploy

10.5.4 ΕΞΑΡΤΗΣΕΙΣ

Το Maven είναι οργανωμένο γύρω από ένα σύστημα συντεταγμένων αναγνώρισης μεμονωμένων αντικειμένων, όπως βιβλιοθήκες λογισμικού ή ενότητες. Το παράδειγμα POM παραπάνω έχει ως αναφορές τις JUnit συντεταγμένες ως άμεση εξάρτηση του έργου. Ένα έργο που χρειάζεται, ας πούμε, η Hibernate βιβλιοθήκη έχει απλά να κηρύξει έργο αδρανοποίησης που είναι οι συντεταγμένες του σε POM . Το Maven θα κατεβάσει αυτόματα την εξάρτηση και τις εξαρτήσεις της αδρανοποίησης που χρειάζεται η ίδια (που ονομάζονται μεταβατικές εξαρτήσεις) και να τα αποθηκεύσει σε τοπικό repository του χρήστη. Το Maven 2 Κεντρικό Αποθετήριο χρησιμοποιείται από προεπιλογή για την αναζήτηση στις βιβλιοθήκες, αλλά μπορεί κανείς να ρυθμίσει τα αποθετήρια που πρέπει να χρησιμοποιούνται (π.χ., η εταιρεία ιδιωτικού αποθετηρίου) εντός του POM.

Υπάρχουν μηχανές αναζήτησης όπως το Maven Central η οποία μπορεί να χρησιμοποιηθεί για να ανακαλύψει συντεταγμένες για διάφορες open-source βιβλιοθήκες και πλαίσια.

Έργα που αναπτύσσονται σε ένα μοναδικό μηχάνημα μπορούν να εξαρτώνται από τα άλλα μέσω του τοπικού repository. Το τοπικό repository είναι μια απλή δομή του φακέλου που δρα τόσο ως cache μνήμη για λήψη εξαρτήσεις και ως κεντρικός χώρος αποθήκευσης για τοπικά χτισμένα αντικείμενα. Στο Maven η εντολή mvn install χτίζει ένα έργο και το τοποθετεί στα εκτελέσιμα του στο τοπικό αποθετήριο. Στη συνέχεια, άλλα έργα μπορούν να χρησιμοποιήθούν από αυτό το έργο καθορίζοντας τις συντεταγμένες του σε Pom τους.

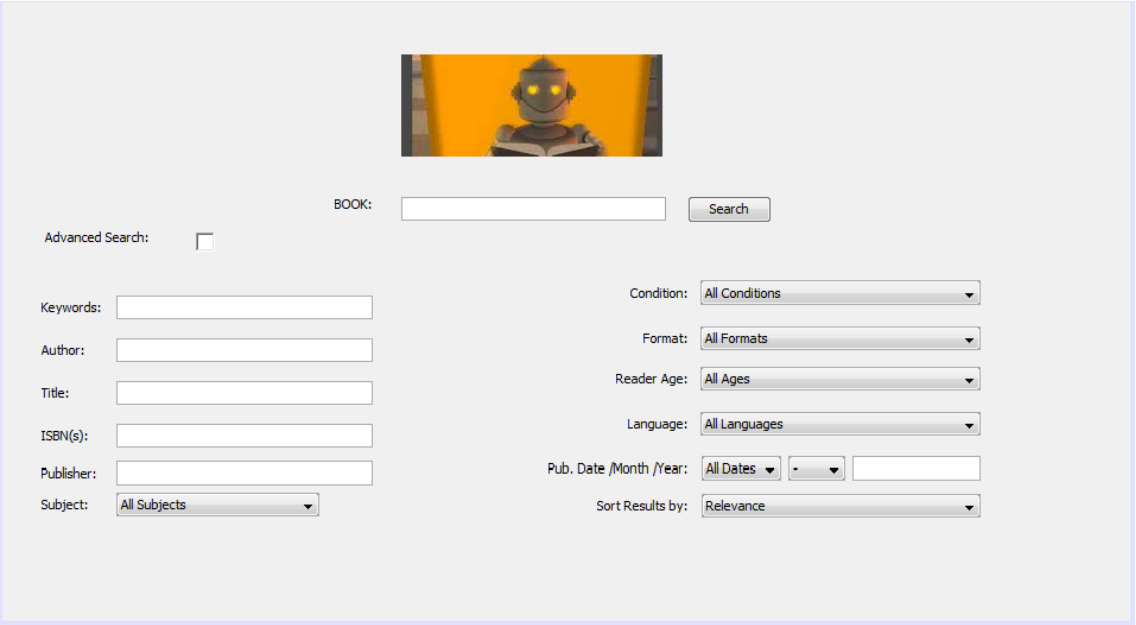
Μέτα την ανάπτυξη όλων αυτών στο επόμενο κεφάλαιο θα μιλήσουμε για το την λειτουργία και την χρήση της εφαρμογής, πως υλοποιήθηκε και πως βοήθησαν οι παραπάνω γλώσσες και εργαλεία τη δημιουργία της.

11 Η ΕΦΑΡΜΟΓΗ

Η εφαρμογή έχει σας αντικείμενο την αναζήτηση επιστημονικών εγγράφων από τη βάση δεδομένων της amazon.com , δίνοντας στο χρήστη τη δυνατότητα να κάνει μία αναζήτηση εγγράφων βάσει πολλών χαρακτηριστικών μερικά από τα χαρακτηριστικά είναι ο τίτλος του βιβλίου , ο συγγραφέας, το ηλικιακό κοινό που απευθύνετε , εκδότης,ISBN(s) , για ποιες ηλικίες απευθύνετε , σε ποια γλώσσα , ποια είναι η ημερομηνία της έκδοσης του, καθώς και θα μπορεί ο χρήστης από την αναζήτηση να λαμβάνει τα αποτελέσματα ταξινομημένα είτε βάση τιμής είτε με κριτήριο το ποιό είναι ποιο δημοφιλές ή ακόμα το πόσο καινούριο είναι .

11.1 INTERFACE

Το interface της εφαρμογής είναι ολοκληρωμένο με java swing είναι ουσιαστικά μία μηχανή αναζήτησης με διάφορα textboxes που επιτρέπουν στο χρήστη να περάσει τα κριτήρια για την αναζήτηση του καθώς και διάφορα compo boxes που πάλι μέσα από αυτά ο χρήστης θέτει τα κριτήρια του για την αναζήτηση του. Όλα αυτά θα τα περιγράψουμε αναλυτικά παρακάτω με την παράθεση και της σχετικής εικόνας .

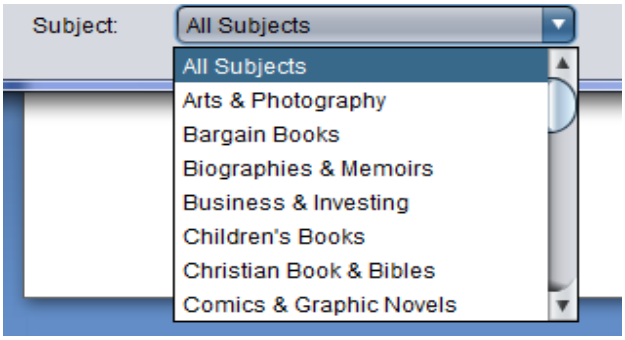


Εικόνα 13 INTERFACE

Στην παραπάνω εικόνα φαίνετε η μορφή του Interface. Αρχικά δίνουμε στον χρήστη τη δυνατότητα της απλής αναζήτησης θέτοντας κάποια χαρακτιριστικά στο textbox με το όνομα book.Διαφορετικά δίνει την δυνατότητα στον χρήστη να επιλέξει την σύνθετη αναζήτηση τσεκάροντας το check box .

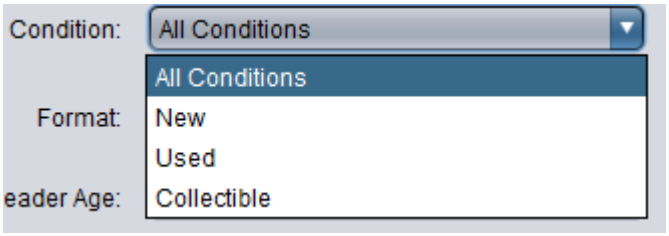
Στη συνέχεια ο χρήστης θέτει στα πεδία της αναζήτησης του, στο text box με όνομα keywords δίνει κάποιες λέξεις κλειδιά ,στο text box με όνομα author δίνει το όνομα του συγγραφέα ,στο text box με όνομα title δίνει τον τίτλο του βιβλίου που τον ενδιαφέρει , στο text box με όνομα ISBN(s) δίνει το ISBN(s) του βιβλίου που ψάχνει σε περίπτωση που το ξέρει , στο text box με όνομα publisher δίνει τον εκδότη του βιβλίου που ψάχνει σε περίπτωση που το ξέρει .

Μετά ακολουθούν compo boxes ώστε ο χρήσης να μπορέσει να επιλέξει αυτά που θέλει , τα οποία θα τα δούμε σε ανάπτυξη στις παρακάτω εικόνες και θα γίνει και η επεξήγηση τους.



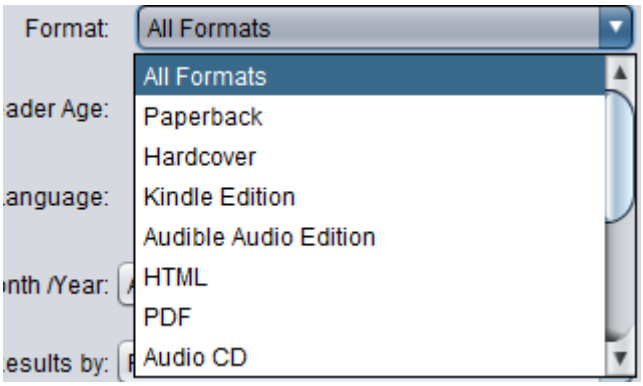
Εικόνα 14 SUBJECT

Με το compo box που ονομάζεται Subject ο χρήστης μπορεί να επιλέξει ποιο είναι το θέμα που τον ενδιαφέρει μέσα από μία μεγάλη ποικιλία όπως βλέπουμε στην εικόνα.



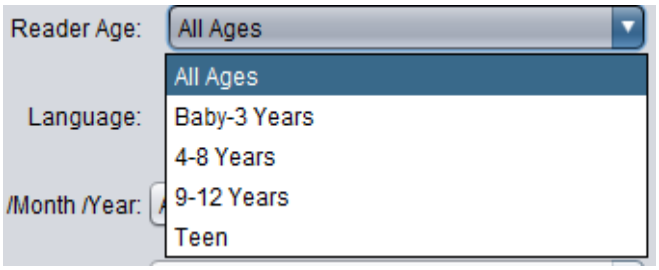
Εικόνα 15 CONDITION

Με το compo box που ονομάζεται Condition ο χρήστης μπορεί να διαλέξει σε τι κατάσταση θέλει να είναι το σύγγραμμα το οποίο τον ενδιαφέρει.



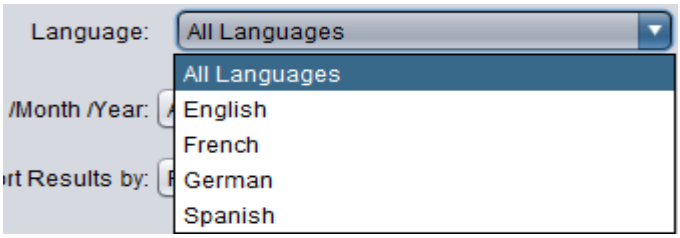
Εικόνα 16 FORMAT

Με το compo box Format ο χρήστης μπορεί να επιλέξει τη μορφή που θέλει να έχει το έγγραφο .



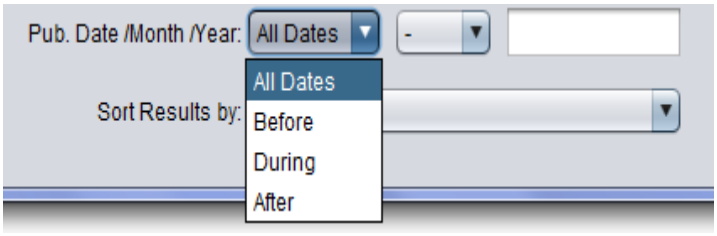
Εικόνα 17 READER AGE

Σε αυτό το σημείο ο χρήστης με το compo box «Reader Age:» βρίσκει την ηλικιακή κατηγορία που τον ενδιαφέρει.



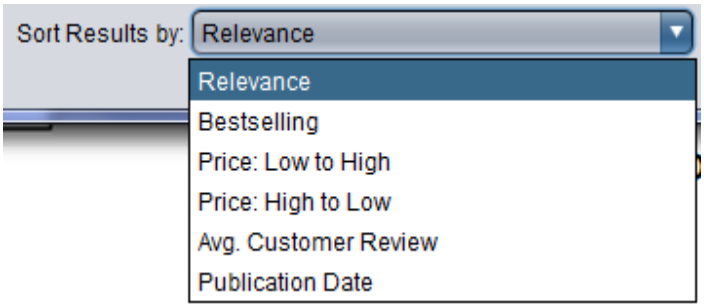
Εικόνα 18 LAGUAGE

Με το compo box «language:» στον χρήστη παρέχετε η δυνατότητα να επιλέξει την γλώσσα που τον ενδιαφέρει.



Εικόνα 19 PUB.DATE/MONTH/YEAR

Εδώ ο χρήστης επιλέγει την ημερομηνία έκδοσης δίνει μήνα και χρόνο και επιλέγει πριν, μετά ή και κατά τη διάρκεια.



Εικόνα 20 SORT RESULTS BY

Εδώ ο χρήστης στο compo box «Sort Results by:» επιλέγει με βάσει τι, θέλει να γίνει η ταξινόμηση.

Αφού ολοκληρώσαμε την επεξήγηση του interface τώρα θα περιγράψουμε πως έγινε η ενοποίηση της java με την prolog.

11.2 ΕΝΣΩΜΑΤΩΣΗ JAVA ΚΑΙ PROLOG

Στην συγκεκριμένη εργασία χρειαζόμασταν την χρήση δυο γλωσσών προγραμματισμού μέσα στο ίδιο project , οπότε για να έχουμε και τις δύο σε συνεργασία έπρεπε να ακολουθηθούν συγκεκριμένα βήματα.

Αρχικά κάναμε εγκατάσταση του το Netbeans IDE το οποίο εμπεριέχει τον maven builder που μας επιτρέπει να δουλέψουμε και να ενσωματώνει μόνο του τα .jar αρχεία . Στη συνέχεια βρήσκουμε το path του maven το οποίο για κάθε έναν μπορεί να είναι διαφορετικό για εμάς είναι το : C:\Program Files\NetBeans 7.3\java\maven\bin\mvn . Μετά από αυτό πρέπει να γίνει εγκατάσταση της SWI Prolog και βρίσκουμε το path του jpl.jar. Για εμάς είναι το C:\Program

Files\swipl\lib\jpl.jar. Μετά προσθέτουμε το jpl.jar στο .m2 repository του υπολογιστή μας. Μέσω μία εντολής η οποία εκτελείτε στο command line. Τέλος πρέπει να προσθέτουμε στο pom.xml του project μας κάποιες εντολές για την jpl βιβλιοθήκη.

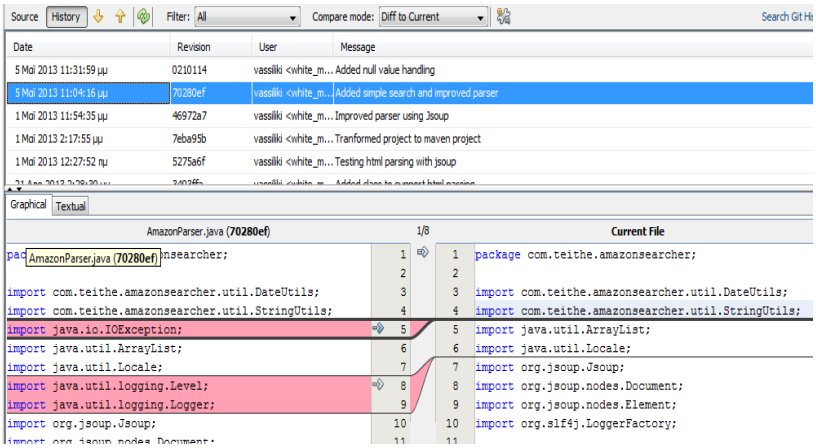
Με αυτόν τον τρόπο έγινε η ενοποίηση των δύο γλωσσών και μας επιτράπηκε να διαχειριστούμε κάποια στοιχεία με τη μία γλώσσα και κάποια άλλα με την άλλη.

Μετά από αυτό θα περιγράψουμε την χρήση του git στην εφαρμογή και θα δείξουμε και την λειτουργικότητα κάποιων εντολών του.

11.3 ΤΟ GIT ΚΑΙ Η ΧΡΗΣΗ ΤΟΥ ΣΤΗΝ ΕΦΑΡΜΟΓΗ

Η εγκατάστασή του είναι εξαιρετικά απλή ουσιαστικά το μόνο που χρειάζεται είναι να εκτελέσουμε την εντολή : msysgit.github.io σε μια γραμμή εντολών ώστε να εγκατασταθεί το git στα windows.

Το Git δίνει την δυνατότητα σε εμάς να κρατάμε ένα αρχείο με τις τροποποιήσεις που κάναμε στον κώδικα ώστε να μπορούμε να επαναφέρουμε κάτι που από λάθος κάναμε. Εδώ παραθέτω μία εικόνα της εφαρμογής ώστε να δει κανείς πόσο εύκολα κάποιος διακρίνει τις τροποποιήσεις.

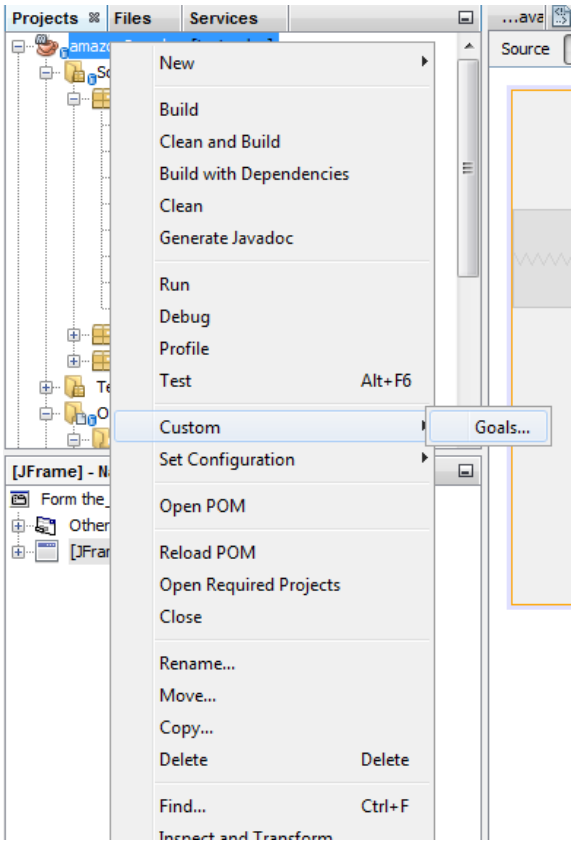


Εικόνα 21 GIT_HISTORY

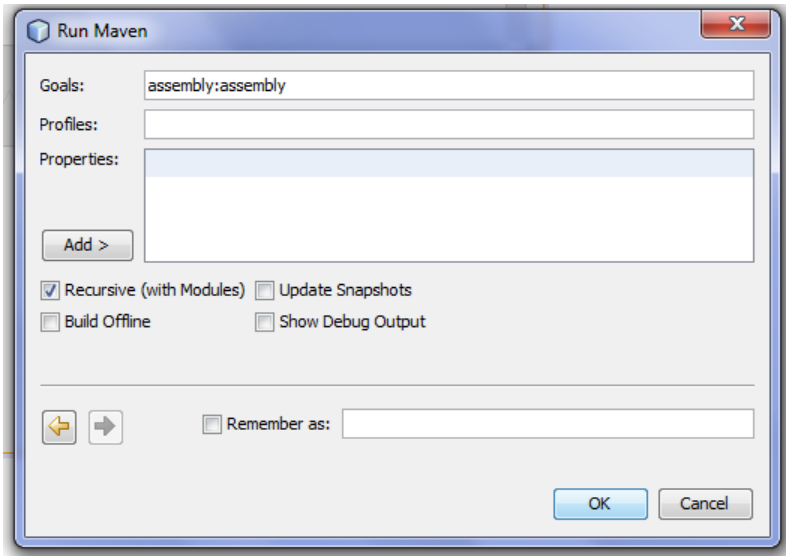
Σε αυτή την εικόνα μπορούμε να δούμε πότε έγινε η τροποποίηση , από ποιόν έγινε ,δίνετε ένας κωδικός κάτι σας έκδοση πού είναι μοναδικός και μπορεί να ξέρει ο προγραμματιστής ότι κανένας άλλος δεν τροποποίησε κάτι εξ ονόματός του.

11.4 ΤΙ ΠΡΟΣΦΕΡΕΙ ΤΟ MAVEN ΣΤΟ PROJECT

Με το maven μπορούμε να συλλέξουμε τα jar αρχεία και να τα κάνουμε build Από το net beans πάμε πάνω στο package κάνουμε δεξί click επιλέγουμε custom μετά επιλέγουμε goals.



Εικόνα 22 CUSTOM_GOALS



Μετά εμφανίζεται το παράθυρο της εικόνας

Εικόνα 23 RUN MAVEN

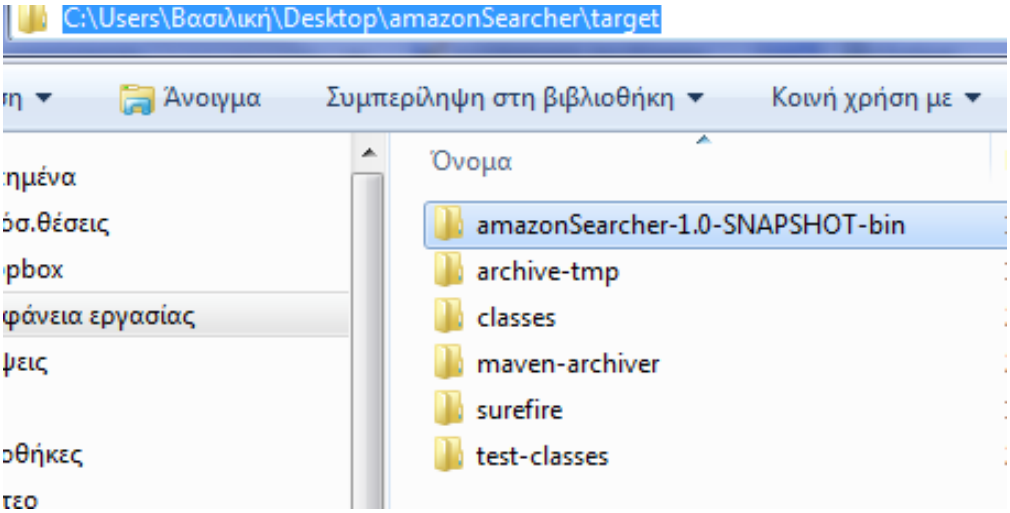
γράφουμε:Assembly:assembly

Πατάμε ok.

Εικόνα 24 BUILD

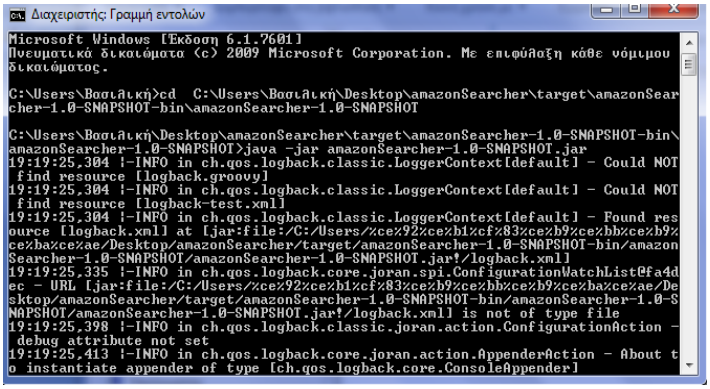
Οπότε και ξεκινάει η διαδικασία ,μόλις το maven ολοκληρώσει τότε εμφανίζετε στην οθόνη build success. Ουσιαστικά τι στιγμή που πατάμε το ok .Εκτελίτε το απόσπασμα από το POM(Θα αναφερθεί στο παράρτημα β' το απόσπασμα του κώδικα).

Οπότε έχει δημιουργηθεί εκτελέσιμο αρχείο και βρίσκετε στον φάκελο του project μας. Είναι σε συμπιεσμένη μορφή και χρειάζεται αποσυμπίεση για να χρησιμοποιηθεί.



Εικόνα 25 DIRECTORY

Εδώ φαίνετε ακριβός που βρίσκετε το αρχείο που μας ενδιαφέρει. Ανάλογα με το που έχουμε αποθηκεύσει το project μας .



Εικόνα 26 COMMAND LINE

Μετά από τη γραμμή εντολών μπαίνουμε στον φάκελο που μας ενδιαφέρει και εκτελούμε την παρακάτω εντολή:

java -jar amazonSearcher-1.0-SNAPSHOT.jar

Μετά εμφανίζεται κανονικά η εφαρμογή μας

12 ΣΥΜΠΕΡΑΣΜΑ

Μετά την δημιουργία του project συμπεραίνουμε ότι η java σα γλώσσα δίνει μεγάλη ελευθερία στον προγραμματιστή να κάνει πραγματικά ότι μπορεί να φανταστεί, παράλληλα το αποτέλεσμα είναι εύχρηστο για τον χρήστη δίνει με μεγάλη ταχύτητα το αποτέλεσμα. Από την άλλη η prolog σα γλώσσα είναι ποιο απόμακρη έχει περισσότερο αφηρημένες έννοιες ,το καλό της είναι πως λόγο των αφηρημένων εννοιών δίνει τη δυνατότητα να χρησιμοποιηθεί από περισσότερες περιπτώσεις.

Πάντως πιστεύω πώς το project είναι λειτουργικό . Δίνει στον χρήστη την άνεση χωρίς κάποιο browser να βρει αυτό που ψάχνει και να το διαχειριστεί αναλόγως.

13 ΑΝΑΦΟΡΕΣ

1)Εισαγωγή στη Java

http://www.it.uom.gr/project/java2001/k11.html

2) Guide to Java Resources

http://www.halcyon.com/mclain/java.htm

3) ΕΙΣΑΓΩΓΗ ΣΤΗ ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ JAVA

http://www.ebooks4greeks.gr/dowloads/Pliroforiki/Glosses.program./Java__Downloaded_from_eBooks4Greeks.gr.pdf

4) What are Applications and Applets

http://www.apl.jhu.edu/~hall/java/beginner/apps.html

5) Wikipedia Prolog

http://en.wikipedia.org/wiki/Prolog

6) Java API overview

http://www.swi-prolog.org/packages/jpl/java_api/high-level_interface.html

7) Git User Guide

https://netbeans.org/kb/docs/ide/git.html

8) Search Engine

http://www.seofuture.gr/page.asp?sc_id=14

9) Apache Maven Project

http://maven.apache.org/background/history-of-maven.html

10) Java Development with NetBeans and Maven

http://charleech.wordpress.com/2010/11/10/java-development-with-netbeans-and-maven-part-1/

11) About Git

http://git-scm.com/about

12)Git Documentation

http://git-scm.com/doc

13) Calling Prolog Rules from Java

http://sujitpal.blogspot.gr/2009/09/calling-prolog-rules-from-java.html

14) Using image resources in a jar using maven

http://stackoverflow.com/questions/9714028/using-image-resources-in-a-jar-using-maven

15) Java - Maven - Setting Icon Images in a Form

http://stackoverflow.com/questions/6767587/java-maven-setting-icon-images-in-a-form

16) Why does my icon handling code throw a NullPointerException?

http://stackoverflow.com/questions/2019914/why-does-my-icon-handling-code-throw-a-nullpointerexception

14 ΠΑΡΑΡΤΗΜΑ Α’

Για την προσθήκη του jpl.jar στο .m2 repository του υπολογιστή μας η εντολή είναι :

"C:\Program Files\NetBeans 7.3\java\maven\bin\mvn" install:install-file D-file="C:\Program Files (x86)\swipl\lib\jpl.jar" -DgroupId=swiprolog -DartifactId=jpl -Dversion=2.0.2 -Dpackaging=jar

Η οποία εκτελείτε σε ένα command line (WinKey + R και πληκτρολογούμε cmd).

Και εδώ είναι οι εντολές που προσθέτουμε στο pom.xml του project μας:

```
<dependency>
    <groupId>swiprolog</groupId> <!-- used for calling SWI Prolog rules -->
    <artifactId>jpl</artifactId>
    <version>2.0.2</version>
</dependency>
```

15 ΠΑΡΑΡΤΗΜΑ Β’

Αυτό το απόσπασμα κώδικα δίνει ουσιαστικά στο maven τη δυνατότητα να συλλέγει όλα τα Jar και να τα κάνει εκτελέσιμη μορφή .

```
<PLUGIN>
        <GROUPID>ORG.APACHE.MAVEN.PLUGINS</GROUPID>
        <ARTIFACTID>MAVEN-ASSEMBLY-PLUGIN</ARTIFACTID>
        <CONFIGURATION>
          <DESCRIPTORS>

<DESCRIPTOR>SRC/MAIN/ASSEMBLY/ASSEMBLY.XML</DESCRIPTOR>
          </DESCRIPTORS>
        </CONFIGURATION>
      </PLUGIN>
```

16 ΠΑΡΑΡΤΗΜΑ Γ’

package com.teithe.amazonsearcher

- Book: η κλάση που περιγράφει την οντότητα "Βιβλίο" ως αποτέλεσμα της αναζήτησης του Amazon Book Search.
- BookSearchParams: Κλάση controller η οποία λαμβάνει τις τιμές των πεδίων της φόρμας αναζήτησης που είναι σε java swing και τα μετατρέπει σε Http GET parameters για να χρησιμοποιηθούν στην κλήση της αναζήτησης
- FileDownloader: κλάση με μεθόδους λήψης αρχείων μέσω HTTP πρωτοκόλλου. Χρησιμοποιείται για τη λήψη των φωτογραφιών των αποτελεσμάτων.
- New_ptyxiaki: Κλάση με το UI της εφαρμογής σε java Swing
- SearchController: Controller class με την οποία πραγματοποιούμε αναζήτηση στο Amazon Search.