**Α.Τ.Ε.Ι. Θεσσαλονίκης**
**ΣΧΟΛΗ Τεχνολογικών Εφαρμογών**

**Τμήμα Πληροφορικής**

# Ανάπτυξη Windows Native εφαρμογής για την διαχείριση ενός Joomla Site

**Ζησιάδης Μιλτιάδης**

**04/2611**

**ΕΠΟΠΤΗΣ ΚΑΘΗΓΗΤΗΣ:Πούλακας Γεώργιος**

## Πρόλογος

Το Joomla είναι ένα ανοιχτού κώδικα σύστημα διαχείρισης περιεχομένου ταχεία εξελισσόμενο το οποίο αναλαμβάνει για σας την δημοσίευση στο διαδίκτυο μιας προσωπικής σελίδας, ακόμη και ενός ολόκληρου εταιρικού διαδικτυακού τόπου. Εξαιρετικά ευέλικτο και φιλικό, ταυτόχρονα γεμάτο δυνατότητες, είναι μια εφαρμογή που μπορεί να χρησιμοποιηθεί από τον καθένα, χωρίς να απαιτεί καμία εξειδικευμένη γνώση από τον χρήστη στον χώρο τον υπολογιστών. Με το Joomla μπορείτε να δημοσιεύσετε απεριόριστες σελίδες, να προσθέσετε forum, photo galleries, βιβλιοθήκες αρχείων, φόρμες επικοινωνίας και πολλά άλλα χωρίς κανέναν κόστος καθώς είναι μια εφαρμογή ανοιχτού λογισμικού.

## Περίληψη

Στο κείμενο που ακολουθεί θα αναφερθούμε στα CMS και σε κάποιες βασικές τους αρχές, θα γίνει μια εισαγωγή στις δυνατότητες του Joomla ρίχνοντας ταυτόχρονα μια ευρεία μάτια στην δομή του. Θα σχολιαστεί εκτενώς η εφαρμογή που δημιουργήσαμε σε προσπάθεια να καταλάβει ο αναγνώστης τον τρόπο με τον οποίο λειτουργεί καθώς και τρόπο με τον όποιον σχεδιάστηκε έτσι ώστε να μπορεί να είναι επεκτάσιμη από οποιονδήποτε.

## Summary

The following text will refer to CMSs and to their core principles. We will make an introduction about Joomla features throwing together a wide eye to the structure. The application created will be discussed in details in an attempt to help the reader to capture how it functions and how it was designed so that it can be extensible by anyone.

## Περιεχόμενα

# 1. Τι είναι τα CMS

Είναι ένα σύνολο από διαδικασίες, εφαρμογές και βάσεις δεδομένων που βοηθούν έναν οργανισμό να δημιουργήσει, να αποθηκεύσει, να συντονίσει, να γνωστοποιήσει πληροφορίες σε ένα χρήσιμο σχήμα με μία συνεπή μέθοδο. Με τον όρο content αναφερόμαστε σε οποιαδήποτε πληροφορία που έχει νόημα , και έχει συγκεκριμένο σχήμα, η οποία θα καταναλωθεί από το κοινό. Με την πάροδο του χρόνου δημιουργήθηκε η ανάγκη στον χρήστη να ψάξει για έναν τρόπο με τον οποίον θα μπορούσε να δημοσιεύσει επαγγελματικού τύπου σελίδες εύκολα και γρήγορα χωρίς να γνωρίζει HTML . Καθώς το διαδίκτυο μεγάλωνε συνεχώς, δημιουργούταν η ανάγκη για έναν εύχρηστο τρόπο διαχείρισης των σελίδων. Έτσι δημιουργήθηκαν τα CMS τα οποία υπόσχονται να λύσουν αυτά τα προβλήματα και να ανταπεξέλθουν στις νέες απαιτήσεις.

## 1.1. Κατηγορίες συστημάτων διαχείρισης

Τα CMS ποικίλουν σε λειτουργικότητα και μπορούν να διαχειριστούν οτιδήποτε στο οποίο γίνονται εργασίες από μια ομάδα ατόμων. Από την διαχείριση απλού στατικού περιεχομένου ενός website, μέχρι την μεταφορά εγγράφων μιας επιχείρησης προς όφελος την συνεργασίας της. Μπορούμε να τα χωρίσουμε σε δυο γενικές κατηγορίες: τα Enterprise CMS και τα Web CMS.

**Enterprise CMS**

Αυτά τα υψηλής χρηστικότητας πακέτα δεδομένων είναι συνήθως ολοκληρωμένες λύσεις, που προσφέρουν αποτελεσματική διαχείριση περιεχομένου, για χρήση σε μια επιχείρηση . Έχουν σχεδιαστεί για να βοηθήσουν μια εταιρεία να γίνει πιο αποτελεσματική και πιο αποδοτική, αυξάνοντας την ακρίβεια και την αποτελεσματικότητα, μειώνοντας έτσι το ανθρώπινο λάθος και τους χρόνους απόκρισης των πελατών. Μπορούν να ενσωματώσουν εταιρικές λειτουργίες όπως η ναυτιλία και τα συστήματα διανομής, τιμολόγηση, ζητήματα των ανθρωπίνων πόρων, σχέσεις με τους πελάτες, καθώς και τη διαχείριση εγγράφων και συστημάτων συναλλαγών (πωλήσεις). Το Enterprise CMS φέρει την διαχείριση των δεδομένων στο επίπεδο χρήστη έτσι ώστε πολλοί χρήστες να μπορούν να προσθέσουν το δικό τους «κομμάτι» στην ολοκληρωμένη εικόνα. Οι εταιρείες λογισμικού που παράγουν αυτά τα πολύπλοκα συστήματα υπερηφανεύονται για την παροχή υψηλής προσαρμοστικότητας λύσεων. Το λογισμικό έχει, συνήθως, σχετικά μεγάλη τιμή.

**Web CMS**

Τα Web CMS επί το πλείστον δημιουργήθηκαν για χρήση στο web. Μπορούν να περιλαμβάνουν πολλαπλές λειτουργίες, ή να έχουν μία συγκεκριμένη λειτουργία στην όποια είναι επικεντρωμένα.

Επιτρέπουν στους χρήστες να ενημερώσουν κομμάτια της τοποθεσίας Web ή να συνεργάζονται σε μια website community . Το Web CMS μπορεί να κάνει τη ζωή ενός προγραμματιστή εύκολη προσφέρνοντας λειτουργικότητα σε μια ιστοσελίδα εύκολα και γρήγορα, και επιτρέποντας στον κύριο προγραμματιστή του έργου να συμπεριλάβει άλλους προγραμματιστές στην

ανάπτυξη και στη συντήρηση του site χωρίς φόβο να εκ-παπεί από τα πρότυπα.

Ανάλογα με το κόστος απόκτησης, την δυνατότητα ελέγχου και το χώρο αποθήκευσης τα CMS χωρίζονται σε κάποιες κατηγορίες.

- **Hosted**

Ο προμηθευτής του CMS αναλαμβάνει την συντήρηση και την φιλοξενία του, έτσι ο πελάτης απελευθερώνετε από ευθύνες διαχείρισης, γλιτώνοντας έτσι κάποια μεσοπρόθεσμα κόστη. Στον αντίποδα όμως το CMS μειώνει το ποσοστό ελέγχου του πελάτη και δημιουργεί μεγαλύτερα μακροπρόθεσμα κόστη.

- **Commercial**

Ο προμηθευτής αναπτύσσει μια εφαρμογή CMS και την πουλάει στον πελάτη, ο οποίος είναι υπεύθυνος να την συντηρήσει. Ο πελάτης έχει έτσι περισσότερο έλεγχο αλλά και ευθύνες. Συνήθως οι εμπορικές εφαρμογές είναι και ακριβές.

- **Non-profit**

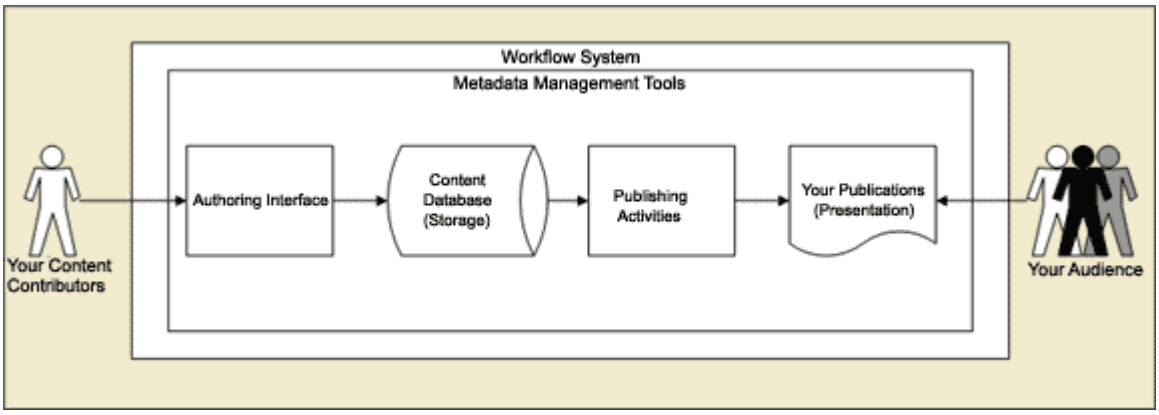
Μερικά CMS αναπτύσσονται από non-profit (μη κερδοσκοπικούς) οργανισμούς για non–profit οργανισμούς, αλλά συνήθως περιέχουν στοιχεία που μόνο οι non-profit βρίσκουν χρήσιμα.

- **Open Source**

Όπως με όλα τα προγράμματα ανοιχτού λογισμικού δεν υπάρχει κόστος για να αγοραστεί το πρόγραμμα. Ο πελ άης έχει αρκετό έλεγχο και πολλές ευθύνες, είναι όμως βασισμένος στην κοινότητα του open source για υποστήριξη. Συνήθως αυτά τα προϊόντα είναι ελλιπώς τεκμηριωμένα.

## 1.2. Πλεονεκτήματα και τρόπος με τον οποίον λειτουργούνε

Πως δουλεύει το CMS




**Εικόνα 1.1 Πως δουλεύει το CMS**

Στο κέντρο του CMS υπάρχει ένα είδος αποθήκης, η βάση δεδομένων του περιεχόμενου. Τα δεδομένα εισέρχονται μέσα στην βάση μέσω του authoring interface και κατηγοριοποιούνται χρησιμοποιώντας τα εργαλεία διαχείρισης metadata. Όταν το περιεχόμενο είναι έτοιμο το CMS παίρνει το περιεχόμενο για να το εκδώσει . Εικόνα 1.1

### Πως δημιουργείται η σελίδα

Σε ένα κανονικό site, όλες οι σελίδες προϋπάρχουν στον server .ενώ στο CMS που είναι server-side application, οι σελίδες δημιουργούνται δυναμικά

όταν ζητηθούν, δεν προϋπάρχουν πριν ο browser τις ζητήσει. Η διαδικασία είναι η εξής :

1. Ο browser του επισκέπτη ζητάει μια σελίδα από τον server
2. Ο server (συνήθως apache) κοιτάζει στην cache του μήπως η σελίδα είναι στην μνήμη, επειδή ζητήθηκε προηγουμένως σε μια κοντινή χρονική περίοδο., Εάν ναι παίρνει την σελίδα και την επιστρέφει. Εάν όχι ζητεί την σελίδα από το CMS
3. Το CMS κοιτάζει στην δικιά του cache, εάν έχει, και εάν εντοπίσει την σελίδα την επιστρέφει. Εάν όχι δημιουργεί την σελίδα, παίρνει τις παραμέτρους έκδοσης και το κείμενο από την βάση δεδομένων, έπειτα τα γραφικά τις εικόνες και ότι άλλο χρειαστεί αντίστοιχα, δημιουργείται η σελίδα και επιστρέφεται στο λογισμικό του server
4. Ο server περνά την σελίδα στον browser, εκτελεί επίσης ένα πλήθος άλλων στοιχείων ταυτόχρονα. Ρωτάει το κορυφαίου επιπέδου αρχείο htaccess, ρώτα το τοπικό αρχείο htaccess για όλα τα είδη επιλογών και μεταβλητών, εξυπηρετεί τα script που σχετίζονται με την σελίδα css, js , εκτελεί στα script, καταγράφει όλη την κυκλοφορία που σχετίζεται με το συγκεκριμένο αίτημα, καταγράφει οποιαδήποτε λάθη. Αυτή η διαδικασία δημιουργεί κάποια μειονεκτήματα.

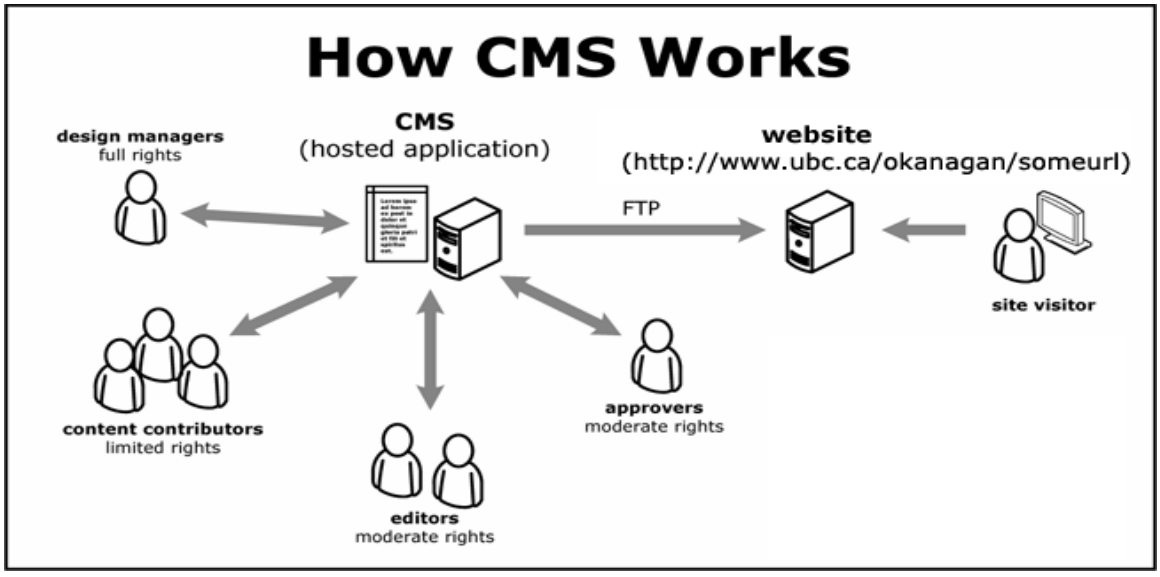
**Υπερφόρτωση του server**

Η όλη αυτή διαδικασία δημιουργεί κάποια υπερφόρτωση στον server κτίζοντας τις σελίδες CMS και υλοποιώντας τις άλλες ενέργειες που αναφέρθηκαν παραπάνω. Χρησιμοποιείται cpu, μνήμη, και άλλοι πόροι του συστήματος. Αυτό μεταφράζεται σε φόρτο του server (server load) και σε μεγάλους χρόνους φόρτωσης σελίδων.

Αυτό μπορεί να μετρηθεί με benchmarking του apache . Για παράδειγμα το φορτίο του server σε κανονικές σελίδες html είναι μεταξύ 100-400 σελίδες το λεπτό. Συνήθως το μικρότερο νούμερο είναι και πιο ρεαλιστικό καθώς για να επιτευχθεί το υψηλό νούμερο πρέπει ένα πλήθος παραμέτρων να έχουν πάρει τις βέλτιστες τιμές. Αυτό σημαίνει ότι ο server μπορεί να παρέχει υπηρεσίες σε πάνω από εκατό επισκέπτες το δευτερόλεπτο.

Εντούτοις, όταν χρησιμοποιείται ένα CMS, οι σελίδες που παρέχονται μπορεί να μειωθούν σε τέσσερις ανά δευτερόλεπτό. Αυτό είναι ένα πολύ πιο ρεαλιστικό νούμερο.

Αυτά τα προβλήματα λύνονται ως ένα σημείο με την cache, τόσο στο CMS όσο και στο server που κρατάει τις δημοφιλείς σελίδες στην μνήμη. Για το λόγο αυτό οι server χρειάζονται αρκετή μνήμη και γρήγορους δίσκους.




**Εικόνα 1-2**

### Πλεονεκτήματα που προκύπτουν από το CMS

Επειδή το περιεχόμενο δεν είναι κολλημένο με το σχήμα προκύπτουν τα εξής πολύ σημαντικά πλεονεκτήματα:

**Φορητότητα περιεχομένου**

Επειδή το CMS αποθηκεύει το περιεχόμενο ως δεδομένα, τα δεδομένα αυτά μπορούν να εισαχθούν σε οποιοδήποτε σχήμα ή template . Έτσι δεν χρειάζεται να γραφτεί το άρθρο δύο φόρες εάν θέλουμε να εμφανίζεται διαφορετικά σε διάφορα μέρη της ιστοσελίδας . Αντί αυτού γράφεται μια φόρα και ανατίθεται στο κατάλληλο template για παρουσίαση.

**Ευελιξία σχεδίασης**

Επειδή το CMS αποθηκεύει το template ξεχωριστά από τα δεδομένα περιεχομένου, εάν θελήσουμε να κάνουμε μια αλλαγή στη σχεδίαση, το μόνο που χρειάζεται να είναι να αλλαχτεί το συγκεκριμένο template

**Ενιαία αποθήκευση σε μια ενιαία θέση**

Σε ένα CMS όλα τα δεδομένα περιεχομένου αποθηκεύονται σε ένα μέρος μια φόρα!, Πχ εάν υπάρχουν διαφορετικές εκδόσεις ενός άρθρου και ο χρήστης δεν ξέρει πια να χρησιμοποιήσει, με ένα CMS θα ήταν πιο εύκολο . Το σύστημα έχει μόνο ένα αντίγραφο του άρθρου και κατηγοριοποιείται σε όποιο section διαλέξει ο χρήστης. Έτσι εάν ένα άρθρο που χρειαζόταν αλλαγή εμφανιζόταν σε τρία sections, χωρίς το CMS θα χρειαζόταν τρείς αλλαγές, ενώ με το CMS αλλάζοντας το ένα και μοναδικό αποθηκευμένο άρθρο, αλλάζουν σε όλα τα sections.

**Διαχείριση ροής**

Ένα καλό CMS θα έχει ένα είδος διαχείρισης ροής . Αυτό συνήθως σημαίνει τον καθορισμό κάποιον ρόλων, author, editor, publisher και την ανάθεση κάποιον ικανοτήτων και ευθυνών σε αυτούς. Η διαχείριση ροής προσφέρει καλύτερη επικοινωνία, καταγραφή προόδου και πιο αποδοτικές μεταβάσεις περιεχομένου. Το κύριο πλεονέκτημα είναι ο έλεγχος με τον οποίο κερδίζουμε σε χρόνο και χρήμα με το να επιταχύνουμε την επικοινωνία και να

αποφύγουμε λάθη. Το σύστημα διαχείρισης ροής χειρίζεται ένα μεγάλο μέρος της επικοινωνίας. Έτσι οι authors, editors, publishers μπορούν να επικεντρωθούν στο γράψιμο, την αναθεώρηση κα την έκδοση χωρίς να χάνουν πολύτιμο χρόνο στο να ελέγχουν τα πράγματα, να κοιτάν για χαμένα άρθρα και να προσπαθούν να συνειδητοποιήσουν που πήγε χαμένος όλος αυτός ο χρόνος

**Αυτοματοποιημένη έκδοση**

Εκεί που το CMS κάνει την διαφορά είναι ότι επιτρέπει σε μη τεχνικά καταρτισμένα άτομα να προγραμματίζουν και να πραγματοποιούν την διαδικασία μετάβασης κάποιου περιεχομένου στο περιβάλλον παραγωγής . Έτσι οι τεχνικοί δεν αναλώνονται στο να κάνουν οτιδήποτε αλλαγές στο περιεχόμενο, καθώς το CMS προσφέρει αυτή τη δυνατότητα σε οποιοδήποτε χρήστη μέσω αυτοματοποιημένων φορμών, αλλά ασχολούνται με την συντήρηση του CMS σε πιο βαθύ επίπεδο.

Εμείς θα ασχοληθούμε με ένα συγκεκριμένο είδος του CMS to WCMS (Web Content Management Systems )

Αυτού του τύπου το CMS ασχολείται με την διαχείριση περιεχομένου του WEB. Τα προϊόντα διαφέρουν σε λειτουργία και πολυπλοκότητα.

Το WCMS είναι ένα πρόγραμμα που είναι εγκατεστημένο στον server και παρουσιάζει σελίδες από ένα website . Οι σελίδες αυτές δεν προϋπάρχουν αλλά δημιουργούνται δυναμικά από μια βάση δεδομένων με την βοήθεια του λογισμικού του CMS . Ο ιδιοκτήτης μπορεί να τροποποιήσει το περιεχόμενο εάν είναι συνδεμένος στο internet χωρίς την βοήθεια του webmaster . Επιπρόσθετες λειτουργίες και συναρτήσεις προστίθενται σαν plug-in έτσι ώστε να μην χρειάζεται προγραμματισμός από τη μεριά του χρήστη. Η σχεδίαση της σελίδας βασίζεται σε κάποια templates. Αυτό σημαίνει ότι το περιεχόμενο είναι διαφορετικό από τη σχεδίαση

**Τα κύρια σημεία του WCMS**

- Οι σελίδες τροποποιούνται online μέσω ενός browser.
- Οι τροποποιήσεις γίνονται σε πραγματικό χρόνο, άμεσα.
- Ο ιδιοκτήτης μπορεί εύκολα να τροποποιήσει, προσθέσει, διαγράψει σελίδες.
- Με μικρή προσπάθεια ο ιδιοκτήτης μπορεί να προσθέσει αντικείμενα σε menu και ακόμα και sections στη σελίδα.
- Η σχεδίαση και ο τρόπος παρουσίασης εξαρτώνται από τα templates, η χειροκίνητη σχεδίαση δεν είναι απαραίτητη.
- Επιπρόσθετα χαρακτηρίστηκα προστίθενται σαν plug-in.
- Το περιεχόμενο πολλών διαφορετικών ειδών, μπορεί να οργανωθεί και να παρουσιαστεί με πολλούς διαφορετικούς τρόπους.

**Πλεονεκτήματα χρήσης του CMS**

Ένα CMS διευκολύνει τον χρήστη αρκετά για τους παρακάτω λογούς:

- Δεν εξαρτάται πλέον από web designers για να κάνουν αλλαγές για αυτόν.
- Οι αλλαγές μπορούν να γίνουν οποιαδήποτε στιγμή θέλει, νύχτα η μέρα . Αυτό είναι πολύ σημαντικό καθώς η επιχείρησή του βασίζεται στο website σαν ένα μέρος επικοινωνίας.
- Όλες οι τεχνικές λεπτομέρειες αποκρύπτονται από το χρήστη, επιτρέποντας στον καθένα με ελάχιστες γνώσεις να διαχειριστεί τη σελίδα.

- Επιτρέπει σε μια ομάδα ατόμων να κρατούνε το site ενήμερο, από το να έχει κάποιο συγκεκριμένο άτομο να το κάνει, το CMS θα κρατήσει αρχείο ποιος αλλάζει τι, και έτσι θα αποφευχθεί η σύγχυση.
- Μπορεί να εξασφαλιστεί ότι το κάθε μέλος μπορεί να ενημερώσει μόνο το section που είναι υπεύθυνο για αυτό.
- Το CMS εξασφαλίζει ότι όλες οι σελίδες είναι συνεπείς στην σχεδίαση τους, και θα δημιουργήσει τα μενού και οτιδήποτε άλλο για την περιήγηση του χρήστη.

## Συχνά χρησιμοποιούμενοι όροι

### Metadata

Είναι δεδομένα που παρέχουν πληροφορίες για τα δεδομένα που περιέχονται μέσα στο CMS . Είναι δηλαδή ένα κομμάτι που περιγράφει το τί είναι το περιεχόμενο και που αναφέρεται . Συχνά περιγράφεται ως πληροφορία για την πληροφορία.

### Template

Το CMS χρησιμοποιεί τα templates για να ελέγξει την επίδειξη του περιεχομένου του site, είναι ο τρόπος με τον οποίο το περιεχόμενο θα εμφανίζεται στο site . Τα template δημιουργούνται από τους web designers και διαχειρίζονται ξεχωριστά από το περιεχόμενο . Κατά το χρόνο της δημοσιοποίησης, το CMS τοποθετεί το περιεχόμενο μέσα στο template για

την τελική παρουσίαση . Είναι σαν άδειες εκδόσεις τύπων σελίδων , μέχρι το CMS να βάλει συγκεκριμένο περιεχόμενο μέσα τους, δεν υπάρχει τίποτα εκεί.

**Content and presentation**

Πολύ συχνά οι χρήστες μπερδεύουν αυτούς τους δύο όρους και νομίζουν ότι είναι το ίδιο. Όταν δημιουργείται κάποιο περιεχόμενο, το CMS απομονώνει τα δεδομένα του περιεχομένου από το σχήμα του περιεχομένου και τα metadata του. Στις ιστοσελίδες τα δεδομένα συνήθως αποτελούνται από το κείμενο και κάποιες εικόνες που εμφανίζονται. Το σχήμα εξαρτάται από το template που ορίζεται, γι' αυτό το περιεχόμενο και τα metadata ορίζονται στη συνέχεια. Το CMS αποθηκεύει αυτά τα συστατικά ξεχωριστά και διατηρεί την σχέση μεταξύ τους . Ο λόγος που γίνεται αυτό είναι για να διατηρείται η φορητότητα των δεδομένων. Εφόσον αποθηκεύονται ξεχωριστά τα δεδομένα και το format τους , μπορούν να χρησιμοποιηθούν με διαφόρους τρόπους.

## 2. Τι είναι το Joomla

Το Joomla είναι ένα βραβευμένο σύστημα διαχείρισης περιεχομένου (CMS), το οποίο επιτρέπει στο χρήστη να δημιουργήσει ιστοσελίδες και ισχυρές online εφαρμογές. Οι πολλές πτυχές του, συμπεριλαμβανομένης της ευκολίας χρήσης και της επεκτασιμότητας, έχουν κάνει το Joomla το δημοφιλέστερο διαθέσιμο λογισμικό Web. Το καλύτερο από όλα, το Joomla είναι μια λύση ανοιχτού λογισμικού που διατίθεται δωρεάν σε όλους.

Σε ένα Joomla website ενώνονται 3 στοιχεία

- Το περιεχόμενο, που είναι κυρίως αποθηκευμένο σε μια βάση δεδομένων
- Το TEMPLATE, που ελέγχει την σχεδίαση και την παρουσίαση του περιεχομένου του site (γραμματοσειρές, χρώματα, διατάξεις)
- Το JOOMLA που είναι το λογισμικό που ενώνει το περιεχόμενο με το template για να παραχθούν σελίδες

### 2.1. Στοιχεία πυρήνα του Joomla

**Τα CORE Features του Joomla**

- Components
- Modules
- Plug-ins
- Templates

**Τα Components χωρίζονται σε**

- Banner
- Contacts
- Newsfeed
- Polls
- Search
- Web links

Λίγα λόγια για το καθένα τους.

**Banner**

Το banner component επιτρέπει στο χρήστη να διαχειριστεί τα banners με categories και clients . Το banner manager έχει τρεις επιλογές banners, clients, categories.

**Banner tab**

Δείχνει μια λεπτομερή λίστα με τα ενεργά banners της σελίδας .Επιτρέπει ακόμη να προστεθούν και να τροποποιηθούν ήδη υπάρχοντα banners.

**Client tab**

Δείχνει μια λίστα με πελάτες και πληροφορίες των εγγραφών τους.

**Categories tab**

Επιτρέπει να οργανωθούν ολόκληρες κατηγορίες από banners και να αλλαχθεί η κατάσταση Published τους.

**Contacts**

Το contact component επιτρέπει στο χρήστη να διαχειριστεί ένα φάκελο με επαφές (contacts) όσον αφορά τα contacts στη σελίδα.

**Contact tab**

Δείχνει μια λεπτομερή λίστα με ενεργά contacts. Επίσης παρέχει τη δυνατότητα να προστεθούν καινούργια ή να τροποποιηθούν τα ήδη υπάρχοντα .

**Newsfeed**

Το newsfeed component ενεργοποιεί τα πιο πρόσφατα άρθρα από εξωτερικά website feeds να συνδεθούν για περαιτέρω διάβασμα.

Το feed stab περιέχει μια λεπτομερή λίστα από όλα τα ενεργά feeds μέσα από το site.

**Categories tab**

Επιτρέπει ολόκληρες κατηγορίες από feeds να οργανωθούν και να αλλαχθεί η κατάσταση published τους.

Τα Feeds διαχειρίζονται μέσα από categories και έχουν διάφορες παραμέτρους όπως Number of articles, cache time και ordering.

**Polls**

Το polls component δείχνει μια λίστα με ενεργά polls της σελίδας . Τα polls διαχειρίζονται εύκολα με παραμέτρους όπως Lag (χρόνος μεταξύ ψήφων) και options, για τα οποία οι επισκέπτες μπορούν να ψηφίσουν.

**Search**

Το Search component παρέχει στατιστικές για searches που έγιναν χρησιμοποιώντας το Joomla search plug-in

Το component παρέχει τις στατιστικές σαν μια απλή λίστα που περιέχει search text σε σύγκριση με time requested, και είναι ταξινομημένα ανά time requested

**Web links**

Το Web link component lists παρέχει διαχείριση ελέγχου για τα web links που εμφανίζονται στην αρχική σελίδα

**Links tab**

Δείχνει μια λίστα με ενεργά links οργανωμένα ανά category και order

**Categories tab**

Επιτρέπει ολόκληρες κατηγορίες από links να οργανωθούν και να αλλάξει η κατάσταση δημοσίευσης τους.

Τα web links ρυθμίζονται στις categories και έχει διάφορες παραμέτρους όπως target και description

**Modules**

- Archived Content
- Banners & Feed
- Custom HTML
- Breadcrumbs
- Footer
- Login
- Menu
- Most Read & Latest News

- News Flash
- Polls
- Random Image
- Related Items
- Search
- Section
- Statistics
- Syndicate
- Who’s online?
- Wrapper

---

**Archived content**

Το Archived content  module επιτρέπει τον προσδιορισμό θέσης ενός display από content items που έχουν γίνει archive από ένα publisher .Τα content items είναι διαθέσιμα από μια λίστα ανά μήνα/χρόνο.

Τα items ταξινομούνται ανά ημερομηνία δημιουργίας. Μόνο τα items με περιεχόμενο θα εμφανιστούν, έτσι δεν θα υπάρχει λίστα από άδειους φακέλους.

**Banners and feed**

Τα modules των banners and feed είναι ο μηχανισμός εμφάνισης για τα αντίστοιχα συστατικά.

Και τα δυο μπορούν να διαμορφωθούν για να εμφανίσουν συγκεκριμένα αντικείμενα σε συγκεκριμένες σελίδες.

Τυπικές παράμετροι όπως menu assignment και details είναι διαθέσιμες.

**Custom HTML**

Το custom html module επιτρέπει τη δημιουργία ενός custom module. Τα custom html modules μπορούν να έχουν περιεχόμενο όπως κείμενο, εικόνες και συνδέσμους.

**Breadcrumbs**

Το breadcrumbs module επιτρέπει την προσθήκη breadcrumbs στην αρχική σελίδα. Τα breadcrumbs είναι μια βοήθεια πλοήγησης που συχνά χρησιμοποιείται στα user interface. Τα breadcrumbs δίνουν στους χρήστες έναν τρόπο να παρακολουθούν την θέση τους μέσα στο website.

**Footer**

Το footer module δείχνει της πληροφορίες copyright του Joomla:

"Copyright © YYYY [Site Name Here]. All Rights Reserved."
"Joomla! is Free Software released under the GNU/GPL License."

**Login**

Το login module δείχνει τα πεδία που είναι απαραίτητο να συμπληρωθούν για να γίνει login, τα default πεδία είναι username, password, remember me.

**Menu**

Το menu module δείχνει ένα μενού που ορίζεται μέσα από το menu manager. Όλα τα Joomla sites έχουν ένα main menu που δεν μπορεί να διαγραφεί . Το menu module επιτρέπει σε νέα μενού να εμφανίζονται σε διαφορετικές θέσεις . Το module έχει διάφορες παραμέτρους όπως menu name, menu style, και επιλογές υπομενού.

**Most Read and Latest news**

Και τα most read, latest news modules δείχνουν μια λίστα από links είτε στα πλέον αναγνωσμένα άρθρα, είτε στα πρόσφατα νέα. Και τα δυο είναι εύκολα διαχειρίσημα μέσω παραμέτρων έτσι ώστε να οριοθετηθούν σε συγκεκριμένα

sections και categories. Στάνταρ παράμετροι είναι menu assignment και details.

**News flash**

Το newsflash module δείχνει content items από ένα συγκεκριμένο section / category list. Στάνταρ παράμετροι είναι menu assignment και details.

**Polls**

Polls module είναι ο μηχανισμός παρουσίασης στην αρχική σελίδα του component polls (ψηφοφορίες).

Τα polls module / component μπορούν να διαμορφωθούν για να δείξουν συγκεκριμένα polls σε συγκεκριμένες σελίδες. Στάνταρ παράμετροι είναι menu assignment και details.

**Random Image**

Το random image module δείχνει μια τυχαία εικόνα από τα περιεχόμενα ενός συγκεκριμένου φακέλου εικόνων . Αυτός ο φάκελος ορίζεται μέσω παραμέτρων . Στάνταρ module παράμετροι όπως menu assignment και details είναι διαθέσιμοι έκτος από ορισμένους όπως οι advanced parameters.

**Related items**

Το related items module χρησιμοποιεί τα μεταδεδομένα που έχουν σχέση με το κάθε περιεχόμενο. Το module θα εμφανίσει content items με τις ίδιες λέξεις κλειδιά, δίνοντας έτσι στο χρήστη επιπρόσθετη δυνατότητα στην ποσότητα των πληροφοριών που θα λάβει από το website.

Βεβαία είναι σημαντικό να εξασφαλίσουμε ότι οι λέξεις κλειδιά θα αντικατοπτρίζουν με ακρίβεια το περιεχόμενο σε κάθε αντικείμενο, αλλιώς το πλεονέκτημα θα χαθεί αμέσως.

Το λίγο είναι σίγουρα περισσότερο (αρκεί να είναι ακριβές!).

**Search**

Το search module παρέχει στο χρήστη την δυνατότητα να πραγματοποιήσει στιγμιαία μια βασική αναζήτηση από την συγκεκριμένη τοποθεσία πού βρίσκεται, και να μεταφερθεί αμέσως στα αποτελέσματα. Αυτό αφαιρεί την ανάγκη να πλοηγείται από ένα παράθυρο search.

**Sections**

Το section module δείχνει μια λίστα από όλα τα article sections του website.

Εάν η παράμετρος show unauthorized links τίθεται στο όχι η λίστα θα οριοθετηθεί στα sections στα οποία ο χρήστης έχει επίπεδο πρόσβασης .

**Statistics**

Το statistic module δείχνει μια βασική λίστα με πληροφορίες όσο αφορά τον server και την εγκατάσταση του Joomla. Στις πληροφορίες που εμφανίζονται περιλαμβάνονται και, το λειτουργικό σύστημα του server, η έκδοση php, η έκδοση mysql, ο χρόνος, η μνήμη cache, τα μέλη, το περιεχόμενο, οι σύνδεσμοι και ποσά άτομα είδαν το περιεχόμενο.

**Syndicate**

Το syndicate module δείχνει μια λίστα εικόνας με όλους τους διαθέσιμους τύπους feeds για την σελίδα. Το module δουλεύει από κοινού με το syndicate component όπου τα πραγματικά feeds διαμορφώνονται. Ένας χρήστης της σελίδας που επιθυμεί να προσθέσει ένα feed, θα χρειαστεί να κάνει δεξί κλικ στην εικόνα, και να αντιγράψει την τοποθεσία του συνδέσμου στο δικό του αναγνώστη news feed, η έτσι όπως είναι πιο κοινό στις μέρες μας να χρησιμοποιήσει τον αναγνώστη news feed ή να περιηγηθεί για να προσθέσει το news feed αυτόματα.

**Who`s online**

Το who`s online module παρέχει μια βασική θέαση του αριθμού των επισκεπτών και του πλήθους των μελών (εγγεγραμμένοι χρήστες) που είναι συνδεμένοι στο website την ώρα που έγινε το ερώτημα.

Το module μπορεί να ρυθμιστεί ώστε να εμφανίζει τα ονόματα των μελών που είναι συνδεμένοι.

**Wrapper**

Το wrapper module περικαλύπτει μια άλλη σελίδα απευθείας μέσα στην σελίδα μας σε μια συγκεκριμένη τοποθεσία που ορίζεται από την θέση του module. Η σελίδα εισέρχεται σαν ευθύγραμμο πλαίσιο μέσα σε σχεδιάγραμμα website template.

Το module μπορεί να χρησιμοποιηθεί για να εμφανίσει ένα άλλο website, αλλά και εναλλακτικές σελίδες από εκείνο το website που είναι εγκατεστημένες σε αυτό, όπως επίσης επιτρέποντας εντελώς ανεξάρτητες εφαρμογές και scripts να τρέξουν στην σελίδα, και να εμφανιστούν οι έξοδοι τους σαν εσωτερικό κομμάτι τις σελίδας.

**Plugins**

- Authentication (Joomla!, LDAP, OpenID, GMail)
- Cache
- Code Highlighter (GeSHi)
- Email Cloaking
- Editors (TinyMCE 2.1 & Xstandard Lite for Joomla!)
- Editors-XTD (Image, Page Break, Readmore)
- Legacy
- Rating
- Search (Categories, Newsfeeds, Sections, Contacts, Content, Weblinks)
- SEF

- XML-RPC (Blogger API, Joomla! API)
- Authentication (Joomla!, LDAP, OpenID, GMail)

Το authentication plug-in προσθέτει διαφορετική μορφή επικύρωσης στο site.

Το Joomla authentication plug-in περιλαμβάνει αρχικά εγγραφή στην σελίδα και έπειτα επιβεβαίωση της ταυτότητας του χρήστη μέσο ενός επιβεβαιωτικού email.

Τα opened, GMail plug-in χρησιμοποιούν προϋπάρχοντες βάσεις δεδομένων για να πιστοποιήσουν τον χρήστη. Από προεπιλογή μόνο το Joomla authentication plug-in είναι ενεργοποιημένο.

**Cache**

Το cache plug-in παρέχει την λειτουργικότητα της αποθήκευσης σελίδων. Αυτό μειώνει το φορτίο στον server και το lag. Με το να εναποθηκεύονται οι σελίδες, οι καινούργιες σελίδες δεν χρειάζονται να δημιουργούνται τόσο συχνά, παρόλα αυτά τα πλεονεκτήματα της εναποθήκευσης απαιτούν υψηλά επίπεδα φόρτου για να γίνουν αντιληπτά.

**Code highlighter(GeSHi)**

GeSHi είναι πρόγραμμα συντακτικού, ανοιχτού λογισμικού που βοηθάει στη επεξήγηση διαφορετικών γλωσσών κώδικα όπως php, JavaScript, html, την κ αθεμία με διαφορετικ ό χρώμα, έτσι ώστε να βοηθήσει με την ευκολία ανάγνωσης κώδικα σε ένα content item. Αυτό το τμήμα κώδικα που εισέρχεται πρέπει να συμπεριληφθεί με τα <pre> </pre> έτσι ώστε να ορίζει ένα μπλοκ με προδιαμορφωμένο κώδικα .

**Email cloaking**

Το email cloaking plug-in προσθέτει ένα στρώμα ασφάλειας στις διευθύνσεις email που περιέχονται μέσα στα content items και στις επαφές. Το plug-in χρησιμοποιεί JavaScript έτσι ώστε να κρύψει αποτελεσματικά την διεύθυνση email, εάν μια διεύθυνση email εμφανίζετε όπως οι ακόλουθες:

address@mydomain.com

- <a xhref="mailto:address@mydomain.com">address@mydomain.com</a>
- <a xhref="mailto:address@mydomain.com">mail Admin!</a>

**Editors (TinyMCE 2.1 & Xstandard Lite for Joomla!)**

Και το **TinyMCE** και το **Xstandard Lite** είναι WYSIWUG (αυτό που βλέπεις είναι αυτό που παίρνεις) editors . Επιτρέπουν την δημιουργία content item και την τροποποίηση περιγραφών, μεταξύ των στόχων τους.

Το **TinyMCE 2.1** είναι ο προεπιλεγμένος editor και ο πιο πλούσιος σε λειτουργιές. Προσφέρει όλα τα χαρακτηριστικά γνωρίσματα που χρειάζονται για στάνταρ χρήση.

Ο προεπιλεγμένος editor μπορεί να τεθεί από το παράθυρο "Global configuration".

**Legacy**

Το legacy plug-in επιτρέπει την υποστήριξη για το Joomla 1.0. Αυτό το plug-in επιτρέπει σε προϋπάρχοντα components, mambots, templates και modules να λειτουργούν στο Joomla 1.5

Από προεπιλογή το legacy plug-in είναι απενεργοποιημένο, αλλά μπορεί να ενεργοποιηθεί εάν χρειαστεί.

Συστήνεται να χρησιμοποιείτε extension που τρέχουν αποκλειστικά στο Joomla 1.5 .

**G Editors-XTD (Image, Pagebreak, Readmore)**

Αυτά τα plug-in προσθέτουν σημαντική λειτουργικότητα στο front-end για να εμφανίζεται το περιεχόμενο και στο backend για να ενεργοποίηση και διαμόρφωση αυτής της λειτουργικότητας.

Το image editor xtd plug-in εμφανίζει ένα κουμπί έτσι ώστε να είναι δυνατόν να προστεθούν εικ άνες σε έν α άρθρο . Το κουμπί δημιουργεί ένα popup, μέσα από το οποίο μπορούν να διαμορφωθούν οι ιδιότητες της εικόνας , και να <ανεβούν> καινούργια αρχεία εικόνων.

Το page editor tad plug-in παρέχει ένα κουμπί που ενεργοποιεί την εισαγωγή ενός page break σε ένα άρθρο. Ένα popup επιτρέπει στο χρήστη να διαμορφώσει τις επιλογές που θα χρησιμοποιηθούν, όπως το title.

Το readMore editor xtd pug in ενεργοποιεί ένα κουμπί έτσι ώστε να είναι δυνατόν να προστεθεί η επιλογή readmore μέσα σε ένα άρθρο.

**Rating**

Το rating plug-in προσθέτει την λειτουργικότητα ενός συστήματος βαθμολόγησης αναγνώστη των content items στο frontend ενός website. Αυτή είναι η συχνότητα 5 αντικειμένων εικόνων που ανάβουν σύμφωνα με τις επιλογές που έγιναν από τους χρήστες στο frontend.

Από προεπιλογή αυτό το plug-in είναι απενεργοποιημένο.

**Search (Categories, Newsfeeds, Sections, Contacts, Content, Web links)**

Τα διαφορά search plug-in ενεργοποιούν το search component να ψάξει όλους τους τύπους περιεχομένου στην σελίδα. Τα διαφορετικά plug-in είναι για κάθε τύπου περιεχόμενο, επιτρέποντας την ενεργοποίηση και απενεργοποίηση της εύρεσης διαφορετικού τύπου περιεχομένου.

Χρησιμοποιώντας αυτήν την τεχνολογία , το περιεχόμενο αποθηκεύεται και διαχειρίζεται από components τρίτων, μπορεί να γίνει εύρεση εάν ένα κατάλληλο plug-in γίνει διαθέσιμο

**SEF**

Το sef plug-in προσθέτει φιλική λειτουργικότητα στα URL links των content items όσο αναφορά τα search engines.Ενεργοποιείται αυτόματα εάν γίνουν published και enabled στο global configuration. Το plug-in λειτουργεί κατευθείαν στον html κώδικα και δεν χρειάζεται special tagging να προστεθεί από τον χρήστη .

**K XML-RPC (Blogger API & Joomla! API)**

Το XML-RPC" (Extensible Markup Language – Remote Procedure Call) Plug-in προσθέτει XML-RPC λειτουργικότητα στο Joomla

Το "XML-RPC Blogger API" επιτρέπει σε εφαρμογές τρίτων όπως w.blogger να επικοινωνούν με το Joomla, δίνοντας στον χρήστη την δυνατότητα να προσθέσει, διαγράψει, επεξεργαστεί και δημοσιοποιήσει content items από απόσταση.

Το XML-RPC Joomla! API" προσθέτει επιπλέον λειτουργικότητα στο Joomla με συμβατές εφαρμογές τρίτων.

**Templates**

- RHUK Milky Way
- Beez

**RHUK Milky Way**

Το RHUK Milky Way template είναι το προεπιλεγμένο template για την εγκατάσταση Joomla. Η καθαρή σχεδίαση αυτού του template το καθιστά πολύ ελαφρύ και γρήγορο.

Το template έχει τρεις παραμέτρους για να ρυθμίσει τα “Colour Variation,” “Background Variation,” και “Template Width.”

**Beez**

Το “Beez” template σχεδιάστηκε με συγκεκριμένη προσοχή για τον προεπιλεγμένο κώδικα, την προσβασιμότητα και την ευελιξία.

Αυτό το template είναι ένα παράδειγμα που παρουσιάζει τα νέα χαρακτηριστικά προσβασιμότητας στο Joomla 1.5 . Εξυπηρετεί σαν μια βάση και μπορεί να τροποποιηθεί και να επεκταθεί όσο θελήσει ο χρήστης.

## 2.2. Χαρακτηριστικά επισκόπησης του Joomla

Το Joomla είναι πολλά περισσότερο από ένα σύστημα διαχείρισης περιεχομένου ( CMS )

### User Management

Το Joomla έχει ένα σύστημα εγγραφών που επιτρέπει στους χρήστες να διαμορφώσουν προσωπικές επιλογές. Υπάρχουν εννέα group χρηστών με διαφόρους τύπους αδειών όσο αφορά στο που επιτρέπεται οι χρήστες να έχουν πρόσβαση, να μπορούν να διαμορφώσουν να δημοσιοποιήσουν ή να έχουν πρόσβαση διαχειριστή.

Η επικύρωση είναι ένα σημαντικό κομμάτι της διαχείρισης χρηστών, και το Joomla υποστηρίζει πολλαπλά πρωτόκολλα, συμπεριλαμβάνοντας τα LDAP, OpenID και ακόμα το Gmail. Αυτό επιτρέπει στους χρήστες να χρησιμοποιούν τις δικές τους υπάρχουσες πληροφορίες λογαριασμού για να βελτιώσουν την διαδικασία εγγραφής.

### Media Manager

Το media manager είναι το εργαλείο για την εύκολη διαχείριση αρχείων media ή φακέλων και μπορεί να διαμορφωθούν οι επιλογές mime type, για να χειριστεί οποιοδήποτε τύπο αρχείου. Το media manager είναι ενσωματωμένο στο article editor tool έτσι ώστε να μπορεί να συλλάβει εικόνες και άλλου τύπου αρχεία οποιαδήποτε στιγμή.

### Language Manager

Υπάρχει διεθνής υποστήριξη για πολλές γλώσσες και κωδικοποίηση UTF8 . Εάν ο χρήστης χρειάζεται το website σε μια γλώσσα και το administrator panel σε μια άλλη, οι πολλαπλές γλώσσες είναι δυνατές.

### Banner Management

Είναι εύκολο να τοποθετηθούν banners στο website χρησιμοποιώντας το banner manager, αρχίζοντας με την δημιουργία ενός προφίλ πελάτη. Εφόσον προστεθούν campaigns και όσα banner χρειάζονται , οι χρήστες μπορούν να θέτουν impression numbers, special urls και άλλα.

### Contact Management

Το contact manager βοηθάει τους χρήστες να βρουν το σωστό άτομο και τις πληροφορίες επικοινωνίας με αυτό. Επίσης υποστηρίζει πολλαπλές φόρμες επικοινωνίας που ανατρέχουν σε μεμονωμένα άτομα ή και σε ομάδες.

### Polls

Εάν θέλει ο διαχειριστής να μάθει περισσότερα για τους χρήστες μπορεί να δημιουργήσει εύκολα polls με πολλαπλές επιλογές.

### Search

Βοηθάει στο να κατευθύνει τους χρήστες στα περισσότερο γνωστά search items και παρέχει στον διαχειριστή search statistics.

## Web Link Management

Το να παρέχονται πόροι από Link στους χρήστες του site είναι απλό και μπορούν να ταξινομηθούν σε κατηγορίες, ακόμα και να μετρηθούν όλα τα κλικ.

## Content Management

Η απλή τοποθέτηση στη σειρά των άρθρων του Joomla κάνει θραύση στην οργάνωση του περιεχομένου . Μπορεί να οργανωθεί το περιεχόμενό με οποιοδήποτε τρόπο και όχι απαραίτητα με το πώς θα είναι στο website. Οι χρήστες του site μπορούν να βαθμολογήσουν άρθρα, να τα στείλουν με email σε κάποιο φίλο ή να τα σώσουνε αυτόματα σαν αρχείο pdf (με υποστήριξη UTF-8 για όλες τις γλώσσες ). Οι διαχειριστές μπορούν να αρχειοθετήσουν το περιεχόμενο για ασφάλεια, αποκρύπτοντας το από τους επισκέπτες του website.

Στα δημόσια website, το ενσωματωμένο email cloacking προστατεύει τις διευθύνσεις email από τα spam bots.

Το να δημιουργηθεί περιεχόμενο είναι εύκολο , με το **WYSIWYG editor.** Δίνοντας έτσι και στους αρχάριους χρήστες την δυνατότητα να συνδυάσουν κείμενο με εικόνες με έναν ωραίο τρόπο. Εφόσον δημιουργηθούν τα άρθρα, υπάρχει ένα πλήθος από προεγκατεστημένα modules που δείχνουν τα πιο δημοφιλή άρθρα, και τελευταία news items, newsflashes, related articles και αλλά.

## Syndication and Newsfeed Management

Με το Joomla είναι εύκολο να προστεθεί το περιεχόμενο του site σε κάποιο συνδικάτο, επιτρέποντας τους χρήστες να εγγραφούν σε καινούργιο περιεχόμενο στον αγαπημένο τους RSS reader. Είναι εξίσου εύκολο να ενσωματωθεί RSS Feeds από άλλες πηγές και να προστεθούν όλα στο site.

## Menu Manager

Το menu manager επιτρέπει την δημιουργία όσων menu / menu items επιθυμεί ο χρήστης . Μπορεί να δομηθεί μια ιεραρχία στα menu εντελώς ανεξάρτητη από την δομή του περιεχομένου . Ένα menu τοποθετείτε σε πολλαπλά μέρη και με οποιοδήποτε style. Χρησιμοποιούνται rollovers, dropdown, flyouts και οποιοδήποτε άλλο σύστημα πλοήγησης μπορείς να φανταστεί κανείς. Επίσης αυτόματα breadcrumbs παράγονται για να βοηθηθεί η πλοήγηση στους χρήστες του site.

## Template Management

Τα templates στο Joomla είναι ένας πανίσχυρος τρόπος να δείχνει ένα site έτσι όπως ακριβώς είναι το επιθυμητό, και είτε να χρησιμοποιηθεί ένα template για ολόκληρο το site ή ξεχωριστά templates για κάθε τομέα του site. Το επίπεδο του οπτικού έλεγχου πάει ένα βήμα παραπέρα, με ισχυρά templates, που επιτρέπουν την διαμόρφωση του κάθε κομματιού των σελίδων.

## Integrated Help System

Το Joomla έχει ένα ενσωματωμένο σύστημα βοήθειας έτσι ώστε να βοηθάει τους χρήστες να βρουν αυτό που ψάχνουν. Ένα γλωσσάρι εξηγεί τους όρους σε απλά αγγλικά , ένας version checker ελέγχει εάν χρησιμοποιείτε την τελευταία έκδοση, ένα system information tool βοηθάει με τα προβλήματα και αν όλα αυτά αποτύχουν συνδέεται σε πηγές online για περαιτέρω βοήθεια και υποστήριξη.

## System Features

Γρήγοροι χρόνοι φόρτωσης σελίδων είναι πιθανοί με την χρήση του page caching, granual level module caching και συμπίεση σελίδων με gzip.

Εάν ο διαχειριστής του συστήματος πρέπει να ανιχνεύσει λάθη σε ένα θέμα, το debugging mode και το error reporting είναι πολύτιμα.

Το στρώμα FTP επιτρέπει τις διαδικασίες αρχείων (όπως την εγκατάσταση προσθέτων ) χωρίς να πρέπει να κάνει όλους τους φακέλους και τα αρχεία εγγράψιμα, κάνοντας έτσι ευκολότερη την ζωή του διαχειριστή και αυξάνοντας την ασφάλεια της σελίδας.

Οι διαχειριστές επικοινωνούν γρήγορα και αποτελεσματικά με τους χρήστες ένας προς έναν μέσω προσωπικών μηνυμάτων, ή προς όλους μέσω του μαζικού συστήματος μηνυμάτων.

### Web Services

Με τις web services μπορούν να χρησιμοποιηθούν απομακρυσμένες κλήσεις διαδικασιών (μέσο HTTP και XML ). Μπορούν επίσης να ενσωματωθούν τα XML-RPC services με το blogger και το Joomla API`s.

### Powerful Extensibility

Αυτά είναι μόνο μερικά από τα βασικά χαρακτηριστικά γνωρίσματα του Joomla και η πραγματική δύναμη είναι στον τρόπο που προσαρμόζει ο χρήστης το Joomla.

## 2.3. Τεχνικές απαιτήσεις και JDBC

### Τεχνικές απαιτήσεις για Joomla 1.5.x

| Λογισμικό | Συνιστώμενο | Ελάχιστο | Αναφορές |
|---|---|---|---|
| **PHP** | **5.2+** | **4.3.10** | **http://www.php.net** |
| **MySQL** | **4.1.x+** | **3.23** | **http://www.mysql.com** |
| **Apache** ***(withmod_mysqlmod_xml and mod_zlib),*** | **2.x+** | **1.3** | **http://www.apache.org** |
| **Microsoft IIS** | **7** | **6** | **http://www.iis.net** |

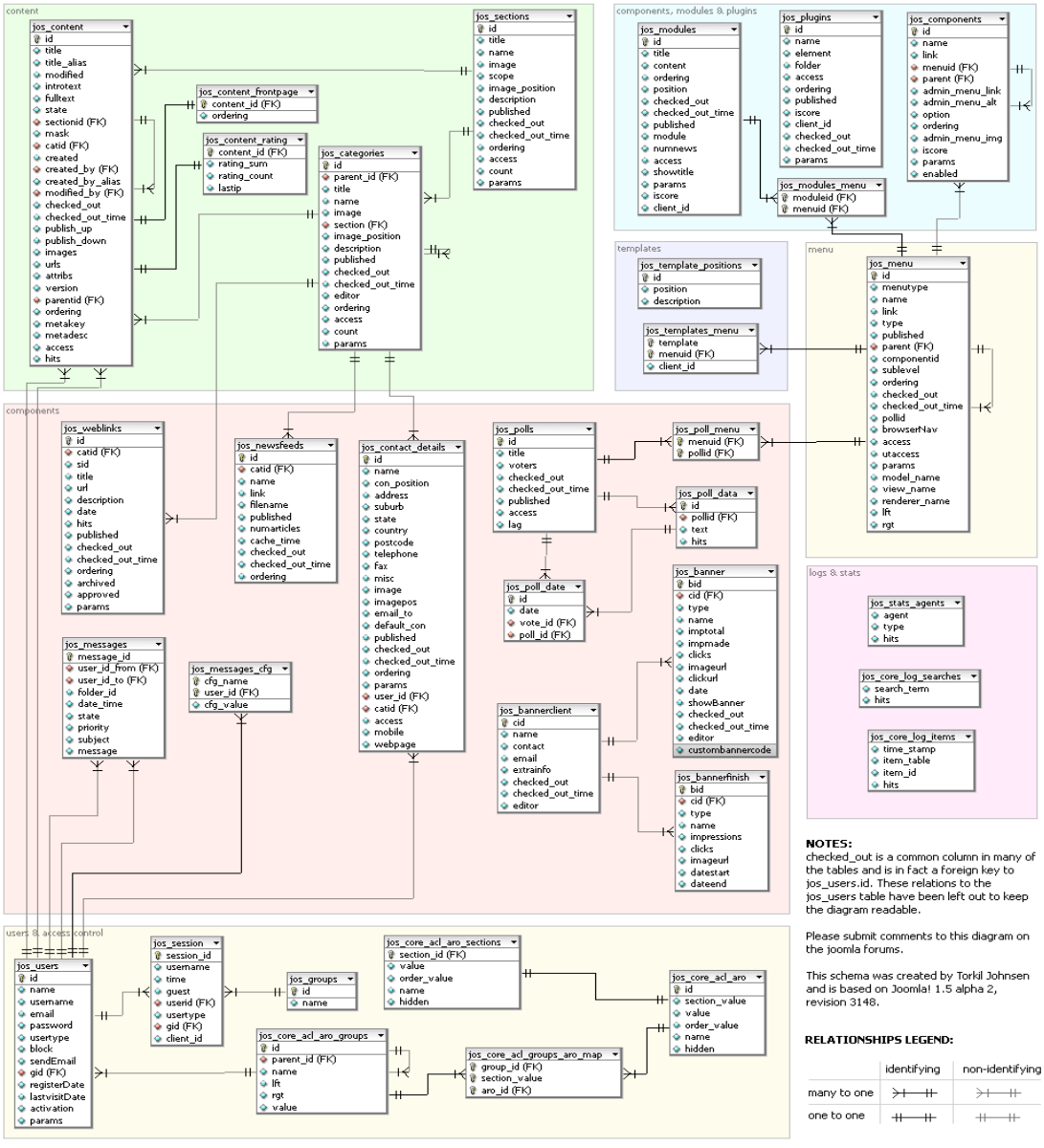
### The Java Database Connectivity (JDBC)

Java Database Connectivity (JDBC) είναι το βιομηχανικό πρότυπο για την ανεξάρτητη συνδεσιμότητα ανάμεσα στην γλώσσα προγραμματισμού java με μια ποικιλία βάσεων δεδομένων. και με πηγές συνοπτικών στοιχείων, όπως είναι τα spread sheets ή τα επίπεδα αρχεία. Το JDBC API παρέχει ένα API σε επίπεδο κλήσης για πρόσβαση σε βάσεις δεδομένων τύπου SQL.

Η τεχνολογία JDBC επιτρέπει να χρησιμοποιήσεις την γλώσσα προγραμματισμού JAVA προς εκμετάλλευση δυνατοτήτων ‘‘΄WRITE ONCE, RUN EVERYWHERE” για εφαρμογές που χρειάζονται πρόσβαση σε επιχειρησιακά δεδομένα. Με τους οδηγούς JDBC μπορούν να συνδέθουν όλα τα εταιρικά δεδομένα ακόμα και σε ένα ετερογενή περιβάλλον.

Παρακάτω φαίνεται το σχήμα της βάσης του Joomla, πίνακας 2-1.

**Πίνακας 2-1**



## 3. Περιγραφή του προβλήματος και μεθοδολογία προσέγγισης

Στην εφαρμογή που αναπτύχθηκε, διαχειριζόμαστε ένα site Joomla . Το Joomla αποθηκεύει τα δεδομένα του site σε μια βάση της MySQL μέσω κάποιον συναρτήσεων. Για να κάνουμε οποιαδήποτε τροποποίηση στα περιεχόμενα του Joomla site πρέπει να εκτελέσουμε κάποια query στην βάση δεδομένων . Δεν ασχολούμαστε καθόλου με τις συναρτήσεις του Joomla. Εξυπακούεται ότι πρέπει να συνδεθούμε σαν διαχειριστές της βάσης για να την αλλάξουμε. Έτσι φτιάξαμε μια εφαρμογή σε Java η οποία συνδέεται σαν διαχειριστής στην βάση δεδομένων, εκτελεί κάποια query, τροποποιώντας έτσι την βάση και το Joomla site

### 3. 1. Πίνακες και σύνδεση στην βάση δεδομένων

Το Joomla περιέχει μια πληθώρα πινάκων έτσι ώστε να μπορεί να προσφέρει δυνατότητες που προαναφέρθηκαν . Θα παρουσιαστούν ονομαστικά οι πίνακες , και θα αναλυθούν αυτοί τους οποίους χρησιμοποιήσαμε για την εργασία μας. Εικόνα 3-1.

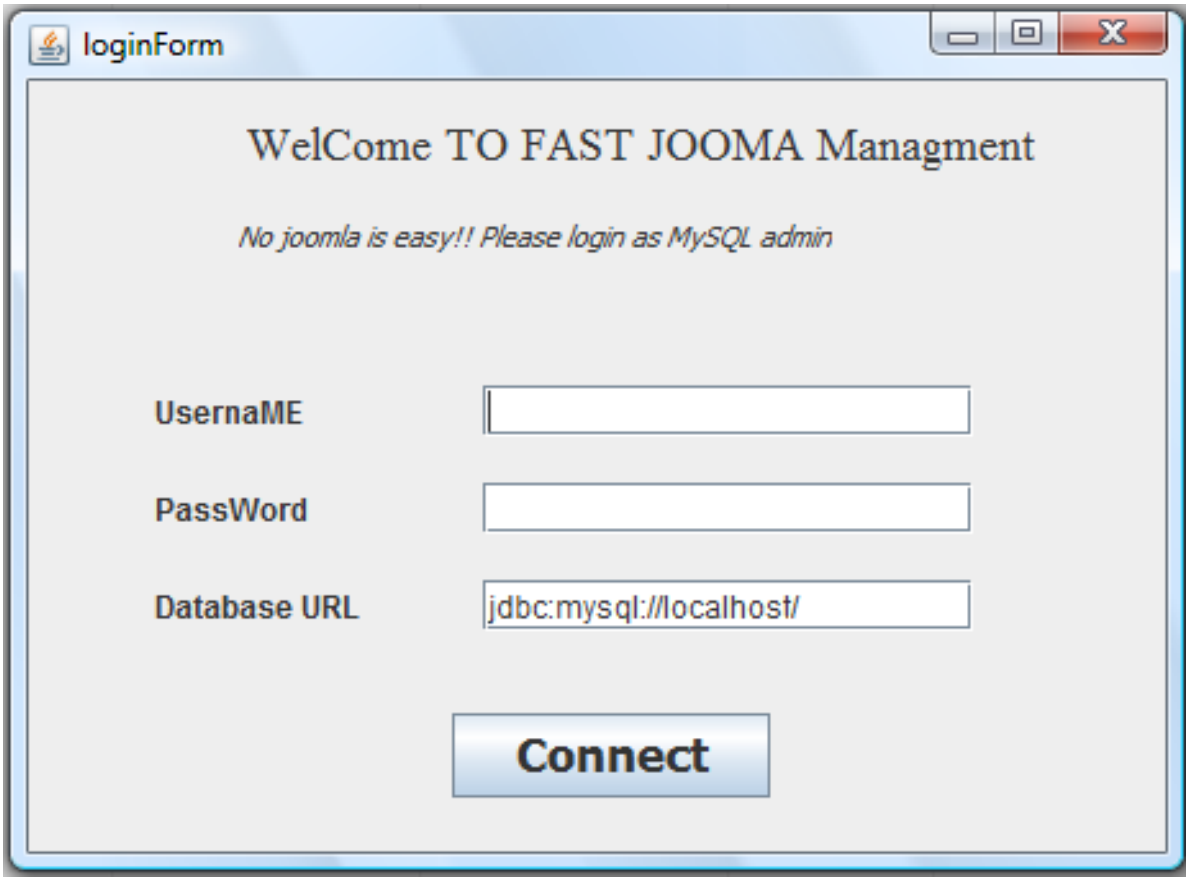
**Εικόνα 3-1**

- **jos_banner**
- **jos_bannerclient**
- **jos_bannertrack**
- **jos_categories**
- **jos_components**
- **jos_contact_details**
- **jos_content**
- **jos_content_frontpage**
- **jos_content_rating**
- **jos_core_acl_aro**
- **jos_core_acl_aro_groups**
- **jos_core_acl_aro_sections**
- **jos_core_acl_groups_aro_map**
- **jos_core_log_items**
- **jos_core_log_searches**
- **jos_groups**
- **jos_menu**
- **jos_menu_types**
- **jos_messages**
- **jos_messages_cfg**
- **jos_modules**
- **jos_modules_menu**
- **jos_newsfeeds**
- **jos_plugins**
- **jos_poll_data**
- **jos_poll_date**
- **jos_poll_menu**
- **jos_polls**
- **jos_sections**
- **jos_session**
- **jos_stats_agents**
- **jos_template_positions**
- **jos_templates_menu**
- **jos_users**
- **jos_weblinks**

Τα πεδία όλων των πινάκων θα αναφερθούν στο παράρτημα

**Login Form**

Μέσω αυτής της φόρμας συνδεόμαστε στην βάση μας, το πρόγραμμά μας τρέχει τοπικά. Εικόνα 3-2 Καλούμαστε να συμπληρώσουμε τα παρακάτω πεδία :




**Εικόνα 3-2: Φόρμα σύνδεσης**

**Username**: Το όνομα χρήστη του διαχειριστή

**Password**: Ο κωδικός του διαχειριστή

**Database Url**: Το πλήρες μονοπάτι της βάσης μας (συμπεριλαμβανομένου του ονόματος της βάσης)

Όταν ο χρήστης πατήσει το κουμπί Connect εκτελείται το παρακάτω κομμάτι κώδικα

```
Connector n1 = new
Connector(Connector.DATABASE_URL,textu.getText(),textp.getText());

if (n1.getConnectorStatus()==true)

{

     mtb = new MyTableModel(n1);

     new NewJFrame2(viewController, NewJFrame.mtb);

}

else

{

 JOptionPane.showMessageDialog(this,"Lathos stoixeia!!!");

 textu.setText(null);

 textp.setText(null);

}
```

Δημιουργείται ένα αντικείμενο τύπου **Connector** για να γίνει η σύνδεση στην βάση δεδομένων. Αν η σύνδεση πετύχει τότε δημιουργείται η επόμενη φόρμα και το **TableModel** για το συγκεκριμένο αντικείμενο αλλιώς εμφανίζεται ένα μήνυμα στην οθόνη μας που μας ενημερώνει ότι βάλαμε λάθος στοιχεία και τα textboxes καθαρίζονται.

Στον παραπάνω κώδικα χρησιμοποιήθηκε η κλάση Connector η οποία είναι υπεύθυνη για την σύνδεση στην βάση.

```
public class Connector {

    static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";

    static final String DATABASE_URL =
       "jdbc:mysql://localhost/books?zeroDateTimeBehavior=convertToNull";

    static Connection connection;

    private boolean isConnected = false;
```

```
public Connector(String DATABASE_URL, String username, String password)
{
    try
    {
        Class.forName("com.mysql.jdbc.Driver");
        connection = DriverManager.getConnection(DATABASE_URL, username,
         password);
        isConnected=true;
        System.out.println("You are sucesfully connected");
    }
    catch(SQLException sqlException)
    {
           System.out.println("Sorry lathos stoixeia");
    }
    catch(ClassNotFoundException classNotFound)
    {
        classNotFound.printStackTrace();
        System.exit(1);
    }
 }//Connector()
```

Η κλάση μας περιέχει τις απαραίτητες μεταβλητές για να χρησιμοποιήσουμε το JDBC καθώς και μια μεταβλητή **isConnected** που μας δείχνει την κατάσταση της σύνδεσης

Αυτή η μεταβλητή επιστρέφεται μέσω της μεθόδου **getConnectorStatus()** και ελέγχεται για να δούμε αν η σύνδεση έγινε επιτυχώς

Επίσης χρησιμοποιείται η κλάση **MyTableModel** η οποία έχει δύο αναφορές μία στην κλάση **Connector** και μια στην **SqlGroup**. Η δεύτερη είναι και υπεύθυνη για τα ερωτήματα sql

```
public class MyTableModel extends AbstractTableModel
{
    Connector connector;
    SqlGroup sqlgroup;
```

Χρησιμοποιείται ένας από τους δομητές της και δίνει τιμές στις αντίστοιχες μεταβλητές.

```
public MyTableModel(Connector  c1)
    {
        connector=c1;
        sqlgroup = new SqlGroup(connector);
    }
```

Επίσης αυτή η κλάση κάνει extend το **AbstractTableModel** και πρέπει να υλοποιήσει κάποιες μεθόδους

```
public Class getColumnClass(int column) throws IllegalStateException
public int getColumnCount() throws IllegalStateException
public String getColumnName(int column) throws IllegalStateException
public int getRowCount()throws IllegalStateException
public Object getValueAt(int row, int column) throws IllegalStateException
```

Έχει προστεθεί επίσης μια μέθοδος για να εκτελείται ένα Sql query μέσω αυτής της κλάσης.

```
public void Executefromsql(String query)
{
   try
```

```
  {
      sqlgroup.stmtExec(query);
      fireTableStructureChanged();
  }
  catch(SQLException sqlException)
  {
      sqlException.printStackTrace();
      System.exit(1);
  }
}
```

Βλέπουμε ότι για το αντικείμενο τύπου SqlGroup που μόλις δημιουργήθηκε, καλείτε η μέθοδος **stmtExec** την οποία θα δούμε παρακάτω . Παρατηρούμε από την δομή του προγράμματος ότι χρησιμοποιούνται πολλές κλάσεις και όχι μια, με βασική αρχή ότι κάθε κλάση κάνει κάτι πολύ συγκεκριμένο.

Παρακάτω θα δούμε την κλάση SQLGroup

```
public class SqlGroup {

    Connector c1;

    private Statement statement;
    private ResultSet resultSet;
    private ResultSetMetaData metaData;
    private int numOfRows;
```

Έχει μια αναφορά σε ένα αντικείμενο **Connector** και τα απαραίτητα μέλη για να εκτελεστεί ένα **SqlQuery**

```
public Statement stmtExec(String query) throws SQLException,
IllegalStateException

{

   try

   {

   statement =
Connector.connection.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);

            resultSet = statement.executeQuery(query);

            metaData = resultSet.getMetaData();

            resultSet.last();

            numOfRows = resultSet.getRow();

            resultSet.first();

   }

            catch(SQLException sqlException)

   {

             sqlException.printStackTrace();

             System.exit(1);

   }


        return statement;

}
```

Και την μέθοδο **stmtExec ()** που εκτελεί το query, γεμίζει το **resultset**, παίρνει κάποια **metadata** τα οποία θα χρειαστούν στην κλάση **MyTableModel** για να υλοποιηθούν κάποιες μέθοδοι και επιστρέφει ένα αντικείμενο τύπου **statement**

## 3. 2 Χρήστες και φόρμα χρηστών

Το Joomla παρέχει τα ακόλουθα επίπεδα χρηστών

Registered, Author, Editor, Publisher, Manager, Administrator, SuperAdministrator με διαφορετικά επίπεδα πρόσβασης για κάθε τύπο χρήση . Οι πληροφορίες για τους χρήστες αποθηκεύονται κυρίως στον πίνακα jos_users.Πίνακας 3-2.

**jos_users**

**Πίνακας 3-2**

**Fields**

| Field | Type | Collation | Null | Key | Default | Extra | Privileges |
|---|---|---|---|---|---|---|---|
| Id | int(11) | NULL | | PRI | (NULL) | auto_increment | select,insert,update,references |
| name | text | utf8_general_ci | | MUL | | | select,insert,update,references |
| username | varchar(150) | utf8_general_ci | | | | | select,insert,update,references |
| email | varchar(100) | utf8_general_ci | | | | | select,insert,update,references |
| password | varchar(100) | utf8_general_ci | | | | | select,insert,update,references |
| usertype | varchar(75) | utf8_general_ci | | MUL | | | select,insert,update,references |
| block | tinyint(4) | NULL | | | 0 | | select,insert,update,references |
| sendEmail | tinyint(4) | NULL | YES | | 0 | | select,insert,update,references |
| Gid | tinyint(3) unsigned | NULL | | | 1 | | select,insert,update,references |
| registerDate | datetime | NULL | | | 0000-00-00 00:00:00 | | select,insert,update,references |
| lastvisitDate | datetime | NULL | | | 0000-00-00 00:00:00 | | select,insert,update,references |
| activation | varchar(100) | utf8_general_ci | | | | | select,insert,update,references |
| params | text | utf8_general_ci | | | | | select,insert,update,references |

Θα μας απασχολήσουν τα πεδία : name,username, email, password, usertype, gid, registerDate, lastvisitDate, params

**Name**: Περιέχει το όνομα του χρήστη.

**Username:** Περιέχει το ψευδώνυμο το οποίο ο χρήστης θα χρησιμοποιεί για να κάνει login στο σύστημα.

**Email:** Περιέχει το email του χρήστη .

**Password:** Περιέχει τον κωδικό του χρήστη, το κείμενο που θα εισαχτεί σαν κωδικός δεν είναι απλό κείμενο, αλλά πέρνα μέσα από μια συνάρτηση κωδικοποίησης MD5 και μετατρέπετε σε μια ακολουθία χαρακτήρων.

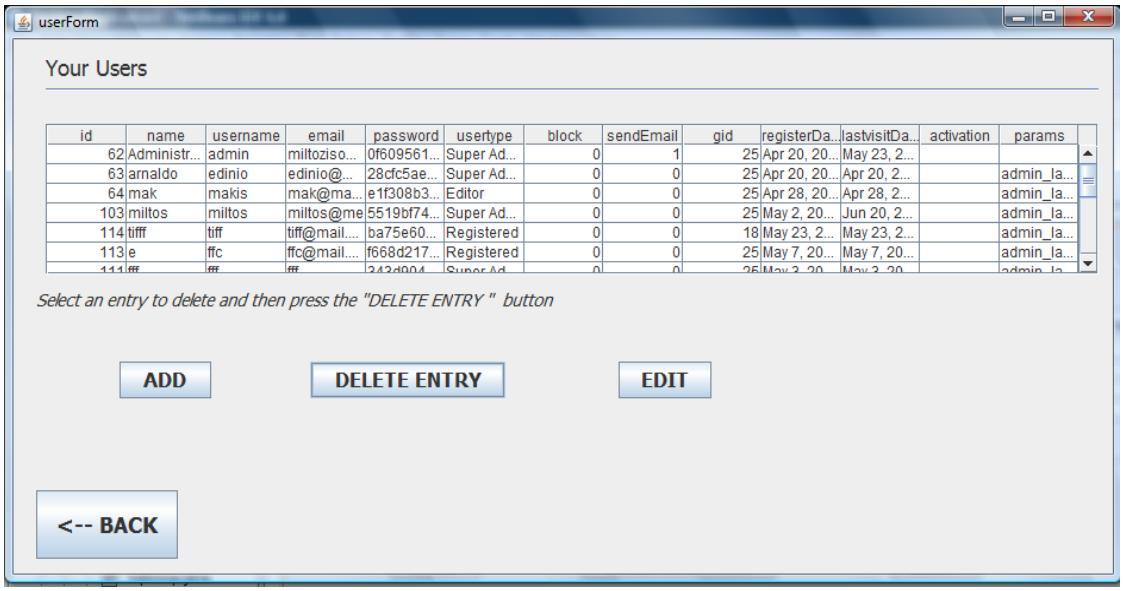
**Usertype:** Το είδος του χρήστη, μπορεί να είναι ανάμεσα σε (Registered Author, Editor, Publisher, Manager, Administrator, Super Administrator).

**RegisterDate:** Η μέρα που έκανε εγγραφή στο site.

**LastvisitDate:** Η τελευταία φορά που έκανε login.

**Params:** Κάποιοι παράμετροι.

Η φόρμα διαχείρισης των χρηστών θα είναι όπως η ακόλουθή. Εικόνα 3-3




**Εικόνα 3-3**

Βλέπουμε ότι μας παρέχονται πληροφορίες για τους χρήστες (μέσω ενός query στον πίνακα jos_users) όπως το ID της εγγραφής τους στην βάση, το username, το email, ο τύπος χρήστη, ο κωδικός του και άλλα .

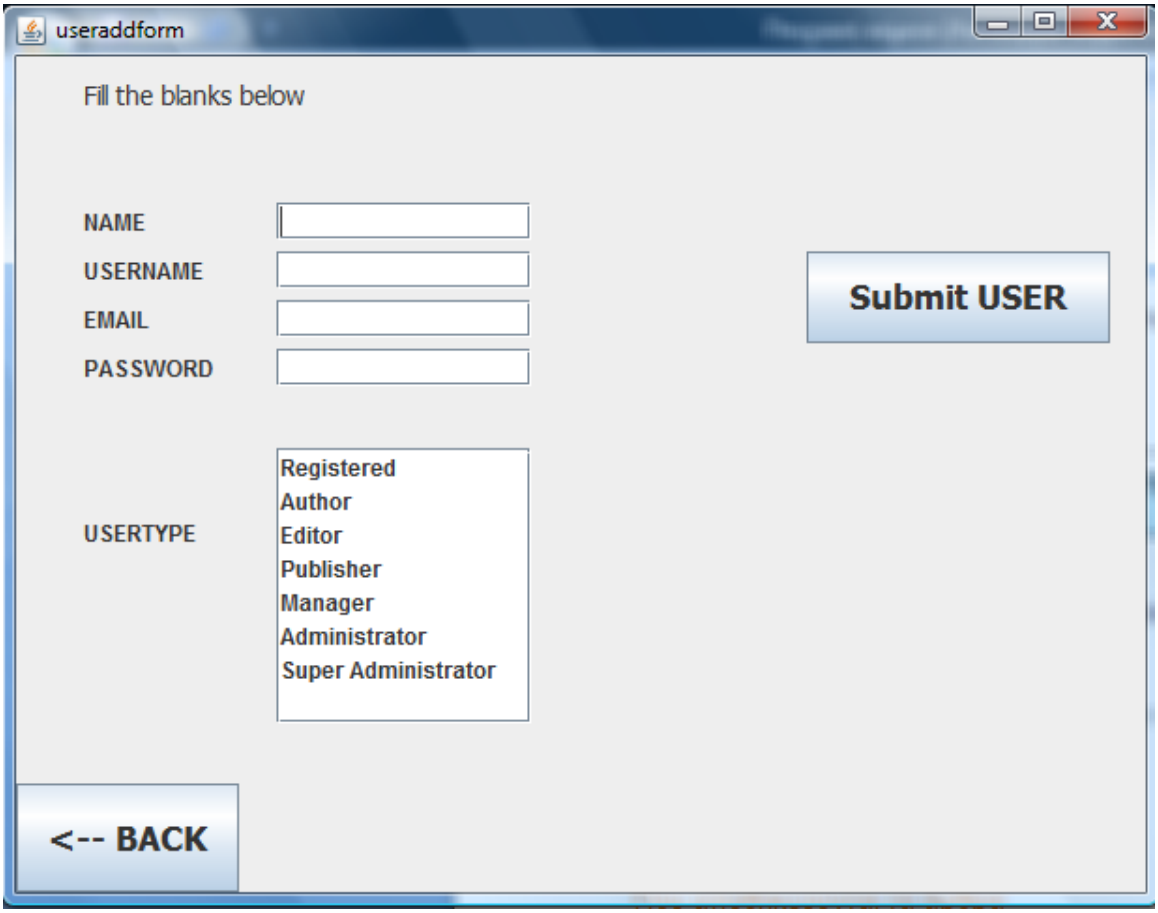
Παρατηρούμε τρία μεγάλα κουμπιά στην φόρμα τα **ADD, DELETE ENTRY**, **EDIT** . Αυτά παρέχουν μια λειτουργικότητα στην εφαρμογή μας για να κάνουμε εισαγωγή ενός χρήστη στην βάση, να σβήσουμε η να τροποποιήσουμε κάποια εγγραφή .

Τα περιεχόμενα του πίνακα έχουν καθοριστεί από μας τρέχοντας κάποιο συγκεκριμένο query .

```
NewJFrame.mtb.Executefromsql("SELECT * FROM jos_users");
```

Παρατηρούμε ότι τρέχουμε ένα query για το static αντικείμενο της **NewJFrame** το **mtb**

Επιλέγοντας το κουμπί Add θα μεταφερθούμε στην φόρμα προσθήκης χρηστών. Εικόνα 3-4




**Εικόνα 3-4**

Εδώ καλούμαστε να συμπληρώσουμε όλα τα απαραίτητα στοιχεία για να προσθέσουμε επιτυχώς έναν χρήστη στην βάση μας, Κάποια πεδία είναι απαραίτητα και κάποια άλλα όχι, αν πατήσουμε το κουμπί **Submit User** και κάποιο πεδίο που χρειάζεται δεν είναι συμπληρωμένο θα εμφανιστεί κατάλληλο μήνυμα και το **query** μας δεν θα εκτελεστεί, επίσης αν ο χρήστης ξεχάσει να επιλέξει μια τιμή από το usertype καταχωρείτε στο σύστημα σαν Registered (χρήστης με τα χαμηλότερα δικαιώματα) . Ο έλεγχος για το αν τα πεδία είναι συμπληρωμένα γίνετε μέσο της μεθόδου **checkRequired() .**

```
public void  checkRequired()
{
       String t=nameField.getText();
       String t1=userNameField.getText();
       String t2=userEmailField.getText();

       if(t.isEmpty())
            JOptionPane.showMessageDialog(this,"User must have a name!!");
       else  if(t1.isEmpty())
        JOptionPane.showMessageDialog(this,"User must have a username!!");
       else  if(t2.isEmpty())
            JOptionPane.showMessageDialog(this,"User must have a valid
             Email!!");
       else if(t2.contains("@")==false)
             JOptionPane.showMessageDialog(this,"Sorry invalid Email, please
             try again!!!!");

       if((!t.isEmpty()) && (!t1.isEmpty()) && (!t2.isEmpty()  &&
             t2.contains("@")))
           canSubmit=true;
 }
```

Εάν η μέθοδός μας εκτελεστεί επιτυχώς τότε η μεταβλητή canSubmit γίνεται true και μπορούμε να εκτελέσουμε το query μας.

```
private void SubmButtonActionPerformed(java.awt.event.ActionEvent evt)
{
    checkRequired();
    if(  canSubmit==true)
    {
      try
```

```
        {

        //bazei to xristi sto pinaka jos USers

         String S="INSERT INTO jos_users(name,username, email, password,
         usertype, gid, registerDate, lastvisitDate, params) VALUES
         ('"+nameField.getText()+"', '"+userNameField.getText()+"',
         '"+userEmailField.getText()+"', MD5('"+userPassField.getText()+"'),
         '"+numToGroup(listSelection)+"', " +gradeSelection+", CURDATE(),
         CURDATE(), 'admin_language=language=editor=helpsite=timezone=0')";

        //ton dilonei sto jos_core_acl_aro

        //edo xreiazomai to id tis egrafis, to opio megalonei mono
         tou(autoincrement) gia afto upotheto oti

        //to username einai monadiko kai brisko to id pou thelo

        String S2="INSERT INTO jos_core_acl_aro(section_value,value,name)
         SELECT 'users', id, name FROM jos_users WHERE id =(SELECT id FROM
         jos_users WHERE username='"+userNameField.getText()+"')";

        //to idio kai sto jos_core_acl_groups_aro_map to gradeSelection einai
         i timi tou

        //Id apo to jos_core_acl_aro_groups kai dilonei ton tupo tou group
         (p.x Super Administrator=25)

         String S3="INSERT INTO  jos_core_acl_groups_aro_map (group_id,
         aro_id) SELECT '"+gradeSelection+"', id FROM jos_core_acl_aro
         WHERE id = (SELECT id FROM jos_core_acl_aro WHERE
         name='"+nameField.getText()+"')";

             mtb.sqlgroup.getStatement().executeUpdate(S);

             mtb.sqlgroup.getStatement().executeUpdate(S2);

             mtb.sqlgroup.getStatement().executeUpdate(S3);


             mtb.Executefromsql("SELECT * FROM jos_users");

             viewController.selectViews("userForm");

}
```

Ο κώδικας που εκτελείται όταν πατηθεί το κουμπί μοιάζει κάπως έτσι. Δημιουργούνται τρία διαφορετικά strings καθώς για να προστεθεί ένας χρήστης πρέπει να τροποποιηθούν τρείς πίνακες.

```
INSERT INTO jos_users(name,username, email, password, usertype, gid, registerDate, lastvisitDate, params) VALUES ('"+nameField.getText()+"', '"+userNameField.getText()+"', '"+userEmailField.getText()+"', MD5('"+userPassField.getText()+"'), '"+numToGroup(listSelection)+"', "+gradeSelection+", CURDATE(), CURDATE(), 'admin_language=language=editor=helpsite=timezone=0')
```

Στο πρώτο query θα κάνει μια εισαγωγή στον πίνακα jos_users με τις απαραίτητες τιμές στα απαραίτητα πεδία . Έπειτα θα προστεθεί εγγραφή στον πίνακας jos_core_acl_aro.

---

```
INSERT INTO jos_core_acl_aro(section_value,value,name) SELECT 'users', id, name FROM jos_users WHERE id =(SELECT id FROM jos_users WHERE username='"+userNameField.getText()+"')
```

Και τέλος στον jos_core_acl_groups_aro_map.

---

```
INSERT INTO  jos_core_acl_groups_aro_map (group_id, aro_id) SELECT '"+gradeSelection+"', id FROM jos_core_acl_aro WHERE id = (SELECT id FROM jos_core_acl_aro WHERE name='"+nameField.getText()+"')";
```

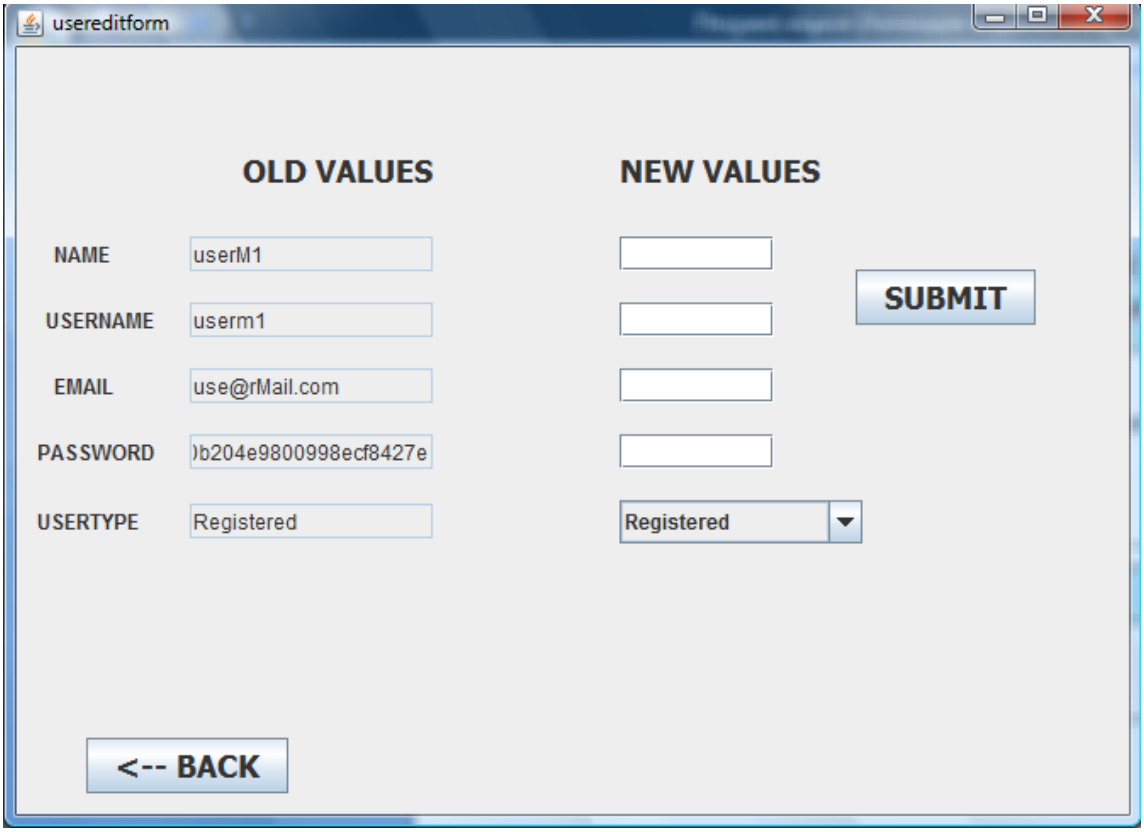
Για να επιστρέψουμε τον πίνακα στην αρχική του μορφή θα εκτελέσουμε ένα query της μορφής.

---

```
mtb.Executefromsql("SELECT * FROM jos_users");
```

---

Πατώντας το κουμπί EDIT θα μεταφερθούμε στην φόρμα τροποποίησης χρηστών, απαραίτητη προϋπόθεση για να κάνουμε EDIT είναι να διαλέξουμε μια εγγραφή από τον αρχικό πίνακα, αλλιώς θα εμφανιστεί μήνυμα ότι πρώτα πρέπει να επιλέξουμε εγγραφή. Εικόνα 3-5.




**Εικόνα 3-5**

Αν ο χρήστης επιλέξει το κουμπί SUBMIT χωρίς να κάνει κάποια αλλαγή θα του εμφανιστεί σχετικό μήνυμα .Στα αριστερά του παραθύρου βλέπουμε τις παλιές τιμές και δεξιά τα textbox που θα συμπληρώσουμε τις νέες,στο κάτω δεξιό μέρος το Combobox περιέχει τις δυνατές τιμές του usertype και

επιλέγετε δυναμικά η τιμή που έχει ο χρήστης κατά την αρχικοποίηση της φόρμας.

Εφόσον έχουν συμπληρωθεί τα πεδία που θα αλλαχτούν, γίνετε ένας έλεγχος για το αν οι τιμές είναι ίδιες με τις παλιές, και όσες τιμές αλλαχτούν τόσες θα προστεθούν στο query που θα εκτελεστεί, ο έλεγχος γίνετε μέσο της μεθόδου **checkUpdate().**

```
public String checkUpdate(){
       String s="UPDATE jos_users SET ";
        String first="";


    if(!newname.getText().isEmpty()==true)
    {
        if(newname.getText().equals(oldname.getText())==false)
        {
            fireit=true;
            changename=true;
            if(first.isEmpty()==true)
            {
                first=newname.getName();
                    s=s+" name='"+newname.getText()+"'";
            }
            else
                s=s+", name='"+newname.getText()+"'";
        }
    }
      if(newuser.getText().isEmpty()==false)
      {
```

```
            if(newuser.getText().equals(olduser.getText())==false)
```

Η μέθοδος είναι αρκετά μεγάλη για αυτό παρουσιάζεται ένα κομμάτι της μόνο.

```
if(first.isEmpty()==true) //ean den proeigite tpt allo prin
{
   if(newusertype!=oldusertype)
   {
       fireit=true;
        s=s+" usertype='"+newusertype+"'";
        first="something"; // apla na gemisei, gia na min einai null
    }
   else
        fireit=false;
   }
else
{
    fireit=true;
        if(newusertype!=oldusertype)
        s=s+", usertype='"+newusertype+"'";
}
        s=s+" WHERE id='"+id+"'";
       return s;
}
```

Επιστρέφει ένα string που θα περιέχει το query με τις αλλαγές, προς εκτέλεση.

.

Το **DELETE BUTTON** για να το πατήσουμε θα πρέπει πρώτα να επιλέξουμε μια εγγραφή που θέλουμε να διαγράψουμε, Εκτελούνται διαδοχικά τα παρακάτω queries.

---

```
DELETE FROM jos_core_acl_groups_aro_map WHERE aro_id=(SELECT id FROM jos_core_acl_aro WHERE value='" +ob+"')";
```

---

```
DELETE FROM jos_core_acl_aro WHERE value="+ob
```

---

```
DELETE FROM jos_users WHERE id="+ob
```

Ο χρήστης πλέον έχει διαγραφεί από το σύστημα επιτυχώς .

Και ΠΑΝΤΑ επιστρέφετε ο πίνακας στην αρχική του μορφή.

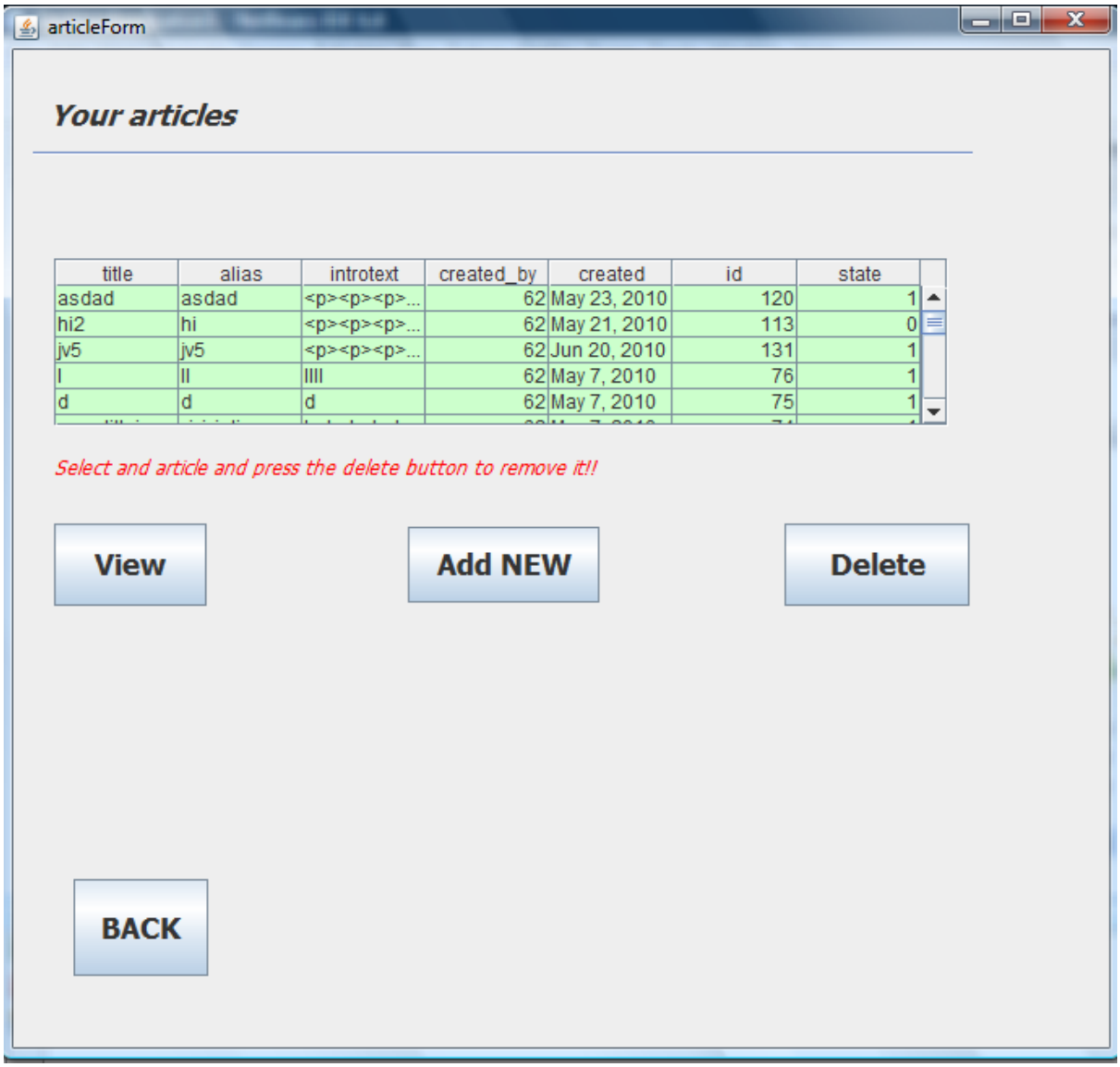
---

```
mtb.Executefromsql("SELECT * FROM jos_users");
```

---

### 3. 3. Άρθρα και φόρμα άρθρων

Επιλέγοντας το κουμπί για να δούμε τα διαθέσιμα άρθρα, μεταφερόμαστε στην φόρμα των άρθρων. Εικόνα 3-6 .




**Εικόνα 3-6**

Μέσω αυτής της φόρμας μας παρέχονται πληροφορίες για τα άρθρα, ο τίτλος του, το alias από ποιόν δημιουργήθηκε και πότε, το id της εγγραφής στον πίνακα και η κατάσταση δημοσίευσης του άρθρου.

Που αποθηκεύονται τα άρθρα

---

Όλα τα άρθρα του Joomla αποθηκεύονται στον πίνακα jos_content της MYSQL .Παρακάτω παρατίθενται τα πεδία του πίνακα. Πίνακας 3-7.

**jos_content**

**Πίνακας 3-7**

**Fields**

| Field | Type | Collation | Null | Key | Default | Extra | Privileges |
|---|---|---|---|---|---|---|---|
| Id | int(11) unsigned | NULL | | PRI | (NULL) | auto_increment | select,insert,update,references |
| Title | text | utf8_general_ci | | | | | select,insert,update,references |
| title_alias | text | utf8_general_ci | | | | | select,insert,update,references |
| introtext | mediumtext | utf8_general_ci | | | | | select,insert,update,references |
| Fulltext | mediumtext | utf8_general_ci | | | | | select,insert,update,references |
| State | tinyint(3) | NULL | | MUL | 0 | | select,insert,update,references |
| sectionid | int(11) unsigned | NULL | | MUL | 0 | | select,insert,update,references |
| Mask | int(11) unsigned | NULL | | MUL | 0 | | select,insert,update,references |
| Catid | int(11) unsigned | NULL | | MUL | 0 | | select,insert,update,references |
| created | datetime | NULL | | | 0000-00-00 00:00:00 | | select,insert,update,references |
| created_by | int(11) unsigned | NULL | | | 0 | | select,insert,update,references |
| created_by_alias | text | utf8_general_ci | | | | | select,insert,update,references |
| modified | datetime | NULL | | | 0000- | | select,insert,update,referenc |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | 00-00 00:00:00 | | es |
| modified_by | int(11) unsigned | NULL | | | 0 | | select,insert,update,references |
| checked_out | int(11) unsigned | NULL | | MUL | 0 | | select,insert,update,references |
| checked_out_time | datetime | NULL | | | 0000-00-00 00:00:00 | | select,insert,update,references |
| publish_up | datetime | NULL | | | 0000-00-00 00:00:00 | | select,insert,update,references |
| publish_down | datetime | NULL | | | 0000-00-00 00:00:00 | | select,insert,update,references |
| images | text | utf8_general_ci | | | | | select,insert,update,references |
| Urls | text | utf8_general_ci | | | | | select,insert,update,references |
| Attribs | text | utf8_general_ci | | | | | select,insert,update,references |
| version | int(11) unsigned | NULL | | | 1 | | select,insert,update,references |
| parentid | int(11) unsigned | NULL | | | 0 | | select,insert,update,references |
| ordering | int(11) | NULL | | | 0 | | select,insert,update,references |
| metakey | text | utf8_general_ci | | | | | select,insert,update,references |
| metadesc | text | utf8_general_ci | | | | | select,insert,update,references |
| access | int(11) unsigned | NULL | | MUL | 0 | | select,insert,update,references |
| Hits | int(11) unsigned | NULL | | | 0 | | select,insert,update,references |
| metadata | text | utf8_general_ci | | | | | select,insert,update,references |

Κατά την εισαγωγή που έγινε από το πρόγραμμα, μας ενδιαφέρουν περισσότερο τα εξής πεδία : Title, Alias, Introtext, State, Sectionid,Catid.

**Title :** Περιέχει τον τίτλο του άρθρου.

**Alias :** Περιέχει το alias του άρθρου.

**Introtext :** Περιέχει το κείμενο που γράφουμε μέσα στο άρθρο.

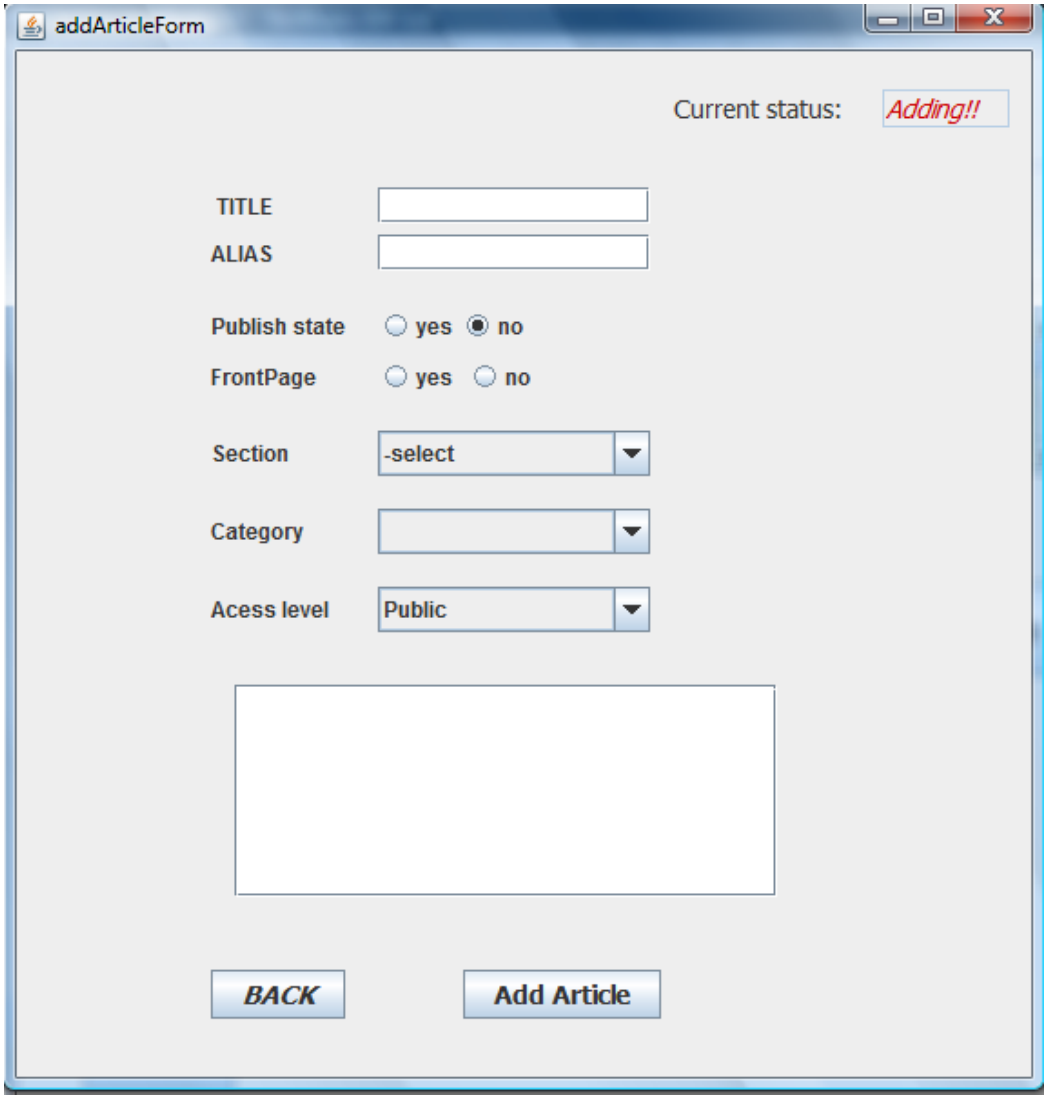
**State:** Δείχνει την κατάσταση του άρθρου.

```
Published = 0 not publish
Published = 1 publish
Published = -2 delete/trash
```

**Sectionid:** Περιέχει το id του πίνακα jos_sections, δηλαδή σε ποιο section ανήκει το παρόν άρθρο.

**Catid:** Περιέχει το id του πίνακα jos_categories, δηλαδή σε ποιο category ανήκει το παρόν άρθρο.

**Access:** Δείχνει το επίπεδο πρόσβασης στο άρθρο (Public, Registered, Special).




**Εικόνα 3-8**

Η παραπάνω φόρμα, εικόνα 3-8 εμφανίζετε όταν πατηθεί το κουμπί **ADD NEW**, στην πάνω δεξιά γωνία μας δείχνει την τρέχων κατάσταση, υπάρχουνε πεδία τίτλου, alias, κατάσταση δημοσίευσης και πρωτοσέλιδο . Έπειτα ακολουθούνε 3 ComboBox στα όποια θα διαλέξουμε το section, category που θα ανήκει το άρθρο καθώς και το access level του. Όταν αρχικοποιηθεί η φόρμα εκτελείτε ένα query και γεμίζει το section combobox με τα διαθέσιμα sections, παρακάτω φαίνετε η μέθοδος που είναι υπεύθυνη για αυτό .

```
public void dynamicSections()
{
    try
    {
        String ids="SELECT title FROM  `jos_sections`";
        mtb.sqlgroup.stmtExec(ids);
        ResultSet rstemp1 = mtb.sqlgroup.getResultSet();
        rstemp1.first();

       while(rstemp1.next())
        {
            jComboBox2.addItem( rstemp1.getString(1));
        }
        mtb.Executefromsql("SELECT title, alias, introtext, created_by,
             created, id, state FROM jos_content");
    }
    catch(SQLException exc)
    {
        exc.printStackTrace();
    }
}
```

Πατώντας στο πρώτο Combobox. το section καλείται η μέθοδος **dynamicCategories(sectiontitle)** όπου **sectiontitle** η αντίστοιχη επιλογή του section και γεμίζει το category combobox με τα categories του εκάστοτε section.

```
  public void dynamicCategories(String s)
  {
```

```
eventfirebysoftware=true;
eventfirebysoftware2=true;
try
{
  String ids="SELECT title FROM jos_categories WHERE section=(SELECT
            id FROM jos_sections WHERE title='"+s+"')";
  mtb.sqlgroup.getStatement().executeQuery(ids);
  ResultSet rstemp1= mtb.sqlgroup.getStatement().executeQuery(ids);
  jComboBox3.removeAllItems();

  while (rstemp1.next())
  {
       jComboBox3.addItem( rstemp1.getString(1));
  }
  if(stateEdit==1)
  {
      String ids2="SELECT title FROM jos_categories WHERE id=(SELECT
                  catid FROM jos_content WHERE id='"+select+"')";
      mtb.sqlgroup.getStatement().executeQuery(ids2);
   ResultSet rstemp2= mtb.sqlgroup.getStatement().executeQuery(ids2);

   System.out.println("Dynamixcategories: Mesa apo  state edit");
   if(rstemp2.first())
   {
       jComboBox3.setSelectedItem(rstemp2.getString(1));
       oldcategory=rstemp2.getString(1);
       System.out.println("I timi tou old category gia to arhtoro
                          einai "+oldcategory);
   }
```

Πατώντας το κουμπί **Add Article** για να καταχωρήσουμε το καινούργιο άρθρο, εκτελείται η μέθοδος **checkRequired()** που ελέγχει ένα έχουν συμπληρωθεί όλα τα απαραίτητα πεδία της φόρμας,

```
public void checkRequired()

{

    String t=jTextFieldTitle.getText();

    String t1=jTextArea1.getText();

    if(t.isEmpty())

            JOptionPane.showMessageDialog(this,"Your article must have a
                                                title!!");

    else  if(t1.isEmpty())

            JOptionPane.showMessageDialog(this,"Your article must have some
                                                text in it!!");

    else  if(jComboBox2.getSelectedObjects().length==0)

            System.out.println("You must set a section /n");


   if((!t.isEmpty()) && (t!=oldtitle) && (!t1.isEmpty())  &&
      (!jComboBox2.getActionCommand().isEmpty() ))

          canSubmit=true;

   else

          canSubmit=false;

}
```

Εάν είναι όλα εντάξει τότε η μεταβλητή canSubmit γίνεται true και μπορεί να εκτελεστεί ο κώδικας για την εισαγωγή.

Άρα για να προστεθεί μια εγγραφή στον πίνακα jos_content θα εκτελεστεί ένα query της μορφής:

```
"INSERT INTO `books`.`jos_content` (`id`, `title`, `alias`, `title_alias`,
`introtext`, `fulltext`, `state`, `sectionid`, `mask`, `catid`, `created`,
```

```
`created_by`, `created_by_alias`, `modified`, `modified_by`, `checked_out`, `checked_out_time`, `publish_up`, `publish_down`, `images`, `urls`, `attribs`, `version`, `parentid`, `ordering`, `metakey`, `metadesc`, `access`, `hits`, `metadata`) VALUES (NULL, '"+jTextFieldTitle.getText()+"', '"+jTextFieldAlias.getText()+"', 'keimeno titlou', '<p>"+jTextArea1.getText()+"</p>', 'keimeno sto fulltext', '"+published+"', '"+section_id+"', '1', '"+category_id+"', NOW(), '62', '','0000-00-00 00:00:00', '0', '62','0000-00-00 00:00:00', NOW(),'0000-00-00 00:00:00', '', '', 'show_title=" +
        "link_titles=" +
        "show_intro=" +
        "show_section=" +
        "link_section=" +
        "show_category=" +
        "link_category=" +
        "show_vote=" +
        "show_author=" +
        "show_create_date=" +
        "show_modify_date=" +
        "show_pdf_icon=" +
        "show_print_icon=" +
        "show_email_icon=" +
        "language=" +
        "keyref=" +
        "readmore='" +
        "', '1', '0', '0', '', '', '"+accesslevel+"', '0', '')";
```

Ο χρήστης καθώς προσθέτει ένα άρθρο έχει την δυνατότητα να επιλέξει αν αυτό θα εμφανίζετε στο FrontPage (πινάκας jos_content_frontpage)

Περιεχόμενα του πίνακα jos_content_frontpage

**jos_content_frontpage**

**Πίνακας 3-9**

**Fields**

| Field | Type | Collation | Null | Key | Default | Extra | Privileges |
|---|---|---|---|---|---|---|---|
| content_id | int(11) | NULL | | PRI | 0 | | select,insert,update,references |
| ordering | int(11) | NULL | | | 0 | | select,insert,update,references |

**Indexes**

Το πεδίο **content_id** περιέχει το id του άρθρου (jos_content) που θα φαίνεται στο FrontPage.Το πεδίο **ordering** περιέχει έναν «αύξοντα αριθμό», κάθε φορά που θα γίνεται μια εισαγωγή στον πίνακα πρέπει εκείνη η εισαγωγή να πάρει την τιμή 1 με προϋπόθεση ότι οι παλαιότερες εγγραφές θα προσαυξηθούν κατά μια μονάδα έτσι ώστε να μην υπάρχουν εγγραφές με ίδια τιμή . Πίνακας 3-9 .

Έτσι πρώτα θα αυξηθούν όλες τις τιμές κατά 1, ώστε οι τιμές να ξεκινούνε από το 2 πλέον.

```
UPDATE jos_content_frontpage SET ordering = ordering + 1 WHERE ordering > 0"
```

Και έπειτα θα γίνει η εισαγωγή στον πινάκα jos_content_frontpage.

```
INSERT INTO `books`.`jos_content_frontpage` (`content_id`,`ordering`)
```

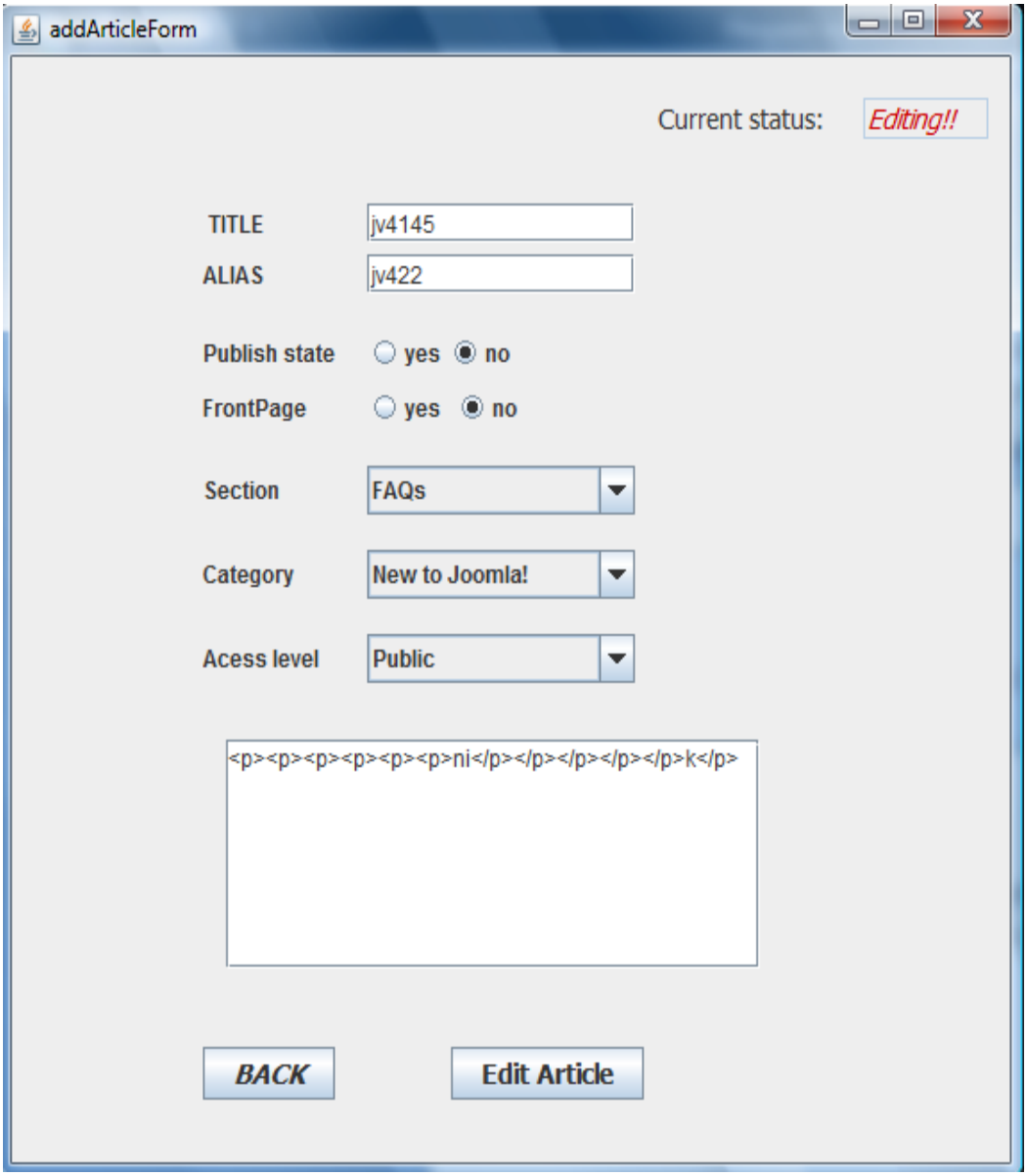
```
VALUES ('"+id+"','1')
```

Η μεταβλητή id είναι το id τις τελευταίας εγγραφής, αυτής δηλαδή που μόλις έγινε. Επειδή όμως την στιγμή που εκτελείτε το query δεν γνωρίζουμε την τιμή του id της εγγραφής που κάνουμε, πρέπει κάπως να ανακτηθεί αυτήν η τιμή. Παρατηρείται ότι το πεδίο id toy jos_content έχει προκαθορισμένη τιμή NULL και ενεργοποιημένη την ιδιότητα auto_increment . Αρά η τελευταία εγγραφή είναι αυτή που μόλις έγινε, δηλαδή η εγγραφή με το μεγαλύτερο id.

```
maxid="SELECT MAX( id ) FROM  `jos_content`";
```

```
while (rstemp.next())
        {
          id = rstemp.getInt(1);
          System.out.println("This record`s ID is =" + id);
        }
```

Πατώντας το κουμπί VIEW από την φόρμα με τα άρθρα θα μας επιτρέψει να διαβάσουμε ένα άρθρο και ταυτόχρονα να τα κάνουμε και edit εάν θελήσουμε. Εικόνα 3-10.




**Εικόνα 3-10**

Υπάρχουνε μέθοδοι που δυναμικά επιλέγουν και βάζουν τιμές στα Combobox και στα radio buttons . πχ η μέθοδος **setPublish()** βάζει δυναμικά την τιμή του published του άρθρου που επιλέξαμε στο radio button.

```
//Bazei tin timi published tou arthrou pou anoiksame, stin forma pou tha
emfanistei
    public void setPublish(Object x)
    {
        if(x.equals(1))
        {
           jRadioYes.setSelected(true);
           jRadioNo.setSelected(false);
           published=1;
           oldpublishstate=1;
           System.out.println("Oldpublished state : "+oldpublishstate);
        }
        else
        {
            jRadioYes.setSelected(false);
            jRadioNo.setSelected(true);
            published=0;
            oldpublishstate=0;
            System.out.println("Oldpublished state : "+oldpublishstate);
        }
    }
```

Και ταυτόχρονα αρχικοποιεί τις μεταβλητές published, oldpublishedstate για τον έλεγχο στο edit αργότερα .

Η μέθοδος **setFrontpage()** αλλάζει την κατάσταση του radio button με το να εκτελέσει ένα query στον πίνακα jos_content_frontpage έτσι ώστε να ελέγξει αν υπάρχει εγγραφή στον πίνακα για το συγκεκριμένο id, πράγμα που θα σημαίνει ότι το άρθρο έχει γίνει published.

```
public void setFrontpage(Object x)

{

    try

    {

         String checkfpage="SELECT content_id FROM jos_content_frontpage
                              WHERE content_id="+x+"";

         mtb.sqlgroup.getStatement().executeQuery(checkfpage);

         ResultSet rstempme=
  mtb.sqlgroup.getStatement().executeQuery(checkfpage);


         if(rstempme.first())

         {

            jRadioFYes.setSelected(true);

            jRadioFNo.setSelected(false);

            frontpage=1;

            oldfrontpagestate=1;

         }

         else

         {

            jRadioFYes.setSelected(false);

            jRadioFNo.setSelected(true);

            frontpage=0;

            oldfrontpagestate=0;

         }


         NewJFrame.mtb.Executefromsql("SELECT title, alias, introtext,
                created_by, created, id, state FROM jos_content");
```

Για να βάλουμε τιμή στο τρίτο Combobox access καλούμε την μέθοδο **setAccessMenu()** η οποία ανάλογα με την πρόσβαση επιλέγει και τιμή για το combobox. Επειδή η τιμή του access για το κάθε άρθρο δεν εμφανίζεται στον πίνακα των άρθρων που βλέπουμε στην πρώτη φόρμα, δεν μπορούμε να την περάσουμε σαν παράμετρο σε αυτήν την φόρμα, άρα πρέπει να εκτελέσουμε ένα query που να μας την επιστρέψει .

```
public void setAccessmenu(Object ob){
    try
      {
            String ids="SELECT access FROM jos_content WHERE id='"+ob+"'";
            mtb.sqlgroup.getStatement().executeQuery(ids);
            ResultSet rstemp1=
                        mtb.sqlgroup.getStatement().executeQuery(ids);

            if(rstemp1.first())
            {
               if(rstemp1.getInt(1)==0)
               {
                   oldaccesslevel=accesslevel=0;
                   jComboBox4.setSelectedIndex(0);
               }
               else if( rstemp1.getInt(1)==1)
               {
                   oldaccesslevel=accesslevel=1;
                   jComboBox4.setSelectedIndex(1);
               }
               else
               {
                   oldaccesslevel=accesslevel=2;
                   jComboBox4.setSelectedIndex(2);
               }
```

```
            System.out.println("access from set access menu is:
                                "+rstemp1.getString(1));


         }

        mtb.Executefromsql("SELECT title, alias, introtext, created_by,
                            created, id, state FROM jos_content");

    }
```

Για να πάρει τιμή το Combobox με το section καλούμε την μέθοδο **setSectionMenu()** η οποία για συγκεκριμένο id toy πίνακα jos_content βρίσκει το sectionid της εγγραφής, πηγαίνει στον πίνακα jos_sections και βρίσκει το αντίστοιχο title του.

```
//arixikopoei to combobox toy section analoga me to dosmeno id

    public void setSectionmenu(String s)

    {

       try

       {

            String ids="SELECT title FROM jos_sections WHERE id=(SELECT
                        sectionid FROM jos_content WHERE id='"+s+"')";

            mtb.sqlgroup.getStatement().executeQuery(ids);

            ResultSet rstemp1=
                        mtb.sqlgroup.getStatement().executeQuery(ids);


            if(rstemp1.first())

            {

                sectiontitle=oldsection=rstemp1.getString(1);// LPE

            }

            jComboBox2.setSelectedItem(oldsection);


        mtb.Executefromsql("SELECT title, alias, introtext, created_by,
                            created, id, state FROM jos_content");

     }
```

```
        catch(SQLException exc)
        {
            exc.printStackTrace();
        }
    }
```

Εδώ χρειάζεται προσοχή η γραμμή!!

```
jComboBox2.setSelectedItem(oldsection);
```

Κάθε φόρα που εκτελείται μια τέτοια εντολή οπουδήποτε μέσα στον κώδικα , αμέσως ενεργοποιείται το event του συγκεκριμένου combobox για το οποίο κλήθηκε η **setSelectedItem()** .Αυτός είναι και ο λόγος που στην φόρμα EDIT δεν γίνεται πουθενά αναφορά σε μία μέθοδο που βάζει categories.

Αυτός είναι ο κώδικας που εκτελείται όταν πατηθεί το κουμπί **VIEW**

```
if(canEdit==true)
        {
         NewJFrame7 njf7 =new NewJFrame7(viewController,NewJFrame.mtb,1,ob);
             njf7.setTextArea(selection[2]);
             njf7.setTextField(selection[1]);
             njf7.setTextField2(selection[0]);
             njf7.setPublish(temp);
             njf7.setFrontpage(ob);
             njf7.setAccessmenu(ob);
             njf7.setSectionmenu(ob.toString());

             viewController.selectViews("addArticleForm");
        }
        else
             JOptionPane.showMessageDialog(this,"Please select an entry to
                                                 View/Edit!!");
```

Έτσι μέσω της μεθόδου **setSectionMenu()** αλλάζουμε το section και ταυτόχρονα επειδή εκτελείται το event της μεθόδου καλείται η μέθοδος **dynamicCategories()**, που εκτός από το να γεμίσει με τα category titles ελέγχει αν είμαστε σε κατάσταση edit , εκτελεί ένα query που για το catid του άρθρου, βρίσκει από  τον πίνακα jos_categories τον τίτλο του και τον βάζει στο combobox.

```
if(stateEdit==1)
{
     String ids2="SELECT title FROM jos_categories WHERE id=(SELECT catid
                    FROM jos_content WHERE id='"+select+"')";
     mtb.sqlgroup.getStatement().executeQuery(ids2);
     ResultSet rstemp2= mtb.sqlgroup.getStatement().executeQuery(ids2);

     if(rstemp2.first())
     {
          jComboBox3.setSelectedItem(rstemp2.getString(1));
          oldcategory=rstemp2.getString(1);
          System.out.println("I timi tou old category gia to arhtoro einai
                              "+oldcategory);
     }
}
```

Για να αποφύγουμε την διαδικασία που κάθε φόρα που θα κάνουμε **setSelectedItem()** σε ένα combobox θα εκτελεί και τον κώδικα του event, μπορούμε να κάνουμε μια μεταβλητή Boolean  eventfirebysoftware όπως και έγινε στο πρόγραμμα μας. Όταν μια μέθοδος θα εκτελεί εντολή **setSelectedItem()**, η μεταβλητή eventfirebysoftware θα γίνετε true και το event θα καταλαβαίνει ότι δεν πρέπει να εκτελεστεί  κάνοντας τον εξής έλεγχο.

```
if(eventfirebysoftware)

{

        eventfirebysoftware=false;

return;   }
```

Ελέγχει εάν είναι true και αν είναι την κάνει false (για επαναχρησιμοποίησή της) και μετά εκτελεί ένα return με αποτέλεσμα να μην εκτελεστεί ο κώδικας που υπάρχει στο event.

Εάν τώρα ο χρήστης θέλει να καταχωρήσει την τροποποιημένη του εγγραφή και πατήσει το Add article button, εκτελούνται οι παρακάτω μέθοδοι:

```
  System.out.println("-------EKTELESI-------");

        catid(category);

        secid(sectiontitle);

        checkRequired();

        checkEdit();
```

Θα τις εξετάσουμε μια προς μια, Πρώτα καλείται ι **catid()** με όρισμα category, η μεταβλητή αυτή παίρνει τιμή όταν εκτελούμε το event από το category combobox.

```
private void jComboBox3ActionPerformed(java.awt.event.ActionEvent evt) {
      if(eventfirebysoftware)
       {
```

```
            eventfirebysoftware=false;
            return;
        }

        JComboBox jc = (JComboBox)evt.getSource();
       category = (String)jc.getSelectedItem();
       System.out.println("Your click selected category is: "+
                    category);
    }
```

Η οποία τιμή όμως είναι ο τίτλος του category, ενώ εμείς για να κάνουμε μια καταχώρηση στην βάση μας θέλουμε το id για αυτόν τον τίτλο, για αυτό καλούμε την μέθοδο **catid()**

```
public void catid(String s){
  try
  {
      String ids="SELECT id FROM jos_categories WHERE title='"+s+"'";
      mtb.sqlgroup.getStatement().executeQuery(ids);
      ResultSet rstemp1= mtb.sqlgroup.getStatement().executeQuery(ids);
       if(rstemp1.first())
                category_id= rstemp1.getInt(1);
      mtb.Executefromsql("SELECT title, alias, introtext, created_by,
             created, id, state FROM jos_content");
  }
  catch(SQLException exc)
  {
      exc.printStackTrace();
  }
}
```

Η μέθοδος βρίσκει το id εκτελώντας ένα query και το βάζει στην μεταβλητή category_id.

Μετά καλείται η μέθοδος **secid()** με όρισμα sectiontitle, η μεταβλητή αυτή παίρνει τιμή όταν εκτελεστεί το event από το combobox του section

```
canSection=true;
JComboBox jc = (JComboBox)evt.getSource();
sectiontitle = (String)jc.getSelectedItem();
System.out.println("Your click selected section is : "+sectiontitle);

dynamicCategories(sectiontitle);

public void secid(String s)
{
        try
        {
            String ids="SELECT id FROM jos_sections WHERE title='"+s+"'";
            mtb.sqlgroup.getStatement().executeQuery(ids);
            ResultSet rstemp1=
                        mtb.sqlgroup.getStatement().executeQuery(ids);

            if(rstemp1.first())
                section_id= rstemp1.getString(1);

              mtb.Executefromsql("SELECT title, alias, introtext,
                    created_by, created, id, state FROM jos_content");
        }
        catch(SQLException exc)
        {
            exc.printStackTrace();
        }
    }
```

Για τον δοσμένο τίτλο εκτελεί ένα **query** στον πίνακα jos_sections βρίσκει το συγκεκριμένο id και το βάζει στην μεταβλητή section_id η οποία μετά μπορεί να χρησιμοποιηθεί για να εκτελεστούν **queries** στην βάση.

Η μέθοδος **checkRequired()** εξετάστηκε νωρίτερα.

Μετά καλείται η μέθοδος **checkedit()** αυτή αναλαμβάνει να ελέγξει εάν μπορούμε να κάνουμε edit, και ποιές θα είναι οι τιμές του πίνακα που θα πάρουν μέρος. Η όλη διαδικασία βασίζεται σε μια σύγκριση παλιών και νέων τιμών που αρχικοποιούνται σε διάφορα στάδια της φόρμας, Όταν εμφανίζεται η φόρμα μπροστά μας, έχουν πάρει τιμή όλες οι oldxxxxx μεταβλητές και κάθε φόρα που κάνουμε μια επιλογή παίρνουν τιμή και οι νέες μεταβλητές, μέσα στην μέθοδο γίνεται σύγκριση εάν είναι ίδιες , αν δεν είναι τότε ενεργοποιείται η μεταβλητή canEdit και στέλνεται το string που περιέχει τις μεταβλητές που παίρνουν μέρος προς εκτέλεση . Παρακάτω παρουσιάζεται ένα μέρος της μεθόδου

```
public void checkEdit()
    {
        String t=jTextFieldTitle.getText();
        String t1=jTextFieldAlias.getText();
        String t2=jTextArea1.getText();
        String first="";
        editquerry="UPDATE jos_content SET ";

        if(t.equals(oldtitle))//title OK
            System.out.println("To title einai idio");
        else
        {
            if (first.isEmpty())
            {
                first="something"; //etsi gia na gemisei mono
```

```
                editquerry+="title='"+t+"'";
            }
            else
                editquerry+=", title='"+t+"'";
        }
//-----------------------------------------
        if( t1.equals(oldalias))//alias OK
            System.out.println("To alias einai idio");
        else
        {
            if (first.isEmpty())
            {
                first="something"; //etsi gia na gemisei mono
                editquerry+="alias='"+t1+"'";
             }
```

Η μέθοδος είναι αρκετά μεγάλη για αυτό παρουσιάζετε ένα μέρος της.

```
  if(sectiontitle.equals(oldsection))//sections
      System.out.println("Ta sections einai idia");
  else
  {
            if (first.isEmpty())
            {
                first="something"; //etsi gia na gemisei mono
                secid(sectiontitle);
                editquerry+="sectionid='"+section_id+"'";
            }
            else
                editquerry+=", sectionid='"+section_id+"'";
  }
   editquerry+=", modified=NOW() WHERE id='"+select+"'";
```

```
if(frontpage==oldfrontpagestate)

        System.out.println("ta frontpage einai isa");

else

{

       System.out.println("ta frontpage DEEEN einai isa");

       if(first.isEmpty())

            editquerry="UPDATE jos_content SET modified=NOW() WHERE
                        id='"+select+"'";

       else

           first="something";

}

     if (t.equals(oldtitle)==false || t1.equals(oldalias)==false ||
     published!=oldpublishstate || frontpage!=oldfrontpagestate ||
     category.equals(oldcategory)==false || oldaccesslevel!=accesslevel ||
     oldareatext.equals(t2)==false
     ||sectiontitle.equals(oldsection)==false)

     {

           canEdit=true;

     }

     else

     {

           canEdit=false;

     }

    System.out.println("To can edit mesa apo to checkEdit einai :
                        "+canEdit);

}
```

Όλες αυτές η μέθοδοι που χρησιμοποιήθηκαν με τα string που συνεχώς αυξάνονται δουλεύουν κάπως έτσι : Στην αρχή δημιουργείτε ένα άδειο string το οποίο θα χρησιμοποιηθεί για να γίνει έλεγχος αν μια μεταβλητή αλλάζει πρώτη ή έχει προηγηθεί μια άλλη αλλαγή. Δηλώνεται ένα string στο οποίο θα γίνουν οι προσθέσεις των άλλων string.

```
editquerry="UPDATE jos_content SET ";
```

Τώρα το πρόβλημά μας είναι αν η μεταβλητή που αλλάζουμε θα είναι η πρώτη ή θα έχει προηγηθεί και άλλη αλλαγή . Εάν έχει συμβεί το δεύτερο τότε στο string που θα προστεθεί πρέπει να μπει κόμμα (,) μπροστά του. Εδώ χρησιμοποιείται η πρώτη άδεια μεταβλητή που δηλώθηκε στην αρχή . Κάθε φόρα που γίνετε οτιδήποτε ελέγχετε αυτήν η μεταβλητή, αν είναι άδεια τότε προφανώς η μεταβλητή που αλλάχτηκε θα είναι και η πρώτη που αλλάχτηκε στην φόρμα, θα προστεθεί μια οποιαδήποτε τιμή μέσα στην μεταβλητή αυτή έτσι ώστε οι μετέπειτα έλεγχοι να γνωρίζουν ότι προηγείται κάτι πριν από αυτούς και να βάλουν το κόμμα. Στο τέλος μετά όλες τις αλλαγές που θα γίνουν θα προστεθεί στο string το

```
editquerry+=", modified=NOW() WHERE id='"+select+"'";
```

Λόγω μιας ιδιαιτερότητας που προκύπτει από την δομή του προγράμματος μας, λόγω της μεταβλητής frontpage, δηλώνουμε ένα εναλλακτικό query το οποίο θα εκτελεστεί έαν για κάποιο λόγο γίνει η canEdit true και δεν έχει αλλάξει κάτι.

Τότε θα σταλεί το

```
editquerry="UPDATE jos_content SET modified=NOW() WHERE id='"+select+"'";
```

Εφόσον εκτελεστούν οι παραπάνω μέθοδοι και γίνουν όλοι οι απαραίτητοι έλεγχοι ανάλογα με τις τιμές κάποιων μεταβλητών θα εκτελεστούν και κάποια κομμάτια κώδικα .

```
 //------------EDIT state--------------
if(stateEdit==1)
{
  if(canEdit==true)
  {
      try
      {
         System.out.println(editquerry);
         mtb.sqlgroup.getStatement().executeUpdate(editquerry);
         System.out.println("mesa sto can edit meta to editquerry");
         //sto rstemp2 tha elenkso an uparxei eggrafi ston pinaka
            jos_content_frontpage, ean uparxei den tha tin peirakso
         String checkfpage="SELECT content_id FROM jos_content_frontpage
                         WHERE content_id="+select+"";
         mtb.sqlgroup.getStatement().executeQuery(checkfpage);
         ResultSet rstemp2=
                   mtb.sqlgroup.getStatement().executeQuery(checkfpage);

         //exei patisei na balei frontpage gia to id=select tha elenskso
            ean uparxei
         //ean uparxei den tha kano tpt
               if(frontpage==1)
               {
                   //edo an den uparxei eggrafi den mpainei katholou
                  if(rstemp2.first())
                  {
                     //den kano tpt
                  }
                  else
```

Ελέγχουμε ένα είμαστε σε κατάσταση edit και εκτελούμε τα query, μετά ελέγχετε αν έχει γίνει το frontpage==1, γιατί δεν περιέχετε κάποια αναφορά στον πινάκα jos_content έαν το άρθρο έχει γίνει front-paged, αλλά μόνο εγγραφές στον πίνακα jos_content_frontpage, για αυτό τις ελέγχουμε.

```
  else

  {      //ean den uparxei tha balo tin eggrafi sto jos_content_frontpage,
             afou prota auksiso kata 1

        //tis proigoumenes egraffes, xrisimopio os id to select giati ksero
             poia eggrafi allazo (tin epeleksa ston pinaak)


        String update_frontpage="UPDATE jos_content_frontpage SET ordering =
                                  ordering + 1 WHERE ordering > 0";


        mtb.sqlgroup.getStatement().executeUpdate(update_frontpage);


       String insert="INSERT INTO `books`.`jos_content_frontpage`
              (`content_id`,`ordering`) VALUES ('"+select+"','1')";

       mtb.sqlgroup.getStatement().executeUpdate(insert);}

     }

else

{

 //ean den exei patisei to frontpage, dld to frontpage=0!! tha prepei na do
             an uparxei i egrafi ston pinakka

 //an uparxei ti sbino, alios ola ok

        if(rstemp2.first())

        {

           String update_frontpage="UPDATE jos_content_frontpage SET ordering
                                  = ordering - 1 WHERE ordering > 0";


        mtb.sqlgroup.getStatement().executeUpdate(update_frontpage);



           String remove_fp="DELETE FROM jos_content_frontpage WHERE
                          content_id='"+select+"'";

           mtb.sqlgroup.getStatement().executeUpdate(remove_fp);

        }
```

```
 }
//eketelo to querry gia na epanafero to JTable stin arxiki tou morfi!!
mtb.Executefromsql("SELECT title, alias, introtext, created_by, created,
                        id, state FROM jos_content");
viewController.selectViews("articleForm");
```

Εάν έχει πατηθεί το toggle button για να γίνει off το frontpage πρέπει να σβηστεί και από τον πινάκα και να επιστρέψουν οι εγγραφές του πίνακα σε μια σωστή κατάσταση.

```
UPDATE jos_content_frontpage SET ordering = ordering - 1 WHERE ordering > 0";
```

Εάν στην αρχική φόρμα πατηθεί το DELETE Button, αφού επιλέγει μια εγγραφή τότε θα σβηστεί το άρθρο.

```
//Delete button
 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    if(canEdit==true)
    {
       try
       {
           String s="DELETE FROM jos_content WHERE id="+ob;
           mtb.sqlgroup.getStatement().executeUpdate(s);

           mtb.Executefromsql("SELECT title, alias, introtext, created_by,
             created, id, state FROM jos_content");
       }
       catch(SQLException sqle)
       {
           sqle.printStackTrace();
       }
    }
    else
```

```
        JOptionPane.showMessageDialog(this,"Please select an entry to
                                          Delete!!");
}
```

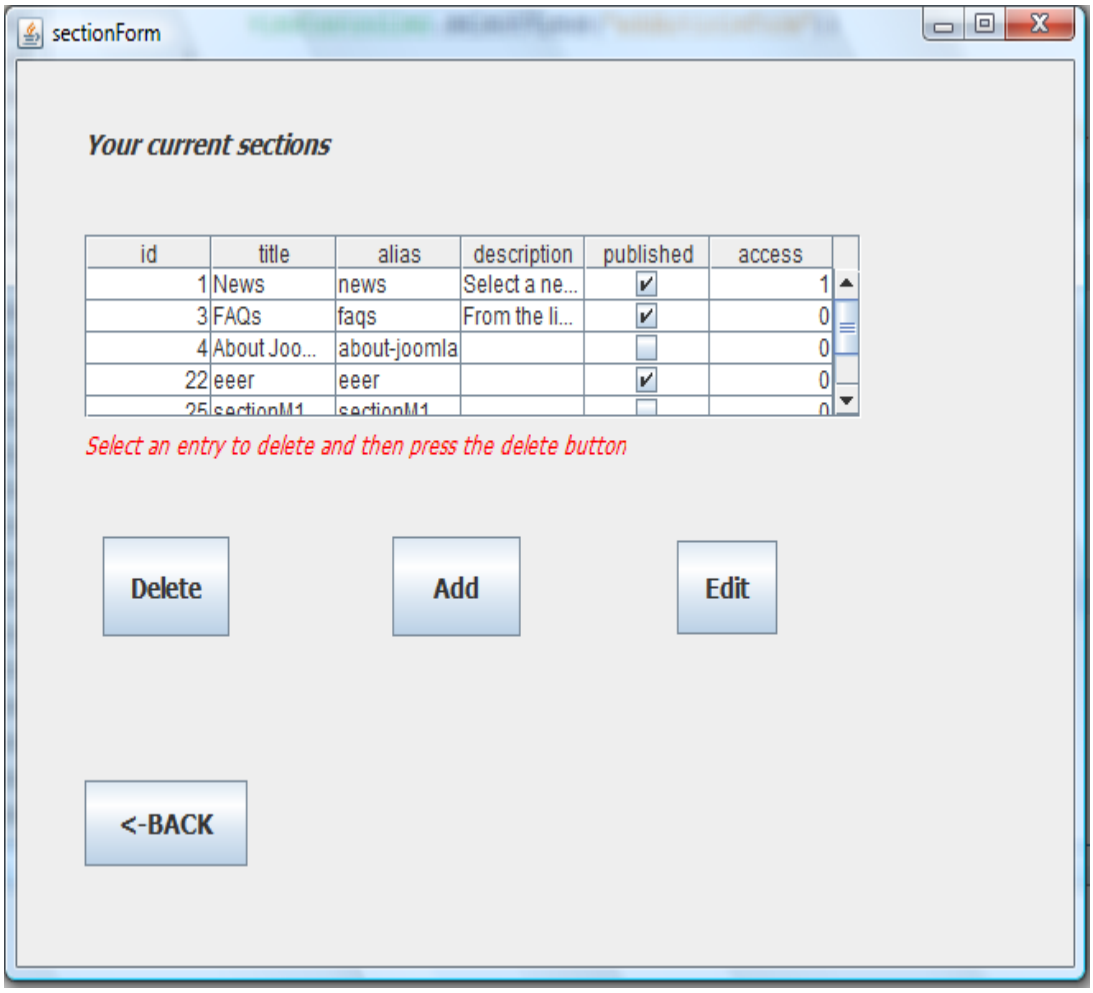
Πάντα πρέπει να επιστρέφετε το JTable στην αρχική του κατάσταση.

Έχουν χρησιμοποιηθεί κάποιες μεταβλητές από την αρχική φόρμα, οι οποίες και περαστήκαν στις add,edit form . Αυτό έγινε με το να αποθηκευτούν κάποιες στήλες του πίνακα για συγκεκριμένη γραμμή .Αυτό εκτελείται κάθε φόρα που επιλέγω μια νέα γραμμή στον πίνακα.

```
//CLICK SE PINAKA
     private void jTable1MouseClicked(java.awt.event.MouseEvent evt) {
     canEdit=true;
     selection[0]=(String)jTable1.getValueAt(jTable1.getSelectedRow(),0);
     selection[1]=(String)jTable1.getValueAt(jTable1.getSelectedRow(),1);
     selection[2]=(String)jTable1.getValueAt(jTable1.getSelectedRow(),2);
     temp=jTable1.getValueAt(jTable1.getSelectedRow(),6);
     ob=jTable1.getValueAt(jTable1.getSelectedRow(),5);

     System.out.println("Article`s id: "+ob);
```

### 3. 4. Sections και φόρμα sections

Πατώντας στην φόρμα με τα sections θα μας εμφανιστούν όλα τα διαθέσιμα sections της βάσης μας. Εικόνα 3-11




**Εικόνα 3-11**

Τα section αποθηκεύονται στον πίνακα jos_sections της βάσης. Πίνακας 3-12

.

## jos_sections

**Πίνακας 3-12**

### Fields

| Field | Type | Collation | Null | Key | Default | Extra | Privileges |
|---|---|---|---|---|---|---|---|
| Id | int(11) | NULL | | PRI | (NULL) | auto_increment | select,insert,update,references |
| Title | text | utf8_general_ci | | | | | select,insert,update,references |
| Name | text | utf8_general_ci | | | | | select,insert,update,references |
| Image | text | utf8_general_ci | | | | | select,insert,update,references |
| Scope | varchar(50) | utf8_general_ci | | MUL | | | select,insert,update,references |
| Image_position | varchar(90) | utf8_general_ci | | | | | select,insert,update,references |
| description | text | utf8_general_ci | | | | | select,insert,update,references |
| published | tinyint(1) | NULL | | | 0 | | select,insert,update,references |
| checked_out | int(11) unsigned | NULL | | | 0 | | select,insert,update,references |
| checked_out_time | datetime | NULL | | | 0000-00-00 00:00:00 | | select,insert,update,references |
| ordering | int(11) | NULL | | | 0 | | select,insert,update,references |
| access | tinyint(3) unsigned | NULL | | | 0 | | select,insert,update,references |
| Count | int(11) | NULL | | | 0 | | select,insert,update,references |
| params | text | utf8_general_ci | | | | | select,insert,update,references |

Μας ενδιαφέρουν περισσότερο τα πεδία title, alias, description, published, access

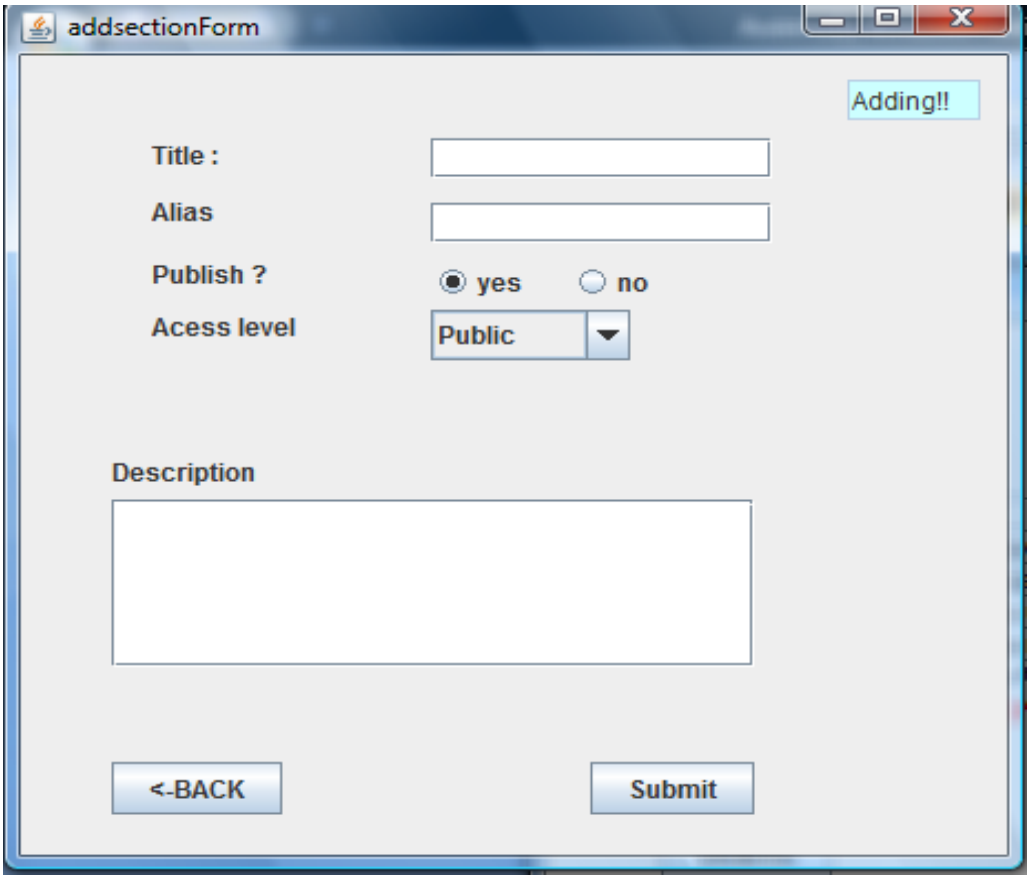
**Title**: Περιέχει τον τίτλο του section.

**Alias**: Περιέχει το alias του section.

**Description**: Περιέχει την περιγραφή του section.

**Published**: Δείχνει εάν έχει γίνει published .

Μπορούμε να κάνουμε επιλογή κάποιου section για να το κάνουμε **Delete** η **Edit** . Εάν ο χρήστης πατήσει το κουμπί **Add** θα μεταφερθεί στην φόρμα που προσθέτουμε sections.Εικόνα 3-13.




**Εικόνα 3-13**

Πάνω δεξιά βλέπουμε την κατάσταση στην οποία βρισκόμαστε (τώρα λέει adding, στην φόρμα του edit θα αλλάξει ). Μπορούμε να βάλουμε title , alias, publish state, Access level και description για το καινούργιο section που πρόκειται να δημιουργήσουμε . Πατώντας το κουμπί submit εκτελούνται οι

μέθοδοι checkRequired() και checkEdit() . Η μέθοδος checkRequired() ελέγχει αν είναι συμπληρωμένα τα απαραίτητα πεδία για να προστεθεί ένα καινούργιο section.

```
//Elenxei an einai simpliromena ta aparaitita paidia gia na kano add ena neo section

    public void  checkRequired()

    {

        String t=jTextFieldTitle.getText();

        String t1=jTextFieldAlias.getText();


        if(t.isEmpty())

            JOptionPane.showMessageDialog(this,"Your section must have a
                                          title!!");

        else if(t1.isEmpty())

            JOptionPane.showMessageDialog(this,"Your section must have an
                                          alias!!");


        if((!t.isEmpty()) && !t1.isEmpty() )

           canSubmit=true;

        else

           canSubmit=false;

    }
```

Εάν όλα είναι εντάξει τότε η μεταβλητή canSubmit γίνετε true και μπορεί να κάνω submit το query .

Έτσι πατώντας το κουμπί **Submit** εκτελείται ο παρακάτω κώδικας

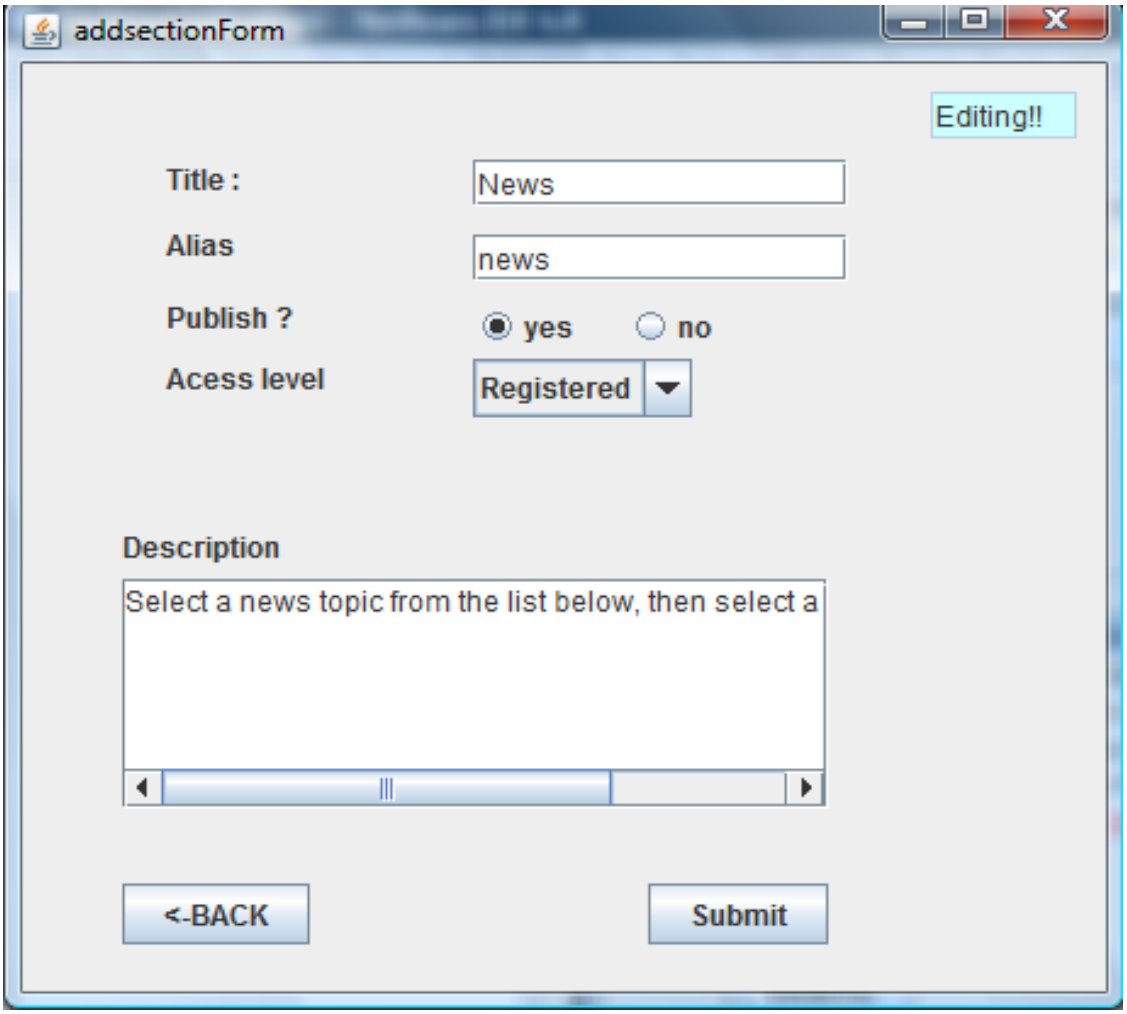
```
if(canSubmit==true)

{

   try

   {
```

```
        String S1="INSERT INTO  `books`.`jos_sections` (`id`,`title`,`name`
         ,`alias`,`image`,`scope`,`image_position`,`description`
         ,`published`,`checked_out`,`checked_out_time`,`ordering`,`access`
         ,`count`,`params`) VALUES (NULL,  '"+jTextFieldTitle.getText()+"',
         '',  '"+jTextFieldAlias.getText()+"',  '',  'content',  '',
         '"+jTextArea1.getText()+"',  '"+published_state+"',  '0',  '2010-05-
         14 00:00:00',  '0',  '"+accesslevel+"',  '0',  '')";

        mtb.sqlgroup.getStatement().executeUpdate(S1);

        mtb.Executefromsql("SELECT id, title, alias, description, published,
                            access FROM jos_sections");

        viewController.selectViews("sectionForm");

    }

    catch(SQLException sqle)

    {

        sqle.printStackTrace();

    }

}

else

    System.out.println("Sorry you have to complete these blanks");
```

Εκτελείτε το query και προστίθεται στην βάση το καινούργιο section, μετά εκτελούμε ένα ακόμα query για να γυρίσουμε τον πίνακα στην κατάσταση που ήταν.

```
INSERT INTO `books`.`jos_sections` (`id`,`title`,`name`,`alias`,`image`,`scope`,`image_position`,`description`,`published`,`checked_out`,`checked_out_time`,`ordering`,`access`,`count`,`params`) VALUES (NULL, '"+jTextFieldTitle.getText()+"', '', '"+jTextFieldAlias.getText()+"', '', 'content', '', '"+jTextArea1.getText()+"', '"+published_state+"', '0', '2010-05-14 00:00:00', '0', '"+accesslevel+"', '0', '')
```

Εάν στην αρχική φόρμα ο χρήστης πατήσει το κουμπί edit  θα μεταφερθούμε στην φόρμα Edit, εικόνα 3-14,και θα εκτελεστεί ο παρακάτω κώδικας




**Εικόνα 3-14**

```
//EDIT button
 private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
  if(canEdit==true)
```

```
  {
      NewJFrame9 njf9 = new NewJFrame9(viewController, NewJFrame.mtb,ob,1);
     //setting values for the EDIT form
njf9.setTextField2((String)jTable1.getValueAt(jTable1.getSelectedRow(),1));
njf9.setTextField((String)jTable1.getValueAt(jTable1.getSelectedRow(),2));
njf9.setTextArea((String)jTable1.getValueAt(jTable1.getSelectedRow(),3));
njf9.setPublish((jTable1.getValueAt(jTable1.getSelectedRow(),4)).toString())
;
njf9.setAcessLevel(jTable1.getValueAt(jTable1.getSelectedRow(),5));
}
else
 JOptionPane.showMessageDialog(this,"Please select an entry to View/Edit!");
 }
```

Όπως βλέπουμε καλείται ένας δομητής με παραμέτρους, μια εξ αυτών και το “1” το οποίο δηλώνει ότι βρισκόμαστε πλέον σε edit State , μετά καλούνται οι διάφορες μέθοδοι για να αρχικοποιηθούν τα πεδία της φόρμας που θα εμφανιστεί μπροστά μας . Οι μέθοδοι **setTextField2()**, **setTextField(), setTextArea()** βάζουν τιμές στα title, alias και Description αντίστοιχα . Μετά καλείτε η μέθοδος **setPublish()** η οποία αλλάζει το toggle button .

```
//change the radio button status in the edit form
    public void setPublish(String x)
    {
        if(x == "true")
        {
            yesradio.setSelected(true);
            noradio.setSelected(false);
```

```
            published_state=1;
            oldpublishstate=1;
        }
        else
        {
            yesradio.setSelected(false);
            noradio.setSelected(true);
            published_state=0;
            oldpublishstate=0;
        }
    }
```

Και θέτει ταυτόχρονα τιμές στα published_state και στα oldpublished για τον έλεγχο που θα γίνει αργότερα. Συνήθως σε όλες τις μεθόδους που καλούμε από την αρχική φόρμα κάνουμε και την αντιστοίχηση των παλιών τιμών με των νέων.

Μετά καλούμε την μέθοδο **setAccessLevel()** για να αλλάξουμε την τιμή του ComboBox.

```
 //Bazei to access level kai epilegei mia timi apo to sigkekrimeno combobox
analoga tin timi pou tou dinoume
//arxikopei kai to oldaccesslevel gia ton metepeita elenxo
    public void setAcessLevel(Object c)
    {
         eventfirebysoftware=true;

        if(c.equals(0))
        {
            accesslevel=0;
            oldaccesslevel=0;
            jComboBox1.setSelectedIndex(0);
```

```
        }
        else if(c.equals(1))
        {
            accesslevel=1;
            oldaccesslevel=1;
            jComboBox1.setSelectedIndex(1);
        }
        else
        {
            accesslevel=2;
            oldaccesslevel=2;
            jComboBox1.setSelectedIndex(2);
        }
    }
```

Η οποία με την σειρά της δίνει και αυτή τιμές στα oldaccess και access level

Εάν πατήσει ο χρήστης το κουμπί **Submit** για να αλλάξει το section στην βάση, καλείται η μέθοδος **checkEdit()** .

Η **checkEdit()** ελέγχει ένα έχουν τροποποιηθεί τα πεδία και εκτελεί το query μόνο για τα πεδία που άλλαξαν.

```
//Elenxei ean mporo na kano edit, i den exo allaksei tpt kai den tha kanei
edit
    public void checkEdit()
    {

        String t=jTextFieldTitle.getText();
        String t1=jTextFieldAlias.getText();
        String t2= jTextArea1.getText();
        String first="";
        editquerry="UPDATE jos_sections SET ";
```

```
if(t.equals(oldtitle))
    System.out.println("To title einai idio");
else
{
    if (first.isEmpty())
    {
        first="something"; //etsi gia na gemisei mono
        editquerry+="title='"+t+"'";
    }
    else
        editquerry+=", title='"+t+"'";
}
```

Χρησιμοποιεί το τέχνασμα που αναφέρθηκε και προηγουμένως με το άδειο String .Η μέθοδος αναφέρετε τμηματικά.

```
editquerry+=" WHERE id='"+ob+"'";
```

```
if(first.isEmpty())
     editquerry="UPDATE jos_sections SET modified=NOW() WHERE
               id='"+ob+"'";
else
   first="something";
```

```
if (t.equals(oldtitle)==false || t1.equals(oldalias)==false ||
published_state!=oldpublishstate || oldaccesslevel!=accesslevel ||
oldtext.equals(t2)==false)
{
    canEdit=true;
```

Η μέθοδος αυτή τροποποιεί το editquery . Εάν το canEdit γίνει true τότε, όταν ο χρήστης πατήσει το κουμπί Submit θα εκτελεστεί το παρακάτω κομμάτι κώδικα. Η μεταβλητή stateEdit είναι ίση με 1 εξ αρχής, καθώς έτσι δηλώθηκε από τον δομητή, πράγμα που σημαίνει ότι είμαστε σε κατάσταση Edit . Μέτα θα εκτελεστεί και το query για να επιστρέψει ο πίνακας στην αρχική του κατάσταση.

```
if(stateEdit==1)
{
  if(canEdit==true)
 {
    try
    {
        mtb.sqlgroup.getStatement().executeUpdate(editquerry);
        mtb.Executefromsql("SELECT id, title, alias, description, published,
                           access FROM jos_sections");
        viewController.selectViews("sectionForm");
        stateEdit=0;
    }
    catch(SQLException exc)
    {
        exc.printStackTrace();
    }
}
else
    System.out.println("Sorry you cannot edit");
```

Ο χρήστης έχει την επιλογή να κάνει **DELETE** ένα section αν στην αρχική φόρμα πατήσει το **delete button.**

```
//DELETE BUTTON OK!
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        if(canEdit==true)
            checkDelete(ob);
        else
            JOptionPane.showMessageDialog(this,"Please select an entry to
                                                Delete!!");
    }
```

Καλείται η μέθοδος **checkDelete()**, με την μέθοδο αυτή εκτελείτε ένα query

```
public void checkDelete(Object ob)
{
  try
  {
       String s="SELECT title FROM jos_categories WHERE section='"+ob+"'";
       mtb.sqlgroup.stmtExec(s);
       ResultSet rstemp=mtb.sqlgroup.getResultSet();
       rstemp.first();
       if(rstemp.first())
            System.out.println("den mporo na ta sbiso upaxei egrafi");
       else
      {
           String S2="DELETE FROM jos_sections WHERE id="+ob;
           mtb.sqlgroup.getStatement().executeUpdate(S2);
      }
         mtb.Executefromsql("select id, title, alias, description,
                          published, access FROM jos_sections");
   }
```

Για να βρεθεί εάν υπάρχει κάποιο category για αυτό το section. Αν ναι ενημερώνει το χρήστη με μήνυμα και δεν το σβήνει .αλλιώς το σβήνει.

### 3. 5. Categories και φόρμα categories

Για να προσθέσουμε ένα καινούργιο category θα ασχοληθούμε με τον πίνακα jos_categories . Πίνακας 3-15.

**Jos_categories**

**Πίνακας 3-15**

**Fields**

| Field | Type | Collation | Null | Key | Default | Extra | Privileges |
|---|---|---|---|---|---|---|---|
| Id | int(11) | NULL | | PRI | (NULL) | auto_increment | select,insert,update,references |
| Parent_id | int(11) | NULL | | | 0 | | select,insert,update,references |
| Title | text | utf8_general_ci | | | | | select,insert,update,references |
| Name | text | utf8_general_ci | | | | | select,insert,update,references |
| Image | varchar(255) | utf8_general_ci | | | | | select,insert,update,references |
| section | varchar(150) | utf8_general_ci | | MUL | | | select,insert,update,references |
| Image_position | varchar(90) | utf8_general_ci | | | | | select,insert,update,references |
| description | text | utf8_general_ci | | | | | select,insert,update,references |
| published | tinyint(1) | NULL | | | 0 | | select,insert,update,references |
| checked_out | int(11) unsigned | NULL | | MUL | 0 | | select,insert,update,references |
| checked_out_time | datetime | NULL | | | 0000-00-00 00:00:00 | | select,insert,update,references |
| Editor | varchar(150) | utf8_general_ci | YES | | (NULL) | | select,insert,update,references |
| ordering | int(11) | NULL | | | 0 | | select,insert,update,references |
| access | tinyint(3) unsigned | NULL | | MUL | 0 | | select,insert,update,references |
| Count | int(11) | NULL | | | 0 | | select,insert,update,references |
| params | text | utf8_general_ci | | | | | select,insert,update,references |

**Indexes**

Μας ενδιαφέρουν περισσότερα τα πεδία title, alias, section, description, published, access.

**Title:** Περιέχει το όνομα του category.

**Alias**: Περιέχει το alias του category.

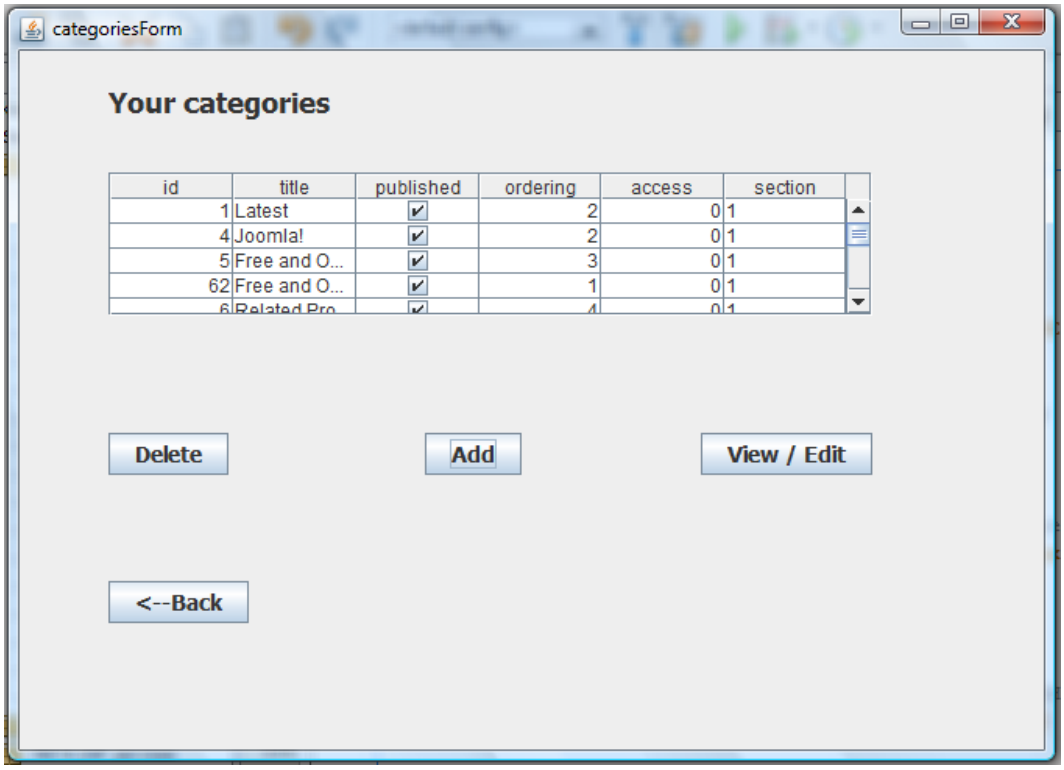
**Section**: Δείχνει σε πιο section ανήκει το συγκεκριμένο category.

**Description**: Περιέχει την περιγραφή του category.

**Published**: Μας δείχνει έαν έχει γίνει published το συγκεκριμένο category.

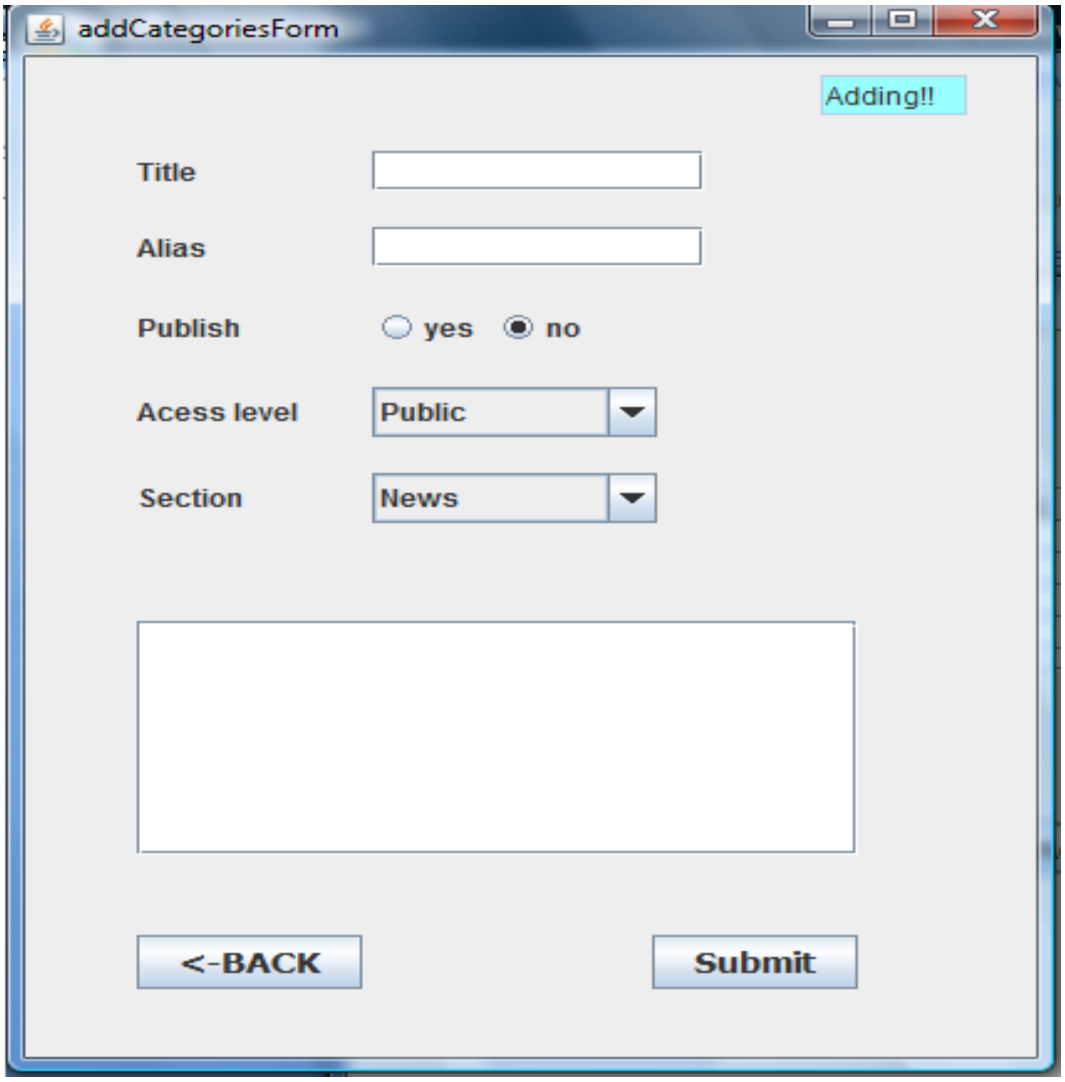
**Access**: Μας δείχνει το επίπεδο πρόσβασης του category.

Πατώντας στο κουμπί για να πάμε πάνω στα categories θα εμφανιστεί αυτή η φόρμα, Περιέχει όλα τα category της βάσης καθώς και κάποιες πληροφορίες για αυτά. Παρακάτω βλέπουμε τρία κουμπιά, **Delete**, **Add**, **View / Edit** . Εικόνα 3-16.




**Εικόνα 3-16**

Πατώντας στο κουμπί Add θα μεταφερθούμε στην φόρμα που προσθέτονται categories, σε αυτήν την φόρμα βλέπουμε τα πεδία τα οποία πρέπει να συμπληρώσουμε για να προσθέσουμε επιτυχώς ένα category. Βλέπουμε ότι υπάρχουν πεδία όπως **title**, **alias publish**, **Access level**, **section**, και το **description** .Βρισκόμαστε σε κατάσταση Add η οποία εμφανίζεται πάνω δεξιά στην φόρμα μας. Εικόνα 3-17.




**Εικόνα 3-17**

Στο Section combobox υπάρχουν όλα τα διαθέσιμα section της βάσης μας, αυτό το combobox γεμίζει δυναμικά, με τους τίτλους των section, καλώντας την μέθοδο **dynamicSections()** κατά την αρχικοποίηση της φόρμας.

```
//emfanizei sto combobox ola ta diathesima sections
    public void dynamicSections(){
        try{


                String ids="SELECT title FROM  `jos_sections`";
                mtb.sqlgroup.getStatement().executeQuery(ids);
                ResultSet rstemp1=
                     mtb.sqlgroup.getStatement().executeQuery(ids);


                while (rstemp1.next())
                {
                  jComboBoxSection.addItem( rstemp1.getString(1));


                }
                mtb.Executefromsql("SELECT id, title, published, ordering,
                            access, section FROM jos_categories");
       }
       catch(SQLException exc)
       {
          exc.printStackTrace();
       }
}
```

Η μέθοδος αυτή εκτελεί ένα **query** το οποίο επιστρέφει όλους τους τίτλους από τον πίνακα jos_sections και τους βάζει στο combobox .

Πατώντας το κουμπί **Submit** τρέχουν οι μέθοδοι:

```
checkEdit();
convertTitletoid(sectiontitle);
checkRequired();
```

Εμείς θα ασχοληθούμε με την **checkRequired()** προς το παρόν. Η μέθοδος αυτή ελέγχει εάν είναι συμπληρωμένα τα απαραίτητα πεδία της φόρμας για να εκτελεστεί το **query**.

```
public void  checkRequired()
{
    String t=jTextTitle.getText();
    String t1=jTextAlias.getText();
    if(t.isEmpty())
          JOptionPane.showMessageDialog(this,"Your category must have a
                                  title!!");
    else if(t1.isEmpty())
            JOptionPane.showMessageDialog(this,"Your category must have an
                    alias!!");
    if((!t.isEmpty()) && !t1.isEmpty() )
           canSubmit=true;
   else
          canSubmit=false;
}
```

Ένα όχι εμφανίζει σχετικό μήνυμα αλλιώς κάνει την μεταβλητή canSubmit true που σημαίνει ότι η εγγραφή είναι έτοιμη προς εκτέλεση . Το **query** εκτελείται και προστίθεται στην βάση.

```
if(canSubmit==true)
{
   try
   {
        String S1="INSERT INTO   `books`.`jos_categories` (`id`,`parent_id`
       ,`title`,`name`,`alias`,`image`,`section`,`image_position`
```

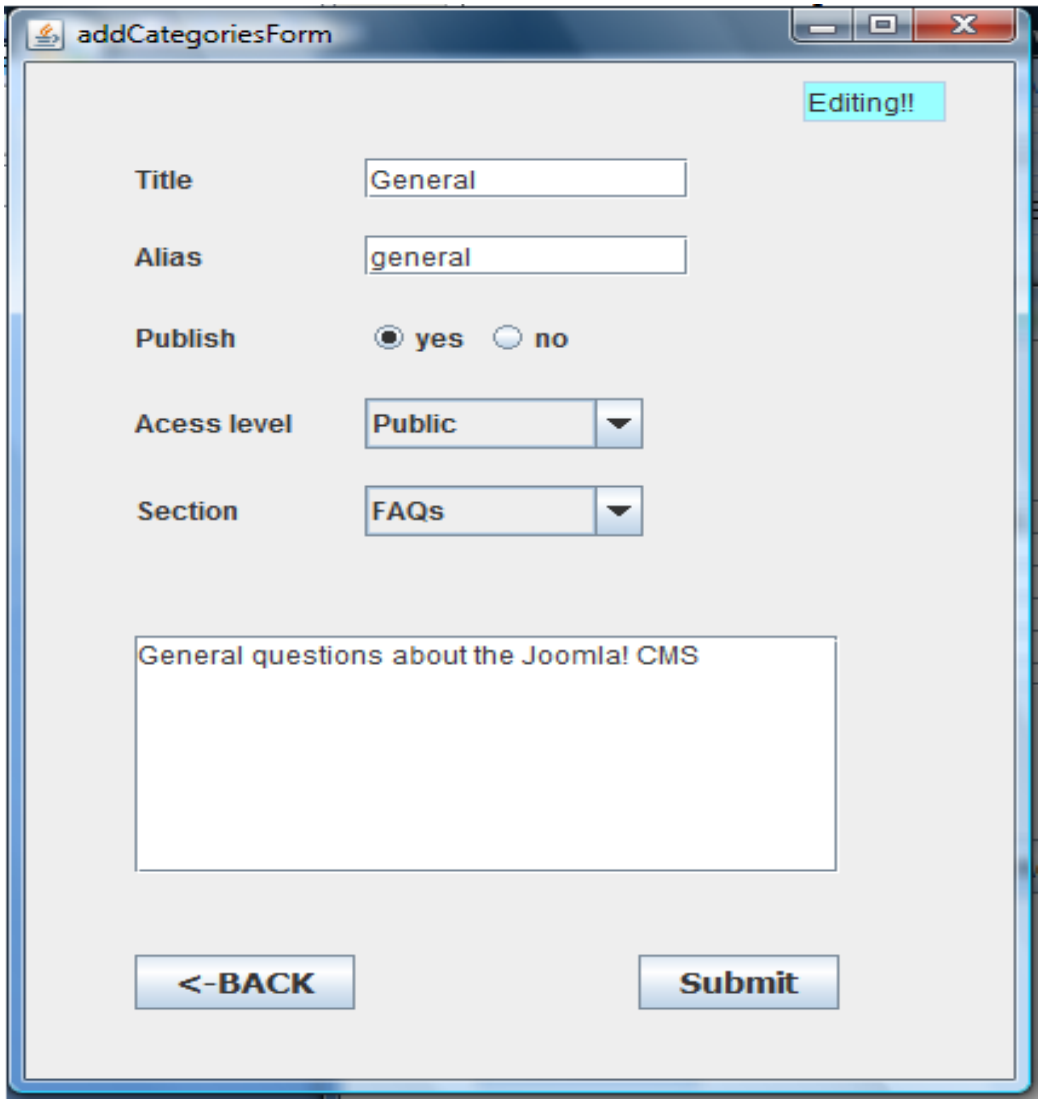
```
        ,`description`,`published`,`checked_out`,`checked_out_time`
        ,`editor`,`ordering`,`access`,`count`,`params`)VALUES (NULL,
'0',  '"+jTextTitle.getText()+"',  '',  '"+jTextAlias.getText()+"',
'',  '"+sectionlevel+"',  'left',  '"+jTextArea1.getText()+"',  '"+
published_state+"',  '0',  '0000-00-00 00:00:00',  'NULL',  '1',
'"+accesslevel+"',  '0',  '')";

         mtb.sqlgroup.getStatement().executeUpdate(S1);

         mtb.Executefromsql("SELECT id, title, published, ordering, access,
             section FROM jos_categories");

    }

    catch(SQLException sqle)

  {

         sqle.printStackTrace();

  }

         viewController.selectViews("categoriesForm");

}

else

        System.out.println("Sorry i cannot submit query");
```

Από την αρχική φόρμα αν πατηθεί το κουμπί Edit θα εκτελεστεί ο παρακάτω κώδικας και θα εμφανιστεί η φόρμα.

```
if(canEdit==true)

{

      NewJFrame11 njf11 =new NewJFrame11(viewController,NewJFrame.mtb,ob,1);

      njf11.setTextField2(selection[0]);//ttle textfield ok!

      njf11.setPublish( selection[1]);// publish toggle button ok!

      njf11.setAcessLevel(temp); //access combobox OK!

      njf11.findAlias(ob);

      njf11.setJcomboSection(temp2);

      try

      {

          String s="SELECT description FROM jos_categories WHERE
                          id='"+ob+"'";

          mtb.sqlgroup.stmtExec(s);

          ResultSet rstemp=mtb.sqlgroup.getResultSet();
```

```
        if(rstemp.first())

             njf11.setTextArea(rstemp.getString(1));


        mtb.Executefromsql("SELECT id, title, published, ordering, access,
           section FROM jos_categories");


    }
```

Βλέπουμε ότι επιλέγεται δομητής με κατάσταση 1 για να πάμε σε Edit mode, καλούνται οι μέθοδοι που αρχικοποιούν τα πεδία της φόρμας, **setTextField2()**, **setPublish()**, **setAccessLevel()**, **findAlias()**, **setJcomboSection()** και εκτελείται ένα **query** για να βάλουμε το description στο textarea . Εικόνα 3-18 .




**Εικόνα 3-18**

Κάνοντας οποιεσδήποτε αλλαγές θέλει ο χρήστης, πατώντας το Submit Button καλείται η μέθοδος checkEdit η οποία κάνει μια σύγκριση ανάμεσα στις παλιές τιμές και στις καινούργιες, και αυτές που άλλαξαν τις προσθέτει στο query για να εκτελεστούν .

```
public void checkEdit()
{
```

```
String t=jTextTitle.getText();
String t1=jTextAlias.getText();
String t2=jTextArea1.getText();
String first="";

editquerry="UPDATE jos_categories SET ";

if(t.equals(oldtitle))
    System.out.println("idios titlos");
else
{
    if (first.isEmpty())
    {
        first="something"; //etsi gia na gemisei mono
        editquerry+="title='"+t+"'";
    }
    else
        editquerry+=", title='"+t+"'";
}
```

Τέλος επιστρέφει το query που περιέχει μόνο τις αλλαγμένες τιμές, η τιμή **canEdit** γίνετε true .Η μέθοδος είναι αρκετά μεγάλη και παρουσιάζετε τμηματικά.

```
if(oldtext.equals(t2))
        System.out.println("Exoune to idio keimeno");
else
{
```

```
             if (first.isEmpty())
            {
                first="something"; //etsi gia na gemisei mono

                editquerry+="description='"+t2+"'";
            }
            else
                editquerry+=", description='"+t2+"'";

        }

editquerry+=" WHERE id='"+ob+"'";

        if (t.equals(oldtitle)==false || t1.equals(oldalias)==false ||
accesslevel!=old_access || old_publishedstate!=published_state ||
oldSectiontitle.equals(sectiontitle)==false || oldtext.equals(t2)==false)
            canEdit=true;
        else
            canEdit=false;
```

Μέσα από τον κώδικα του **Submit Button** εκτελείται το κατάλληλο query .

```
if(stateEdit==1)
{
   if(canEdit==true)
   {
     try
     {
           System.out.println(editquerry);
           mtb.sqlgroup.getStatement().executeUpdate( editquerry);
```

```
            mtb.Executefromsql("SELECT id, title, published, ordering,
                    access, section FROM jos_categories");

            viewController.selectViews("sectionForm");

            stateEdit=0;

            viewController.selectViews("categoriesForm");

    }

    catch(SQLException exc)

    {

         exc.printStackTrace();

    }

}

else

       System.out.println("Sorry you cannot edit");
```

Πατώντας στην αρχική φόρμα το κουμπί **delete** καλείται η μέθοδος **checkDelete()**

```
//DELETE button

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

    if(canEdit==true)

        checkDelete(ob);

    else

        JOptionPane.showMessageDialog(this,"Please select an entry to
Delete!!");

    }
```

Η μέθοδος **checkDelete()** ελέγχει εάν υπάρχει άρθρο που ανήκει στο συγκεκριμένο category , εάν ναι, εμφανίζει μήνυμα και δεν το σβήνει, αλλιώς αν το category είναι άδειο το σβήνει .

```
public void checkDelete(Object ob)
{
   try
   {
        String s="SELECT title FROM jos_content WHERE catid='"+ob+"'";
        mtb.sqlgroup.stmtExec(s);

        ResultSet rstemp2=mtb.sqlgroup.getResultSet();
        rstemp2.first();
        if(rstemp2.first())
        {
            System.out.println("den mporo na ta sbiso upaxei egrafi");
        }
        else
        {
            String S1="DELETE FROM jos_categories WHERE id=" +ob;
            mtb.sqlgroup.getStatement().executeUpdate(S1);
        }

     mtb.Executefromsql("SELECT id, title, published, ordering,
           access, section FROM jos_categories");
```

**4. Βιβλιογραφία**

www.Joomla.org

**http://docs.Joomla.org/**

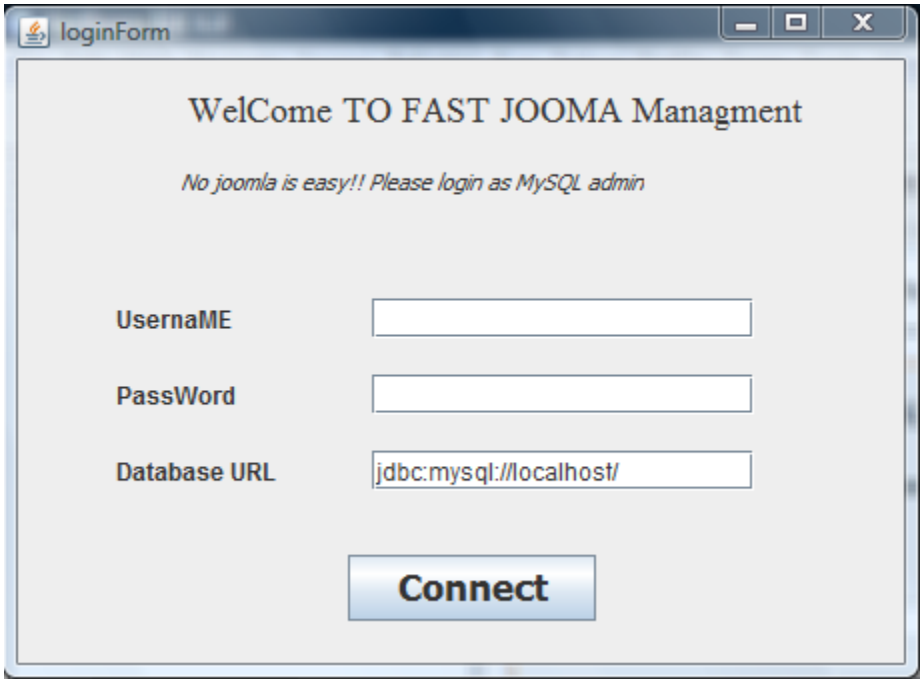
## 4. Παραρτήματα

Οι πίνακες με τα πεδία του Joomla βρίσκονται στο

**http://dev.Joomla.org/downloads/Joomla15_DB-Schema.htm**

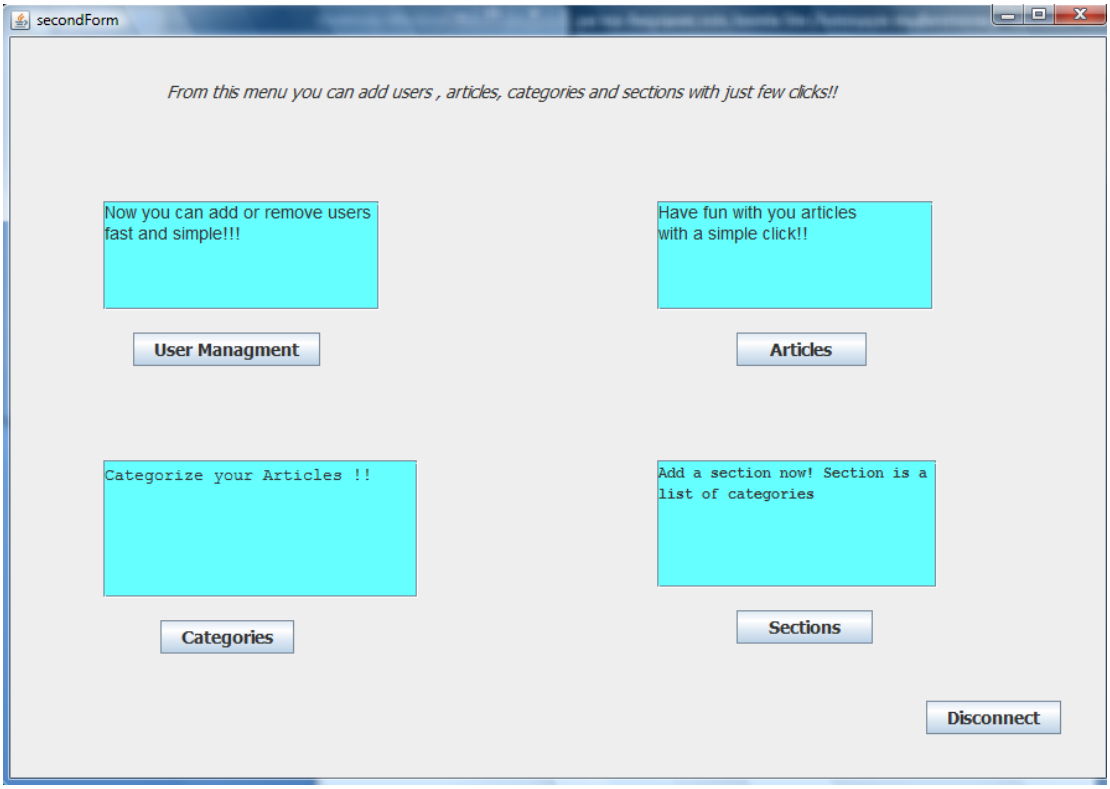
## 6. Οδηγός Χρήσης Λογισμικού

Εφόσον ο χρήστης θα καταφέρει να συνδεθεί επιτυχώς από την φόρμα Login. Εικόνα U1.




**U 1**

Θα μεταφερθεί στην οθόνη διαχείρισης, μέσω αυτής θα μπορέσει να εκτελέσει κάποιες απλές λειτουργίες όσο αναφορά τα άρθρα, τους χρήστες τα categories και τα sections. Εικόνα U2.




**U 2**

Από εδώ και πέρα επιλέγοντας το κατάλληλο κουμπί μεταφέρεται και στην αντίστοιχη επιλογή.