ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Το πλαίσιο ανάπτυξης εφαρμογών διαδικτύου ***Ruby on Rails***

Του φοιτητή

Κυριάκου Διαμαντή

Αρ. Μητρώου: 04/2698

Επιβλέπων καθηγητής

Ηλίας Νίτσος

Θεσσαλονίκη 2011

ΠΡΟΛΟΓΟΣ

Στόχος της πτυχιακής είναι η μελέτη του πλαισίου ανάπτυξης διαδικτυακών εφαρμογών Ruby on Rails, η ανάλυση των βασικών του χαρακτηριστικών, μια σύντομη παρουσίαση της Ruby (της γλώσσας προγραμματισμού που χρησιμοποιεί το Ruby on Rails). Επιχειρείται επίσης σύγκριση του Ruby on Rails με άλλα πλαίσια ανάπτυξης διαδικτυακών εφαρμογών. Επίσης στόχο αποτελεί η εγκατάσταση του απαιτούμενου λογισμικού για την ανάπτυξη εφαρμογών με το συγκεκριμένο πλαίσιο ανάπτυξης και η ανάπτυξη ενδεικτικής εφαρμογής για την διαχείριση ηλεκτρονικού βιβλιοπωλείου.

Η παρούσα πτυχιακή εργασία εκπονήθηκε στα πλαίσια του Τμήματος Πληροφορικής, της σχολής Τεχνολογικών Εφαρμογών του Αλεξάνδρειου Τεχνολογικού Ιδρύματος Θεσσαλονίκης.

ΠΕΡΙΛΗΨΗ

Η δημιουργία διαδικτυακών εφαρμογών έχει σκοπό να κάνει ευκολότερη και πιο διαδραστική την περιήγηση των ανθρώπων στον παγκόσμιο ιστό. Η ανάπτυξη πλαισίων ανάπτυξης (framework) στηριγμένα σε μία συγκεκριμένη γλώσσα προγραμματισμού έχει σκοπό να κάνει ευκολότερη και πιο αποδοτική την ζωή των προγραμματιστών. Συνοψίζοντας τις παραπάνω σκέψεις μπορούμε να επισημάνουμε ότι το πλαίσιο ανάπτυξης Ruby on Rails δημιουργήθηκε για να βοηθήσει τους προγραμματιστές διαδικτυακών εφαρμογών να γίνουν πιο παραγωγικοί, πιο αποδοτικοί και πολύ πιο γρήγοροι κατά την ανάπτυξη διαδικτυακών εφαρμογών.

ABSTRACT

The development of web applications aims to a more usable and interactive experience in the World Wide Web navigation. The development of web application frameworks based on a particular programming language aims to make the programmers' life easier and more productive. Summarizing these thoughts we can conclude that the Ruby on Rails framework was made to help the web programmers be more productive and faster during the development of a web application.

ΠΕΡΙΕΧΟΜΕΝΑ

Ευρετήριο σχημάτων

Ευρετήριο πινάκων

Ευρετήριο εικόνων

Συντομογραφίες

API: Application Programming Interface

CGI: Common Gateway Interface

COC : Convention Over Configuration

CRUD : Create Read Update Delete

CSS: Cascading Style Sheets

DDL: Data Definition Language

DRY : Don’t Repeat Yourself

DSL : Domain Specific Language

ERb : Embedded Ruby

FCGI: Fast Common Gateway Interface

Gem : Προγράμματα – Βιβλιοθήκες Ruby

GUI: Graphical User Interface

HTML: HyperText Markup Language

HTTP: HyperText Transfer Protocol

MVC : Model View Controller

ORM : Object Relational Mapping

RDBMS: Relational DataBase Management System

REST : Representational State Transer

RHTML: Ruby HyperText Markup Language

RJS: Ruby JavaScript

SQL: Structured Query Language

URI : Uniform Resource Identifier

URL : Uniform Resource Locator

WWW: World Wide Web

XML: eXtensible Markup Language

ΕΙΣΑΓΩΓΗ

Στην παρούσα πτυχιακή εργασία πραγματοποιείται μια προσπάθεια μελέτης του πλαισίου ανάπτυξης διαδικτυακών εφαρμογών Ruby on Rails που είναι υλοποιημένο στην γλώσσα προγραμματισμού Ruby, για την οποία πραγματοποιείται μια σύντομη εισαγωγή στα ενδότερα της. Πραγματοποιείται επίσης μια εκτενής αναφορά στα κυριότερα συστατικά αυτού του πλαισίου, καθώς και αναλύονται τα πλεονεκτήματα και τα μειονεκτήματα του σε σχέση με υπάρχοντα ευρέως χρησιμοποιούμενα πλαίσια ανάπτυξης. Επίσης παρουσιάζεται το λογισμικό που είναι απαραίτητο για την λειτουργία μιας τέτοιας εφαρμογής. Και τέλος παρουσιάζεται η εφαρμογή που αναπτύχθηκε στα πλαίσια της πτυχιακής με την χρήση του συγκεκριμένου πλαισίου ανάπτυξης.

Στο 1^{ο} κεφάλαιο πραγματοποιείται μια εισαγωγή στην γλώσσα προγραμματισμού Ruby. Η Ruby είναι η γλώσσα πάνω στην οποία αναπτύχτηκε το πλαίσιο Ruby on Rails και ένας από τους λόγους της δημοτικότητας του.

Στο 2^{ο} κεφάλαιο πραγματοποιείται μια εκτενής αναφορά στα σημαντικότερα συστατικά του πλαισίου ανάπτυξης Ruby on Rails. Αναλύονται οι αρχές σχεδίασης και τα πρότυπα που ακολουθούνται κατά την ανάπτυξη διαδικτυακών εφαρμογών και εμβαθύνουμε στον τρόπο λειτουργίας και στην αρχιτεκτονική του πλαισίου.

Στο 3^{ο} κεφάλαιο επιχειρείται μια σύγκρισή του πλαισίου Ruby on Rails με άλλα ευρέως χρησιμοποιούμενα πλαίσια ανάπτυξης βασισμένα σε γνωστές γλώσσες προγραμματισμού. Εξάγονται χρήσιμα συμπεράσματα σχετικά με τα πλεονεκτήματα και τα μειονεκτήματα του πλαισίου.

Στο 4^{ο} κεφάλαιο παρουσιάζονται οι ενέργειες και το λογισμικό που είναι απαραίτητο για να υλοποιηθεί μια εφαρμογή Ruby on Rails. Επίσης πραγματοποιείται παρουσίαση της εγκατάστασης των απαραίτητων προγραμμάτων, όπως διακομιστής http, για την λειτουργία μιας εφαρμογής Ruby on Rails.

Στο 5^{ο} κεφάλαιο παρουσιάζεται η εφαρμογή που αναπτύχθηκε στα πλαίσια αυτής της πτυχιακής εργασίας. Πρόκειται για ένα ηλεκτρονικό κατάστημα βιβλιοπωλείου που αναπτύχθηκε εξ ολοκλήρου με την χρήση του πλαισίου Ruby on Rails.

1 Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ RUBY

Σε αυτό το κεφάλαιο πραγματοποιούμε μια εισαγωγή στην γλώσσα προγραμματισμού Ruby, αναφέροντας κάποια ιστορικά στοιχεία και τον σκοπό δημιουργίας της. Αναφέρουμε συνοπτικά το συντακτικό της γλώσσας καθώς και τις δομές που είναι απαραίτητες σε μία γλώσσα προγραμματισμού, όπως δομές ελέγχου και επανάληψης. Επίσης θα ανακαλύψουμε γιατί η Ruby είναι μια πλήρως αντικειμενοστραφής γλώσσα προγραμματισμού.

1.1 Βασικά στοιχεία & Ιστορική αναδρομή

1.1.1 Τι είναι η Ruby

Η Ruby είναι μια αντικειμενοστραφής διερμηνευόμενη γλώσσα προγραμματισμού που μπορεί να χρησιμοποιηθεί για ανάπτυξη διαφόρων τύπων εφαρμογών, με ιδιαίτερα χαρακτηριστικά και δυνατότητες. Θεωρείται εύκολη στην εκμάθηση της, είναι ιδιαίτερα εκφραστική και στοχεύει στον δημιουργικό και παραγωγικό προγραμματισμό. Αυτό δεν σημαίνει όμως ότι είναι μια απλή γλώσσα. Είναι μια πλήρης αντικειμενοστραφής γλώσσα και θεωρείται από τις πιο δυνατές δυναμικές metaprogramming γλώσσες προγραμματισμού.

1.1.2 Ιστορία της Ruby

Ο Yukihiro Matsumoto δημιούργησε τη γλώσσα προγραμματισμού Ruby το 1993 (Flanagan et al., 2008) χρησιμοποιώντας στοιχεία από τις αγαπημένες του γλώσσες προγραμματισμού (Smalltalk, Perl, Eiffel, Ada, και Lisp). Η αρχική της ανάπτυξη πραγματοποιήθηκε στα χρόνια που ακολούθησαν μέχρι την 21 Δεκεμβρίου του 1995 που δημοσιεύτηκε η πρώτη έκδοση της γλώσσας (Ruby 0.95) σε εγχώρια newsgroup της Ιαπωνίας. Η έκδοση 1.0 της ruby δημοσιεύτηκε στις 25 Δεκεμβρίου 1996 και από τότε άρχισε η συνεχόμενη ανάπτυξη της. Άργησε να γίνει γνωστή γιατί η τεκμηρίωση (documentation) της μεταφράστηκε στα αγγλικά το 1998 (το πρωτότυπο ήταν στα γιαπωνέζικα). Χρησιμοποιούνταν από μικρή μερίδα προγραμματιστών μέχρι το 2004 που άλλαξαν όλα. Τότε πρωτοεμφανίστηκε το πλαίσιο ανάπτυξης διαδικτυακών εφαρμογών Ruby on Rails, το οποίο απογείωσε την δημοτικότητα της ruby παγκοσμίως. Σήμερα έχουμε φτάσει στην 1.9.2 έκδοση (με την 2.0 έκδοση να είναι στα σκαριά) της γλώσσας, και η γλώσσα έχει αποκτήσει πολλά νέα χαρακτηριστικά και η αξιοπιστία της συνεχώς αυξάνεται.

- *Σπουδαιότερες δημοσιευμένες εκδόσεις* :

Πίνακας 1 "Εκδόσεις της Ruby"

Έκδοση	Ημερομηνία
Ruby 0.95	21 Δεκεμβρίου 1995
Ruby 1.0	25 Δεκεμβρίου 1996
Ruby 1.2.1	11 Ιανουαρίου 1999
Ruby 1.4.0	13 Αυγούστου 1999
Ruby 1.6.0	19 Σεπτεμβρίου 2000
Ruby 1.8.0	4 Αυγούστου 2003
Ruby 1.8.5	25 Αυγούστου 2006
Ruby 1.9.1	30 Ιανουαρίου 2009

1.1.3 Σκοπός δημιουργίας

Παρακάτω βλέπουμε μία δήλωση του Yukihiro Matsumoto.

> *I wanted to minimize my frustration during programming, so I want to minimize my effort in programming. That was my primary goal in designing Ruby. I want to have fun in programming myself.*
>
> —Yukihiro Matsumoto (Matz), δημιουργός της Ruby

Με λίγα λόγια ο Matsumoto διατυπώνει ότι θέλησε να δημιουργήσει μια γλώσσα προγραμματισμού που θα είναι ευχάριστη στην χρήση της και ο προγραμματισμός εφαρμογών της θα απαιτεί λιγότερη προσπάθεια από την πλευρά του προγραμματιστή. Όπως αναφέρθηκε παραπάνω ο Yukihiro Matsumoto χρησιμοποιώντας στοιχεία από τις αγαπημένες του γλώσσες προγραμματισμού θέλησε να δημιουργήσει μια γλώσσα που θα χρησιμοποιούσε τις αρχές και τις μεθοδολογίες τόσο του συναρτησιακού όσο και του προστακτικού προγραμματισμού. Θέλησε να δημιουργήσει μια γλώσσα σεναρίων (scripting language) πιο ισχυρή από την Perl και πιο αντικειμενοστραφή από την Python. Επίσης ήθελε η χρήση της γλώσσας να είναι ευχάριστη στους προγραμματιστές, έτσι ώστε σε μικρότερο χρόνο και με λιγότερο κόπο να αναπτύσσουν εφαρμογές, οι οποίες με την χρήση άλλων γλωσσών θα απαιτούσαν πολλαπλάσια προσπάθεια και χρόνο.

1.1.4 Χαρακτηριστικά της Ruby

Ευκολία

Η Ruby έχει πολύ εύκολο και ‘καθαρό’ συντακτικό που επιτρέπει τους προγραμματιστές να μάθουν την γλώσσα γρήγορα και εύκολα. Ο ίδιος ο δημιουργός της άλλωστε είχε δηλώσει ότι ήθελε να δημιουργήσει μια γλώσσα που θα πολλαπλασιάζει την παραγωγικότητα του προγραμματιστή και παράλληλα θα είναι ευχάριστη στην χρήση της. Το συντακτικό της Ruby είναι παρόμοιο με πολλές γλώσσες προγραμματισμού όπως η C++ και η Perl. Το γεγονός αυτό επιτρέπει τους προγραμματιστές να μάθουν εύκολα την Ruby και να αρχίσουν να παράγουν κώδικα σε σύντομο χρονικό διάστημα. Η δύναμη της Ruby βρίσκεται στην απλότητα και την περιεκτικότητα του κώδικα της.

Φορητότητα

Η Ruby μπορεί να λειτουργήσει σε πολλά λειτουργικά συστήματα όπως τα Microsoft Windows, το linux, το MAC OS X, και γενικά κάθε παράγωγο του UNIX. Κώδικας γραμμένος σε μία πλατφόρμα (π.χ. Windows) μπορεί να εκτελεστεί σε όλες τις πλατφόρμες που μπορεί να εγκατασταθεί η ruby, χωρίς (ιδιαίτερες) αλλαγές, κάτι αντίστοιχο με την λειτουργία της Java και το JVM.

Επεκτασιμότητα

Υπάρχουν πάρα πολλές βιβλιοθήκες λογισμικού (ανεξαρτήτως γλώσσας προγραμματισμού) διαθέσιμες οι οποίες επιτελούν συγκεκριμένες εργασίες. Αντί να υλοποιήσουμε τις ίδιες βιβλιοθήκες σε Ruby, μπορούμε να τις χρησιμοποιήσουμε μέσα από την Ruby! Επιπρόσθετα υπάρχουν περιπτώσεις που χρειαζόμαστε καλύτερες επιδόσεις (ταχύτητα, διαχείριση πόρων) από αυτές μιας διερμηνευόμενης γλώσσας όπως είναι η Ruby. Μπορούμε να γράψουμε κομμάτια του προγράμματος σε μια γρήγορη και αποδοτική γλώσσα (όπως η C), και να τα χρησιμοποιήσουμε μέσω της ruby. Στην ruby θα φαίνονται σαν επεκτάσεις (extensions-modules) του κώδικα, τα οποία όμως θα είναι υλοποιημένα σε άλλη γλώσσα προγραμματισμού.

Κόστος

Η Ruby είναι λογισμικό ανοικτού κώδικα. Μπορείτε να κατεβάσετε τον κώδικα της, να τον τροποποιήσετε, να τον χρησιμοποιήσετε και να τον διανείμετε για οποιονδήποτε λόγο (και για εμπορικούς σκοπούς) χωρίς κανένα κόστος.

1.2 Συντακτικό της Ruby

Η Ruby έχει πολύ ‘καθαρό’ και περιεκτικό συντακτικό, που επιτρέπει την δημιουργία εφαρμογής της οποίας ο κώδικας είναι πιο μικρός σε μέγεθος, πιο εύκολα αναγνώσιμος και πολύ πιο εύκολος στην διατήρηση του, από αυτόν μια εφαρμογής που έχει γραφεί με ‘παραδοσιακές’ γλώσσες προγραμματισμού, όπως η Java, C, C++. Το συντακτικό της μοιάζει με αυτό των Perl και Python. Ο διαχωρισμός των εντολών γίνεται με την αλλαγή γραμμής, και όχι με το ελληνικό ερωτηματικό ‘;’ , το οποίο ωστόσο χρησιμοποιείται όταν δηλώνουμε πολλές εντολές σε μία γραμμή. Οι παρενθέσεις δεν είναι απαραίτητες κατά την κλήση των μεθόδων-συναρτήσεων. Επίσης απουσιάζουν τα γνωστά άγκιστρα (‘{’, ‘}’, χρησιμοποιούνται όταν δηλώνουμε ένα μπλοκ κώδικα σε μία γραμμή), με τις δεσμευμένες λέξεις να καθορίζουν το μπλοκ κώδικα.

1.2.2 Μεταβλητές

Οποιαδήποτε απλή, με πεζούς χαρακτήρες λέξη μπορεί να θεωρηθεί σαν μεταβλητή από την Ruby. Οι μεταβλητές μπορούν να αποτελούνται από γράμματα, αριθμούς και τον χαρακτήρα underscore(_), αλλά θα πρέπει να ξεκινάνε με γράμμα ή την κάτω παύλα(_). Δεν είναι απαραίτητο να δηλώσουμε τον τύπο της μεταβλητής, ο οποίος μπορεί να μεταβάλλεται δυναμικά κατά της εκτέλεση ενός προγράμματος.

```
i = 0
_metavliti = 12.5
_user, book, author_  = 1, 2, 3
0book = 0                                   # Μη έγκυρο όνομα
```

1.2.2 Σταθερές

Μερικές φορές είναι απαραίτητο να δηλώσουμε μεταβλητές οι οποίες έχουν σταθερή τιμή (δεν είναι στην ουσία μεταβλητές!!), όπως για παράδειγμα την σταθερά π. Αυτές τις τιμές τις ονομάζουμε σταθερές. Στην Ruby μπορούμε να τις δηλώσουμε σαν μεταβλητές οι οποίες ξεκινάνε με κεφαλαίο γράμμα

```
Pi = 3.14159
Pi = 100                # warning: already initialized constant Pi
                        # η τιμή του Pi δεν μπορεί να μεταβληθεί
```

1.2.3 Αριθμοί

Ακόμη και οι αριθμοί στην Ruby είναι αντικείμενα μιας κλάσης. Για παράδειγμα ο αριθμός ‘10’ είναι αντικείμενο της κλάσης *Fixnum*. Η Ruby υποστηρίζει όλες τις γνωστές πράξεις μεταξύ των αριθμών και για πιο εξειδικευμένους αριθμητικούς υπολογισμούς περιλαμβάνει την κλάση Math. Μπορούμε να δηλώσουμε δεκαδικούς αριθμούς με την χρήση της τελείας ‘ . ’. Το κόμμα δεν επιτρέπεται στους αριθμούς, αλλά αν θέλουμε να ξεχωρίσουμε τις χιλιάδες σε έναν αριθμό μπορούμε να χρησιμοποιήσουμε την κάτω παύλα ‘ _ ’. Μερικά παραδείγματα παρακάτω:

```
10 + 2.5                                    # => 12.5
10 * 2                                      # => 20
10 / 2                                      # => 5
10 % 3                                      # => 1
5 ** 2                                      # => 25
3.14, -50.25
population = 11000000                       #
population = 11_000_000                     #Το ίδιο με την από πάνω(για λόγους
                                            #αναγνωσιμότητας)
```

1.2.3 Συμβολοσειρές

Μία συμβολοσειρά είναι μια ακολουθία από γράμματα, αριθμούς και άλλους ειδικούς και μη χαρακτήρες. Υπάρχουν διάφοροι τρόποι να δημιουργήσουμε συμβολοσειρές όπως φαίνεται στο παρακάτω παράδειγμα. Στην Ruby μία συμβολοσειρά είναι αντικείμενο της κλάσης String (όπως έχουμε πει, τα πάντα είναι αντικείμενα στην Ruby), δίνοντας μας την δυνατότητα να χρησιμοποιήσουμε τις πολλές ενσωματωμένες μεθόδους της για την διαχείριση της συμβολοσειράς.

```
a_string = “ Μπορούμε να χρησιμοποιησουμε μονά ”
b_string = ‘ ή διπλά εισαγωγικά  ’
b_string.length                             # => 21
b_string.upcase                             # => ‘ Ή ΔΙΠΛΑ ΕΙΣΑΓΩΓΙΚΆ’
b_string[4..12]                             # => ‘διπλά ει’
```

1.2.4 Σύμβολα

Τα σύμβολα αποτελούν μια ειδική περίπτωση της Ruby, και αποτελεί πηγή συγχύσεων στους προγραμματιστές που μαθαίνουν ruby. Τα σύμβολα είναι λέξεις, τα οποία όπως και οι μεταβλητές μπορούν να περιέχουν γράμματα, αριθμούς και την

κάτω παύλα. Μόνο που πρέπει να ξεκινάνε με τον χαρακτήρα ΄:΄ (άνω κάτω τελεία). Όποτε χρησιμοποιούμε μία συγκεκριμένη συμβολοσειρά πολλές φορές, είναι καλύτερο να την δηλώσουμε σαν σύμβολο, για να κάνουμε οικονομία πόρων (μνήμης). Τα σύμβολα αρχικοποιούνται και αποθηκεύονται στην μνήμη μόνο μια φορά κατά την διάρκεια ενός σεναρίου ruby.

```
#Πλεονεκτήματα στην χρήση ευρετηρίων
Book1 = { “title” => “Δίκτυα υπολογιστών”, “quantity” => 2 }
Book2 = { “title” => “Τηλεπικοινωνίες”, “quantity” => 5 }
…
Book100 = { “title” => “Java σε 24 ώρες”, “quantity” => 3 }
```

Κάθε φορά που δημιουργούμε ένα ευρετήριο bookXX, δημιουργούνται και αποθηκεύονται στην μνήμη οι συμβολοσειρές ‘title’ και ‘quantity’. Δηλαδή για το παραπάνω μπλοκ κώδικα έχουν δημιουργηθεί εκατό διαφορετικές μεταβλητές με τιμή ‘title’ και άλλες εκατό με τιμή ‘quantity’. Ενώ αν χρησιμοποιήσουμε σύμβολα :

```
Book1 = { :title => “Δίκτυα υπολογιστών”, :quantity => 2 }
Book2 = { :title   => “Τηλεπικοινωνίες”, :quantity => 5 }
…
Book100 = { :title => “Java σε 24 ώρες”, :quantity => 3 }
```

Τα σύμβολα :title, :quantity, δημιουργούνται μόνο μια φορά και αποθηκεύονται στον πίνακα συμβόλων της ruby. Γίνεται άμεσα αντιληπτό η οικονομία σε πόρους που μπορούμε να επιτύχουμε με την χρήση των συμβόλων.

1.2.5 Σχόλια

Τα σχόλια κρύβουν τις γραμμές από τον διερμηνέα της Ruby, με αποτέλεσμα να αγνοούνται από τον διερμηνέα και να μην εκτελούνται. Τα σχόλια επιτρέπουν στους προγραμματιστές να εισάγουν χρήσιμες πληροφορίες μέσα στον κώδικα που επεξηγούν τον ίδιο τον κώδικα και τον σκοπό του. Υπάρχουν 2 είδη στυλ σχολίων στην Ruby όπως φαίνεται στο παρακάτω παράδειγμα

```
#Με το σύμβολο της δίεσης δηλώνεται ένα σχόλιο
Puts “Ruby on Rails”          #Σχόλιο στην ίδια γραμμή με εντολή
#Σχόλια μπορούν
#να περιέχονται
#σε πολλές γραμμές

=begin
```

```
Το δεύτερο στυλ σχολίων
δηλώνεται με την χρήση
του μπλοκ  =begin …σχολια… =end
που μπορεί να εκτείνεται
σε πολλές γραμμές
=end
```

1.2.5 Πίνακες

Οι πίνακες αποτελούν γραμμικές διατεταγμένες λίστες αντικειμένων. Ένας πίνακας περιέχει διακριτές τιμές οι οποίες μπορούν να προσπελαστούν με την χρήση του δείκτη. Ο δείκτης ξεκινάει από το 0 (μηδέν), δηλαδή το πρώτο στοιχείο του πίνακα βρίσκεται στην θέση '0' του πίνακα. Ένας πίνακας στην ruby μπορεί να περιέχει αριθμούς, συμβολοσειρές ή και αντικείμενα μια κλάσης μέσα στον ίδιο πίνακα, δηλαδή δεν χρειάζεται να δηλώσουμε συγκεκριμένο τύπο δεδομένων που θα περιέχει ο πίνακας (όπως πχ στην java και την c++), ούτε όμως και συγκεκριμένο μέγεθος. Οποιοδήποτε αντικείμενο μπορεί να δημιουργήσει η ruby, μπορεί να περιέχεται σε έναν πίνακα !! Η ruby περιέχει αρκετές χρήσιμες μεθόδους με τις οποίες επιτελούμε διάφορες ενέργειες πάνω στους πίνακες.

```
pinakas = [1, 2, 3.14, 'ruby']
pinakas[0]                                  # => 1
pinakas[3] = 'book'                         # θέτουμε το 3ο στοιχείο
p = %w{Δίκτυα υπολογιστών}                  # p[0] => "Δίκτυα"
                                            # p[1] => "υπολογιστών"
```

1.2.6 Ευρετήρια

Τα ευρετήρια είναι παρόμοια με τους πίνακες με την διαφορά ότι οι δείκτες για την προσπέλαση τους δεν είναι ακέραιοι αριθμοί, αλλά μπορεί να είναι οποιοδήποτε αντικείμενο της Ruby. Δεν μπορούν να προσπελαστούν σειριακά, αλλά με τον δείκτη-'κλειδί' του κάθε στοιχείου. Δηλώνονται με την χρήση αγκίστρων '{', '}', και αποδίδουμε τιμή σε έναν δείκτη με την χρήση του συμβόλου '**=>**'.

```
user = Hash.new
user['username']  = "nick"
user['firstname'] = "Νίκος"
user1 = {'username'  => "nick",
         'firstname' => "Νίκος"}
```

```
user.empty?                          # => false, δεν είναι άδειος
user1.length                               # => 2
user1.size                                 # => 2
```

1.2.7 Δομές ελέγχου

Η Ruby διαθέτει όλες τις γνωστές δομές ελέγχου που περιέχονται σε αναγνωρισμένες γλώσσες προγραμματισμού, τόσο δομές επιλογής όσο και δομές επανάληψης.

Δομές επιλογής :

Δομή **if … else**

Η πιο κλασική και γνωστή δομή επιλογής που χρησιμοποιείται σε όλες τις γλώσσες προγραμματισμού. Η σύνταξη της στην ruby διαφέρει ελάχιστα με τις άλλες γλώσσες.

```
#Κλασικός τρόπος
if 1 == 1 [then]                           #το then είναι προαιρετικό
    puts “Σωστό”
end

#Σε μία γραμμή
if 1 == 1 then puts “Σωστο” [end]        #το end είναι προαιρετικό
if 1 == 1 :    puts “Σωστο” [end]

#Μετά από μία εντολή
puts “Σωστό” if 1 == 1
```

Δομή **if … elsif**

Όταν έχουμε παραπάνω από δύο περιπτώσεις χρησιμοποιούμε την ανεπτυγμένη μορφή της if, δηλώνοντας τις επιπλέον περιπτώσεις με την λέξη ‘elsif’ (και όχι ‘else if’ ή ‘elseif’)

```
#Κλασικός τρόπος
```

```
if i > 0                                    #το then είναι προαιρετικό
    puts “Θετικός”
elsif i < 0
    puts “Αρνητικός”
else
    puts “Μηδέν”
end
```

Δομή .. ? .. :

Η δομή .. ? .. : , που είναι γνωστή ως ‘ternary operator’ σε άλλες γλώσσες όπως η c++ και η java, χρησιμοποιείται αντί της if..else σαν ένας πιο κομψός τρόπος διατύπωσης όταν έχουμε μόνο 2 διαφορετικές καταστάσεις. Όταν είναι true η παράσταση στο αριστερό μέλος τότε εκτελείται η εντολή μετά τον χαρακτήρα ‘?’, αλλιώς εκτελείται η εντολή μετά τον χαρακτήρα ‘:’ .

```
# συνθήκη
i >= 0     ?  puts “Μη αρνητικός”  : puts “Αρνητικός”
                #αν είναι true           #αν είναι false
```

Δομή case :

Αντί για την δομή if..elsif μπορεί να χρησιμοποιηθεί η δομή case που παρέχει πιο καθαρό τρόπο διατύπωσης και είναι κατάλληλη όταν υπάρχουν πολλές διαφορετικές περιπτώσεις που ισχύουν

```
vathmos = 7                         # ανάλογα με την τιμή
case vathmos                        # της μεταβλητής, θα
  when    0: puts "χειριστος"       # εκτελεστεί η κατάλληλη
  when 1..3: puts "πολύ κακώς"      # περίπτωση
  when 4..5: puts "κακώς"
  when 5..7: puts "καλώς"
  when 8..9: puts "λίαν καλώς"
  when   10: puts "αριστα"
  else       puts "εκτός κλίμακας"
end
                                    # => ‘καλώς’
```

```

```

Δομές επανάληψης :

Δομή **while :**

Η κλασική δομή επανάληψης που περιέχεται στις περισσότερες γνωστές γλώσσες προγραμματισμού. Ο βρόχος επαναλαμβάνεται συνεχώς όσο η συνθήκη που έχουμε ορίσει είναι αληθής. Μπορεί να δηλωθεί με δύο τρόπους, ανάλογα που θέλουμε να τοποθετήσουμε την συνθήκη, όπως φαίνεται στο παράδειγμα

```
while [συνθήκη] do                  # Ο βρόχος μπορεί να μην
  #εντολές                          # εκτελεστεί καθόλου
end

begin                               # Ο βρόχος θα εκτελεστεί
  #εντολές                          # τουλάχιστον μια φορά
end while [συνθήκη]
```

Δομή **until** ή **unless :**

Η δομή until (ή unless, και οι δύο δεσμευμένες λέξεις έχουν την ίδια χρήση) είναι παρόμοια με την δομή while, μόνο που ο βρόχος εκτελείται όσο η συνθήκη που έχουμε δηλώσει είναι ψευδής (και όχι αληθής).

```
until [συνθήκη] do                  # Όσο η συνθηκη είναι ψευδής
  #εντολές                          # ο βρόχος θα εκτελείται
end

begin                               # Μπορούν να χρησιμοποιηθούν
  #εντολές                          # και οι δύο τρόποι δήλωσης
end unless [συνθήκη]                # και από τις δύο λέξεις
                                   # (είτε until είτε unless)
```

Δομή **for :**

Επίσης άλλη μια γνωστή δομή στους προγραμματιστές, η επαναληπτική δομή for, την οποία χρησιμοποιούμε όταν γνωρίζουμε εκ των προτέρων τον αριθμό των επαναλήψεων του βρόχου. Στην ruby διαφέρει ο τρόπος δηλωσής της όπως φαίνεται παρακάτω.

```
# Ο βρόχος θα εκτελεστεί 10 φορές, δεν χρειάζεται να
# αρχικοποιησουμε την μεταβλητή i, παίρνει αυτόματα την
# πρώτη τιμή του εύρους που δηλώνουμε, με βήμα αύξησης 1
for i in 1..10 [do]                 # Όσο η συνθηκη είναι ψευδής
  #εντολές                          # ο βρόχος θα εκτελείται
end


for i in 1..10 do #εντολή end # Συνεπτυγμένη μορφή
```

1.3 Κλάσεις, αντικείμενα και μέθοδοι

Όπως έχει ήδη αναφερθεί η Ruby είναι μια πλήρως αντικειμενοστραφής γλώσσα. Τα πάντα θεωρούνται αντικείμενα γιατί κληρονομούν τις μεθόδους και τις ιδιότητες της κλάσης Object.

1.3.1 Κλάσεις

Ο ορισμός μιας κλάσης γίνεται με την χρήση της δεσμευμένης λέξης **class** και της λέξης **end**, που δηλώνει το τέλος της δήλωσης της κλάσης, αλλά και οποιουδήποτε μπλοκ κώδικα. Το όνομα της κλάσης θα πρέπει να ξεκινάει με κεφαλαίο γράμμα (γιατί θεωρείται σταθερά, και οι σταθερές πρέπει να ξεκινούν με κεφαλαίο γράμμα).

```
class Book                    # Οι αγκύλες δεν είναι απαραίτητες
    # Ορισμός των ιδιοτήτων και
    # των μεθόδων της κλάσης
end
```

1.3.2 Ιδιότητες

Κάθε κλάση μπορεί να περιέχει ιδιότητες-μεταβλητές που είναι ορατές μόνο μέσα από το ίδιο αντικείμενο της κλάσης. Οι μεταβλητές αυτές ξεκινάνε με τον χαρακτήρα '@', και δεν χρειάζεται να τις δηλώσουμε

```
class Book
    @title = "Δίκτυα υπολογιστών"
end
```

1.3.2 Μέθοδοι

Μια κλάση περιέχει μεθόδους οι οποίες παρέχουν ενέργειες που μπορούν να εκτελεστούν στα αντικείμενα της κλάσης. Το σύνολο των μεθόδων αποτελεί την συμπεριφορά της κλάσης. Η δήλωση μιας μεθόδου γίνεται με την χρήση της δεσμευμένης λέξης **def** … **end**. Με τον ίδιο τρόπο δηλώνουμε τις συναρτήσεις, δηλαδή μεθόδους που δεν ανήκουν σε κάποια κλάση. Μπορεί να φαίνεται οξύμωρο στην αρχή (η ύπαρξη συναρτήσεων), καθώς έχουμε πει ότι τα πάντα στην ruby είναι αντικείμενα, αλλά μια συνάρτηση είναι στην ουσία μια μέθοδος που ανήκει στην κλάση Object (την οποία κληρονομούν όλες οι κλάσεις). Η ορατότητα των μεθόδων μπορεί να έχει 3 καταστάσεις:

- Δημόσιες (Public) είναι οι μέθοδοι που μπορούν να κληθούν από οποιοδήποτε αντικείμενο ή μέθοδο του προγράμματος
- Ιδιωτικές (Private) είναι οι μέθοδοι που μπορούν να κληθούν μόνο από τις υπόλοιπες μεθόδους που βρίσκονται στην κλάση τους.
- Προστατευμένες (Protected) είναι οι μέθοδοι που μπορούν να κληθούν από τις μεθόδους που βρίσκονται στην κλάση τους αλλά και όσες βρίσκονται σε υποκλάσεις της.

```
class Book
    @title = "Δίκτυα υπολογιστών"
    @price = 0.0

    def change_price(new_p)
        @price = new_p
    end
end
```

```
def do_something                    # Στην ουσία η do_something
    #εντολές                        # δεν είναι συνάρτηση
end                                 # αλλά μέθοδος της κλάσης
                                    # Object
```

1.3.2 Ενσωμάτωση άλλου κώδικα

Η εντολή ‘require’ είναι παρόμοια με την εντολή ‘include’ της C και της C++, και με την εντολή ‘import’ της Java. Με την χρήση της μπορούμε να συμπεριλάβουμε στον κώδικά μας τις υπάρχουσες βιβλιοθήκες της γλώσσας ή οποιοδήποτε αρχείο κώδικα ruby επιθυμούμε.

```
require ‘Math.rb’     # Μπορούμε να χρησιμοποιήσουμε τις
                      # μεθόδους της κλάσης Math στον
                      # κωδικά μας
```

1.3.2 Κληρονομικότητα

Σε όλες τις αντικειμενοστραφείς γλώσσες υπάρχει η δυνατότητα μία κλάση να κληρονομεί ιδιότητες και μεθόδους από άλλες κλάσεις. Το χαρακτηριστικό αυτό ονομάζεται κληρονομικότητα και βοηθά στην δομημένη ανάπτυξη και στην αποφυγή συγγραφής του ίδιου κώδικα πολλές φορές. Η Ruby υποστηρίζει απλή κληρονομικότητα όπως η Java, και όχι πολλαπλή κληρονομικότητα όπως η C++. Μία κλάση μπορεί να κληρονομεί μόνο από μία άλλη κλάση. Η δήλωση της κληρονομικότητας επιτυγχάνεται με τον χαρακτήρα ‘<’.

```
class Book < Product                # Η κλάση Book κληρονομεί
end                                 # από την κλάση Product
```

Επίλογος

Η ruby όπως διαπιστώσαμε είναι μια δυνατή και πλήρης αντικειμενοστραφής γλώσσα προγραμματισμού με πλούσια χαρακτηριστικά και δυνατότητες που δεν έχει τίποτα να ζηλέψει από τις παραδοσιακές ‘ισχυρές’ γλώσσες προγραμματισμού, όπως η C, η C++ και η Java. Στο κεφάλαιο που ακολουθεί θα δούμε τον λόγο που εκτόξευσε την δημοτικότητα της Ruby, η εμφάνιση του πλαισίου ανάπτυξης διαδικτυακών εφαρμογών Ruby on Rails.

2 ΤΟ ΠΕΡΙΒΑΛΛΟΝ RUBY ON RAILS

Σε αυτό το κεφάλαιο πραγματοποιούμε μια εκτενή εισαγωγή στο πλαίσιο ανάπτυξης Ruby on Rails. Αναλύεται η δομή των εφαρμογών Ruby on Rails, ο τρόπος λειτουργίας τους και τα επιμέρους συστατικά από τα οποία αποτελούνται. Επιπλέον εμβαθύνουμε στα βασικά συστατικά του Rails και στον τρόπο που χρησιμοποιούνται για να διευκολύνουν την ανάπτυξη μιας εφαρμογής.

2.1 Βασικά στοιχεία & Ιστορική αναδρομή

2.1.1 Τι είναι το Ruby on Rails

Το Ruby on Rails είναι ένα πλαίσιο ανάπτυξης διαδικτυακών εφαρμογών ανοικτού κώδικα που βασίζεται στην γλώσσα προγραμματισμού Ruby, το οποίο ακολουθεί συγκεκριμένα πρότυπα, δοκιμασμένες αρχές σχεδίασης και ευέλικτες μεθοδολογίες για την ταχεία ανάπτυξη διαδικτυακών εφαρμογών. Αποτελεί ένα ενεργό project που αναπτύσσεται και εξελίσσεται συνεχώς από την κοινότητα προγραμματιστών. Η απήχηση του στην κοινότητα των προγραμματιστών συνεχώς αυξάνεται λόγω της αυξημένης παραγωγικότητας που επιτυγχάνεται με την χρήση του.

2.1.2 Ιστορία του Ruby on Rails

Το Rails προέκυψε από τον Δανό προγραμματιστή David Heinemeier Hansson από την δουλειά του πάνω στο 'Basecamp', μια διαδικτυακή εφαρμογή διαχείρισης - οργάνωσης έργων (project management). Ο αρχικός πηγαίος κώδικας δημοσιεύτηκε στο διαδίκτυο τον Ιούλιο του 2004. Η έκδοση 1.0 που αποτέλεσε την πρώτη οργανωμένη προσπάθεια του Rails δημοσιεύτηκε στις 13 Δεκεμβρίου 2005. Από τότε συνεχίζεται διαρκώς η ανάπτυξη του, στην οποία συμμετέχει αυξανόμενο πλήθος προγραμματιστών. Ορόσημο στην ιστορία του αποτελεί η ανακοίνωση της Apple τον Αύγουστο του 2006, για την ένταξη του Rails στο λειτουργικό της Mac OS X v10.5 . Η εξέλιξη του Rails φαίνεται στον πίνακα παρακάτω

- *Σπουδαιότερες δημοσιευμένες εκδόσεις* :

Πίνακας 2 "Εκδόσεις Ruby on Rails"

Έκδοση	Ημερομηνία
1.0	13 Δεκεμβρίου 2005
1.2	19 Ιανουαρίου 2007
2.0	7 Δεκεμβρίου 2007
2.1	1 Ιουνίου 2008
2.2	21 Νοεμβρίου 2008
2.3	16 Μαρτίου 2009
3.0	29 Αυγούστου 2010

2.1.3 Σκοπός δημιουργίας

Όπως αναφέρθηκε προηγουμένως το Ruby on Rails δημιουργήθηκε σιγά σιγά από την δουλειά του Hansson πάνω στο Basecamp ([12]). Δεν δημιουργήθηκε από την αρχή για να αποτελέσει ένα ακόμα πλαίσιο ανάπτυξης διαδικτυακών εφαρμογών. Αντιθέτως δημιουργήθηκε από ένα σύνολο αρχείων και βιβλιοθηκών που βοηθούσαν τον Hansson να παράγει πιο γρήγορα και εύκολα αποτελεσματικό κώδικα πάνω στα έργα που διαχειριζόταν. Στην ουσία δημιουργήθηκε σαν βάση κώδικα που βοηθούσε τον ίδιο και τους συνεργάτες του στην εταιρία 37signals να φτιάξουν μια συγκεκριμένη υποδομή για τα έργα που ανέπτυσσαν.

2.1.4 Χαρακτηριστικά του Ruby on Rails

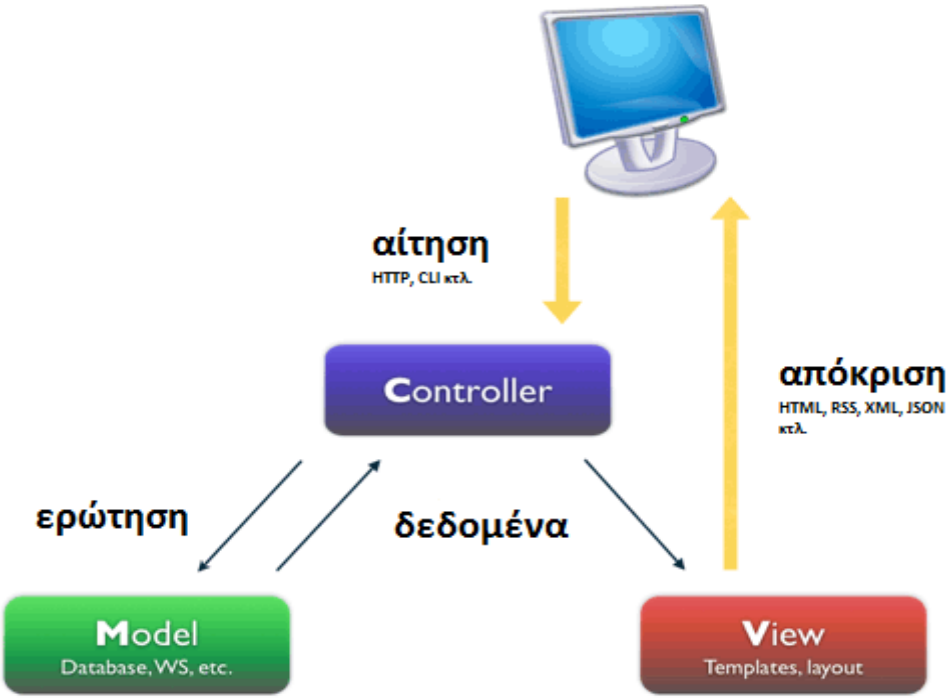
Το πλαίσιο Ruby on Rails αναπτύχθηκε για να βοηθήσει στην γρήγορη ανάπτυξη εφαρμογών. Ακολουθεί συγκεκριμένες αρχές σχεδίασης για να το επιτύχει, όπως την Προτυποποίηση – έναντι – Διαμόρφωσης (COC). Χρησιμοποιεί το πρότυπο Μοντέλο – Προβολή – Ελεγκτής (MVC) για την αρχιτεκτονική της εφαρμογής και τον διαχωρισμό του κώδικα σε συγκεκριμένες μονάδες. Ενσωματώνει στην φιλοσοφία του την αρχή σχεδίασης DRY, που αποτρέπει την επανάληψη της ίδιας γνώσης. Διαθέτει εξελιγμένη βιβλιοθήκη που διαχειρίζεται την πρόσβαση και την διαχείριση των δεδομένων της εφαρμογής. Περιέχει βιβλιοθήκες και συστατικά που χρησιμοποιούνται ευρέως στις διαδικτυακές εφαρμογές. Γενικά ακολουθεί τις τάσεις της αγοράς στην ανάπτυξη διαδικτυακών εφαρμογών και τις ενσωματώνει γρήγορα στην δομή του.

2.2 Το πρότυπο σχεδίασης Μοντέλο – Προβολή – Ελεγκτή (MVC)

2.2.1 Εισαγωγή στο πρότυπο

Το πρότυπο σχεδίασης Μοντέλο - Προβολή - Ελεγκτή (Model-Controller-View) αναφέρεται στην μεθοδολογία δόμησης και διαχωρισμού της εφαρμογής σε τρία ‘υποσυστήματα’, τα δεδομένα (μοντέλο-model), την συμπεριφορά (ελεγκτής-controller) και την διάδραση με τον χρήστη (προβολή-view) . Ο διαχωρισμός αυτός επιτρέπει οι αλλαγές σε ένα υποσύστημα να μην επηρεάζουν τα άλλα δύο υποσυστήματα. Κάθε υποσύστημα μπορεί να λειτουργεί αυτόνομα, αλλά και να συνεργάζεται με τα άλλα δύο.

Το πρότυπο MVC έχει υιοθετηθεί σε πολλά πλαίσια ανάπτυξης διαδικτυακών εφαρμογών, με διαφορετικό τρόπο ίσως στην υλοποίηση του, άλλα ακολουθώντας την ίδια λογική. Άλλωστε κατά την ανάπτυξη διαδικτυακών εφαρμογών είναι πολύ πιθανό να παρουσιαστεί το φαινόμενο να γράψουμε ένα κομμάτι κώδικα, στο οποίο περιέχεται κώδικας που στέλνει ερωτήματα στην βάση δεδομένων, κώδικας που πραγματοποιεί αυθεντικοποίηση του χρήστη και html κώδικας. Χρησιμοποιώντας το πρότυπο αυτό επιτυγχάνουμε το διαχωρισμό τους σε σαφή και συγκεκριμένα επίπεδα (κλάσεις και ξεχωριστά αρχεία).



Σχήμα 1 "Αναπαράσταση του προτύπου MVC"

2.2.2 Μοντέλο (Model)

Το μοντέλο είναι υπεύθυνο για την διαχείριση των δεδομένων της εφαρμογής. Το μοντέλο δημιουργεί, επεξεργάζεται και διαγράφει τα δεδομένα της εφαρμογής, συνήθως κάνοντας χρήση ενός συστήματος βάσεων δεδομένων. Το μοντέλο είναι κάτι παραπάνω από τα δεδομένα, επιβάλει την επιχειρησιακή λογική (επιχειρησιακούς κανόνες - business logic) στα δεδομένα. Για παράδειγμα αν θέλουμε οι παραγγελίες πάνω από 100 € να έχουν έκπτωση 5%, το μοντέλο είναι αυτό που θα επιβάλει αυτόν τον κανόνα. Με την επιβολή των επιχειρησιακών κανόνων και την διαχείριση των δεδομένων να πραγματοποιούνται μόνο στο μοντέλο, μπορούμε να είμαστε σίγουροι ότι κανένα άλλο υποσύστημα της εφαρμογής δεν μπορεί να καταστήσει τα δεδομένα μας ανακριβή. Μπορούμε να παρομοιάσουμε το μοντέλο σαν ένα φύλακα ανάμεσα στον έξω κόσμο και την αποθήκη δεδομένων μας. Το Rails επιτυγχάνει αυτή την λειτουργικότητα με την βιβλιοθήκη Active Record που θα αναλυθεί παρακάτω.

2.2.3 Ελεγκτής (Controller)

Ο Ελεγκτής είναι ο ‘μαέστρος’ της εφαρμογής μας. Οι ελεγκτές δέχονται τις ενέργειες-αιτήσεις του χρήστη, αλληλεπιδρούν με το μοντέλο και παρουσιάζουν την κατάλληλη προβολή στον χρήστη. Λειτουργεί ως ενδιάμεσος ανάμεσα στο μοντέλο και την προβολή, κάνοντας διαθέσιμα τα δεδομένα του μοντέλου στην προβολή για να παρουσιαστούν στον χρήστη. Εκτός από την αλληλεπίδραση με τα δεδομένα της εφαρμογής, παρέχει και υπηρεσίες που είναι απαραίτητες για την λειτουργία μιας εφαρμογής. Στο πλαίσιο Ruby on Rails υλοποιείται με την χρήση της βιβλιοθήκης Action Controller.

2.2.4 Προβολή (View)

Η προβολή αποτελεί την διεπιφάνεια διάδρασης με τον χρήστη. Στην ουσία αποτελεί την ‘εικόνα’ – το ‘γραφικό περιβάλλον’ της εφαρμογής. Σε μια διαδικτυακή εφαρμογή θα έχει την μορφή ιστοσελίδων, html κώδικα που λαμβάνει ο φυλλομετρητής του χρήστη από τον διακομιστή της εφαρμογής. Οι προβολές δεν επηρεάζουν τα δεδομένα, απλώς δέχονται δεδομένα και τα επεξεργάζονται κατάλληλα για να παρουσιαστούν στον χρήστη. Το Rails χρησιμοποιεί το συστατικό Action View και μια σειρά βιβλιοθηκών για να μας βοηθήσει στην σχεδίαση σύγχρονων και λειτουργικών ιστοσελίδων, που ακολουθούν τις τάσεις της αγοράς (web 2.0, ajax κτλ.).

2.3 Αρχές Σχεδίασης

2.3.1 DRY (Μην επαναλαμβάνεις τον εαυτό σου)

Η αρχή σχεδίασης DRY στοχεύει στην μείωση επαναλαμβανόμενου κώδικα, με οφέλη όπως η μεγαλύτερη παραγωγικότητα, η μείωση λαθών και η πιο εύκολη και γρήγορη διατήρηση και διαχείριση του κώδικα. Μπορούμε να το εκφράσουμε σαν ‘κάθε κομμάτι γνώσης σε ένα σύστημα, πρέπει να δηλώνετε σε ένα μόνο μέρος’ (Hansson Heinemeier et al., 2011). Αυτό σημαίνει ότι δεν θα πρέπει να γράφουμε τον ίδιο κώδικα σε δύο διαφορετικά μέρη, δεν θα πρέπει να υπάρχει η πληροφορία σε δύο ή περισσότερα μέρη. Το Rails μας προσφέρει κατάλληλα εργαλεία για να ακολουθήσουμε αυτή την αρχή με συνέπεια στις εφαρμογές μας.

2.3.2 COC (Προτυποποίηση αντί διαμόρφωσης)

Το Rails χρησιμοποιεί στην φιλοσοφία του την αρχή σχεδίασης COC (Convention over Configuration) ή αλλιώς Προτυποποίηση - αντί - Διαμόρφωσης ή Σύμβαση - έναντι - Ρύθμισης. Το COC αποτελεί μια αρχή σχεδίασης που ελαχιστοποιεί την ανάγκη διαμόρφωσης. Αυτό σημαίνει ότι το Rails έχει προεπιλεγμένες τιμές και συμπεριφορές για οτιδήποτε σχετικό με μία εφαρμογή, μειώνοντας τον χρόνο που απαιτείται για την διαμόρφωση μιας εφαρμογής από την πλευρά των προγραμματιστών.

Η κοινότητα προγραμματιστών του Rails διέκρινε ότι, τις περισσότερες φορές, οι σχέσεις μεταξύ των μερών του προτύπου MVC είναι προφανείς, επαναλαμβανόμενες, και δεν χρειάζεται να απαιτούν διαμόρφωση. Η διαπίστωση αυτή οδήγησε τους προγραμματιστές του Rails να δημιουργήσουν ένα σύνολο συμβάσεων σχετικά με το πώς υλοποιούνται κοινά στοιχεία εφαρμογής: πρότυπα για την ονομασία των κλάσεων, την ονομασία των ελεγκτών και των ενεργειών τους, για τα ονόματα των αρχείων, και την δομή μιας εφαρμογής.

Ας πάρουμε για παράδειγμα την οντότητα βιβλία (books) της εφαρμογής μας. Ο πίνακας στην βάση μας ονομάζεται `books`, το μοντέλο ονομάζεται `Book`, ο ελεγκτής ονομάζεται `BooksController` και οι προβολές βρίσκονται στον φάκελο `app/views/books/*`.

Το μοντέλο **Book**

Όνομα αρχείου : **book.rb**

Διαδρομή αρχείου : app/models/**book.rb**

```
class Book < ActiveRecord::Base
end
```

Ο ελεγκτής για το μοντέλο **Book**

Όνομα αρχείου : **books**_controller.rb

Διαδρομή αρχείου : app/controllers/**books**_controller.rb

```
class BooksController < ApplicationController
    def search
    end

    def show
```

```
    end
end
```

Οι προβολές για το μοντέλο **Book**

Όνομα αρχείων : **μέθοδος_του_ελεγκτή**.html.rb

Διαδρομή αρχείων : app/views/books/**μέθοδος_του_ελεγκτή**.html.rb

```
# Στις προβολές περιέχεται κώδικας html μαζί με ενσωματωμένο κώδικα ruby
    Αποτελέσματα αναζήτησης

# Με την χρήση των <%= … %> ή <% … %> ενσωματώνουμε κώδικα ruby
<%  render              'comp/books' %>
```

2.3.3 REST

Το REST είναι το ακρωνύμιο για την έκφραση REpresentational State Transfer. Είναι ένα αρχιτεκτονικό στυλ για κατανεμημένα συστήματα υπερμέσων, όπως ο Παγκόσμιο Ιστός (World Wide Web). Χρησιμοποιώντας την φιλοσοφία του REST σε μια εφαρμογή την οργανώνουμε σε πόρους (resources) πάνω στους οποίους εκτελούμε συγκεκριμένες εργασίες. Η αναφορά στους πόρους γίνεται με την χρήση ενός καθολικού ταυτοποιητή (πχ. URI). Σε μια διαδικτυακή εφαρμογή οι πόροι είναι στην ουσία οι οντότητες της εφαρμογής μας και οι ενέργειες πάνω σε αυτές καλούνται με την χρήση κατάλληλων http μεθόδων (ή ρημάτων) και διευθύνσεων (Carlson et al., 2008). Ο σημαντικότερος πόρος στην εφαρμογή ενός βιβλιοπωλείου είναι ‘το βιβλίο’. Πάνω σε αυτόν τον πόρο μπορούμε να εκτελέσουμε κάποιες ενέργειες. Οι ενέργειες που είναι απαραίτητες για την διαχείριση ενός πόρου καθώς και η αντιστοίχιση τους με ερωτήματα SQL, μεθόδους του Rails και τα ρήματα του πρωτοκόλλου HTTP φαίνονται στον πίνακα παρακάτω.

Πίνακας 3 Αντιστοίχηση ρημάτων REST με ρήματα http, μεθόδους rails και εντολές βάσης δεδομένων

	Create	Read	Update	Delete
HTTP	POST	GET	PUT	DELETE
Rails	CREATE	SHOW	UPDATE	DESTROY
Βάση δεδομ.	INSERT	SELECT	UPDATE	DELETE

2.4 Συστατικά του Ruby on Rails

Το πλαίσιο Ruby on Rails αποτελείται από τα εξής συστατικά ([21]):

- Action Pack
  - Action Controller
  - Action Dispatch
  - Action View
- Action Mailer
- Active Model
- Active Record
- Active Resource
- Active Support
- Railties

Action Pack

Το Action Pack είναι ένα ενιαίο συστατικό (gem) που περιλαμβάνει τα Action Controller, Action View και Action Dispatch. Αποτελεί το μέρος “VC” από το πρότυπο MVC.

Action Controller

Το Action Controller είναι το συστατικό που διαχειρίζεται του ελεγκτές σε μία Ruby on Rails εφαρμογή. Επεξεργάζεται τις εισερχόμενες αιτήσεις, εξάγει παραμέτρους και τις στέλνει στην κατάλληλη ενέργεια. Επίσης παρέχει υπηρεσίες όπως διαχείριση συνεδριών (sessions) και ανακατευθύνσεων (redirect).

Action View

Το Action View είναι το συστατικό που διαχειρίζεται τις προβολές μιας Ruby on Rails εφαρμογή. Μπορεί να παράγει html και xml περιεχόμενο με τις προεπιλεγμένες ρυθμίσεις. Διαχειρίζεται τα πρότυπα (templates) και παρέχει ενσωματωμένες δυνατότητες για αιτήσεις μέσω ajax.

Action Dispatch

Το Action Dispatch είναι υπεύθυνο για την δρομολόγηση των αιτήσεων. Δέχεται τις αιτήσεις από τον χρήστη και τις διανέμει στην εφαρμογή.

Action Mailer

Το Action Mailer είναι μια βιβλιοθήκη για δημιουργία υπηρεσιών ηλεκτρονικών μηνυμάτων (email). Χρησιμοποιείται για να στέλνουμε και να λαμβάνουμε email με απλό κείμενο ή πιο σύνθετο περιεχόμενο (πχ. αρχεία) από και προς την εφαρμογή μας.

Active Model

Το Active Model παρέχει μια προκαθορισμένη διεπαφή μεταξύ των λειτουργιών του συστατικού (gem) Action Pack και των ORM συστατικών όπως το Active Record. Λόγω αυτής της διεπαφής μας επιτρέπει να χρησιμοποιήσουμε διαφορετικές ORM βιβλιοθήκες – συστατικά στην θέση του Active Record.

Active Record

Το Active Record είναι η βάση για τα μοντέλα σε μια Rails εφαρμογή. Παρέχει ανεξαρτησία από τα συστήματα βάσεων δεδομένων, βασικές δυνατότητες αλληλεπίδρασης με την βάση (CRUD – Δημιουργία - Ανάγνωση – Ενημέρωση - Διαγραφή), προηγμένες δυνατότητες ερωτημάτων (SQL) προς την βάση, και μας δίνει την ικανότητα να ορίσουμε συσχετίσεις μεταξύ των μοντέλων (πινάκων).

Active Resource

Το Active Resource είναι μια βιβλιοθήκη που διαχειρίζεται την σύνδεση της εφαρμογής μας με εξωτερικές ηλεκτρονικές υπηρεσίες (web services). Επίσης μπορούμε να δημιουργήσουμε τις δικές μας ηλεκτρονικές υπηρεσίες στα πλαίσια της εφαρμογής μας.

Active Support

Το Active Support αποτελεί μια εκτεταμένη συλλογή από χρήσιμες βοηθητικές κλάσεις και επεκτάσεις της Ruby που χρησιμοποιούνται τόσο από τον πυρήνα του Rails όσο και από τις εφαρμογές μας.

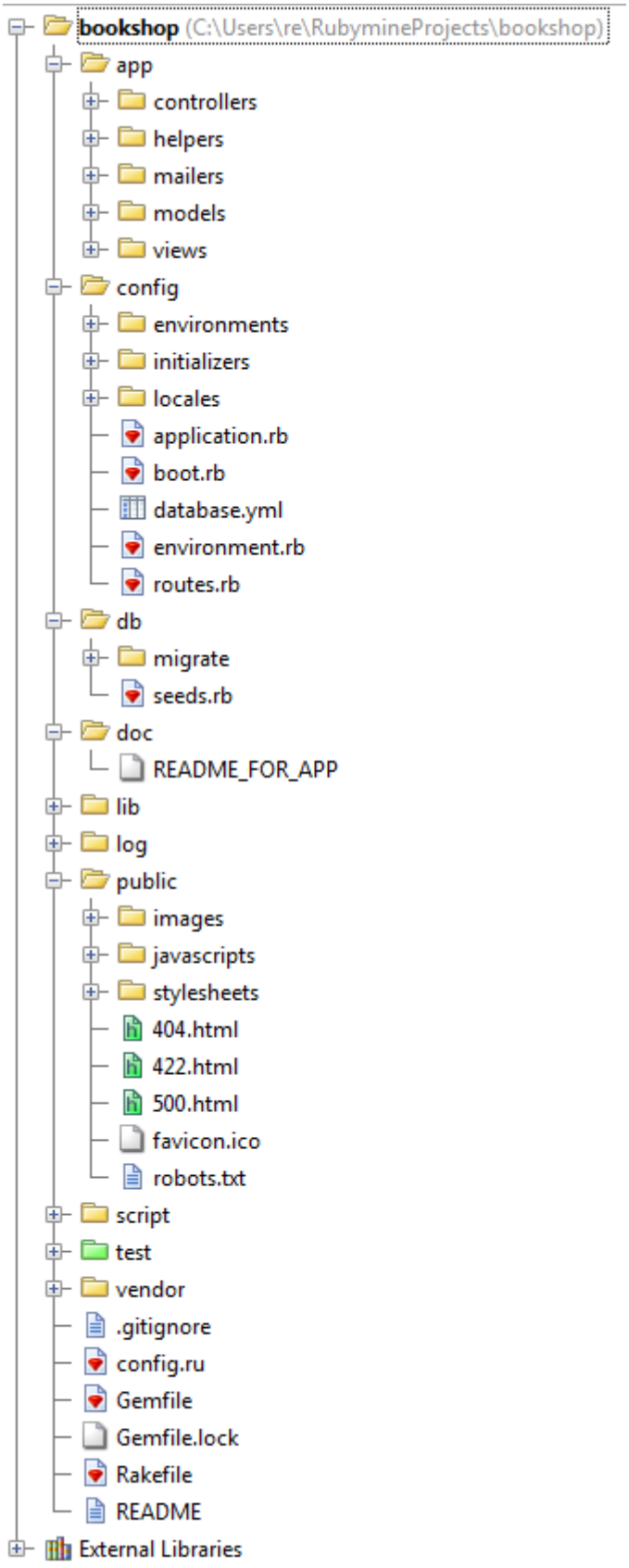
Railties

Το Railties είναι ο πυρήνας του πλαισίου Ruby on Rails. Δημιουργεί νέες εφαρμογές και ενώνει τις επιμέρους βιβλιοθήκες και τα συστατικά του πλαισίου μαζί.

Αν και όλα τα συστατικά του πλαισίου είναι χρήσιμα μπορούμε να διαπιστώσουμε ότι τα κυριότερα συστατικά του πλαισίου Rails είναι το Active Record και το Action Pack το οποίο αποτελείται από τα Action Controller, Action View, και Action Dispatch. Τα τρία αυτά συστατικά συνεργάζονται μεταξύ τους, το Action Dispatch δρομολογεί τις αιτήσεις στους κατάλληλους ελεγκτές, το Action Controller δημιουργεί τις απαντήσεις στις αιτήσεις και το Action View διαμορφώνει τον τρόπο παρουσίασης των απαντήσεων. Παρακάτω θα δούμε αναλυτικά το κάθε συστατικό.

2.5 Δομή της εφαρμογής

Το Rails χρησιμοποιεί μια συγκεκριμένη δομή καταλόγων της εφαρμογής που μας βοηθά στην καλύτερη δόμηση της εφαρμογής. Στο σχήμα 2 βλέπουμε την δομή της εφαρμογής του βιβλιοπωλείου.



Σχήμα 2 "Δομή εφαρμογής Ruby on Rails"

Στον παρακάτω πίνακα παρουσιάζεται με συνοπτική περιγραφή του περιεχομένου του κάθε φακέλου της εφαρμογής και των σημαντικότερων αρχείων της.

Πίνακας 4 "Δομή καταλόγων εφαρμογής"

Αρχείο/Κατάλογος	Περιεχόμενα
app/	Βασικός κώδικας της εφαρμογής, εδώ βρίσκονται οι ελεγκτές, τα μοντέλα, οι προβολές και οι βοηθοί
app/controllers	Οι ελεγκτές της εφαρμογής
app/models	Τα μοντέλα της εφαρμογής
app/views	Οι προβολές κάθε ελεγκτή
config/	Ρυθμίσεις της εφαρμογής
db/	Αρχεία διαχείρισης-επικοινωνίας με την βάση δεδομένων
doc/	Τεκμηρίωση για την εφαρμογή
lib/	Βοηθητικές βιβλιοθήκες που χρησιμοποιεί η εφαρμογή
log/	Αρχεία καταγραφής ενεργειών
public/	Αρχεία προσπελάσιμα από το διαδίκτυο, όπως εικόνες και αρχεία css
public/images	Εικόνες, γραφήματα
public/javascripts	Αρχεία javascript της εφαρμογής
public/stylesheets	Αρχεία στυλ css της εφαρμογής
script/rails	Αρχείο script που εκτελούμε όταν θέλουμε να δημιουργήσουμε κώδικα ή να ξεκινήσουμε τον τοπικό διακομιστή
test/	Αρχεία τεστ της εφαρμογής
tmp/	Προσωρινά αρχεία, όπως αρχεία συνεδριών (sessions)
vendor/	Κώδικας τρίτων, όπως τα plugins (πρόσθετα) και gems
README	Σύντομη περιγραφή της εφαρμογής
Rakefile	Βοηθητικά προγράμματα εκτελέσιμα μέσω της εντολής rake
Gemfile	Αρχείο που περιέχει όλες τις βιβλιοθήκες (gems) που είναι απαραίτητες από την εφαρμογή μας

2.6 Active Record

Το Active Record αποτελεί βασικότατο συστατικό του Rails. Το Active Record είναι η υλοποίηση της ORM τεχνικής στο Rails. Αντιπροσωπεύει το Model (Μοντέλο) στην αρχιτεκτονική MVC. Ακολουθεί αρκετές συμβάσεις για να ελαχιστοποιήσει την ανάγκη διαμόρφωσης, όπως παρουσιάστηκε στο παράδειγμα για το COC. Χρησιμοποιεί επίσης την δύναμη της ruby στον μεταπρογραμματισμό, για να παρέχει ευκολίες που δεν συναντώνται σε άλλα πλαίσια ανάπτυξης. Το Active Record προσθέτει ένα επίπεδο αφαίρεσης στην εφαρμογή μας, αφού λειτουργεί με πολλές βάσεις δεδομένων χωρίς να χρειαστεί να τροποποιήσουμε τον κώδικα μας. Το Active Record είναι αυτόνομη βιβλιοθήκη, η οποία μπορεί να χρησιμοποιηθεί και από οποιαδήποτε εφαρμογή ruby (όχι απαραίτητα διαδικτυακή). Συνοπτικά οι δυνατότητες του ActiveRecord αναφέρονται παρακάτω:

- Εύκολη παραμετροποίηση σε συνδυασμό με πλήθος συμβάσεων
- Αυτόματη αντιστοίχηση μεταξύ δεδομένων της βάσης και αντικειμένων
- Συσχετίσεις μεταξύ αντικειμένων
- Έλεγχος εγκυρότητας δεδομένων
- Μεταφορά βάσης δεδομένων
- Λειτουργία ανεξαρτήτως βάσης δεδομένων μέσω οδηγών (drivers)

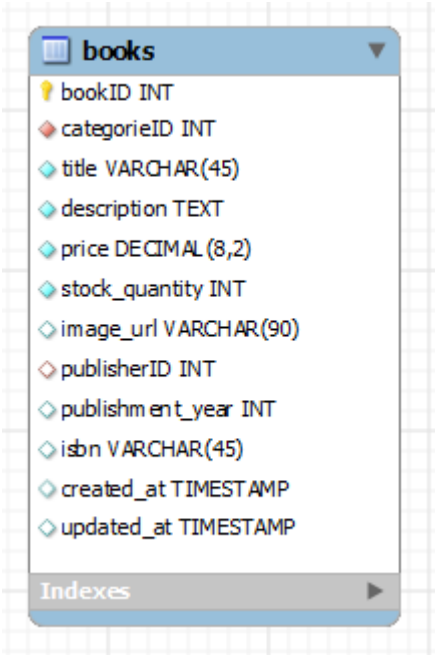
2.6.1 ORM (Object – Relational – Mapping)

Το ORM (Object-Relational Mapping) είναι μια τεχνική αντιστοίχισης - μετατροπής των δεδομένων σχεσιακών βάσεων δεδομένων σε κλάσεις αντικειμένων μιας αντικειμενοστραφούς γλώσσας προγραμματισμού, με σκοπό τον ευκολότερο χειρισμό τους μέσα στην εφαρμογή. Το Active Record ακολουθεί τα στάνταρ του ORM μοντέλου, οι πίνακες αντιστοιχούν στις κλάσεις, οι γραμμές στα αντικείμενα, και τα πεδία στις ιδιότητες των αντικειμένων. Διαφέρει όμως από τις περισσότερες ORM βιβλιοθήκες στον τρόπο με τον οποίο παραμετροποιείται. Έχοντας σαν βάση συγκεκριμένες συμβάσεις και ρυθμίσεις (θυμηθείτε την έμφαση του Rails στο COC), ελαχιστοποιεί την ανάγκη παραμετροποίησης από την πλευρά του προγραμματιστή. Ας δούμε ένα παράδειγμα χρησιμοποιώντας τις οντότητες της εφαρμογής μας.

Πίνακας 5 "Αντιστοίχηση οντοτήτων σε μοντέλα και πίνακες"

Οντότητα	Όνομα Κλάσης - Μοντέλου	Πίνακας
Συγγραφέας	Author	authors
Βιβλίο	Book	books
Κατηγορία	Category	categories
Εκδότης	Publisher	publishers
Χρήστης	User	users

Ο πίνακας που έχουμε δημιουργήσει για την οντότητα ‘βιβλίο’ μαζί με όλα τα πεδία του φαίνεται στο σχήμα 3 δεξιά. Οι πίνακες και γενικά το σχήμα της βάσης μπορεί να δημιουργηθεί μέσω της κονσόλας. Παρακάτω βλέπουμε μια συνοπτική δήλωση του μοντέλου Book.



Σχήμα 3 "Ο πίνακας books της οντότητας βιβλίο"

```
class Book < ActiveRecord::Base
    set_primary_key “bookID”
end

book = Book.find(1)
book.title = "Καινούριος Τίτλος"
book.price = 35.50
book.save
```

Το Active Record ακολουθεί μια σειρά από συμβάσεις. Για παράδειγμα δηλώνοντας την κλάση Book, περιμένει ότι θα έχουμε έναν πίνακα στην βάση μας με όνομα ‘books’ (η οντότητα στον πληθυντικό) και ότι έχει ένα πεδίο ‘id’ (κωδικός). Στην περίπτωση μας το κύριο κλειδί είναι το ‘bookID’ και όχι ‘id’ γι αυτό το δηλώνουμε μέσα στην κλάση. Ο κώδικας εκτός κλάσης βρίσκει το βιβλίο με κωδικό (bookID) 1 από τον πίνακα ‘books’, αλλάζει τον τίτλο του και την τιμή του και αποθηκεύει τις αλλαγές στον πίνακα. Η μεταβλητή book είναι ένα αντικείμενο της κλάσης Book, με ιδιότητες τα πεδία του πίνακα ‘books’. Πως προέκυψαν όμως οι ιδιότητες της κλάσης; Αφού δεν έχουμε δηλώσει καμία ιδιότητα title στην κλάση μας; Εδώ υπεισέρχεται το ActiveRecord με την βοήθεια των δυνατοτήτων της ruby στον μεταπρογραμματισμό, ανιχνεύει δυναμικά κατά την εκτέλεση της εφαρμογής το σχήμα της βάσης και εισάγει τα πεδία του πίνακα ‘books’ σαν ιδιότητες της κλάσης. Επιπλέον προσθέτει πλήθος μεθόδων που μας βοηθούν στην διαχείριση των αντικειμένων. Με την χρήση αυτών των μεθόδων ελαχιστοποιείται η ανάγκη δημιουργίας ερωτημάτων SQL προς την βάση δεδομένων.

```
book = Book.find_by_title(“Δίκτυα υπολογιστών”)
books = Book.find_by_stock_quantity(0)
# Οι μέθοδοι  find_by_title και find_by_stock_quantity προστέθηκαν
# δυναμικά στον μοντέλο-κλάση Book κατά την διάρκεια της εκτέλεσης
```

2.6.2 Μεταφορά βάσης (Migrations)

Η χρήση των migrations είναι ένας βολικός τρόπος να τροποποιήσουμε την βάση δεδομένων μας με δομημένο και οργανωμένο τρόπο. Παρέχει ένα επίπεδο αφαίρεσης αφού μας επιτρέπει αντί να γράφουμε SQL κώδικα που λειτουργεί σε συγκεκριμένο σύστημα βάσης δεδομένων (mysql, sql server κτλ.), να γράφουμε κώδικα ruby που λειτουργεί σε όλα τα υποστηριζόμενα συστήματα βάσεων δεδομένων. Τα αρχεία migrations δημιουργούνται με την χρήση του εργαλείου scaffold μέσω της κονσόλας όπως θα δούμε παρακάτω.

2.6.3 Έλεγχοι εγκυρότητας δεδομένων

Κατά την διάρκεια εκτέλεσης μιας εφαρμογής Rails δημιουργούνται, ενημερώνονται και διαγράφονται συνεχώς αντικείμενα. Οι έλεγχοι εγκυρότητας μας δίνουν την δυνατότητα να εξασφαλίσουμε ότι μόνο έγκυρα δεδομένα θα αποθηκεύονται στην βάση μας. Το Rails προσφέρει τις κατάλληλες βιβλιοθήκες (validation helpers). Ας δούμε ένα παράδειγμα με τους χρήστες (users) της εφαρμογής μας

```
class User < ActiveRecord::Base
    validates :firstname, :presence => true
    validates :email, :presence => true, :email => true
end
```

Παραπάνω δηλώνουμε ότι ο χρήστης θα πρέπει να έχει κάποιο όνομα, και κάποιο email το οποίο θα είναι έγκυρη ηλεκτρονική διεύθυνση. Η δήλωση γίνεται `validates :ιδιότητα, :έλεγχος => τιμή`. Υπάρχουν οι παρακάτω διαθέσιμοι έλεγχοι.

```
:presence => true                    # ελέγχει την ύπαρξη
:uniqueness => true                  # ελέγχει την μοναδικότητα
:numericality => true                # ελέγχει αν είναι αριθμός
:length => { :minimum => 0, maximum => 2000 } # ελέγχει το μεγεθος
:format => { :with => /.*/ }         # μορφή πεδιού, με κανονικές εκφράσεις
:inclusion => { :in => [1,2,3] }     # όρια τιμών
:exclusion => { :in => [1,2,3] }     #
:acceptance => true
:confirmation => true
```

2.6.4 Callbacks

Τα callbacks είναι μέθοδοι που εκτελούνται σε συγκεκριμένες στιγμές κατά την διάρκεια ζωής ενός αντικειμένου. Με την χρήση των callbacks μπορούμε να γράψουμε κώδικα που εκτελείται όταν ένα αντικείμενο ενός μοντέλου μας, δημιουργείται, αποθηκεύεται, ενημερώνεται και διαγράφεται.

```
# Δημιουργία ενός αντικειμένου
before_validation       # Πριν τον έλεγχο εγκυρότητας δεδομένων
after_validation        # Μετά τον έλεγχο εγκυρότητας δεδομένων
before_save             # Πρίν την αποθήκευση του αντικειμένου στην βάση
after_save              # Μετά την αποθήκευση του αντικειμένου στην βάση
before_create           # Πριν την δημιουργία του αντικειμένου
around_create           #
after_create            # Μετά την δημιουργία αντικειμένου
# Ενημέρωση ενός αντικειμένου
before_validation
after_validation
before_save
after_save
before_update           # Πριν την ενημέρωση του αντικειμενου στην βάση
around_update
after_update            # Μετά την ενημέρωση του αντικειμενου στην βάση
# Διαγραφή ενός αντικειμένου
before_destroy
after_destroy
around_destroy
```

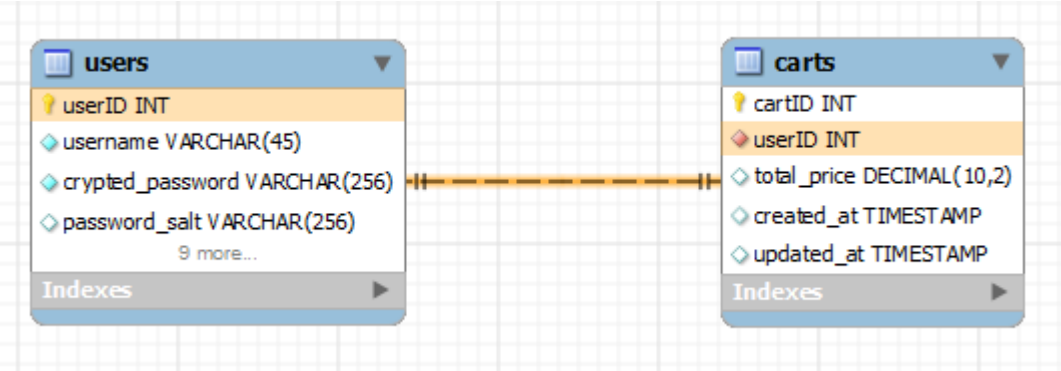
2.6.5 Συσχετίσεις

Σε ένα σχεσιακό σχήμα βάσης οι σχέσεις μεταξύ των διαφορετικών οντοτήτων υλοποιούνται με την χρήση των ξένων κλειδιών (foreign keys). Υπάρχουν 3 βασικοί τύποι συσχετίσεων μεταξύ των οντοτήτων, σχέσεις 1 προς 1, σχέσεις 1 προς πολλά και σχέσεις πολλά προς πολλά. Το Active Record παρέχει δυνατότητα να διευκρινίσουμε ακριβώς την σχέση των οντοτήτων μεταξύ τους. Χρησιμοποιούνται γι αυτόν τον σκοπό οι παρακάτω δηλώσεις.

```
belongs_to
has_one
has_many
has_many :through
has_one :through
has_and_belongs_to_many
```

belongs_to (ανήκει σε) και has_one (έχει ένα)

Οι προτάσεις belongs_to και has_one αναφέρονται στην συσχέτιση 1 προς 1. Στην εφαρμογή μας ένας χρήστης μπορεί να έχει ένα μόνο καλάθι αγορών στο κατάστημα μας, ενώ ένα καλάθι αγορών ανήκει σε έναν μόνο χρήστη. Η διαφορά τους είναι ότι το belongs_to το χρησιμοποιούμε στο μοντέλο-οντότητα το οποίο περιέχει το ξένο κλειδί (καθώς μία από τις δύο οντότητες πρέπει να περιέχει σαν ξένο κλειδί το κλειδί της άλλης), ενώ στο άλλο μοντέλο-οντότητα χρησιμοποιούμε το has_one. Για να γίνει καλύτερα αντιληπτό δείτε το παράδειγμα παρακάτω.



Σχήμα 4 "Συσχέτιση 1 προς 1"

Οι κλάσεις-μοντέλα `User` και `Cart` θα έχουν την παρακάτω μορφή

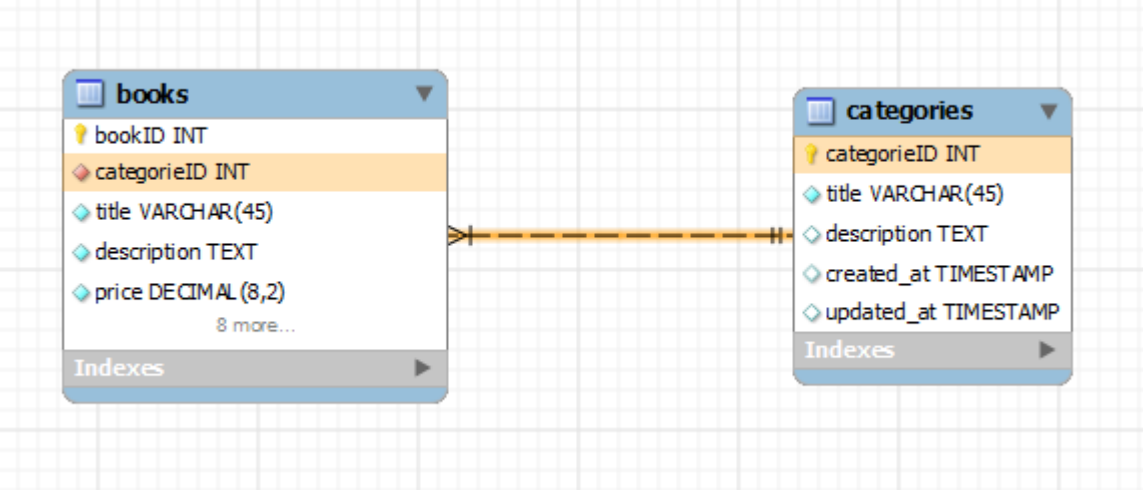
```
class User < ActiveRecord::Base
  has_one :cart
end

class Cart < ActiveRecord::Base
  belongs_to :user, :foreign_key => 'userID'
end
# Με το :foreign_key μπορούμε να δηλώσουμε
# το όνομα του ξένου κλειδιού.
```

```
# Αν είχαμε ονομάσει το ξένο κλειδί user_id, δεν θα χρειαζόταν να
# το δηλώσουμε, καθώς το Rails περιμένει το ξένο κλειδί να έχει
# σαν όνομα ‘μοντέλο_id’, το οποίο αποτελει άλλη μία σύμβαση
```

has_many (έχει πολλά)

Η πρόταση has_many αναφέρεται στην συσχέτιση 1 προς πολλά. Για παράδειγμα στην εφαρμογή του βιβλιοπωλείου, ένα βιβλίο ανήκει σε μία συγκεκριμένη κατηγορία, ενώ σε μία κατηγορία μπορούν να ανήκουν πολλά βιβλία. Η συσχέτιση φαίνεται στο παρακάτω ER διάγραμμα.



Σχήμα 5 "Συσχέτιση 1 προς πολλά"

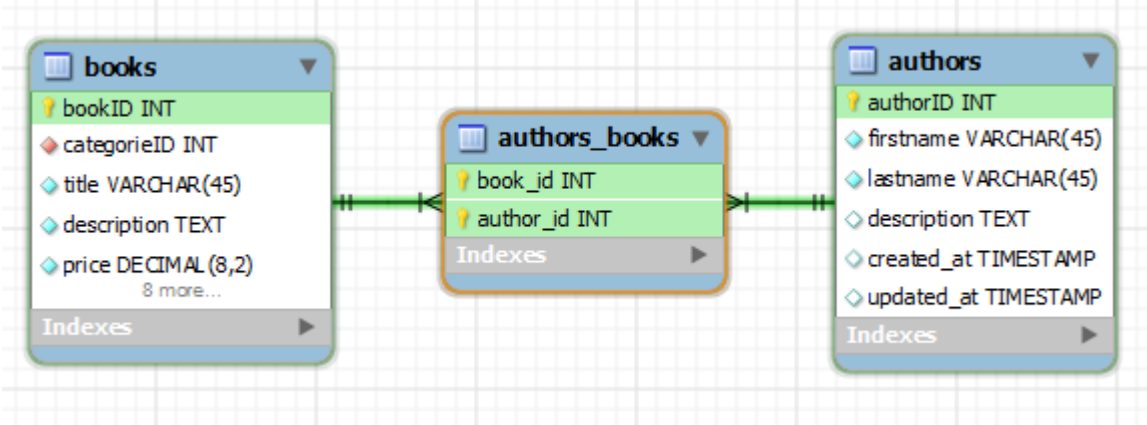
Οι κλάσεις-μοντέλα Book και Category θα έχουν την παρακάτω μορφή

```
class Book < ActiveRecord::Base
  belongs_to :category, :foreign_key => 'categorieID'
end

class Category < ActiveRecord::Base
  has_many :books, :foreign_key => 'categorieID'
end
```

has_and_belongs_to_many

Η πρόταση has_and_belongs_to_many αναφέρεται στην συσχέτιση πολλά προς πολλά. Στην εφαρμογή μας ένα βιβλίο μπορεί να έχει γραφεί από πολλούς συγγραφείς, ενώ ένας συγγραφέας μπορεί να έχει συγγράψει πολλά βιβλία. Για την σύνδεση των δύο πινάκων δημιουργείται ένας πίνακας που περιέχει τα κλειδιά τους. Το rails περιμένει ότι ο πίνακας που δημιουργείται λόγω της συσχέτισης θα ονομάζεται ως εξής 1^{ο}μοντέλο_2^{ο}μοντέλο, διατεταγμένα αλφαβητικά και τα ονόματα στον πληθυντικό αριθμό. Επίσης περιμένει ότι θα έχει δύο ξένα κλειδιά με ονομασία (όνομα μοντέλου)_id . Άλλη μία χρήση της αρχής COC (Σύμβαση έναντι Ρύθμισης). Φυσικά η προεπιλεγμένη συμπεριφορά μπορεί να αλλαχτεί. Ας δούμε ένα παράδειγμα.



Σχήμα 6 "Συσχέτιση πολλά προς πολλά"

Οι κλάσεις-μοντέλα Book και Author θα έχουν την παρακάτω μορφή

```
class Book < ActiveRecord::Base
  has_and_belongs_to_many :authors
end

class Author < ActiveRecord::Base
  has_and_belongs_to_many :books
end
```

2.6.6 Ρύθμιση παραμέτρων βάσης δεδομένων

Για να πραγματοποιηθεί μια επιτυχής σύνδεση με το σύστημα διαχείρισης βάσεων δεδομένων που θέλουμε να χρησιμοποιήσουμε και συγκεκριμένα με την βάση δεδομένων μας πρέπει να ρυθμίσουμε το ActiveRecord κατάλληλα. Σε μία εφαρμογή rails το αρχείο που πρέπει να παραμετροποιήσουμε βρίσκεται στην διαδρομή `config/database.yml` . Το αρχείο αυτό έχει την μορφή :

```
development:
  adapter: mysql        # Συστημα βάσεων δεδομένων, mysql, oracle
  encoding: utf8        # Η κωδικοποίηση που χρησιμοποιείται
  database: bookshop    # Όνομα βάσης δεδομένων
  username: *****       # Τα στοιχεία που χρησιμοποιούμε
  password: *****       # για να συνδεθούμε στην βάση
  host: localhost       # Η διεύθυνση ip (ή domain) του
                        # διακομιστή της βάσης
test:
  .
  .
production:
  .
  .
```

2.6.7 Υποστήριξη βάσεων δεδομένων

Το Active Record μπορεί να συνεργαστεί με πλήθος συστημάτων βάσεων δεδομένων. Υπάρχουν οδηγοί (drivers) για τα γνωστότερα συστήματα βάσεων. Χρησιμοποιώντας την κατάλληλη εντολή στο αρχείο `config/database.yml` που είδαμε παραπάνω καθώς και τον απαραίτητο οδηγό (driver), μπορεί να συνεργαστεί με τα παρακάτω συστήματα βάσεων δεδομένων :

- IBM DB2
- MySQL
- Microsoft SQL Server
- Oracle
- PostgreSQL
- SQLite
- Sybase

2.7 Action Controller

Το Action Controller είναι ένα ακόμη βασικό συστατικό του rails. Αποτελεί την υλοποίηση του Ελεγκτή (στο πρότυπο MVC) στο Rails. Ο ελεγκτής δέχεται τις αιτήσεις του χρήστη και κάνει τις ανάλογες ενέργειες. Στους ελεγκτές συγκεντρώνεται η συμπεριφορά της εφαρμογής. Συντονίζει την αλληλεπίδραση του χρήστη με τα δεδομένα (μοντέλα) και τις διεπιφάνειες (προβολές). Οι ελεγκτές είναι κλάσεις που κληρονομούν την κλάση ActionController::Base. Οι μέθοδοι που δηλώνονται μέσα στον κάθε ελεγκτή αποτελούν τις ενέργειες που μπορεί να εκτελέσει ο ελεγκτής.

2.7.1 Ενέργειες ελεγκτή

Οι μέθοδοι που δηλώνονται ως δημόσιες (public) στον ελεγκτή αποτελούν ενέργειες που μπορούν να κληθούν από τον χρήστη. Όταν η εφαρμογή δεχτεί μια αίτηση ο δρομολογητής αποφασίζει ποιος ελεγκτής και ποια ενέργεια θα εκτελεστεί. Τότε δημιουργείται ένα αντικείμενο του ελεγκτή και εκτελείται η μέθοδος που έχει το ίδιο όνομα με την ενέργεια. Ο ελεγκτής για την οντότητα Book της εφαρμογής μας παρουσιάζεται παρακάτω.

```
class BooksController < ApplicationController
  def search                          # Πραγματοποιεί αναζήτηση βιβλίων
    #εντολές
  end
  def show                            # Εμφανίζει ένα συγκεκριμένο βιβλίο
    #εντολές
  end
end
```

Αυτός ο ελεγκτής μας παρέχει δύο διαφορετικές ενέργειες που μπορούμε να εκτελέσουμε πάνω στα βιβλία. Να κάνουμε αναζήτηση για ένα ή περισσότερα βιβλία ή να μας εμφανίσει τα στοιχεία ενός βιβλίου. Αν ο χρήστης επισκεφτεί την διεύθυνση http://bookshop.com/books/search για να πραγματοποιήσει μια αναζήτηση βιβλίων, τότε θα δημιουργηθεί ένα αντικείμενο του ελεγκτή `BooksController` και θα εκτελεστεί η μέθοδος `search`.

2.7.2 Παράμετροι

Οι ενέργειες του ελεγκτή τις περισσότερες φορές δέχονται κάποιες παραμέτρους από τον χρήστη. Οι φυλλομετρητές χρησιμοποιούν κυρίως 2 μεθόδους του πρωτοκόλλου http για να αλληλεπιδράσουν με τον διακομιστή, το get και το post. Για να περάσουμε παραμέτρους σε μια διαδικτυακή εφαρμογή χρησιμοποιώντας αίτηση http get, χρησιμοποιούμε μέρος της διεύθυνσης (url). Για παράδειγμα για να αναζητήσουμε βιβλία με τίτλο '*Δίκτυα*', η διεύθυνση θα είχε την μορφή http://bookshop.com/books/search?title="Δίκτυα". Όποιος τρόπος και να χρησιμοποιηθεί οι παράμετροι μπορούν προσπελαστούν από το ευρετήριο `params`, χρησιμοποιώντας σαν κλειδί το όνομα της παραμέτρου.

```
class BooksController < ApplicationController
  def search
    @books = Book.find_by_title(params[:title]) # params[:title] =>
  end                                           # "Δίκτυα"
end
```

2.7.3 Αποκρίσεις στις αιτήσεις

Οι ελεγκτές μπορούν να ανταποκριθούν στις αιτήσεις του χρήστη σε πολλές μορφές, συνήθως με κώδικα html. Οι αποκρίσεις είναι οι απαντήσεις των ελεγκτών στις αιτήσεις που δέχονται από τον χρήστη. Για το προηγούμενο παράδειγμα ο ελεγκτής θα προσπελάσει το αρχείο `app/views/books/search.html.erb`, θα εκτελεστεί ο ενσωματωμένος κώδικας ruby και το αποτέλεσμα του (html κώδικας) θα προωθηθεί στον φυλλομετρητή του χρήστη. Με τις προεπιλεγμένες ρυθμίσεις μια εφαρμογή Rails μπορεί να επιστρέψει κώδικα html, xml, json ή javascript ως απόκριση σε μία αίτηση.

2.7.4 Φίλτρα (filters)

Τα φίλτρα είναι μέθοδοι που εκτελούνται πριν, μετά ή ενδιάμεσα της εκτέλεσης μιας ενέργειας του ελεγκτή. Δηλώνονται στην αρχή της κλάσης με την χρήση των παρακάτω εντολών.

```
before_filter   # Μεθοδοι που εκτελούνται πριν από οποιαδήποτε ενέργεια
after_filter    # Μεθοδοι που εκτελούνται μετά από οποιαδήποτε ενέργεια
around_filter   # Μεθοδοι που εκτελούνται κατά την διάρκεια της ενέργειας
before_filter :require_login
# Δηλώνουμε μια μέθοδο που αυθεντικοποιεί τον χρήστη (require_login)
# πριν αυτός αποκτήσει πρόσβαση στις ενέργειες του ελεγκτή
```

2.8 Action View

Το Action View συμπληρώνει την τριπλέτα του MVC αφού αποτελεί την υλοποίηση της Προβολής (στο πρότυπο MVC) στο Rails. Είναι υπεύθυνο με την εμφάνιση και παρουσίαση των δεδομένων στον χρήστη. Οι προβολές (views) περιλαμβάνουν ως επί το πλείστον κώδικα html μέσα στον οποίον υπάρχει εμφωλευμένος κώδικας σε ruby. Η ενσωμάτωση του κώδικα ruby γίνεται με την χρήση των συμβόλων <%= … %> ή <% … %> . Όταν χρησιμοποιείται ο 1ος συμβολισμός εκτυπώνεται το αποτέλεσμα του κώδικα, ενώ με τον 2ο δεν εκτυπώνεται.

2.8.1 Καθολικές προβολές

Οι περισσότερες ιστοσελίδες στο διαδίκτυο έχουν μια συγκεκριμένη δομή και εμφάνιση που διατηρείται κατά μήκος των υποσελίδων τους. Συνήθως στο πάνω μέρος (header) έχουν ένα λογότυπο ή/και το μενού περιήγησης στην ιστοσελίδα, ένα κυρίως μέρος που εμφανίζεται το περιεχόμενο της κάθε υποσελίδας και στο κάτω μέρος κάποιο σταθερό περιεχόμενο (footer). Οι καθολικές προβολές (layouts) είναι html αρχεία που αναπαριστούν την γενική μορφοποίηση της ιστοσελίδας, μέσα στις οποίες εμφωλεύονται οι επιμέρους προβολές των ελεγκτών. Τοποθετούνται στον φάκελο `app/views/layouts`, ενώ η προεπιλεγμένη καθολική προβολή βρίσκεται στην διαδρομή `app/views/layouts/application.html.erb`. Επίσης κάθε ελεγκτής μπορεί να χρησιμοποιήσει διαφορετική καθολική προβολή. Στην πιο απλή της μορφή μια καθολική προβολή θα έχει την εξής μορφή.

```
<html>
<head>
<title>Τίτλος σελίδας</title>
</head>
<body>
      <%= yield %>                    # κώδικας ruby
</body>
</html>
```

Με την χρήση της εντολής `yield` προσδιορίζουμε το μέρος στο οποίο θα ενσωματωθούν οι προβολές (views).

2.8.2 Μερικές προβολές

Όπως αναφέρθηκε παραπάνω μια ιστοσελίδα μπορεί να αποτελείται από μέρη που περιέχουν σταθερό περιεχόμενο όπως το πάνω μέρος (header) της σελίδας. Με την χρήση των μερικών προβολών (partial templates) μπορούμε να χωρίσουμε την ιστοσελίδα μας σε μικρότερα και ευκολότερα διαχειρίσιμα μέρη. Αποτελούν ένα τρόπο να μειώσουμε το μέγεθος των καθολικών προβολών και γενικά μεγάλων προβολών, καθώς και να συγκεντρώσουμε σε ένα αρχείο, κώδικα που χρησιμοποιείται από πολλές διαφορετικές προβολές. Το όνομα μιας μερικής προβολής πρέπει να ξεκινάει με κάτω παύλα '_'. Στην ιστοσελίδα του βιβλιοπωλείου, στο δεξί υπομενού που περιέχει εμφανίζεται μια λίστα με τις κατηγορίες των βιβλίων. Το υπομενού αυτό δημιουργείται από μία μερική προβολή τα περιεχόμενα της οποίας παρουσιάζονται παρακάτω.

```
#Όνομα αρχείου: app/views/comp/_categories.erb
<p> <a>Κατηγορίες</a> </p><br/>          #html
<ul>                                     #html
<% for cat in @categories %>             #κωδικας ruby
    <li><%= link_to cat.title,  category_path(@cat) %></li>
<% end %>
</ul>


#Για να ενσωματώσουμε την μερική προβολή μέσα σε άλλη προβολή
#χρησιμοποιούμε την εντολή render
<%= render :partial => 'comp/categories' %>
```

2.9 Action Dispatch

Το Action Dispatch είναι το συστατικό που είναι υπεύθυνο με την δρομολόγηση των αιτήσεων του χρήστη στους κατάλληλους ελεγκτές. Αυτό επιτυγχάνεται με την κατάλληλη αποκωδικοποίηση των διευθύνσεων (url). Το σύστημα δρομολόγησης εξετάζει την διεύθυνση (url) της αίτησης και αποφασίσει ποια ενέργεια θα γίνει από την μεριά της εφαρμογής. Το σύστημα δρομολόγησης αντιστοιχεί διευθύνσεις σε ενέργειες ελεγκτών (actions).

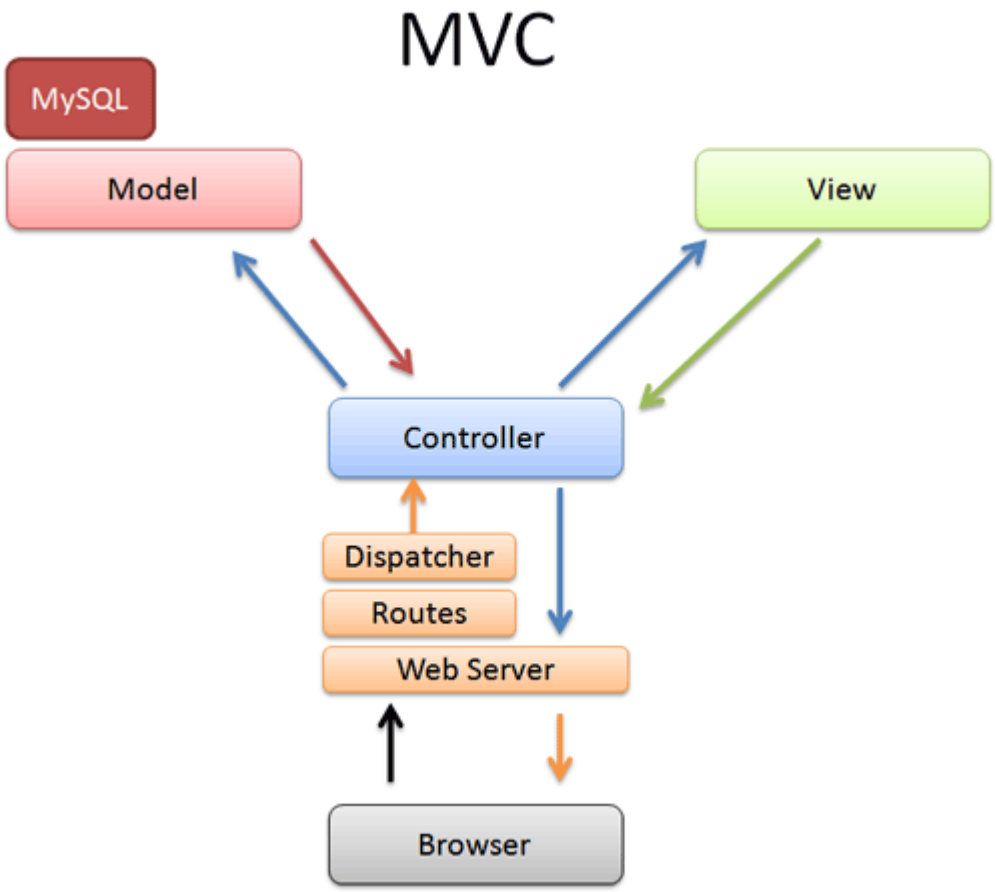
2.9.1 Routes

Το βασικότερο αρχείο του Action Dispatch είναι το αρχείο ‘routes.rb’, το οποίο βρίσκεται στην διαδρομή `config/routes.rb`. Δημιουργείται αυτόματα κατά την δημιουργία της εφαρμογής. Περιέχει όλες τις εντολές που χρειάζονται από τον δρομολογητή για να πάρει τις κατάλληλες αποφάσεις.

```
Bookshop::Application.routes.draw do
  # Η αρχική σελίδα της εφαρμογής αντιστοιχεί στην ενέργεια
  # index του ελεγκτή home. Δηλαδή θα εκτελεστεί η μέθοδος index
  # του home_controller.rb
  root :to => "home#index"
end
```

2.9.2 Λειτουργία Action Pack – Active Record

Για να γίνει καλύτερα αντιληπτό πως αλληλεπιδρούν τα συστατικά του Rails μεταξύ τους και με τον χρήστη θα αναλύσουμε μια περίπτωση χρήσης ([2]).



Σχήμα 7 "Διάγραμμα λειτουργίας Action Pack – Active Record"

1. Ο φυλλομετρητής (browser) στέλνει μια αίτηση, την διεύθυνση http://bookshop.com/categories/show_cat/12 .

2. Ο διακομιστής (apache, webrick κτλ.) λαμβάνει την αίτηση. Το Action Dispatch χρησιμοποιεί το αρχείο 'routes.rb' για να βρει σε ποιο ελεγκτή θα προωθήσει την αίτηση. Η προεπιλεγμένη ρύθμιση είναι /controller/action/id, στην περίπτωση μας ο ελεγκτής '**categories**', η ενέργεια '**show_cat**' και id '**12**'. Ο διακομιστής τότε δημιουργεί ένα αντικείμενο του ελεγκτή categories (της κλάσης categories_controller).

3. Το αντικείμενο του ελεγκτή εκτελεί την μέθοδο '**show_cat**' και ρωτάει το μοντέλο για τα δεδομένα. Ζητάει τα βιβλία που ανήκουν στην κατηγορία με κωδικό **12**.

4. Το μοντέλο αλληλεπιδρά με την βάση δεδομένων και επιστρέφει τα δεδομένα που του ζητήθηκαν στον ελεγκτή.

5. Ο ελεγκτής περνάει τα δεδομένα από το μοντέλο στην κατάλληλη προβολή, την views/categories/**show_cat**.html.erb. Δημιουργείται ο html κώδικας από την προβολή.

6. Ο ελεγκτής επιστρέφει την απάντηση (html κώδικας) στον φυλλομετρητή του χρήστη.

2.10 Action Mailer

Το Action Mailer είναι το συστατικό του Rails που μας επιτρέπει να στέλνουμε και να λαμβάνουμε ηλεκτρονικά μηνύματα (email) από και προς την εφαρμογή μας. Τα ηλεκτρονικά μηνύματα είναι μια πολύ συχνά χρησιμοποιούμενη διαδικασία για την ασύγχρονη επικοινωνία μεταξύ της εφαρμογής και των χρηστών της. Είναι ζωτικής σημασίας για τις σύγχρονες διαδικτυακές εφαρμογές, που χρησιμοποιούνται για την επιβεβαίωση εγγραφής σε μια ιστοσελίδα, για ανάκτηση χαμένων κωδικών και για αποστολή ενημερωτικών μηνυμάτων (newsletter).

2.11 Scaffolding

Το Rails περιέχει ένα εργαλείο με το οποίο μπορούμε πολύ γρήγορα να δημιουργήσουμε τον βασικό κορμό της εφαρμογής μας. Χρησιμοποιείται μέσω της κονσόλας (είτε ms-dos είτε ένα τερματικό στα unix λειτουργικά) εκτελώντας κάποιες απλές εντολές. Με την χρήση του μπορούμε να δημιουργήσουμε εύκολα και αστραπιαία ελεγκτές, προβολές και μοντέλα για τις οντότητες της εφαρμογής μας. Για παράδειγμα για την οντότητα Book της εφαρμογής μας αν εκτελέσουμε την εντολή `rails generate scaffold Book title:string description:text` θα δημιουργηθούν αυτόματα τα αρχεία που φαίνονται στον πίνακα 6.

Πίνακας 6 "Αρχεία που δημιουργούνται με την χρήση scaffold"

Αρχείο	Σκοπός
db/migrate/20100209617127_create_books.rb	Αρχείο migration για την δημιουργία του πίνακα books στην βάση
app/models/book.rb	Το μοντέλο για την οντότητα Book
test/fixtures/books.yml	Πλασματικές καταχωρήσεις βιβλίων για την χρήση κατά το τεσταρισμα
app/controllers/books_controller.rb	Ο ελεγκτής για την οντότητα Book
app/views/books/index.html.erb	Μια προβολή που εμφανίζει ένα κατάλογο όλων των βιβλίων
app/views/books/edit.html.erb	Μια προβολή για την επεξεργασία των στοιχείων ενός βιβλίου
app/views/books/show.html.erb	Μια προβολή που εμφανίζει τα στοιχεία ενός βιβλίου
app/views/books/new.html.erb	Μια προβολή για να δημιουργήσουμε ένα καινούριο βιβλίο
app/views/books/_form.html.erb	Μερική προβολή που περιλαμβάνει μια φόρμα για την δημιουργία και την ενημέρωση βιβλίων
app/helpers/books_helper.rb	Βοηθητικές μέθοδοι που μπορούν να χρησιμοποιηθούν από τις προβολές
test/unit/book_test.rb	Τεστ για το μοντέλο Book
test/functional/books_controller_test.rb	Τεστ για τον ελεγκτή του Book
test/unit/helpers/books_helper_test.rb	Τεστ για τις βοηθητικές μεθόδους του Book
config/routes.rb	Θα τροποποιηθεί για να περιλαμβάνει εντολές δρομολόγησης για την οντότητα Book
public/stylesheets/scaffold.css	Αρχείο στυλ css για την παρουσίαση των δημιουργημένων προβολών

2.12 Υποστήριξη κοινότητας & εργαλεία ανάπτυξης

2.12.1 Τεκμηρίωση & API

Η κοινότητα του Ruby on Rails έχει μεγαλώσει στο πέρασμα των χρόνων. Αυτό έχει σαν αποτέλεσμα την ύπαρξη ώριμης και συνεχώς ανανεώσιμης τεκμηρίωσης πάνω στο πλαίσιο. Η τεκμηρίωση του πλαισίου είναι αρκετά πλήρης με σαφή παραδείγματα. Η επίσημη τεκμηρίωση του πλαισίου βρίσκεται στην διεύθυνση http://api.rubyonrails.org/ και αναφέρεται στην πιο πρόσφατη έκδοση του πλαισίου. Επίσης μπορούμε να βρούμε μια διαφορετική παρουσίαση της τεκμηρίωσης στην διεύθυνση http://railsapi.com/doc/rails-vXXX/, ανάλογα την έκδοση του πλαισίου που χρησιμοποιούμε. Εκτός όμως από την τεκμηρίωση για το API του πλαισίου, υπάρχουν και αναλυτικοί οδηγοί που μας καθοδηγούν πάνω σε συγκεκριμένα θέματα και ενέργειες που επιχειρούμε. Αυτοί οι οδηγοί είναι προσπελάσιμοι στην διεύθυνση http://guides.rubyonrails.org/ οι οποίοι ανανεώνονται και εμπλουτίζονται συνεχώς.

2.12.2 Εργαλεία ανάπτυξης

Καθώς η δημοτικότητα του Rails αυξάνεται με την πάροδο του χρόνου, εμφανίζονται στην αγορά εργαλεία ανάπτυξης εξειδικευμένα στο Rails. Πλέον υπάρχουν πολλά εργαλεία ανάπτυξης εφαρμογών που μας βοηθούν στην δημιουργία διαδικτυακών εφαρμογών με χρήση του Rails. Με χαρακτηριστικά όπως συμπλήρωση κώδικα, αποσφαλμάτωση, συνεργασία με συστήματα ελέγχου εκδόσεων και πολλά άλλα. Παρακάτω αναφέρονται μερικά χρήσιμα εργαλεία.

- 3rdRail
- ActiveState Komodo
- Aptana Radrails
- Eclipse
- jEdit
- Ruby In Steel
- Jetbrains RubyMine
- Netbeans
- TextMate

Επίλογος

Πραγματοποιήθηκε ανάλυση στα βασικά συστατικά του πλαισίου Rails. Αναπτύχθηκαν τα σημαντικότερα και καίρια ζητήματα κατά την ανάπτυξη μιας Ruby on Rails εφαρμογής. Επίσης παρουσιάστηκαν και αναλύθηκαν τα κυριότερα συστατικά και βιβλιοθήκες του Ruby on Rails και ο τρόπος συνεργασίας τους για την δημιουργία διαδραστικών εφαρμογών διαδικτύου. Στο κεφάλαιο που ακολουθεί επιχειρείται μια σύγκριση του Ruby on Rails με άλλα πλαίσια ανάπτυξης διαδικτυακών εφαρμογών.

3 ΣΥΓΚΡΙΣΗ ΜΕ ΚΑΘΙΕΡΩΜΕΝΑ ΠΛΑΙΣΙΑ

Σε αυτό το κεφάλαιο γίνεται μια προσπάθεια να συγκριθεί το πλαίσιο ανάπτυξης διαδικτυακών εφαρμογών Ruby on Rails με άλλα παρόμοια πλαίσια γραμμένα σε ευρέως χρησιμοποιούμενες γλώσσες προγραμματισμού για διαδικτυακό προγραμματισμό. Τα πλαίσια ανάπτυξης διαδικτυακών εφαρμογών είναι πάρα πολλά, γι αυτό θα επικεντρωθούμε στα ASP.NET, Spring, Cakephp. Μέσω της διαδικασίας αυτής θα συγκρίνουμε και την Ruby με τις αντίστοιχες γλώσσες.

3.1 Μέτρα σύγκρισης

Το διαδίκτυο έχει ενσωματωθεί στην καθημερινότητα μας, και έχει αλλάξει τον τρόπο που επικοινωνούν, ενημερώνονται και συναλλάσσονται οι άνθρωποι. Γι αυτόν τον λόγο η ανάπτυξη διαδικτυακών εφαρμογών και ιστοτόπων συνεχώς εξελίσσεται. Τα πλαίσια ανάπτυξης βοηθούν και επιταχύνουν την ανάπτυξη εφαρμογών. Υπάρχουν πάρα πολλά πλαίσια ανάπτυξης διαδικτυακών εφαρμογών. Για το συγκριτικό προσπάθησα να επιλέξω ένα από τα ευρέως χρησιμοποιούμενα από κάθε γλώσσα προγραμματισμού.

Η επιλογή ενός πλαισίου ανάπτυξης δεν είναι εύκολη διαδικασία. Ακόμη και η διαφορετική οπτική γωνία παίζει μεγάλο ρόλο. Για παράδειγμα ένας προγραμματιστής θα θελήσει να ασχοληθεί-μάθει ένα πλαίσιο ανάπτυξης το οποίο είναι ευρέως χρησιμοποιούμενο για να έχει περισσότερες πιθανότητες πρόσληψης από μία εταιρία. Μία εταιρία θα επιλέξει ένα πλαίσιο ανάπτυξης με το οποίο θα μπορεί να αναπτύξει εφαρμογές σε σύντομο χρονικό διάστημα και με το λιγότερο κόστος. Είναι πάρα πολλοί οι παράγοντες που παίζουν ρόλο σε μία τέτοια απόφαση.

Η σύγκριση μεταξύ διαφόρων πλαισίων ανάπτυξης είναι αρκετά ασαφής διαδικασία. Τα κριτήρια σύγκρισης που έχουν μεγαλύτερη σημασία σε ένα τέτοιο συγκριτικό δεν είναι καν μετρήσιμα χαρακτηριστικά. Για παράδειγμα όταν επιλέγουμε ένα πλαίσιο ανάπτυξης μας ενδιαφέρει πολύ η κοινότητα του πλαισίου. Δηλαδή πόσοι προγραμματιστές και χρήστες ασχολούνται και χρησιμοποιούν το συγκεκριμένο πλαίσιο. Αν χρειαστούμε βοήθεια θα υπάρχει κάποιο μέρος (ιστοσελίδα, φόρουμ, εταιρία) να απευθυνθούμε; Υπάρχουν αρκετοί επαγγελματίες προγραμματιστές που χρησιμοποιούν το συγκεκριμένο πλαίσιο σε περίπτωση πρόσληψης προσωπικού;

Επίπεδα σύγκρισης

- Άδεια χρήσης
- Γλώσσα προγραμματισμού
- Εκπαίδευση
- Περιβάλλον ανάπτυξης
- Τάσεις αγοράς
- Τεκμηρίωση & Κοινότητα
- Υλοποίηση

Για το συγκεκριμένο συγκριτικό έχουν επιλεχτεί το Cakephp που είναι υλοποιημένο σε php, το ASP.NET, το Spring που είναι υλοποιημένο σε Java και φυσικά το Ruby on Rails. Παρακάτω στον πίνακα φαίνονται τα κύρια χαρακτηριστικά του κάθε πλαισίου ([31]).

Πίνακας 7 "Χαρακτηριστικά του κάθε πλαισίου"

Πλαίσιο	Ajax	MVC πρότυπο	i18n & l10n	ORM	Testing framework	DB migration framework(s	Security Framework(s)
Cakephp	Prototype, script.aculo.us, jQuery, MooTools	ναι	ναι	Data Mapper	Unit Test, Fixtures, Code Coverage	ναι	ACL-based
Ruby on Rails	Prototype, script.aculo.us, jQuery	ActiveRecord ActionPack	ναι	ActiveRecord	Unit Test, Functional Tests and Integration tests	ναι	Plug-in
Spring	ναι	ναι	ναι	Hibernate, iBatis	Unit Test	—	Spring Security
Asp.NET	ναι	Asp.NET MVC	—	—	Unit Test	—	ASP.NET Forms Auth

3.2 Εκπαίδευση

Χρόνος εκμάθησης

Το Ruby on Rails και το Cakephp απαιτούν λιγότερο χρόνο εκμάθησης του πλαισίου από ότι του spring και του asp.net. Το asp.net κυρίως και το spring χρησιμοποιείται σε εταιρικά περιβάλλοντα καθώς έχουν περισσότερες δυνατότητες από τα ruby on rails και cakephp. Η εκμαθησή τους απαιτεί περισσότερη προσπάθεια και χρόνο λόγω του ογκώδους API τους.

Βιβλιογραφία

Ένα αντικειμενικό κριτήριο για να υπολογιστεί η βιβλιογραφία που είναι διαθέσιμη για το κάθε πλαίσιο ανάπτυξης θα ήταν η καταμέτρηση των βιβλίων που υπάρχουν διαθέσιμα σε ευρέως χρησιμοποιούμενα βιβλιοπωλεία. Στην παρούσα μέτρηση συμμετέχουν τα βιβλιοπωλεία Amazon, Barnes&Noble και το γνωστό ελληνικό βιβλιοπωλείο Παπασωτηρίου. Χρησιμοποιήθηκαν οι όροι αναζήτησης 'ruby on rails', 'asp.net', 'cakephp', 'spring framework'. Παρακάτω στον πίνακα 9 παρουσιάζονται τα αποτελέσματα για το κάθε πλαίσιο στο καθένα βιβλιοπωλείο ξεχωριστά.

Πίνακας 8 "Πλήθος βιβλίων σε γνωστά βιβλιοπωλεία για το κάθε πλαίσιο ανάπτυξης"

	Asp.net	Cakephp	Ruby on Rails	Spring
Amazon	1122	23	160	70
Barnes&Noble	417	11	63	28
Παπασωτηρίου	428	2	62	26

Προέλευση: Amazon.com, Barnesandnoble.com, Papasotiriou.gr

Συμπερασματικά βλέπουμε ότι το asp.net είναι πολύ ώριμο πλαίσιο ανάπτυξης με εκτενή βιβλιογραφία. Για το Ruby on Rails παρατηρούμε ότι υπάρχει επαρκής βιβλιογραφία στην αγορά για την εκπαίδευση των προγραμματιστών. Στα αποτελέσματα παίζει ρόλο ότι υπάρχουν πολλά πλαίσια ανάπτυξης για τις γλώσσες java και php σε αντίθεση με την ruby που υπάρχουν άλλα 4 και με τις γλώσσες προγραμματισμού της Microsoft που χρησιμοποιείται κυρίως το asp.net. Για την java και την php η βιβλιογραφία έχει επεκταθεί στα πολλά πλαίσια ανάπτυξης που υπάρχουν γι αυτές. Για την ruby η βιβλιογραφία έχει επικεντρωθεί στο κυρίαρχο πλαίσιο της, το Ruby on Rails.

3.3 Γλώσσα προγραμματισμού

Το κάθε πλαίσιο χρησιμοποιεί συγκεκριμένη γλώσσα προγραμματισμού. Το Ruby on Rails είναι γραμμένο σε ruby, το Cakephp σε php, το Spring σε Java και με την πλατφόρμα ASP.NET μπορούν να χρησιμοποιηθούν διάφορες γλώσσες όπως η C#, C++, VB κτλ. Στον παρακάτω πίνακα είναι συγκεντρωμένα τα κύρια χαρακτηριστικά της κάθε γλώσσας που χρησιμοποιείται από το αντίστοιχο πλαίσιο ανάπτυξης.

Πίνακας 9 "Χαρακτηριστικά γλωσσών προγραμματισμού"

Ruby	PHP	Java	C#
Δυναμικοί τύποι μεταβλητών	Δυναμικοί τύποι μεταβλητών	Στατικοί τύποι μεταβλητών	Στατικοί τύποι μεταβλητών
Διερμηνευόμενη	Διερμηνευόμενη	(Ψευδό)Μεταγλωττιζόμενη	Μεταγλωττιζόμενη
Αντικειμενοστραφής	Πολλοί τύποι προγραμματισμού	Αντικειμενοστραφής	Πολλοί τύποι προγραμματισμού
Περιεκτική σύνταξη	Δυσανάγνωστος κώδικας	Φλύαρη γλώσσα	Ακολουθεί την Java και C++
Ευανάγνωστος κώδικας	Πλήθος βιβλιοθηκών	Πλήθος βιβλιοθηκών	Πλήθος βιβλιοθηκών
Μεταφερσιμότητα	Μεταφερσιμότητα	Μεταφερσιμότητα	-

3.4 Άδεια χρήσης

Το Ruby on Rails διανέμεται με άδεια χρήσης MIT. Αυτό σημαίνει ότι οποιοσδήποτε μπορεί να κατεβάσει τον κώδικα του, να τον χρησιμοποιήσει, να τον τροποποιήσει, να τον δημοσιεύσει ακόμα και να τον πουλήσει χωρίς καμία υποχρέωση προς τους δημιουργούς του. Η άδεια χρήσης MIT αποτελεί μια τελείως ανοικτή άδεια, που επιτρέπει πλήρη ελευθερία με το λογισμικό που διανέμεται υπό αυτήν. Και το πλαίσιο ανάπτυξης cakephp διανέμεται με άδεια χρήσης MIT, που σημαίνει ότι ισχύουν τα ίδια με το ruby on rails.

Το πλαίσιο ανάπτυξης Spring και οι βιβλιοθήκες του είναι πλήρως ανοικτού κώδικα λογισμικά. Διανέμονται με άδεια χρήσης Apache, που είναι παρόμοια με την άδεια χρήσης MIT και επιτρέπουν πλήρη ελευθερία με το λογισμικό που εκδίδεται υπό αυτήν.

Το πλαίσιο ανάπτυξης ASP.NET είναι κλειστού κώδικα αλλά ελεύθερα διαθέσιμο ιδιωτικό λογισμικό. Δεν μπορεί να τροποποιηθεί ή να διανεμηθεί κάτω από οποιεσδήποτε συνθήκες. Προορίζεται να χρησιμοποιηθεί ως είναι για την ανάπτυξη διαδικτυακών εφαρμογών στην πλατφόρμα των Windows.

Το Ruby on Rails, το Cakephp και το Spring αποτελούν πλήρως ανοικτά λογισμικά που μπορούν να χρησιμοποιηθούν σε πολλές πλατφόρμες (όπως Windows, UNIX κτλ.) και με πολλούς διακομιστές http. Αποτελούν και τα 3 προσπάθειες της κοινότητας του ανοικτού λογισμικού. Αντιθέτως το ASP.NET είναι κλειστό λογισμικό σχεδιασμένο να χρησιμοποιείται μόνο σε πλατφόρμα Windows και κυρίως με τον διακομιστή http IIS της Microsoft, αν και υπάρχουν υλοποιήσεις για να τρέχει και σε άλλους διακομιστές http (πχ. Apache) και σε άλλα λειτουργικά συστήματα.

Η χρήση ανοικτού λογισμικού είναι πολύ πιο συμφέρουσα οικονομικά για μια εταιρία, γιατί δεν χρειάζεται να δαπανηθούν οικονομικοί πόροι στις άδειες χρήσης λογισμικού όπως λειτουργικά συστήματα, διακομιστές http, συστήματα διαχείρισης βάσεων δεδομένων και περιβάλλοντα ανάπτυξης λογισμικού.

3.5 Τεκμηρίωση και Κοινότητα

Αυτό το χαρακτηριστικό αναφέρεται στην κοινότητα και στην βοήθεια που υπάρχει διαθέσιμη για κάθε πλαίσιο ανάπτυξης. Στο πόσο εύκολο είναι να βρεθούν απαντήσεις στις ερωτήσεις και απορίες ενός προγραμματιστή. Ένα πλαίσιο ανάπτυξης που υποστηρίζεται από πολύ

Το Ruby on Rails πρωτοδημοσιεύτηκε το 2004. Από τότε έχουν περάσει πολλά χρόνια και η δημοτικότητα του έχει αυξηθεί κατά πολύ. Με αποτέλεσμα να υπάρχουν πολλές πηγές βοήθειας για τους χρήστες-προγραμματιστές του πλαισίου. Η επίσημη τεκμηρίωση του πλαισίου είναι πλήρης και αναλυτική. Υπάρχουν επίσης αναλυτικοί οδηγοί για την χρήση του κάθε συστατικού (gem).

Το ASP.NET χρησιμοποιείται αρκετά συχνά από επιχειρήσεις που δραστηριοποιούνται στο διαδίκτυο. Υπάρχουν πολλοί επαγγελματίες που χρησιμοποιούν το ASP.NET για την ανάπτυξη διαδικτυακών εφαρμογών. Η Microsoft γενικά προσέφερε συνήθως περιεκτική τεκμηρίωση για το λογισμικό της. Το ίδιο ισχύει και στην περίπτωση του ASP.NET. Παρέχεται περιεκτικότατη τεκμηρίωση με πολλά παραδείγματα και επεξηγήσεις. Η περιήγηση σε αυτό ίσως παρουσιάσει δυσκολία γιατί το ASP.NET είναι μεγάλο πλαίσιο που περιλαμβάνει αρκετές βιβλιοθήκες.

3.6 Περιβάλλον ανάπτυξης

Το περιβάλλον ανάπτυξης για μια γλώσσα προγραμματισμού και ένα πλαίσιο ανάπτυξης παίζουν σημαντικό ρόλο κατά την ανάπτυξη εφαρμογών. Παρέχουν ευκολίες, αυτοματοποιούν εργασίες και γενικά επιταχύνουν την ανάπτυξη. Τρία κυρίαρχα εργαλεία που χρησιμοποιούνται κατά κόρον είναι το Eclipse, το Netbeans και το Visual Studio. Επίσης υπάρχουν εργαλεία εξειδικευμένα σε μία γλώσσα προγραμματισμού.

Για το Ruby on Rails υπάρχουν μερικά ώριμα περιβάλλοντα ανάπτυξης στην αγορά, τα οποία αναφέρθηκαν στο προηγούμενο κεφάλαιο. Υπάρχει πλήρης υποστήριξη για το πλαίσιο από τα εργαλεία Netbeans και Eclipse, που μπορούν να αντικαταστήσουν σε ένα βαθμό και την κονσόλα (πχ για την δημιουργία μιας εφαρμογής, δημιουργία ελεγκτή, μοντέλου). Για μια ολοκληρωμένη ανάπτυξη μιας εφαρμογής Ruby on Rails μπορούν να χρησιμοποιηθούν μόνο η κονσόλα, το περιβάλλον ανάπτυξης και ένας φυλλομετρητής.

Για το Cakephp δεν υπάρχει κάποιο περιβάλλον ανάπτυξης που να το υποστηρίζει πλήρως, αλλά υπάρχουν πολλά καλά εργαλεία σχεδιασμένα για την php. Το Netbeans και το Eclipse υποστηρίζουν πλήρως την php. Όπως και στο Ruby on Rails η χρήση της κονσόλας είναι απαραίτητη για την ανάπτυξης μιας εφαρμογής με την χρήση cakephp.

Για το Spring framework υπάρχει πλήρης υποστήριξη από το Netbeans και το Eclipse. Εκτός από αυτά όμως υπάρχουν πολλά χρήσιμα και ώριμα περιβάλλοντα ανάπτυξης για την java για όλες τις ανάγκες των προγραμματιστών. Πολλά από αυτά είναι προϊόντα ανοικτού λογισμικού και δεν απαιτείται κάποιο χρηματικό ποσό για την χρήση τους.

Για το ASP.NET υπάρχει το γνωστό Visual Studio της Microsoft, το οποίο είναι ένα πλήρες περιβάλλον ανάπτυξης. Είναι το κύριο εργαλείο ανάπτυξης για το πλαίσιο ASP.NET. Πρόκειται για ένα ώριμο περιβάλλον ανάπτυξης που ενσωματώνει όλες τις σύγχρονες τεχνολογίες. Σε αντίθεση όμως με τα Netbeans και Eclipse που είναι δωρεάν εργαλεία της κοινότητας του ανοικτού λογισμικού, το Visual Studio είναι εργαλείο που πρέπει να αγορασθεί με κάποιο χρηματικό αντίτιμο (υπάρχει ωστόσο δωρεάν έκδοση με περιορισμένες δυνατότητες).

3.8 Υλοποίηση

Η υλοποίηση μιας php εφαρμογής είναι πολύ εύκολη υπόθεση με την δημοτικότητα που έχει αποκτήσει. Οι περισσότεροι πάροχοι υπηρεσιών φιλοξενίας έχουν πακέτα με την χρήση apache-php και συνήθως είναι τα πιο οικονομικά σε συνδυασμό με την χρήση μιας διανομής linux.

Αντιθέτως με την php, η υλοποίηση μιας Rails εφαρμογής ή μιας Spring εφαρμογής αποτελεί δυσκολότερο έργο. Το κυρίως πρόβλημα έγκειται στην χρήση ενός διακομιστή εφαρμογών (application server) που θα εκτελεί τον κώδικα Ruby ή Java. Η υλοποίηση μιας εφαρμογής απαιτεί περισσότερη έρευνα σε σχέση με την php για να λειτουργήσει η εφαρμογής απροβλημάτιστα (ειδικότερα σε συνθήκες υψηλού φόρτου).

Η υλοποίηση μιας asp.net εφαρμογής συνήθως γίνεται με την χρήση του IIS διακομιστή http σε περιβάλλον windows. Το κόστος για τις άδειες χρήσης του λογισμικού που απαιτείται για την υλοποίηση μιας εφαρμογής asp.net είναι συνήθως απαγορευτικό για μικρές επιχειρήσεις και ιδιώτες. Το κόστος που απαιτείται αφορά τις άδειες χρήσης του λειτουργικού συστήματος, του διακομιστή http και ίσως του συστήματος βάσης δεδομένων.

Η υλοποίηση μιας asp.net εφαρμογής είναι η πιο δαπανηρή σε σχέση με τις υπόλοιπες τεχνολογίες και πλαίσια ανάπτυξης. Αυτό συνήθως αποτελεί μια αιτία που η χρήση του αποφεύγεται και οι προγραμματιστές στρέφονται σε λύσεις ανοικτού λογισμικού. Το Ruby on Rails όταν παρουσιάστηκε, παρουσίαζε προβλήματα στην υλοποίηση σε περιβάλλον παραγωγής. Με την πάροδο των χρόνων και την εμφάνιση εξειδικευμένων εργαλείων (όπως το Phusion Passenger) η υλοποίηση του έχει απλοποιηθεί σε μεγάλο βαθμό. Ίσως όχι στο βαθμό υλοποίησης μια php εφαρμογής, αλλά έχει απλοποιηθεί αρκετά.

3.9 Τάσεις αγοράς

Σε αυτό το σημείο θα παρουσιαστούν στατιστικά στοιχεία από γνωστές ιστοσελίδες που δείχνουν την απήχηση του κάθε πλαισίου στην αγορά.

Ιστοσελίδες ευρέσεως αγγελιών εργασίας

Ένα κριτήριο για να αξιολογηθεί η απήχηση του κάθε πλαισίου στην αγορά είναι να ερευνήσουμε το πλήθος των αγγελιών εργασίας που υπάρχουν σε γνωστές ιστοσελίδες ευρέσεως αγγελιών εργασίας σχετικές με το κάθε πλαίσιο. Γι αυτόν τον σκοπό θα χρησιμοποιηθούν οι ιστοσελίδες Indeed.com, Freelancer.com, Kariera.gr και Freestuff.gr IT Jobs. Το Freelancer.com αποτελεί μια γνωστή ιστοσελίδα που την επισκέπτονται εργοδότες που ψάχνουν επαγγελματίες της πληροφορικής (κυρίως προγραμματιστές) για να αναθέσουν τα έργα τους. Το Kariera.gr αποτελεί μια ελληνική ιστοσελίδα που περιέχει αγγελίες εργασίας, όπως και το Freestuff.gr IT Jobs. Στο πλαίσιο αυτό πραγματοποιήθηκαν αναζητήσεις στις ιστοσελίδες αυτές για τους όρους “ruby on rails”, “java spring”, “cakephp”, “ASP.NET” και τα αποτελέσματα που ανέκυψαν παρουσιάζονται στον πίνακα παρακάτω.

Πίνακας 10 "Αποτελέσματα αναζητήσεων σε μηχανές ευρέσεως εργασίας"

	Asp.net	Cakephp	Ruby on Rails	Spring
Freelancer.com	65	11	19	11
Freestuff.gr IT jobs	16	3	4	1
Kariera.gr	3	1	3	7

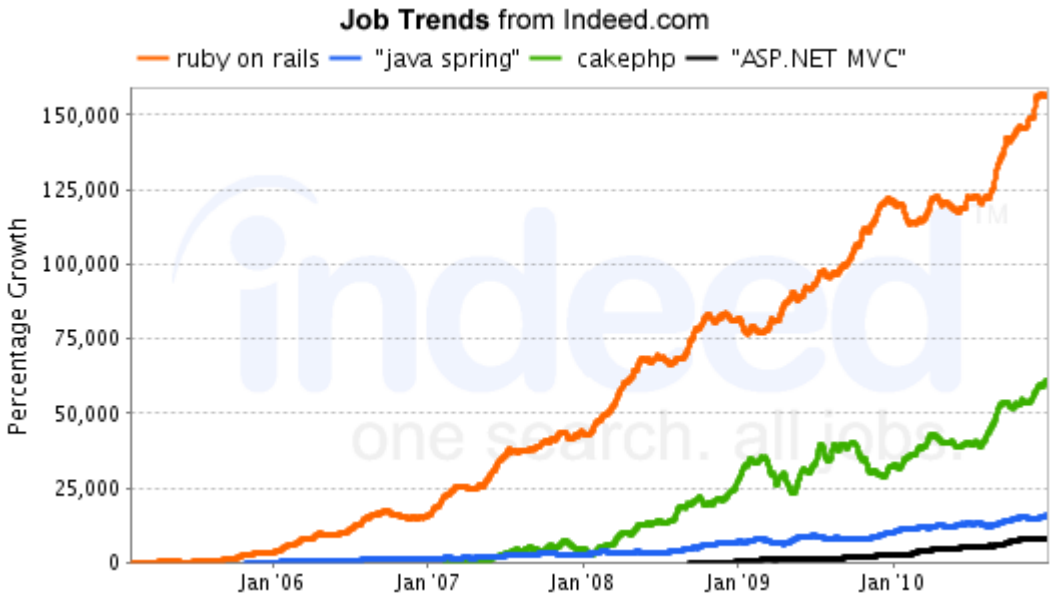
Προέλευση: Amazon.com, Freelancer.com, Jobs.freestuff.gr, τα αποτελέσματα ανακτήθηκαν στις 8/4/2011

Από τα αποτελέσματα που λάβαμε φαίνεται ξεκάθαρα ότι το asp.net έχει μεγαλύτερη απήχηση στην αγορά απ ότι τα άλλα πλαίσια. Στον ελληνικό χώρο οι διαθέσιμες εργασίες είναι λίγες, ωστόσο φαίνεται ότι υπάρχει μικρό ενδιαφέρον για το Ruby on Rails.

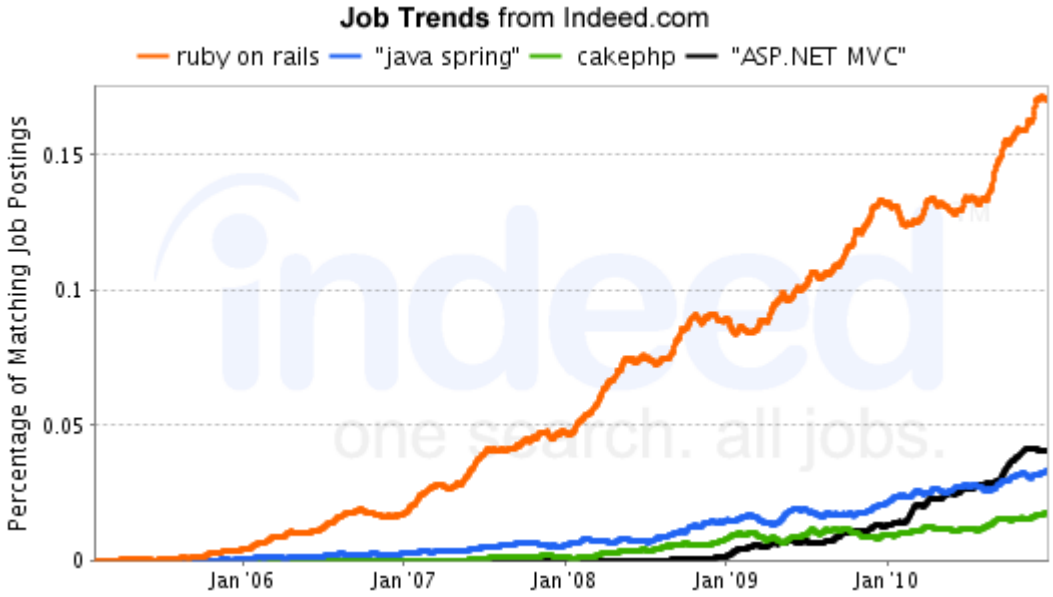
Indeed.com

Ο ιστοτόπος indeed.com ([7]) είναι μια γνωστή μηχανή αναζήτησης αγγελιών εργασίας (κατατάσσεται στην θέση 408 της παγκόσμιας κατάταξης επισκεψιμότητας σύμφωνα με το Alexa.com). Περιλαμβάνει αγγελίες από εκατοντάδες πηγές όπως ιστοτόπους ευρέσεως εργασίας και εφημερίδες. Μας παρέχει μέσω της ιστοσελίδας της ένα χρήσιμο εργαλείο για δούμε τις τάσεις της αγοράς.

Παρακάτω στο γράφημα βλέπουμε τα αποτελέσματα για τους όρους αναζήτησης “ruby on rails”, “java spring”, “cakephp” “ASP.NET MVC”.



Σχήμα 8 "Ποσοστό αύξησης"



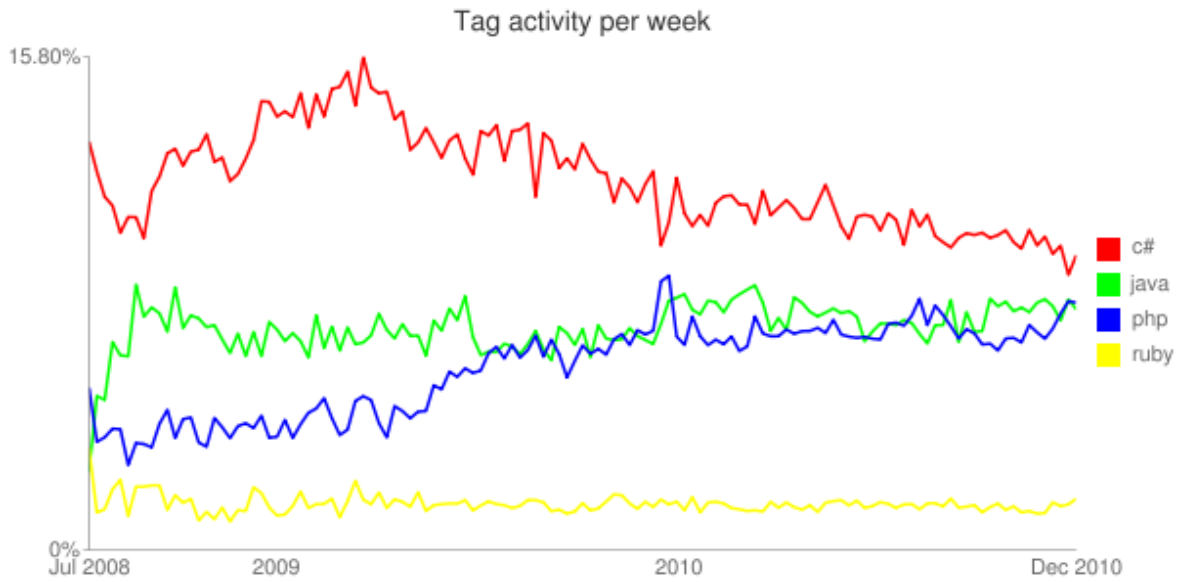
Σχήμα 9 "Πλήθος δουλειών"

Παρατηρούμε ότι υπάρχει αισθητά μεγαλύτερη (και αυξανόμενη) ζήτηση προγραμματιστών στην πλατφόρμα “ruby on rails” σε σχέση με τα άλλα πλαίσια ανάπτυξης.

StackOverflow.com

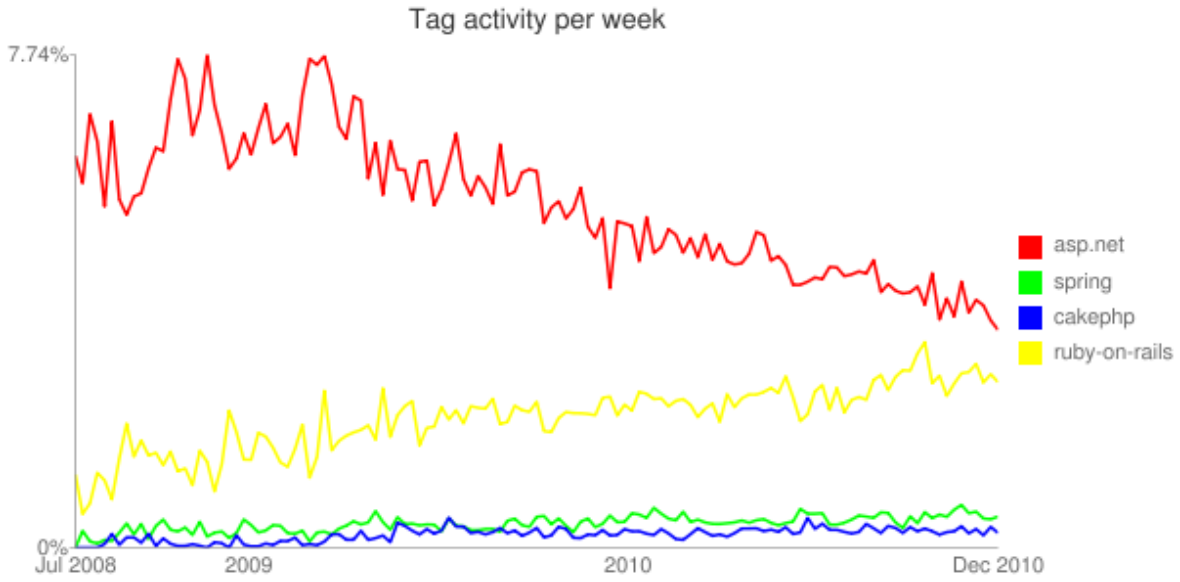
Το StackOverflow ([26]) είναι μια κοινότητα ερωτήσεων-απαντήσεων σχετικές με θέματα προγραμματισμού. Οποιοσδήποτε χρήστης μπορεί να δημοσιεύσει την ερώτηση του και να λάβει απάντηση από άλλους χρήστες. Χρησιμοποιείται αρκετά από προγραμματιστές για εξειδικευμένες ερωτήσεις πάνω σε συγκεκριμένες τεχνολογίες και γλώσσες προγραμματισμού.

Παρακάτω στο γράφημα βλέπουμε τα αποτελέσματα για τους όρους αναζήτησης “c# ”, “java”, “php” “ruby”, τις γλώσσες προγραμματισμού που χρησιμοποιούνται από τα πλαίσια ανάπτυξης που συγκρίνουμε.



Σχήμα 10 "Ποσοστό ερωτημάτων ανα γλώσσα προγραμματισμού"

Παρατηρούμε ότι υπάρχουν σαφώς περισσότερες ερωτήσεις για τις γλώσσες c#, java και php, κάτι που είναι φυσιολογικό βέβαια καθώς οι c#, java και php είναι πολύ πιο ευρέως χρησιμοποιούμενες γλώσσες προγραμματισμού από την ruby. Για να βγάλουμε κάποια πιο ασφαλή συμπεράσματα θα πραγματοποιήσουμε μια σύγκριση με όρους τα ονόματα των πλαισίων που συγκρίνουμε.



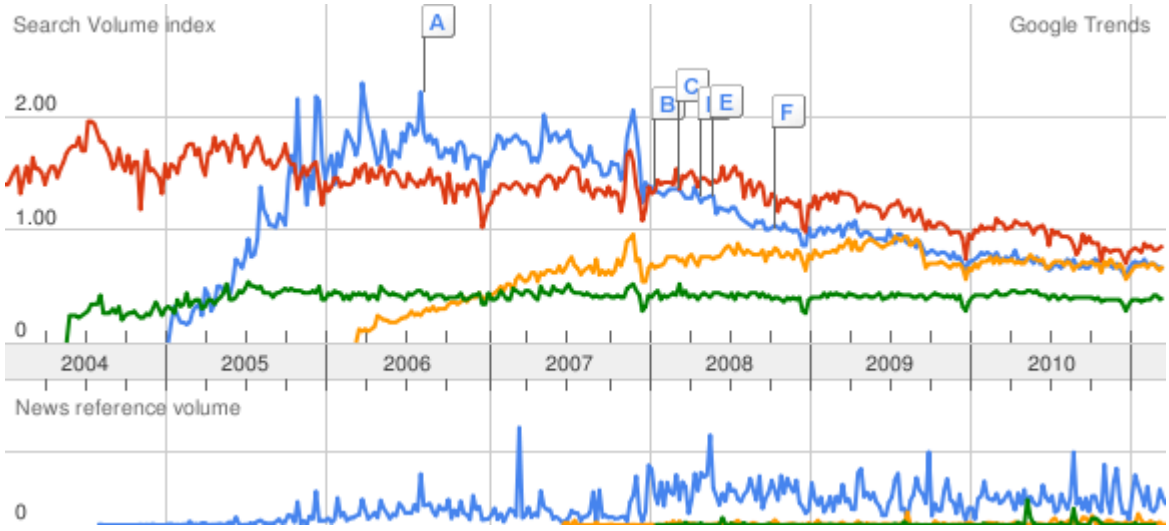
Σχήμα 11 "Ποσοστό ερωτημάτων ανα πλαίσιο ανάπτυξης"

Παρατηρούμε ότι υπάρχει μια αυξανόμενη τάση ερωτήσεων σχετικά με το ruby on rails από ότι σε σχέση με τα spring και cakephp. Αυτό μπορεί να σημαίνει ότι η χρήση

του ruby on rails παρουσιάζει προβλήματα στους προγραμματιστές, και γι αυτό δημοσιεύουν ερωτήσεις σχετικές με το ruby on rails. Αλλά πιστεύω μπορεί να θεωρηθεί ότι το πλαίσιο ruby on rails χρησιμοποιείται από περισσότερους προγραμματιστές από ότι τα spring και cakephp.

Google Trends

Το Google Trends ([6]) είναι μια ιστοσελίδα της Google, που μας δείχνει πόσο συχνά χρησιμοποιείται ένας όρος αναζήτησης σε σχέση με το συνολικό πλήθος αναζητήσεων στην μηχανή αναζήτησης της Google. Επίσης μας επιτρέπει να συγκρίνουμε το πλήθος αναζητήσεων ανάμεσα σε δύο ή περισσότερους όρους αναζήτησης. Αν χρησιμοποιηθούν εύστοχοι όροι μπορεί να μας δείξει το ενδιαφέρον των χρηστών του διαδικτύου πάνω σε ένα συγκεκριμένο θέμα.



Σχήμα 12 "Google trends αναζητήσεις για κάθε όρο"

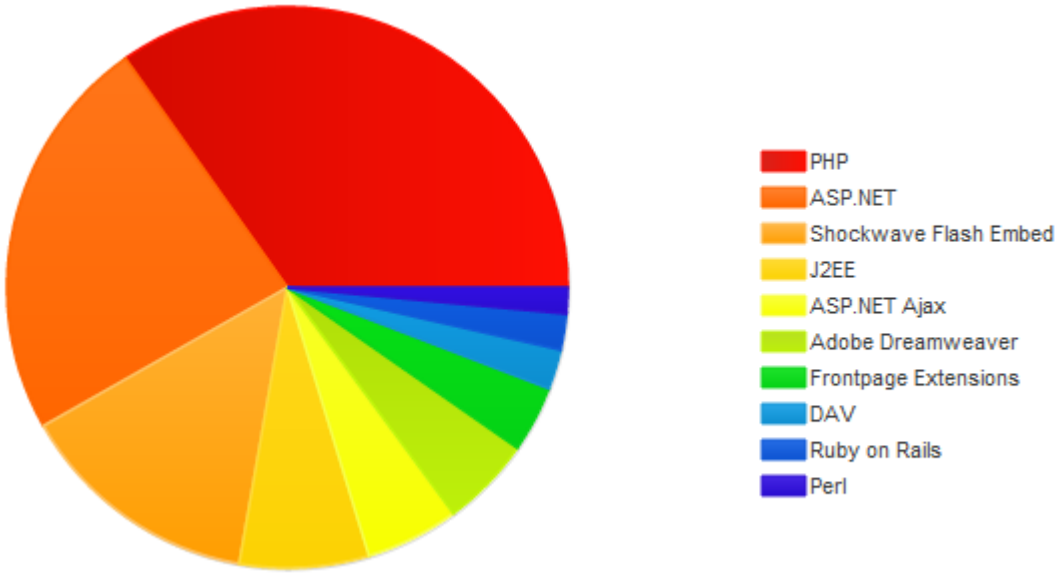
Όροι αναζήτησης και δείκτες αναλογίας

ruby on rails	1.00
asp.net c#	1.32
cakephp	0.46
spring java	0.38

Συγκρίνοντας τους όρους “ruby on rails”, “asp.net c#”, “cakephp”, “spring java” δημιουργείται το παραπάνω γράφημα. Οι αναζητήσεις για κάθε όρο παρουσιάζονται ως ποσοστό αναλογίας σε σχέση με το “ruby on rails” (συντελεστής αναλογίας 1.00). Παρατηρούμε ότι υπήρχε μια έξαρση στο αρχικό διάστημα που παρουσιάστηκε το Rails, το 2004. Συνεχώς ελαττώνεται μέχρι το 2010 που τείνει να σταθεροποιηθεί. Ωστόσο δεν μπορούν να εξαχθούν ασφαλή συμπεράσματα, αλλά είναι ένα δείγμα για το ενδιαφέρον των ανθρώπων για το κάθε πλαίσιο ανάπτυξης.

Builtwith.com

Το Builtwith.com ([3]) είναι μία ιστοσελίδα που αναλύει τις τεχνολογίες (γλώσσα προγραμματισμού, πλαίσιο ανάπτυξης, διακομιστές, βιβλιοθήκες javascript, συστήματα διαχείρισης περιεχομένου κτλ.) που χρησιμοποιούν οι δημοφιλέστερες ιστοσελίδες στο διαδίκτυο. Η ανάλυση που διεξάγει σε μια ιστοσελίδα μπορεί να μην αποδεικνύεται πάντα έγκυρη, καθώς μερικές ιστοσελίδες προσπαθούν να αποκρύψουν τις τεχνολογίες που χρησιμοποιούν για λόγους ασφαλείας.



Σχήμα 13 "Ποσοστό χρήσης τεχνολογιών σε δημοφιλείς ιστοσελίδες" πηγή: Builtwith.com

Στο σχήμα 13 φαίνεται το ποσοστό χρήσης κάθε τεχνολογίας (γλώσσα προγραμματισμού ή πλαίσιο ανάπτυξης) στο πλήθος των 10.000 δημοφιλέστερων ιστοσελίδων στο διαδίκτυο σύμφωνα με στοιχεία της ιστοσελίδας Builtwith.com. Τα στοιχεία που εμφανίζονται υπολογιστήκαν την 8η Μαρτίου 2011. Στον πίνακα 12 φαίνεται αριθμητικά το ποσοστό της εκάστοτε τεχνολογίας.

Πίνακας 11 "Ποσοστό χρήσης τεχνολογιών σε δημοφιλείς ιστοσελίδες"

Τεχνολογία	Ποσοστό χρήσης
PHP	31.95%
ASP.NET	21.49%
Shockwave Flash Embed	12.9%
J2EE	6.84%
ASP.NET Ajax	4.95%
Adobe Dreamweaver	4.79%
Frontpage Extensions	3.51%
DAV	2.11%
Ruby on Rails	1.9%
Perl	1.37%
Adobe ColdFusion	1.33%
Java Servlet	1.09%
Visual Studio	0.81%
ASP.NET MVC	0.64%
Microsoft Frontpage	0.61%
Ruby on Rails Token	0.59%
Python	0.51%

3.10 Υλοποιήσεις της Ruby

Στο τέλος, αν και δεν αποτελεί κριτήριο σύγκρισης, θα ήθελα να αναφέρω τις διαφορετικές υλοποιήσεις της Ruby πάνω σε δοκιμασμένα συστήματα ανάπτυξης εφαρμογών. Εκτός από την επίσημη έκδοση της γλώσσας Ruby αναπτύσσονται παράλληλα από διαφορετικούς φορείς και άλλες υλοποιήσεις της. Οι πιο γνώστες υλοποιήσεις της είναι οι MacRuby, JRuby, IronRuby και Rubinius. Οι διαφορετικές υλοποιήσεις που αναπτύσσονται δείχνουν την επίδραση της Ruby στις άλλες πλατφόρμες ανάπτυξης εφαρμογών. Μέσα από αυτές έχει γίνει προσπάθεια να χρησιμοποιηθεί η δύναμη της Ruby σε συνδυασμό με την εκάστοτε πλατφόρμα που αναπτύσσεται.

Η IronRuby ([8]) είναι η υλοποίηση της Ruby για το πλαίσιο .NET της Microsoft. Αναπτύσσεται από την ίδια την Microsoft και είναι υλοποιημένη στην γλώσσα C#. Επίσης είναι λογισμικό ανοικτού κώδικα. Με την χρήση της μπορεί να συνδυαστεί η δύναμη της Ruby, η δύναμη του πλαισίου .NET και του εργαλείου Visual Studio.

Η JRuby ([9]) είναι η υλοποίηση της Ruby για την εικονική μηχανή της Java (JVM). Είναι υλοποιημένη πλήρως σε Java και αποτελεί λογισμικό ανοικτού κώδικα. Με την χρήση της μπορεί να συνδυαστεί το πλήθος των βιβλιοθηκών και η δύναμη της Java, με την ταχύτητα του JVM και την δύναμη της Ruby. Έχει ωριμάσει αρκετά ως λογισμικό καθώς αναπτύσσεται συνεχώς από το 2001.

Οι εναλλακτικές υλοποιήσεις της Ruby με διαφορετικούς διερμηνείς που ενσωματώνονται σε διαφορετικές πλατφόρμες ανάπτυξης εφαρμογών δείχνουν την επίδραση της Ruby στις κοινότητες των προγραμματιστών που χρησιμοποιούν τις αντίστοιχες πλατφόρμες. Η φιλοσοφία της Ruby, η απλότητα αλλά συνάμα η πολυπλοκότητα της είναι χαρακτηριστικά που επιθυμούν οι προγραμματιστές σε μία γλώσσα προγραμματισμού. Συμπερασματικά μπορούμε να πούμε ότι μπορεί να χρησιμοποιηθεί η Ruby σε ήδη γνωστές και δοκιμασμένες λύσεις ανάπτυξης εφαρμογών, όπως είναι η Java και το πλαίσιο .NET.

3.11 Χρήση σε ιστοσελίδες μεγάλης επισκεψιμότητας

Στην κοινότητα των προγραμματιστών υπάρχει η παραδοχή ότι το Ruby on Rails δεν μπορεί να χρησιμοποιηθεί σε ιστοσελίδες με μεγάλη επισκεψιμότητα καθώς δεν αποδίδει καλά όπως άλλες enterprise λύσεις όπως το ASP.NET και η Java EE.

Ο πιο γνωστός ιστότοπος που χρησιμοποιείται το Ruby on Rails είναι η κοινωνική πλατφόρμα μικρο-blogging Twitter. Το Rails χρησιμοποιούνταν από την έναρξη του Twitter, αν και κάποια κομμάτια της εφαρμογής έχουν μεταφερθεί σε άλλες γλώσσες προγραμματισμού, ωστόσο το γραφικό περιβάλλον του υποστηρίζεται από το Rails.

Όσον αφορά τις ελληνικές ιστοσελίδες ο πιο γνωστός ιστότοπος που είναι βασισμένος σε Ruby on Rails, είναι η γνωστή μηχανή σύγκρισης τιμών το skroutz.gr. Οι δημιουργοί του χρησιμοποίησαν το Rails από τις πρώτες αρχικές εκδόσεις του. Με 110.000 επισκέπτες καθημερινά ίσως είναι δείγμα ότι το Ruby on Rails μπορεί να χρησιμοποιηθεί από ιστοσελίδες με μεγάλη επισκεψιμότητα.

3.12 Συμπεράσματα

Το μόνο σίγουρο συμπέρασμα που μπορούμε να εξάγουμε είναι ότι κανένα πλαίσιο ανάπτυξης δεν είναι τέλειο. Το καθένα έχει τα πλεονεκτήματα και τα μειονεκτήματα του. Ή αν παραφράσουμε την πρώτη πρόταση, το μόνο σίγουρο συμπέρασμα που μπορούμε να εξάγουμε είναι ότι κανέναν πλαίσιο ανάπτυξης δεν είναι τέλειο για όλες τις χρήσεις. Κάθε πλαίσιο έχει την χρήση του η οποία εξαρτάται σε κάποιο βαθμό από το μέγεθος και την πολυπλοκότητα της εφαρμογής. Υπάρχει η ‘αντίληψη’ στην κοινότητα των προγραμματιστών ότι για πιο σοβαρές εφαρμογές ή αλλιώς enterprise-level όπως συχνά αναφέρεται θα πρέπει να χρησιμοποιούνται λύσεις ανεπτυγμένες σε ASP.NET ή σε Java EE.

Η επιλογή ενός πλαισίου ανάπτυξης ή ακόμη και το αν θα χρησιμοποιηθεί κάποιο πλαίσιο ανάπτυξης επηρεάζεται από πάρα πολλούς λόγους. Αυτό που φαίνεται από την τωρινή κατάσταση του διαδικτύου είναι ότι κυριαρχούν σαν τεχνολογίες η php και τα δεκάδες πλαίσια της και συστήματα διαχείρισης περιεχομένου (CMS), το asp.net και λιγότερο η Java.

Πλεονεκτήματα του Ruby On Rails

Συμπερασματικά τα πλεονεκτήματα του πλαισίου είναι:

- ✓ Ταχύτητα ανάπτυξης
- ✓ Διατήρηση κώδικα
- ✓ Ακολουθεί τα πρότυπα και τις τάσεις της αγοράς
- ✓ Δύναμη της Ruby
- ✓ Σύμβαση-έναντι-Ρύθμισης (COC)

Μειονεκτήματα του Ruby On Rails

Συμπερασματικά τα μειονεκτήματα του πλαισίου είναι:

- ❖ Χρησιμοποιεί πολλού πόρους του συστήματος
- ❖ Προβλήματα σε εφαρμογές με πολλούς ταυτόχρονους χρήστες
- ❖ Είναι δύσκολο στην υλοποίηση (deployment)
- ❖ Περιορισμένη υποστήριξη από τους παρόχους φιλοξενίας

Επίλογος

Σε αυτό το κεφάλαιο επιχειρήθηκε η σύγκριση του πλαισίου Ruby on Rails σε σχέση με τα καθιερωμένα πλαίσια ανάπτυξης και τις καθιερωμένες γλώσσες προγραμματισμού που χρησιμοποιούνται για την ανάπτυξη διαδικτυακών εφαρμογών. Ένα πολύ μεγάλο μέρος του διαδικτύου βασίζεται στις php, java και asp.net και γι αυτό επιχειρήθηκε σύγκριση με τις συγκεκριμένες τεχνολογίες. Και χρησιμοποιώ την λέξη ‘επιχειρήθηκε’ γιατί δεν μπορεί να εξαχθεί ασφαλές συμπέρασμα (θα ήταν λάθος στην προσπάθεια και μόνο) για το ποιο είναι το καλύτερο πλαίσιο ανάπτυξης ή ποια είναι η καλύτερη γλώσσα προγραμματισμού. Το καθένα και η καθεμία έχει τις χρήσεις του.

4 ΕΓΚΑΤΑΣΤΑΣΗ ΛΟΓΙΣΜΙΚΟΥ

Σε αυτό το κεφάλαιο αναλύεται το λογισμικό που απαιτείται για να λειτουργήσει μια εφαρμογή ruby on rails. Παρουσιάζεται βήμα βήμα η εγκατάσταση των απαραίτητων διακομιστών και συστημάτων που αποτελούν την υποδομή για την υλοποίηση μιας εφαρμογής. Η διαδικασία αυτή παρουσιάζεται τόσο σε περιβάλλον Windows όσο και σε περιβάλλον Linux.

4.1 Απαιτούμενο λογισμικό

Σε αυτό το κεφάλαιο θα παρουσιάσουμε την εγκατάσταση του απαιτούμενου λογισμικού. Μια εφαρμογή Ruby on Rails για να λειτουργήσει χρειάζεται το παρακάτω λογισμικό:

1. Ruby (διερμηνέας της γλώσσας)
2. Rubygems (διαχειριστής πακέτων-βιβλιοθηκών της ruby)
3. Ruby on Rails
4. Διακομιστής http (ή/και διακομιστής εφαρμογών)
5. Σύστημα διαχείρισης βάσεων δεδομένων

Θα παρουσιαστούν δύο σενάρια εγκατάστασης του απαραίτητου λογισμικού. Το πρώτο αφορά εγκατάσταση σε περιβάλλον Windows που θα χρησιμοποιηθεί στην φάση ανάπτυξης της εφαρμογής και το δεύτερο σε περιβάλλον Linux που θα χρησιμοποιηθεί στην φάση μεταφοράς της εφαρμογής σε περιβάλλον παραγωγής.

Ruby

Η ruby έχει αποκτήσει αρκετή δυναμικότητα και σαν γλώσσα ανοικτού κώδικα που είναι έχει μεταφερθεί σε πολλές πλατφόρμες και αρχιτεκτονικές των ηλεκτρονικών υπολογιστών. Τα προγράμματα γραμμένα σε ruby μπορούν να ‘τρέξουν’ σε διάφορες αρχιτεκτονικές όπως :

- Microsoft Windows 95, 98, XP, Vista και 7
- Mac OS X (όλες οι εκδόσεις)
- Linux (όλες οι μεγάλες διανομές)
- MS-DOS
- BSDs (περιλαμβάνοντας το FreeBSD και το OpenBSD)
- BeOS
- Acorn RISC OS
- OS/2
- Amiga
- Symbian Series 60 (κινητά τηλέφωνα)
- Οποιαδήποτε πλατφόρμα για την οποία υπάρχει Εικονική Μηχανή της Java (JVM), χρησιμοποιώντας την JRuby

Rubygems

Το Rubygems είναι ο διαχειριστής πακέτων της ruby. Με αυτόν εγκαθιστούμε προγράμματα και βιβλιοθήκες (τα/οι οποία/ες ονομάζονται gems) της ruby στο σύστημα μας. Είναι ένα εργαλείο που έχει δημιουργηθεί για διαχειριζόμαστε εύκολα την εγκατάσταση/απεγκατάσταση των gems.

Διακομιστής http

Όπως είδαμε στο 2ο κεφάλαιο το rails έρχεται με προεγκατεστημένο διακομιστή http, τον WEBrick. Η χρήση του όμως αφορά την φάση ανάπτυξης της εφαρμογής και δεν προορίζεται για χρήση σε περιβάλλον παραγωγής γιατί παρουσιάζει προβλήματα απόδοσης. Αποτελεί ένα ακόμη εργαλείο του rails που μας βοηθά στην φάση ανάπτυξης να μην ασχολούμαστε με την εγκατάσταση και παραμετροποίηση του διακομιστή που θα φιλοξενεί την εφαρμογή μας. Η ruby υποστηρίζεται από τους περισσότερους διακομιστές http που χρησιμοποιούνται στο διαδίκτυο.

- Apache
- Lighttpd
- Nginx
- Οποιοσδήποτε διακομιστής υποστηρίζει FastCGI

Σύστημα διαχείρισης βάσεων δεδομένων

Όπως αναφέρθηκε στο 2ο κεφάλαιο το Rails (και συγκεκριμένα το ActiveRecord) συνεργάζεται με τις περισσότερες ευρέως χρησιμοποιούμενες βάσεις δεδομένων (σελ. 48).

Διακομιστής εφαρμογών (application server)

Εκτός από τον διακομιστή http, οι εφαρμογές rails χρειάζονται και έναν διακομιστή εφαρμογών (application server) για να λειτουργήσουν. Ο διακομιστής εφαρμογών δέχεται τις αιτήσεις από τον διακομιστή http και εκτελεί τον κώδικα ruby. Ο προτεινόμενος τρόπος υλοποίησης σε περιβάλλον παραγωγής από τους δημιουργούς του Ruby on Rails είναι η χρήση του διακομιστή Apache σε συνδυασμό με το Phusion Passenger. Οι ποιο συχνά χρησιμοποιούμενοι διακομιστές εφαρμογών για την ruby και το rails είναι οι εξής:

Πίνακας 12 "Διακομιστές εφαρμογών με υποστήριξη της Ruby"

	Διακομιστής http	Λειτουργεί με	Πλατφόρμες
Mongrel	✓	Apache, lighttpd, nginx	Linux, Windows
Passenger	✗	Apache, nginx	Linux, BSD, OS X
Thin	✓	Apache, nginx	Linux, Windows
Unicorn	✓	Apache, nginx	Unix, Linux
WebROaR	✓	Αυτόνομος	Linux, OS X

4.2 Εγκατάσταση σε περιβάλλον Windows

Σε αυτό το σενάριο θα εγκατασταθεί το παρακάτω λογισμικό σε λειτουργικό Windows XP x86 SP3.

1. Ruby 1.9.2p180
2. RubyGems 1.4.2
3. Ruby on Rails 3.0.3
4. SQLite3

Σημείωση !!!

Αν χρησιμοποιήσετε την κονσόλα σε ελληνική εκδοση των Windows θα χρειαστεί να εκτελέσετε την εντολή `chcp 850` (αλλάζει την κωδικοσελίδα της κονσόλας), πριν προβείτε στην εγκατάσταση της Ruby και του Rails.

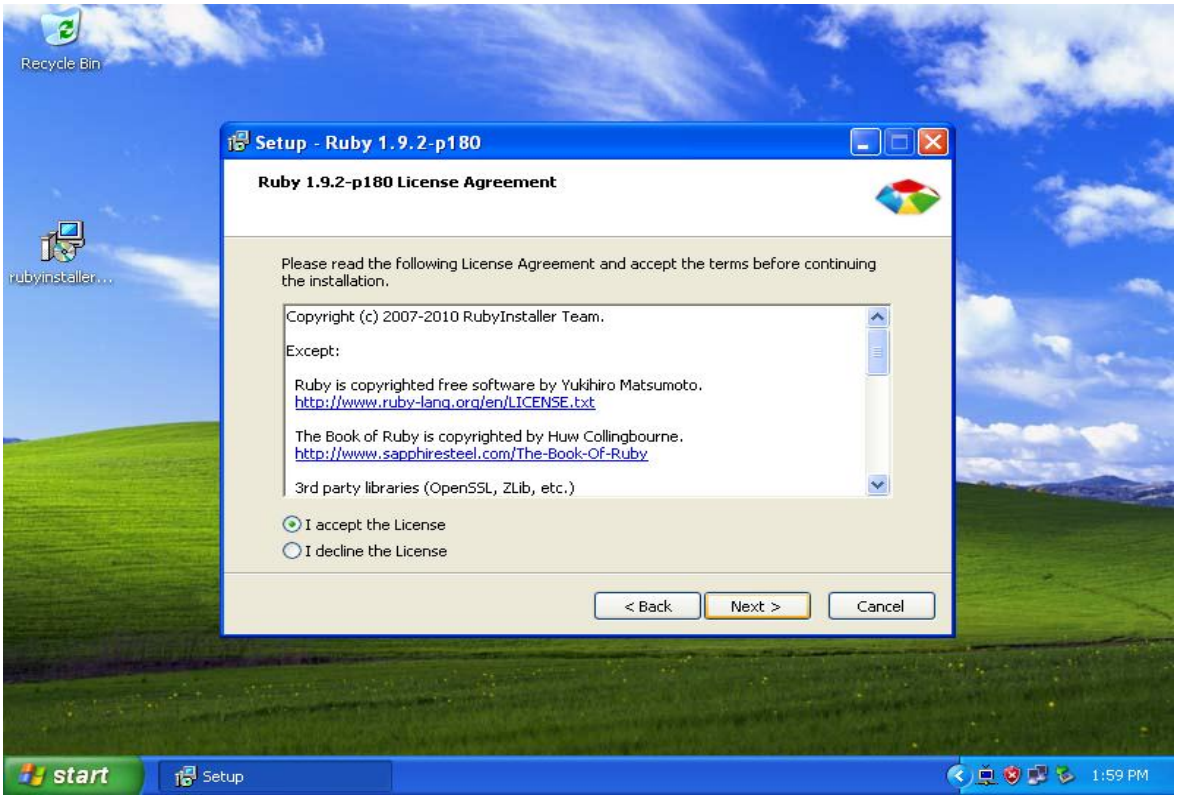
4.2.1 Εγκατάσταση Ruby

Η ruby αρχικά σχεδιάστηκε για χρήση σε περιβάλλον Unix (και τα παράγωγα του), αλλά η χρήση της έχει επεκταθεί και στην οικογένεια λειτουργικών συστημάτων Windows της Microsoft. Άλλωστε τα Windows είναι το δημοφιλέστερο λειτουργικό στους οικιακούς υπολογιστές. Η ruby εγκαθίσταται πολύ εύκολα στα Windows, αφού υπάρχει έτοιμο πρόγραμμα εγκατάστασης, το οποίο εκτός από τον διερμηνέα της ruby, εγκαθιστά χρήσιμες επεκτάσεις, την τεκμηρίωση της γλώσσας καθώς και ένα περιβάλλον εργασίας (editor). Παρακάτω φαίνεται αναλυτικά η διαδικασία εγκατάσταση :

1. Κατεβάστε από την διεύθυνση http://www.ruby-lang.org/en/downloads/ το πρόγραμμα εγκατάστασης
2. Τρέξτε το πρόγραμμα εγκατάστασης και πατήστε το ‘Next >’

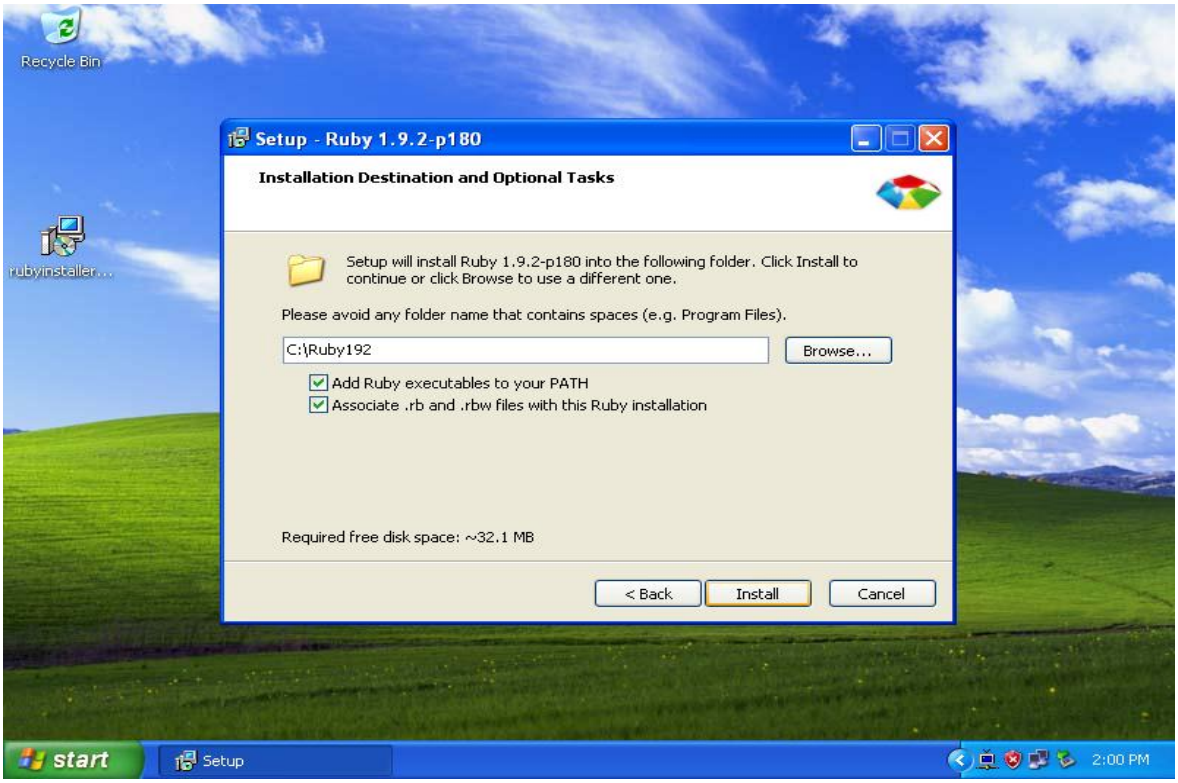
Εικόνα 1 "Εγκατάσταση Ruby σε Windows - βημα 2ο"

3. Επιλέξετε 'I accept the License' για να συμφωνήσετε με τους όρους χρήσης και πατήστε το 'Next >'



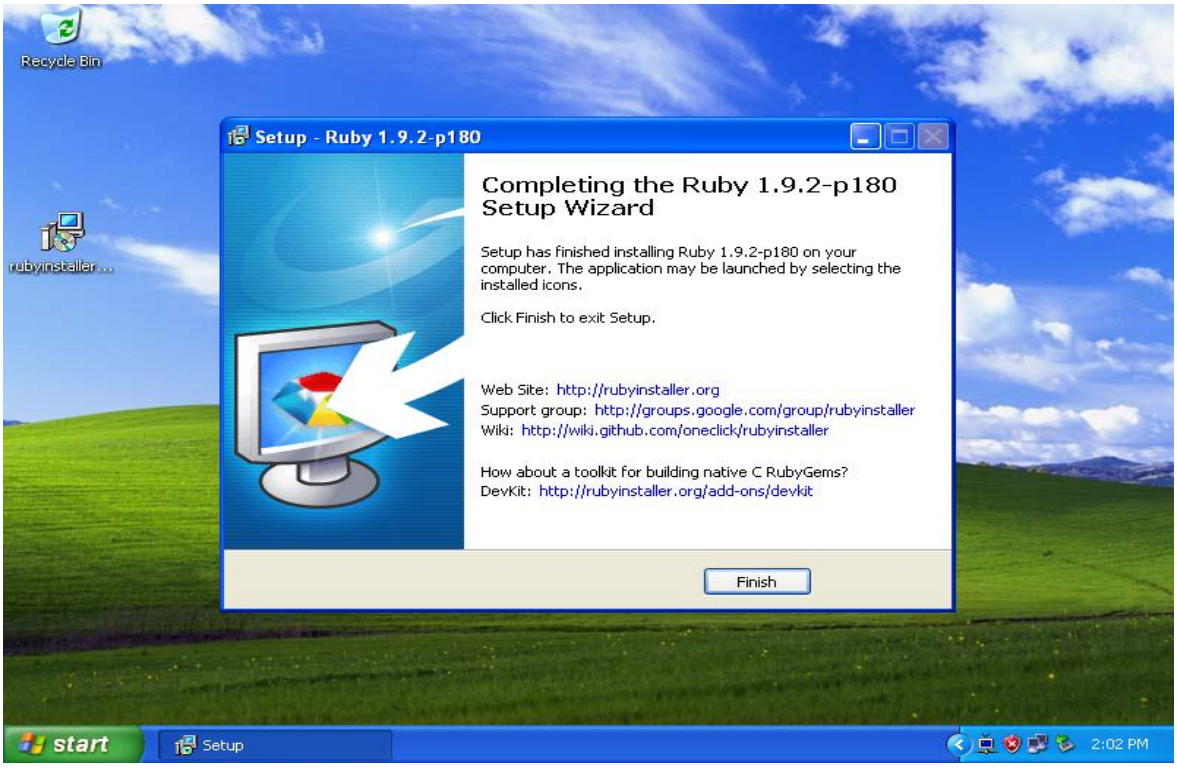
Εικόνα 2 "Εγκατάσταση Ruby σε Windows - βήμα 3ο"

4. Επιλέξτε τα δύο checkboxes και την διαδρομή που θέλετε να εγκατασταθεί η ruby και πατήστε ‘Install’



Εικόνα 3 "Εγκατάσταση Ruby σε Windows - βημα 4ο"

5. Περιμένετε μέχρι να ολοκληρωθεί η εγκατάσταση και αν είναι επιτυχής θα αντικρύσετε αυτή την οθόνη

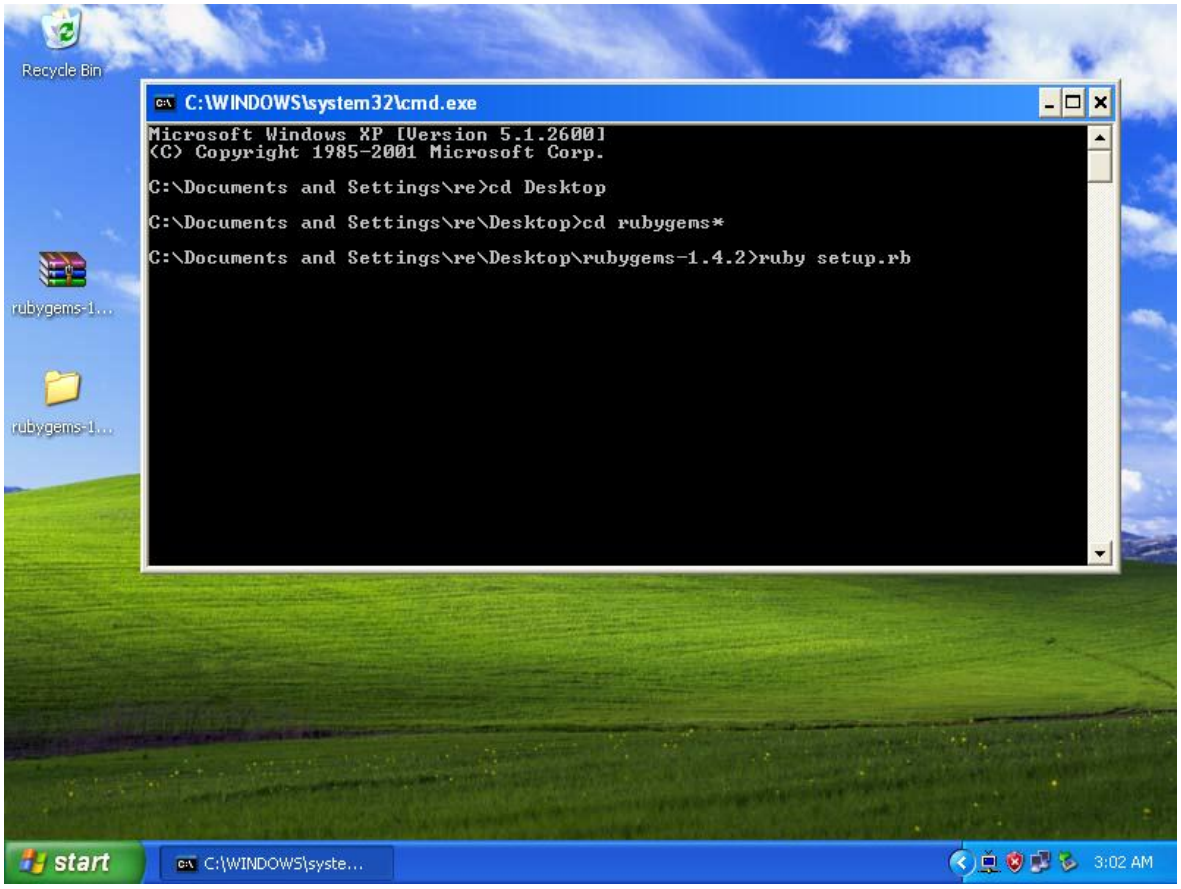


Εικόνα 4 "Εγκατάσταση Ruby σε Windows - βημα 5ο"

4.2.2 Εγκατάσταση του Rubygems

Το Rubygems είναι ο διαχειριστής πακέτων της ruby. Με αυτόν εγκαθιστούμε επεκτάσεις και πρόσθετα της ruby στο συστημά μας.

1. Κατεβάστε στον υπολογιστή σας την τελευταία έκδοση του Rubygems από την σελίδα http://rubyforge.org/frs/?group_id=126
2. Αποσυμπιέστε το συμπιεσμένο αρχείο σε κάποιον φάκελο.
3. Ανοίξτε μια κονσόλα ms-dos και περιηγηθείτε στον κατάλογο που αποσυμπιέσατε το αρχείο. Τώρα εκτελέστε την εντολή `ruby setup.rb`

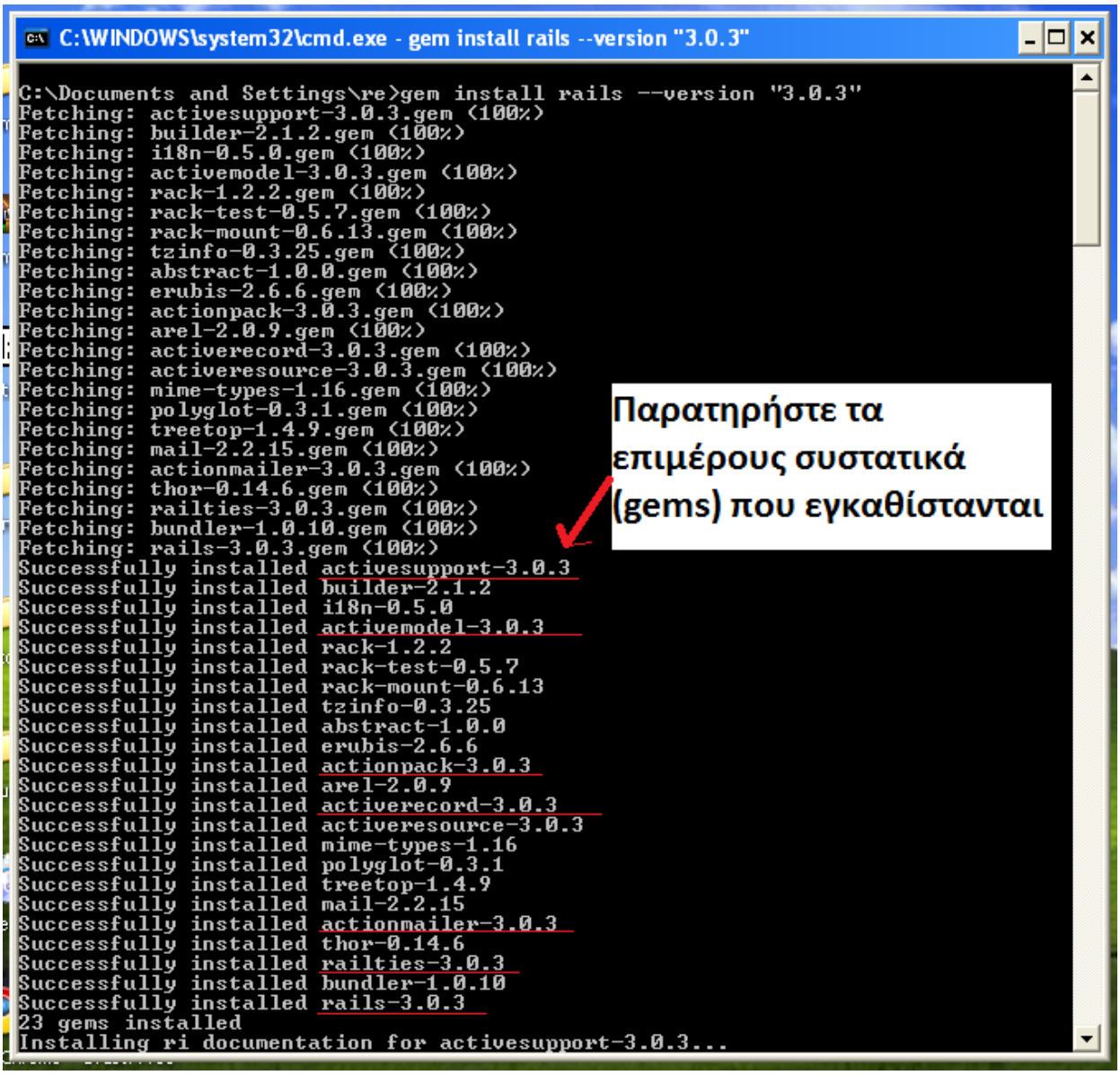


Εικόνα 5 "Εγκατάσταση Rubygems σε Windows"

4.2.3 Εγκατάσταση του Ruby on Rails

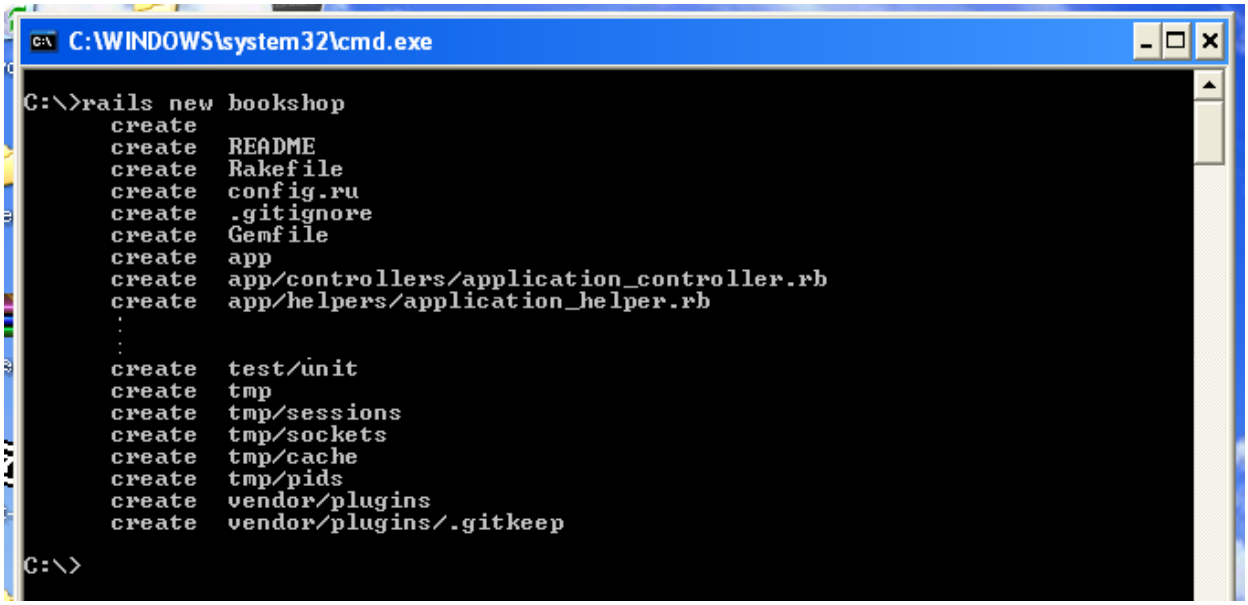
Για να εγκαταστήσετε την τελευταία έκδοση του Rails, ανοίξτε μια κονσόλα ms-dos και εκτελέστε την εντολή `gem install rails`.

Στο παρόν σενάριο θα εγκαταστήσουμε την έκδοση 3.0.3 του Rails. Ανοίξτε μια κονσόλα dos και εκτελέστε την εντολή `gem install rails –version “3.0.3”`.



Εικόνα 6 "Εγκατάσταση Ruby on Rails σε Windows"

Για να δημιουργήσετε μία καινούρια εφαρμογή rails, σε μια κονσόλα ms-dos εκτελέστε την εντολή `rails new διαδρομή/της/εφαρμογής.` Για παράδειγμα αν θέλετε να δημιουργήσετε μια εφαρμογή bookshop στον αρχικό κατάλογο του σκληρού σας δίσκου, θα χρειαστεί να εκτελέσετε την εντολή `rails new C:\bookshop`



Εικόνα 7 "Δημιουργία νέας Ruby on Rails εφαρμογής"

4.2.4 Εγκατάσταση του διακομιστή http (Web server)

Σε αυτό το σενάριο θα χρησιμοποιήσουμε τον ενσωματωμένο διακομιστή http που εγκαθίσταται μαζί με το Ruby on Rails, τον WEBrick, οπότε δεν χρειάζεται να εγκαταστήσουμε κάτι παραπάνω για την ώρα. Ο WEBrick λειτουργεί και ως διακομιστής http και ως διακομιστής εφαρμογών.

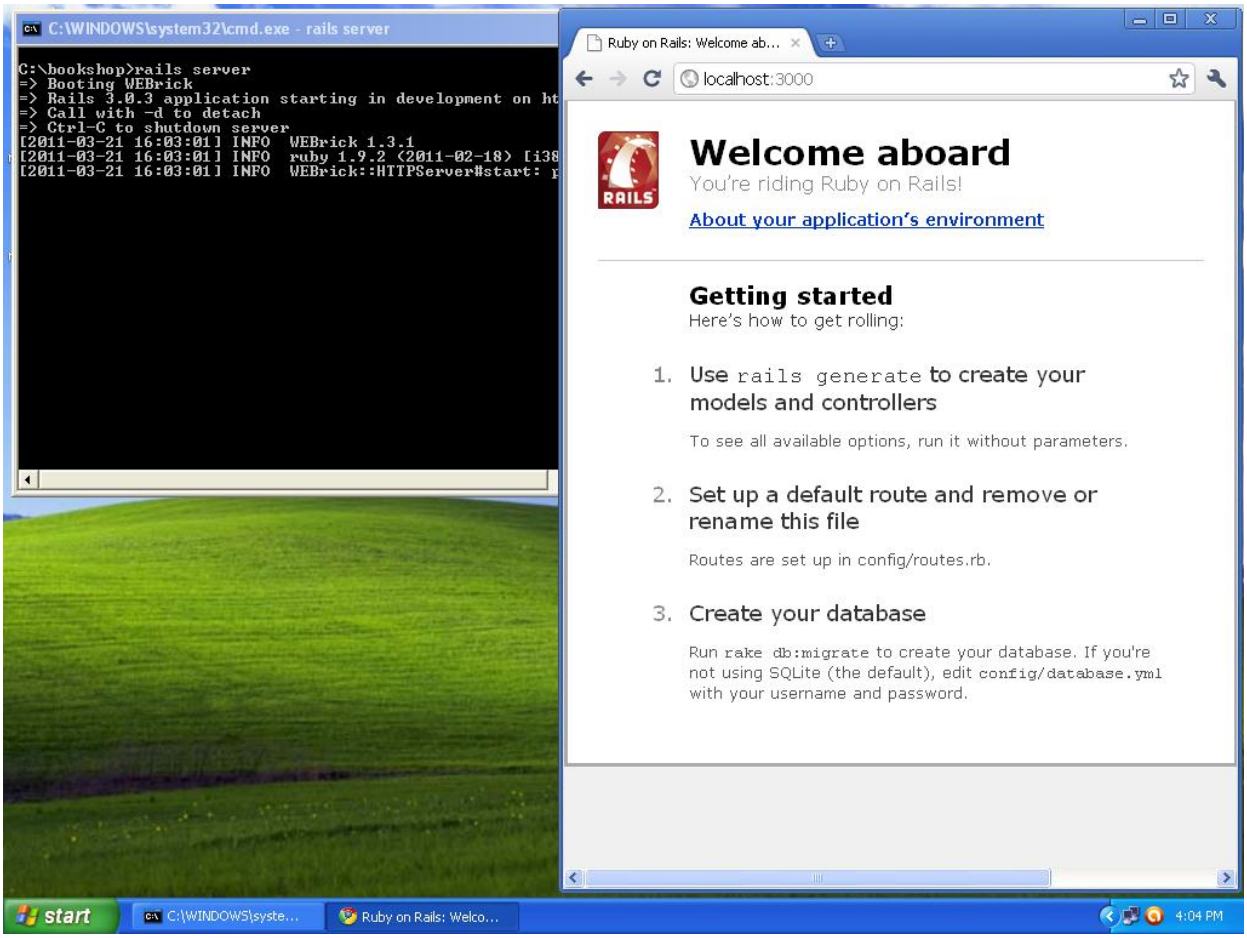
4.2.5 Εγκατάσταση της βάσης δεδομένων

Σε αυτό το σενάριο θα χρησιμοποιήσουμε την βάση δεδομένων SQLite. Για να την εγκαταστήσουμε εκτελούμε σε μια κονσόλα την εντολή `gem install sqlite3-ruby`. Για να μπορέσει να λειτουργήσει η sqlite θα πρέπει να εγκαταστήσουμε το αρχείο sqlite3.dll από την διεύθυνση http://www.sqlite.org/download.html στον φάκελο bin μέσα στο φάκελο εγκατάστασης της Ruby (πχ. C:\Ruby192\bin).

4.2.6 Εκτέλεση της εφαρμογής

Για να δούμε την εφαρμογή που έχει δημιουργηθεί μέχρι τώρα, ανοίγουμε μια κονσόλα πηγαίνουμε στον φάκελο που έχουμε δημιουργήσει την εφαρμογή και εκτελούμε την εντολή `rails server`. Με αυτή την εντολή ξεκινάμε στον διακομιστή WEBrick και πλέον η εφαρμογή μας είναι προσπελάσιμη μέσω κάποιου φυλλομετρητή στην διεύθυνση http://localhost:3000.

```
cd C:\bookshop
rails server
```



Εικόνα 8 "Εκτέλεση εφαρμογής Ruby on Rails"

4.3 Εγκατάσταση σε περιβάλλον Linux

Το προτεινόμενο λειτουργικό σύστημα για να υλοποιήσουμε μια Ruby on Rails σε περιβάλλον παραγωγής είναι οποιαδήποτε διανομή linux (Begin et al., 2008). Γι αυτό στο σενάριο αυτό θα εγκατασταθεί ένα πλήρες λειτουργικό περιβάλλον για εφαρμογές Ruby on Rails σε λειτουργικό σύστημα Ubuntu 10.10 x86. Το λογισμικό αποτελείται από τα εξής:

1. Ruby 1.9.2p0
2. RubyGems 1.6.2
3. Ruby on Rails 3.0.3
4. Apache 2.2
5. Mysql 5.1.49
6. Phusion Passenger 3.0.5

4.3.1 Εγκατάσταση Ruby

Ανάλογα την διανομή linux που χρησιμοποιούμε, υπάρχουν διάφοροι τρόποι να εγκατασταθεί η ruby. Ο πρώτος τρόπος είναι να κατεβάσουμε τον κώδικα και να τον κάνουμε ‘compile’ στο σύστημα μας. Σε μερικές διανομές όμως μπορούμε να την εγκαταστήσουμε με τον διαχειριστή πακέτων-εφαρμογών της διανομής, όπως στις διανομές Debian και Ubuntu. Για να εγκαταστήσουμε την Ruby (έκδοση 1.9.2p0) εκτελούμε την εντολή

```
$ sudo apt-get install ruby1.9.1
```

4.3.2 Εγκατάσταση του Rubygems

Για να εγκαταστήσουμε το Rubygems στο σύστημά μας, κατεβάζουμε την έκδοση που θέλουμε από την διεύθυνση http://rubyforge.org/frs/?group_id=126, αποσυμπιέζουμε το αρχείο και τρέχουμε τις κατάλληλες εντολές. Παρακάτω βλέπουμε τις κατάλληλες εντολές.

```
$ wget http://rubyforge.org/frs/download.php/74445/rubygems-
1.6.2.tgz
```

```
$ tar xzvf rubygems-1.6.2.tgz
$ cd rubygems-1.6.2
$ sudo ruby1.9.1 setup.rb
$ sudo ln –s /usr/bin/gem1.9.1 /usr/bin/gem
```

4.3.3 Εγκατάσταση του Ruby on Rails

Για να εγκαταστήσουμε την τελευταία έκδοση του Ruby on Rails χρειάζεται να τρέξουμε την εντολή `gem install rails`. Για το δικό μας σενάριο εκτελούμε την εντολή

```
$ sudo gem1.9.1 install rails –version “3.0.3”
```

Με αυτήν την εντολή εγκαθιστούμε την έκδοση 3.0.3 του Rails.

4.3.4 Εγκατάσταση του Apache 2.2

Σε αυτό το σενάριο θα χρησιμοποιήσουμε τον δημοφιλέστερο διακομιστή http του διαδικτύου, τον Apache. Για να τον εγκαταστήσουμε εκτελούμε την παρακάτω εντολή

```
$ sudo apt-get install apache2
```

Για να τρέξετε τον διακομιστή Apache εκτελέστε την εντολή

```
$ sudo /etc/init.d/apache2 start
```

4.3.3 Εγκατάσταση της MySQL

Για να εγκαταστήσουμε το σύστημα διαχείρισης βάσεων δεδομένων MySQL στο συστημά μας εκτελούμε την εντολή

```
$ sudo apt-get install mysql-server
```

4.3.3 Εγκατάσταση του Phusion Passenger

```
$ sudo gem1.9.1 install passenger
```

```
$ sudo passenger-install-apache2-module
```

Για να φορτώσει ο Apache το module του Passenger πρέπει να παραμετροποιήσουμε το αρχείο ρυθμίσεων του (apache2.conf ή httpd.conf). Στην περίπτωση μας βρίσκεται στην διαδρομή /etc/apache2/apache2.conf . Ανοίγουμε το αρχείο με έναν επεξεργαστή κειμένου και προσθέτουμε τις γραμμές :

```
#Φορτώνουμε το module του passenger
LoadModule passenger_module /usr/lib/ruby/gems/1.9.1/gems/passenger-
3.0.5/ext/apache2/mod_passenger.so


#Δηλώνουμε που βρίσκεται το gem του passenger
PassengerRoot /usr/lib/ruby/gems/1.9.1/gems/passenger-3.0.5


#Δηλώνουμε τον διερμηνέα της ruby που θα χρησιμοποιήσει το passenger
PassengerRuby /usr/bin/ruby1.9.1
```

Για να μπορέσει ο Apache να φορτώσει την εφαρμογή μας πρέπει να προσθέσουμε τις παρακάτω εντολές στο ίδιο αρχείο (apache.conf ή httpd.conf) και να επανακινήσουμε τον Apache.

```
<VirtualHost *:80>
  ServerName www.mydomain.com        # το domain της σελίδας μας
  DocumentRoot /somewhere/public     # πρέπει να δείχνει στον 'public'
  <Directory /somewhere/public>      # της εφαρμογής μας
     AllowOverride all
     Options -MultiViews
  </Directory>
</VirtualHost>
```

Αν η διεύθυνση που δηλώσουμε στο ServerName είναι ένα έγκυρο domain που αντιστοιχεί στην ip διεύθυνση του μηχανήματος που εγκαταστήσαμε τον Apache τότε η εφαρμογή μπορεί να προσπελαστεί μέσω του domain, δηλαδή έχουμε υλοποιήσει μια εφαρμογή που είναι πλήρως προσβάσιμη από το διαδίκτυο.

4.4 Εγκατάσταση με αυτοματοποιημένους εγκαταστάτες

4.4.1 Εγκατάσταση με χρήση του RailsInstaller

Αντί να εγκαταστήσουμε την ruby, το rails και την SQLite ένα ένα, μπορούμε να χρησιμοποιήσουμε το πακέτο RailsInstaller. Το RailsInstaller είναι ένα πρόγραμμα που εγκαθιστά την ruby, το rails και την SQLite στον υπολογιστή μας. Προς το παρόν διατίθεται για λειτουργικά συστήματα Windows.

4.4.2 Εγκατάσταση με χρήση του Rubystack

Εκτός από το RailsInstaller υπάρχει και το Bitnami RubyStack που ακολουθεί την ίδια φιλοσοφία. Με την χρήση του μπορούμε να εγκαταστήσουμε στο σύστημά μας τον διακομιστή Apache, την βάση δεδομένων MySQL, την Ruby, το Ruby on Rails και άλλα χρήσιμα προγράμματα, πλήρως παραμετροποιημένα για να λειτουργήσουν μεταξύ τους. Διατίθεται για Windows, Linux και Mac OS X.

Επίλογος

Σε αυτό το κεφάλαιο αναλύθηκε το λογισμικό που απαιτείται για να υλοποιηθούν Ruby on Rails εφαρμογές τόσο σε περιβάλλον Windows που χρησιμοποιήθηκε ως περιβάλλον ανάπτυξης αλλά και σε περιβάλλον Linux που χρησιμοποιήθηκε ως περιβάλλον παραγωγής. Παρουσιάστηκε ξεχωριστά η εγκατάσταση των απαραίτητων συστατικών όπως η Ruby, το Rubygems, το Ruby on Rails, οι διακομιστές http και τα συστήματα βάσεων δεδομένων.

5 ΥΛΟΠΟΙΗΣΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΕΦΑΡΜΟΓΗΣ ΒΙΒΛΙΟΠΩΛΕΙΟΥ

Σε αυτό το κεφάλαιο υλοποιείται μια διαδικτυακή εφαρμογή ηλεκτρονικού βιβλιοπωλείου με την χρήση του πλαισίου Ruby on Rails. Επίσης παρουσιάζεται συνοπτικά η διαδικασία ανάπτυξης και τα εργαλεία που χρησιμοποιούνται για μία εφαρμογής Ruby on Rails.

5.1 Υλοποίηση εφαρμογής Ruby on Rails

Για να εκτελεστεί μία εφαρμογή Ruby on Rails χρειάζεται το λογισμικό που αναφέρθηκε στον προηγούμενο κεφάλαιο, δηλαδή ο διερμηνέας της Ruby, το Ruby on Rails, ένας διακομιστής http (ή και ένας διακομιστής εφαρμογών) και ένα σύστημα βάσης δεδομένων. Για να αναπτύξουμε μια Ruby on Rails εφαρμογή χρειαζόμαστε όλα τα παραπάνω συν ένα περιβάλλον ανάπτυξης (πχ. Netbeans) και έναν φυλλομετρητή (πχ. Firefox).

5.1.1 Εργαλεία υλοποίησης

Στα πλαίσια ανάπτυξης της παρούσας πτυχιακής εργασίας υλοποιήθηκε μια διαδικτυακή εφαρμογή ηλεκτρονικού βιβλιοπωλείου γραμμένη εξ ολοκλήρου σε Ruby on Rails. Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκαν τα εξής εργαλεία :

- Ruby 1.92-p136 – windows installer
- Rubygems 1.6.0 – διαχειριστής πακέτων ruby
- Rails 3.0.3
- MySQL 5.1.43 – σύστημα διαχείρισης βάσεων δεδομένων
- WEBrick - διακομιστής http
- Jetbrains Rubymine 3.0.1 – περιβάλλον ανάπτυξης (IDE)
- MySQL Workbench 5.2.31 – περιβάλλον σχεδίασης βάσης δεδομένων για την MySQL

5.2 Τεχνικές προδιαγραφές

Στα πλαίσια της παρούσας πτυχιακής εργασίας αναπτύχθηκε ηλεκτρονικό κατάστημα βιβλιοπωλείου πάνω στο πλαίσιο ανάπτυξης Ruby on Rails. Το ηλεκτρονικό κατάστημα χωρίζεται σε δύο κύρια μέρη. Το κομμάτι των χρηστών – πελατών, το οποίο είναι η ‘βιτρίνα’ του καταστήματος και το κομμάτι των διαχειριστών του καταστήματος.

Ο πελάτης χρήστης του καταστήματος μπορεί :

- Να κάνει εγγραφή στο κατάστημα
- Να συνδέεται και να αποσυνδέεται με τον προσωπικό του λογαριασμό στο κατάστημα
- Να αλλάξει τα στοιχεία του προφίλ του

- Να κάνει σχόλια στα βιβλία
- Να προσθέσει και να αφαιρέσει βιβλία από το καλάθι αγορών
- Να επεξεργαστεί το καλάθι αγορών

Ο διαχειριστής του καταστήματος μπορεί :

- Να συνδέεται και να αποσυνδέεται από το διαχειριστικό κομμάτι του καταστήματος
- Να δημιουργεί / επεξεργάζεται / διαγράφει βιβλία / συγγραφείς / εκδότες / κατηγορίες
- Να επεξεργάζεται / διαγράφει τα σχόλια των βιβλίων
- Να επεξεργάζεται τις παραγγελίες των πελατών

5.3 Μοντέλα και πίνακες της εφαρμογής

Σε αυτό το σημείο θα παρουσιαστούν τα μοντέλα και οι πίνακες που δημιουργήθηκαν για την εφαρμογή. Επιγραμματικά οι πίνακες της βάσης αναφέρονται στον πίνακα 14.

Πίνακας 13 "Οι πίνακες της βάσης του καταστήματος"

ΠΙΝΑΚΑΣ	ΠΕΡΙΕΧΟΜΕΝΟ
administrators	Οι διαχειριστές του καταστήματος
authors	Οι συγγραφείς των βιβλίων
authors_books	Οι συσχετίσεις μεταξύ βιβλίων και συγγραφέων
books	Τα στοιχεία των βιβλίων του καταστήματος
carts	Τα καλάθια αγορών των πελατών
cart_items	Τα επιμέρους προϊόντα των καλαθιών αγορών
categories	Οι κατηγορίες που ανήκουν τα βιβλία
comments	Τα σχόλια των χρηστών για τα βιβλία
orders	Οι παραγγελίες των χρηστών
orders_items	Τα επιμέρους προϊόντα των παραγγελιών
publishers	Οι εκδότες των βιβλίων
users	Οι χρήστες-πελάτες τους καταστήματος

5.3.1 Μοντέλα

Τα μοντέλα που χρησιμοποιήθηκαν για το ηλεκτρονικό βιβλιοπωλείο είναι τα παρακάτω:

- Αντικείμενο καλαθιού (CartItem)
- Αντικείμενο παραγγελίας (OrderItem)
- Βιβλία-Συγγραφείς (AuthorsBook)
- Βιβλίο (Book)
- Διαχειριστής (Administrator)
- Εκδότης (Publisher)
- Καλάθι αγορών (Cart)
- Κατηγορία (Category)
- Παραγγελίες (Order)
- Συγγραφέας (Author)
- Σχόλιο (Comment)
- Χρήστης (User)

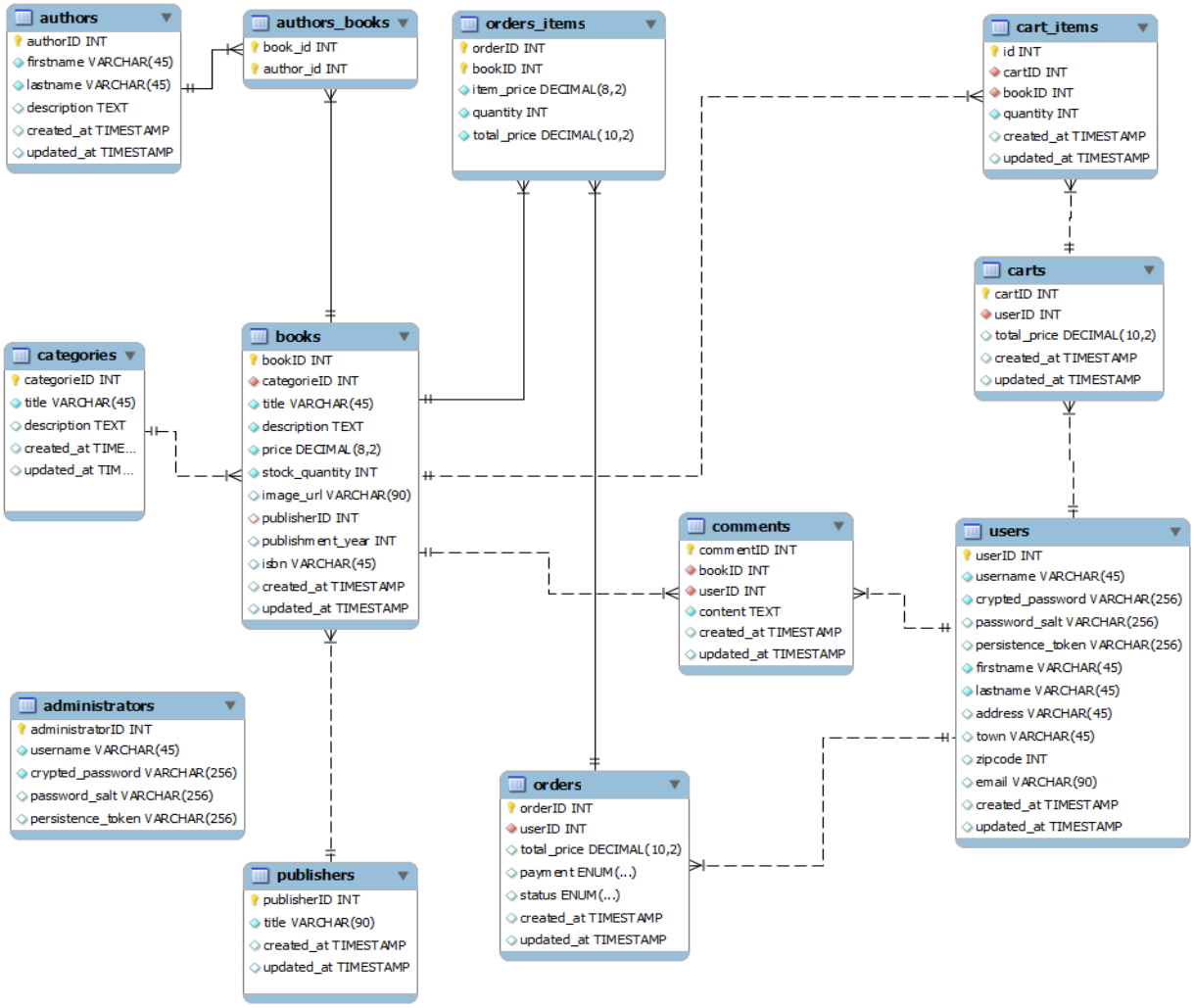
- Session Χρήστη (UserSession)
- Session Διαχειριστή (AdministratorSession)

5.3.2 Συσχετίσεις μεταξύ των μοντέλων

1. Ένα **βιβλίο** ανήκει σε μία κατηγορία (Category), έχει γραφεί από πολλούς συγγραφείς (Author), έχει έναν εκδότη (Publisher), μπορεί να έχει πολλά σχόλια (Comment), μπορεί να ανήκει σε ένα αντικείμενο καλαθιού (CartItem) και μπορεί να ανήκει σε ένα αντικείμενο παραγγελίας (OrderItem).
2. Μια **κατηγορία** μπορεί να έχει πολλά βιβλία (Book).
3. Ενας **συγγραφέας** μπορεί να έχει πολλά βιβλία (Book).
4. Ένας **εκδότης** μπορεί να έχει πολλά βιβλία (Book).
5. Ένα **σχόλιο** ανήκει σε ένα χρήστη (User) και σε ένα βιβλίο (Book).
6. Ένα **καλάθι αγορών** ανήκει σε έναν χρήστη (User) και μπορεί να έχει πολλά αντικείμενα καλαθιού (CartItem).
7. Ένα **αντικείμενο καλαθιού** ανήκει σε ένα καλάθι αγορών (Cart) και έχει ένα βιβλίο (Book).
8. Μία **παραγγελία** ανήκει σε έναν χρήστη (User) και μπορεί να έχει πολλά αντικείμενα παραγγελίας (OrderItem).
9. Ένα **αντικείμενο παραγγελίας** ανήκει σε μία παραγγελία (Order) και έχει ένα βιβλίο (Book).
10. Ένας **χρήστης** μπορεί να έχει πολλά σχόλια (Comment), πολλές παραγγελίες (Order) και ένα καλάθι αγορών (Cart).

5.3.3 Διάγραμμα ER της βάσης δεδομένων

Παρακάτω στο σχήμα φαίνεται το διάγραμμα ER των πινάκων της βάσης δεδομένων της εφαρμογής, που περιέχει όλους τους πίνακες της βάσης με τα πεδία τους και τις συσχετίσεις μεταξύ τους.



Σχήμα 14 "Διάγραμμα ER της βάσης του καταστήματος"

5.4 Ελεγκτές της εφαρμογής

Όπως αναφέρθηκε πιο πάνω το κατάστημα χωρίζεται σε δύο μέρη, το κυρίως μέρος (κατάστημα) και το διαχειριστικό μέρος. Για κάθε μέρος έχουν δημιουργηθεί οι κατάλληλοι ελεγκτές.

5.4.1 Ελεγκτές για την ιστοσελίδα του καταστήματος

Οι ελεγκτές που δημιουργήθηκαν για το κομμάτι της εφαρμογής που είναι ορατό στους πελάτες είναι οι εξής:

- Ελεγκτής για την οντότητα Administrator (AdministratorsController)
  Περιέχει μεθόδους για την είσοδο και έξοδο των διαχειριστών από την εφαρμογή

- Ελεγκτής για τις συνεδρίες των Administrators (AdministratorSessionsController)
  Περιέχει μεθόδους για την αυθεντικοποίηση των διαχειριστών και την διαχείριση των συνεδριών τους με την εφαρμογή

- Ο Γενικός Ελεγκτής της εφαρμογής τον οποίο κληρονομούν όλοι οι άλλοι ελεγκτές (ApplicationController)
  Περιέχει βοηθητικές μεθόδους

- Ελεγκτής για την οντότητα Author (AuthorsController)
  Περιέχει 2 ενέργειες, προβολή καταλόγου συγγραφέων και προβολή ενός συγγραφέα

- Ελεγκτής για την οντότητα Book (BooksController)
  Περιέχει 2 ενέργειες, προβολή ενός βιβλίου και αναζήτηση βιβλίων

- Ελεγκτής για την οντότητα Cart (CartsController)

- Ελεγκτής για την οντότητα Category (CategoriesController)
  Περιέχει 3 ενέργειες, προβολή/ταξινόμηση των βιβλιων μιας κατηγορίας, προβολή όλων των κατηγοριών

- Ελεγκτής για την οντότητα Comment (CommentsController)
  Περιέχει 1 ενέργεια, προσθήκη σχολίου

- Ελεγκτής για την κεντρική σελίδα και το στατικό περιεχόμενο (HomeController)

Περιέχει 2 ενέργειες, προβολή κεντρικής σελίδας και προβολή φόρμας επικοινωνίας

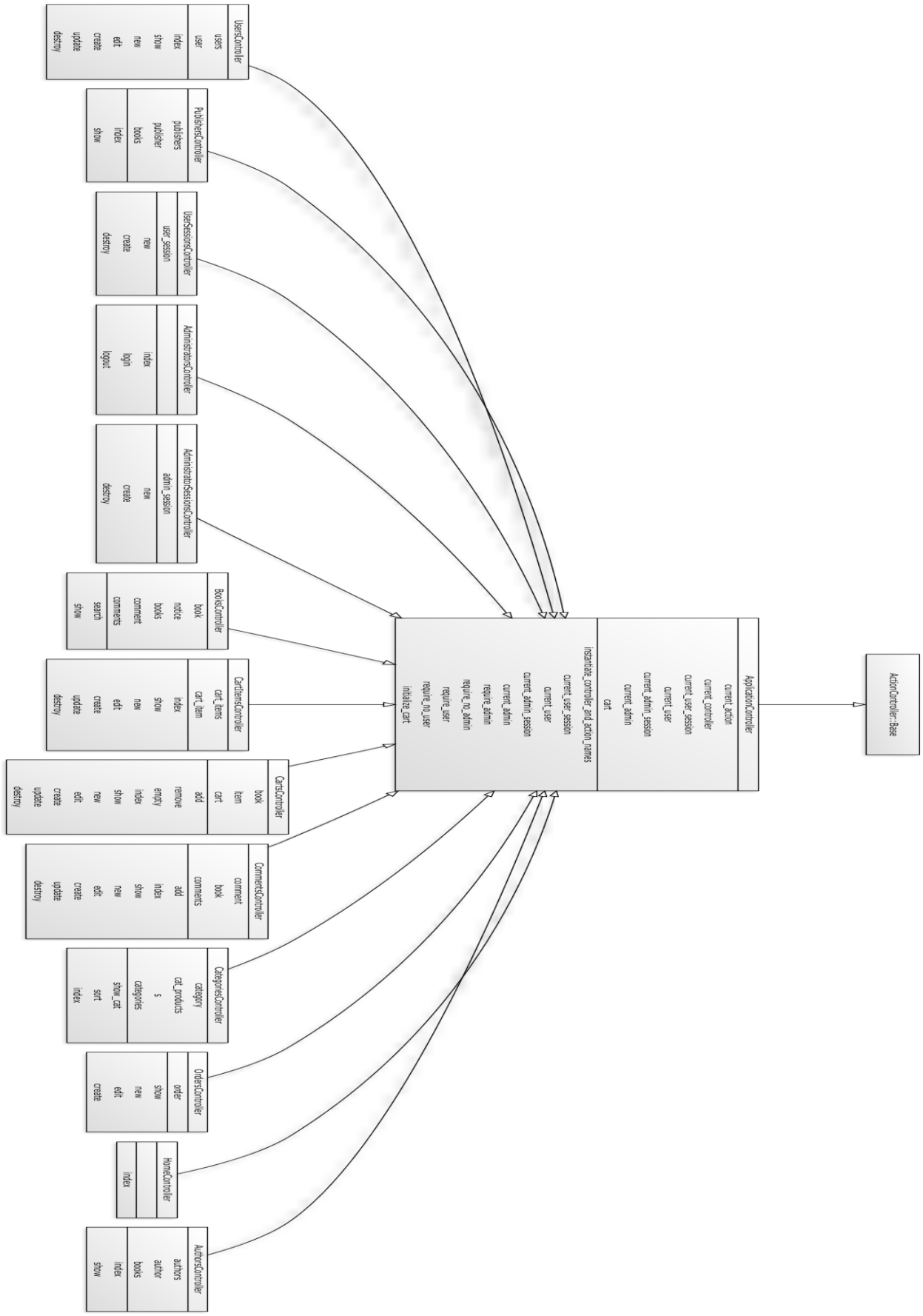
- Ελεγκτής για την οντότητα Order (OrdersController)

- Ελεγκτής για την οντότητα Publisher (PublishersController)
  Περιέχει 2 ενέργειες, προβολή καταλόγου εκδοτών και προβολή ενός εκδότη

- Ελεγκτής για την οντότητα User (UsersController)
  Περιέχει 5 ενέργειες, δημιουργία / προβολή / ενημέρωση χρήστη

- Ελεγκτής για τις συνεδρίες των Users (UserSessionsController)
  Περιέχει μεθόδους για την αυθεντικοποίηση των χρηστών και την διαχείριση των συνεδριών τους με την εφαρμογή

Το διάγραμμα κλάσεων των ελεγκτών για το κομμάτι της εφαρμογής που είναι προσβάσιμο στους πελάτες φαίνεται παρακάτω:



Σχήμα 15 "Διάγραμμα κλάσεων των ελεγκτών της εφαρμογής - 1"

5.4.2 Ελεγκτές για το διαχειριστικό κομμάτι

Οι ελεγκτές που δημιουργήθηκαν για το διαχειριστικό κομμάτι της εφαρμογής που είναι προσπελάσιμο μόνο στους διαχειριστές είναι οι εξής:

- Ελεγκτής για την οντότητα Author (AuthorsController)
  Περιέχει 5 ενέργειες, προβολή καταλόγου συγγραφέων, δημιουργία / προβολή / επεξεργασία / διαγραφή συγγραφέα.

- Ελεγκτής για την οντότητα Book (BooksController)
- *Περιέχει 5 ενέργειες, προβολή καταλόγου βιβλίων, δημιουργία / προβολή / επεξεργασία / διαγραφή βιβλίου.*

- Ελεγκτής για την οντότητα Cart (CartsController)

- Ελεγκτής για την οντότητα Category (CategoriesController)
- *Περιέχει 5 ενέργειες, προβολή καταλόγου κατηγοριών, δημιουργία / προβολή / επεξεργασία / διαγραφή κατηγορίας*

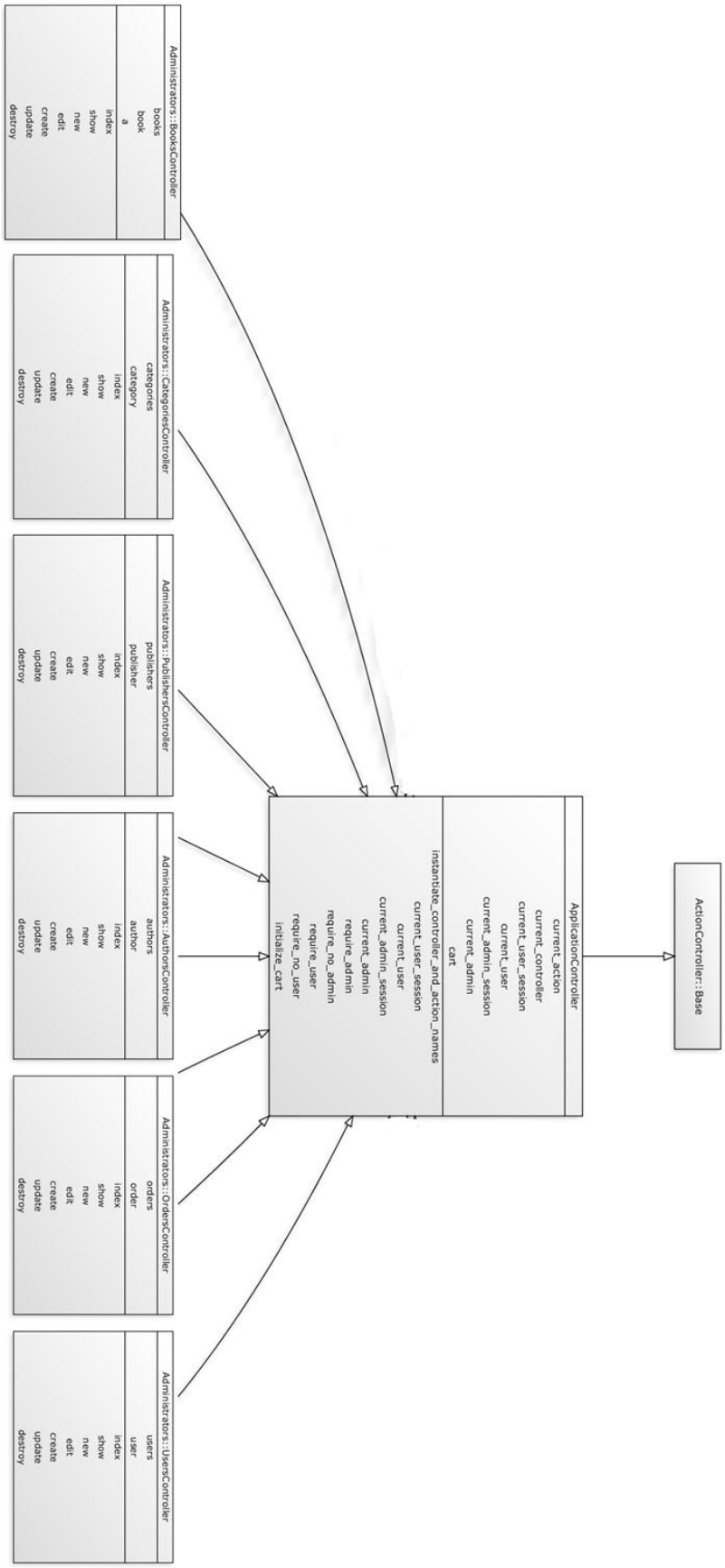
- Ελεγκτής για την οντότητα Comment (CommentsController)
  Περιέχει 1 ενέργεια, προσθήκη σχολίου

- Ελεγκτής για την οντότητα Order (OrdersController)

- Ελεγκτής για την οντότητα Publisher (PublishersController)
- *Περιέχει 5 ενέργειες, προβολή καταλόγου εκδοτών, δημιουργία / προβολή / επεξεργασία / διαγραφή εκδότη*

- Ελεγκτής για την οντότητα User (UsersController)
  Περιέχει 2 ενέργειες, προβολή καταλόγου χρηστών και προβολή χρήστη

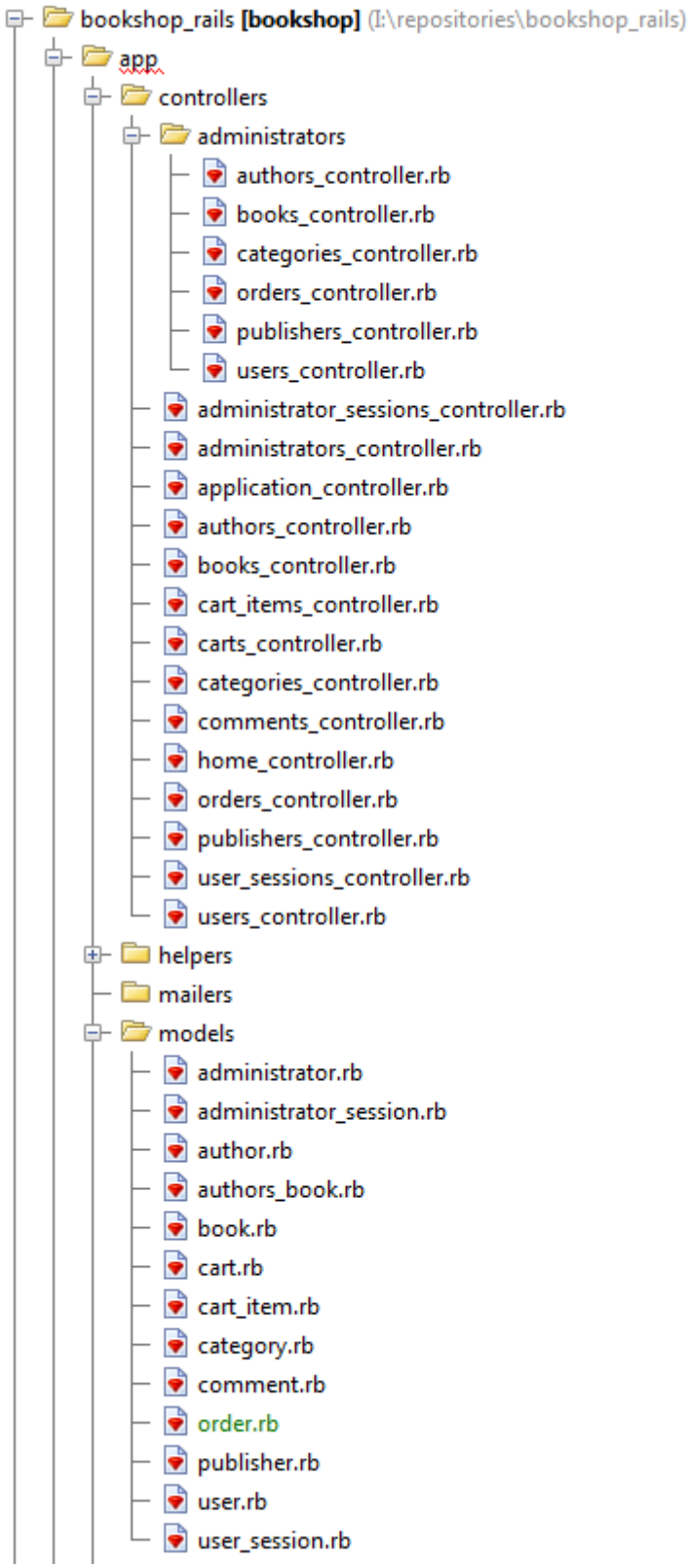
Το διάγραμμα κλάσεων των ελεγκτών για το κομμάτι της εφαρμογής που είναι προσβάσιμο μόνο στους διαχειριστές φαίνεται στο παρακάτω σχήμα:



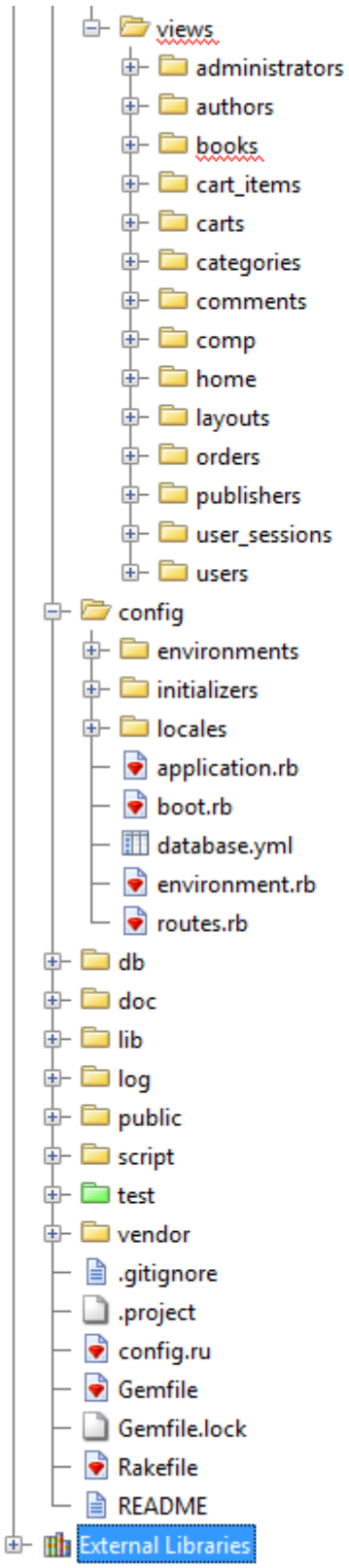
Σχήμα 16 "Διάγραμμα κλάσεων των ελεγκτών της εφαρμογής - 2"

5.5 Δομή της εφαρμογής

Η δομή της εφαρμογής και τα επιμέρους αρχεία που την αποτελούν φαίνονται στις παρακάτω εικόνες που παρουσιάζουν τα αρχεία σε μορφή δέντρου.



Εικόνα 9 "Δομή της εφαρμογής του βιβλιοπωλείου-μέρος 1ο"



Εικόνα 10 "Δομή της εφαρμογής του βιβλιοπωλείου-μέρος 2ο"

5.6 Δημιουργία της βασικής δομής

Η βασική δομή της εφαρμογής δημιουργήθηκε με το εργαλείο scaffold του rails που αναλύθηκε στην σελίδα 54. Για την δημιουργία της βασικής δομή εκτελέστηκαν οι παρακάτω εντολές στην κονσόλα:

```
$ rails generate scaffold Book bookID:integer
categorieID:integer title:string description:text
price:decimal stock_quantity:integer image_url:string
publisherID:integer publishment_year:integer isbn:string

$ rails generate scaffold Author authorID:integer
firstname:string lastname:string description:text

$ rails generate scaffold Category categorieID:integer
title:string description:text

$ rails generate scaffold Cart cartID:integer userID:integer
total_price:decimal

$ rails generate scaffold CartItem cartID:integer
bookID:integer quantity:integer

$ rails generate scaffold Comment commentID:integer
bookID:integer userID:integer content:string

$ rails generate scaffold Order orderID:integer userID:integer
total_price:decimal payment:string status:string

$ rails generate scaffold Order orderID:integer bookID:integer
item_price:decimal quantity:integer total_price:decimal

$ rails generate scaffold Administrator
administratorID:integer username:string
crypted_passoword:string password_salt:string
persistence_token:string

$ rails generate scaffold User userID:integer username:string
crypted_passoword:string password_salt:string
persistence_token:string firstname:string lastname:string
address:string email:string
```

Με την εκτέλεση των εντολών δημιουργούνται οι ελεγκτές, τα μοντέλα και οι προεπιλεγμένες προβολές για όλες τις οντότητες του βιβλιοπωλείου.

5.7 Παρουσίαση περιβάλλοντος χρήστη

Όπως έχει αναφερθεί η εφαρμογή αποτελείται από δύο διαφορετικές σελίδες ανεξάρτητες μεταξύ τους, μία ιστοσελίδα των χρηστών και μία των διαχειριστών. Παρακάτω παρουσιάζονται εικόνες από το περιβάλλον του χρήστη-πελάτη.

5.7.1 Αρχική σελίδα

Η αρχική σελίδα του βιβλιοπωλείου για τους χρήστες είναι η παρακάτω εικόνα.

Εικόνα 11 "Αρχική σελίδα του ηλεκτρονικού βιβλιοπωλείου"

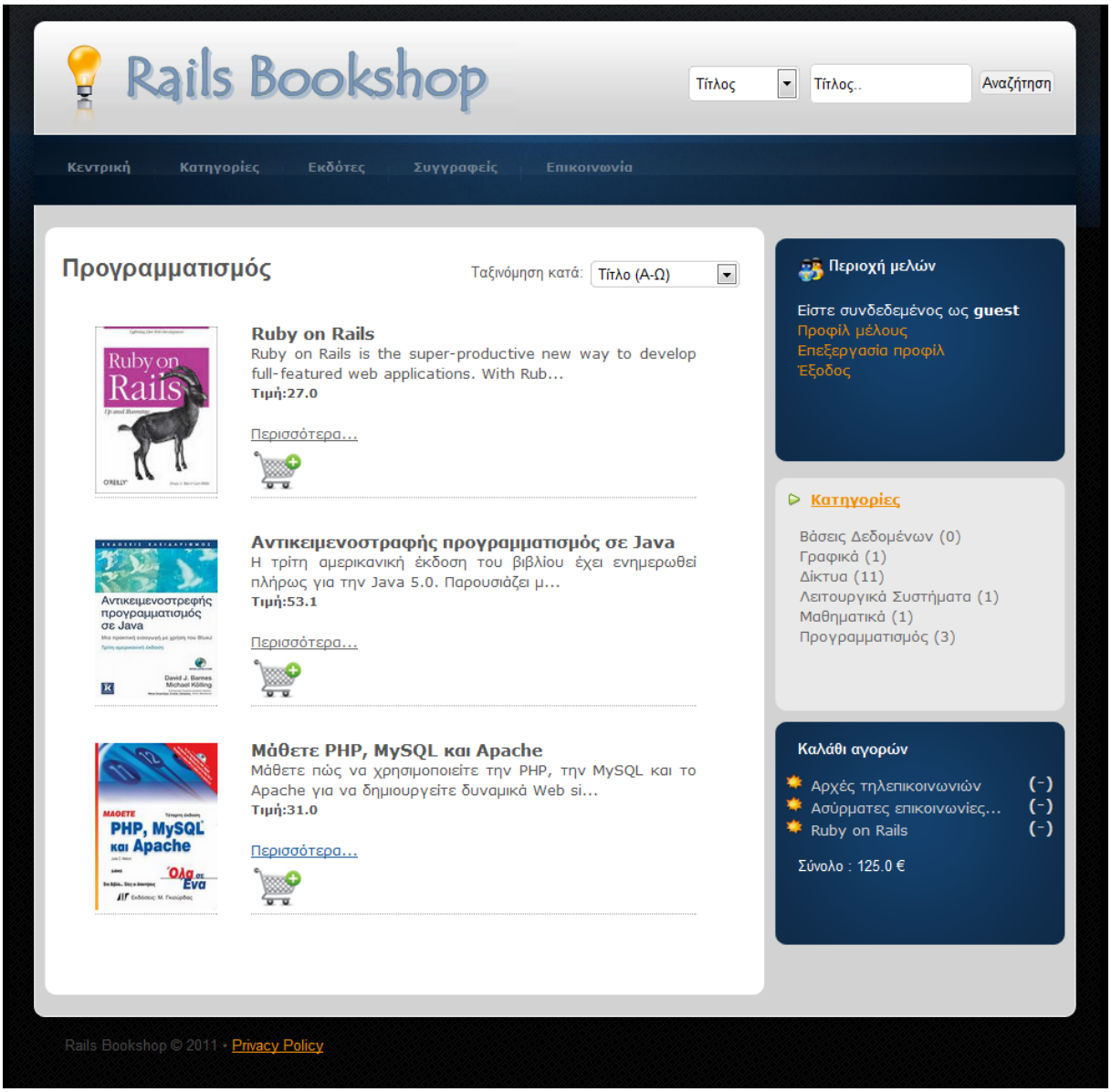
Η ιστοσελίδα αποτελείται από την επικεφαλίδα (header), την αριστερή στήλη, την δεξιά στήλη και το υποσέλιδο (footer). Η επικεφαλίδα περιλαμβάνει το λογότυπο του καταστήματος, την φόρμα αναζήτησης βιβλίων και το κυρίως μενού. Στην αριστερή στήλη τοποθετείται το κυρίως περιεχόμενο που είναι αποτέλεσμα της κάθε ενέργειας και της προβολή της. Η δεξιά στήλη περιλαμβάνει την περιοχή των μελών, τις κατηγορίες των βιβλίων και το καλάθι αγορών. Στο υποσέλιδο περιέχονται διάφοροι σύνδεσμοι και η υπογραφή του καταστήματος.

Η κάθε υποσελίδα δημιουργείται με τον συνδυασμό διάφορων προβολών. Πρώτα αναλύεται η καθολική προβολή (`app/views/layouts/application.htm.erb`), μετά ενσωματώνονται οι μερικές προβολές _search_area, _member_area, _categories, _cart και τέλος το η προβολή της ενέργειας του ελεγκτή που εκτελείται για την συγκεκριμένη διεύθυνση (url).

Εικόνα 12 "Τα επιμέρους μέρη-προβολές από τα οποία δημιουργούνται οι υποσελίδες "

5.7.2 Υποσελίδα μιας κατηγορίας

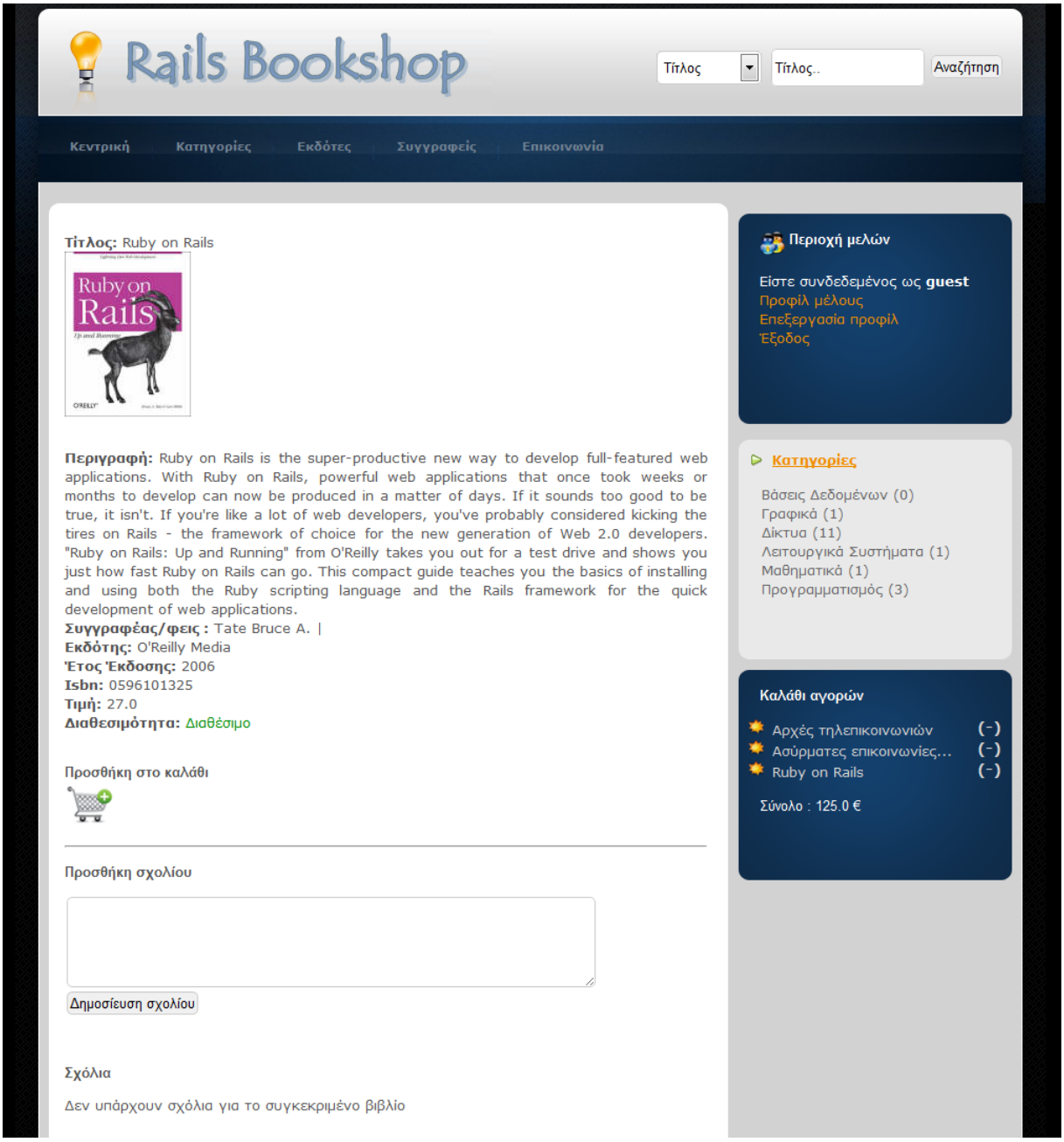
Παρακάτω εμφανίζεται η υποσελίδα μιας κατηγορίας βιβλίων που περιλαμβάνει όλα τα βιβλία που ανήκουν σε αυτήν. Στο κυρίως μέρος της σελίδας παρουσιάζονται τα βιβλία της κατηγορίας επιγραμματικά. Επίσης παρέχεται δυνατότητα για ταξινόμηση των βιβλίων με διάφορα κριτήρια.



Εικόνα 13 "Μία κατηγορία βιβλίων"

5.7.3 Υποσελίδα ενός βιβλίου

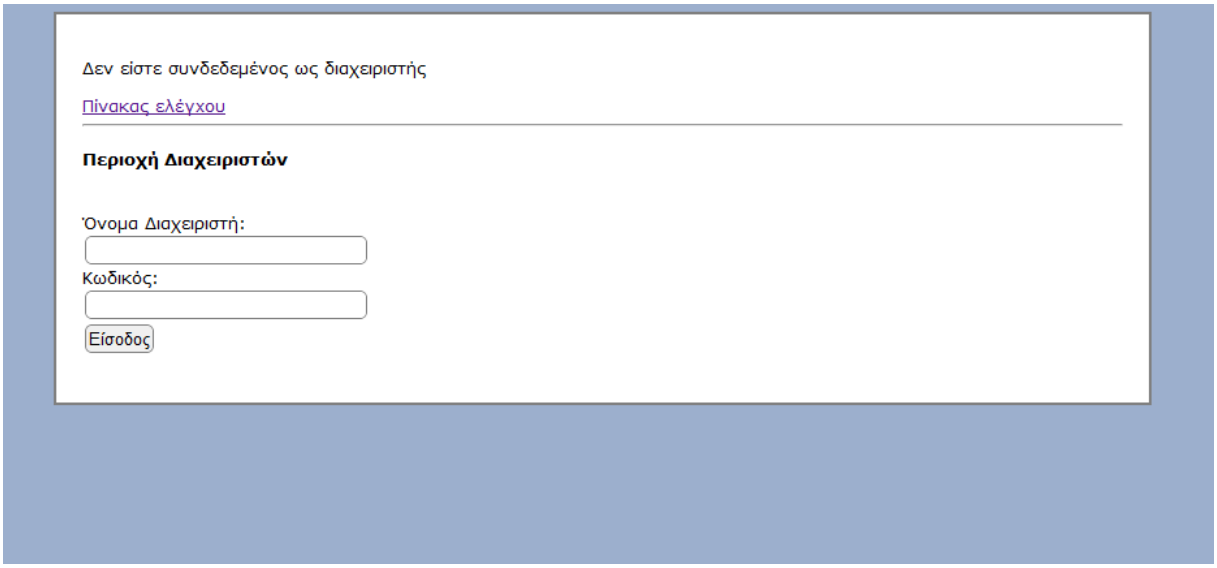
Εδώ παρουσιάζεται η υποσελίδα του κάθε βιβλίου που περιέχει αναλυτικά στοιχεία για το βιβλίο. Επίσης υπάρχει η δυνατότητα προσθήκης στο καλάθι αγορών και η δυνατότητα προσθήκης σχολίων για το βιβλίο.



Εικόνα 14 "Υποσελίδα ενός συγκεκριμένου βιβλίου"

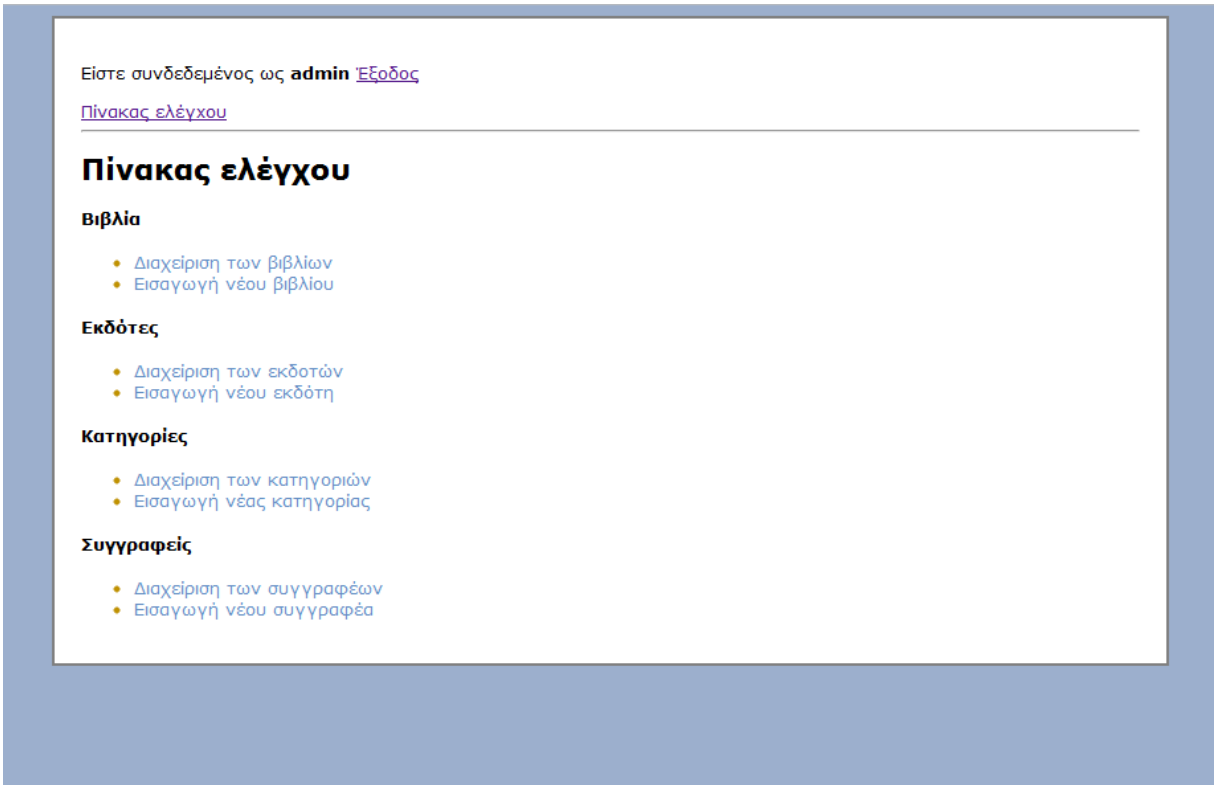
5.8 Παρουσίαση περιβάλλοντος διαχειριστή

Στο περιβάλλον διαχείρισης έχουν πρόσβαση μόνο οι διαχειριστές και η είσοδος σε αυτό πραγματοποιείται με την εισαγωγή των κατάλληλων στοιχείων. Αν δεν έχει πραγματοποιηθεί σύνδεση στο σύστημα, οποιαδήποτε διεύθυνση (της μορφής http://www.mydomain.com/administrators/*) και να προσπελαστεί ανακατευθύνεται στην διεύθυνση http://www.mydomain.com/administrators/login για λόγους ασφαλείας. Η διεύθυνση αυτή παρουσιάζεται στην παρακάτω εικόνα.



Εικόνα 15 "Φόρμα εισαγωγής διαχειριστή"

Αν πραγματοποιηθεί επιτυχής είσοδος στο σύστημα, τότε ο διαχειριστής ανακατευθύνεται στην αρχική σελίδα διαχείρισης. Στην σελίδα διαχείρισης υπάρχει η δυνατότητα προσθήκης /επεξεργασίας / διαγραφής των βιβλίων / κατηγοριών / εκδοτών / συγγραφέων / σχολίων / παραγγελιών.



Εικόνα 16 "Αρχική σελίδα διαχείρισης"

Επίλογος

Σε αυτό το κεφάλαιο αναλύθηκε η διαδικτυακή εφαρμογή ηλεκτρονικού βιβλιοπωλείου που αναπτύχθηκε στα πλαίσια της πτυχιακής εργασίας με την χρήση του Ruby on Rails. Αναφέρθηκε το λογισμικό και τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη της. Επίσης παρουσιάστηκε η αρχιτεκτονική της εφαρμογής με αναφορά στα μοντέλα και τους ελεγκτές που δημιουργήθηκαν, καθώς και τα αντίστοιχα διαγράμματα ER και κλάσεων.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Στην παρούσα πτυχιακή εργασία, πραγματοποιήθηκε μια μελέτη του πλαισίου ανάπτυξης διαδικτυακών εφαρμογών Ruby on Rails. Αρχικά πραγματοποιήθηκε μια εισαγωγή στην γλώσσα προγραμματισμού Ruby, πάνω στην οποία είναι βασισμένο το Ruby on Rails. Αποδείχτηκε μέσω παραδειγμάτων κώδικα κυρίως ότι η σύνταξη της Ruby είναι πιο εύκολη και κατανοητή σε σχέση με άλλες γλώσσες προγραμματισμού. Λόγω του ευανάγνωστου κώδικα της και την περιεκτική της σύνταξη τα προγράμματα που δημιουργούνται είναι μικρότερα σε μέγεθος και πιο εύκολα στην μετέπειτα συντήρηση τους. Ωστόσο υπολείπεται σε απόδοση σε σχέση με άλλες γλώσσες υψηλού επιπέδου όπως η C++ και η Java.

Στη συνέχεια, έγινε παρουσίαση του πλαισίου Ruby on Rails καθώς και των αρχών σχεδίασης και των προτύπων που ακολουθεί. Το Rails με το πλήθος συμβάσεων που ακολουθεί ελαχιστοποιεί την ανάγκη διαμόρφωσης το οποίο έχει ως αποτέλεσμα την συγγραφή μικρότερου σε μέγεθος κώδικα και την ευκολότερη και πιο γρήγορη εργασία από μέρους των προγραμματιστών.

Από τους τέσσερις περίπου μήνες που χρειάστηκαν για την συγγραφή της παρούσας πτυχιακή εργασίας οι εντυπώσεις που αποκομίστηκαν από την χρήση του Ruby on Rails και της Ruby είναι πολύ θετικές, μιας και το Ruby on Rails και η Ruby αποτελούν κάτι διαφορετικό στην κοινότητα του διαδικτυακού (και όχι μόνο) προγραμματισμού. Η σχεδίαση της γλώσσας και του πλαισίου έχει γίνει με τέτοιο τρόπο ώστε να πολλαπλασιάζει την παραγωγικότητα των προγραμματιστών.

ΑΝΑΦΟΡΕΣ

[1]: *ASP.NET*. Ανακτήθηκε 15 Φεβρουαρίου, 2011, από http://www.asp.net/

[2]: *BetterExplained*. Ανακτήθηκε 3 Μαρτίου, 2011, από http://betterexplained.com/articles/intermediate-rails-understanding-models-views-and-controllers/.

[3]: *Builtwith Trends*. Ανακτήθηκε 5 Μαρτίου, 2011, από http://trends.builtwith.com.

[4]: *Builtwith*. Ανακτήθηκε 5 Μαρτίου, 2011, από http://www.builtwith.com.

[5]: *Cakephp*. Ανακτήθηκε 27 Φεβρουαρίου, 2011, από http://www.cakephp.org.

[6]: *Google Trends*. Ανακτήθηκε 5 Μαρτίου, 2011, από http://www.google.com/trends.

[7]: *Indeed*. Ανακτήθηκε 7 Μαρτίου, 2011, από http://www.indeed.com.

[8]: *IronRuby*. Ανακτήθηκε 16 Φεβρουαρίου, 2011, από http://www.ironruby.net.

[9]: *JRuby*. Ανακτήθηκε 7 Φεβρουαρίου, 2011, από http://www.jruby.org.

[10]: *Langpop*. Ανακτήθηκε 7 Μαρτίου, 2011, από http://www.langpop.com.

[11]: *Loudthinking.com, The Rails Myths*. Ανακτήθηκε 8 Ιανουαρίου, 2011, από http://www.loudthinking.com/posts/29-the-rails-myths.

[12]: *MySQL, Interview with David Heinemeier Hansson from Ruby on Rails*. Ανακτήθηκε 10 Ιανουαρίου, 2011, από http://dev.mysql.com/tech-resources/interviews/david-heinemeier-hansson-rails.html.

[13]: *Netuts+, 15 Most Important Considerations when Choosing a Web Development Framework*. Ανακτήθηκε 23 Φεβρουαρίου, 2011, από http://net.tutsplus.com/tutorials/other/15-most-important-considerations-when-choosing-a-web-development-framework/.

[14]: *Railscasts*. Ανακτήθηκε 12 Ιανουαρίου, 2011, από http://www.railscasts.com.

[15]: *RailRoad*. Ανακτήθηκε 12 Μαρτίου, 2011, από http://railroad.rubyforge.org/.

[16]: *Railsforum*. Ανακτήθηκε 19 Φεβρουαρίου, 2011, από http://www.railsforum.com.

[17]: *Railsnotes*. Ανακτήθηκε 2 Φεβρουαρίου, 2011, από http://www.railsnotes.com.

[18]: *Ruby Documentation*. Ανακτήθηκε 11 Ιανουαρίου, 2011, από http://www.ruby-doc.org/core/.

[19]: *Ruby Inside*. Ανακτήθηκε 13 Μαρτίου, 2011, από http://www.rubyinside.com.

[20]: *Ruby on Rails Documentation*. Ανακτήθηκε 20 Ιανουαρίου, 2011, από http://api.rubyonrails.org/.

[21]: *Ruby on Rails guides*. Ανακτήθηκε 22 Ιανουαρίου, 2011, από http://guides.rubyonrails.org/.

[22]: *Ruby on Rails wiki*. Ανακτήθηκε 24 Ιανουαρίου, 2011, από http://wiki.rubyonrails.org/.

[23]: *Ruby on Rails*. Ανακτήθηκε 6 Ιανουαρίου, 2011, από http://www.rubyonrails.org/.

[24]: *Rubystes*. Ανακτήθηκε 23 Φεβρουαρίου, 2011, από http://www.rubyst.es/.

[25]: *SpringSource*. Ανακτήθηκε 14 Μαρτίου, 2011, από http://www.springsource.org/.

[26]: *StackOverflow*. Ανακτήθηκε 9 Μαρτίου, 2011, από http://www.stackoverflow.com/.

[27]: *Tiobe*. Ανακτήθηκε 1 Μαρτίου, 2011, από http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html.

[28]: *WebOnRails*. Ανακτήθηκε 19 Μαρτίου, 2011, από http://www.webonrails.com.

[29]: *Wikipedia IronRuby*. Ανακτήθηκε 2 Μαρτίου, 2011, από http://en.wikipedia.org/wiki/Ironruby.

[30]: *Wikipedia JRuby*. Ανακτήθηκε 2 Μαρτίου, 2011, από http://en.wikipedia.org/wiki/Jruby.

[31]: *Wikipedia, Comparison of web application frameworks*. Ανακτήθηκε 26 Φεβρουαρίου, 2011, από http://en.wikipedia.org/wiki/Comparison_of_web_application_frameworks.

[32]: *Wikipedia, Representational State Transfer*. Ανακτήθηκε 17 Φεβρουαρίου, 2011, από http://en.wikipedia.org/wiki/Representational_State_Transfer.

[33]: *Working with Rails*. Ανακτήθηκε 31 Ιανουαρίου, 2011, από http://www.workingwithrails.com.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Al Barazi R., Carneiro Jr. C. (2010), Beginning Rails 3, Apress, USA

Begin C., Tate A. B., Zygmuntowicz E. (2008), Deploying Rails Applications, The Pragmatic Bookshelf, USA

Benson E. (2008), The Art of Rails, Wiley Publishing Inc., USA

Black A. D. (2006), Ruby for Rails, Manning Publications, USA

Carlson L., Hibbs C., Tate A. B. (2008), Rails: Up and Running Second Edition, O'Reilly Media Inc, USA

Clark M. (2008), Advanced Rails Recipes, The Pragmatic Bookshelf, USA

Dumbill E., St.Laurent S. (2008), Learning Rails, O'Reilly Media Inc, USA

Ediger B. (2008), Advanced Rails, O'Reilly Media Inc, USA

Fernandez O., (2010), The Rails 3 Way, Addison-Wesley, USA

Flanagan D., Matsumoto Y. (2008), The Ruby programming language, O'Reilly Media Inc, Sebastopol, CA, USA

Fowler C., Hunt A., Thomas D. (2009), Programming Ruby 1.9: The Pragmatic Programmer's Guide (Facets of Ruby), The Pragmatic Bookshelf, USA

Griffiths D. (2009), Head first Rails, O'Reilly Media Inc, Sebastopol, CA, USA

Hansson Heinemeier D., Ruby S., Thomas D. (2011), Agile Web Development with Rails (4th Edition), The Pragmatic Programmers, USA

Marshall K., Pytel C., Yurek J. (2007), Pro Active Record: Databases with Ruby and Rails, Apress, USA

Nichols R., Smith E. (2007), Ruby on Rails Enterprise Application Development, Packt Publishing Ltd., UK

Tate A. B., Hibbs C. (2006), Ruby on Rails: Up and Running, O'Reilly Media Inc, USA

Williams J. (2007), Rails Solutions: Ruby on Rails Made Easy, Friendsof, USA