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

Πτυχιακή Εργασία

Ανάπτυξη ιστοσελίδας με το "Yii" ένα framework για την PHP.

Φοιτητής:
Ματκάρης Χρήστος
ΑΜ: 04/2674

Επιβλέπων Καθηγητής:
Σφέτσος Παναγιώτης

Θεσσαλονίκη 2014

Πρόλογος

Τα τελευταία χρόνια, ένας νέος κλάδος προγραμματισμού αναπτύσσεται στους κόλπους της πληροφορικής. Αυτός του διαδικτυακού προγραμματισμού. Η εξάπλωση των broadband συνδέσεων έκαναν το διαδίκτυο προσβάσιμο σε ολοένα και μεγαλύτερη μερίδα του πληθυσμού και άρα αυξήθηκε η χρήση του. Τα περισσότερο χρησιμοποιούμενα προγράμματα σήμερα είναι οι browsers, τα “παράθυρα στο internet” μέσα από τους οποίους οι χρήστες μπορούν να χρησιμοποιήσουνε Online εφαρμογές από οποιοδήποτε σημείο του κόσμου με μόνη απαίτηση την πρόσβαση στο internet. Όλες οι εφαρμογές τείνουν να ανεβαίνουν στο Internet και καινούριες ιδέες, όπως οι εφαρμογές κοινωνικής δικτύωσης, βρίσκουν πρόσφορο έδαφος στο cloud.

Ο διαδικτυακός προγραμματισμός μετράει περίπου 2 δεκαετίες από την γέννησή του, και σήμερα, όσο ποτέ άλλοτε, η κοινότητα των διαδικτυακών προγραμματιστών, στο άνθος της ηλικίας της και με την πείρα τόσων χρόνων, είναι γεμάτη από εργαλεία και ιδέες που βοηθούν στην ανάπτυξη τέτοιων εφαρμογών. Ένα τέτοιο εργαλείο είναι και τα frameworks τα οποία επιταχύνουν σε μεγάλο βαθμό την ανάπτυξη εφαρμογών, ενώ παράλληλα διευκολύνουν κατά πολύ την συντήρησή τους.

Ένα τέτοιο framework είναι και το **Yii**. Βασισμένο στην PHP, την γνωστότερη γλώσσα διαδικτυακού προγραμματισμού, και σε μια σειρά από μεθοδολογίες και αρχιτεκτονικές σχεδίασης αντικειμενοστρεφούς προγραμματισμού, μπορεί να ανταποκριθεί και στην πλέον απαιτητικές εφαρμογές.

Στην εργασία αυτή γίνεται μια αναδρομή του διαδικτυακού προγραμματισμού και των τεχνολογιών του, παρουσιάζεται το Yii Framework και τέλος, δείχνουμε πως μπορούμε με αυτό να δημιουργήσουμε μια web εφαρμογή.

Περίληψη

Το περιεχόμενο της εργασίας αυτής είναι μια εισαγωγή στις δυνατότητες και τις λειτουργίες του Yii Framework και υλοποίηση μιας εφαρμογής με αυτό.

Το Yii είναι βασισμένο στην αρχιτεκτονική **MVC (Model-View-Controller).** Το πρότυπο αυτό μας βοηθά να κρατάμε οργανωμένο τον κώδικά έτσι ώστε να είναι πιο κατανοητός και πιο εύκολα συντηρούμενος.

Η εφαρμογή θα είναι μια πλατφόρμα όπου οι χρήστες θα μπορούν να μοιράζονται τις διαδρομές που σκοπεύουν να πραγματοποιήσουν με κάποιο όχημα τους έτσι ώστε αν βολεύουν και άλλους χρήστες, να έρθουν σε επικοινωνία και να ταξιδέψουν μαζί μοιραζόμενοι τα έξοδα. Το σύστημα αυτό είναι γνωστό ως carpooling.

Για την ολοκλήρωση θα χρησιμοποιηθούν επιπλέον τεχνολογίες και εργαλεία όπως: MVC Framework (Yii), xHTML, CSS, PHP5 ,mySQL XML AJAX (jQuery) κ.α.

Λέξεις κλειδιά: PHP, MVC Framework, HTML, CSS, Carpooling

Abstract

This thesis content is an introduction to the features and abilities of ii Frameork and the development of an application with it.

Yii is based on MVC (Model-View-Controller) architecture. This design pattern helps to keep the source code well organized so that to be easier to read and cultivated.

The application which will be developed will be a platform where users can share their car rides, so that other users who want to have the same ride can contact them, travel together and share the cost. This way of traveling is known as carpooling.

For completing this development a bunch of technologies and tools is going to be needed. Such as MVC Framework (Yii), xHTML, CSS, PHP5 ,mySQL XML AJAX (jQuery) κ.α.

Key words: PHP, MVC Framework, HTML, CSS, Carpooling

Στους γονείς μου, που με κάνανε αυτό που είμαι.

Στα αδέρφια μου, που είναι πάντα δίπλα μου.

Στην Γεωργία, που με στηρίζει ψυχολογικά και υλικά.
Χωρίς αυτήν θα ήμουν ακόμα στην αρχή.

…Σε όλους όσους βαρέθηκαν να με ακούνε να λέω:
"…έχω να κάνω την πτυχιακή…"

Να ευχαριστήσω επίσης θερμά τον κ. Σφέτσο Παναγιώτη για την πολύτιμη βοήθεια του για την πραγματοποίηση της εργασίας αυτής.

Πίνακας Περιεχομένων

1η Ενότητα

Προγραμματισμός στον Ιστό

(Web Programming)

Κεφάλαιο 1: Ιστός και προγραμματισμός

Ενώ ο προγραμματισμός εφαρμογών αριθμεί δεκάδες χρόνια στον χώρο της Πληροφορικής, ένα νέο πεδίο προγραμματισμού εμφανίστηκε εδώ και λίγα χρόνια, μετά την εμφάνιση του Παγκόσμιου Ιστού (World Wide Web) στις αρχές του 1990. Οι διαδικτυακές εφαρμογές ή web εφαρμογές ή cloud εφαρμογές έχουν ήδη καθιερωθεί στην Πληροφορική και έχουν επιφέρει αλλαγές σε όλους τους τομείς της ζωής μας.

Στο κεφάλαιο αυτό θα δούμε τις τεχνολογίες από τις οποίες υλοποιούνται οι web εφαρμογές, την ιστορία και την εξέλιξή τους.

1.1 Λίγα λόγια για τον Παγκόσμιο Ιστό (World Wide Web)

Στα τέλη του 1980 ο μηχανικός Tim Berners-Lee προσπαθούσε να ενώσει την ιδέα του HyperText (διαδραστικό κείμενο) με το Internet (ένα παγκόσμιο δίκτυο υπολογιστών) προσεγγίζοντας ανθρώπους που ασχολούνταν με το ένα ή με το άλλο πεδίο. Αφού κανείς δεν ενδιαφέρθηκε, προχώρησε μόνος του την προσπάθεια. Την εποχή εκείνη εργαζόταν στο ερευνητικό κέντρο CERN και τον Νοέμβριο του 1990, πρότεινε την δημιουργία του World Wide Web. Ένα “δίχτυ(=web)” από κόμβους που περιέχουν “HyperText έγγραφα” τα οποία οι χρήστες θα μπορούν να εμφανίζουν κατά βούληση στον “φυλλομετρητή (=browser)” τους. Το όλο σύστημα θα λειτουργούσε με την ήδη γνωστή αρχιτεκτονική client-server(=πελάτης-εξυπηρετητής). Μέχρι το τέλος του 1990 είχαν δημιουργήσει τον πρώτο browser (ο client), τον πρώτο web server, και τις πρώτες ιστοσελίδες, το περιεχόμενο του συστήματος, οι οποίες περιγράφανε το project. Αυτό ήταν η αρχή του World Wide Web και η βασική λειτουργία διατηρείτε μέχρι σήμερα.

Η λειτουργία είναι η εξής: Ο web server(=web διακομιστής), ένα πρόγραμμα, εγκαθιστάτε σε έναν υπολογιστή -ο οποίος βρίσκεται συνεχώς σε λειτουργία- και η δουλειά του web server είναι να περιμένει αιτήσεις από κάποιον browser. Όταν μια αίτηση για κάποια ιστοσελίδα φτάσει ο web server αναλαμβάνει να ψάξει την ιστοσελίδα αυτή στον υπολογιστή. Η ιστοσελίδα δεν είναι τίποτα άλλο από ένα αρχείο το οποίο αν βρεθεί, αποστέλλεται στον browser που την ζήτησε. Η

δουλειά του browser τώρα είναι να οπτικοποιήσει με φιλικό προς τον χρήστη τρόπο της πληροφορίες του αρχείου.

Για την διαδικασία αυτή χρειάζονται 3 πράγματα:

1. Ένας καθολικός τρόπος εντοπισμού αρχείων για όλο το Internet.
2. Έναν τρόπο επικοινωνίας μεταξύ Web Server και Browser.
3. Μια γλώσσα κωδικοποίησης σε αρχείο κειμένου των πληροφοριών.

Και τα 3 δημιουργήθηκαν από τον Tim Berners-Lee:

1. Ο **UDI**, **U**niversal **D**ocument **I**dentifier (Καθολικός Προσδιοριστής Εγγράφων) ο οποίος πλέον είναι γνωστός ως URI ή URL ή απλά Διεύθυνση και έχει την μορφή: <ΠρωτόκολλοΕπικοινωνίας>**://**<ΌνομαΚόμβου>***/***<ΔιαδρομήΑρχείουΣτονΚόμβο>
2. Το πρωτόκολλο επικοινωνίας “**HTTP**”, **H**yper**T**ext **T**ransfer **P**rotocol. Ένα σύνολο κανόνων και εντολών που επιτρέπει την επικοινωνία μεταξύ web server και client.
3. Την **HTML**, **H**yper**T**ext **M**arkup **L**anguage για την οποία θα μιλήσουμε παρακάτω.

Ενώ η βασική λειτουργία παραμένει ίδια μέχρι και σήμερα, το World Wide Web έχει εμπλουτιστεί με ποικίλες τεχνολογίες και δεν χρησιμοποιείται απλά για ανταλλαγή πληροφοριών όπως είχε ξεκινήσει. Σήμερα σε αντίθεση με τα πρώτα χρόνια, οι χρήστες μπορούν να εισάγουν οι ίδιοι περιεχόμενο σε μια ιστοσελίδα χωρίς να γνωρίζουν πώς λειτουργεί το World Wide Web ή η HTML. Εκτός από τις στατικές σελίδες που υπήρχαν τα πρώτα χρόνια σήμερα στο Web μπορούν να υπάρξουν ολόκληρες εφαρμογές, που πολύ λίγα έχουν να ζηλέψουν από τις παραδοσιακές desktop εφαρμογές. Μάλιστα υπάρχει μια τάση να μεταφέρονται οι desktop εφαρμογές στο Web ή όπως αποκαλείται συχνότερα στο “cloud” (=σύννεφο, όπως συνηθίζεται να απεικονίζεται το Internet σε σχεδιαγράμματα). Εφαρμογές γραφείου, e-mail managers, εγκυκλοπαίδειες, λεξικά ακόμα και ηλεκτρονικά παιχνίδια μεταφέρονται στο Web ακόμα και χώροι αποθήκευσης

δεδομένων οπότε κάποιος μπορεί να έχει πρόσβαση στα αρχεία του από οποιοδήποτε –με πρόσβαση στο Internet- υπολογιστή τον κόσμο.

Αυτό που κυριολεκτικά οργιάζει είναι η υπηρεσίες σε πολλούς τομείς που έχουν μεταφερθεί στο Web: Τραπεζικές συναλλαγές, ηλεκτρονικά καταστήματα, ηλεκτρονική διακυβέρνηση, εφημερίδες, εκπομπές βίντεο, ηλεκτρονικές εκδόσεις, εφαρμογές κοινωνικής δικτύωσης, τηλεφωνία και άλλες πολλές. Όλα τείνουν να γίνουν ηλεκτρονικά και να προστίθεται στο όνομα τους το περίφημο “e-“ (e-banking, e-shop κτλ).

Το World Wide Web στα 23 χρόνια ζωής του έχει επιφέρει τεράστιες αλλαγές σε όλους τους τομείς τις ανθρώπινης ζωής. Πολλοί το θεωρούν την μεγαλύτερη εφεύρεση του ανθρώπου μετά τον τροχό και την τυπογραφία. Έχει βοηθήσει την πρόοδο πολλών τομέων όπως την επιστήμη, την εκπαίδευση. Ενώ φυσικά δεν λείπουν και οι κίνδυνοι που εγείρει η χρήση του. Θέματα περί προσωπικών δεδομένων, ασφάλειας, εθισμού, ιδιωτικότητας και ιδιοκτησίας παίρνουν και αυτά το “e-“ μπροστά στο όνομά τους.

1.2 HTML

Η HTML ή HyperText Markup Language (=Γλώσσα Σήμανσης ΥπερΚειμένου) είναι μα γλώσσα μορφοποίησης περιεχομένου και όχι κλασική γλώσσα προγραμματισμού εντολών. Αποτελεί υποσύνολο της γλώσσας SGML (Standard Generalized Markup Language) που επινοήθηκε από την IBM προκειμένου να λυθεί το πρόβλημα της μη τυποποιημένης εμφάνισης κειμένων στα διάφορα υπολογιστικά συστήματα. Χρησιμοποιείται για να ορίσει πως θα εμφανίσει ένας browser το κείμενο και τα πολυμέσα που περιέχει ένα αρχείο κειμένου html.

Η HTML γράφεται υπό μορφή στοιχείων HTML τα οποία αποτελούνται από ετικέτες, οι οποίες περικλείονται μέσα στα σύμβολα < και > (για παράδειγμα <html>), μέσα στο περιεχόμενο της ιστοσελίδας. Οι ετικέτες (tags) HTML συνήθως λειτουργούν ανά ζεύγη (για παράδειγμα <h1> και </h1>), με την πρώτη να ονομάζεται ετικέτα έναρξης και τη δεύτερη ετικέτα λήξης. Η HTML επιτρέπει την ενσωμάτωση εικόνων και άλλων αντικειμένων μέσα στη σελίδα

μέσω των ετικετών, και μπορεί να χρησιμοποιηθεί για να εμφανίσει διαδραστικές φόρμες.

Ο σκοπός ενός web browser είναι να διαβάζει τα έγγραφα HTML και τα συνθέτει σε σελίδες που μπορεί κανείς να διαβάσει ή να ακούσει. Ο browser δεν εμφανίζει τις ετικέτες HTML, αλλά τις χρησιμοποιεί για να εμφανίσει ανάλογα το περιεχόμενο της σελίδας.

Οι ιδιότητες της HTML μπορούν να διαιρούν το κείμενο ενός εγγράφου σε πακέτα που ονομάζονται elements. Τα πακέτα αυτά μπορούν να χωριστούν σε δύο ευρείες κατηγορίες: 1.Εκείνα που καθορίζουν το πώς το κυρίως σώμα (“body”) του εγγράφου θα εμφανίζεται από το πρόγραμμα περιήγησης, 2.Τις διατάξεις που ορίζουν τις πληροφορίες για το έγγραφο, όπως ο τίτλος ή οι σχέσεις με άλλα έγγραφα.

Τα στοιχεία HTML στην πιο γενική μορφή τους έχουν τρία συστατικά: ένα ζεύγος από ετικέτες, την «ετικέτα εκκίνησης» και την «ετικέτα τερματισμού», μερικές ιδιότητες μέσα στην ετικέτα εκκίνησης, και τέλος το κείμενο ή το γραφικό περιεχόμενο μεταξύ των ετικετών, το οποίο μπορεί να περιλαμβάνει και άλλα στοιχεία εμφωλευμένα μέσα του. Το στοιχείο HTML μπορεί να είναι οτιδήποτε ανάμεσα στις ετικέτες εκκίνησης και τερματισμού. Τέλος, κάθε ετικέτα περικλείεται σε σύμβολα «μεγαλύτερο από» (<) και «μικρότερο από» (>), δηλαδή < και >.

Επομένως, η γενική μορφή ενός στοιχείου HTML είναι:

<tag attribute1="value1" attribute2="value2">content</tag>.

Μερικά στοιχεία HTML περιγράφονται ως άδεια στοιχεία, έχουν τη μορφή

<tag attribute1="value1" attribute2="value2" >

και δεν έχουν καθόλου περιεχόμενο. Το όνομα κάθε στοιχείου HTML είναι το ίδιο όνομα που χρησιμοποιείται στις αντίστοιχες ετικέτες. Το όνομα της ετικέτας τερματισμού ξεκινά με μια κάθετο «/», η οποία παραλείπεται στα άδεια στοιχεία.

Τέλος, αν δεν αναφέρονται ρητά οι ιδιότητες ενός στοιχείου, τότε χρησιμοποιούνται οι προεπιλογές σε κάθε περίπτωση.

Παράδειγμα:

```
<!DOCTYPE html>
<html>
	<head>
		<title>Hello HTML</title>
	</head>
	<body>
		<p>Hello World!</p>
	</body>
</html>
```

(Το κείμενο ανάμεσα στο <html> και το </html> περιγράφει την ιστοσελίδα, και το κείμενο μεταξύ του <body> και του </body> είναι το ορατό μέρος της. Το σημασμένο κείμενο '<title>Hello HTML</title>' καθορίζει τον τίτλο που θα εμφανίζεται στην μπάρα τίτλου του browser.). Το Document Type Declaration στον πιο πάνω κώδικα είναι για την HTML5.

Παρακάτω παρουσιάζονται περιληπτικά οι σημαντικότερες και συχνότερες ετικέτες HTML.

Πίνακας 1.2.1 : Ετικέτες Κειμένου

< ! --...-->	Ορίζει ένα σχόλιο. Το σχόλιο αυτό δεν εμφανίζεται στον browser.
 	Ορίζει μια αλλαγή γραμμής.
...	Ορίζει μια παράγραφο.
.	Ορίζει ένα κείμενο σε έντονη γραφή.
.	Ορίζει ένα κείμενο σε πλάγια γραφή.
<u>...</u>	Ορίζει ένα υπογραμμισμένο κείμενο.
<tt>.</tt>	Ορίζει ένα κείμενο με γραμματοσειρά γραφομηχανής.
<hl>...</hl>	Ορίζει μια επικεφαλίδα. Το I ορίζει το επίπεδο και παίρνει τιμές από 1 έως 6 (hi - h6).
<font >...< / font >	Ορίζει την γραμματοσειρά του κειμένου.
<hr>	Ορίζει μια οριζόντια γραμμή.
<sup>...</ sup>	Ορίζει έναν εκθέτη.
_{...}	Ορίζει ένα δείκτη.
...	Τοποθετεί το κείμενο στο κέντρο.

Πίνακας 1.2.2 : Ετικέτες Καταλόγων (Λίστες)

...	Ορίζει έναν μη αριθμημένο κατάλογο.
...	Ορίζει έναν αριθμημένο κατάλογο.
...	Ορίζει ένα στοιχείο καταλόγου.
<dl>...</dl>	Ορίζει έναν ερμηνευτικό κατάλογο.
<dt>...</dt>	Ορίζει έναν όρο επεξήγησης.
<dd>...</dd>	Ορίζει ένα κείμενο επεξήγησης.

Πίνακας 1.2.3: Ετικέτες Υπερσυνδέσμων

<a>...	Ορίζει έναν υπερσύνδεσμο. Η διεύυνση του ορίζεται στο attribute «href».

Πίνακας 1.2.4 : Ετικέτες Πολυμέσων

< img>...</ img>	Τοποθετεί μια εικόνα στη σελίδα.
<embed>...</embed>	Τοποθετεί ένα πολυμεσικό στοιχείο (ήχο ή βίντεο) στη σελίδα με δυνατότητα αναπαραγωγής.

Πίνακας 1.2.5: Ετικέτες Πινάκων

<table>...</table>	Ορίζει έναν πίνακα.
<tr>...</tr>	Ορίζει μια γραμμή πίνακα.
<td>...</td>	Ορίζει ένα κελί πίνακα.
<th>...</th>	Ορίζει μια επικεφαλίδα πίνακα.

Πίνακας 1.2.6: Ετικέτες Πλαισίων

<frameset>...</frameset>	Ορίζει ένα σετ πλαισίων.
<frame>...</frame>	Ορίζει ένα πλαίσιο.

Πίνακας 1.2.7: Ετικέτες Φόρμας

< form>...</form>	Ορίζει μια φόρμα.
< input >...</input >	Ορίζει ένα στοιχείο εισόδου της φόρμας..
<select>...</select>	Ορίζει μια λίστα επιλογών της φόρμας.

<opt ion>...</opt ion>	Ορίζει μια επιλογή της λίστας.
<textarea>...</textarea>	Ορίζει ένα κυλιόμενο πεδίο εισαγωγής κειμένου.

Οι φόρμες είναι ένας τρόπος αποστολής δεδομένων από τον χρήστη προς έναν ιστότοπο μέσω της μεθόδου POST που ορίζει το HTTP. Ένας άλλος τρόπος είναι ο GET όπου τα δεδομένα συμπεριλαμβάνονται στην διεύθυνση URL σε μορφή μεταβλητών. Μετά τα τέλος του html αρχείου ακολουθεί το ? και έπειτα οι μεταβλητές με τις τιμές τους. Πχ https://www.teithe.gr/index.html?department=it.

1.2.1 JavaScript

Η JavaScript είναι ίσως η πιο διαδεδομένη γλώσσα σεναρίου (script language) και χρησιμοποιείται για να εισάγουμε την έννοια της διαδραστικότητας στις HTML σελίδες. Αποτελεί μια ερμηνευτική γλώσσα (interpreted language), δηλαδή τα scripts εκτελούνται χωρίς να έχει προηγηθεί μεταγλώττιση του κώδικα. Αναπτύχθηκε από τον Brendan Eich και εμφανίστηκε στους browsers το 1996, ενώ το 1997 η επίσημη τυποποίησή της (standardization) θεσπίστηκε από την ECMA Organization.

Η JavaScript είναι μία σειρά από δηλώσεις (statements) που θα εκτελέσει ο browser στην σελίδα που ήδη παρουσιάζει. Δηλαδή ένα JavaScript statement είναι μια εντολή προς τον browser, η οποία του ορίζει ποια ενέργεια πρέπει να εκτελέσει. Με την JavaScript μπορούμε να εκτελέσουμε ενέργειες όταν συμβαίνει ένα γεγονός, για παράδειγμα όταν ο χρήστης κάνει κλικ σε ένα HTML στοιχείο, να εκτελείται κάποιο script και να λαμβάνουμε τα αντίστοιχα αποτελέσματα. Μπορεί να διαβάσει και να αλλάξει τα περιεχόμενα ενός HTML στοιχείου, ενώ μας δίνει τη δυνατότητα να τη χρησιμοποιήσουμε για να επικυρώσουμε τα δεδομένα μιας φόρμας (validate) προτού να υποβληθούν στον server. Με την βοήθεια της JavaScript μπορούμε να εντοπίσουμε τον browser του επισκέπτη και ανάλογα με τον browser να φορτώσουμε την αντίστοιχη σελίδα που είναι φτιαγμένη για αυτόν τον browser, ενώ μας επιτρέπει να δημιουργήσουμε cookies, δηλαδή να αποθηκεύουμε και να λαμβάνουμε πληροφορίες στον υπολογιστή του επισκέπτη.

Για να εισάγουμε JavaScript σε ένα HTML αρχείο χρησιμοποιούμε τη ετικέτα <script> και μέσα σε αυτή την ετικέτα χρησιμοποιούμε το όρισμα "type" για να ορίσουμε την γλώσσα script που θα χρησιμοποιήσουμε. Οι ετικέτες <script>...</script> μας δηλώνουν που αρχίζει και που τελειώνει η JavaScript. Αν δεν βάλουμε τον κώδικα JavaScript μέσα σε ετικέτες script ο browser τον αντιλαμβάνεται σαν απλό κείμενο και δεν τον εκτελεί.

Για παράδειγμα ας δούμε το παρακάτω html έγγραφο:

Το script ορίσει μια function “displayDate()” η οποία λέει στον browser να βρεί το element με id=demo και να ορίσει το .innerHTML του, δηλαδή το κείμενο ανάμεσα στα tags του, την τρέχουσα ημερομηνία. Έπειτα στο element button του λέμε ότι όταν γίνει click να τρέξει την function “dislpalyDate()”. Έτσι όταν ο

```
<html>
<head>
<script type="text/javascript">
     function displayDate() {
          document.getElementById("demo").innerHTML=Date(
     );
     }
</script>
</head>
<body>
<h1>Καλώς ήρθατε στη σελίδα μου!</h1>
<p id="demo">Μόλις πατήσετε το κουμπί, σε αυτήν την
παράγραφο θα εμφανιστεί η ημερομηνία.</p>
<button type="button" onclick="displayDate()">Εμφάνιση
Ημερομηνίας </button>
</body>
</html>
```

χρήστης κάνει κλίκ στο κουπί αυτό το κείμενο ανάμεσα στα   θα αντικατασταθεί από την τρέχουσα ημερομηνία.

1.2.2 CSS

Ο όρος CSS είναι το ακρωνύμιο των λέξεων Cascading Style Sheets, δηλαδή Διαδοχικά Φύλλα Στυλ. Η χρήση των φύλλων στυλ για τη διαμόρφωση της εμφάνισης του τελικού αποτελέσματος της σελίδας στην οθόνη έχουν εισαχθεί με τις προδιαγραφές της HTML 4.0 και παρέχουν στον προγραμματιστή τη

δυνατότητα ομοιόμορφης διαμόρφωσης των ιστοσελίδων με ένα γενικό και συγκεντρωτικό τρόπο. Η εισαγωγή των φύλλων στυλ έχει σαν σκοπό την αύξηση των δυνατοτήτων της HTML για παρουσίαση πληροφοριών με έναν προκαθορισμένο τρόπο.

Τα style sheets μπορούν να δημιουργηθούν μέσα στο ίδιο το έγγραφο με την χρήση της ετικέτας <style> ή σε ένα εξωτερικό αρχείο και να καλούνται στη συνέχεια από το έγγραφο. Μας δίνεται έτσι η δυνατότητα να διαμορφώσουμε πλήθος συσχετιζόμενων ιστοσελίδων χρησιμοποιώντας μόλις ένα φύλλο στυλ, με τρόπο ομοιόμορφο και σταθερό. Επιπλέον, οι κανόνες στυλ που εφαρμόζονται με την χρήση εξωτερικών style sheets (εξωτερικών αρχείων), είναι δυνατό να μεταβληθούν εύκολα από τον προγραμματιστή και στη συνέχεια να εφαρμοσθούν στο έγγραφο χωρία καμία παρεμβολή στον κώδικα του εγγράφου.

Σε γενικές γραμμές μπορούμε να πούμε ότι όλα τα στυλ θα καταλήξουν (cascade) σ' ένα νέο εικονικό (virtual) Φύλλο Στυλ σύμφωνα με τους παρακάτω κανόνες, όπου ο μεγαλύτερος αριθμός έχει και την υψηλότερη προτεραιότητα :

1)Προεπιλογή του browser.

2)Εξωτερικό Φύλλο Στυλ (External Style Sheet).

```
<head>
   <link rel="stylesheet" type="text/css"
href="mystyle.css" />
</head>
```

3) Εσωτερικό Φύλλο Στυλ (Internal Style Sheet), μέσα στο τμήμα <head> του εγγράφου.

```
<head>
     <style type="text/css">
          hr {color:red;}
          p {margin-left:20px;}
          body {background-image:url("images/back-
          img.gif");}
     </style>
```

```
</head>
```

4) Inline Style, μέσα στην HTML ετικέτα.

```
<p style="color:red;margin-left:2 0px">
     Αυτή  είναι  μια παράγραφος.
</p>
```

Η σύνταξη των CSS αποτελείται από τρία μέρη : έναν επιλογέα (selector), μια ιδιότητα (property) και μια τιμή (value) στην μορφή:

επιλογέας {ιδιότητα: τιμή}

selector {property: value}

Ο επιλογέας είναι συνήθως η ετικέτα που θέλουμε να ορίσουμε, η ιδιότητα είναι το χαρακτηριστικό που θέλουμε να αλλάξουμε και η κάθε ιδιότητα μπορεί να πάρει μια τιμή. Η ιδιότητα και η τιμή ξεχωρίζουν από τον χαρακτήρα “:” και περικλείονται από τους χαρακτήρες "{" και "}", ως εξής :

```
body {color: black}
```

Αν η τιμή αποτελείται από πολλές λέξεις, πρέπει να τοποθετήσουμε εισαγωγικά :

```
p {font-family: "courier new"}
```

Αν θέλουμε να ορίσουμε περισσότερες από μία ιδιότητες, πρέπει να ξεχωρίσουμε την κάθε ιδιότητα με τον χαρακτήρα “;” Το παρακάτω παράδειγμα δείχνει πώς μπορούμε να ορίσουμε μια κεντραρισμένη παράγραφο με χρώμα κειμένου μπλε :

```
p {text-align: center; color: blue}
```

Επίσης μπορούμε να ομαδοποιήσουμε τους επιλογείς, ξεχωρίζοντας τον κάθε επιλογέα με κόμμα. Στο παρακάτω παράδειγμα έχουμε ομαδοποιήσει όλες τις ετικέτες επικεφαλίδας. Το κείμενο της κάθε επικεφαλίδας θα είναι κόκκινο :

```
hi, h2, h3, h4, h5, h6 {
    color: red }
```

Με το χαρακτηριστικό (attribute) class μπορούμε να ορίσουμε διαφορετικά στυλ για την ίδια ετικέτα. Ας υποθέσουμε ότι θέλουμε να έχουμε δύο είδη παραγράφων στο έγγραφό μας, μια δεξιά στοιχισμένη παράγραφο και μια κεντραρισμένη παράγραφο.

Να πώς μπορούμε να το κάνουμε αυτό με τα στυλ :

```
p.right {text-align: right} p.center {text-align:
center}
```

Και πρέπει τώρα να χρησιμοποιήσουμε το χαρακτηριστικό class στο HTML έγγραφο, ως εξής :

```
<p class="right">
    Αυτή είναι μια παράγραφος. Το κείμενο αυτής της
    παραγράφου θα είναι δεξιά στοιχισμένο.
</p>
<p class="center">
    Αυτή είναι μια άλλη παράγραφος. Το κείμενο αυτής της
    παραγράφου θα είναι κεντραρισμένο.
</p>
```

Με το χαρακτηριστικό id μπορούμε να ορίσουμε ένα μοναδικό στυλ που μπορούμε να χρησιμοποιήσουμε σε πολλές ετικέτες. Ορίστε πώς μπορεί να γίνει αυτό με τα στυλ :

```
#right {text-align: right}
```

Στο HTML έγγραφο πρέπει να γράψουμε τα εξής :

```
<p id="right">
    Αυτή είναι μια παράγραφος. Το κείμενο αυτής της
    παραγράφου θα είναι δεξιά στοιχισμένο.
</p>
<h3  id="right">
    Αυτή είναι μια επικεφαλίδα. Αυτή η επικεφαλίδα
    θα είναι επίσης δεξιά στοιχισμένη.
</h3>
```

1.3 CGI

Με την χρήση της HTML οι ιστοσελίδες που δημιουργούνται είναι στατικές. Από την στιγμή που θα αποθηκευτούν στον server, οι ενέργειες ενός επισκέπτη της ιστοσελίδας δεν μπορούν να αλλάξουν το περιεχόμενο τους (Το περιεχόμενο των αρχείων στον server και όχι την προβολή τους από τον browser όπως κάνει η JavaScript). Για να γίνουν οι σελίδες δυναμικές και να αλλάζουν το περιεχόμενο τους ανάλογα με τις επιταγές του επισκέπτη, εμφανίστηκε το CGI αρχικά των **C**ommon **G**ateway **I**nterface. Το CGI "πάντρεψε" την HTML με τις κλασικές γλώσσες προγραμματισμού, ανοίγοντας τον δρόμο για τον διαδικτυακό προγραμματισμό και κάνοντας δυνατές τις εφαρμογές στο web.

Το CGI είναι ένας τρόπος επικοινωνίας του web server με μια οποιαδήποτε γλώσσα προγραμματισμού ανεξαρτήτως λειτουργικού συστήματος. Δεν αποτελεί το ίδιο μια γλώσσα προγραμματισμού. Έτσι μπορούν να υπάρξουν CGI scripts, όπως συνηθίζονται να λέγονται, τα οποία είναι γραμμένα σε C, Perl, Python, Visual Basic και σε οποιαδήποτε γλώσσα προγραμματισμού ή ακόμα και γλώσσα σεναρίου εντολών όπως είναι τα batch files των windows ή τα bash scripts του Unix. Ιστορικά, επειδή η Perl παρείχε ευκολίες στην επεξεργασία strings συνδέθηκε περισσότερο με το CGI και δημιουργήθηκαν πληθώρα βιβλιοθηκών που διευκόλυναν των διαδικτυακό προγραμματισμό.

Σύμφωνα με το CGI ο web server στέλνει σαν είσοδο στο προς εκτέλεση πρόγραμμα ορισμένες μεταβλητές περιβάλλοντος. Το μόνο που απαιτείται από την μεριά του προγράμματος είναι η έξοδός του να αποτελεί ένα string το οποίο

είναι ένα HTML έγγραφο. Ουσιαστικά λοιπόν αυτό που κάνει ο web server μέσω του CGI είναι να αποστέλλει την έξοδο του προγράμματος στον browser και αυτός να εμφανίζει την σελίδα. Ο επισκέπτης μπορεί πλέον να επηρεάσει την έξοδο του προγράμματος καθώς μέσα στις μεταβλητές περιβάλλοντος είναι τα περιεχόμενα μιας HTML φόρμας τα οποία έχει εισάγει ο ίδιος. Όπως είδαμε υπάρχουν 2 μέθοδοι αποστολής στον web server πληροφοριών μέσω μιας φόρμας. Η μέθοδος POST και η μέθοδος GET. Επίσης μεταβλητές μπορούν αν δοθούν απευθείας από το URL του script ως query string (αυτός είναι και ο τρόπος με τον οποίο αποστέλλονται τα δεδομένα μιας φόρμας με την μέθοδο GET)

Σήμερα, συνηθίζεται, οι διακομιστές να μην εκτελούν εξωτερικά τα προγράμματα script αλλά εσωτερικά μέσω module. Για παράδειγμα, στον Apache μπορούμε να προσθέσουμε ένα module για την PHP (θα μιλήσουμε για αυτήν παρακάτω) και έτσι θα τρέχει τον parser της εσωτερικά σαν μέρος του κώδικα του, χωρίς εξωτερική κλήση.

1.4 Βάσεις Δεδομένων

Αφού λοιπόν, με το CGI, προγράμματα μπορούσαν να εκτελεστούν μέσω web, αυτόματα οι βάσεις δεδομένων μπήκαν και αυτές στο παιχνίδι μιας και κάθε γλώσσα προγραμματισμού που σέβεται τον εαυτό της παρέχει μεθόδους αλληλεπίδρασης με τουλάχιστον ένα σύστημα βάσης δεδομένων. Πλέον οι ιστοσελίδες άρχισαν να μοιάζουν με εφαρμογές, αφού γίνεται ευκολότερη η επεξεργασία τεράστιου όγκου δεδομένων.

Τι είναι όμως μια βάση δεδομένων;

Πολύ απλά, μια βάση δεδομένων είναι μια συλλογή δεδομένων ή πληροφοριών με συγκεκριμένη οργάνωση και δομή, οι οποίες σχετίζονται με ένα συγκεκριμένο θέμα ή σκοπό και είναι αποθηκευμένες σε ένα αποθηκευτικό μέσο. Παραδείγματα βάσεων δεδομένων είναι ο τηλεφωνικός κατάλογος, η διατήρηση μιας συλλογής μουσικών κομματιών, η παρακολούθηση των παραγγελιών των πελατών και των ειδών των προϊόντων ενός καταστήματος και άλλες.

Τα δεδομένα μιας βάσης δεδομένων (π.χ. όνομα, διεύθυνση, τηλέφωνο) έχουν αξία όταν μπορούν να συσχετιστούν και να αποκτήσουν συγκεκριμένο νόημα (π.χ. ταυτοποίηση προσώπου) προκειμένου μα παραχθεί χρήσιμη πληροφορία. Μια βάση δεδομένων σχεδιάζεται, κτίζεται και αποθηκεύει πληροφορίες για ένα ειδικό τμήμα του κόσμου ή σκοπό, όπως:

- Μαθητολόγιο / Πελατολόγιο
- Φορολογία εισοδήματος
- Τραπεζικές συναλλαγές

Όπως όλα έτσι και μια ηλεκτρονική βάση δεδομένων προσφέρουν πολλά περισσότερα. Πέρα από την εγγενή της ικανότητα να αποθηκεύει δεδομένα, η ηλεκτρονική βάση δεδομένων παρέχει βάσει του σχεδιασμού και του τρόπου ιεράρχησης των δεδομένων της σε προγράμματα ή συλλογές προγραμμάτων,

τα αποκαλούμενα συστήματα διαχείρισης περιεχομένου, τη δυνατότητα γρήγορης άντλησης και ανανέωσης των δεδομένων. Η ηλεκτρονική βάση δεδομένων χρησιμοποιεί ιδιαίτερου τύπου λογισμικό προκειμένου να οργανώσει την αποθήκευση των δεδομένων της. Το διακριτό αυτό λογισμικό είναι γνωστό ως Σύστημα Διαχείρισης Βάσης δεδομένων, ή εν συντομία, ΣΔΒΣ και αγγλιστί DataBase Mangement System, DBMS.

Η web εφαρμογή της παρούσας πτυχιακής εργασίας υλοποιείται με PHP (ως CGI Script) και MySQL (ως DBMS). Περιλαμβάνει μια βάση δεδομένων η οποία περιέχει πίνακες δεδομένων. Μέσα σε κάθε πίνακα δηλώνουμε τον τύπο κάθε πεδίου (π.χ. Char, int, date) και το πρωτεύων του κλειδί (primary key), το οποίο μπορεί να αποτελείται από ένα πεδίο ή και από συνδυασμό περισσότερων πεδίων με την προϋπόθεση να είναι μοναδικό (unique). Οι πίνακες μέσα στη βάση συνδέονται μεταξύ τους με τα πρωτεύοντα κλειδιά, έτσι ώστε εάν ένας πίνακας αλλάξει δεδομένα, να ενημερώνονται και οι υπόλοιποι πίνακες που είναι συνδεδεμένοι με αυτόν. Επίσης, μέσω των ξένων κλειδιών (foreign keys), τα οποία αποτελούνται από πεδία-ορόσημα για την περάτωση του σκοπού μας, καταφέρνουμε να συνδέσουμε δυναμικά ή στατικά τους πίνακες, για να εμφανίζουμε τα πεδία ή τους πίνακες που επιθυμούμε κατά την περιήγηση στο site.

Κεφάλαιο 2: Η PHP

Μέσα από μια πληθώρα γλωσσών προγραμματισμού web εφαρμογών όπως:

- Python
- Perl
- JSP
- ASP
- Ruby

για την πραγματοποίηση της πτυχιακής επιλέχθηκε η PHP (την οποία χρησιμοποιεί το Yii). Μια ανοικτού κώδικα γλώσσα προγραμματισμού με την μεγαλύτερη κοινότητα υποστήριξης και η πιο διαδεδομένη γλώσσα για δημιουργία web εφαρμογών. Το κεφάλαιο αυτό αποτελεί μια παρουσίαση της PHP.

2.1 Ιστορία

Η ιστορία της PHP ξεκινά από το 1994, όταν ένας φοιτητής, ο Rasmus Lerdorf δημιούργησε χρησιμοποιώντας τη γλώσσα προγραμματισμού Perl ένα απλό CGI script με όνομα php.cgi, για προσωπική χρήση. Το script αυτό είχε σαν σκοπό να διατηρεί μια λίστα στατιστικών για τα άτομα που έβλεπαν το online βιογραφικό του σημείωμα. Αργότερα αυτό το script το διέθεσε και σε φίλους του, οι οποίοι άρχισαν να του ζητούν να προσθέσει περισσότερες δυνατότητες. Η γλώσσα τότε ονομαζόταν PHP/FI από τα αρχικά Personal Home Page/Form Interpreter. Το 1997 η PHP/FI έφθασε στην έκδοση 2.0, βασιζόμενη αυτή τη φορά στη γλώσσα C και αριθμώντας περισσότερους από 50.000

ιστότοπους που τη χρησιμοποιούσαν, ενώ αργότερα την ίδια χρονιά οι Andi Gutmans και Zeev Suraski ξαναέγραψαν τη γλώσσα από την αρχή, βασιζόμενοι όμως αρκετά στην PHP/FI 2.0. Έτσι η PHP έφθασε στην έκδοση 3.0 η οποία θύμιζε περισσότερο τη σημερινή μορφή της. Στη συνέχεια, οι Zeev και Andi δημιούργησαν την εταιρεία Zend (από τα αρχικά των ονομάτων τους), η οποία συνεχίζει μέχρι και σήμερα την ανάπτυξη και εξέλιξη της γλώσσας PHP. Ακολούθησε το 1998 η έκδοση 4 της PHP, τον Ιούλιο του 2004 διατέθηκε η έκδοση 5, ενώ αυτή τη στιγμή έχουν ήδη διατεθεί και οι πρώτες δοκιμαστικές εκδόσεις της επερχόμενης PHP 6, για οποιονδήποτε προγραμματιστή θέλει να τη χρησιμοποιήσει. Οι περισσότεροι ιστότοποι επί του παρόντος χρησιμοποιούν κυρίως τις εκδόσεις 4 και 5 της PHP.

2.2 Λειτουργία της PHP

Μια σελίδα PHP περνά από επεξεργασία από ένα συμβατό διακομιστή (ο διακομιστής η web server, εξηγήθηκε στο υποκεφάλαιο 1.1) του Παγκόσμιου Ιστού (π.χ. Apache), ώστε να παραχθεί σε πραγματικό χρόνο το τελικό HTML περιεχόμενο, που θα σταλεί στο πρόγραμμα περιήγησης των επισκεπτών σε μορφή κώδικα HTML.
Απλή απάντηση, αλλά τι σημαίνει; Ένα παράδειγμα. Έστω το εξής HTML έγγραφο.

```
<html> <head>
<title>Example</title>
</head>
<body>
<?php echo "Hi, I'm a PHP script!"; ?>
</body> </html>
```

Σχήμα 2.1

Παρατηρήστε πως αυτό είναι διαφορετικό από ένα script γραμμένο σε άλλες γλώσσες προγραμματισμού όπως η Perl ή η C : Αντί να γράφετε ένα πρόγραμμα με πολλές εντολές για να εξάγετε HTML, γράφετε ένα HTML κείμενο με κάποιο ενσωματωμένο κώδικα για να κάνει κάτι (σε αυτή την περίπτωση, να

εμφανίζει κάποιο κείμενο). Ο κώδικας PHP είναι εσώκλειστος σε ειδικά tags (ετικέτες) αρχής και τέλους που σας επιτρέπουν να μεταφέρεστε μέσα και έξω από το "PHP mode" δηλαδή τα σημεία όπου εκτελείται κώδικας PHP. Αυτό που διαχωρίζει την PHP από την JavaScript είναι ότι ο κώδικας εκτελείται στον server.

Αν είχατε ένα script σαν το παραπάνω στον server σας, και ένας client το ζητούσε, ο διακομιστής, θα εκτελούσε όλες τις PHP εντολές και θα έστελνε το αποτέλεσα –ένα html αρχείο- στον client, χωρίς να υπάρχει κανένας τρόπος να καταλάβει τι κώδικας υπάρχει από κάτω. Δηλαδή η γραμμή με τα php tags <?php … ?> θα αντικαθιστόταν από το string της εντολής echo που σημαίνει εκτύπωσε στην έξοδο.

Ένα αρχείο με κώδικα PHP θα πρέπει να έχει την κατάλληλη επέκταση (π.χ. *.php, *.php4, *.phtml κ.ά.). Η ενσωμάτωση κώδικα σε ένα αρχείο επέκτασης .html δεν θα λειτουργήσει και θα εμφανίσει στον browser τον κώδικα χωρίς καμία επεξεργασία, εκτός αν έχει γίνει η κατάλληλη ρύθμιση στα MIME types του server. Επίσης ακόμη κι όταν ένα αρχείο έχει την επέκταση .php, θα πρέπει ο server να είναι ρυθμισμένος για να επεξεργάζεται κώδικα PHP. Ο διακομιστής Apache, που χρησιμοποιείται σήμερα ευρέως σε συστήματα με τα λειτουργικά συστήματα Linux και Microsoft Windows, υποστηρίζει εξ ορισμού την εκτέλεση κώδικα PHP.

2.3 Δυνατότητες της PHP

Η PHP επικεντρώνεται κυρίως στο server-side scripting, έτσι μπορεί να κάνει ότι και ένα άλλο CGI script, όπως να συλλέγει δεδομένα, να παράγει δυναμικό περιεχόμενο σελίδων, ή να στείλει και να πάρει cookies. Αλλά η PHP μπορεί να κάνει πολύ περισσότερα. Υπάρχουν τρεις κύριοι τομείς που χρησιμοποιείται.

- Server-side scripting. Αυτό είναι το πιο παραδοσιακό και το κύριο πεδίο για την PHP. Για να δουλέψει χρειαζόμαστε τον PHP μεταγλωττιστή (parser) (CGI ή server module), έναν web server (διακομιστή σελίδων) και έναν web-browser ("φυλλομετρητή"). Πρέπει να τρέξει ο web-server, με μια

συνδεδεμένη εγκατάσταση της PHP, ενώ τα αποτελέσματα του PHP προγράμματος μπορούν να προσπελαστούν με τη χρήση ενός web-browser, βλέποντας την σελίδα PHP μέσα από τον server.

- Command line scripting. Μπορεί να φτιαχτεί ένα PHP script για να τρέχει χωρίς server ή browser. Αρκεί μόνο ο PHP μεταγλωττιστής για να την χρησιμοποιήσει με αυτό τον τρόπο. Αυτός ο τύπος είναι ιδανικός για script που εκτελούνται συχνά με τη χρήση της cron (σε Unix ή Linux) ή με τον Task Scheduler (στα Windows). Μπορούν επίσης να χρησιμοποιηθούν για απλές εργασίες επεξεργασίες κειμένου.
- Εγγραφή client-side GUI εφαρμογών (Γραφικά περιβάλλοντα χρηστών). Η PHP ίσως να μην είναι η πιο καλή γλώσσα για να γράψει κανείς παραθυρικές εφαρμογές, αλλά για αυτούς που ξέρουν PHP πολύ καλά και θέλουν να χρησιμοποιήσουν κάποια προχωρημένα χαρακτηριστικά της PHP στις client side εφαρμογές τους, μπορούν επίσης να χρησιμοποιήσουν το PHP-GTK για αυτού του είδους τα προγράμματα. Υπάρχει επίσης η δυνατότητα να γραφτούν cross-platform εφαρμογές με αυτό τον τρόπο. Το PHP-GTK είναι μια επέκταση της PHP και δεν συμπεριλαμβάνεται στην κύρια διανομή.

Η PHP μπορεί να χρησιμοποιηθεί σε όλα τα κύρια λειτουργικά συστήματα, συμπεριλαμβανομένου του Linux, πολλών εκδοχών του Unix (HP-UX, Solaris και OpenBSD), Microsoft Windows, Mac OS X, RISC OS και πιθανώς σε άλλα. Η PHP υποστηρίζεται επίσης από τους Apache, NginX, Microsoft Internet Information Server, Personal Web Server, Netscape και iPlanet servers, Oreilly Website Pro server, Caudium, Xitami, OmniHTTPd, και πολλούς άλλους webserver. Για την πλειοψηφία των server η PHP έχει ένα module, για τους υπόλοιπους η PHP μπορεί να λειτουργήσει μέσω CGI.

Έτσι με την PHP υπάρχει ελευθερία επιλογής για τον web server και το λειτουργικό σύστημα που θα χρησιμοποιηθούν, ενώ είναι δυνατό να χρησιμοποιηθεί αντικειμενοστραφής (object oriented) ή συναρτησιακός (procedural) προγραμματισμός ή μια ανάμειξη τους.

Η PHP δεν περιορίζεται μόνο στην εξαγωγή HTML. Οι δυνατότητες της PHP συμπεριλαμβάνουν την εξαγωγή εικόνων, αρχείων PDF, ακόμη και ταινίες Flash

παράγονται αμέσως. Μπορεί επίσης να εξαχθεί εύκολα οποιοδήποτε κείμενο όπως XHTML και οποιοδήποτε άλλο XML αρχείο. Η PHP μπορεί να δημιουργεί αυτόματα αυτά τα αρχεία και να τα αποθηκεύει στο σύστημα αρχείων, αντί να τα εκτυπώνει, αποτελώντας έτσι μια server-side cache για το δυναμικό σας περιεχόμενο.

Ένα από τα πιο δυνατά και σημαντικά χαρακτηριστικά της PHP είναι η υποστήριξη που έχει για ένα μεγάλο σύνολο βάσεων δεδομένων. Η συγγραφή μιας σελίδας που υποστηρίζει βάσεις δεδομένων είναι εξαιρετικά απλή. Υπάρχει επίσης μια αφαιρετική επέκταση DBX βάσεων δεδομένων (DBX database abstraction extension) που επιτρέπει διάφανα να χρησιμοποιηθεί οποιαδήποτε βάση δεδομένων υποστηρίζεται από αυτή την επέκταση. Επιπλέον η PHP υποστηρίζει το ODBC, το Open Database Connection standard (Ανοιχτό πρότυπο Σύνδεσης Βάσεων δεδομένων) έτσι μπορεί κάποιος να συνδεθεί σε οποιαδήποτε βάση δεδομένων που υποστηρίζει αυτό το παγκόσμιο πρότυπο.

Η PHP έχει επίσης υποστήριξη για επικοινωνία με άλλες υπηρεσίες χρησιμοποιώντας πρωτόκολλα όπως LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (στα Windows) και αμέτρητα άλλα. Μπορεί επίσης να ανοιχθεί raw network sockets και να αλληλεπιδράσει με οποιοδήποτε άλλο πρωτόκολλο. Η PHP έχει ακόμη υποστήριξη για την περίπλοκη ανταλλαγή δεδομένων WDDX, μεταξύ σχεδόν όλων των Web programming γλωσσών. Μιλώντας για δια-επικοινωνία, η PHP υποστηρίζει instantiation αντικειμένων Java και τα χρησιμοποιεί διάφανα σαν αντικείμενα PHP. Μπορεί επίσης να χρησιμοποιηθεί η CORBA επέκταση της, για την προσπέλαση remote (απομακρυσμένων) αντικείμενων.

Η PHP έχει εξαιρετικά χρήσιμα χαρακτηριστικά επεξεργασίας κειμένων, από την POSIX επέκταση ή τις Perl regular expressions μέχρι XML parsing αρχείων. Για τη μεταγλώττιση και την πρόσβαση αρχείων XML, υποστηρίζει τα πρότυπα SAX και DOM. Μπορεί να χρησιμοποιηθεί η XSLT επέκταση για να μετατροπή των XML αρχείων σε άλλες μορφές.

Τελευταίο αλλά σημαντικό, έχει πολλές άλλες ενδιαφέρουσες επεκτάσεις, τις mnoGoSearch search engine συναρτήσεις, πολλά εργαλεία συμπίεσης (gzip, bz2), μετατροπές ημερολογίου, μεταφράσεις κ.ά.

2.4 Προγραμματιστικές διευκολύνσεις

Εδώ θα παρουσιάσουμε περιληπτικά τις βασικές διευκολύνσεις που προσφέρει η PHP σε έναν προγραμματιστή.

1. Χαλαρού τύπου μεταβλητές (Loose type variables)

Όλες οι μεταβλητές στην php θεωρούνται string και μετατρέπονται αυτόματα όταν χρειάζεται (πχ σε μαθηματικές πράξεις) στον τύπο που χρειάζεται.

2. Πίνακες

Οι πίνακες στην php είναι ιδιαίτεροι. Δεν έχουν συγκεκριμένο τύπο δεδομένων, έτσι μέσα σε ένα πίνακα έχουμε τη δυνατότητα να αποθηκεύσουμε πολλές διαφορετικές πληροφορίες. Σαν αποτέλεσμα προκύπτει μια λίστα τιμών, κάθε μια από τις οποίες μπορεί να είναι συμβολοσειρά, αριθμός, ή ένας άλλος πίνακας. Η PHP υποστηρίζει δύο είδη πινάκων: *αριθμοδεικτών,* που χρησιμοποιούν αριθμούς ως κλειδιά και *συνειρμικούς,* που χρησιμοποιούν ως κλειδιά συμβολοσειρές.
Για να αναφερθούμε σε μια συγκεκριμένη τιμή ενός πίνακα γράφουμε πρώτα το όνομα του πίνακα και έπειτα το κλειδί μέσα σε αγκύλες:
Πχ. **echo $authors[0]; echo $books['BK'j;**

3. Καθολικοί Πίνακες

Καθολικές μεταβλητές ονομάζονται οι έτοιμοι πίνακες που διαθέτει η PHP και είναι προσπελάσιμοι ανα πάσα στιγμή σε οποιοδήποτε php αρχείο και είναι οι εξής: $_GET, $_POST, $_REQUEST, $_SERVER, $_ENV, $_SESSION και $_COOKIE.

Στις μεταβλητές $_SERVER και $_ENV αποθηκεύονται πληροφορίες σχετικά με τον διακομιστή. Για τις υπόλοιπες μιλάμε παρακάτω.

4. Εύκολη διαχείριση δεδομένων χρήστη

Αν σε ένα php script έχουν σταλεί δεδομένα είτε μέσω του URL (μέθοδος GET) είτε μέσω HTML φόρμας (μέθοδος POST) αυτές μπορούν να προσπελαστούν μέσα από τους πίνακες $_GET και $_POST αντίστοιχα. Αν πχ μια φόρμα περιέχει ένα στοιχείο εισόδου κειμένου με το attribute name=”username” στο script που θα σταλεί μπορούμε να προσπελάσουμε την τιμή που έχει δώσει ο χρήστης μέσω του $_POST[‘username’].

5. Διαχείρηση cookies

Μέσω των cookies δίνεται η δυνατότητα στον διακομιστή της αποθήκευσης πληροφοριών στον υπολογιστή του χρήστη. Έτσι μια τοποθεσία μπορεί να θυμάται τον χρήστη που την επισκέπτεται.

Η συνάρτηση μέσω της οποίας στέλνονται τα μπισκότα ονομάζεται **setcookie().**

πχ. **setcookie(όνομα, τιμή, λήξη);**

setcookie('user_name', 'sofia’, time()+1800);

Μέσω της καθολικής μεταβλητής $_COOKIE γίνεται η ανάκτηση του μπισκότου: **$_COOKIE['user_name'].** Το όρισμα λήξης καθορίζει τη διάρκεια ζωής του μπισκότου σε δευτερόλεπτα από την εποχή του Unix. Η παραπάνω γραμμή κώδικά καθορίζει το χρόνο λήξης του μπισκότου σε 1800 δευτερόλεπτα = 30 λεπτά.

6. Session (Περίοδος εργασίας)

Μια άλλη μέθοδος για την διάθεση των δεδομένων σε πολλές σελίδες είναι οι περίοδοι εργασίας. Με την χρήση περιόδων εργασίας τα δεδομένα δεν αποθηκεύονται στο φυλλομετρητή, αλλά στο διακομιστή.

Για να χρησιμοποιήσει μια σελίδα τις περιόδους εργασίας πρέπει να καλέσει την συνάρτηση session_start() η οποία πληροφορεί την PHP ώστε να ξεκινήσει μια νέα περίοδο εργασίας ή να προσπελάσει μια υπάρχουσα. Όταν η συνάρτηση αυτή καλείται για πρώτη φορά προσπαθεί να στείλει ένα μπισκότο στο φυλλομετρητή με όνομα PHPSESSID (το αναγνωρστικό της περιόδου εργασίας) γι' αυτό και πρέπει να καλείται πριν από την αποστολή οποιονδήποτε δεδομένων.

Έπειτα ο ορισμός και η ανάκτηση μεταβλητών της session γίνεται μέσω του πίνακα $_SESSION

Δημιουργία νέας session

```
session_start();
```

Ορισμός και ανάκτηση μεταβλητών:

```
$_SESSION['name'] = 'Sofia';
echo $_SESSION['name’];
```

Διαγραφή μιας μεταβλητής:

```
unset($_SESSION['μεταβλητή'];
```

Αφαίρεση όλων των δεδομένων της session από το διακομιστή:

```
session_destroy();
```

Session VS Cookies

- Οι sessions είναι πιο ασφαλείς, γιατί οι πληροφορίες παραμένουν αποθηκευμένες στο διακομιστή.
- Με τη χρήση περιόδων εργασίας μπορούμε να αποθηκεύσουμε περισσότερα δεδομένα.
- Οι χρήστες συνηθίζουν να απενεργοποιούν τα cookies. Οι περίοδοι εργασίας όμως μπορούν να λειτουργήσουν και χωρίς τα cookies.

6. Include (Συμπερίληψη πολλών αρχείων)

Η PHP χρησιμοποιώντας εξωτερικά αρχεία, διαιρεί τα σενάρια της τοποθεσίες σε διακριτά τμήματα. Έτσι ο κώδικας HTML αποσπάται από τον κώδικα PHP, ή ακόμη και κομμάτια του κώδικα PHP αποσπούνται από ένα σενάριο. Η συμπερίληψη των αρχείων γίνεται με την χρήση των παρακάτω συναρτήσεων: include(), include_once(), require(), require_once().

Πχ. include('database_config.php');

```
require('path/to/filename.html');
```

Οι συναρτήσεις include() και require() διαφέρουν στη λειτουργία τους μόνο όταν αποτυγχάνουν. Αν μια συνάρτηση include() δεν επιτύχει, το σενάριο θα εξακολουθήσει να εκτελείτε κανονικά και απλά στο φυλλομετρητή θα εμφανιστεί μια προειδοποίηση. Αν αποτύχει η require(), το σενάριο θα διακοπεί εμφανίζοντας ένα μήνυμα σφάλματος. Επίσης, όταν γράφουμε και το χαρακτηριστικό *_once(), τότε το εξωτερικό αρχείο συμπεριλαμβάνεται στον κώδικα μόνο μία φορά.

Κεφάλαιο 3: Η MySql

3.1 Τι είναι η MySQL

Η MySQL είναι ένα σύστημα διαχείρισης σχεσιακής βάσης δεδομένων (relational database management system - RDBMS). Επειδή είναι ανοικτού κώδικα (open source), οποιοσδήποτε μπορεί να αποκτήσει την MySQL και να την διαμορφώσει σύμφωνα με τις ανάγκες του σύμφωνα πάντα με την άδεια που την συνοδεύει. Η MySQL είναι γνωστή κυρίως για την ταχύτητα, την αξιοπιστία, και την ευελιξία που παρέχει. Οι περισσότεροι συμφωνούν ωστόσο ότι δουλεύει καλύτερα όταν διαχειρίζεται περιεχόμενο και όχι όταν εκτελεί συναλλαγές. Η MySQL αυτή τη στιγμή μπορεί να λειτουργήσει σε περιβάλλον Linux, Unix, και Windows. Υποστηρίζει ένα υποσύνολο του Ansi SQL και περιλαμβάνει πολλές επεκτάσεις.

Τι είναι η SQL;

Είναι ένα διεθνές standard που αποκαλείται Δομημένη Γλώσσα Ερωτημάτων (Structured Query Language) και ορίζει ένα ενιαίο συντακτικό για την αποθήκευση και ανάκληση δεδομένων προς και από ένα RDMBS. Οι εντολές της SQL αποκαλούνται επίσης και ερωτήματα (queries). Υποστηρίζεται από τα περισσότερα, αν όχι όλα, συστήματα διαχείρισης βάσεων δεδομένων όπως η MySQL, PostgreSQL, Oracle, Sybase και Microsoft SQL Server.

3.2 Χαρακτηριστικά της MySQL

Μερικά χαρακτηριστικά γνωρίσματα:

+ πολυνηματώδης.

+ όλη η κυκλοφορία κωδικού πρόσβασης κρυπτογραφείται.

+ όλες οι στήλες περιλαμβάνουν προκαθορισμένες τιμές.
+ έλεγχος και τροποποίηση πινάκων.
+ ψευδώνυμα πινάκων και στηλών σύμφωνα με τα πρότυπα SQL92.
+ όλες οι συνενώσεις (joins) γίνονται σε ένα πέρασμα.
+ εγγραφές σταθερού και μεταβλητού μήκους.
+ triggers και scheduled events

Διεπαφές: SQL, ODBC, C, Perl, PHP, JAVA, C++, Python, command line
Μέθοδοι πρόσβασης: B-tree στο δίσκο, hash tables στη μνήμη
Πολυχρηστικό: Ναι
Δοσοληψίες: Ναι, υποστηρίζει και foreign key constraints
Κατανεμημένο: Όχι, υπάρχει η δυνατότητα για mirroring
Γλώσσα Ερωτημάτων: SQL
Όρια: Πάνω από 32 indexes / table. Κάθε index αποτελείται από 1 έως 16 στήλες. Το μέγιστο πλάτος του index είναι 500 bytes
Ανθεκτικότητα: Ο κώδικας του B-tree είναι εξαιρετικά σταθερός, εφικτή η 24-ωρη λειτουργία

Υποστηριζόμενες Πλατφόρμες: **BSDOS, SunOS, Solaris, Linux, IRIX, AIX, OSF1, BSD/OS, FreeBSD**

3.3 Πλεονεκτήματα της MySQL

Μερικοί από τους κύριους ανταγωνιστές της MySQL είναι οι PostgreSQL, Microsoft SQL Server και Oracle. Η MySQL έχει πολλά πλεονεκτήματα, όπως χαμηλό κόστος, εύκολη διαμόρφωση και μάθηση και ο κώδικας προέλευσης είναι διαθέσιμος.

Απόδοση

Η MySQL είναι χωρίς αμφιβολία γρήγορη. Μπορείτε να δείτε την σελίδα δοκιμών http://web.mysql.com/benchmark.html . Πολλές από αυτές τις δοκιμές δείχνουν ότι η MySQL είναι αρκετά πιο γρήγορη από τον ανταγωνισμό.

Χαμηλό κόστος

Η MySQL είναι διαθέσιμη δωρεάν , με άδεια ανοικτού κώδικα (Open Source) ή με χαμηλό κόστος , αν πάρετε εμπορική άδεια, αν απαιτείται από την εφαρμογή σας.

Ευκολία Χρήσης

Οι περισσότερες μοντέρνες βάσεις δεδομένων χρησιμοποιούν SQL. Αν έχετε χρησιμοποιήσει ένα άλλο σύστημα διαχείρισης βάσεων δεδομένων δεν θα έχετε πρόβλημα να προσαρμοστείτε σε αυτό.

Μεταφερσιμότητα

Η MySQL μπορεί να χρησιμοποιηθεί σε πολλά διαφορετικά συστήματα Unix όπως επίσης και στα Microsoft Windows .

Ελεύθερος Πηγαίος Κώδικας

Όπως και με την PHP , μπορείτε να πάρετε και να τροποποιήσετε σύμφωνα με τις ανάγκες σας τον πηγαίο κώδικα της MySQL.

Κεφάλαιο 4: PHP Frameworks

Κάθε προγραμματιστής που ασχολείται καιρό με την δημιουργία εφαρμογών (web ή desktop) γρήγορα αντιλαμβάνεται ότι κάποιες τεχνικές προγραμματισμού ή και ακόμα αυτούσια κομμάτια κώδικα χρησιμοποιούνται σε όλες τις εφαρμογές με μικρές αλλαγές κάθε φορά. Ένα σύνολο από μεθόδους, συναρτήσεις ή κλάσεις μιας γλώσσας που μπορούν να χρησιμοποιηθούν αφαιρετικά, λέγονται βιβλιοθήκες. Πρόκειται για έτοιμα κομμάτια κώδικα που μπορούν να χρησιμοποιηθούν σε διαφορετικές εφαρμογές ως επιμέρους τμήματα της. Θα μπορούσαμε να πούμε ότι ένα Framework είναι μια συλλογή από βιβλιοθήκες δομημένες και οργανωμένες έτσι ώστε πολύ γρήγορα (στο Yii με μια μόνο εντολή στο τερματικό) να έχουμε έτοιμο τον βασικό σκελετό μιας εφαρμογής. Παρέχουν ακόμα μια σειρά από προγραμματιστικά εργαλεία για την εύκολη πραγματοποίηση αναγκαίων λειτουργιών μιας εφαρμογής. Για παράδειγμα στην web εφαρμογή της παρούσας πτυχιακής, δεν δακτυλογραφήθηκε ούτε ένα SQL ερώτημα. Τα δημιουργεί όλα, εσωτερικά, το Yii.

Επιγραμματικά θα μπορούσαμε να πούμε ότι ένα Framework παρέχει «τον σκελετό μιας εφαρμογής που μπορεί εύκολα να προσαρμοστεί από τον προγραμματιστή».

Ο λόγος ανάπτυξης των frameworks είναι ότι δίνουν την δυνατότητα στους προγραμματιστές και σχεδιαστές λογισμικού να διαθέτουν περισσότερο χρόνο στο να ικανοποιήσουν τις απαιτήσεις που πρέπει να πληρεί η εφαρμογή και όχι στις χαμηλού επιπέδου λειτουργίες του συστήματος της εφαρμογής.

Στο κεφάλαιο αυτό θα δούμε κάποια frameworks που χρησιμοποιούν την PHP.

4.1 Δυνατότητες

Γιατί να χρησιμοποιήσουμε κάποιο framework και όχι αποκλειστικά δικό μας κώδικα PHP για την δημιουργία μιας web εφαρμογής;
Μερικά από τα πλεονεκτήματα χρήσης κάποιου framework είναι τα εξής:

- ✓ Επιτάχυνση στην ανάπτυξη λογισμικού/ιστοσελίδων.

  Όταν έχουμε κάποια κομμάτια κώδικα έτοιμα, γλυτώνουμε τον χρόνο συγγραφής τους ο οποίος είναι τεράστιος. Επίσης δεν χρειάζεται να δημιουργήσουμε από την αρχή μεθόδους για την εκτέλεση βασικών λειτουργιών (πχ αλληλεπίδραση με την βάση δεδομένων) αφού τα τις παρέχουν τα frameworks.

- ✓ Ευκολία στην συντήρηση.

  Όσο πιο καθαρός είναι ο πηγαίος κώδικας μιας εφαρμογής τόσο πιο εύκολη είναι η συντήρηση (τροποποίηση/διόρθωση/επέκταση) του. Τα περισσότερα frameworks χρησιμοποιούν τεχνικές και μεθόδους που διαχωρίζουν σε τμήματα την εφαρμογής από άποψη λειτουργικότητας, λογικής και ρυθμίσεων. Έτσι όταν χρειαστεί να αλλάξουμε κάτι στην εφαρμογή να χρειάζεται να αλλάξουμε τον κώδικα σε ελάχιστα σημεία.

- ✓ Μεγάλη παρακαταθήκη από βιβλιοθήκες.

  Τα frameworks και ιδιαίτερα τα Open Source περιλαμβάνουν μια τεράστια παρακαταθήκη βιβλιοθηκών ή ακόμα και έτοιμων components τα οποία μπορούν να παραμετροποιηθούν για οποιοδήποτε project επιθυμούμε. Στα open source μάλιστα η κοινότητα δημιουργεί επιπλέον βιβλιοθήκες ή components τα οποία μπορούν να χρησιμοποιηθούν. Όσο μεγαλύτερη η κοινότητα τόσο περισσότερες οι 3rd party βιβλιοθήκες.

- ✓ Ασφάλεια και αξιοπιστία

  Ακριβώς επειδή η ομάδα δημιουργίας ενός framework δημιουργεί το ίδιο το framework και όχι μια ακόμα web εφαρμογή, δίνει ιδιαίτερη βαρύτητα στην ασφάλεια που έχουν οι βιβλιοθήκες του. Έχουν μηδενικά bugs ή leaks αφού

δοκιμάζονται εξονυχιστικά. Είναι σίγουρα περισσότερο δοκιμασμένο από οποιοδήποτε κομμάτι κώδικα γράψει οποιοσδήποτε μεμονωμένος προγραμματιστής αφού χρησιμοποιείται από χιλιάδες άτομα παγκοσμίως.

- ✓ Εύκολη προσθήκη επεκτάσεων της εφαρμογής.

  Όλες οι επεκτάσεις των frameworks έχουν αφαιρετική σχεδίαση κάνοντας εύκολη την προσαρμογή ή σύνδεση τους με οποιοδήποτε project. Πολλά frameworks μάλιστα επιτρέπουν την χρήση επεκτάσεων σχεδιασμένα για άλλο framework!

- ✓ Μεγάλες κοινότητες για να σε υποστηρίξουν.

  Στην Open Source κοσμοθεωρία, οι χρήστες ενός framework (η κοινότητα του) έρχονται κοντά βοηθώντας ο ένας τον άλλον, ανταλλάσοντας γνώσεις και απόψεις ή μπορεί ακόμα και να βοηθήσουν στην εξέλιξη ενός framework. Κάποιος αρχάριος μπορεί πολύ εύκολα να βρει βοήθεια για ένα θέμα που τον δυσκολεύει.

- ✓ “Πολυνηματική” Ανάπτυξη μιας εφαρμογής.

  Με το να απομονώσουν τις επιμέρους λειτουργίες μια εφαρμογής. Τα frameworks δίνουν την δυνατότητα σε μια ομάδα προγραμματιστών, να ασχοληθεί ταυτόχρονα με διαφορετικά τμήματα της. Έτσι ενώ ένα προγραμματιστής μπορεί να ασχοληθεί με την εμφάνιση της σελίδας, ένας άλλος, εντελώς ανεξάρτητος, ασχολείται με την λογική της

4.2 Δημοφιλέστερα PHP Frameworks

Θα δούμε τώρα τα γνωστότερα PHP Frameworks

1. Zend

Το **Zend Framework** είναι ένα αντικειμενοστραφές framework γραμμένο σε **PHP5**. Είναι βασισμένο στην απλότητα, με φιλική προς τις εταιρείες άδεια χρήσης. Η αφαιρετικά συνδεδεμένη αρχιτεκτονική της, επιτρέπει στους προγραμματιστές να χρησιμοποιούν στοιχεία του **Framework** μέσα σε άλλα frameworks. Το μεγαλύτερο μειονέκτημα του είναι η μεγάλη καμπύλη μάθησης.

2. Symfony

Το **Symfony** είναι ένα *PHP 5 Framework*, που παρέχει μια αρχιτεκτονική, στοιχεία και εργαλεία στους προγραμματιστές για ανάπτυξη σύνθετων εφαρμογών διαδικτύου γρήγορα. Στον επίσημο διαδικτυακό χώρο του Framework θα βρείτε επίσης και πρακτικά μαθήματα για εκμάθηση του Framework που είναι κατάλληλα και για αρχάριους.

3. CodeIgniter

Το **CodeIgniter** είναι και αυτό ακόμα ένα δημοφιλές **PHP Framework**. Έχει το πλεονέκτημα που χρησιμοποιεί ένα wiki για ποιο εύκολη πλοήγηση. Οι νέοι χρήστες μπορούν να ξεκινήσουν εύκολα διαβάζοντας το εγχειρίδιο χρήσης.

4. CakePHP

Χρησιμοποιεί κοινά μοτίβα σχεδίασης όπως είναι το MVC και ORM. Το **CakePHP** μειώνει το κόστος ανάπτυξης και βοηθάει τους προγραμματιστές να γράψουν λιγότερο κώδικα.

5. Prado

Το **Prado** για να λειτουργήσει χρειάζεται PHP 5 ή νεότερη. Αυτό το **Framework** είναι βασισμένο σε *components* και σε συμβάντα. Επίσης παρέχουν και ένα tutorial για ταχύτερη εκμάθηση. Πρόκειται για τον πρόγονο του Yii.

6. Kohana

Το **Kohana** είναι ένα PHP Framework που χρησιμοποιεί το μοντέλο ανάπτυξης MVC. Σκοπός του είναι η ασφάλεια, να είναι ελαφρύ και εύκολο στη χρήση. Το **Kohana** έχει βασιστεί στο **CodeIgniter** αλλά είναι αυστηρά PHP 5 και αντικειμενοστραφές.

7. Solar

Το **Solar Framework** είναι PHP 5 για κατασκευή ιστοσελίδων. Υποστηρίζει πλήρως τα name spaces της PHP. Επίσης έχει ενσωματωμένο localization και configuration για κάθε επίπεδο.

8. Fuse

Το Fuse είναι ένα framework με MVC για PHP. Έχει επιρροές από άλλα frameworks όπως είναι το Ruby on Rails και το CakePHP.

9. Akelos

Το Akelos PHP Framework είναι μια πλατφόρμα ανάπτυξης διαδικτυακών εφαρμογών βασισμένη και αυτή στο μοντέλο MVC.

2η Ενότητα

Το Yii framework

Κεφάλαιο 5: Γνωρίζοντας το Yii Framework

5.1 Τι είναι το Yii

Το Yii είναι ένα υψηλής απόδοσης, βασισμένο σε components (αυτόνομα μέρη μιας εφαρμογής) PHP Framework για την γρήγορη δημιουργία μεγάλης κλίμακας web εφαρμογών.

Το όνομα του (προφέρεται Yee ή Γίι) είναι ακρωνύμιο της φράσης "**Y**es **I**t **I**s!" και είναι η πιο ακριβής και περιεκτική απάντηση που δίνεται στις πρώτες και συχνότερες ερωτήσεις που κάνει κάποιος για ένα framework:

-Είναι γρήγορο;

-Είναι ασφαλές;

-Είναι επαγγελματικό;

-Είναι καλό για την εφαρμογή που θέλω να κάνω;

Yes It Is!

Επίσης, είναι παρόμοιο με τον κινέζικο χαρακτήρα "Yi" που αντιπροσωπεύει την ευκολία, την απλότητα και την ευλυγισία.

Δημιουργήθηκε από το Qiang Xue και η πρώτη του έκδοση εμφανίστηκε το 2008. Ο Qiang είναι επίσης ο ιδρυτής του Prado Framework. Για την δημιουργία του Yii πήρε τα καλύτερα κομμάτια και αρχιτεκτονικές των Prado, CakePHP, Symfony και Ruby On Rails (Framework για την γλώσσα Ruby).

5.2 Φιλοσοφία

Το Yii χρησιμοποιεί αντικειμενοστρεφή προγραμματισμό και μόνο. Για να λειτουργήσει απαιτεί τουλάχιστον την PHP 5 στην οποία έγιναν σημαντικές βελτιώσεις ως προς την αντικειμενοστρέφεια της.

Κάνει χρήση του MVC σχεδιαστικού προτύπου, το οποίο τείνει να γίνει standard μεταξύ των frameworks, το οποίο εξηγείται στο αμέσως επόμενο κεφάλαιο.

Σήμερα, σχεδόν κάθε web εφαρμογή υποστηρίζεται από τουλάχιστον μια βάση δεδομένων. Άρα το πώς ένα framework διαχειρίζεται την αλληλεπίδρασή

με τις βάσεις δεδομένων είναι ζωτικής σημασίας για την εφαρμογή. Το Yii συνεργάζεται με τις βάσεις δεδομένων με διάφορους τρόπους. Ο standard είναι μέσω Object Mapping Mappin (ORM) με το πρότυπο των Active Record (AR) Αυτό που κάνει η τεχνική ORM είναι να μετατρέπει δεδομένα από μια πηγή σε μια άλλη. Στην περίπτωσή μας ένα PHP αντικείμενο σε μια εγγραφή βάσης δεδομένων και το αντίστροφο. Το σχεδιαστικό πρότυπο AR (εισήχθει από τον Martin Fowler το 2003) υλοποιεί με αντικειμενοστρεφή προγραμματισμό την τεχνική αυτή.

Για την διαχείριση των συναλλαγών με την βάση σε χαμηλό επίπεδο, το Yii χρησιμοποιεί τα PHP Data Objects (PDO, διαθέσιμα από την έκδοση 5). Αυτά προσφέρουν πρόσβαση σε βάση δεδομένων με αφαιρετικό τρόπο. Δηλαδή χρησιμοποιείται ο ίδιος κώδικας ανεξάρτητα με το πιο RDBMS γίνεται η συνδιαλλαγή.

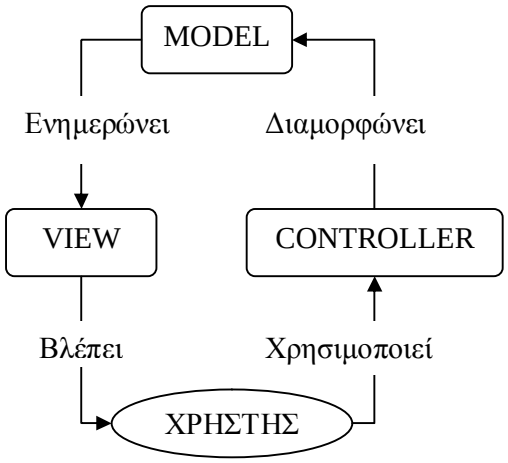
Κεφάλαιο 6: Τα Θεμελιώδη

6.1 MVC Design Pattern

Το Yii υλοποιεί το σχεδιαστικό πρότυπο **MVC** (**M**odel-**V**iew-**C**ontroller), το οποίο χρησιμοποιείται ευρέως στον δικτυακό προγραμματισμό. Το πρότυπο αυτό αποσκοπεί στο να διαχωρίσει την λογική μιας εφαρμογής από την εμφάνιση της. Ο σαφής διαχωρισμός των δύο αυτών στοιχείων μιας εφαρμογής βοηθάει τους προγραμματιστές να κάνουν ευκολότερα αλλαγές χωρίς οι αλλαγές του ενός μέρους να επηρεάζουν σε μεγάλο βαθμό το άλλο. Στο πρότυπο αυτό, το Model αναπαριστά τα δεδομένα και την λογική της εφαρμογής, η View περιέχει όλα τα γραφικά στοιχεία της διεπαφής του χρήστη και ο Controller είναι το στοιχεία που διαχειρίζεται την επικοινωνία μεταξύ Model και View μέσω αυτού ζητάμε:

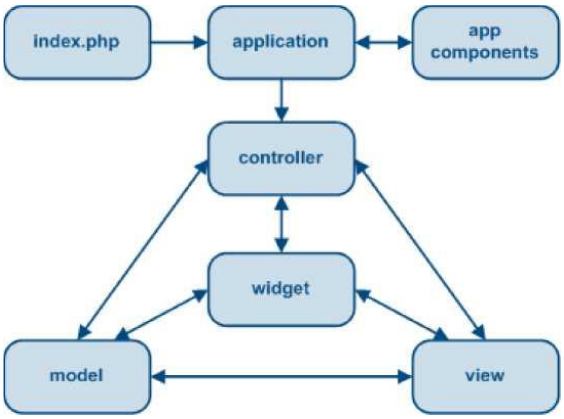
1. Από το Model, ποια στοιχεία θέλουμε να επεξεργαστούμε ή να εμφανιστούν.
2. Από το View, πως αυτά θα παρουσιαστούν στον χρήστη.

Το παρακάτω σχεδιάγραμμα βοηθάει να δούμε τις αλληλεπιδράσεις μεταξύ Model, View και Controller.



Σχεδιάγραμμα 6.1: Αλληλεπιδράσεις MVC

Εκτός από το MVC, το Yii χρησιμοποιεί έναν επιπλέον «πρωτεύοντα» controller, τον Application, ο οποίος αναλαμβάνει να μεταβιβάσει τις αιτήσεις του χρήστη, στον κατάλληλο controller της εφαρμογής ή κάποιου αυτόνομου component της. Στο σχεδιάγραμμα φαίνεται η δομή μιας εφαρμογής του Yii.



Σχεδιάγραμμα 6.2: Αλληλεπιδράσεις των οντοτήτων στο Yii

Ας δούμε λίγο αναλυτικότερα τις παραπάνω οντότητες.

6.1.1 Application

Το αντικείμενο Application αποτελεί τον κορμό μιας εφαρμογής. Η κύρια δουλειά του είναι να συλλέξει βασικές πληροφορίες για το προς εκτέλεση αίτημα και να το μεταβιβάσει στον κατάλληλο controller για ολοκλήρωση. Επίσης σε αυτό είναι αποθηκευμένες οι γενικές ρυθμίσεις τις εφαρμογής και για αυτό τον λόγο ονομάζεται και *front-controller.*

Το script εκκίνησης δημιουργεί ένα μοναδικό στιγμιότυπο Application στο οποίο μπορούμε να έχουμε πρόσβαση μέσω της αναφοράς Yii::app().

Εξ ορισμού, το Application είναι ένα στιγμιότυπο της κλάσης συστήματος *CWebApplication*. Για να τροποποιήσουμε τις ρυθμίσεις του (που αντιπροσωπεύουν τις γενικές ρυθμίσεις της εφαρμογής) υπάρχουν 2 τρόποι. Είτε περνώντας σαν παράμετρο στον κατασκευαστή του αντικειμένου ένα αρχείο ρυθμίσεων, είτε επεκτείνοντας με κληρονομικότητα την κλάση CWebApplication. Το αρχείο ρυθμίσεων δεν είναι τίποτα άλλο παρά ένας php array με κλειδί το όνομα μιας ρύθμισης και value την τιμή της. Για παράδειγμα ο παρακάτω array ορίζει το όνομα και τον προεπιλεγμένο controller της εφαρμογής

```
array(
   'name'=>'Yii Framework',
   'defaultController'=>'site',
)
```

Ο συνηθέστερος τρόπος για να περνάμε τις ρυθμίσεις στο Application είναι να αποθηκεύουμε τον πίνακα σε ένα ξεχωριστό αρχείο, το οποίο περνάμε σαν όρισμα στον κατασκευαστή του Application. Δηλαδή:

Αποθηκεύουμε το αρχείο protected/config/main.php με τον κώδικα

```
return array(...);
```

και στο index.php δίνουμε:

```
$app=Yii::createWebApplication($configFile);
```

Η λειτουργικότητα του αντικειμένου application μπορεί εύκολα να τροποποιηθεί και να εμπλουτιστεί χάρη στην ευέλικτη αρχιτεκτονική των components που υιοθετεί το Yii. Το αντικείμενο χειρίζεται ένα σύνολο από διάφορα component καθένα από το οποίο υλοποιεί ξεχωριστές λειτουργίες. Για παράδειγμα, η αρχική διαδικασία ανάλυσης ενός αιτήματος γίνεται με την βοήθεια των componenets *CUrlManager* και *CHttpRequest*.

Η ενεργοποίηση των επιμέρους components και η διαμόρφωση τους, γίνεται μέσω τις ιδιότητας *components* του αντικειμένου application. Για παράδειγμα, μπορούμε να διαμορφώσουμε το component *CMemCache* έτσι ώστε να χρησιμοποιεί πολλαπλούς servers για caching με τον εξής κώδικα:

```
array(
 'components'=>array(
  'cache'=>array(
   'class'=>'CMemCache',
   'servers'=>array(
     array('host'=>'server1','port'=>11211,'weight'=>60),
     array('host'=>'server2','port'=>11211,'weight'=>40),
```

```
        ),
    ),
),
)
```

Στο παραπάνω κώδικα, προσθέσαμε το στοιχείο *cache* στον array *components*, του οποίου η κλάση είναι η *CMemCache* και η ιδιότητα servers θα αρχικοποιηθεί όπως φαίνεται. Προγραμματιστικά, η πρόσβαση στο παραπάνω component (και αντίστοιχα στο κάθε component) γίνεται με την αναφορά *Yii:app()->cache* όπου το *app()* είναι η αναφορά στο αντικείμενο application.

Το Yii έχει ορίσει ενα σύνολο βασικών components τα οποία εκτελούν κοινές λειτουργίες για κάθε εφαρμογή στο web. Ακολουθεί μια λίστα των components αυτών (σε παρένθεση το όνομα της κλάσης συστήματος που τις περιγράφει).

- assetManager (CAssetManager):
- authManager(CAuthManager): διαχειρίζεται την πρόσβαση με βάση τους ρόλους των χρηστών (πχ admin,moderator,author).
- cache (CCache): υπεύθυνο για την χρήση των διάφορων cache.
- clientScript (CClientScript): χειρίζειται τα client scripts (javascript και css).
- coreMessages (CPhpMessageSource): παρέχει μεταφρασμένα μηνύματα του Yii Framework.
- db (CDbConnection): παρέχει την σύνδεση με την βάση δεδομένων. Σημειώνετε οτι πρέπει πρώτα να ρυθμίσουνε την ιδιότητα *connectionString* ετσι ωστε να μπορέσει να συνδεθεί σε μια βάση δεδομένων.
- errorHandler (CErrorHandler): χειρίζεται Errors και exceptions της PHP.
- format (CFormatter): διαμορφώνει τις τιμές των δεδομένων για την παρουσίαση τους.
- messages (CPhpMessageSource): παραχωρεί μεταφρασμένα μύνηματα τα οποία χρησιμοποιούνται απο την εφαρμογή.
- request (CHttpRequest): παραχωρεί πληροφορίες σχετικά με τα αιτήματα του χρήστη.

- securityManager (CSecurityManager): παραχωρεί υπηρεσίες σχετικές με την ασφάλεια όπως πχ hashing και κρυπτογράφηση.
- session (CHttpSession): χειρίζεται τις sessions της PHP.
- statePersister (CStatePersister):
- urlManager (CUrlManager): υπεύθυνο για την ανάλύση και δημιουργεία των URLs.
- user (CWebUser): εχει τις πληροφορίες σχετικά με την ταυτότητα του τρέχοντος χρήστη.
- themeManager (CThemeManager): υπέυθυνο για την διαχείρηση των θεμάτων της εφαρμογής.

6.1.2 Controller

Οι Controller είναι στιγμιότυπα της κλάσης *CController* ή μιας κλάσης που την κληρονομεί. Δημιουργούνται από το αντικείμενο application όταν απαιτείται από κάποιο αίτημα του χρήστη. Όταν ένας controller εκτελείται, πραγματοποιεί την action που ζητείται, η οποία ανατρέχει στα απαιτούμενα models και δημιουργεί τις κατάλληλες views. Μια action, στην απλούστερη μορφή της, είναι απλά μια μέθοδος στην κλάση του controller της οποίας το όνομα ξεκινάει με *action*.

Κάθε controller έχει μια προκαθορισμένη action. Όταν το αίτημα ενός χρήστη δεν διευκρινίζει την ενέργεια που επιθυμεί να εκτελεστεί, εκτελείται η προκαθορισμένη action. Εξ ορισμού η προκαθορισμένη ενέργεια ονομάζεται *index* το οποίο μπορεί να αλλαχτεί από την δημόσια μεταβλητή *CController::defaultAction.*

Ο παρακάτω κώδικας ορίζει έναν controller με το όνομα site, μια action *index* (την προκαθορισμένη), και μια action με το όνομα *contact*.

```
class SiteController extends CController{
   public function actionIndex(){
      // ...
   }
   public function actionContact(){
```

```
        // ...
    }
}
```

Οι Controllers και οι actions ταυτοποιούνται μέσω των ID. Το ID ενός Controller έχει την μορφή path/to/anyname, το οποίο αντιστοιχεί στο αρχείο protected/controllers/path/to/AnynameController.php και αποτελεί την κλάση του controller. Το ID μιας action είναι το όνομα μιας μεθόδου action της κλάσης κάποιου controller χωρίς το πρόθεμα *action*. Για παράδειγμα, αν η κλάση κάποιου controller περιέχει μια μέθοδο με το όνομα actionEdit, το ID της αντίστοιχης action θα είναι το *edit*.

Ο χρήστης ζητάει μια συγκεκριμένη action ενός συγκεκριμένου controller με την βοήθεια μιας μεταβλητής δρομολογητή. Ένας δρομολογητής έχει την μορφή <controllerID>/<actionID>. Για παράδειγμα το URL http://hostname/index.php?r=post/edit ζητάει την action *edit* του controller *PostController*. Υπεύθυνο για την δημιουργία του αντικειμένου του controller που απαιτείται κάθε φορά είναι το αντικείμενο Application και για η διαδικασία που ακολουθεί είναι η εξής:

- Εάν η μεταβλητή *CWebApplication::catchAllRequest* έχει οριστεί, θα δημιουργηθεί ο controller που ορίζει, και το ID που ζητάει ο χρήστης θα αγνοηθεί. Ο έλεγχος αυτός χρησιμοποιείται κυρίως για την περίπτωση που η εφαρμογή έχει τεθεί σε κατάσταση συντήρησης οπότε δεν επιτρέπεται να προσπελαστεί κανένα από τα περιεχόμενά της.
- Εάν το ID βρεθεί στην μεταβλητή *CWebApplication::controllerMap*, ο αντίστοιχος controller θα δημιουργηθεί
- Εάν το ID είναι στην μορφή path/to/anyname, υποθέτει πως το όνομα της κλάσης του controller είναι Anyname και το αρχείο βρίσκεται στο μονοπάτι protected/controllers/path/to/AnynameController.phh. Εάν το αρχείο κλάσης δεν βρεθεί, θα προκληθεί μια εξαίρεση 404 CHttpException.

6.1.3 Model

Ένα Model είναι στιγμιότυπο της CModel ή κάποιας κλάσης που την κληρονομεί. Τα Model χρησιμοποιούνται για τον χειρισμό των δεδομένων και τους κανόνες που τα διέπουν.

Ένα Model αντιπροσωπεύει ένα μοναδικό αντικείμενο δεδομένων. Μπορεί να είναι μια εγγραφή ενός πίνακα μιας βάσης δεδομένων ή τα στοιχεία μιας Html Form. Κάθε πεδίο του αντικειμένου δεδομένων αντιστοιχίζεται σε ένα attribute (χαρακτηριστικό) του Model. Τα attributes έχουν ονομαστικές ετικέτες και επικυρώνονται μέσω καθορισμένων κανόνων.

Το Yii υλοποιεί 2 είδη Models: Form Models και Active Records. Το πρώτο κληρονομεί την (CFormModel) είναι για φόρμες HTML ενώ το δεύτερο (CActiveRecord) για εγγραφές από μια βάση δεδομένων.

Τα Form Models χρησιμοποιούνται για την επεξεργασία δεδομένων που εισάγωνται από τον χρήστη τα απoία αφού αξιοποιηθούν δεν αποηκέυονται κάπου. Ένα παράδειγμα είναι τα στοιχεία που δίνει ο χρήστης κατά το login.

Τα ActiveRecord Models είναι υλοποίηση του Active Record σχεδιαστικού προτύπου το οποίο δίνει την δυνατότητα χειρισμού μιας βάσης δεδομένων με αντικειμενοστρεφή τρόπο. Κάθε αντικείμενο μιας CActiveRecord κλάσης αντιστοιχεί σε μια γραμμή ενός πίνακα βάσης δεδομένων και η κάθε στήλη της γραμμής αυτής είναι ιδιότητες του αντικειμένου αυτού.

6.1.2 View

Τα View είναι αρχεία php τα οποία αποτελούνται κυρίως από HTML κώδικα. Περιέχουν και μερικές php εντολές (κυρίως widgets) αλλά αποφεύγεται να γίνεται κάποια αλλαγή στα δεδομένα. Είναι απλά η διεπαφή του χρήστη και της εφαρμογής.

Ένα view file περνιέται σαν όρισμα στην μέθοδο render() των controllers οπότε και αποστέλλεται στον browser. Μαζί με το όνομα της view δίνουμε στα ορίσματα της render() και όσες μεταβλητές χρειάζεται η view. Πχ ένα Model με τα στοιχεία που θέλουμε να εμφανίσουμε στον χρήστη.

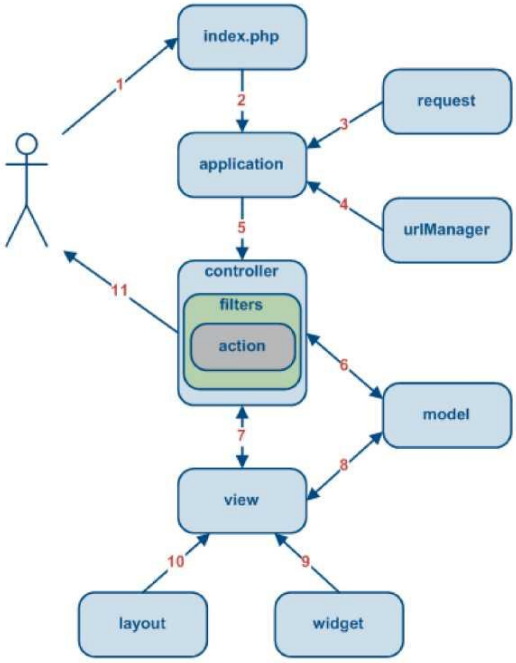
Εδώ θα εξηγήσουμε και την έννοια των Layout αρχείων. Ένα layout αρχείο περιέχει html κώδικα ο οποίος ορίζει τον σκελετό μιας σελίδας. Πχ αν θα έχει 2 στήλες ή μία. Περιέχει δηλαδή κομμάτια μιας σελίδας τα οποία είναι κοινά σε

περισσότερες από μια σελίδες τέτοια κομμάτια είναι συνήθως ο header και ο footer μιας σελίδας. Δηλαδή το πάνω (που συνήθως περιλαμβάνει ένα Logo και το menu) και το κάτω μέρος. Τα σημεία που αλλάζουν είναι μια php μεταβλητή την θέση της οποίας παίρνει ένα διαφορετικό view file κάθε φορά. Σε κάθε controller ορίζουμε ποιο θα είναι το προεπιλεγμένο layout αρχείο του.

Τέλος, τα Widgets. Πρόκειται για έτοιμα αυτόνομα κομμάτια μιας διεπαφής. Για παράδειγμα ένα ημερολόγιο ή ένας πίνακας διαχείρισης χρηστών. Τα Widgets είναι στιγμιότυπα της κλάσης CWidget. Ένα Widget μπορεί να έχει το δικό του View file στο οποίο ορίζεται πως θα παρουσιάσει τα επιμέρους στοιχεία του.

6.2 Τυπική ροή μιας εφαρμογής

Το παρακάτω διάγραμμα δείχνει την τυπική ροή που ακολουθεί μια εφαρμογή όταν χειρίζεται κάποιο αίτημα του χρήστη:



Σχεδιάγραμμα 6.3: Τυπική ροή μιας εφαρμογής στο Yii

1. Σε ένα παράδειγμα όπου ο χρήστης κάνει μια αίτηση με το URL http://www.example.com/index.php?r=post/show&id=1 και ο web server εκτελεί το scrip εκκίνησης, το index.php

2. Το script εκκίνησης δημιουργεί ένα στιγμιότυπο του αντικειμένου Application και το εκτελεί.
3. Το Application λαμβάνει αναλυτικές πληροφορίες σχετικά με το αίτημα του χρήστη από το component *request*
4. Το component *urlManager* δείχνει στην εφαρμογή ποιο controller και ποια action απαιτεί το αίτημα του χρήστη. Για το παράδειγμά μας, απαιτείται ο controller *post* (r=post στο URL) που αντιστοιχεί στην κλάση *PostController* της εφαρμογής μας, και η action είναι η show η οποία προγραμματιστικά υλοποιείται μέσα στον controller ως function.
5. Η εφαρμογή δημιουργεί ένα στιγμιότυπο του controller που ζητήθηκε, ο οποίος αναλαμβάνει την περεταίρω ολοκλήρωση του αιτήματος. Ο controller αποφασίζει ότι η action *show* αναφέρεται σε μια μέθοδο της κλάσης του με το όνομα a*ctionShow*. Πριν εκτελέσει την action, δημιουργεί και τρέχει κάποια φίλτρα που σχετίζονται με την action έτσι ώστε να αποφασίσει αν αυτή μπορεί ή πρέπει να εκτελεστεί.
6. Η action διαβάζει, από την βάση δεδομένων, ένα model με το όνομα *Post* του οποίου το ID ισούται με 1.
7. Η action δημιουργεί την εμφάνιση μιας view με το όνομα show με τα στοιχεία του model *Post*
8. Η view διαβάζει και εμφανίζει τα χαρακτηριστικά του model *Post*
9. Η view εκτελεί πιθανά widgets.
10. Το τελικό αποτέλεσμα της view ενθυλακώνεται μέσα σε ένα layout σελίδας.
11. Η action ολοκληρώνει τον οπτικό σχεδιασμό της view και εμφανίζει το αποτέλεσμα στον χρήστη

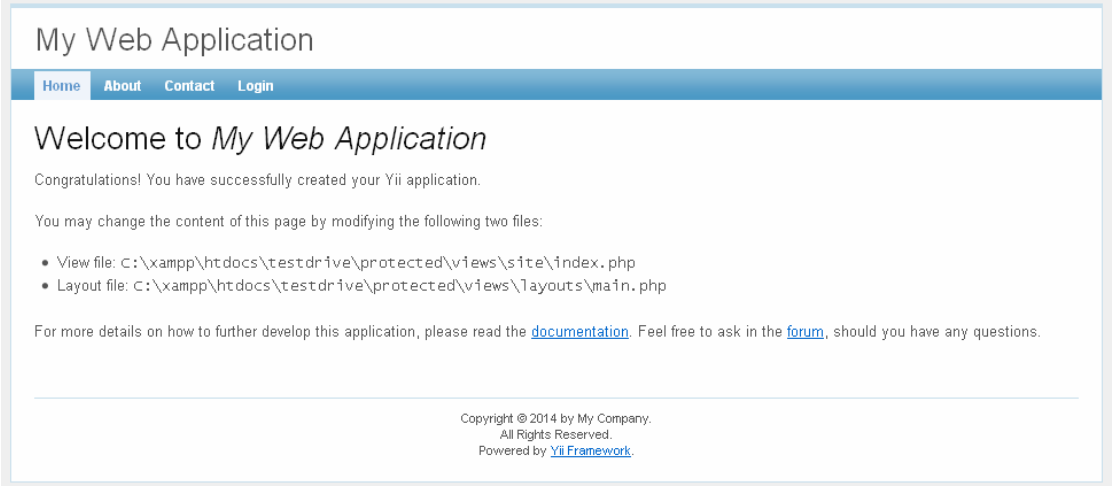
6.3 yiic

Το yiic είναι ένα εργαλείο γραμμής εντολών το οποίο έχει διάφορες χρήσεις. Εδώ θα δούμε την βασική του που είναι να δημιουργήσει τον βασικό κορμό μιας web εφαρμογής. Ας θεωρήσουμε ότι έχουμε εγκαταστήσει το yii στον φάκελο YiiRoot και ο φάκελος που είναι προσβάσιμος από επισκέπτες στον Web server μας ονομάζεται htdocs (στον ίδιο φάκελο με τον YiiRoot). Σε ένα τερματικό

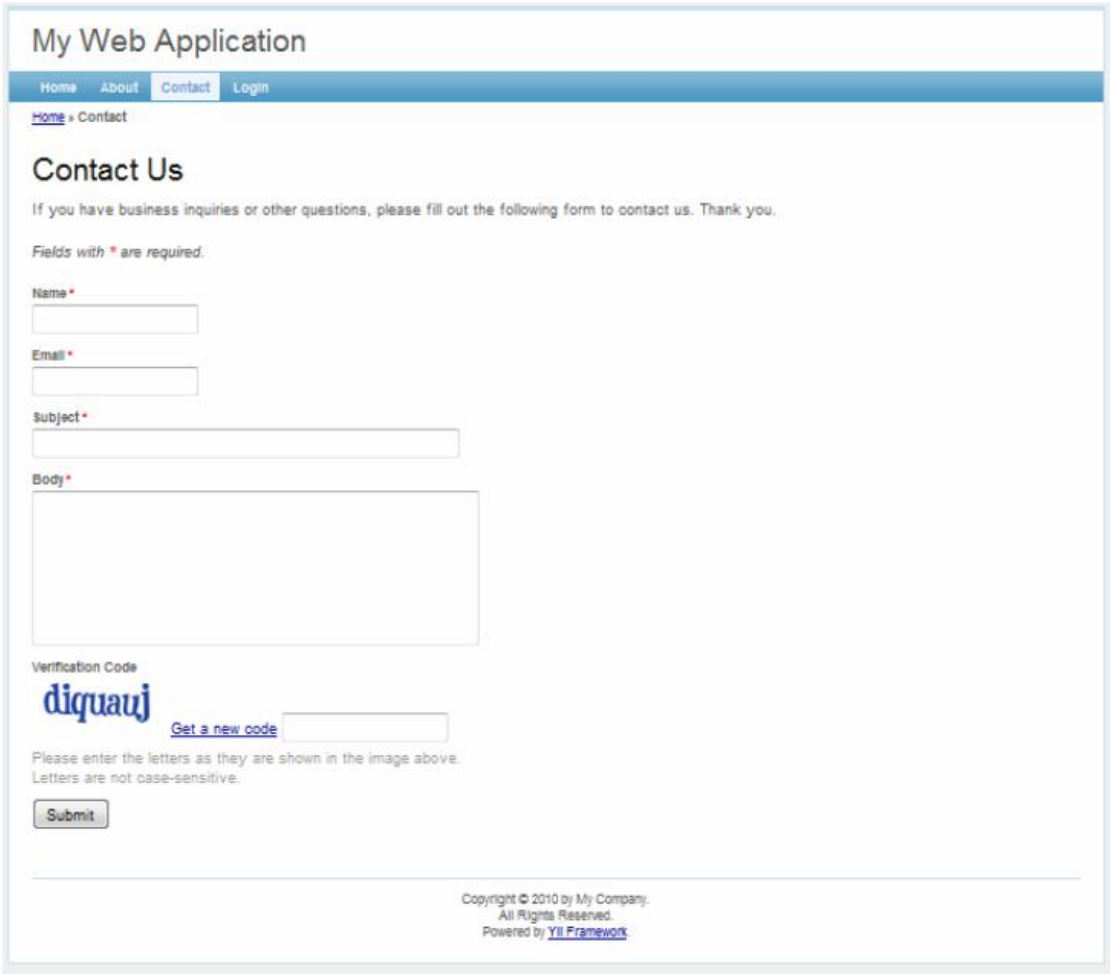
μεταβαίνουμε στον φάκελο που περιέχει τους YiiRoot και htdocs και δίνουμε την εντολή

>**YiiRoot/framework/yiic webapp htdocs/MyWebApp**

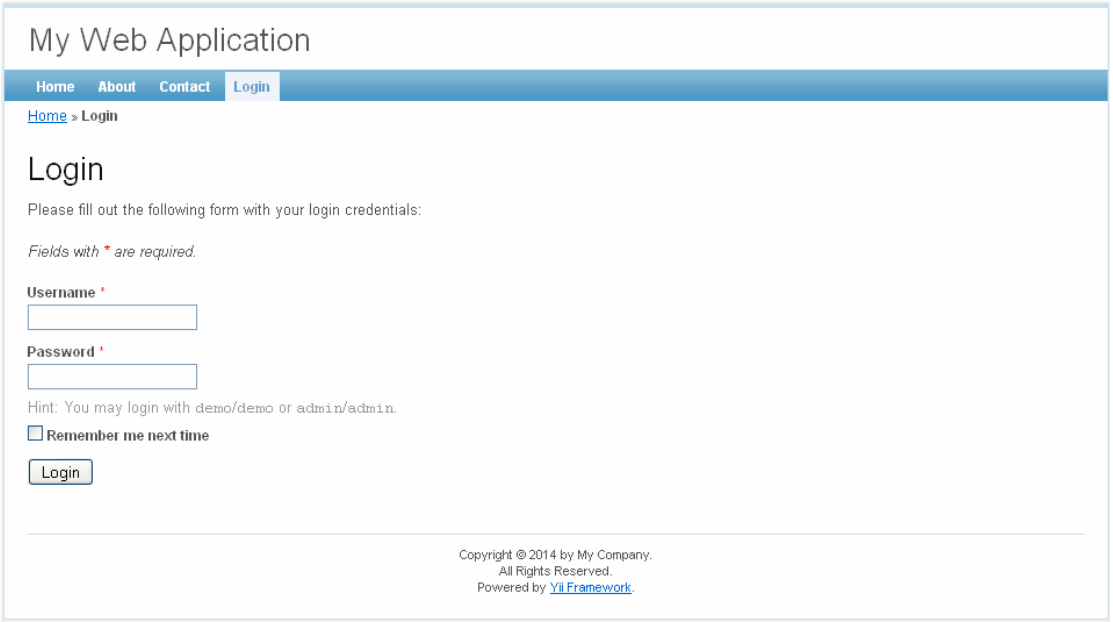
Αν τώρα μεταβούμε στην διεύθυνση http://hostname/mywebapp/index.php θα δούμε μια πλήρως λειτουργική εφαρμογή με αρχική σελίδα, σελίδες about, λειτουργική σελίδα επικοινωνίας με captcha και λειτουργία login για admin και user!



Είκονα 6.1 – Η αρχική σελίδα

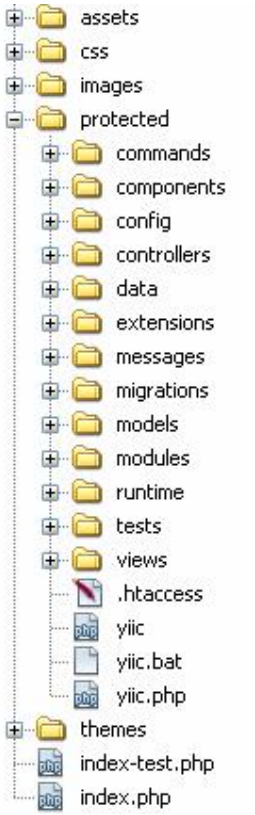


Εικόνα 6.2 – Η σελίδα επικοινωνίας



Εικόνα 6.3 – Η σελίδα login

Επίσης ο testdrive που περιλαμβάνει την εφαρμογή έχει την εξής οργανωμένη δομή:



Εικόνα 6.4 – Η δομή του φακέλου της εφαρμογής

Στον φάκελο protected βρίσκονται τα κύρια αρχεία της εφαρμογής μας και οι βασικοί υποφάκελοι είναι:

controllers: Περιέχει τις κλάσεις των controllers

models: Περιέχει όλες τις κλάσεις των models

views: Περιέχει φακέλους με τα ονόματα τον controllers και οι οποίοι περιέχουν τα views που ο καθένας χρησιμοποιεί

config: Περιέχει αρχεία ρυθμίσεων

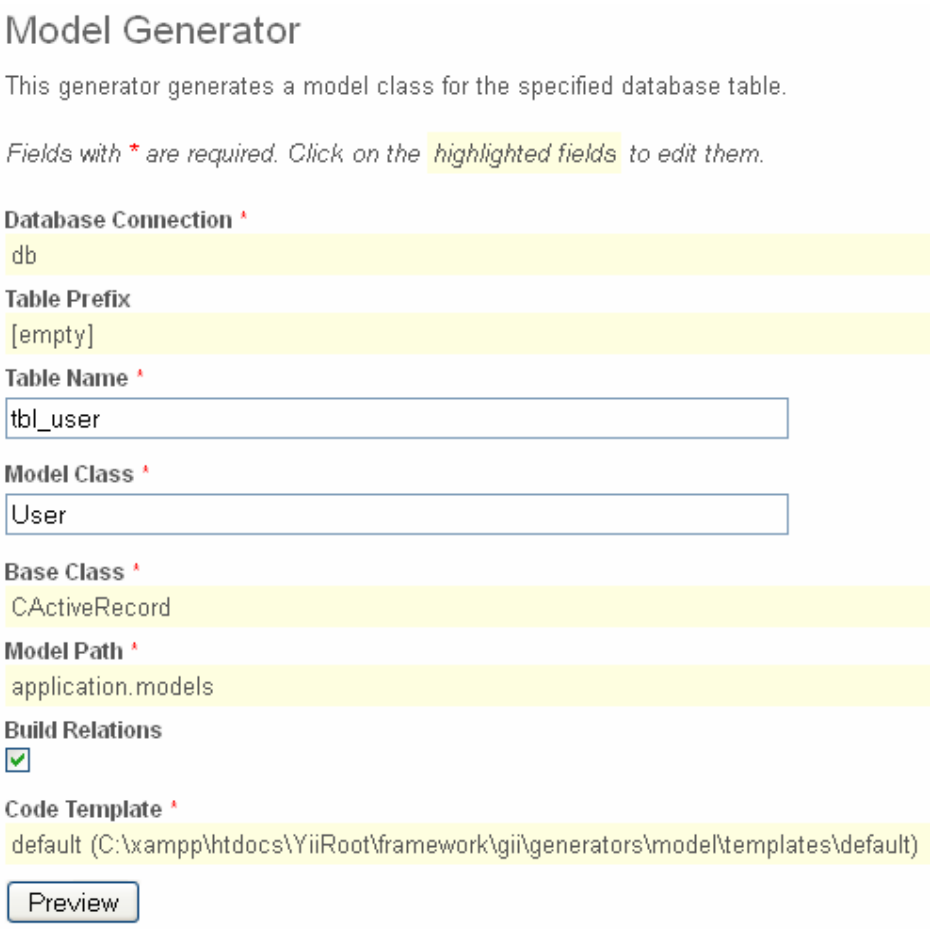
6.4 Gii

Το Yii περιέχει ένα καταπληκτικό εργαλείο, το Gii. Με αυτό μπορούμε πολύ εύκολα και γρήγορα να δημιουργήσουμε CRUD (Create, Read, Update, Delete) λειτουργίες για δεδομένα της εφαρμογής μας. Πρόκειται ουσιαστικά για μια web εφαρμογή οι οποία δημιουργεί κώδικα για την εφαρμογή μας. Ας δούμε πως χρησιμοποιείται.

Έστω ότι θέλουμε να δημιουργήσουμε ένα Model για τους χρήστες τις εφαρμογής που δημιουργήσαμε παραπάνω. Στο αρχείο protected/config/main.php (το κύριο αρχείο ρυθμίσεων) βγάζουμε τα σχόλια για το gii στον πίνακα modules έτσι ώστε να το ενεργοποιήσουμε και ορίζουμε το password που ζητείται για να έχουμε πρόσβαση στο gii. Εδώ θα πρέπει να ορίσουμε και τον τρόπο με τον οποίο η εφαρμογή θα συνδέεται στην βάση δεδομένων μας κάτι το οποίο εξηγείται σε επόμενο κεφάλαιο. Προς το παρόν ο κώδικας που δημιούργησε το yiic περιλαμβάνει μια sqlite βάση δεδομένων με έναν πίνακα tbl_user με πεδία (id, username, password, email) και μερικές εγγραφές. Έπειτα μένει να επισκεφτούμε την διεύθυνση:

http://hostname/testdrive/index.php?r=gii

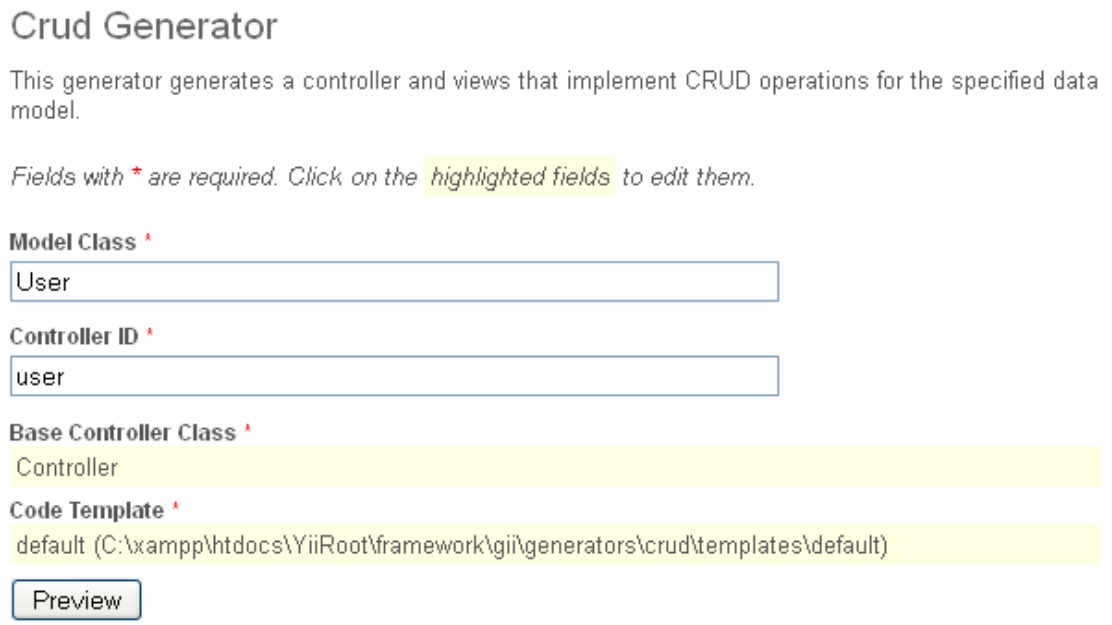
Μετά το login κάνουμε κλίκ στο μενού Model Generator και μεταβαίνουμε στην εξής σελίδα:



Εικόνα 6.5: Model Generation στο gii

Στο πεδίο Table Name δίνουμε το όνομα του πίνακα που θέλουμε να αντιστοιχεί στο Model που θα δημιουργήσουμε. Αυτόματα το Model Class παίρνει την τιμή TblUser, το οποίο αλλάζουμε σε User. Επιλέγουμε Preview και στην συνέχεια Generate και πλέον έχει δημιουργηθεί το αρχείο User.php στον φάκελο protected/models της εφαρμογής μας. Μέσω αυτής της κλάσης θα έχουμε πρόσβαση στον πίνακα tbl_user της βάση μας. Επίσης το Gii έχει δημιουργήσει και τους κανόνες για τα πεδία του πίνακα, πχ τύπος πεδίου ή μέγεθος string σύμφωνα με ότι δηλώσαμε τα πεδία στην βάση δεδομένων.

Αφού δημιουργήσαμε το model θα δημιουργήσουμε και τον κώδικα για τις CRUD λειτουργίες του. Επιλέγουμε στο Gii την επιλογή “Crud Generator” και εμφανίζεται η εξής φόρμα:



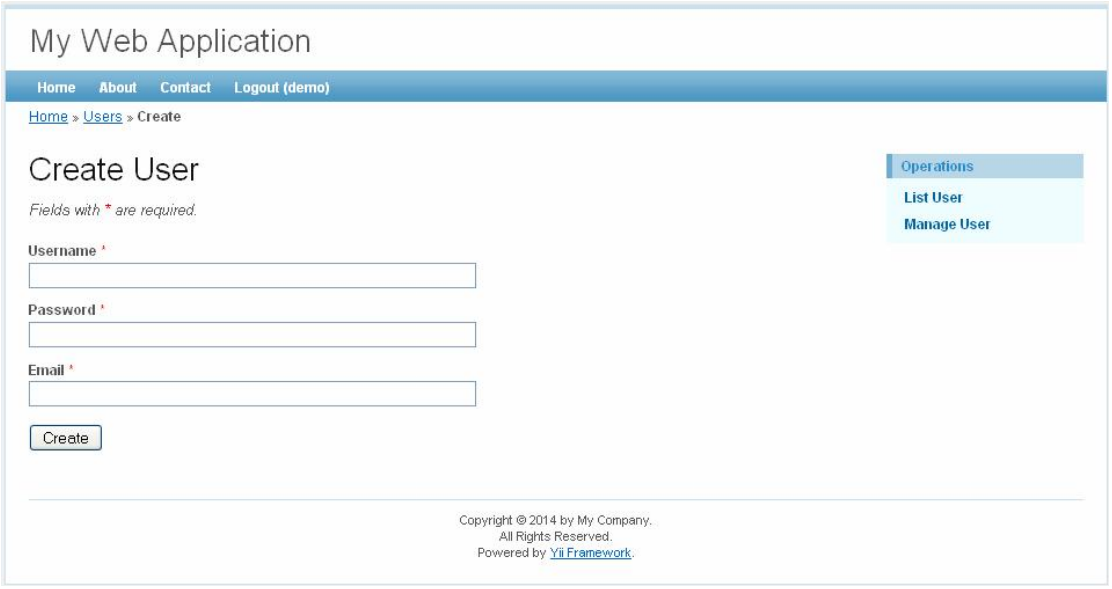
Εικόνα 6.6:CRUD Generation στο gii

Δίνουμε το όνομα της Model Class για την οποία θέλουμε να δημιουργήσουμε CRUD λειτουργία (User) και αυτόματα το πεδίο Controller ID παίρνει το ίδιο όνομα σε lower case. Επιλέγουμε πάλι Preview και έπειτα Generate.

Το Gii αυτόματα δημιούργησε την κλάση UserController για τον Controller που θα διαχειρίζεται τα δεδομένα του Model User καθώς και τα αρχεία View που απαιτούνται για την διεπαφή της εφαρμογής για τις λειτουργίες CRUD. Τώρα μπορούμε να δούμε τα αποτελέσματα αυτών που κάναμε στο URL

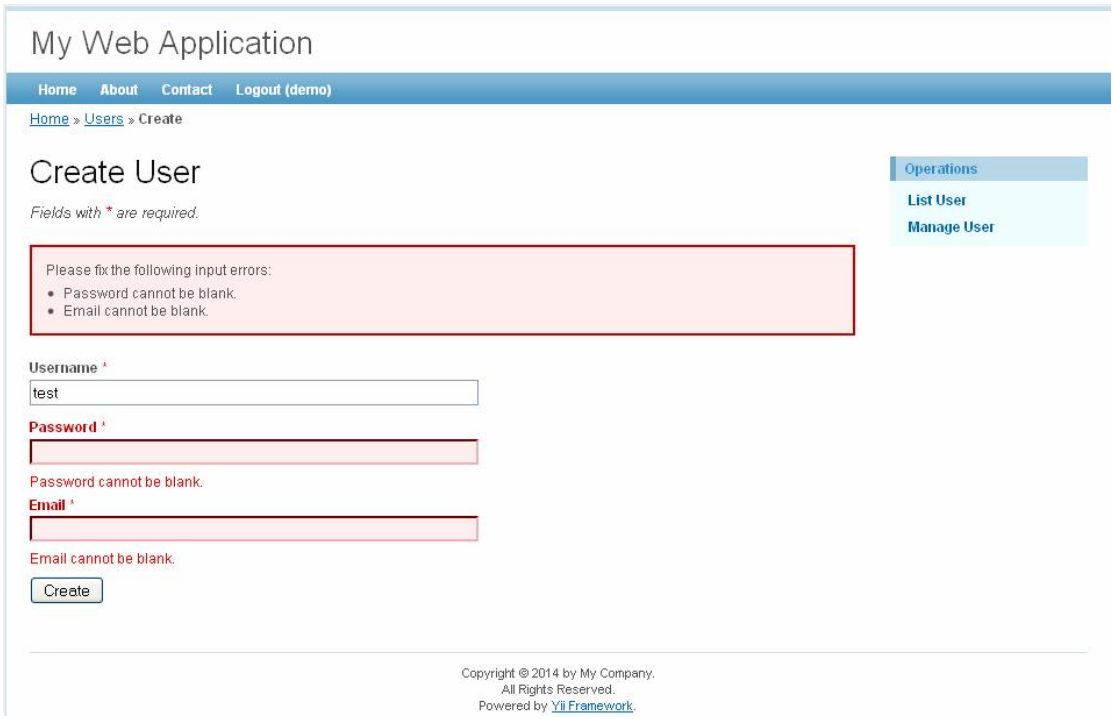
http://hostname/testdrive/index.php?r=user

Εδώ εμφανίζεται μια λίστα με όλες τις εγγραφές του πίνακα tbl_user σε περίπτωση που υπάρχουν. Επιλέγουμε Create user από το menu operations δεξιά και αν δεν έχουμε κάνει login η εφαρμογή μας ανακατευθύνει στην σελίδα login οπου μπορούμε να συνδεθούμε με username: demo και password: demo ή με admin/admin αλλιώς εμφανίζεται η εξής οθόνη:



Εικόνα 6.7:Η οθόνη δημιουργίας χρήστη

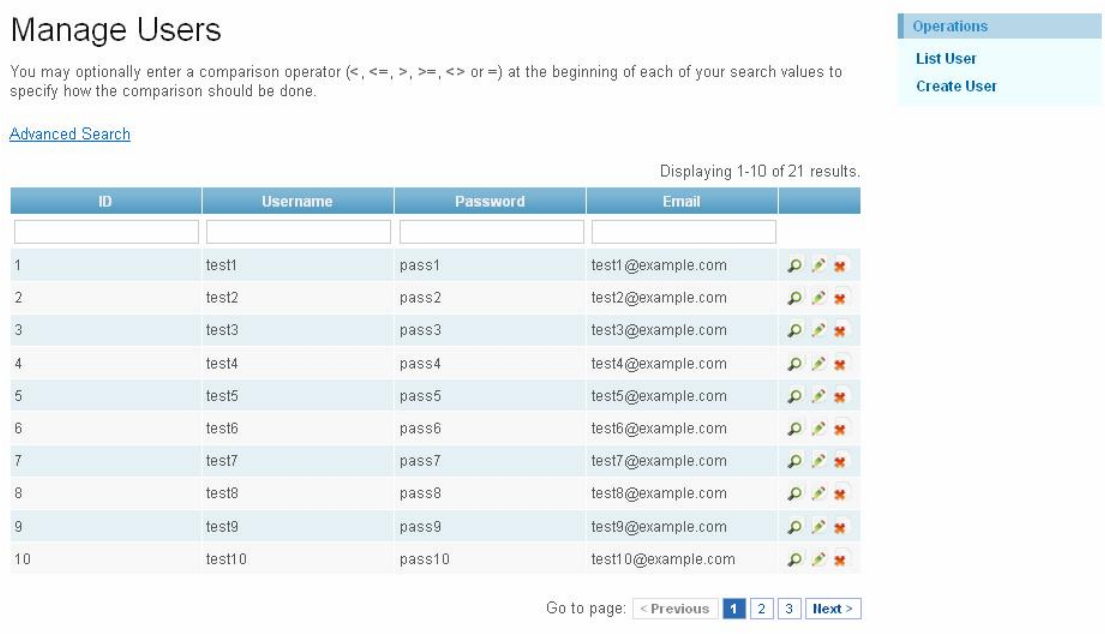
Μπορούμε να δώσουμε δοκιμαστικά τιμή μόνο στο username και να επιλέξουμε Create.



Εικόνα 6.8:Λάθος κατά την εισαγωγή στοιχείων

Ο κώδικας που έχει δημιουργήσει το Gii περιέχει και validation των στοιχείων που εισάγουμε και μας ενημερώνει για τα λάθη που κάναμε!

Τέλος αν έχουμε κάνει login ως admin μπορούμε να μεταβούμε στην επιλογή manage user (αν είμαστε logged in με demo θα μας ενημερώσει ότι δεν ήμαστε authorized να πραγματοποιήσουμε αυτή την ενέργεια). Εδώ εμφανίζονται συγκεντρωτικά όλοι οι user σε έναν εύχρηστο πίνακα που μπορούμε να αναζητήσουμε, να ενημερώσουμε ή να διαγράψουμε κάποια εγγραφή του πίνακα tbl_user.



Εικόνα 6.7: Το GridView Widget

Και όλα αυτά χωρίς να γράψουμε ούτε μια γραμμή κώδικα!!

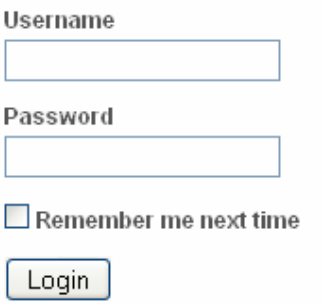
6.5 Δημιουργία φορμών

Το Yii μας παρέχει κάποιες ευκολίες για το πιο χρησιμοποιημένο HTML element σε μια web εφαρμογή, το <form>. Μέσω των forms ο χρήστης εισαγει δεδομένα στην εισαγωγή. Θα δούμε τον κώδικα της δοκιμαστικής εφαρμογής που δημιουργήσαμε για την φόρμα της σελίδας του login. Για την δημιουργία μιας φόρμας και γενικά των html οντοτήτων το Yii μας παρέχει την βοηθητική κλάση CHtml.

Στον παρακάτω κώδικα υποτίθεται ότι έχουμε δημιουργήσει ένα model (τύπου CFormModel) με το όνομα LoginForm του οποίου η μεταβλητή $model είναι ένα στιγμιότυπο. Στο model αυτό έχουμε ορίσει τις ιδιότητες username,password και rememberMe.

```
<div class="form">
<?php echo CHtml::beginForm(); ?>
  <?php echo CHtml::errorSummary($model); ?>
  <div class="row">
   <?php echo CHtml::activeLabel($model,'username');?>
   <?php echo CHtml::activeTextField($model,'username');?>
```

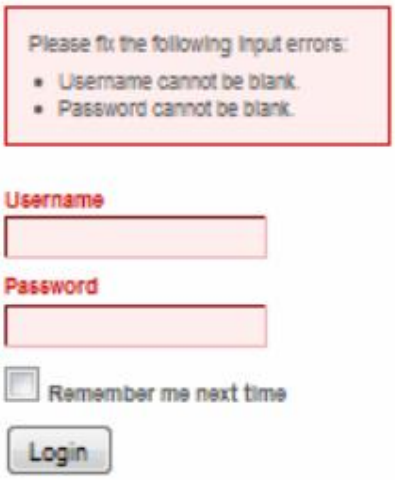
```
  </div>
  <div class="row">
   <?php echo CHtml::activeLabel($model,'password');?>
   <?php echo CHtml::activePasswordField(
                          $model,'password');?>
  </div>
  <div class="row rememberMe">
   <?php echo CHtml::activeCheckBox($model,'rememberMe');?>
   <?php echo CHtml::activeLabel($model,'rememberMe'); ?>
  </div>
  <div class="row submit">
   <?php echo CHtml::submitButton('Login'); ?>
  </div>
<?php echo CHtml::endForm(); ?> </div><!-- form -->
```



Εικόνα 6.9: Η φόρμα του login

Η φόρμα που παράγεται με τον τρόπο αυτό, είναι σχετικά δυναμική. Για παράδειγμα, η εντολή CHtml::inputTextField($model,'username') παράγει ένα text input element το οποίο σχετίζεται με την ιδιότητα username του αντικειμένου $model. Έτσι όταν η φόρμα υποβληθεί το περιεχόμενο του θα ελεγχθεί σύμφωνα με τους κανόνες που έχουμε ορίσει για αυτό στην κλάση του $model. Και μάλιστα με τα εξ' ορισμού CSS θα χρωματιστεί κόκκινο επισημαίνοντας το λάθος. Με την εντολή CHtml::activeLabel($model,'username') από την άλλη το Yii τοποθετήσει εκεί την ετικέτα που έχουμε ορίσει για την ιδιότητα username στην κλάση του $model. Έτσι αν θέλουμε να αλλάξουμε την ετικέτα του

username το κάνουμε μια φορά στην κλάση LoginForm και αυτόματα α αλλάξει σε κάθε σελίδα που εμφανίζεται.



Εικόνα 6.8:Η φόρμα μετά από αποστολή μη αποδεκτών τιμών

Από την έκδοση 1.1.1 και μετά μπορεί να χρησιμοποιηθεί το widget CActiveForm για την δημιουργία φόρμας. Μεταξύ των πλεονεκτημάτων είναι ότι η επικύρωση των δεδομένων μπορεί να γίνει και από την μεριά του browser με javascript χωρίς την αποστολή της φόρμας στον server. Με την χρήση CActiveForm ο παραπάνω κώδικας γίνεται ως εξής:

```
<div class="form">
<?php $form=$this->beginWidget('CActiveForm'); ?>
   <?php echo $form->errorSummary $model); ?>
   <div class="row">
      <?php echo $form->label($model,'username'); ?>
      <?php echo $form->textField($model,'username'); ?>
   </div>

   <div class="row">
      <?php echo $form->label($model,'password'); ?>
      <?php echo $form->passwordField($model,'password'); ?>
   </div>

   <div class="row rememberMe">
```

```
      <?php echo $form->checkBox($model,'rememberMe'); ?>
      <?php echo $form->label($model,'rememberMe'); ?>
   </div>

   <div class="row submit">
      <?php echo CHtml::submitButton('Login'); ?>
   </div>

<?php $this->endWidget(); ?>
</div><!-- form -- >
```

Κεφάλαιο 7: Αλληλεπίδραση με βάσεις δεδομένων

Το Yii διαθέτει ισχυρή υποστήριξη για την εργασία με βάσεις δεδομένων. Πρώτα με τα Database Access Objects (DAO) τα οποία ενθυλακώνουν τα PHP Data Objects (PDO) δίνει την δυνατότητα με τον ίδιο κώδικα να «συενοούμαστε» με διαφορετικά ΣΔΒΔ. Αυτό σημαίνει ότι αν για κάποιον λόγο αλλάξουμε ΣΔΒΔ για την εφαρμογή μας, δεν θα χρειαστεί να αλλάξουμε τον κώδικα μας.

Έπειτα παρέχει τον Query Builder με τον οποίο μπορούμε να δημιουργήσουμε SQL ερωτήματα με αντικειμενοστρεφή τρόπο, μειώνοντας τον κίνδυνο των SQL injections.

Τέλος τα Active Records (AR) που υλοποιούν την τεχνική ORM οπού είδαμε παραπάνω. Με τα AR αντικείμενα αντιστοιχίζουμε έναν πίνακα μιας βάσης με μια κλάση και μια εγγραφή ενός πίνακα με ένα αντικείμενο της κλάσης αυτής. Με την τρόπο αυτό ελαχιστοποιούνται οι επαναλήψεις SQL κώδικα για τις διαδικασίες CRUD με την βάση δεδομένων.

Παρακάτω θα δούμε αναλυτικότερα τις μεθόδους αυτές.

7.1 Database Access Objects (DAO)

Τα DAO αποτελούνται κυρίως από τρεις κλάσεις:
CDbConnection: αντιπροσωπεύει μια σύνδεση με την βάση δεδομένων,
CDbCommand: αντιπροσωπεύει μια SQL δήλωση προς εκτέλεση,
CDbDataReader: αντιπροσωπεύει ένα σετ δεδομένων που αποστέλλει η βάση,

Μια σύνδεση με μια βάση δεδομένων δημιουργείται με αυτόν τον τρόπο:

```
$connection=new CDbConnection($dsn,$username,$password);
$connection->active=true;
......
$connection->active=false;     // κλείνει την σύνδεση
```

Το πρώτο όρισμα αποτελεί το Data Source Name (DSN) το οποίο χρειάζεται στα PDO και είναι ένα string που αποτελείται από το όνομα PDO driver για ένα συγκεκριμένο ΣΔΒΔ την διεύθυνση του server και το όνομα μιας βάσης

δεδομένων. Το 2ο και το 3ο όρισμα αποτελούν το username και το password ενός χρήστη της βάσης αντίστοιχα. Τα DSN για τα πιο γνωστά ΣΔΒΔ έχουν ως εξής:

- SQLite: “sqlite:/path/to/dbfile”
- MySQL: “mysql:host=localhost;dbname=testdb”
- PostgreSQL: “pgsql:host=localhost;port=5432;dbname=testdb”
- SQL Server: “mssql:host=localhost;dbname=testdb”
- Oracle: “oci:dbname=//localhost:1521/testdb”

Επειδή η CDbConnection κληρονομεί την CApplicationComponent μπορεί να δηλωθεί ως component της εφαρμογής στο αρχείο αρχικών ρυθμίσεών της (protected/config/main.php) και έτσι να είναι διαθέσιμο σε κάθε σημείο του κώδικα με την εντολή Yii::app()->db (Υποθέτοντας ότι έχουμε ονομάσει το component “db”)

Αφού έχει ρυθμιστεί το CDbConnection αντικείμενο μπορούμε μέσω της μεθόδου του CDbConnection::createCommand() να δημιουργήσουμε ένα στιγμιότυπο της CDbCommand παίρνοντας στην μέθοδο σαν όρισμα μια SQL δήλωση.

```
$command= Yii::app()->db->createCommand($sql);
```

Η δήλωση αυτή τώρα μπορεί να εκτελεστεί με 2 μεθόδους:

1. $command->execute(): για δηλώσεις που δεν επιστρέφουν δεδομένα (INSERT, UPDATE κτλ)
2. $command->query(): για δηλώσεις που επιστρέφουν δεδομένα (όπως η SELECT), οπότε και επιστρέφεται ένα CDbReader αντικείμενο μέσω του οποίο μπορούμε να διαβούμε τις εγγραφές που επεστράφησαν.

Κάθε κλήση της CDbReader::read() επιστρέφει την επόμενη εγγραφή. Μπορούμε να διαβάσουμε τις εγγραφές με τους εξής τρόπους:

```
// καλώντας την read() επανειλημμένα μέχρι να επιστρέψει
false
while(($row=$dataReader->read())!==false) { ... }
// με την χρήση της  foreach της php
```

```
foreach($dataReader as $row) { ... }
// μπορούμε επίσης να ανακτήσουμε όλες τις εγγραφές σε έναν
array
$rows=$dataReader->readAll() ;
```

7.2 Query Builder

Ο QueryBuilder του Yii μας δίνει την δυνατότητα να δημιουργήσουμε SQL ερωτήματα με αντικειμενοστρεφή τρόπο Με την χρήση μεθόδων και ιδιοτήτων ορίζουμε επιμέρους στοιχεία ενός ερωτήματος κι έπειτα ο QB συναρμολογεί ένα SQL ερώτημα το οποίο έπειτα χειριζόμαστε όπως παραπάνω (μέσω DAO). Ας δούμε τον κώδικα δημιουργίας ενός απλού SELECT ερωτήματος:

```
$user=Yii::app()->db->createCommand()
                    ->select('id,username,profile')
                    ->from('tbl_user u')
                    ->join('tbl_profile p',
                                'u.id=p.user_id')
                    ->where('id=:id',array(':id'=>$id))
```

Αυτό που γίνεται στην μέθοδο where λέγεται parameter binding και βοηθά στην αποφυγή SQL injections. Μέσα στο ερώτημα δημιουργείται μια θέση με το όνομα :id και έπειτα λέμε ποια μεταβλητή θα πάρει αυτή τη θέση.

Ο Query Builder είναι χρήσιμος στις περιπτώσεις όπου χρειαζόμαστε να δημιουργήσουμε μια SQL δήλωση ακολουθιακά ή με βάση κάποια λογική της εφαρμογής μας. Τα κυρίως πλεονεκτήματα που μας προσφέρει ο Query Builder είναι:

- Επιτρέπει την δημιουργία πολύπλοκων SQL δηλώσεων προγραμματιστικά
- Αυτόματα βάζει σε εισαγωγικά τα ονόματα πινάκων και στηλών για να μην ταυτιστούν με δεσμευμένες λέξεις ή ειδικούς χαρακτήρες της SQL
- Μπορεί να χρησιμοποιηθεί parameter binding για αποφυγή SQL Injections

- Προσφέρει έναν βαθμό αφαίρεσης με την διασύνδεση της βάσης και απλοποιεί έτσι την αλλαγή ΣΔΒΔ.

7.3 Active Record

Όσων αφορά τις βάσεις δεδομένων, τα μεγαλύτερο μέρος του κώδικα μιας εφαρμογής, έχει να κάνει με τις λειτουργίες CRUD. Ακόμα και τις 2 παραπάνω ευκολίες ο κώδικας που αφορά τα SQL ερωτήματα επαναλαμβάνεται συχνά. Την λύση για τις επαναλήψεις αυτές το Yii τις παρέχει μέσω τον Active Records. Κάθε κλάση CActiveRecord ή κάποια που την κληρονομεί, αντιστοιχεί σε έναν πίνακα της βάσης. Κάθε αντικείμενο τέτοιων κλάσεων αντιστοιχεί σε μια γραμμή ενός πίνακα βάσης δεδομένων και η κάθε στήλη του πίνακα είναι οι ιδιότητες του αντικειμένου αυτού. Επίσης οι λειτουργίες CRUD γίνονται με απόλυτα αντικειμενοστεφή τρόπο.

Για παράδειγμα, έστω ότι στην βάση μας έχουμε τον πίνακα post(id,title,content) όπου αποθηκεύονται τα posts μιας εφαρμογής. Η Active Record κλάση που θα συνδεθεί με τον πίνακα αυτό είναι η ακόλουθη:

```
class Post extends CActiveRecord{

   public static function model($className=__CLASS__){
      return parent::model($className);}

   public function tableName(){
      return 'tbl_post';}
}
```

Όπως βλέπουμε το μόνο πράγμα που συνδέει την κλάση με την βάση είναι το όνομα του πίνακα το οποίο ορίζουμε στην μέθοδο tableName(). Οι AR κλάσης, εξ' ορισμού, υποθέτουν ότι το component της εφαρμογής με όνομα "db" είναι αυτό που επιστρέφει το απαραίτητο CDbConnection αντικείμενο για να συνδεθούν στην βάση που περιέχει τον πίνακα. Η μέθοδος model() χρειάζεται για την ανάκτηση δεδομένων από την βάση και θα εξηγηθεί αργότερα.

Αφού λοιπόν έχουμε ορίσει την αντίστοιχη κλάση, η εισαγωγή ενός post στην βάση έχει ως εξής:

```
$post=new Post;
$post->title='sample post';
$post->content='content for the sample post';
$post->create_time=time();
$post->save();
```

Τόσο απλά. Στην κλάση Post όντως δεν έχουμε ορίσει τις ιδιότητες title και content. Το Yii μέσα από τα metadata του πίνακα και με την χρήση των “μαγικών μεθόδων” __get() και __set() τις php, χειρίζεται τις ιδιότητες του αντικειμένου.
Η ανάκτηση περιεχομένων του πίνακα γίνεται μέσω μιας σειράς μεθόδων find() της CActiveRecord στην οποία αποκτάμε πρόσβαση μέσω της στατικής μεθόδου model().

```
// Βρίσκει την πρώτη εγγραφή που ικανοποιεί την ορισμένη
συνθήκη $condition
$post=Post::model()->find($condition,$params);
// Βρίσκει την εγγραφή με το συγκεκριμένο πρωτεύων κλειδί
$post=Post::model()->findByPk($postID,$condition,$params);
//Βρίσκει την εγγραφή με τα συγκεκριμένα πεδία (τιμές
στήλών)
$post=Post::model()->
     findByAttributes($attributes,$condition,$params);
// Βρίσκει την πρώτη εγγραφή που ικανοποιεί την
συγκεκριμένη sql δήλωση.
$post=Post::model()->findBySql($sql,$params);
```

Αν μια find μέθοδος βρει μια εγγραφή που να ικανοποιεί τις συνθήκες που ζητήθηκαν, θα επιστρέψει ένα στιγμιότυπο της Post του οποίου οι ιδιότητες θα έχουν τις αντίστοιχες τιμές των στηλών της εγγραφής. Αν δεν βρεί τέτοια

εγγραφή, επιστρέφει null. Για παράδειγμα θέλαμε να εμφανίσουμε τον τίτλο του post με id 10 θα δίναμε:

```
$post=Post::model()->find('postID=:postID',
                                array(':postID'=>10));
echo $post->title;
```

Αν θέλουμε παραπάνω από μία εγγραφές, αντί των μεθόδων find θα χρησιμοποιήσουμε τις μεθόδους findAll οι οποίες κατά αντιστοιχία είναι:

find -> findAll
findByPk -> findAllByPk
findByAttributes -> findAllByAttributes
findBySql -> findAllBySql

Οι μέθοδοι αυτές επιστρέφουν έναν array με τις εγγραφές που βρέθηκαν ή έναν κενό array αν δεν βρέθηκαν εγγραφές.

Αν θέλουμε να άλλάξουμε τιμή σε μια εγγραφή, βρίσκουμε την εγγραφή, τις αλλάζουμε την ιδιότητα που θέλουμε να αλλάξουμε και καλούμε την save().

```
$post=Post::model()->findByPk(10);
$post->title='new post title';
$post->save();
```

Αυτό που κάνει η save() είναι να αποθηκεύει στην βάση τις τρέχουσες τιμές των ιδιοτήτων του αντικειμένου/εγγραφή. Επίσης πριν επιχειρήσει να αποθηκεύσει τα δεδομένα στην βάση, τα επικυρώνει σύμφωνα με τους κανόνες που έχουμε ορίσει στην μέθοδο rules() της κλάσης της AR μας. Ένα παράδειγμα της rules για την Post θα μπορούσε να είναι το εξής:

```
public function rules()
     return array(
//Τα πεδία title και content είναι υποχρεωτικά
          array('title, content', 'required'),
//Το πεδίο title να είναι μεταξύ 5 και 20 χαρακτήρων
          array('title', 'length', 'min'=>5, 'max'=>20),
);
```

Τέλος αν θέλουμε να διαγράψουμε μια εγγραφή από την βάση καλούμε την μέθοδο delete(). Αν δηλαδή θέλουμε να διαγράψουμε το post με id=10

```
$post=Post::model()->findByPk(10);
$post->delete();
```

Το αντικείμενο παραμένει ανέπαφο αλλά η εγγραφή στην βάση διαγράφεται. Υπάρχουν επίσης οι μέθοδοι

```
Post::model()->deleteAll($condition,$params);
```

και

```
Post::model()->deleteByPk($pk,$condition,$params);
```

για την απευθείας διαγραφή εγγραφών χωρίς την δημιουργία αντικειμένων

7.4 Relational Active Records

Παραπάνω είδαμε πως μπορούμε να επιλέξουμε δεδομένα από έναν πίνακα με την χρήση των Active Records. Σε κάθε εφαρμογή όμως οι πίνακες συσχετίζονται μεταξύ τους. Για παράδειγμα ο πίνακας που αποθηκεύουμε τα άρθρα ενός χρήστη περιέχει ένα πεδίο με το id (συνήθως) του χρήστη που το έγραψε. Έτσι συνδέονται μεταξύ τους ο πίνακας των χρηστών με τον πίνακα των άρθρων.

Το Yii παρέχει την δυνατότητα να χειριστούμε τέτοιες σχέσεις μεταξύ των πινάκων (και άρα των Active Record κλάσεων). Οι συσχετίσεις υποστηρίζονται και από τα ΣΔΒΔ και μάλιστα υπάρχουν 3 τύποι συσχετίσεων: *"Ένα-προς-πολλά"* : Όταν η τιμή ενός πεδίου εμφανίζεται πολλές φορές στον άλλο πίνακα, *"Ένα-προς-ένα"*: Όταν η τιμή εμφανίζεται μια φορά και *"Πολλά-προς-πολλά"* όταν πολλές τιμές ενός πεδίου εμφανίζονται πολλές φορές στον άλλο πίνακα (συνήθως πρόκειται για έναν πίνακα που περιέχει ζεύγη τιμών από δυο διαφορετικούς πίνακες). Στις AR του Yii μπορούμε να ορίσουμε 4 τύπους συσχετίσεων:

- **BELONGS_TO:** Αν η σχέση μεταξύ ενός πίνακα A και ενός πίνακα B είναι ένα-προς-πολλά, τότε οι εγγραφές του B ανήκουν στον (belongs_to)

Α. (πχ πολλά **Post** ανήκουν σε έναν **User)**. Δηλώνεται στην AR του πίνακα Β.

- **HAS_MANY:** Αν η σχέση μεταξύ ενός πίνακα Α και ενός πίνακα Β είναι ένα-προς-πολλά, τότε ο Α έχει πολλές (has_many) εγγραφές του Β. (πχ Ένας **User** έχει πολλά **Post)**. Στην ουσία είναι η αντίστροφη της BELONGS_TO και δηλώνεται στην AR του πίνακα Α.
- **HAS_ONE:** Πρόκειται για ειδική περίπτωση της συσχέτισης HAS_MANY όπου ο Α έχει το πολύ μια εγγραφή (HAS_ONE) στον πίνακα Β. (πχ Ένας User έχει το πολύ ένα **Profile)**.
- **MANY_MANY:** Η συσχέτιση αυτή αντιστοιχεί στην συσχέτιση πολλά-προς-πολλά των ΣΔΒΔ. Για την υλοποίηση αυτής της συσχέτισης στα περισσότερα ΣΔΒΔ απαιτείται ένα ξεχωριστός πίνακας ο οποίος έχει από μια ένα-προς-πολλά σχέση με κάθε πίνακα που συσχετίζεται. Για παράδειγμα, ένα πίνακας **"**post_category(post_id,category_id)**"** όπου μας λέει σε ποια κατηγορία ανήκει ένα post, συσχετίζεται με ένα-προς-πολλά με τον πίνακα **post** και επίσης με ένα-προς-πολλά με τον πίνακα **category**. Στην ορολοία των AR, η σχέση εξηγείται σαν ένα συνδιασμός των σχέσεων **BELONGS_TO** και **HAS_MANY.** Δηλαδή, ένα **Post** Ανήκει σε πολλές **Category** και μια **Category** έχει πολλά **Post.**

Η δήλωση των σχέσεων γίνεται στην μέθοδο **relations()** των CActiveRecord η οποία επιστρέφει έναν array με τις ρυθμίσεις των σχέσεων. Ας δούμε τις μεθόδους των κλάσεων Post και User με τις μεταξύ τους σχέσεις σε μια εφαρμογή Blog.

```
class Post extends CActiveRecord{
  public function relations(){
    return array(
      'author'=>array(self::BELONGS_TO,'User','author_id'),
      'categories'=>array(self::MANY_MANY,'Category',
              'post_category(post_id, category_id)'),
   );}}
```

```
class User extends CActiveRecord{
  public function relations(){
    return array(
      'posts'=>array(self::HAS_MANY,'Post','author_id'),
      'profile'=>array(self::HAS_ONE,'Profile','owner_id'),
   );}}
```

Αφού έχουμε ορίσει τις σχέσεις των 2 αυτών κλάσεων. Μπορούμε να αναφερούμε σε αυτές με 2 τρόπους. Lazy ή eager loading. Με τον lazy loading κάνουμε το εξής:

```
//Ανακτούμε το post με id=10
$post=Post::model()->findByPk(10);
// ανακτούμε τον συγγραφέα του post αυτού μέσω της σχέσης
που έχουμε δηλώσει στην Post.
$author=$post->author;
// η $author πλέον περιέχει ένα αντικείμενο τύπου User
```

Αυτό όμως δεν είναι και τόσο αποδοτικό καθώς σε κάθε αναφορά στην σχέση γίνεται ένα SQL ερώτημα στην βάση δεδομένων. Ενώ με το eager loading το ερώτημα που ζητάει εξαρχής το Post περιέχει και τα πεδία της σχέσης. Ο τρόπος για eager loading είναι ο εξής:

```
$post=Post::model()->with('author')->findByPk(10);
// πλέον έχομε πρόσβαση στο αντικείμενο User ως εξής
$author_name=$post->author->name;
```

7.5 Named Scopes

Μια ιδέα από το Ruby on Rails framework, είναι η ιδέα των named scopes. Πρόκειται για ένα προκαθορισμένο sql ερώτημα το οποίο χρησιμοποιούμε για να φιλτράρουμε την αναζήτηση μας. Ένα παράδειγμα θα ξεκαθαρίσει την χρήση τους. Τα scopes δηλώνονται στην μέθοδο scopes() των AR models.

```
class Post extends CActiveRecord{
   public function scopes(){
     return array(
       'published'=>array(
```

```
                'condition'=>'status=1',
            ),
        'recently'=>array(
                'order'=>'create_time DESC',
                'limit'=>5,
            ),
    );}}
```

Δηλώσαμε λοιπόν δύο scopes με ονόματα *published* που ζητάει posts με status=1 και *recently* η οποία αφού ταξινομήσει με χρονική σειρά τα post ζητάει τα 5 πιο πρόσφατα. Οπότε τώρα αν θέλουμε να πάρουμε τα τελευταία 5 δημοσιευμένα posts δίνουμε την εξής εντολή:

```
$posts=Post::model()->published()->recently()->findAll();
```

7.6 DB Migration

Όπως και με τον πηγαίο κώδικα μιας εφαρμογής, η δομή μιας βάσης δεδομένων εξελίσσεται καθώς η εφαρμογή όλο και αναπτύσσετε. Ίσως χρειαστεί κατά την ανάπτυξη ένας επιπλέον πίνακας, ή ακόμα και όταν η εφαρμογή βγει σε λειτουργία ίσως εμφανιστεί η ανάγκη να δημιουργηθεί ένα νέο ξένο κλειδί. Είναι σημαντικό οι αλλαγές αυτές να συμβαδίζουν γιαυτό είναι κρίσιμο να καταγράφονται οι σταδιακές αλλαγές της βάσης κάτι που ονομάζεται migration. Αν η έκδοση του πηγαίου κώδικα και η έκδοση της δομής τις βάσης είναι ασυγχρόνιστα είναι προφανές ότι δημιουργείται σοβαρό πρόβλημα. Για τους λόγους αυτούς το Yii παρέχει ένα εργαλείο που διατηρεί ιστορικό του migration της βάσης, μέσω του οποίου μπορούμε να εφαρμόσουμε νεότερες εκδόσεις αλλά και να μεταβούμε σε παλιότερες.

Ας δούμε πως μπορούμε να δημιουργήσουμε ένα migration. Έστω ότι αποφασίζουμε την δημιουργία ενός νέου πίνακα για την εφαρμογή μας με το όνομα news, Για την δημιουργία migration χρησιμοποιούμε το γνωστό yiic. Για τον πίνακα news θα δώσουμε την εξής εντολή:

yiic migrate create news_table_creation

Το yiic θα δημιουργήσει στον φάκελορ protected/migration της εφαρμογής μας το αρχείο "**m<timestamp>_<migration name>.php"** εστω ότι την στιγμή που δημιουργήθηκε ο πίνακας news στην εφαρμογή μας το αρχείο ονομαζόταν m101129_185401_news_table_creation.php. Ο αρχικός κώδικας που περιέχει το αρχείο είναι ο εξής

```
class m101129_185401_news_table_creation extends
CDbMigration{
   public function up()
   {
   }

   public function down(){
      echo "m101129_185401_news_table_creation does not
      support migration down.\n"; return false;
   }

   /*
   // implement safeUp/safeDown instead if
   transaction is needed public function
   safeUp()
   {
   }
   public function safeDown()
   {
   }*/
}
```

Η μέθοδος up() θα πρέπει να περιέχει τον κώδικα για την εφαρμογή της αλλαγής ενώ η μέθοδος down() τον κώδικα για την αναίρεση της αλλαγής, Οπότε στην περίπτωση μας εισάγουμε τα εξής:

```
public function up(){
     $this->createTable('news', array(
                         'id' => 'pk',
                         'title' => 'string NOT NULL',
                         'content' => 'text',
                    ));
}

public function down(){
     $this->dropTable('news');
}
```

Αν τώρα θελήσουμε να ενημερώσουμε την βάση μας στην τελευταία εκδοση της θα δώσουμε σε ένα τερματικό την εντολή **yiic migrate** και το yiic θα μας εμφανίσει μια λίστα με όλες τις διαθέσιμες εκδόσεις τις βάσεις. Αν επιλέξουμε να τις τρέξουμε όλες το yiic θα τρέξει την μέθοδο up() της κάθε migration με την σειρά. Μπορούμε επίσης να δώσουμε και **yiic migrate up 3** οπότε και θα εφαρμοστούν μόνο οι επόμενες 3 ενημερώσεις ή ακόμα και **yiic migrate to <timestamp>** για να εφαρμόσουμε μέχρι την συγκεκριμένη migration.

Τέλος για να αναιρέσουμε αλλαγές δίνουμε **yiic migrate down [step]** όπου το [step] αντιστοιχεί στον αριθμό των αλλαγών που θέλουμε να αναιρέσουμε.

Κεφάλαιο 8: Επεκτείνοντας το Yii

Η επέκταση του Yii είναι κάτι συχνό κατά την ανάπτυξη μιας εφαρμογής. Όταν δημιουργούμε έναν controller επεκτείνουμε την κλάση CController. Ή αν δημιουργήσουμε ένα νέο Widget επεκτείνουμε την κλάση CWidget. Αν τώρα αυτός ο κώδικας είναι σχεδιασμένος ώστε να επαναχρησιμοποιηθεί από τρίτους, αποκαλείται *extension* (επέκταση). Στο Yii μπορούμε να ταξινομήσουμε τα extensions στις κατηγορίες:

- application component
- behavior
- widget
- controller
- action
- filter
- console command
- validator
- helper
- module

8.1 Χρήση Επεκτάσεων

Για να εισάγουμε μια επέκταση στην εφαρμογή μας συνήθως ακολουθούμε την εξής διαδικασία:

1. Κατεβάζουμε την επέκταση από την βάση επεκτάσεων του Yii (yiiframeowrk.com/extentions).
2. Αποσυμπιέζουμε το αρχείο που κατεβάσαμε στον υποφάκελο extensions/<ονομα επέκτασης> του βασικού φακέλου της εφαρμογής μας.
3. Κάνουμε import, ρυθμίζουμε και χρησιμοποιούμε την επέκταση

Κάθε επέκταση έχει ένα μοναδικό όνομα το οποίο το ξεχωρίζει από τα υπόλοιπα. Αν βρίσκεται μεσα στον φακελο extensions μπορούμε να αναφερθούμε στον δικό του φάκελο με πρόθεμα το path alias “ext” πχ

“ext.userProfile” αν ο φάκελος που περιέχει την επέκταση ονομάζεται userProfile.

Γενικά κάθε extension έχει διαφορετικές απαιτήσεις σχετικά με την εισαγωγή, ρυθμιση και χρήση του. Ας δούμε όμως περιληπτικά πώς μπορούμε συνήθως να εισάγουμε ένα extension από τις παραπάνω κατηγορίες στην εφαρμογή μας.

Application Component

Για να χρησιμοποιήσουμε ένα component εφαρμογής πρέπει πρώτα να το δηλώσουμε στον πίνακα *components* στο αρχείο ρυθμίσεων της εφαρμογής μας:

```
return array(
   // ’preload’=>array(xyz,…),
   'components'=>array(
   'xyz'=>array(
        'class'=>'ext.xyz.XyzClaa
        ss',
        'property1'=>'value1',
        'property2'=>'value2',
     ),
     // άλλες ρυθμίσεις του component
   ),
);
```

Έπειτα σε οποιοδήποτε σημείο του κώδικα της εφαρμογής μας με την εντολή Yii::app()->xyz αποκτάμε πρόσβαση σε ένα αντικείμενο του component αυτού. Αν δεν το συμπεριλάβουμε στον πίνακα preload θα δημιουργηθεί εκεί την στιγμή αλλιώς στην αρχικοποίηση της εφαρμογής μας. (Χρήσιμο όταν το χρησιμοποιούμε σπάνια ή συχνά αντίστοιχα)

Behavior

Οι behaviors είναι ένα είδος κληρονομικότητας που εισάγει το Yii. Περιέχουν απλά μεθόδους και ιδιότητες οι οποίες προσάπτονται σε όποιο αντικείμενο της

συνδέσουμε και μπορεί να της χρησιμοποιεί ως δικές του. Χρησιμοποιούνται για να εμπλουτίσουν την λειτουργικότητα ενός αντικειμένου, και βοηθάν στην μείωση του επαναλαμβανόμενου κώδικα. Η σύνδεση ενός αντικειμένου με μια behavior γίνεται ως εξής:

```
// Η $name θα αναγνωρίζει μοναδικά την behavior μέσα
στο component που την δέχεται
$component->attachBehavior($name,$behavior);
// Η behavior περιέχει την μέθοδο test()
$component->test();
```

Στους controllers και τα models οι behaviors δηλώνονται στην μέθοδο behaviors() και η σύνδεση γίνεται στην αρχικοποίηση των αντικειμένων.

```
public function behaviors(){
   return array(
      'xyz'=>array(
         'class'=>'ext.xyz.XyzBehavior',
         'property1'=>'value1',
         'property2'=>'value2',
      ),
   );}
```

Widget

Τα widgets χρησιμοποιούνται κυρίως στα αρχεία views. Αν έχουμε μια κλάση widget με το όνομα XyzClass η οποία ανήκει στο extension xyz, τότε θα το χρησιμοποιήσουμε σε μια view ως εξής:

```
// widget χωρίς επιπλέον στοιχεία
<?php $this->widget('ext.xyz.XyzClass', array
     'property1'=>'value1',
     'property2'=>'value2')); ?>

// widget με επιπλέον στοιχεία
<?php $this->beginWidget('ext.xyz.XyzClass', array(
```

```
        'property1'=>'value1',
        'property2'=>'value2')); ?>
...επιπλέον στοιχεία του widget...
<?php $this->endWidget(); ?>
```

Controller

Όπως είδαμε οι controllers περιέχουν ειδικές μεθόδους τις actions τις οποίες μπορούν να ζητήσουν οι χρήστες για εκτέλεση. Για να χρησιμοποιήσουμε έναν controller ως extention θα πρέπει στο αρχείο ρυθμίσεων τις εφαρμογής να ορίσουμε την ιδιότητα “controllerMap” ως εξής:

```
return array(
    'controllerMap'=>array(
        'xyz'=>array(
            'class'=>'ext.xyz.XyzClass
            ', 'property1'=>'value1',
            'property2'=>'value2',
        ),
        // άλλοι controllers
    ),
);
```

Actions

Οι actions είναι οι μέθοδοι που μπορεί να εκτελέσει κάποιος χρήστης μέσω των controllers. Για να χρησιμοποιηθεί μια επέκταση κλάσης action από έναν controller δηλώνεται στην μέθοδο actions() ως εξής:

```
class TestController extends CController{
    public function actions(){
        return array(
            'xyz'=>array(
```

```
            'class'=>'ext.xyz.XyzClass',
            'property1'=>'value1',
            'property2'=>'value2',
        ),
        // άλλες actions
    );}}
```

Filter

Τα filters χρησιμοποιούνται από τους controllers, προτού ή αφού εκτελεστεί μια action τους. Μεταξύ άλλων, μπορούμε να τα χρησιμοποιήσουμε για να ελέγξουμε αν αυτός που εκτελεί την action έχει δικαίωμα να το κάνει ή για να χρονομετρήσουμε μια action. Τα filters συνδέονται με τις actions ενός controller ως εξής:

```
class TestController extends CController{
   public function filters(){
      return array(
         array(
            'ext.xyz.XyzClass',
            'property1'=>'value1',
            'property2'=>'value2',
         ),
         // άλλα filters
      )}};
```

Validator

Οι validators κυρίως χρησιμοποιούνται στα models για τον έλεγχο των ιδιοτήτων τους. Ελέγχουν αν τα δεδομένα που πάνε προς αποθήκευση πληρούν τους κανόνες που ορίζουν (τύπος, μέγεθος, κτλ) Αυτοί τοποθετούνται στην μέθοδο rules των models.

```
class MyModel extends CActiveRecord{ // ή CFormModel
   public function rules(){
      return array(
         array(
```

```
            'attr1, attr2',//ιδιότητες του Model προς έλεγχο
            'ext.xyz.XyzClass',
            'property1'=>'value1',
            'property2'=>'value2',
        ),
    );}}
```

Console Command

Ένα extension εντολής κονσόλας συνήθως προσθέτει στο *yiic* μια επιπλέον εντολή. Ένα τέτοιο extension δηλώνεται στην ιδιότητα commandMap στο αρχείο ρυθμίσεων της εφαρμογής ως εξής:

```
return array(
    'commandMap'=>array(
        'xyz'=>array(
            'class'=>'ext.xyz.XyzClass'
            , 'property1'=>'value1',
            'property2'=>'value2',
        ),
        //άλλες commands
    ),
);
```

Μετά από αυτό μπορούμε να εκτελέσουμε με το yiic την εντολή **xyz**

Modules

Τα modules είναι αυτοτελή components που περιέχουν τα δικά τους models, views, controllers ακόμα και components. Από πολλές απόψεις, ένα module μοιάζει με μια εφαρμογή. Η μεγαλύτερη διαφορά είναι ότι ένα module δεν μπορεί να διατεθεί μόνο του ως εφαρμογή, αλλά πρέπει να βρίσκεται μέσα σε μία. Οι χρήστε έχουν πρόσβαση στους controllers ενός module αρκριβώς όπως σε αυτούς της εφαρμογής.

Τα modules μας χρησιμεύουν σε διάφορες περιπτώσεις. Ένα μεγάλο project μπορούμε να το χωρίσουμε σε modules όπου το καθένα αναπτύσσεται και συντηρείται χωριστά. Επίσης κάποιες κοινές λειτουργίες των εφαρμογών, όπως η διαχείριση των χρηστών, ή η διαχείριση των σχολίων ενός blog, μπορούν να γίνουν ξεχωριστά modules τα οποία να τα επαναχρησιμοποιούμαι αυτούσια σε μελλοντικά projects μας.

Για να χρησιμοποιήσουμε ένα module, εγκαθιστούμε τον φάκελο του στον φάκελο “modules” τον βασικό φάκελο της εφαρμογής μας. Έπειτα δηλώνουμε το ID του module (το όνομα του φακέλου του) την ιδιότητα *modules* της εφαρμογής μας, στο αρχείο ρυθμίσεων της. Αν θέλουμε να εγκαταστήσουμε ένα module με όνομα φακέλου “forum” το κάνουμε ως εξής:

```
return array(
   'modules'=>array('forum',...),
);
```

Αν τώρα θελήσουμε να καλέσουμε μια action ενός από τους controllers του αυτό γίνεται με το route moduleID/controllerID/actionID. Για παράδειγμα αν το παραπάνω forum module έχει έναν controller με το όνομα PostController, και την action createAction για να αποκτήσουμε πρόσβαση σε αυτήν θα ζητήσουμε το route *forum/post/create* . Δηλαδή το url που θα επισκεφτούμε θα είναι το http://www.example.com/index.php?r=forum/post/create

8.2 Προεγκατεστημένες Επεκτάσεις (Zii και jQuery UI)

Το Yii περιέχει προεγκατεστημένες κάποιες επεκτάσεις, κυρίως widget, που βοηθάνε πάρα πολύ στην ανάπτυξη του interface μιας εφαρμογής. Το set αυτό των επεκτάσεων, ονομάζεται Zii. Η πρόσβαση σε κάποιο από τα widgets του γίνεται με το προκαθορισμένο path allias “zii”. Αν δηλαδή θέλουμε να αναφερθούμε στην κλάση του GridView θα το κάνουμε ως εξης: “zii.widgets.grid.CGridView”. Ας δούμε τώρα τα βασικά widgets της βιβλιοθήκης zii.

CMenu

Η κλάση CMenu μας παρέχει έναν τρόπο για να προβάλουμε ένα μενού. Στο κώδικα της αρχικής εφαρμογής που παράγει το yiic μπορούμε να δούμε στο αρχείο main.php (που είναι layout) τον εξής κώδικα:

```
$this->widget('zii.widgets.CMenu', array (
'items'=>array(
  array('label'=>'Home','url'=>array ('/site/index')),
  array('label'=>'About',
          'url'=>array('/site/page','view'=>'about')),
  array('label'=>'Contact','url'=>array ('/site/contact')),
  array('label'=>'Login',
          'url'=>array ('/site/login'),
          'visible'=>Yii::app()->user->isGuest),
  array('label'=>'Logout ('.Yii::app()->user->name.')',
          'url'=>array ('/site/logout'),
          'visible'=>!Yii::app()->user->isGuest)
),
));
```

Κάθε item είναι και ένα στοιχείο του μενού. Η ιδιότητα label είναι το κείμενο που θα εμφανίζεται, το url είναι η διεύθυνση που θα μεταβαίνει η επιλογή του συγκεκριμένου μενού, και η ιδιότητα visible μας λέει πότε το item αυτό θα είναι εμφανές. Εδώ η επιλογή Login είναι εμφανής όταν ο τρέχω χρήστης είναι επισκέπτης, δηλαδή δεν έχει κάνει Login. Και όταν είναι συνδεδεμένος (δηλαδή δεν είναι επισκέπτης) εμφανίζεται η επιλογή Logout(<όνομα χρήστη>) ενώ η επιλογή Login, όχι.

Το CMenu μας δίνει την δυνατότητα να δημιουργούμε ένα απλοϊκό αλλά αποτελεσματικό μενού πολύ εύκολα και γρήγορα. Ο παραπάνω κώδικας με τα css του yiic εμφανίζει έτσι το μενού:

Εικόνα 8.1: Η όψη του CMenu

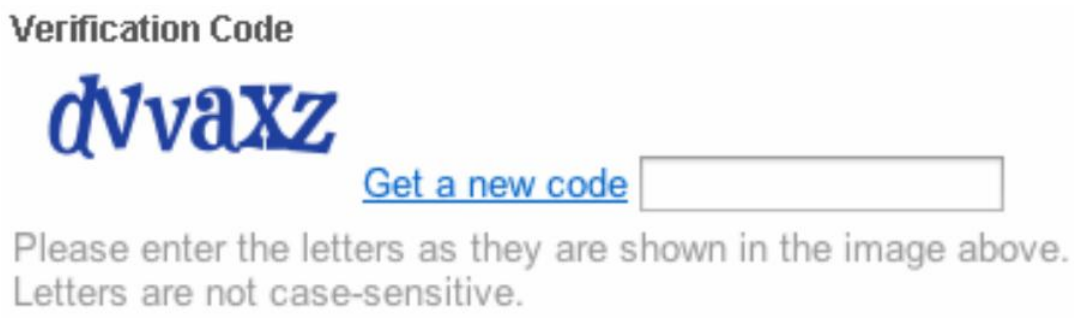
CBreadcrumbs

Τα breadcrumbs ή “ψίχουλα” δεν είναι τίποτα άλλο από το μονοπάτι σελίδων που ακολουθήσαμε για να βρεθούμε από την αρχική σελίδα της εφαρμογής, μέχρι την τρέχουσα. Συνήθως εμφανίζεται στο πάνω μέρος της οθόνης, κάπως έτσι:

Εικόνα 8.2: Η όψη του CBreadcrumbs

CCaptcha

Η γνωστή CAPTCHA τεχνική για τον αποκλεισμό των bots. Συμπεριλαμβάνεται στο τέλος κάποιας φόρμας και μας ζητάει να εισάγουμε κάποιους δυσανάγνωστους χαρακτήρες από μια εικόνα για να επιβεβαιώσουμε το ότι είμαστε άνθρωποι και όχι κάποιο λογισμικό. Η τεχνική αυτή μας βοηθάει να αποτρέψουμε σε προγράμματα να εισάγουν διαφημίσεις ή κακόβουλο κώδικα στην σελίδα μας. Το CAPTCHA του Zii φένεται έτσι:



Εικόνα 8.3: Ο CAPTCHA έλεγχος

Τα 3 επόμενα widgets που θα δούμε είναι παρόμοια μεταξύ τους γιατί χρησιμοποιούνται για παρουσίαση δεδομένων. Διαφέρει το πλήθος των δεδομένων, και οι λειτουργίες που παρέχουν. Το ιδιαίτερο τους χαρακτηριστικό είναι ότι τα δεδομένα που θα παρουσιάσουν τα αντλούν από ένα αντικείμενο τύπου DataProvider.

CListView

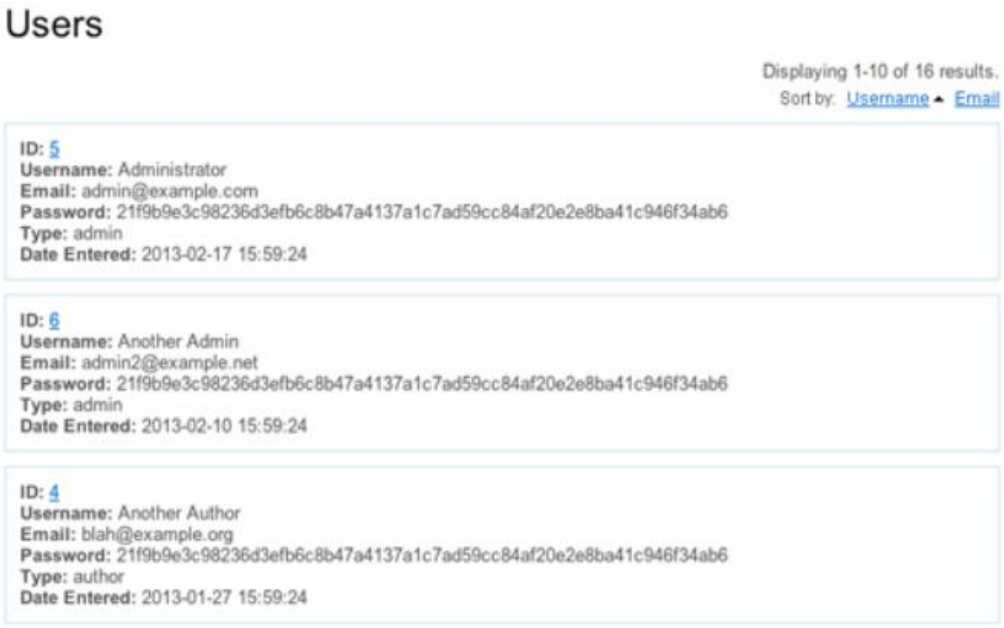
Το CListView προβάλει πολλαπλές εγγραφές δεδομένων, (models ή arrays) πχ τους χρήστες της εφαρμογής μας, σε μορφή λίστας. Μας δίνει επίσης την

δυνατότητα ταξινόμησης ανάλογα με τα κριτήρια που θα ορίσουμε, ενώ χρησιμοποιεί ένα view file στο οποίο ορίζουμε το πώς θα εμφανίζονται.

Ο κώδικας δημιουργίας ενός CListView widget είναι ο εξής:

```
<?php $this->widget('zii.widgets.CListView',
    array( 'dataProvider'=>$dataProvider,
    'itemView'=>'_view',
)); ?>
```

Και η default εμφάνιση είναι αυτή:



Εικόνα 8.4: Η όψη του CListView

Ενώ το αρχείο _view.php που ορίζει πως θα εμφανίζεται κάθε αντικείμενο του dataProvider είναι κάπως έτσι:

```
<div class="view">
<b>
<?php echo CHtml::encode($data->getAttributeLabel('id')
                          );?>: </b>
<?php echo CHtml::link(CHtml::encode( $data-
    >id),
                    array('view','id'=>$data->id)); ?>
<b> <br/>
<?php echo CHtml::encode($data-
>getAttributeLabel('username')); ?>:</b>
<?php echo CHtml::encode( $data->username); ?>
<br />
//Και  συνεχίζει  έτσι  με  τα  υπόλοιπα  στοιχεία  του  User
```

Η μεταβλητή $data αναφέρεται στο τρέχον αντικείμενο του dataProvider. Στο παραπάνω παράδειγμα ο dataProvider περιέχει αντικείμενα τύπου ActiveRecord.

CDetailView

Το CDetailView widget χρησιμοποιείται για την προβολή μιας εγγραφής δεδομένων (model ή array). Εμφανίζει τα χαρακτηριστικά της εγγραφής που εμείς ορίζουμε. Εδώ δεν χρησιμοποιείται dataProvider αφού πρόκειται για μοναδική εγγραφή.

Εδώ μπορούμε να δούμε τον κώδικα που δημιουργεί ένα CDetailView widget για ένα Page model (μια σελίδα ενός blog) ορισμένο στην μεταβλητή $model, το οποίο ορίζουμε στον δείκτη ‘data’ του array ρυθμίσεων, ενώ στον δείκτη ‘attributes’ ορίζουμε ποια attributes του $model θέλουμε να παρουσιαστούν.

```
<?php $this->widget('zii.widgets.CDetailView', array(
    'data'=>$model,
    'attributes'=>array( 'id',
        'user_id',
        'live',
        'title',
        'content',
    ))
); ?>
```

Θα εμφανιστεί αυτό:

Εικόνα 8.5: Η όψη του CDetailView

CGridView

Το CGridView είναι το widget που κυριολεκτικά μας λύνει τα χέρια κυρίως στο backend μέρος της εφαρμογής μας και κυρίως σε ότι έχει να κάνει με διαχείριση εγγραφών από τους χρήστες της εφαρμογής. Παρουσιάζει σε έναν πίνακα μια σειρά εγγραφών και παρέχει τις εξής λειτουργίες:

- ✓ Σελιδοποίηση
- ✓ Ταξινόμηση
- ✓ Δυνατότητα για προβολή, επεξεργασία ή διαγραφή κάθε εγγραφής
- ✓ Βασική αναζήτηση ανά πεδίο
- ✓ Ειδική αναζήτηση
- ✓ Άμεσα αποτελέσματα αναζήτησης με χρήση AJAX

Εικόνα 8.6: Η όψη του CGridView

Αυτός είναι ο κώδικας που δημιουργεί ένα τέτοιο widget.

```
<?php $this->widget('zii.widgets.grid.CGridView', array(
                'id'=>'user-grid',
                'dataProvider'=>$model->search(),
                'filter'=>$model,
                 'columns'=>array(
                            'id',
                            'username',
                            'email',
                            'pass',
                            'type',
                            'date_entered',
                            array('class'=>'CButtonColumn')
                ))); ?>
```

Όπως φαίνεται στο array ρυθμίσεων περνάμε 4 χαρακτηριστικά: id, dataProvider, filter και columns.

Το id είναι απαραίτητο για την αναφορά στο widget από διάφορα Javascript προγράμματα.

Το dataProvider είναι η δομή όπου υπάρχουν οι εγγραφές που θα προβληθούν. Εδώ σαν όρισμα δίνεται η μέθοδος search() του $model. Η search() επιστρέφει εγγραφές που ταιριάζουν στα κριτήρια που θέτει κάθε φορά ο χρήστης.

Στην μεταβλητή filter λέμε στο widget σε ποιο model να ψάχνει για τις τιμές που δίνει ο χρήστης στα πεδία αναζήτησης που φαίνονται στην εικόνα του widget. Αν δεν ορίσουμε αυτή την μεταβλητή τα πεδία φιλτραρίσματος δεν θα εμφανίζονται.

Τέλος στην μεταβλητή columns ορίζουμε ποιες θα είναι οι στήλες του πίνακα και με ποια σειρά (είναι η σειρά που δηλώνονται) .

sΈνα άλλο χρήσιμο πακέτο με widgets είναι αυτό των jQuery UserInterface widgets. Το yii έχει ενσωματώσει όλα τα components του jQuery UI ως widgets στο πακέτο “zii.widgets.jui”. Τα εύχρηστα αυτά widgets απαντώνται πλέον σε κάθε online εφαρμογή. Το Yii περιέχει τα εξής:

- *Accordion*
- *Autocomplete*
- *Datepicker*
- *Dialog*

- *Menu*
- *Slider*
- *Spinner*
- *Tabs*
- *Tooltips*

Ας δούμε πως μπορούμε να χρησιμοποιήσουμε το Datepicker:

Εικόνα 8.7: Το Datepicker του Jquery UI

Όταν κάνουμε κλικ σε ένα πεδίο κειμένου εμφανίζεται το παραπάνω εύχρηστο ημερολόγιο για να επιλέξουμε μια ημερομηνία. Όταν το δημιουργούμε του ορίζουμε με ποιο attribute, ποιου model θα συσχετιστεί το text input. Ο κώδικας που υλοποιεί αυτή την λειτουργία είναι ο εξής:

```
$this->widget('zii.widgets.jui.CJuiDatePicker' , array(
    'attribute'=>'date_published',
    'model' => $model
    ));
```

Κεφάλαιο 9: Ειδικότερες λειτουργίες

9.1 Caching

Το caching ή προσωρινή μνήμη, είναι ένας εύκολος και φθηνός τρόπος για να αυξήσουμε την απόδοση της εφαρμογής μας. Αυτό που κάνει είναι να αποθηκεύει σχετικά στατικά στοιχεία, και όταν αυτά ζητηθούν να δίνονται από την cache (κρυφή μνήμη) αντί να παράγονται εξαρχής. Έτσι γλυτώνουμε χρόνο και υπολογιστική ισχύ. Το Yii μας παρέχει 8 κλάσεις για caching:

- **CMemCache**: χρησιμοποιεί το memcache extension της PHP το οποίο χρησιμοποιεί την μνήμη του server για την αποθήκευση δεδομένων.
- **CDbCache**: χρησιμοποιεί έναν πίνακα σε βάση δεδομένων για την αποθήκευση των δεδομένων της cache. Εξ' ορισμού δημιουργεί μια sqlite βάση δεδομένων στον φάκελο runtime της εφαρμογής.
- **CFileCache**: χρησιμοποιεί αρχεία για την αποθήκευση της cache. Η κλάση αυτή είναι χρήση για μεγάλα κομμάτια δεδομένων όπως ολόκληρες σελίδες.
- **CApcCache**: χρησιμοποιεί το APC extension της PHP.
- **CXCache**: χρησιμοποιεί το XCache extension της PHP.
- **CEAcceleratorCache**: χρησιμοποιεί το EAccelerator extension της PHP.
- **CZendDataCache**: χρησιμοποιεί το Zend Data Cache ως μέσο για caching.
- **CDummyCache**: Η κλάση αυτή δεν κάνει καθόλου caching. Ο ρόλος της είναι να απλοποιήσει τον κώδικα κατά την ανάπτυξη της εφαρμογής. Αν για παράδειγμα κατά την ανάπτυξη ο server δεν έχει υποστήριξη για caching, μπορούμε να χρησιμοποιήσουμε αυτή την κλάση και όταν ο server αποκτήσει υποστήριξη, ο κώδικας θα είναι έτοιμος.

Επειδή όλες οι παραπάνω κλάσεις κληρονομούνε την CCache η αναφορά σε αυτές είναι η ίδια για όλες: **Yii::app()->cache.**

Με το caching component μπορούμε να αποθηκεύσουμε στην προσωρινή μνήμη: μεμονωμένες μεταβλητές, sql ερωτήματα, τμήματα μια σελίδας ακόμα και

ολόκληρες σελίδες. Τα δεδομένα αποθηκεύονται στην cache, για όση ώρα ορίσουμε εμείς.

9.2 URL Management

Με το component CurlManager το Yii διαχειρίζεται τα URLs. Η δουλειά του είναι να δημιουργεί user-friendly URLs αλλά και να μεταφράζει τα URLs αυτά σε κατανοητά προς την εφαρμογή routes. Επίσης βοηθάει τον προγραμματιστή να δημιουργήσει urls με προγραμματιστικό τρόπο. Μπορούμε ακόμα να δημιουργήσουμε εικονικά hostnames. Το url http://paul.bestwebapp.com/profile μπορεί εσωτερικά, για την εφαρμογή μας, να μεταφραστεί ως http://bestwebapp.com/index.php?r=profile&user=paul και απλά να εμφανίσει το προφίλ του χρήστη “paul”. Μας δίνει επίσης την δυνατότητα να κρύψουμε το index.php (με την βοήθεια του web server προγράμματος) καθώς και να βάλουμε ψεύτικες καταλήξεις, κάνοντας το site μας να φαίνεται στατικό.

9.3 Ασφάλεια

Στο θέμα της ασφάλειας το Yii παρέχει επαρκή εργαλεία για την προστασία της εφαρμογής μας. Αρχικά με την βοήθεια του CHtmlPurifier μπορούμε να αποτρέψουμε CSS επιθέσεις. Μια Cross-Site Scripting επίθεση γίνεται ως εξής: Έστω ότι η εφαρμογή μας είναι ένα forum. Κάποιος κακόβουλος χρήστης, μπορεί να εισάγει στο πεδίο του post του, κάποιο Javascript, VBScript, ActiveX ή flash κι έτσι όταν κάποιος άλλος χρήστης το εμφανίσει στον browser του, το πρόγραμμα θα εκτελεστεί κάνοντας κάτι που δεν πρέπει στον υπολογιστή του. Το σημαντικότερο μέτρο κατά των CSS επιθέσεων είναι να ελέγχουμε δεδομένα που έχουν εισάγει χρήστες προτού τα εμφανίσουμε. Όπως είδαμε και σε κάθε κ΄ψδικα που εμδανίσαμε δεδομένα πρώτα τα περνάμε από html encoding με την μέθοδο encode της CHtml. Η βιβλιοθήκη HTMLPurifier της PHP είναι ένα εργαλείο που μας δίνει πολλές δυνατότητες και καθαρίζει δεδομένα από κακόβουλο κώδικα. Η κλάση CHtmlPurifier είναι ουσιαστικά ένα περιτύλιγμα της HTMLPurifier.

Ένας ακόμα τρόπος επίθεσης είναι ο CSR ή Cross-Site Request. Εδώ ένα κακόβουλο site μπορεί να εκμεταλλευτεί τα cookies που έχει κάποιος χρήστης από την δική μας εφαρμογή και να κάνει μια ενέργεια στην εφαρμογή μας χωρίς ο χρήστης να το καταλάβει. Για παράδειγμα. Έχουμε μια εφαρμογή web banking

(www.bank.example) ένας από τους χρήστες μας έχει κάνει login και σε ένα άλλο site ανοίγει μια σελίδα η οποία ως src μιας εικόνας έχει το url http://bank.example/withdraw?transfer=10000&to=someone. Στην εφαρμογή μας φαίνεται πως η αίτηση έρχεται από τον browser του χρήστη ο οποίος είναι ήδη logged in οπότε και την εκτελεί. Κι έτσι, 10000 ευρώ μεταφέρονται σε κάποιον λογαριασμό. Μια λύση, είναι πρώτα να μην υλοποιούμε τέτοια εντολή με μέθοδο GET. Κι έπειτα στην φόρμα που αποστέλλεται με την μέθοδο POST για να εκτελέσει ένα τέτοιο αίτημα, συμπεριλαμβάνουμε και μια τυχαία τιμή που μόνο η εφαρμογή μας μπορεί να αναγνωρίσει. Την τυχαία αυτή τιμή κατά το log-in την αποθηκεύει σε ένα cookie και σε κάθε αίτημα συγκρίνει τις 2 τιμές.

9.4 Διεθνοποίηση

Η διεθνοποίηση ή Internationalization ή “i18n” εν συντομία (όπου 18 είναι το πλήθος των ενδιάμεσων γραμμάτων της λέξης internationalization), είναι η διαδικασία σχεδιασμού μιας εφαρμογής έτσι ώστε να μπορεί να προσαρμοστεί σε οποιαδήποτε γλώσσα χωρίς την ανάγκη αλλαγής του κώδικα. Κάτι που στις εφαρμογές ιστού είναι μείζονος σημασίας αφού οι χρήστες είναι από όλο τον κόσμο.

Το Yii υποστηρίζει i18n με αρκετούς τρόπους. Παρέχει

- ✓ Τοπικές ρυθμίσεις και προτιμήσεις για όλες τις γλώσσες.
- ✓ Δυνατότητα μετάφρασης αρχείων ή και φράσεων.
- ✓ Τοπική διαμόρφωση ημερομηνίας και ώρας.
- ✓ Τοπική διαμόρφωση αριθμών.

Το Yii διαχωρίζει της γλώσσες σε target και source. Η πρώτη είναι η γλώσσα των χρηστών στους οποίους απευθύνεται η εφαρμογή ενώ η δεύτερη είναι η γλώσσα στην οποία είναι γραμμένα τα κείμενα μιας εφαρμογής. Η μετάφραση γίνεται πολύ απλά με την μέθοδο Yii::t(‘<componentID>’,’messageToTranslate’). Τα μεταφρασμένα κείμενα μπορούν να αποθηκευτούν είτε σε php αρχείο ως στοιχεία ενός php array, είτε σε βάση δεδομένων είτε σε GNU Gettext αρχεία.

3η Ενότητα

Το Yii στην πράξη

(Υλοποίηση Εφαρμογής)

Κεφάλαιο 10: Ανάλυση της Εφαρμογής

10.1 Περίληψη

Για να δείξουμε το Yii στην πράξη αποφασίστηκε να δημιουργηθεί μια μικρή εφαρμογή, χρησιμοποιώντας κυρίως components που έρχονται μαζί με το Yii. Οπότε θα εκτελέσουμε το yiic και με βάση τον default σκελετό που αυτό παράγει θα δημιουργήσουμε μια εφαρμογή τροποποιώντας μέρη της και προσθέτοντας κώδικα και components όπου χρειάζεται έτσι ώστε να έχουμε μια λειτουργική εφαρμογή.

10.2 Η εφαρμογή

Προς υλοποίηση επιλέχτηκε μια εφαρμογή «συν-οδήγησης» ή carpooling. Πρόκειται για μια πλατφόρμα όπου οι χρήστες καταχωρούν ένα ταξίδι ή μια διαδρομή που θα πραγματοποιήσουν με το όχημα τους, μαζί με κάποιες επιπλέον πληροφορίες όπως διαθέσιμες κενές θέσεις, το κόστος της διαδρομής. Από την στιγμή που γίνει η καταχώρηση η διαδρομή προβάλλεται στην πλατφόρμα και οι υπόλοιποι χρήστες μπορούν να την δουν, κι αν η διαδρομή τους εξυπηρετεί, να έρθουν σε επαφή με τον δημιουργό της και να συν-ταξιδέψουν μοιραζόμενοι τα έξοδα. Ουσιαστικά δηλαδή μιλάμε για “ηλεκτρονικό ωτοστόπ” όπου όμως το κόστος της διαδρομής μοιράζεται στους επιβαίνοντες. Ο όνομα της εφαρμογής είναι το «Πάμε Ρεφενέ». Ρεφενέ σημαίνει τα έξοδα μοιρασμένα.

10.3 Απαιτήσεις

Η εφαρμογή θα πρέπει να παρέχει τις εξής λειτουργίες:

- ✓ Εγγραφή χρήστη
- ✓ Δυνατότητα ανάκτησης κωδικού εισόδου
- ✓ Καταχώρηση διαδρομής (ως προσφορά ή ως ζήτηση)
- ✓ Αναζήτηση διαδρομών με κριτήρια
- ✓ Επεξεργασία προφίλ χρήστη

- ✓ Panel διαχείρισης χρηστών και διαδρομών για τους admin χρήστες

Κεφάλαιο 11: Σχεδιασμός της Βάσης Δεδομένων

11.1 Πίνακες

Δύο είναι οι βασικοί πίνακες της εφαρμογής: Ο route όπου αποθηκεύονται οι διαδρομές που καταχωρούνται και ο user όπου αποθηκεύονται οι χρήστες και διάφορα στοιχεία για αυτούς. Υπάρχουν επίσης ο πίνακας county όπου υπάρχουν τα ονόματα των νομών και ο πίνακας town με τα ονόματα των πόλεων. Για την ονομασία τους υιοθετήθηκε η πρόταση του yii για lower case και ενικό αριθμό.

route(id, type, start_id, dest_id, date, time, cost, comment, seats, user_id, created).
user(id, email, pass, type, registered, activation_key, active, last_login, name, year_born, lives_in_id, phone, has_driving_license, smoking).
town(id,name,county_id).
county(id,name).

11.2 Αναλυτική παρουσίαση

Ακολουθεί η αναλυτική παρουσίαση των πινάκων, με όλα τα πεδία τους και την περιγραφή κάθε πεδίου.

Πίνακας: route	**Περιγραφή πεδίου**
id	Το id της εγγραφής.
type	Ο τύπος του διαδρομής. Προσφορά ή ζήτηση.
start_id	Το id της πόλης αναχώρησης.
dest_id	Το id της πόλης άφιξης.
date	Η ημερομηνία της διαδρομής.
time	Η ώρα αναχώρησης.
cost	Το κόστος της διαδρομής.
comment	Σχόλιο για την διαδρομή αυτή από τον χρήστη.
seats	Διαθέσιμες κενές θέσεις.
user_id	Το id του χρήστη που καταχώρησε την διαδρομή.
created	timestamp καταχώρησης της διαδρομής στην εφαρμογή

Πίνακας: user	**Περιγραφή πεδίου**
id	Το id της εγγραφής.
email	Η διεύθυνση email του χρήστη.
pass	Το hass του κωδικού εισόδου του χρήστη.
type	Ο τύπος του χρήστη, Απλός user ή administrator.
registered	Το timestamp εγγραφής.
activation_key	Το κλειδί ενεργοποίησης του λογαριασμού του χρήστη.
active	Αν το μέλος είναι ενεργοποιημένο ή όχι.
last_login	timestamp τελευταίας εισόδου στην εφαρμογή.
name	Το όνομα του χρήστη.
year_born	Το έτος γέννησης του χρήστη.
lives_in_id	Το idτης πόλης κατοικίας. (από τον πίνακα town)
phone	Τηλέφωνο του χρήστη.
has_driving_license	Αν ο χρήστης έχει άδεια οδήγησης.
smoking	Αν ο χρήστης καπνίζει ή όχι.

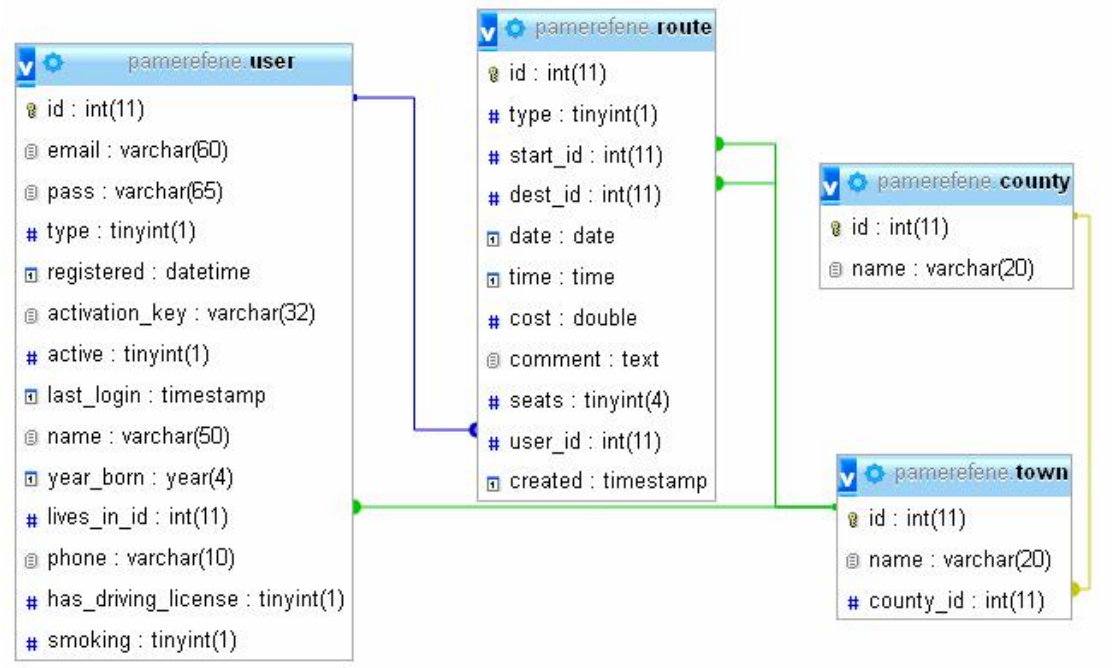
Πίνακας: town	**Περιγραφή πεδίου**
id	Το id της εγγραφής.
name	Το όνομα της πόλης.
conty_id	Το id του νομού που βρίσκεται η πόλη (πινακας county)

Πίνακας: county	**Περιγραφή πεδίου**
id	Το id της εγγραφής.
name	Το όνομα του Νομού

11.3 Σχέσεις Πίνάκων

Οι πίνακες της εφαρμογής μας σχετίζονται μεταξύ τους αναφερόμενοι ο ένας σε πεδία του άλλου. Στον πίνακα **route** το πεδίο *user_id* παίρνει τιμές του πεδίου *id* του πίνακα **user**. Επίσης τα πεδία *start_id* και *dest_id* παίρνουν τιμές από το πεδίο *id* του πίνακα **town**. Τέλος στον πίνακα **town** το πεδίο *county_id* παίρνει τιμές από το πεδίο *id* του πίνακα **county**. Τα πεδία που αναφέρονται σε

άλλον πίνακα είναι γνωστά ως “ξένα κλειδιά”. Οι σχέσεις που δηλώνονται στην βάση δεδομένων, περνάνε και στο yii μέσω του gii κατά την δημιουργία models. Στην παρακάτω εικόνα βλέπουμε διαγραμματικά της σχέσης των πινάκων.



Κεφάλαιο 12: Υλοποίηση της εφαρμογής με το Yii

12.1 Εγκατάσταση του περιβάλλοντος εργασίας

Το περιβάλλον εργασίας για την ανάπτυξη της εφαρμογής αποτελείται από 3 μέρη: Το XAMPP, το Yii και το NetBeans IDE. Ας δούμε τι είναι το καθένα και πως μπορούμε να το προμηθευτούμε.

XAMPP

Το XAMPP είναι ένα πακέτο που εγκαθιστά στον υπολογιστή μας ότι χρειάζεται ώστε να γίνει server, έτσι ώστε να τεστάρουμε την εφαρμογή μας τοπικά. Περιλαμβάνει τον Apache ως web server, την PHP ως scripting γλώσσα, την MySQL ως ΣΔΒΔ, και ένα εργαλείο το PhpMyAdmin για την γραφική διαχείρηση της MySQL. Το όνομα του είναι τα αρχικά των παραπάνω με το X να σημαίνει ότι διατίθεται για οποιοδήποτε λειτουργικό σύστημα.

Το XAMPP είναι ελεύθερο λογισμικό και μπορούμε να το προμηθευτούμε από την διεύθυνση http://www.apachefriends.org/en/ . Αφού το εγκαταστήσουμε στον φάκελο που επιλέξουμε θα βρούμε τον φάκελο **htdocs** ο οποίος είναι προσπελάσιμος από τον browser μας μέσω της διεύθυνσης **http://localhost.** Αν την επισκεφθούμε ανακατευθυνόμαστε στον φάκελο **xampp** δηλαδή στην διεύθυνση **http://localhost/xampp**. Αν όλα έχουνε πάει καλά θα πρέπει να βλέπουμε την παρακάτω εικόνα:

Εικόνα 12.1: Αρχική οθόνη του XAMPP

Στην διεύθυνση **http://localhost/phpmyadmin/** μπορούμε να χειριστούμε το phpMyAdmin και να δημιουργήσουμε τους πίνακες της εφαρμογής μας.

Yii

Το Yii είναι το framework που θα χρησιμοποιήσουμε. Το βρίσκουμε στην διεύθυνση http://www.yiiframework.com όπου κατεβάζουμε την τελευταία έκδοση σε μορφή zip και μέσα στο zip, αυτό που μας ενδιαφέρει είναι ο φάκελος framework. Αυτός περιέχει όλες τις κλάσεις του framework καθώς και το εργαλείο yiic με το οποίο θα δημιουργήσουμε τον σκελετό της εφαρμογής μας. Μπορούμε να τον τοποθετήσουμε μέσα στον φάκελο htdocs αν και συνηθίζεται για λόγους ασφαλείας να μην τοποθετείται σε φάκελο προσβάσιμο από οποιονδήποτε. Όταν ανεβάσουμε την εφαρμογή μας στο διαδίκτυο καλό είναι να τον βάλουμε έναν φάκελο πιο πάνω από τον public.

NetBeans

Το NetBeans είναι ένα IDE. Ουσιαστικά είναι ένας editor αλλά με πάρα πολλές διευκολύνσεις για την ανάπτυξη κώδικα, προς τον προγραμματιστή. Από προτάσεις για συμπλήρωση κώδικα μέχρι και πλήρη λίστα ιδιοτήτων και μεθόδων των αντικειμένων.

Με την εγκατάσταση των παραπάνω, το περιβάλλον εργασίας μας είναι έτοιμο και μπορούμε να αρχίσουμε την ανάπτυξη της εφαρμογής μας.

12.2 Δημιουργία της βάσης δεδομένων

Αρχίζουμε με την δημιουργία της βάσης δεδομένων. Επισκεπτόμαστε την διεύθυνση http://localhost/phpMyAdmin. Εκεί δημιουργούμε μια καινούρια βάση με το όνομα pamerefene. Στην βάση αυτή δίνουμε τον κώδικα SQL που δημιουργεί τους πίνακες:

```
CREATE TABLE `route` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `type` tinyint(1) NOT NULL COMMENT '1=offer_ride
0=request_ride',
```

```
  `start_id` int(11) NOT NULL COMMENT 'Starting town id',
  `dest_id` int(11) NOT NULL,
  `date` date NOT NULL,
  `time` time NOT NULL,
  `cost` double DEFAULT NULL,
  `comment` text COLLATE utf8_unicode_ci,
  `seats` tinyint(4) DEFAULT NULL,
  `user_id` int(11) DEFAULT NULL,
  `created` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`),
  KEY `dest_id` (`dest_id`),
  KEY `start_id` (`start_id`),
  KEY `user_id` (`user_id`)
);

CREATE TABLE `user` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `email` varchar(60) COLLATE utf8_unicode_ci NOT NULL,
  `pass` varchar(65) COLLATE utf8_unicode_ci NOT NULL COMMENT
'php hash_hmac(''sha256'',' ,
  `type` tinyint(1) NOT NULL DEFAULT '1' COMMENT '0=admin,
1=user',
  `registered` datetime NOT NULL,
  `activation_key` varchar(32) COLLATE utf8_unicode_ci DEFAULT
NULL,
  `active` tinyint(1) NOT NULL DEFAULT '0',
  `last_login` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON
UPDATE CURRENT_TIMESTAMP,
  `name` varchar(50) COLLATE utf8_unicode_ci DEFAULT NULL,
  `year_born` year(4) DEFAULT NULL,
  `lives_in_id` int(11) DEFAULT NULL,
  `phone` varchar(10) COLLATE utf8_unicode_ci DEFAULT NULL,
  `has_driving_license` tinyint(1) DEFAULT NULL,
  `smoking` tinyint(1) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `lives_in_id` (`lives_in_id`)
);
```

```
CREATE TABLE `county` (
  `id` int(11) NOT NULL,
  `name` varchar(20) COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (`id`)
);

CREATE TABLE `town` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(20) COLLATE utf8_unicode_ci NOT NULL,
  `county_id` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `county_id` (`county_id`)
);
```

Και μετά τις συσχετίσεις των πινάκων:

```
ALTER TABLE `route`
  ADD CONSTRAINT `route_ibfk_5` FOREIGN KEY (`start_id`)
REFERENCES `town` (`id`) ON DELETE NO ACTION ON UPDATE NO
ACTION,
  ADD CONSTRAINT `route_ibfk_6` FOREIGN KEY (`dest_id`)
REFERENCES `town` (`id`) ON DELETE NO ACTION ON UPDATE NO
ACTION,
  ADD CONSTRAINT `route_ibfk_8` FOREIGN KEY (`user_id`)
REFERENCES `user` (`id`) ON DELETE SET NULL ON UPDATE SET NULL;

ALTER TABLE `user`
  ADD CONSTRAINT `user_ibfk_1` FOREIGN KEY (`lives_in_id`)
REFERENCES `town` (`id`) ON DELETE NO ACTION ON UPDATE NO
ACTION;

ALTER TABLE `town`
  ADD CONSTRAINT `town_ibfk_1` FOREIGN KEY (`county_id`)
REFERENCES `county` (`id`) ON DELETE NO ACTION;
```

Πλέον η βάση μας είναι έτοιμη να δεχθεί δεδομένα.

12.3 Σκελετός της εφαρμογής και αρχικές ρυθμίσεις.

Αποσυμπιέζουμε το αρχείο που κατεβάσαμε από το site του Yii. Μετονομάζουμε τον φάκελο που προέκυψε σε **YiiRoot** και τον τοποθετούμε στον φάκελο που βρίσκεται ο htdocs δηλαδή στον φάκελο που εγκαταστήσαμε το XAMPP. Ανοίγουμε ένα τερματικό και μεταβαίνουμε στον htdocs εκεί δίνουμε την εξής εντολή:

>..\YiiRoot\framework\yiic webapp pamerefene

Και επιβεβαιώνουμε με y και enter. Μετά από αυτό, η εφαρμογή μας είναι έτοιμη και επισκέψιμη στην διεύθυνση http://localhost/pamerefene ενώ στον φάκελο htdocs/pamerefene έχουν δημιουργηθεί όλα τα αρχεία της.

Τώρα θα ετοιμάσουμε το NetBeans για την επεξεργασία του κώδικα της εφαρμογής. Ανοίγουμε το NetBeans και επιλέγουμε File->New Project… Στην πρώτη οθόνη επιλέγουμε ως τύπο “PHP Application”. Στην 2η ως ProjetName δίνουμε “pamerefene” και Sources Folder δίνουμε τον φάκελο που δημιούργησε το yiic. Τις επόμενες 2 οθόνες τις αφήνουμε ως έχουν. Στα αριστερά του παραθύρου του NetBeans βλέπουμε την δομή των φάκελων της εφαρμογής μας. Ανοίγουμε τον φάκελο *protected* τον βασικό φάκελο της εφαρμογής. Εκεί μέσα βρίσκεται ο φάκελος *config* και μέσα σε αυτόν το αρχείο *main.php* όπου βρίσκονται οι βασικές ρυθμίσεις της εφαρμογής.

Οι ρυθμίσεις είναι σε μορφή array όπου τα κλειδιά του είναι τα ονόματα των εκάστοτε ρυθμίσεων. Αρχικά δίνουμε όνομα στην εφαρμογή μας αλλάζοντας την τιμή του κλειδιού ‘name’ από ‘My Web Application’ σε ‘Πάμε Ρεφενέ’. Ακριβώς από κάτω προσθέτουμε την γραμμή “ 'language'=>'el', “ για να ορίσουμε την γλώσσα για τις μεταφράσεις σε ελληνικά. Παρακάτω στο κλειδί ‘modules’ αποσχολιάζουμε τον πίνακα gii ώστε να μπορέσουμε να το χρησιμοποιήσουμε αργότερα, και στο κλειδί του ‘password’ δίνουμε έναν κωδικό που θα μας ζητηθεί όταν προσπαθήσουμε να επισκεφτούμε το gii στον browser μας (θα το δούμε παρακάτω).

Παρακάτω αποσχολιάζουμε το urlManager στα components για να έχουμε URLs σε μορφή path. Ας εξηγήσουμε λίγο πως δουλεύει το component αυτό. Εδώ θέτουμε κανόνες για να κάνουμε τα URLs περισσότερο ures-friendly. Αν ορισουμε το urlFormat σε path, ένα route με την μορφή

“/index.php?r=post/read&id=100” θα γίνει “/index.php/post/read/id/100”. Με την χρήση κανόνων όμως μπορούμε να το κάνουμε ακόμα πιο user-friendly στην ακόμα πιο απλή μορφή: “/post/100” (για να κρύψουμε το index.php πρέπει να κάνουμε και κάτι άλλο στον Apache που θα δούμε αργότερα). Η μορφή αυτή επιτυγχάνετε με τον πρώτο κανόνα:

```
<controller:\w+>/<id:\d+>>'=>'<controller>/view',
```

Ο κανόνας αυτός λέει ότι αν δώσουμε ένα url στην μορφή <λέξη>/<αριθμός> αυτό θα μετατραπεί στο route <λέξη>/view και ο αριθμός θα περαστεί σαν όρισμα στην action view.

Αμέσως μετά σειρά έχει η σύνδεση με την βάση δεδομένων. Αυτή την στιγμή η εφαρμογή είναι ρυθμισμένη να συνδέεται σε μια sqlite βάση δεδομένων που περιέχει δεδομένα για να είναι λειτουργική η εφαρμογή μας. Διαγράφουμε αυτή την ρύθμιση και αποσχολιάζουμε την παρακάτω η οποία είναι η default ρύθμιση για την MySQL σε τοπικό server. Το μόνο που πρέπει να αλλάξουμε είναι η τιμή του ‘dbname’ από testdrive σε pamerefene, το όνομα της βάσης που έχουμε δημιουργήσει στην MySQL μας. Όταν αργότερα ανεβάσουμε την εφαρμογή σε κάποιον server στο διαδίκτυο, εδώ θα ορίσουμε και τα αντίστοιχα πεδία username, password και host. Τώρα μπορούμε να χρησιμοποιήσουμε το gii και να δημιουργήσουμε τα model της εφαρμογής.

Το μόνο που μένει στις ρυθμίσεις είναι στο τέλος το κλειδί ‘adminEmail’ οπου ορίζουμε το δικό μας e-mail.

Για να κρύψουμε και το index.php από τα Urls δημιουργούμε στον κυρίως φάκελο της εφαρμογής δίπλα στο indx.php ένα αρχείο με όνομα “.htaccess” με περιεχόμενο:

```
RewriteEngine on
# Don't perform redirects for files and directories that exist:
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
# For everything else, redirect to index.php:
RewriteRule ^(.*)$ index.php/$1
```

Το αρχείο αυτό λέει στον Apache πως όταν δεν δίνεται κάποιο αρχείο στην αίτηση, να εννοείται το index.php. Αυτό που μένει είναι να πούμε στο yii να μην βάζει το index.php στα links που αυτό δημιουργεί. Αυτό γίνεται προσθέτοντας στον urlManager την εξής ρύθμιση:

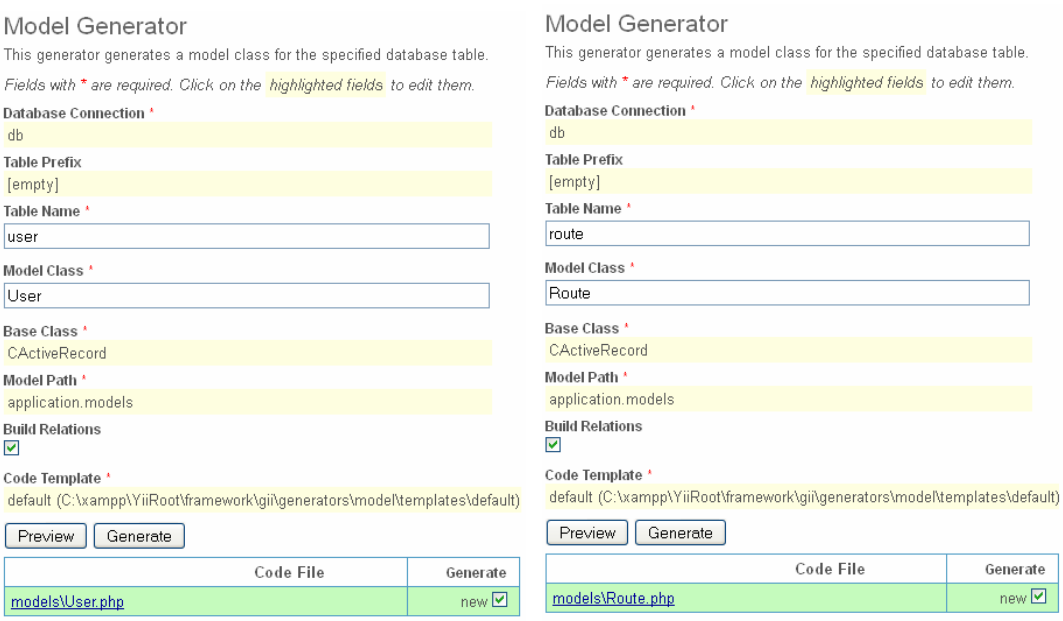
```
'showScriptName'=>false,
```

Προς το παρόν τελειώσαμε με τις ρυθμίσεις της εφαρμογής. Αποθηκεύουμε το αρχείο και ανοίγουμε έναν browser για να δουλέψουμε το gii.

12.4 Δημιουργία models και controllers με το gii.

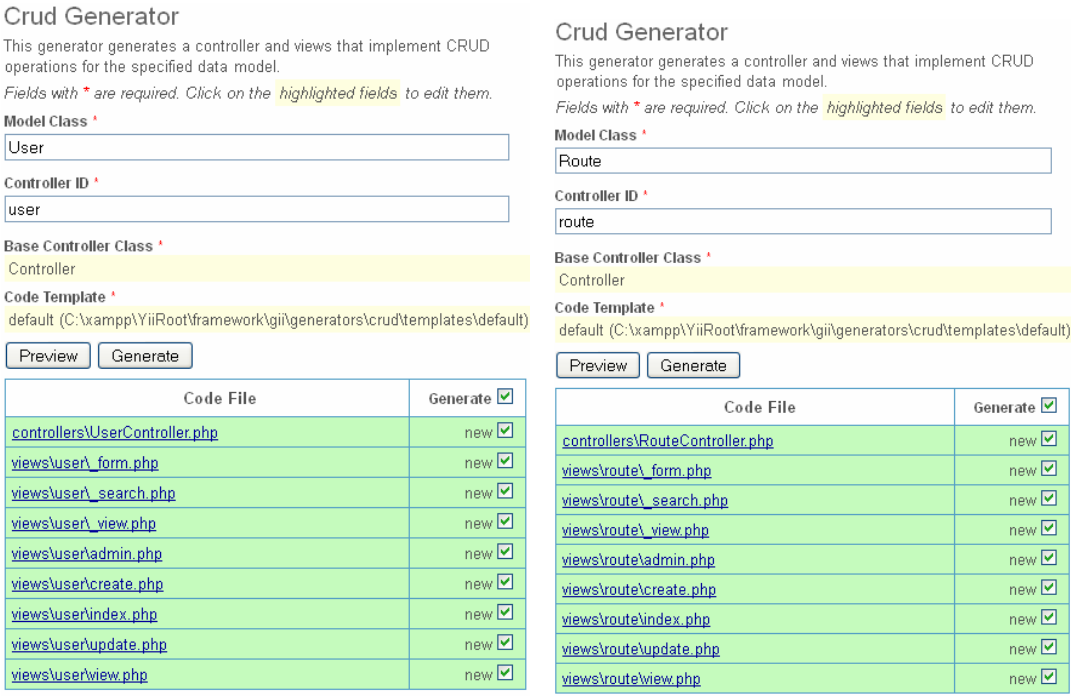
Επισκεπτόμαστε την διεύθυνση http:://localhost/pamerefene/gii δίνουμε τον κωδικό που ορίσαμε στο main.php. Εδώ θα δημιουργήσουμε models για τους πίνακες user και route καθώς και τις Crud λειτουργίες γιαυτούς, με την διαδικασία που δείξαμε στο κεφάλαιο 6.3. Παρακάτω δείχνουμε τις εικόνες για το κάθε στάδιο

Τα models:



Εικόνα 12.2: Δημιουργία models

Οι CRUD λειτουργίες και οι Controllers:



Εικόνα 12.3: Δημιουργία CRUD

Πλέον έχουμε έτοιμες τις βασικές κλάσεις τις εφαρμογής μας. Τα models, οι controllers και οι views είναι έτοιμες. Απλά θα τις τροποποιήσουμε για να δουλεύουν και να εμφανίζονται όπως εμείς θέλουμε. Μας μένει όμως να εγκαταστήσουμε ένα module.

12.5 Εγκατάσταση module για διαχείριση λογαριασμών χρηστών.

Για την διαχείριση των λογαριασμών των χρηστών θα εγκαταστήσουμε ένα module από την βάση extensions του Yii, για να δείξουμε και την διαδικασία που ακολουθείτε. Αρχικά πηγαίνουμε στην διεύθυνση http://yiiframework.com /extension/usr και κατεβάζουμε την πιο πρόσφατη έκδοση. Αποσυμπιέζουμε το αρχείο στον φάκελο protected/modules της εφαρμογής μας. Όλα τα models έχουν ένα README αρχείο όπου εξηγούν την διαδικασία εγκατάστασης τους και την ρύθμισή τους. Εκέι μας λέει ότι πρέπει να το δηλώσουμε στο αρχείο main.php στα modules ως εξής:

```
'modules'=>array(
        'usr'=>array(
                'userIdentityClass' => 'UserIdentity',
        ),
```

Η κλάση UserIdentity είναι ένα component του Yii (στον φάκελο protected/components) που την περιέχει ήδη μετά το yiic κληρονομέι την CUserIdentity και είναι αυτή που αναπαριστά την ταυτότητα του τρέχον χρήστη που χρησιμοποιεί την εφαρμογή. Το usr module έχει παραδείγματα για την κλάση UserIdentity και το model του user που θα υλοποιήσουμε τα οποία περιέχουν τις μεθόδους που χρειάζεται το module για να λειτουργήσει σωστά. Τις τροποποιούμε ανάλογα με βάση τα στοιχεία των δικών μας users. Έτσι αντιγράφουμε την ExampleUserIdentity στον φάκελο protected/componenets της εφαρμογής μας την μετονομάζουμε σε UserIdentity και κάνουμε τις ανάλογες αλλαγές ώστε να ταιριάζει με τα στοιχεία των δικών μας users.

Επίσης, στην παραδειγματική κλάση ExampleUser βλέπουμε 3 μεθόδους τις οποίες χρειάζεται το usr για να λειτουργήσει. Αυτές είναι οι hashPassword() και verifyPassword() τις οποίες ορίζουμε στο model User μας ως εξής:

```
public static function hashPassword($password){
  return hash_hmac('sha256',
     $password.Yii::app()->params['salt'],
     Yii::app()->params['encryptionKey']);
}

public function verifyPassword($password){
  return $this->pass == self::hashPassword($password);
}
```

Στην μέθοδο haspassword χρησιμοποιούμε μια μέθοδο hash με επιπλέον salt στον κωδικό. Οι τιμές “Yii::app()->params['salt']” και “['encryptionKey']);” είναι αποθηκευμένες στο main.php στο κλειδί params για να είναι προσβάσιμες από οποιοδήποτε σημείο της εφαρμογής.

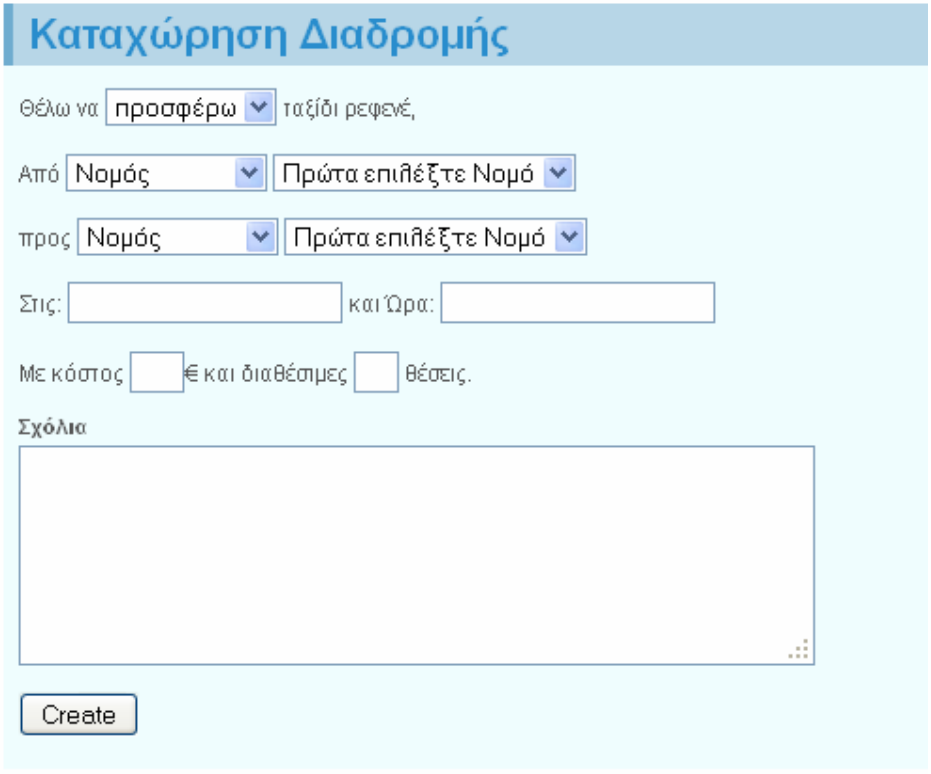
Τέλος, το route του controller του module είναι /usr/default/<action>. Αν θέλουμε να το κάνουμε πιο user friendly, το README μας προτίνει να προσθέσουμε τον εξής κανόνα στο urlManager module:

```
'<action:(login|logout|reset|recovery|register|profile)>'=>'usr/default
/<action>',
```

Μέσα στην παρένθεση είναι όλες οι actions του default controller ενω το “|” δηλώνει ότι μπορεί να ισχύει μία από αυτές.

12.6 Καταχώρηση διαδρομής

Θα διαμορφώσουμε τώρα την view “create.php” του RouteController, ώστε να μοιάζει ως εξής:



Εικόνα 12.3: Καταχώρηση διαδρομής

Η create.php κάνει χρήση της _form.php η οποία χρησιμοποιείτε και από την update.php για την επεξεργασία μιας διαδρομής. Θα παραθέσουμε εδώ τον κώδικα για τις συνδεδεμένες drop down lists νομού-πόλης:

```
<div class="row">
  προς
  <?php echo CHtml::dropDownList('dest_county_id', '',
  CHtml::listData(Town::model()->with('county')->findAll(),
'county.id', 'county.name'),
      array(
       'prompt' => 'Νομός',//Τι να εμφανίζεται στην αρχή
       'ajax' => array(
            'type'=>'GET', //μέοδος αίτησης AJAX
            'url'=>Yii::app()->createUrl('Route/dynamicCities'),
            //Ποιο πεδίο να ενημερωθεί με τα AJAX αποτελέσματα
            'update'=>'#'. CHtml::activeId($model, 'dest_id'),
            'data'=>array('county_id'=>'js:this.value')
  )));  ?>
  <?php //Η 2η dropDownList
```

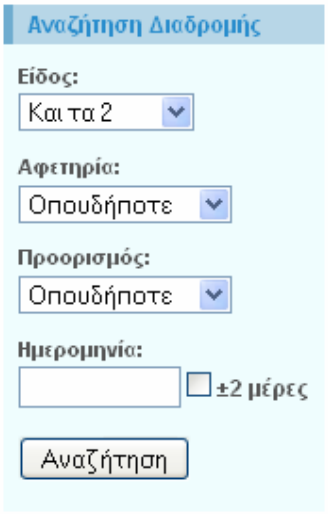
```
 echo $form->dropDownList($model, 'dest_id',
empty($model->dest_id) ? //ελέγχει αν το πεδίο είναι άδειο οπότε
είμαστε σε δημιουργία διαδρομής και θα μπεί prompt
array ( 'prompt' => 'Πρώτα επιλέξτε Νομό' ) :
//αλλιώς είμαστε σε επεξεργασία διαδρομής οπότε εμφανίζει όλους
τους νομούς
CHtml::listData(Town::model()->findAll(), 'id', 'name')
                    ); ?>
<?php echo $form->error($model,'dest_id'); ?>
</div>
```

Η μέθοδος Route/dynamicCities που επιστρέφει τις πόλεις του νομού που θα επιλεγεί είναι η εξής:

```
public function actionDynamiccities(){
        $q="SELECT * FROM town WHERE county_id = :county_id";
        $cmd=Yii::app()->db->createCommand($q);
        $cmd->bindParam(":county_id", $_GET["county_id"]);
        $result=$cmd->query();
        foreach($result as $row){
           echo CHtml::tag('option',
                array('value'=>$row['id']),CHtml::encode($row['name']),true);
      }
}
```

12.7 Αναζήτηση διαδρομής

Για την αναζήτηση διαδρομής θα δημιουργήσουμε ένα widget που θα είναι στα δεξιά της σελίδας και θα φαίνεται έτσι:



Εικόνα 12.4: Το Widget Αναζήτησης

Θα το ενσωματώσουμε σε ένα layout view το οποίο θα ορίζουμε στην ιδιότητα layout όποιου controller θέλουμε να εμφανίζεται το παραπάνω widget αναζήτησης. Το widgets έχει 2 μέρη. Το μέρος προβολής του και το μέρος υλοποίησης της αναζήτησης στο model Route. Στο layout αρχείο έχουμε τον εξής κώδικα:

```
<div id="sidebar">
 <?php $this->beginWidget('zii.widgets.CPortlet', array(
                'title'=>'Αναζήτηση  Διαδρομής',
           ));?>
<div class="form">
 <?php  $model=new Route('search');
        $model->unsetAttributes();
        If (isset($_POST["Route"]))//αν έχει γίνει αναζήτηση
           $model->attributes=$_POST["Route"];
        $form=$this->beginWidget('CActiveForm',
           array(
            'action'=>Yii::app()->createUrl('route/index'),
            'method'=>'post',
            'id'=>'route-form'
                 )); ?>

<div class="row">
<label>Είδος:</label>
<?php echo $form->dropDownList($model, 'type',
array(1=>'Προσφορά',0=>'Ζήτηση', null=>'Και τα 2')
); ?>
</div>

<div class="row">
<label>Αφετηρία:</label>
<?php echo $form->dropDownList($model, 'start_id',
     CHtml::listData(Town::model()->findAll(), 'id', 'name'),
     array ( 'prompt' => 'Οπουδήποτε' )
)?>
</div>

<div class="row">
<label>Προορισμός:</label>
<?php echo $form->dropDownList($model, 'dest_id',
     CHtml::listData(Town::model()->findAll(), 'id', 'name'),
     array ( 'prompt' => 'Οπουδήποτε' )
)?>
</div>

<div class="row">
<label>Ημερομηνία:</label>
<?php $this->widget('zii.widgets.jui.CJuiDatePicker',
                array(
                      'name'=>'Route[date]',
                      'value'=>$model->date,
                      'language' => 'el',
```

```
                    'themeUrl' => Yii::app()-
                >baseUrl.'/assets/f62b1645/jui/css' ,
                    'theme'=>'base',
                    'cssFile'=>array('jquery-ui.css'),
                    'options' => array(
                    'dateFormat'=>'dd/mm/yy',
                    'minDate'=>'new Date()',
                    'changeMonth' => 'true',
                    'changeYear' => 'true',
                    'constrainInput' => 'true',
                    'duration'=>'normal',
                    'showAnim' =>'fade',
                ),
                'htmlOptions' => array(
                    'size' => '10',
                    'maxlength' => '10',
                )
        )
 );
echo ' '.$form->checkBox($model,'days2_range',array('value' =>
'1'));?> <strong> ±2 μέρες</strong>
</div>

<div class="row buttons">
<?php echo CHtml::submitButton('Αναζήτηση'); ?>
</div>
</div>
<?php $this->endWidget();
$this->endWidget();?>
```

Ουσιαστικά κάνουμε μια φόρμα με πεδιά τα κριτήρια αναζήτησης και action την μέθοδο Index του routeController, η οποία έχει τον εξής κώδικα:

```
if(isset($_POST['Route'])){//an exei ginei search provalei ta apotelesmata
        $model=new Route('search');
        $model->attributes=$_POST['Route'];
        $dataProvider=$model->quickSearch();
.....
$this->render('index',array(dataProvider'=>$dataProvider,));
```

Και η μέθοδος quickSearch() του Route model είναι αυτή που κάνει την αναζήτηση και επιστρέφει τον dataProvider που θα εμφανίσει τις διαδρομές που βρέθηκαν. Ο κώδικας της quickSearch() είναι ο εξής:

```
public function quickSearch(){
        $criteria=new CDbCriteria;
        $criteria->with=(array ('start','dest'));
        $criteria->compare('type',$this->type);
        $criteria->compare('start_id',$this->start_id,TRUE);
        $criteria->compare('dest_id',$this->dest_id,TRUE);
        if ($this->days2_range && $this->date!=null)
                $criteria->addBetweenCondition('date'
                                ,$this->dateDistanceDays($this->date,'-2')
```

```
                                        ,$this->dateDistanceDays($this->date,'+2')
                );
        else
                $criteria->compare('date',$this->fDateForDb($this-
        >date,'d/m/y'),true);

        return new CActiveDataProvider($this, array(
                'criteria'=>$criteria,
                'sort'=>$this->viewRoutesDataProviderSort
               ));
        }
```

12.8 Δημιουργία Control Panel

Θα δημιουργήσουμε ένα πανελ διαχείρησης χρηστών και διαδρομών με την χρήστη 2 GridView Widgets και το αποτέλεσμα θα είναι κάπως έτσι:

Panel Διαχείρησης

Στα πεδία αναζήτησης μπορείτε να εισάγεται κάποιον τελεστή σύγκρισης (<, <=, >, >=, <> ή =) στην αρχή κάθε πεδίου.

Διαδρομές

Εμφάνιση 1-8 από 8 αποτελέσματα.

ID	Είδος	Αφετερία	Προορισμός	Ημέρα	Ώρα	User	Δημιουργία	Κόστος	Ενέργειες
	Όλα								
1	Προσφορά	Πτολεμαΐδα	Θεσσαλονίκη	15/04/14	18:00	5	2013-11-24 23:06:22	30	
2	Ζήτηση	Θεσσαλονίκη	Ορεστιάδα	30/06/14	12:00	11	2014-01-09 18:13:37		
3	Προσφορά	Πτολεμαΐδα	Θεσσαλονίκη	20/05/14	16:00	2	2014-01-12 13:26:19	30	
5	Προσφορά	Θεσσαλονίκη	Νέο Ηράκλειο	25/06/14	19:00	1	2014-01-15 13:35:02	60	
6	Προσφορά	Θεσσαλονίκη	Πτολεμαΐδα	12/03/14	18:00	3	2014-01-16 14:05:43	50	
7	Προσφορά	Κοζάνη	Νέο Ηράκλειο	11/03/14	13:00	1	2014-02-13 19:21:44	40	
8	Ζήτηση	Καλλιθέα	Νέο Ηράκλειο	19/03/14	20:00	4	2014-02-14 19:37:24		
9	Προσφορά	Πτολεμαΐδα	Ορεστιάδα	19/03/14	13:00	7	2014-02-16 14:58:36	40	

Εικόνα 12.5 Το πάνελ διαχείρησης

Στον SiteController δημιουργούμε την actionAdmin με τον εξής κώδικα:

```
public function actionAdmin(){
     $routeModel=new Route('search');
     $routeModel->unsetAttributes();
     $userModel=new User('search');
     $userModel->unsetAttributes();
     if(isset($_GET['Route']))
          $routeModel->attributes=$_GET['Route'];
     if(isset($_GET['User']))
          $userModel->attributes=$_GET['User'];
     $this->render('admin',
```

```
     array('routeModel'=>$routeModel,
           'userModel'=>$userModel));
}
```

Ενώ η view admin εμφανίζει 2 DataGrid widgets. Ένα για τις διαδρομές και ένα για τους χρήστες.

Για να έχουν πρόσβαση μόνο οι διαχειριστές στην σελίδα αυτή, ορίζουμε την μέθοδο accessRules() του SiteController ως εξής:

```
public function accessRules(){
     return array(
          array('allow',
               'actions'=>array('admin'),
               'expression'=>'isset(Yii::app()->user->type)
&&
                                   Yii::app()->user->type==0',
          ),
}
```

Ο κανόνας αυτός ορίζει να επιτρέπονται στην action admin μόνο οι χρήστες με type==0 που είναι η διαχειριστές.

12.9 Κυρίως Menu

Ο κώδικας δημιουργίας του menu που βρίσκεται στο layout main.php είναι ο εξής:

```
$this->widget('zii.widgets.CMenu',array(

'items'=>array(
  array('label'=>'Αρχική', 'url'=>Yii::app()->homeUrl),

  array('label'=>'Σχετικά',
        'url'=>array('/site/page',
        'view'=>'about')),

  array('label'=>'Επικοινωνία',
        'url'=>array('/site/contact')),

  array('label'=>'Διαδρομές',
        'url'=>array('/route/index')),

  array('label'=>'Χρήστες',
        'url'=>array('/user/index'),
        'visible'=>!Yii::app()->user->isGuest,),

  array('label'=>'Σύνδεση',
```

```
        'url'=>array('/login'),
        'visible'=>Yii::app()->user->isGuest,
        'itemOptions'=>array('style'=>'float:right;')),

  array('label'=>'Εγγραφή',
        'url'=>array('/register'),
        'visible'=>Yii::app()->user->isGuest,
        'itemOptions'=>array('style'=>'float:right;')),

  array('label'=>'Αποσύνδεση ('.current(explode('@',
isset(Yii::app()->user->email)? Yii::app()->user->email
:'')).')',//εφανιζει το πρώτο μέρος του email
        'url'=>array('/logout'),
        'visible'=>!Yii::app()->user->isGuest,
        'itemOptions'=>array('style'=>'float:right;')),

array('label'=>'AdminPanel',
      'url'=>array('/admin'),
      'visible'=>isset(Yii::app()->user->type) &&
                      Yii::app()->user->type==0,
      'itemOptions'=>array('style'=>'float:right;')),

array('label'=>'Προφίλ',
      'url'=>array('/profile'),
      'visible'=>!Yii::app()->user->isGuest,
      'itemOptions'=>array('style'=>'float:right;')),
),
));
```

Βιβλιογραφία

1. http://www.w3schools.com
2. http://en.wikipedia.org/wiki/HTML
3. http://en.wikipedia.org/wiki/Common_Gateway_Interface
4. http://www.php.net/manual/en/history.php.php
5. http://www.mysql.com
6. http://www.yiiframework.com
7. http://www.yiiframework.com/wiki
8. “The Definitive Guide to Yii 1.1”, Self-published, Qang Xue and Xiang Wei Zhuo, 2008-2012.
9. “The Yii Book by Larry Ullman”, Self-published, http://yii.larryullman.com, 2013, Revision: 0.61
10. “Yii 1.1 Application Development Cookbook”, Alexander Makarov, 2011.