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

Πτυχιακή εργασία

Ανάπτυξη διαδικτυακής εφαρμογής πώλησης κουπονιών (groupon clone) με χρήση PHP, MySQL και JavaScript

Της φοιτήτριας:
Αντωνοπούλου Άννα
Αρ. Μητρώου: 608

Επιβλέπων καθηγητής:
Διαμαντάρας Κων/νος

Θεσσαλονίκη 2012

Στην οικογένειά μου

ΠΡΟΛΟΓΟΣ

Η εφαρμογή αυτή δημιουργήθηκε το ακαδημαϊκό έτος 2011-2012 από την Αντωνοπούλου Άννα, φοιτήτρια του τμήματος Πληροφορικής του ΑΤΕΙ, υπό την εποπτεία του καθηγητή κ. Κωνσταντίνο Διαμαντάρα, στο πλαίσιο εκπόνησης πτυχιακής εργασίας με τίτλο: Ανάπτυξη διαδικτυακής εφαρμογής πώλησης κουπονιών (groupon clone) με χρήση PHP, MySQL και JavaScript.
Η ιδέα του θέματος προέκυψε όταν το 2011 άρχισαν να «ξεφυτρώνουν» όλο και περισσότερα eshop προσφορών (deal sites γνωστά και ως groupon clone)

Περίληψη

Στο πλαίσιο της πτυχιακής αυτής, θα γίνει σχεδίαση και υλοποίηση διαδικτυακής εφαρμογής - eshop πώλησης κουπονιών προσφοράς, με χρήση των τεχνολογιών HTML, CSS, PHP, MySQL και JavaScript. Κατά τη διαχείριση της εφαρμογής θα εισάγονται στο σύστημα οι προσφορές - κουπόνια που θα βάζουν οι διάφορες εταιρίες, ανά πόλη. Θα δηλώνονται οι διευθύνσεις των καταστημάτων που θα συμμετέχουν στην προσφορά. Θα εισάγεται στο σύστημα το προϊόν της προσφοράς, τα διαθέσιμα τεμάχια αν υπάρχουν, καθώς και το μέγιστο αριθμό κουπονιών που θα μπορεί να αγοράζει ο κάθε επισκέπτης του eshop. Ο επισκέπτης του eshop θα επιλέγει αρχικά την πόλη που τον ενδιαφέρει να δει τις προσφορές και θα βάζει στο καλάθι του την προσφορά /ες που τον ενδιαφέρει. Για κάθε προσφορά θα μπορεί να βάζει τα ονόματα των δικαιούχων για το κάθε κουπόνι που θα εκδοθεί. Στο τέλος της παραγγελία θα εκτυπώνονται θα κουπόνια που θα εκδίδονται για τον κάθε δικαιούχο. Στο τέλος, ο διαχειριστής του συστήματος θα μπορεί αφού έχει λήξει η προσφορά να παίρνει review με όλα τα κουπόνια και να τα αποστέλλει στην εταιρία.
Το σύστημα θα υλοποιηθεί με τη χρήση PHP/MySQL και τη βοήθεια του DOTCMS το εργαλείο διαχείρισης που έχει υλοποιήσει η εταιρεία NEWMEDIA.

Λέξεις κλειδιά: PHP, MySQL, HTML, CSS, Javascript, jQuery.

Abstract

During this thesis, a web - eshop application of selling offering coupons will be designed and implemented by using the HTML, CSS, PHP, MySQL and JaveScript technologies. During the administration of the application, the offers-coupons will be entered the system from the respective per city, companies. The addresses of the shops which will participate in the coupon offering will be registered in the system. The following information will be inserted into the system: The products and their availability -if exists- and also the maximum number of coupons an e-shop visitor can buy. The eshop visitor will initially choose the city of interest, see the offers and add the offers in the basket. For each offer, the visitor can place the name(s) of the beneficiary person(s) under which the offers will be issued. At the end of the order, the coupons will be printed for each of the beneficiary. Finaly, when the offer is expired, the administrator can take a review with all the coupons and send it to the company.
This system will be implemented by using PHP/MySQL and supported by a specialized administration framework tool called DOTCMS, which is developed by NEWMEDIA company.

Keywords: PHP, MySQL, HTML, CSS, Javascript, jQuery.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 5

ΚΕΦΑΛΑΙΟ 6

Ευρετήριο Σχημάτων και Πινάκων

ΕΙΣΑΓΩΓΗ

Τα τελευταία χρόνια, οι ραγδαίες εξελίξεις στην οικονομία έχουν στρέψει τον κόσμο στην αναζήτηση ολοένα και φτηνότερων λύσεων για την αγορά προϊόντων και υπηρεσιών. Αυτή η τάση της αγοράς επεκτάθηκε όπως ήταν αναμενόμενο και στις αγορές μέσω internet. Για τον σκοπό αυτό η αγορά οδηγήθηκε στη δημιουργία ηλεκτρονικών καταστημάτων που δεν πωλούν προϊόντα αλλά εκπτωτικά κουπόνια για προϊόντα και υπηρεσίες τρίτων ή αλλιώς social buying.

Στην Ελλάδα η ανάπτυξη και η εξάπλωση τέτοιων ηλεκτρονικών καταστημάτων ήταν ραγδαία. Ξεκίνησε το 2010 όταν ήρθε στην Ελλάδα η εταιρεία Groupon η οποία διαθέτει social buying ιστοσελίδες σε 500 πόλεις παγκοσμίως. Στην Ελλάδα, στο τέλος του 2012, υπάρχουν πάνω από 100 τέτοιες ιστοσελίδες.

Σε αυτή τη πτυχιακή εργασία αναπτύσσεται μια εφαρμογή λογισμικού η οποία έχει στόχο την δημιουργία ηλεκτρονικών καταστημάτων ηλεκτρονικών κουπονιών.

Ως μοντέλο για την υλοποίηση της εφαρμογής χρησιμοποιήθηκε το όνομα MVAdeals.

Τα κεφάλαια που ακολουθούν χωρίζονται σε δύο μέρη:

Στο πρώτο μέρος της πτυχιακής εργασίας αναλύονται τι είναι και πως λειτουργεί ένα deal site κι έπειτα αναπτύσσονται οι διάφορες τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής.

Στο δεύτερο μέρος αναπτύσσεται η ανάλυση της εφαρμογής καθώς και τα κυριότερα κομμάτια του κώδικα καθώς και το εγχειρίδιο χρήσης της.

ΜΕΡΟΣ ΠΡΩΤΟ

ΚΕΦΑΛΑΙΟ 1 - DEAL SITES

1.1Εισαγωγή

Σε μια εποχή που η οικονομική κρίση επηρεάζει όχι μόνο τον καταναλωτή αλλά και τις επιχειρήσεις το καλύτερο όπλο της αγοράς είναι η **διαφήμιση και η προώθηση της**. Αυτό είναι η μεγαλύτερη κινητήρια δύναμη όλα τα χρόνια που ο άνθρωπος μέσα από την καταναλωτική του ανάγκη προσπαθεί να καλύψει τα πρέπει και τα θέλω του σε όλη τη διάρκεια της ζωής του. Αναζητώ κάτι, στην καλύτερη τιμή της αγοράς αλλά θα προτιμούσα να το πάρω ακόμα φθηνότερα για να εξοικονομήσω χρήματα.

Από την πλευρά της, μια επιχείρηση για να σταθεί στα πόδια της και να ορθοποδήσει χρειάζεται δουλειά που σημαίνει πουλάω – αγοράζεις. Αυτό δεν γίνετε χωρίς τον καταναλωτή όμως. Για να μπορέσεις ως εταιρεία να προσελκύσεις κοινό, θα πρέπει πέρα από το **να προσφέρεις τις κατάλληλες υπηρεσίες ή προϊόντα** σε ανθρώπους που τα έχουν ανάγκη και ενδιαφέρονται γι′ αυτά, **να σε γνωρίσουν**. Αυτό γίνετε με έναν τρόπο που αντικρίζουμε συνεχώς γύρω μας **ΤΗ ΔΙΑΦΗΜΙΣΗ.** Η υπερφόρτωση του μυαλού μας με πληροφορίες που αφομοιώνονται αυτόματα και μας ωθούν να θέλουμε να μπούμε σε ένα μαγαζί και να αγοράσουμε.

Τα χρήματα είναι πολλά για μία διαφημιστική καμπάνια. Ένα τύπωμα και μοίρασμα φυλλαδίων σημαίνει επιπλέον έξοδα χωρίς πάντα το επιθυμείτο αποτέλεσμα καθώς δεν καλύπτετε αρκετά μεγάλο ποσοστό πληθυσμού. Τη λύση σε αυτό το πρόβλημα μπορούν να δώσουν εξυπηρετώντας και την εταιρεία αλλά και τον καταναλωτή, sites εταιρειών που προσφέρουν διαφήμιση μέσω internet, προωθώντας τις προσφορές μιας επιχείρησης από το διαδίκτυο, τα γνωστά **DEAL Sites**.

Τα **deal sites** *–ή social bying ή group buying sites* όπως αποκαλούνται διεθνώς- είναι μία ιδέα που λειτουργεί με μεγάλη επιτυχία στο εξωτερικό από το 2008. Είναι ο νούμερο ένα προορισμός μετά τα κοινωνικά δίκτυα όπου επισκέπτονται καθημερινά εκατομμύρια Ήταν θέμα χρόνου το concept της ηλεκτρονικής «ομαδικής» αγοράς να φτάσει στην Ελλάδα.

1.2 Η ιδέα

Τι πιο καλή «κίνηση» για την αύξηση του πελατολογίου σε μια εταιρεία από το να κάνει μία μεγάλη προσφορά σε ένα προϊόν ή υπηρεσία ώστε να προσελκύσει τα βλέμματα και να έρθει κοντά της ο καταναλωτής, να την γνωρίσει και να τον κερδίσει ως μόνιμο πελάτη ή συνεργάτη εφόσον μείνει ικανοποιημένος από την εξυπηρέτηση της προσφοράς.

Μια εταιρεία, μέσα από τα sites αυτά, τα οποία αναλαμβάνουν την προώθηση της προσφοράς σε χιλιάδες δυνητικούς πελάτες, κατορθώνει γρήγορα να γίνει γνωστή σε ένα εκπληκτικά μεγαλύτερο μέρος της αγοράς, καθώς το internet το χρησιμοποιούν περισσότεροι από αυτούς που θα λάβουν ή θα πετάξουν ένα παραδοσιακό διαφημιστικό φυλλάδιο.

Το concept πίσω από αυτά τα sites είναι **η εξασφάλιση μιας προσφοράς με σημαντική έκπτωση για κάποιο προϊόν ή υπηρεσία**. Το site προβάλλει τουλάχιστον μία καθημερινή προσφορά και υπόσχεται στον προμηθευτή έναν ελάχιστο αριθμό αγοραστών, που είναι η προϋπόθεση για την έκπτωση. Αν ένας τέτοιος αριθμός ανθρώπων δηλώσει online πως ενδιαφέρεται για την προσφορά, αυτή ενεργοποιείται, και κρατούνται τα αντίστοιχα ποσά από τους αγοραστές. Αν δεν συμπληρωθούν οι προβλεπόμενοι ενδιαφερόμενοι, η προσφορά ακυρώνεται και δεν κρατούνται χρήματα από όσους έχουν δώσει τα στοιχεία τους.

Το σημαντικότερο όλων αυτών είναι ότι η επιχείρηση **δεν χρειάζεται να πληρώσει τίποτα για την δημιουργία και ανάρτηση της προσφοράς.** Η επιχείρηση κερδίζει τον πελάτη και η εταιρεία που προωθεί την προσφορά σας κρατά ένα μικρό ποσοστό από την εξαργύρωση των κουπονιών της προσφοράς. Δεν ήταν επιτυχής η προσφορά και δεν πούλησε για κάποιο λόγο κουπόνια; Η επιχείρηση δεν πληρώνει τίποτα**!** Αντί αυτού όμως έχει κερδίσει μία **ΔΩΡΕΑΝ**

προβολή στο διαδίκτυο και όταν κάποιος αναζητήσει τις υπηρεσίες ή τα προϊόντα που προσφέρει ακόμα και μετά τη λήξη της προσφοράς, θα προσπαθήσει να επικοινωνήσει μαζί της, είτε γιατί θα αναζητήσει αυτό που ψάχνει σε κάποια μηχανή αναζήτησης και θα εμφανιστείτε μπροστά του, είτε **γιατί θα έχει δει την προσφορά της και σίγουρα θα ξέρει το αντικείμενο με το οποίο ασχολείστε και θα είναι αυτό που ψάχνει.**

Τι κερδίζει η κάθε πλευρά από αυτήν την ιστορία; Οι αγοραστές κερδίζουν το προφανές: **διόλου αμελητέα έκπτωση**. Με προσφορές που σε ορισμένες περιπτώσεις ξεπερνούν σε ποσοστό έκπτωσης το 90%, οι αγοραστές των ηλεκτρονικών κουπονιών αποκτούν προϊόντα σε τιμές που ούτε θα μπορούσαν να φανταστούν στο συμβατικό εμπόριο.

Με άλλα λόγια, **κερδίζει ο αγοραστής, κερδίζει και η επιχείρηση**, παρά το μεγάλο ποσοστό έκπτωσης, καθώς εξασφαλίζεται μια ομαδική «πελατεία». Όσο κερδίζει η επιχείρηση τόσο κερδίζει και το deal site κάτι που θα του αποφέρει και την βιωσιμότητά του στον χώρο.

1.3 Τι είδους προϊόντα υπάρχουν σε προσφορά

Με ένα σύντομο ηλεκτρονικό «ξεφύλλισμα» των sites ημερήσιων προσφορών, θα βρούμε κυρίως προϊόντα και υπηρεσίες αισθητικής (αποτρίχωση, μασάζ, αισθητικές θεραπείες), σημαντικές εκπτώσεις σε εστιατόρια καθώς και μεγάλες εκπτώσεις για διακοπές, ταξίδια, ξενοδοχεία.

Στο μέλλον προβλέπεται η ένταξη κι άλλων ειδών προσφερόμενων προϊόντων που θα έχουν ζήτηση, όπως είδη supermarket (τρόφιμα, απορρυπαντικά).

Τα κριτήρια με τα οποία επιλέγεται η προσφορά της ημέρας είναι: Η αναγνωρισιμότητα της επιχείρησης, η φυσική της τοποθεσία, το website της και φυσικά η δελεαστικότητα της προσφοράς. Ένα στοιχείο ακόμα που μπορούμε να προσθέσουμε στα κριτήρια είναι το κριτήριο της πρωτοτυπίας. Μπορεί ένα προϊόν που δεν το περιμένεις να έχει μεγάλη απήχηση στο αγοραστικό κοινό. Επίσης, ένα σημαντικό κριτήριο είναι η **ανάγκη για διασκέδαση** και η απόλαυση της ζωής γι' αυτό και αυξάνονται και οι προσφορές στην εστίαση και στις διακοπές.

1.4 Ποιοι αγοράζουν

Σύμφωνα με τους αριθμούς, το γυναικείο κοινό είναι αυτό που κάνει τις περισσότερες αγορές. Αυτό αποδεικνύει ότι οι γυναίκες είναι πιο εξοικειωμένες με το Internet, τις online συναλλαγές και το μοντέλο αγορών που προτείνουν τα deal sites. Επίσης, η πλειοψηφία των εγγεγραμμένων χρηστών στα site αποτελείτε από γυναίκες, όπως και ένα ποσοστό μεγαλύτερο από 60% των φίλων των group αυτών των site στα facebook.

Γενικότερα, αυτοί που προσχώρησαν στην νέα μόδα αγορών αυξάνονται με ταχύτατους ρυθμούς, ανεξαρτήτως φύλου, και όλα δείχνουν ότι με την συνδρομή των κοινωνικών δικτύων τύπου Facebook οι ενδιαφερόμενοι θα πολλαπλασιαστούν. Υπάρχουν site που προσφέρουν πχ πίστωση 1 ευρώ για κάθε φίλο που προσκαλείς και πραγματοποιεί κάποια αγορά μέσα σε 3 μέρες.

Όλοι οι υπεύθυνοι των σχετικών sites τονίζουν πως η **online αγορά είναι απολύτως ασφαλής για τους καταναλωτές.** Η διασφάλιση ενός τέτοιου πλαισίου είναι καθοριστικής σημασίας για το κοινό το οποίο δεν είναι απόλυτα εξοικειωμένο με τις online αγορές, γεγονός που μπορεί να το καθιστά καχύποπτο απέναντι σε τέτοιες ιδέες.

Ο Έλληνας καταναλωτής φαίνεται να έχει αλλάξει συμπεριφορά τα τελευταία χρόνια. Ο καταναλωτής εμπιστεύεται πλέον τις online συναλλαγές, γεγονός που βοηθάει και τα deal sites να αναπτυχθούν.

1.5 Η επιτυχία της ιδέας

Με τα group buying sites να εμφανίζονται το ένα μετά το άλλο και να αυξάνουν τις επιλογές του κοινού, η ιδέα έχει βρει πρόσφορο έδαφος και **τέτοιου είδους αγορές γίνονται σταδιακά μόδα**. Στα περισσότερα site αυτού του είδους υπάρχει μεγάλη λίστα αναμονής από προμηθευτές, οι οποίοι θέλουν να συνεργαστούν στο νέο επιχειρηματικό μοντέλο.

Ενδεικτική της επιτυχίας του project είναι **η είσοδος στο «παιχνίδι» αρκετών online καταστημάτων**, τα οποία, αντιλαμβανόμενα το «ρεύμα» του concept εντάσσουν ημερήσιες προσφορές στην καθημερινή τους δραστηριότητα.

Στην ένταση της δημοσιότητας του φαινομένου συντελούν άλλωστε και τα sites που εμφανίστηκαν, τα οποία **συγκεντρώνουν καθημερινά τις προσφορές από όλα τα deal sites** της ελληνικής ηλεκτρονικής αγοράς. Μαζεύουν καθημερινά όλες τις προσφορές από τα ελληνικά deal sites σε μία σελίδα, και έτσι ο χρήστης μπορεί με μια γρήγορη ματιά να ενημερώνεται για όλες τις προσφορές που υπάρχουν.

1.6 Ανταγωνισμός

Η λειτουργία εκατό τουλάχιστον τέτοιων ιστοσελίδων αυτή τη στιγμή δημιουργεί μεγάλο ανταγωνισμό, αλλά έχει εδραιώσει και το νέο αυτό αγοραστικό πρότυπο. Το σίγουρο είναι ότι ο ανταγωνισμός θα ωφελήσει το κοινό, εφόσον οι υπεύθυνοι για φροντίζουν για συνεχώς καλύτερες προσφορές. Το κάθε site, άλλωστε, ψάχνει να εφαρμόσει τα δικά του επιχειρηματικά πλεονεκτήματα. Οι υπεύθυνοι προσπαθούν κάνοντας έρευνες και ρωτώντας τους ίδιους τους καταναλωτές να βρουν ποια προϊόντα και υπηρεσίες τους ενδιαφέρουν να βλέπουν ώστε να ξέρουν σε ποιες επιχειρήσεις να στοχεύσουν.

Ενδεικτικές προσφορές που μπορούμε να δούμε σε αυτά τα sites είναι:

- **€129 από €300** για Μία Θεραπεία με Ενέσιμο μπότοξ Full Face (για Ρυτίδες)
- **€29** από **€70** για Έναν Βιολογικό Καθαρισμό Εσωτερικού Αυτοκινήτου
- **€6** από **€12** για Ένα Κιλό Χειροποίητα Μελομακάρονα.

Ανάλογες προσφορές βγαίνουν στον αέρα κάθε μέρα, λαμβάνοντας υπόψη την περιοχή σας. Σιγά-σιγά, μάλιστα, η δράση των deal sites εξαπλώνεται και σε άλλες πόλεις ανά την Ελλάδα.

1.7 Επίλογος

Τα deal sites έχουν γίνει πλέον αναπόσπαστο μέρος της καθημερινότητας των καταναλωτών και ήδη δείχνουν μία σημαντική αλλαγή του τρόπου που οι έλληνες καταναλωτές βλέπουν τις online αγορές. Τα deal sites είναι κι αυτά αποτέλεσμα της αλλαγής που έχει γίνει στην όλη φιλοσοφία του internet και των υπηρεσιών που βασίζονται στο web με την εξέλιξή του από web 1.0 σε web 2.0.

ΚΕΦΑΛΑΙΟ 2

WEB 2.0

2.1 Εισαγωγή

Ένα από τα πλέον δημοφιλή θέματα τον τελευταίο καιρό στο χώρο του διαδικτύου και της πληροφορικής είναι η εξέλιξη από το παραδοσιακού WEB σε αυτό που έχει επικρατήσει να ονομάζεται WEB 2.0. Η χρήση του διαδικτύου σε ολοένα και περισσότερες ανθρώπινες δραστηριότητες προσδίδει ιδιαίτερη σημασία σε αυτή την εξέλιξη. Οι χρήστες, είτε πρόκειται για ιδιώτες, είτε για επιχειρήσεις, οργανισμούς, εκπαιδευτικά ιδρύματα, κλπ. σταδιακά ενημερώνονται για τα στοιχεία και τις τεχνολογίες που συνιστούν το WEB 2.0 και επωφελούνται από τα πλεονεκτήματά του.

2.2 Τι είναι το WEB 2.0

Το Web 2.0 είναι μία φράση που πρωτοχρησιμοποιήθηκε από την εταιρία O'Reilly Media το 2004 και αναφέρεται σε ένα όραμα, ή σε μία πρόταση για μία δεύτερη γενιά υπηρεσιών που βασίζονται στο Web – όπως τα social networking sites, τα wikis, τα εργαλεία επικοινωνίας κλπ – που δίνουν έμφαση στην online συνεργασία μεταξύ των χρηστών. Η O'Reilly Media, σε συνεργασία με την MediaLive International, χρησιμοποίησε τη φράση σαν τίτλο για μια σειρά συνεδρίων, και από το 2004 μερικοί τεχνικοί καθώς και παράγοντες της αγοράς έχουν υιοθετήσει τον όρο. Το ακριβές νόημα παραμένει ανοιχτό σε αναλύσεις και ορισμούς και πολλοί τεχνολογικοί παράγοντες, όπως ο Tim Berners Lee, αναρωτιούνται το κατά πόσο ο όρος έχει καθόλου νόημα!

Ο τελευταίος ορισμός του Web 2.0, σύμφωνα με τον Tim O'Reilly (ιδρυτή της O'Reilly Media) είναι ο ακόλουθος:

«Το Web 2.0 είναι η επιχειρηματική επανάσταση στη βιομηχανία των υπολογιστών που συντελείται από την μετακίνηση στο Internet σαν πλατφόρμα, και μια προσπάθεια να κατανοήσουμε τους όρους της επιτυχίας αυτής της νέας πλατφόρμας. Ο βασικότερος ανάμεσα στους όρους είναι ο εξής: Δημιουργήστε

εφαρμογές που θα οδηγήσουν την απήχηση του δικτύου να είναι μεγαλύτερη και να φέρει περισσότερο κόσμο να το χρησιμοποιεί»

Αν και ο όρος WEB 2.0 δίνει την αίσθηση ότι αποτελεί μια νέα έκδοση του WEB, ουσιαστικά δεν πρόκειται για κάποιο καινούργιο πρωτόκολλό του αλλά για αλλαγές στον τρόπο που χρησιμοποιούνται ήδη υπάρχουσες τεχνολογίες και στον τρόπο που οι σχεδιαστές πληροφοριακών συστημάτων και οι χρήστες χρησιμοποιούν το διαδίκτυο. Η κυρίαρχη τάση είναι να χρησιμοποιείται ως το μέσο (**πλατφόρμα**) πάνω στο οποίο θα τρέχουν οι εφαρμογές και υπηρεσίες, πολλές από τις οποίες μέχρι τώρα έτρεχαν τοπικά στους Η/Υ.

2.3 Εφαρμογές και παραδείγματα

Το Web 2.0 υπάρχει γιατί προσφέρει υπηρεσίες, εφαρμογές, εργαλεία και λειτουργίες που είναι καινοτόμα και διευκολύνουν τους χρήστες. Άλλωστε για αυτό το λόγο όταν υλοποιήθηκαν έγιναν αμέσως αποδεκτά. **Μερικές από τις κυριότερες περιγράφονται παρακάτω :**

- Τα **wikis** είναι ιστοσελίδες των οποίων το περιεχόμενο μπορεί να διαμορφωθεί από τον ίδιο τον χρήστη με τρόπο μάλιστα πολύ απλό. Κάθε φορά που ο χρήστης τροποποιεί τη σελίδα η προηγούμενη έκδοσή της εξακολουθεί να είναι διαθέσιμη, ακόμη και να επαναφερθεί. Τα wikis είναι αρκετά διαδεδομένα σαν μέσο συλλογικής εργασίας πάνω σε κάποιο θέμα. Ακόμη, χρησιμοποιούνται και μέσα στις επιχειρήσεις, στους οργανισμούς, στις υπηρεσίες. Οι εργαζόμενοι πιο εύκολα ενημερώνονται για ό,τι συμβαίνει στην εταιρεία έχοντας τα wikis σαν σελίδες προόδου των εργασιών. Το πιο χαρακτηριστικό παράδειγμα είναι η **wikipedia**, που αποτελεί μια ηλεκτρονική εγκυκλοπαίδεια στην οποία υπάρχουν εκατομμύρια άρθρα με ορισμούς και πληροφορίες σε πολλές γλώσσες. Η σύνταξη της γίνεται από τους ίδιους τους χρήστες, αφού ο καθένας μπορεί να γράψει ένα άρθρο, να προσθέσει κάτι σε αυτά που ήδη υπάρχουν. Η δημοτικότητά της αυξάνει συνεχώς και βάση επισκέψεων βρίσκεται μέσα στα 10 πιο δημοφιλή sites παγκοσμίως .
- **Τα ιστολόγια ή blogs** είναι ιστοσελίδες που φιλοξενούν διάφορες απόψεις για πολλά και διαφορετικά πράγματα, πληροφορίες, προσωπικές καταχωρήσεις **(posts)**, φωτογραφίες κ.τ.λ. Οι καταχωρήσεις ταξινομούνται με χρονολογική

σειρά με την πιο πρόσφατη να εμφανίζεται πρώτη. Ξεκινούν με μία άποψη ή ένα σχόλιο του ανθρώπου που το δημιουργεί για κάποιο θέμα πολιτικό, κοινωνικό, ιατρικό, της καθημερινότητας. Είναι πολύ διαδεδομένα ανάμεσα στους χρήστες γιατί ο οποιοσδήποτε μπορεί να μπει σε ένα ιστολόγιο και να γράψει ό,τι σχόλιο θέλει, ξεκινώντας μια δημόσια διαδικτυακή συζήτηση, στην οποία μπορεί να συμμετέχει πάλι ο οποιοσδήποτε.

- Ιστοσελίδες όπου μπορεί κανείς να *ανεβάσει* βίντεο, τραγούδια, ταινίες, φωτογραφίες και να κάνει γνωριμίες μέσα από αυτές **(youtube.com, facebook.com)**.
- **Tagging** ονομάζουμε τη δυνατότητα χαρακτηρισμού με σημασιολογικές λέξεις (tags), ιστοσελίδων, φωτογραφιών, κειμένων, γενικά του διαδικτυακού περιεχομένου. Από εδώ προέρχεται και ο όρος social bookmarking. Οι προσωπικές προτιμήσεις και ενδιαφέροντα των χρηστών για οτιδήποτε τους ενδιαφέρει μπορούν να ταξινομηθούν και να είναι διαθέσιμα και στους υπολοίπους. Για παράδειγμα, πολύ δημοφιλής είναι η ιστοσελίδα Del.icio.us όπου οι χρήστες παρουσιάζουν και χαρακτηρίζουν με tags τις αγαπημένες τους ιστοσελίδες (bookmarks), αλλά και το Flickr όπου οι χρήστες μοιράζονται και χαρακτηρίζουν τις φωτογραφίες που *ανεβάζουν* . Έτσι, από τη μια οι χρήστες οργανώνουν τα δεδομένα πολύ καλύτερα και από την άλλη κοινωνικοποιούνται, γνωρίζοντας και άλλα άτομα μέσα από τους κοινούς χαρακτηρισμούς για τις φωτογραφίες τους.
- Ο συνδυασμός και η χρήση δεδομένων και εφαρμογών από διαφορετικές ιστοσελίδες σε μια, είναι γνωστός ως **mash-up**. Υλοποιούνται μέσω ανοιχτών interfaces προγραμματισμού εφαρμογών (open APIs'- Application Programming Interfaces) και βοηθούν στη βελτίωση της λειτουργικότητας των ιστοσελίδων. Σαν παράδειγμα μπορούμε να αναφέρουμε τις ιστοσελίδες ξενοδοχείων, που ενσωματώνουν χάρτες από άλλη υπηρεσία (π.χ. Google maps) για να δείξουν στους χρήστες ποια είναι η ακριβής τοποθεσία του παρέχοντας έτσι πληρέστερη πληροφόρηση.
- Τα **RSS (Real Simple Syndication) feeds**, έχουν ως λειτουργία να προσφέρουν στους χρήστες νέες πληροφορίες από διάφορες ιστοσελίδες, τη στιγμή που δημοσιεύονται, χωρίς να χρειάζεται να τις επισκεφθούν. Η ενημέρωση αύτη γίνεται στον browser του Η/Υ του χρήστη ή στο κινητό του ή στο PDA6 του κ.τ.λ. Έτσι έχουμε μια πιο άμεση σχέση με το διαδίκτυο.

2.4 Τεχνολογία

Σε αυτή την ενότητα θα ασχοληθούμε συνοπτικά με τις σημαντικότερες τεχνολογίες που χρησιμοποιεί το Web 2.0 κάνοντάς το να διαφέρει ως προς τον τρόπο λειτουργίας και παρουσίασης των ιστοσελίδων σε σχέση με το Web 1.0:

- **Πλούσια και διαδραστικά interfaces χρηστών (RIA)** που χρησιμοποιούν τεχνολογία Flash, Javascript και την Ajax, που αντιπροσωπεύει την τάση του Web 2.0 για καλύτερη εκμετάλλευση του δικτύου. Αντί να φορτώνεται ξανά ολόκληρη η ιστοσελίδα, ανανεώνονται μόνο τα νέα δεδομένα που αλλάζουν όσο ο χρήστης βρίσκεται ή επανέρχεται σε αυτή. (π.χ. Στο Gmail ο υπολογισμός του διαθέσιμου αποθηκευτικού χώρου ανανεώνεται σε πραγματικό χρόνο και από όλη τη σελίδα αλλάζει μόνο αυτός).
- **Χρήση CSS (Cascading Style Sheets)** για να διαχωρίζονται τα δεδομένα καθαρής πληροφορίας από τα δεδομένα μορφοποίησης σε μια ιστοσελίδα. Αυτό, πέρα από την οικονομία στο εύρος ζώνης του δικτύου, προσφέρει και ευελιξία στον τρόπο παρουσίασης των δεδομένων, αφού ο χρήστης βλέπει τα δεδομένα ανάλογα με το CSS που ο ίδιος έχει (π.χ. τα ίδια δεδομένα ανάλογα με το CSS μπορούν να παρουσιαστούν σε οθόνη υπολογιστή, κατευθείαν σε εκτυπωτή, σε μορφή ανάγνωσης για τυφλούς ή και να μετατραπούν σε φωνή και με χρήση κατάλληλου λογισμικού).
- **Ελαφρά πρωτόκολλα δικτύου REST και SOAP** που χρησιμοποιούν απλές εντολές HTTP (get, post, put) για ανάκτηση δεδομένων από τους servers.
- **Αρχιτεκτονικές SOA (Service Oriented Architecture)** που επιτρέπουν το διαμοιρασμό και την επαναχρησιμοποίηση υπηρεσιών-εφαρμογών από διαφορετικά προγράμματα λογισμικού και SaaS (Software as a Service) όπου οι εφαρμογές είναι εγκατεστημένες σε κεντρικό server στο δίκτυο και οι χρήστες τις χρησιμοποιούν μέσω browser ανεξαρτήτως Η/Υ, τόπου, και χρονικής στιγμής.
- **Χρήση ανοικτού λογαριασμού** (Linux σαν λειτουργικό σύστημα, Apache σαν web server, MySQL σαν βάση δεδομένων, PHP, Pearl, σαν γλώσσες προγραμματισμού).

- **Χρήση σημασιολογικών δεδομένων και microformats** για να περιγράφεται η σημασία των δεδομένων που περιέχουν οι ιστοσελίδες. Με αυτόν τον τρόπο τοποθετούνται σε κατηγορίες και η αναζήτησή τους γίνεται πιο εύκολη και πιο αποδοτική.
- **Χρήση RSS feeds.**

2.5 WEB 2.0 και σύγχρονη επιχείρηση

Οι εφαρμογές και τα χαρακτηριστικά του Web 2.0 έχουν ήδη αρχίσει να επιδρούν στον τομέα των επιχειρήσεων. Οι επιπτώσεις επικεντρώνονται στην εσωτερική τους συγκέντρωση, στην εμπορική δραστηριότητα, και στην πολιτική τους προς τους καταναλωτές. Παρακάτω αναλύονται οι επιπτώσεις αυτές.

2.5.1 WEB 2.0 και marketing

Στο Web 1.0 οι στρατηγικές μάρκετινγκ των εταιριών στηρίζονταν κυρίως σε 3 τομείς :

α) στην άμεση προώθηση πληροφοριακού υλικού στον πελάτη μέσω του ηλεκτρονικού ταχυδρομείου

β) στη βελτιστοποίηση θέσεως της ιστοσελίδας της εταιρίας στα αποτελέσματα των Μηχανών Αναζήτησης του διαδικτύου, κατα την αναζήτηση πληροφοριών από τους χρήστες

γ) ορισμένες φορές στην ύπαρξη συμμαχιών , στην δημιουργία δηλαδή παραπομπών-συνδέσμων μεταξύ διαφορετικών ιστοσελίδων.

Με την έλευση του Web 2.0 παρουσιάζονται εκατοντάδες νέοι τρόποι να γίνει προσέγγιση του πελάτη, οι περισσότεροι από τους οποίους δημιουργούνται από τους ίδιους τους καταναλωτές, μέσω της χρήσης και διάδοσης του περιεχομένου των social media. Πολλές επιχειρήσεις και στελέχη μάρκετινγκ έχουν ήδη κατανοήσει πόσο σημαντικό είναι να αυξήσουν την παρουσία τους στην νέα αυτή διαδικτυακή πραγματικότητα.

2.5.2 Ενίσχυση και Προώθηση Εμπορικών Σημάτων

Μέσω των εφαρμογών του Web 2.0 αυξάνεται η ενημέρωση των καταναλωτών σχετικά με την ύπαρξη της εταιρίας ή των προϊόντων της. Συζητήσεις για την εταιρία πραγματοποιούνται στο διαδίκτυο με ή όχι την δική της συμμετοχή. Καθήκον και στρατηγική απόφαση της εταιρίας αποτελεί η συμμετοχή της στην συζήτηση αυτή με τρόπους που δίνουν έμφαση στα ενδιαφέροντα και τις ανάγκες των καταναλωτών – τροφοδοτώντας τους με πληροφορίες και επιδρώντας σε αυτούς με τρόπους που θα ενισχύσουν το εμπορικό της σήμα και θα οδηγούν στην διασφάλιση της αναγνωρισιμότητας της εταιρίας και του κύρους του εμπορικού της σήματος. Η επιλογή μιας εταιρίας ως θέμα συζήτησης αποτελεί επιτυχία για την εταιρία που έχει ξεκινήσει να αναμιγνύεται με το Web 2.0.

2.5.3 Ανάπτυξη Νέου Προϊόντος

Η συμβολή των χρηστών και καταναλωτών στο Web 2.0 δεν περιορίζεται μόνο στην δημιουργία και διακίνηση περιεχομένου. Η επιχείρηση μπορεί να αξιοποιήσει την άμεση ανατροφοδότηση που έχει από το καταναλωτικό κοινό και για την επίτευξη στόχων, όπως η ανάπτυξη νέων προϊόντων ή η βελτίωση των ήδη υπαρχόντων. Οι επιχειρήσεις θα πρέπει να δίνουν την ευκαιρία στους χρήστες να συμμετέχουν σε όλη την διαδικασία βελτίωσης ή ανάπτυξης ενός προϊόντος επιτρέποντας τους να εκφράσουν ελεύθερα την γνώμη τους σχετικά με τον επανασχεδιασμό, την αναβάθμιση, την λειτουργικότητα, κ.τ.λ. των προϊόντων και υπηρεσιών της εκμεταλλευόμενη όλη την πληροφορία που συσσωρεύεται στο Web 2.0.

2.5.4 Διαφήμιση

Η παραδοσιακή διαφήμιση των προϊόντων, μέσα από καταχωρήσεις σε έντυπο υλικό, ραδιόφωνο και φυσικά τηλεόραση, απαιτεί υψηλό κόστος και απευθύνεται σε ένα γενικό κοινό. Στην περίπτωση που δημιουργηθούν διαφορετικές εκδοχές της ώστε να στοχεύει σε πιο συγκεκριμένες κατηγορίες ατόμων, το κόστος αυξάνεται ακόμη περισσότερο. Από την άλλη, η διαδικτυακή διαφήμιση απαιτεί λιγότερα χρήματα για να υλοποιηθεί. Μέσω των εφαρμογών και της τεχνολογίας Web 2.0 μπορεί να γίνει ακόμη πιο στοχευμένη, εκμεταλλευόμενη της προσωπικές

προτιμήσεις των χρηστών που δημοσιοποιούνται από τις προηγούμενες αγορές τους, την κατάθεση των απόψεών τους και το social bookmarking. Με την ολοένα και μεγαλύτερη αύξηση των χρηστών του διαδικτύου προσφέρει εύκολη και άμεση πρόσβαση σε ένα τεράστιο αγοραστικό κοινό. Ένα χαρακτηριστικό παράδειγμα Web 2.0 διαφήμισης είναι αυτή της εταιρίας General Motors για το SUV Chevy Tahoe. Η εταιρία έδινε την ευκαιρία στους επισκέπτες της ιστοσελίδας της να δημιουργήσουν αυτοί τη διαφήμιση video για το συγκεκριμένο αμάξι, με κίνητρο κάποιο χρηματικό έπαθλο. Με τη λήξη του διαγωνισμού 629 χιλιάδες άτομα είχαν επισκεφθεί την ιστοσελίδα 22 χιλιάδες video είχαν ανέβει σε αυτή. Πέρα από αυτό, η δημιουργία και blog, στο οποίο συζητιούνταν τα πλεονεκτήματα και τα μειονεκτήματα του αμαξιού, βοήθησε την εταιρία να ξεχωρίσει και να δώσει έμφαση στα σημεία που υπερτερεί σε σχέση με τον ανταγωνισμό. Ακόμη και η αρνητική κριτική-διαφήμιση που έγινε, κυρίως για την ρύπανση που προκαλούν αμάξια τέτοιου τύπου, είχε τελικά αποτελέσματα στη δημοτικότητα του προϊόντος.

2.5.5 Φαινόμενο Long tail

Πολλές είναι οι επιχειρήσεις που εκμεταλλεύτηκαν το φαινόμενο αυτό επωφελούμενες τις τεχνολογίες του Web 2.0. Σύμφωνα με αυτό, πολλά πεδία της αγοράς από μια εκθετική καμπύλη αναζήτησης. Στην αρχή της, βρίσκονται οι υψήλες πωλήσεις που ανήκουν στα δημοφιλή προϊόντα τα οποία είναι τα κυρίαρχα στην κατηγορίας τους. Το κοινό των συγκεκριμένων πωλήσεων, είναι το λεγόμενο ευρύ κοινό που προσεγγίζεται με τους παραδοσιακούς τρόπους διαφήμισης. Όσο μετατοπιζόμαστε δεξιά της καμπύλης, οι πωλήσεις μειώνονται και εκεί ανήκουν τα λιγότερα δημοφιλή προϊόντα.

Αυτά είναι διαφοροποιημένα σε σχέση με τα κυρίαρχα, συνήθως πιο εξειδικευμένα και δεν έχουν τη δυνατότητα μαζικής διαφήμισης. Όμως, το πληθός αυτών των προϊόντων είναι μεγάλο και προσθέτοντας τις χαμήλες πωλήσεις τους, διαπιστώνεται ότι αποτελούν ένα σημαντικό μερίδιο αγοράς. Το διαδίκτυο μέσα από τις εφαρμογές Web 2.0 είναι ένας πολύ αποτελεσματικός τρόπος για να αναδειχθεί αυτό το πεδίο της αγοράς, το κοινό της οποίας αποτελείται σε μεγάλο ποσοστό απο διαδικτυακούς χρήστες που χρησιμοποιούν το μέσο για να κάνουν τις αγορές τους.

Το ακόλουθο παράδειγμα είναι χαρακτηριστικό, στο χώρο της διαδικτυακής ενοικίασης βίντεο :

Στην ιστοσελίδα blockbuster.com που απευθύνεται στο ευρύ κοινό, το 70% των ενοικιάσεων ανήκει στις νέες ταινίες που είναι ήδη πολύ γνωστές και οι παλιοί τίτλοι δεν παρουσιάζουν ιδιαίτερη κίνηση. Σε αντίθεση, στο netflix.com το 70% των ενοικιάσεων ανήκει σε ταινίες που δεν είναι καινούργιες. Έτσι, εκμεταλλεύεται καλύτερα το σύνολο του καταλόγου του γεμίζοντας και νέες ανάγκες την αγορά, τη στιγμή που το blockbuster.com ικανοποιεί μόνο την δεδομένη ζήτηση. Αυτό το πετυχαίνει με τη διατήρηση βάσης ενός δισεκατομμυρίου βαθμολογιών και απόψεων χρηστών και μιας προσεχτικά σχεδιασμένης μηχανής που προσωποποιεί τις προτιμήσεις και προτείνει εναλλακτικές επιλογές.Είναι δε τόσο αποτελεσματική, όπως φαίνεται παρακάτω : Ταινία ενός συγκεκριμένου σκηνοθέτη είχε αγνοηθεί όταν κυκλοφόρησε για πρώτη φορά. Όταν επόμενη ταινία του ίδιου του σκηνοθέτη σημείωσε επιτυχία, το site πρότεινε σε αυτούς που την είδαν και την παλαιά. Το αποτέλεσμα ήταν ότι η παλαιότερη ταινία σημείωσε τελικά διπλάσιες ενοικιάσεις σε σχέση με την καινούργια.

2.6 Ενδοεταιρική οργάνωση και τη λειτουργία

Για την ενδοεταιρική οργάνωση και τη λειτουργία της σύγχρονης επιχείρησης χρησιμοποιούνται διάφορες τεχνολογίες και εφαρμογές του Web 2.0, με πολλά οφέλη. Η διάδοση αυτή έχει δημιουργήσει τον όρο Enterprise 2.0. Οι σημαντικότερες εξ 'αυτών είναι :

2.6.1 Χρήση των ιστολογιών blogs

Τα ιστολόγια προσφέρουν στα στελέχη των επιχειρήσεων νέες δυνατότητες επικοινωνίας, ανταλλαγής απόψεων και τεχνογνωσίας ακόμη και αν δεν βρίσκονται στον ίδιο φυσικό χώρο. Η ανάπτυξη της συνεργασίας μέσα από αυτά, εκτός από τη σύσφιγξη σχέσεων, έχει και πρακτικά αποτελέσματα στην επίλυση διαφόρων θεμάτων. Στα blogs μπορούν να συνεισφέρουν και άτομα εκτός της επιχείρησης που βρίσκουν ενδιαφέροντα τα θέματά τους. Η συμβολή τους, εκτός από την παροχή ιδεών και λύσεων, είναι σημαντική όταν η εταιρία απευθύνεται σε καταναλωτές. Με το να καταθέτουν την άποψή τους, δίνουν στην εταιρεία ένα

σαφές δείγμα για την τάση της αγοράς και έτσι συμβάλλουν στη διαμόρφωση ενός προϊόντος για παράδειγμα. Τέλος, υπάρχουν και ιστοσελίδες που χρησιμοποιούν blogs ή forum για να δώσουν λύση σε θέματα ή προβλήματα που απασχολούν εταιρίες. Οι επισκέπτες τους αμοίβονται με ποσά που μπορεί να φτάσουν τα χιλιάδες ευρώ.

2.6.2 Χρήση των wikis

Η ιστοσελίδα του wiki μιας εταιρίας γίνεται το σημείο αναφοράς για πληροφορίες σχετικά με την εξέλιξη στην κατασκευή ενός προϊόντος, την πορεία ενός έργου, την πρόοδο μιας μελέτης, την ανάπτυξη κώδικα λογισμικού, κ.τ.λ. Υπάρχει μια λεπτομερής καταγραφή του κάθε σταδίου, που μπορεί κάποιος υπάλληλος να αναζητήσει πολύ εύκολα. Μπορεί να χρησιμοποιηθεί και σαν μέσο επικοινωνίας μεταξύ τους. Ο τρόπος καταγραφής των δεδομένων μέσα στην σελίδα είναι πολύ απλός ακόμη και σε εργαζομένους που δεν εξοικειωμένοι με τέτοιου είδους τεχνολογία. Η χρήση τους συμβάλλει στην καλύτερη οργάνωση, στην άμεση και λεπτομερή ενημέρωση των υπαλλήλων, στην εξοικονόμηση χρόνου και ενέργειας και στη μείωση κίνησης στο εταιρικό δίκτυο από σχετικά e-mails.

2.6.3 Tagging

Η αναζήτηση συγκεκριμένων δεδομένων από το μεγάλο πλήθος που έχουν οι επιχειρήσεις, γίνεται πολύ πιο εύκολα και γρήγορα με τη προσθήκη περιγραφικών λέξεων για αυτά. Πολύ χρήσιμη στα e-mails.

Χρήση των social networking ιστοσελίδων με ειδίκευση στα στελέχη επιχειρήσεων όπως το linkedin .

Σε αυτή την ιστοσελίδα είναι εγγεγραμμένα πάνω από 14.000.000 στελέχη, από 150 περίπου κλάδους. Οι χρήστες δημιουργούν προφίλ και προσθέτουν τις επαφές τους με γνωστά τους άτομα ή με άλλα στελέχη που έχουν κοινά ενδιαφέροντα και ασχολίες. Τα μέλη-χρήστες μπορούν να δώσουν εμπιστευτικές συστάσεις για άτομα των επαφών τους σε άλλα μέλη και έτσι να φέρουν σε επαφή άτομα που αλλιώς θα ήταν δύσκολο να βρεθούν. Η ιστοσελίδα χρησιμοποιείται ακόμη και σαν μέσο εύρεσης εργασίας, επαγγελματικών συνεργασιών και αναζήτησης κατάλληλων στελεχών από επιχειρηματίες.

2.6.4 Υιοθέτηση εφαρμογών αρχιτεκτονικής Web 2.0

Οι εφαρμογές και τα προγράμματα του Web 2.0 είναι ελαφριά, εύκολα στην εγκατάσταση, χρήση, συντήρηση και ανανέωση. Πολύ σημαντική και η τάση που υπάρχει για την αρχιτεκτονική Saas (Software as a Service) σύμφωνα με την οποία οι εφαρμογές λειτουργούν μέσω διαδικτύου. Η αποθήκευση και επεξεργασία των δεδομένων γίνεται σε κεντρικό διαδικτυακό server.Ο χρήστης εγκαθιστά ελάχιστα πράγματα και έχει πρόσβαση σε προγράμματα και αρχεία της εργασίας του από παντού και πάντα, αρκεί ο Η/Υ να είναι συνδεδεμένος. Έτσι ο τρόπος εργασίας είναι πιο εύκολος πιο παραγωγικός και υπάρχει και οικονομία εξοπλισμού (hardware, software).

2.6.5 Podcasts

Είναι αρχεία audio ή και video που μπορεί να περιέχουν το περιεχόμενο συναντήσεων, συνεδρίων, σεμιναρίων, εκδηλώσεων ή και διάφορα ηχητικά μηνύματα και οδηγίες που απευθύνονται σε υπαλλήλους μιας επιχείρησης. Οι χρήστες μιας τέτοιας τεχνολογίας μπορούν να ενημερώνονται άμεσα στον browser τους ή ακόμα και σε κινητές συσκευές για το πότε ένα Podcast είναι διαθέσιμο και μετά να το ακούνε.

2.6.6 RSS

Η ενσωμάτωση των RSS feeds που προέρχονται από συγκεκριμένες ιστοσελίδες στον browser του Η/Υ ή του κινητού τηλεφώνου, εξασφαλίζει τακτική επικοινωνία με τις σελίδες αυτές και προβολή των αλλαγών στο περιεχόμενό τους π.χ. νέα, ανακοινώσεις, κλπ. Έτσι, ακόμη και οι εργαζόμενοι που δεν συμμετέχουν σε κάποιο wiki, ή βρίσκονται εκτός γραφείου, μπορούν να ενημερώνονται για ό,τι σημαντικό συμβαίνει στην εταιρία και τους αφορά.

2.6.7 Πιθανά προβλήματα και δυσκολίες

Παρακάτω παρουσιάζονται διάφοροι τομείς στους οποίους θα πρέπει να δοθεί προσοχή από τις εταιρίες, προκειμένου να αποφύγουν πιθανά προβλήματα και δυσκολίες κατά τη υιοθέτηση και χρησιμοποίηση των WEB 2.0 τεχνολογιών.

Αν και η ανάπτυξη του διαδικτύου είναι μεγάλη, ένα σημαντικό κομμάτι του πληθυσμού δεν έχει ακόμα πρόσβαση στις ευρυζωνικές υπηρεσίες. Ιδιαίτερα σε χώρες όχι τόσο τεχνολογικά ανεπτυγμένες το ποσοστό αυτό είναι μεγάλο. Άρα, προκειμένου να καλυφτεί μεγαλύτερο πληθυσμιακό κομμάτι θα πρέπει οι εταιρίες να εξακολουθούν να χρησιμοποιούν και τους παραδοσιακούς τρόπους διαφήμισης, όπου είναι απαραίτητο.

Επίσης, εφόσον χρησιμοποιούν τις απόψεις και προτιμήσεις των χρηστών για τη διαμόρφωση της στρατηγικής και των προϊόντων τους, πρέπει να έχουν σαφή άποψη για το προφίλ τους γιατί αυτό σε αρκετές περιπτώσεις έχει διαφορές σε σχέση με το υπόλοιπο καταναλωτικό κοινό. Οφείλουν να είναι συνεχώς ενήμερες για τις απόψεις των χρηστών, ώστε να απαντούν σε αρνητικές κριτικές ή και να εντοπίζουν εσκεμμένα κακόβουλα σχόλια που πιθανόν να υπάρξουν. Έτσι, ενισχύουν και την αξιοπιστία τους απέναντι στους χρήστες

Σε ό,τι αφορά την ενδοεταιρική οργάνωση των επιχειρήσεων, πολύ σημαντικό είναι το θέμα της ασφάλειας των δεδομένων, κυρίως των WEB 2.0 εφαρμογών που χρησιμοποιούν το διαδίκτυο σαν πλατφόρμα. Εάν δεν υπάρχει σιγουριά για την ασφάλεια, τότε οι επιχειρήσεις είναι απίθανο να επενδύσουν σε αυτές και να τις χρησιμοποιήσουν.

Πολλές από τις WEB 2.0 εφαρμογές είναι ανεπτυγμένες με λογισμικού ανοικτού κώδικα, βρίσκονται σε συνεχή ανάπτυξη και είναι καινούργιες χωρίς μεγάλη περίοδο εμπειρίας λειτουργίας. Όμως, θα πρέπει αυτές που διανέμονται για εταιρική χρήση να είναι απόλυτα αξιόπιστες χωρίς προβλήματα λειτουργίας (bugs), να παρέχεται τεχνική υποστήριξη, κλπ.

Πολύ πιθανό είναι να χρειαστεί και η πρόσληψη ή χρήση μέσα από την εταιρία εξειδικευμένου προσωπικού που θα εποπτεύει την λειτουργία αυτών των εφαρμογών. Πέρα από την συνεχή ενημέρωση για αναβαθμίσεις και νέες εφαρμογές που προκύπτουν, θα πρέπει να επιβλέπει τη ορθή χρήση τους. Όποτε χρειάζεται, να εξηγεί στους υπόλοιπους εργαζόμενους την λειτουργία τους και να ετοιμάζει το υλικό για διάφορα rss feeds, podcasts, vidcasts, κλπ.

Ακόμη, οι απόψεις που δημοσιοποιούνται σε blogs που είναι ανοιχτά σε εξωτερικούς χρήστες θα πρέπει να είναι υπεύθυνες και ελεγχόμενες από την εταιρία για αποφυγή παρερμηνειών και παρεξηγήσεων. Στα ενδοεταιρικά blogs και wikis υπάρχει πάντα ο κίνδυνος λόγω φόρτου εργασίας να μην ενημερώνονται σωστά ή να απαιτείται σημαντικό μέρος από τον πολύτιμο χρόνο των στελεχών που είναι σε θέση να δώσουν απαντήσεις σε διάφορα ζητήματα που προκύπτουν.

Τέλος, θα πρέπει να πεισθούν οι εργαζόμενοι ότι πρόκειται για εφαρμογές που θα διευκολύνουν το έργο τους και όχι ένα μέσο συνεχούς ελέγχου της προόδου τους, όπως θα μπορούσε κάποιος να εκλάβει τη συνεχή καταγραφή των δραστηριοτήτων σε wikis, και να καλλιεργηθεί σε αυτούς η φιλοσοφία της συνεργασίας.

2.7 Χρήστες

Το παραδοσιακό WEB αποτέλεσε το μέσο χάρη στο οποίο οι χρήστες απέκτησαν πρόσβαση σε πληθώρα δεδομένων και εκτεταμένο περιεχόμενο στον παγκόσμιο ιστό και να έχουν μια πρώτη μορφή επικοινωνίας μεταξύ τους. Σε αυτή τη βάση, θα μπορούσε κανείς να ισχυριστεί ότι το WEB 2.0 αποτελεί την εξέλιξη που έχει σαν κινητήριο δύναμη τους ίδιους τους χρήστες και διαμορφώνεται από τις ανάγκες τους. Ανάγκες για αρτιότερη, ευκολότερη και πιο αποτελεσματική επικοινωνία, τροποποίηση των υπηρεσιών και λειτουργιών με βάση το πώς οι ίδιοι επιθυμούν να χρησιμοποιούν το διαδίκτυο και να επηρεάζουν τις διάφορες ανθρώπινες δραστηριότητες, κλπ. Ίσως είναι και η πρώτη φορά που οι χρήστες καθορίζουν τις εξελίξεις σε τόσο μεγάλο βαθμό. Σε αυτό το πλαίσιο, αξίζει να αναφερθεί ότι ακόμη και το περιοδικό Time ανακήρυξε σαν σημαντικότερο πρόσωπο της χρονιάς 2006 τους χρήστες, σαν αναγνώριση της καταλυτικής τους επίδρασης στις εξελίξεις:

Μερικές από τις σημαντικότερες συνέπειες για τους χρήστες, αναφέρονται παρακάτω:

- **Χρήση εφαρμογών που εξυπηρετούν τα συμφέροντα-ανάγκες των χρηστών:** «Ελαφριά» τεχνολογία σε πρωτόκολλα (REST), open source εφαρμογές πολλές φορές σχεδιασμένες με τη συμβολή και άποψη των

ίδιων των χρηστών, απλότητα στο προγραμματιστικό και λειτουργικό σχεδιασμό (π.χ γλώσσα PHP αντί για C# ή Java, εφαρμογές βασισμένες στην τεχνολογία Ajax), δυνατότητα παραμετροποίησης ιστοσελίδων σύμφωνα με τις προτιμήσεις τους (Netvibes.com) και ευκολότερη αναζήτηση πληροφορίας μέσω tagging. Όλα αυτά προσφέρουν στους χρήστες πολύ καλύτερη, άμεση και πιο ουσιαστική εμπειρία χρήσης του διαδικτύου. Ακόμη, η αντικατάσταση πολλών παραδοσιακών εφαρμογών που μέχρι πριν λίγο καιρό οι χρήστες υποχρεούνταν να αγοράζουν (λειτουργικά συστήματα, προγράμματα e-mail, Office, κλπ) με αντίστοιχες ανοιχτού κώδικα και διαδικτυακές εφαρμογές (Linux, g-mail, google docs) έχει σημαντικά οικονομικά οφέλη γι' αυτούς.

- **Νέες διαστάσεις στην επικοινωνία μεταξύ των χρηστών:** Εφαρμογές όπως τα Skype, MSN Messenger, κλπ, προσφέρουν στους χρήστες, ανεξάρτητα με το που βρίσκονται, άμεση επικοινωνία με κείμενο, φωνή και εικόνα με σχεδόν μηδενικό κόστος. Επιπλέον, η πρωτοφανής διάδοση των blogs, καθιστά την επικοινωνία ευκολότερη, μαζικότερη, πιο πλούσια και ουσιαστική. Ο καθένας μπορεί να εκφράσει τις σκέψεις και τις απόψεις του, με αποδέκτες όλους τους χρήστες του διαδικτύου και όσοι ενδιαφέρονται για αυτές, επικοινωνούν μαζί του μέσω σχόλιων στο blog. Ακόμη, με το social bookmarking (π.χ http://del.icio.us/) μπορεί κάποιος εύκολα να βρει και να επικοινωνήσει με άτομα που έχουν τα ίδια ενδιαφέροντα. Τέλος, το φαινόμενο του social networking που αναφέρθηκε και στο 1.2 Εφαρμογές του WEB 2.0, τείνει να αποτελέσει μία από τις κυρίαρχες μορφές κοινωνικοποίησης ιδιαίτερα μεταξύ εφήβων και νεαρών χρηστών.
- **Ελεύθερη δημοσιοποίηση δεξιοτήτων, έκφρασης δημιουργικότητας, ευκαιρίες ανάδειξης:** Δύο από τις ιστοσελίδες που βρίσκονται σταθερά ανάμεσα στις 10 πρώτες παγκοσμίως σε κίνηση31 είναι οι YouTube.com, και Myspace.com. Σε αυτές, πέρα από τα video γενικού περιεχομένου, πολλοί χρήστες δημοσιοποιούν τις δεξιότητές τους (π.χ. καλλιτεχνικές, video, μουσική, φωτογραφία) και έτσι έχουν πρόσβαση σε ένα ευρύ κοινό που δεν θα είχαν διαφορετικά. Χαρακτηριστικό παράδειγμα της «γενιάς καλλιτεχνών του Myspace» όπως ονομάστηκε, είναι το βρετανικό συγκρότημα Arctic Monkeys (http://www.myspace.com/arcticmonkeys) που έγινε γνωστό αρχικά μέσω του διαδικτύου και μετά υπέγραψε συμβόλαιο με

δισκογραφική εταιρία και κυκλοφόρησε τη δουλειά του σε άλμπουμ. Κατάφερε να γίνει το γκρουπ με τις περισσότερες πωλήσεις (363.735 αντίτυπα) σε μια εβδομάδα στην ιστορία της Βρετανικής μουσικής32.

- **Αντικειμενικότερη ενημέρωση:** Σε πολλές ειδησεογραφικές ιστοσελίδες, η δυνατότητα σχολιασμού των ειδήσεων από τους χρήστες, η συζήτησή τους σε blogs και η δημοσιοποίηση video ή φωτογραφιών που οι ίδιο τράβηξαν, προσφέρει ακόμη μία προοπτική αντικειμενικού ελέγχου της πληροφορίας.
- **Εκμετάλλευση της γνώσης των χρηστών:** Οι χρήστες μπορούν να γίνουν “σοφότεροι” εκμεταλλευόμενοι τις πληροφορίες και γνώσεις που καταθέτουν οι υπόλοιποι μέσω blogs, wikis, και forums για κάποιο θέμα. Από γνώμες για καταναλωτικά προϊόντα μέχρι συμβουλές για ιατρικά θέματα, η διαδικτυακή κοινότητα προσφέρει γνώση και εμπειρία που οι χρήστες δείχνουν να εμπιστεύονται ολοένα και περισσότερο. Ακόμη, γνώσεις σε επιστημονικά, πρακτικά και κοινωνικά θέματα, προσφέρονται ελεύθερα. Ιστοσελίδες όπως το netmums.com αποδεικνύεται ότι επιτελούν σημαντικό κοινωνικό έργο και αντικαθιστούν παραδοσιακές μορφές κοινωνικής μέριμνας.
- **Δημοκρατικότητα, αίσθηση ένταξης σε κοινότητα, συνεργασιμότητα και συνεισφορά:** Με τα wikis, τα blogs και τα forums, οι χρήστες ανεξαρτήτως γεωγραφικής θέσης και κοινωνικών, φυλετικών χαρακτηριστικών, μπορούν να ενταχθούν σε μια κοινότητα που ασχολείται με ένα θέμα που τους ενδιαφέρει, να ανταλλάξουν απόψεις, να συνεργαστούν και να συνεισφέρουν στην επίτευξη ενός κοινού σκοπού. Ιδιαίτερα η εθελοντική συνεισφορά είναι τόσο διαδεδομένη ανάμεσα στους χρήστες του διαδικτύου όσο ίσως σε κανένα άλλο τομέα της κοινωνικής ζωής. Το μέγεθος αυτού του φαινόμενου και η κατάργηση των παραδοσιακών φραγμών, κάνει πολλούς να υποστηρίζουν ότι μια νέα κοινωνική επανάσταση συντελείται.
- **Αμφίδρομη επικοινωνία του χρήστη με επιχειρήσεις ή οργανισμούς και επίδρασή του στη υιοθέτηση κατευθύνσεων:** Μέσα από τις εφαρμογές WEB 2.0, οι απόψεις των χρηστών δημοσιοποιούνται και ανάλογα με το πόσο συγκλίνουν σε μια θέση και το πλήθος τους, αποκτούν τέτοια σημασία που επιχειρήσεις, οργανισμοί αλλά και πολιτικοί φορείς αναγκάζονται να τις λάβουν υπόψιν τους.

- **Ενίσχυση της διαπραγματευτικής δύναμης των χρηστών στις εμπορικές συναλλαγές:** Η επιλογή ενός προϊόντος από μια παγκόσμια αγορά μέσω του διαδικτύου, η δυνατότητα ανάγνωσης της άποψης-εκτίμησης για ένα προϊόν από άτομα που το έχουν ήδη αγοράσει και η εύκολη σύγκριση τιμών, ενισχύει την θέση των χρηστών-καταναλωτών απέναντι στις εταιρίες. Η παραδοσιακή διαφήμιση δεν είναι το ίδιο αποτελεσματική όσο παλιότερα, και οι εταιρίες αναγκάζονται να αντιμετωπίσουν τους καταναλωτές πιο υπεύθυνα.
- **Καλύτερη εξυπηρέτηση των πολιτών από υπηρεσίες, οργανισμούς**: Η διεκπεραίωση υποθέσεων μέσω του διαδικτύου και η δυνατότητα καταχώρησης των στοιχείων των πολιτών σε κοινή φόρμα δεδομένων, εξυπηρετεί τους πολίτες και διευκολύνει τις καθημερινές τους συναλλαγές με τις δημόσιες υπηρεσίες

2.7.1 Πιθανά προβλήματα και δυσκολίες

Ένα από τα θέματα που προκαλεί προβληματισμό, όσον αφορά τους **χρήστες, είναι η χρήση των προσωπικών τους δεδομένων**, όπως αυτά προκύπτουν μέσα από τις δημοσιοποιημένες προτιμήσεις τους στο διαδίκτυο. Για παράδειγμα, διαφημιστικές και όχι μόνο, εταιρίες μπορούν να ερευνήσουν τις προτιμήσεις, αγορές και δημοσιοποιημένες σε forums και blogs απόψεις των χρηστών και να τους στέλνουν συνεχώς μηνύματα για διάφορα σχετικά προϊόντα. Μέσα από το social bookmarking και το tagging, έχουν τη δυνατότητα να διαμορφώσουν προφίλ για τον κάθε χρήστη και να το χρησιμοποιήσουν για εμπορικούς σκοπούς χωρίς την συγκατάθεσή του.

Τα στοιχεία που εισάγει (λέξεις κλειδιά) ο χρήστης στις μηχανές αναζήτησης, αποτελούν προσωπικά δεδομένα. Μερικές από αυτές (π.χ.Google) από τη στιγμή που τα αποθηκεύουν, μπορούν θεωρητικά να τα διαθέσουν σε τρίτους. Αυτό αποτέλεσε και θέμα διαμάχης πρόσφατα, με τη Google να δεσμεύεται ότι αυτά τα δεδομένα μετά από 18 μήνες θα διαγράφονται.

Η αγορά μερικών από των δημοφιλέστερων WEB 2.0 ιστοσελίδων από τις μεγάλες επιχειρήσεις του χώρου (Google, Yahoo!, Microsoft, News Corporation),

δημιουργεί ανησυχίες, κυρίως ως προς το ποιος τελικά κατέχει και πως εκμεταλλεύεται τα δεδομένα των χρηστών που δημιουργούνται.

Επιπλέον, η περίπτωση **απώλειας, λόγω παραβίασης ασφάλειας, ευαίσθητων προσωπικών δεδομένων των χρηστών** (τραπεζικοί κωδικοί, ιατρικό ιστορικό, κλπ.) μπορεί να έχει πολύ δυσάρεστες συνέπειες. Αν προσθέσουμε σε όλα αυτά και την κακόβουλη συλλογή στοιχείων (π.χ. πολιτικών, κοινωνικών απόψεων σε blogs, forums), τότε συμπεραίνουμε πως είναι απαραίτητη η ύπαρξη ενός νομικού πλαισίου που θα προστατεύει τους χρήστες και η υιοθέτηση εφαρμογών που θα εγγυώνται την υψηλή ασφάλεια των προσωπικών τους δεδομένων. Παράλληλα θα πρέπει να γίνονται έλεγχοι από ειδικευμένες αρχές και να παρέχεται ενημέρωση για τις περιπτώσεις πιθανού κινδύνου.

Επίσης, μερικές φορές η **ενημέρωση που παρέχεται μέσα από forums και blogs μπορεί να μην είναι ακριβής** γι' αυτό και οι χρήστες θα πρέπει πάντα να διασταυρώνουν τις πληροφορίες που παίρνουν. Σε θέματα βαρύνουσας σημασίας (νομικά, υγείας, κλπ) να ζητούν τη γνώμη των αρμοδίων.

Οφείλουν να είναι ιδιαίτερα προσεκτικοί στη χρήση προγραμμάτων διαμοίρασης αρχείων (π.χ. Torrents), γιατί πιθανή **παραβίαση δικαιωμάτων πνευματικής ιδιοκτησίας** μπορεί να έχει δυσάρεστες νομικές συνέπειες γι' αυτούς.

Στο χώρο της εργασίας, λόγω του ότι μέσω των εφαρμογών WEB 2.0 μπορεί κάποιος να εργάζεται ακόμη και εκτός εταιρίας (π.χ. από το σπίτι του) πρέπει να υπάρξει μέριμνα ώστε να μη χρησιμοποιηθεί αυτό ως τρόπος υπερωριακής απασχόλησης, που αντιτίθεται στα δικαιώματα του εργαζομένου.

2.8 Ανάπτυξη Web 2.0 ιστοσελίδων

Το Web 2.0 έφερε επίσης μεγάλες αλλαγές στον τρόπο με τον οποίο σχεδιάζονται και αναπτύσσονται οι ιστοσελίδες. Ο βασικός κανόνας που ακολουθείτε πλέον είναι η λειτουργικότητα και η επάρκεια του περιεχομένου. .

Τα κύρια χαρακτηριστικά των Web 2.0 ιστοσελίδων είναι:

- Μεγαλύτερες γραμματοσειρές. Η ανάλυση της οθόνης γίνεται όλο και μεγαλύτερη, οπότε θα πρέπει οι τίτλοι και τα κείμενα να έχουν μεγαλύτερο μέγεθος έτσι ώστε να διαβάζονται εύκολα.
- Απλότητα. Ένα Web 2.0 layout θα πρέπει να είναι απλό, χωρίς περιττά στοιχεία.
- Εύκολη πλοήγηση. Η μέθοδος πλοήγησης που χρησιμοποιείτε θα πρέπει να είναι απλή, να τραβάει την προσοχή να είναι εύκολη επιλέξιμη και σταθερή σε κάθε σελίδα.
- Έντονο λογότυπο. Θα πρέπει να χρησιμοποιούνται μεγάλα και έντονα λογότυπα. Αν ένα site δεν είναι αξιοσημείωτο, ο επισκέπτης δεν θα επιστρέψει. Ένα μοναδικό και «δυνατό» λογότυπο και όνομα ή slogan είναι πολύ σημαντικό για να κάνει τον επισκέπτη να μην το ξεχάσει.
- Gradients, Gloss, Flat Colors. Some Web 2.0 sites make use of strong, flat, colors. The plastic and gradient effects are often overused, but can look good if used sparingly
- Χρήση εικονιδίων. Η χρήση εικονιδίων Drive by icons. Elements are more noticeable if they have relevant icons near them
- Χρήση δύο ή τρεις στηλών. Πλέον οι ιστοσελίδες θα πρέπει να σχεδιάζονται με δύο στήλες

2.9 Επίλογος

Παρά τις αλλαγές που έχει επιφέρει το Web 2.0 στη χρήση του παγκόσμιου ιστού, οι τεχνολογίες που χρησιμοποιούνται είναι οι ίδιες. Οι τεχνολογίες αυτές παρουσιάζονται στα παρακάτω κεφάλαια.

ΚΕΦΑΛΑΙΟ 3

HTML

3.1 Εισαγωγή

Η HTML (ακρωνύμιο του αγγλικού HyperText Markup Language, ελλ. Γλώσσα Σήμανσης Υπερκειμένου) είναι η κύρια γλώσσα σήμανσης για τις ιστοσελίδες, και τα στοιχεία της είναι τα βασικά δομικά στοιχεία των ιστοσελίδων.

Η HTML γράφεται υπό μορφή στοιχείων HTML τα οποία αποτελούνται από ετικέτες, οι οποίες περικλείονται μέσα σε σύμβολα «μεγαλύτερο από» και «μικρότερο από» (για παράδειγμα <html>), μέσα στο περιεχόμενο της ιστοσελίδας. Οι ετικέτες HTML συνήθως λειτουργούν ανά ζεύγη (για παράδειγμα <h1> και </h1>), με την πρώτη να ονομάζεται ετικέτα έναρξης και τη δεύτερη ετικέτα λήξης (ή σε άλλες περιπτώσεις ετικέτα ανοίγματος και ετικέτα κλεισίματος αντίστοιχα). Ανάμεσα στις ετικέτες, οι σχεδιαστές ιστοσελίδων μπορούν να τοποθετήσουν κείμενο, πίνακες, εικόνες κλπ.

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 ερμηνεύσει το περιεχόμενο της σελίδας.

Τα στοιχεία της HTML χρησιμοποιούνται για να κτίσουν όλους του ιστότοπους. Η HTML επιτρέπει την ενσωμάτωση εικόνων και άλλων αντικειμένων μέσα στη σελίδα, και μπορεί να χρησιμοποιηθεί για να εμφανίσει διαδραστικές φόρμες. Παρέχει τις μεθόδους δημιουργίας δομημένων εγγράφων (δηλαδή εγγράφων που αποτελούνται από το περιεχόμενο που μεταφέρουν και από τον κώδικα μορφοποίησης του περιεχομένου) καθορίζοντας δομικά σημαντικά στοιχεία για το κείμενο, όπως κεφαλίδες, παραγράφους, λίστες, συνδέσμους, παραθέσεις και άλλα. Μπορούν επίσης να ενσωματώνονται σενάρια εντολών σε γλώσσες όπως η JavaScript, τα οποία επηρεάζουν τη συμπεριφορά των ιστοσελίδων HTML.

Οι Web browsers μπορούν επίσης να αναφέρονται σε στυλ μορφοποίησης CSS για να ορίζουν την εμφάνιση και τη διάταξη του κειμένου και του υπόλοιπου υλικού. Ο οργανισμός W3C, ο οποίος δημιουργεί και συντηρεί τα πρότυπα για την HTML και τα CSS, ενθαρρύνει τη χρήση των CSS αντί διαφόρων στοιχείων της HTML για σκοπούς παρουσίασης του περιεχομένου.

3.2 Εκδόσεις HTML

Η HTML είναι μια αναπτυσσόμενη γλώσσα, και σε κάθε νέα έκδοση αποδίδετε και ένας αριθμός. Η πρώτη έκδοση που ορίστηκε ήταν η HTML 2.0. Αυτή η έκδοση είχε τα περισσότερα από στα στοιχεία που ήδη γνωρίζουμε, αλλά έλειπαν μερικές από τις επεκτάσεις της Microsoft και της Netscape, ενώ δεν υποστήριζε πίνακες και ιδιότητες στοίχισης.

Η HTML 3, ήταν μια φιλόδοξη προσπάθεια από τη μεριά του Dave Raggett να αναβαθμίσει τα χαρακτηριστικά και τα βοηθήματα της HTML. Ωστόσο η έκδοσης 3 δεν ολοκληρώθηκε ή υλοποιήθηκε ποτέ, παρόλο που πολλά από τα χαρακτηριστικά της ολοκληρώθηκαν στην επόμενη επίσημη έκδοση της HTML γνωστή ως HTML 3.2.

Η HTML 3.2 ήταν η επόμενη επίσημη έκδοση της HTML, η οποία υποστήριζε πίνακες, εικόνες, επικεφαλίδες, ιδιότητες στοίχισης και κάποιες άλλες σχολαστικές λεπτομέρειες. Η HTML 3.2 είναι τρέχουσα «παγκόσμια» διάλεκτος, την οποία κατανοούν πλήρως όλοι οι περιηγητές διαδικτύου. Ωστόσο και από αυτή την έκδοση έλοιπαν επεκτάσεις για τον Internet Explorer και τον Netscape όπως είναι για παράδειγμα τα Frames, Embed και τα Applets. Η υποστήριξη για αυτές τις επεκτάσεις ήρθαν αργότερα με την HTML 4.0.

Η HTML 4.01 είναι το επόμενο επίσημο πρότυπο. Υποστηρίζει τις περισσότερες απαραίτητες επεκτάσεις, και επιπλέον υποστηρίζει και παραπάνω χαρακτηριστικά (έγγραφα σε πολλές γλώσσες, CSS, επιπλέον πίνακες, φόρμες και συμβατότητα με τη JavaScript).

Το web άλλαξε δραματικά από το 1999 όταν και έγινε διαθέσιμη στο κοινό η προηγούμενη αναθεώρηση του προτύπου της HTML, η HTML 4.01. Από τότε δημιουργήθηκαν νέες συνήθειες και τάσεις των χρηστών του web, όπως η ραγδαία αύξηση του αριθμού των οπτικοακουστικών μέσων (βίντεο και audio) που χρησιμοποιούνται στις ιστοσελίδες, η δημοσίευση κειμένων από εξωτερικά blogs και φόρουμ (content sharing), αλλά και η ανάγκη να γράφουν περισσότερες

πληροφορίες για το περιεχόμενο του site τους ώστε να αυξάνουν τις πιθανότητες εμφάνισης του στις μηχανές αναζήτησης. Όλες αυτές οι νέες τάσεις που δημιούργησε, κυρίως η όλο και αυξανόμενη ταχύτητα πρόσβασης στο Internet, δημιούργησε επίσης την ανάγκη ενός ευκολότερου τρόπου εισαγωγής τεχνολογιών στις ιστοσελίδες, χωρίς να χρειάζεται η ανάπτυξη μεγάλων κομματιών κώδικα JavaScript για την υλοποίηση τους. Οι νέες τάσεις χρήσης του web και η ανάγκες που οι ίδιες επέβαλαν, έγιναν αιτία να αναπτυχθεί ένα νέο πρότυπο της HTML, αυτό της HTML5.

3.2.1 HTML 5

Η HTML5 είναι το νέο standard πρότυπο για την HTML, την XHTML και την HTML DOM. Η ανάπτυξη της HTML5 έγινε με την συνεργασία της World Wide Web Consortium (W3C) και της Web Hypertext Application Technology Working Group (WHATWG). Η WHATWG εργαζόταν επάνω στις web φόρμες και τις web εφαρμογές, ενώ η W3C, η οποία δημιούργησε και διαχειρίζεται τα πρότυπα της HTML και της XHTML, ασχολήθηκε με την ανάπτυξη του νέου προτύπου XHTML 2.0. Το 2006 αποφάσισαν να συνεργαστούν για να δημιουργήσουν το νέο πρότυπο, την HTML5.

Η HTML5 αναπτύχθηκε με βάση τους παρακάτω κανόνες:

- νέα χαρακτηριστικά έπρεπε να προστεθούν στην δομή των HTML, CSS και JavaScript
- μείωση των περιπτώσεων που χρειάζεται η εγκατάσταση plugins στον browser για κάποιου συγκεκριμένου τύπου στοιχείων (όπως βίντεο και audio)
- καλύτερη διαχείριση σφαλμάτων
- προσθήκη περισσότερων ετικετών οι οποίες θα αντικαταστήσουν κομμάτια κώδικα JavaScript που χρησιμοποιούσαν συχνά οι web designers
- το νέο πρότυπο θα έπρεπε να είναι αυτόνομο χωρίς να χρειάζεται να καλεί κομμάτια κώδικα από άλλα πρότυπα
- τα βήματα του σχεδιασμού και της ανάπτυξης του νέου προτύπου θα έπρεπε να είναι ορατά στο κοινό.

3.2.2 Νέα χαρακτηριστικά της HTML5

Μερικά από τα νέα χαρακτηριστικά του νέου προτύπου είναι τα παρακάτω:

- δυνατότητα σχεδιασμού γραφικών με χρήση JavaScript (νέα ετικέτα canvas)
- αναπαραγωγή βίντεο και audio χωρίς να χρειάζεται η εγκατάσταση plugins (νέες ετικέτες video και audio)
- προσθήκη νέων ετικετών που κάνουν την δημιουργία και την διαχείριση των ιστοσελίδων, ακόμη πιο εύκολη (νέες ετικέτες article, footer, header κτλ.)
- νέα στοιχεία στις HTML φόρμες (calendar, date, time, search κτλ.)

3.2.3 HTML5 και browsers

Οι τελευταίες εκδόσεις των browsers Firefox, Chrome, Opera, Safari καθώς και ο Internet Explorer 9, υποστηρίζουν μόνο μερικά από τα χαρακτηριστικά της HTML5.

Στο μέλλον θα υποστηρίζουν όλοι οι browser όλα τα χαρακτηριστικά του προτύπου.

Για τον παραπάνω λόγο, για τη δημιουργία της ιστοσελίδας της εργασίας θα χρησιμοποιήσουμε HTML 4.01.

3.3 Η δομή ενός εγγράφου HTML

- Κάθε HTML έγγραφο αρχίζει και τελειώνει με τις ετικέτες <html> και </html>
- Κάθε HTML έγγραφο αποτελείτε από το τμήμα HEAD που περιβάλλεται από τις ετικέτες <head> και </head> και ένα τμήμα BODY που περιβάλλεται από τις ετικέτες <body> και </body> αντίστοιχα.
- Οι περισσότερες ετικέτες στην HTML είναι σε ζευγάρια (εκτός από μερικές που δεν έχουν ετικέτα τερματισμού), με την ετικέτα τερματισμού να έχει το σύμβολο «/» στα αριστερά από το όνομά της.
- Μερικές ετικέτες μπορούν να περιέχουν μέσα τους άλλες ετικέτες. Αυτό αποκαλείτε ενθυλάκωση. Στην περίπτωση μας η ετικέτες <title>…</title> είναι ενθυλακωμένες στις ετικέτες <head>…</head>.

- Το κείμενο μέσα στις ετικέτες <title></title> εμφανίζεται στον τίτλο του παραθύρου του Web Browser.
- Το κείμενο μέσα στις ετικέτες <body></body> εμφανίζεται μέσα στο παράθυρο του Web Browser.

3.4 Το τμήμα HEAD μιας σελίδας

Το τμήμα head περιέχει γενικές και μέτα πληροφορίες για το έγγραφο. Το head είναι πρώτο τμήμα κάθε HTML εγγράφου, το οποίο βρίσκεται ακριβώς πριν την ετικέτα <body> και ακριβώς μετά την ετικέτα <html>.

Τα περιεχόμενα του head δεν εμφανίζονται ως μέρος του συνολικού εγγράφου. Τα στοιχεία που εμφανίζονται ως μέρος του εγγράφου τοποθετούνται στο τμήμα body της σελίδας. Συνεπώς μόνο μερικά στοιχεία της HTML μπορούν να γραφτούν μέσα σε αυτό το τμήμα και είναι τα ακόλουθα:

<base>

Μια εγγραφή του αυθεντικού URL της σελίδας. Αυτό μας επιτρέπει ακόμα και αν μετακινήσουμε το έγγραφο σε ένα νέο κατάλογο να έχουμε σχετικές διαδρομές προς τις πηγές του εγγραφου.

<link>

Ορίζει τη σχέση αυτού του εγγράφου με άλλα έγγραφα. Ένα έγγραφο HTML μπορεί να περιέχει περισσότερες από μια ετικέτες link.

<meta>

Η meta είναι μια ετικέτα για μέτα πληροφορίες σχετικά με το έγγραφο, οι οποίες όμως δεν μπορούν να εκφραστούν με κάποια από τις ετικέτες, link, base ή κάποιο άλλο στοιχείο του τμήματος head. Η ετικέτα αυτή είναι πολύ βασική γιατί μπορούμε να ορίσουμε την περιγραφή της σελίδας καθώς και τις λέξεις κλειδιά. Αυτές τις ετικέτες τις διαβάζουν κατεξοχήν οι μηχανές αναζήτησης και βοηθούν την κατάταξη της ιστοσελίδας σε αυτές.

<title>

Ο τίτλος ενός εγγράφου καθορίζετε από την ετικέτα title, η οποία πρέπει να τοποθετηθεί στο τμήμα head της σελίδας. Σε κάθε έγγραφο θα πρέπει να έχετε

μόνο μια ετικέτα title, και με τον τίτλο να προσδιορίζετε σε γενικές γραμμές το περιεχόμενο της σελίδας.
Συχνά ο τίτλος χρησιμοποιείτε για να δώσει όνομα στο παράθυρο του περιηγητή διαδικτύου, να αποδώσει ένα τίτλο για τη σελίδα στο ιστορικό περιήγησης του browser και άλλα. Επίσης ο τίτλος θα πρέπει να είναι σύντομος και να μην ξεπερνά τους 64 χαρακτήρες.

<style>
Εντολές CSS για το έγγραφο. Οι εντολές CSS ενημερώνουν τον Web Browser για το πως θα πρέπει να μορφοποιηθούν τα στοιχεία της σελίδας (χρώμα γραμματοσειράς, χρώμα φόντου, κ.α.)

<script>
Κώδικας μιας γλώσσας σεναρίων όπως είναι η JavaScript.

3.5 Το τμήμα BODY μιας σελίδας

Παρακάτω παρουσιάζονται οι HTML ετικέτες που έχουν χρησιμοποιηθεί στην εφαρμογή καθώς και οι διαθέσιμες ιδιότητές τους.

3.5.1 Βασικές ετικέτες

Επικεφαλίδες
Οι επικεφαλίδες βοηθούν στον ορισμό της μορφοποίηση και της δομής ενός HTML εγγράφου. Είναι ένα πολύτιμο εργαλείο με τη βοήθεια του οποίου μπορούμε να επισημάνουμε σημαντικά σημεία και να ορίσουμε τη συνολική δομή ενός εγγράφου.
Υπάρχουν έξι επίπεδα επικεφαλίδων στην HTML που ορίζονται από τις ακόλουθες ετικέτες: <h1>, <h2>, <h3>, <h4>, <h5> και <h6> και τυπικά εμφανίζονται με μεγαλύτερο ή με πιο έντονο περιεχόμενο από το κανονικό κείμενο.

```
<h1>Επικεφαλίδα1</h1>
<h2>Επικεφαλίδα2</h2>
<h3>Επικεφαλίδα3</h3>
<h4>Επικεφαλίδα4</h4>
<h5>Επικεφαλίδα5</h5>
```

<h6>Επικεφαλίδα6</h6>

Ετικέτα <p>

Δημιουργεί νέα παράγραφο με μια κενή διαχωριστική γραμμή

**Ετικέτα
**

Η ετικέτα
 χρησιμοποιείται όταν θέλουμε να τελειώσουμε μια γραμμή κειμένου και να αρχίσουμε μια καινούργια

Ετικέτα σχολίων

<!-- This is a comment -->

Τα σχόλια χρησιμοποιούνται για να γράφουμε σημειώσεις μέσα στον πηγαίο κώδικα. Δεν εμφανίζονται στην οθόνη του browser.

3.5.2 Ιδιότητες ετικετών

Οι ιδιότητες (attributes) των ετικετών είναι τιμές που δίνουν στην ετικέτα διάφορα χαρακτηριστικά. Κάθε μια από αυτές τις τιμές επιδρά διαφορετικά στην εμφάνιση ή την λειτουργία των ετικετών. Μια ιδιότητα μπαίνει αμέσως μετά το όνομα της ετικέτας και αποτελείται από το όνομα της και μια τιμή μέσα σε διπλά εισαγωγικά.

Μια ετικέτα με ιδιότητες είναι κάπως έτσι:

<όνομα-ετικέτας ιδιότητα1="τιμη" ιδιότητα2="τιμη" ιδιότητα3="τιμη">...</όνομα-ετικέτας>

Αν και για κάθε ετικέτα υπάρχει μια συγκεκριμένη λίστα διαθέσιμων ιδιοτήτων, υπάρχουν και κάποιες ιδιότητες που μπορούν να εφαρμοστούν σχεδόν σε όλες τις ετικέτες. Αυτές είναι οι παρακάτω :

- Η ιδιότητα ***id*** παρέχει ένα αναγνωριστικό για ένα στοιχείο το οποίο είναι μοναδικό σε ολόκληρο το έγγραφο. Χρησιμοποιείται για να ταυτοποιεί το στοιχείο ώστε τα CSS να μπορούν να αλλάξουν τον τρόπο που αυτό εμφανίζεται, καθώς και τα σενάρια μπορούν να αλλάξουν, να μετακινήσουν ή να διαγράψουν τα περιεχόμενα ή την εμφάνισή του.

- Η ιδιότητα ***class*** παρέχει τη δυνατότητα ταξινόμησης παρόμοιων αντικειμένων στην ίδια κλάση. Μπορεί να χρησιμοποιηθεί για να να αποδώσει κάποια σημασία στο στοιχείο, ή για σκοπούς εμφάνισης. Για παράδειγμα, ένα έγγραφο HTML μπορεί να χρησιμοποιεί την επισήμανση class="notation" σε μερικά στοιχεία για να ξεχωρίσει από το υπόλοιπο κείμενο του εγγράφου. Κατά την εμφάνιση του εγγράφου, αυτά τα στοιχεία μπορεί -για παράδειγμα- να εμφανίζονται όλα μαζί στο τέλος της σελίδας ως υποσημειώσεις, άσχετα με την θέση που εμφανίζονται μέσα στον κώδικα. Επίσης οι ιδιότητες class χρησιμοποιούνται σημασιολογικά στα microformat. Ένα στοιχείο μπορεί να έχει πολλαπλές κλάσεις, για παράδειγμα το class="notation important" βάζει το στοιχείο τόσο στην κλάση «notation» όσο και στην «important».
- Η ιδιότητα ***style*** εφαρμόζει στυλ εμφάνισης σε συγκεκριμένα στοιχεία. Θεωρείται καλύτερη τακτική να χρησιμοποιούνται οι ιδιότητες id ή class ώστε να επιλέγεται το στοιχείο μέσα σε ένα CSS, αλλά μερικές φορές μπορεί να είναι πιο απλό να ανατεθούν style κατευθείαν στο στοιχείο.
- Η ιδιότητα ***title*** προσθέτει μια εξήγηση στο στοιχείο στο οποίο εφαρμόζεται. Στους περισσότερους browser αυτή η ιδιότητα εμφανίζεται ως αναδυόμενο παράθυρο βοήθειας.
- Η ιδιότητα ***lang*** ταυτοποιεί την φυσική γλώσσα των περιεχομένων του στοιχείου, η οποία μπορεί να είναι διαφορετική από το υπόλοιπο έγγραφο.

3.5.3 Μορφοποίηση κειμένου

Στην HTML υπάρχουν ετικέτες που ορίζουν την εμφάνιση του κειμένου. Παρακάτω παρουσιάζονται αυτές οι ετικέτες

Η ετικέτα <b>, και <big>

Και οι τρεις αυτές ετικέτες εμφανίζουν το κείμενο με έντονη μορφή. Η <b> και έχουν την ίδια ακριβώς επίδραση στο κείμενο, ενώ η <big> εμφανίζει το κείμενο με έντονη μορφή όπως οι <b> και αλλά με ελαφρώς μεγαλύτερα γράμματα.

**Η ετικέτα **

Η ετικέτα εμφανίζει το κείμενο με πλάγιους και κάπως αχνά γραμμένους χαρακτήρες. Το em είναι τα δύο πρώτα γράμματα από την λέξη emphasize.

Η ετικέτα <i>

Η ετικέτα <i> εμφανίζει το κείμενο με πλάγιους χαρακτήρες. Το i είναι το πρώτο γράμμα από την λέξη italics

3.5.4 Ετικέτα

Η ετικέτα ορίζει ένα τμήμα στο HTML έγγραφο. Ένα τμήμα ομαδοποιεί τα περιεχόμενα που είναι μέσα στις ετικέτες ... ώστε να μπορούμε να ορίζουμε ένα ενιαίο στυλ (χρώμα, γραμματοσειρά, στοίχιση) με την ιδιότητα style. Είναι η πλέον πιο πολυχρησιμοποιημένη ετικέτα γιατί βοηθάει στον ορισμό μεγάλων μπλοκ ώστε να μπορούν να μορφοποιηθεί πιο εύκολα με css ή να εμφανίσουμε ένα περιεχόμενο καλούμενο από την εκτέλεση ενός javascript.

3.5.5 Ετικέτα

Με την ετικέτα ομαδοποιούμε κείμενο για να ορίσουμε κοινό στυλ

3.5.6 Λίστες

Οι λίστες μπορούν να παρουσιάσουν αντικείμενα από πληροφορίες που είναι σε τέτοια μορφή που επιτρέπουν να διαβαστούν ευκολότερα.

Μη αριθμημένη λίστα

Η ετικέτα **<ul>** εισάγει μια μη αριθμημένη λίστα στην σελίδα μας. Το ul είναι τα δύο πρώτα γράμματα από το Unordered List. Η ετικέτα **<li>** προσθέτει γραμμές στην λίστα. Η ιδιότητα type της ετικέτας <ul> ορίζει το σύμβολο που μπαίνει μπροστά από κάθε γραμμή της λίστας. Οι τιμές που παίρνει η ιδιότητα type είναι οι εξής: *disc, circle, square*

Αριθμημένη λίστα

Η ετικέτα **<ol>** εισάγει μια αριθμημένη λίστα στην σελίδα μας, Το ol είναι τα δύο πρώτα γράμματα από το Ordered List. Η ιδιότητα type της ετικέτας <ol> ορίζει τον τύπο της ταξινόμησης. Οι τιμές που παίρνει η ιδιότητα type είναι οι εξής: A, a, I, i

Λίστα ορισμών

Η ετικέτα **<dl>** εισάγει μια λίστα ορισμών στην σελίδα μας, Το dl είναι τα δύο πρώτα γράμματα από το Definition List. Η ετικέτα **<dt>** ορίζει τον περιγραφικό τίτλο των γραμμών που ακολουθούν, οι οποίες γραμμές ορίζονται από την ετικέτα **<dd>**

3.5.7 Εικόνες

Με την HTML μπορούμε να εισάγουμε εικόνες στην σελίδα μας

Η ετικέτα <img>

Με την ετικέτα αυτή εισάγουμε μια εικόνα στην σελίδα μας. Η ετικέτα <img /> απαιτείται η χρήση της ιδιότητας src στην οποίο δηλώνουμε το URL του αρχείου. Η ετικέτα <img> δεν έχει ετικέτα τέλους, οπότε βάζουμε τον χαρακτήρα / πριν τον χαρακτήρα >
Εκτός από την ιδιότητα ***src*** η ετικέτα <img /> έχει κι άλλες ιδιότητες.

Η ιδιότητα ***alt*** είναι μία από τις επίσης σημαντικές ιδιότητες. Υπάρχουν ορισμένοι browsers που δεν υποστηρίζουν την εμφάνιση γραφικών με αποτέλεσμα να μην εμφανίζονται οι εικόνες που τοποθετούμε στις σελίδες μας. Η χρήση της ιδιότητας alt έχει σαν αποτέλεσμα σε έναν τέτοιο browser να εμφανίζετε αντί της εικόνας, το κείμενο το οποίο ορίζεται με την ιδιότητα. Συνήθως το κείμενο αυτό περιγράφει την εικόνα έτσι ώστε ο χρήστης που δεν μπορεί να την δει, να πάρει μια ιδέα για το τι απεικονίζετε σε αυτήν. Το alt είναι τα τρια πρώτα γράμματα από την λέξη alternative

Σημαντικές επίσης είναι οι ιδιότητες ***width*** και ***height*** οι οποίες ορίζουν τη διάσταση σε pixels όπου θα εμφανίζεται η εικόνα.
Το συνηθέστερο είναι να γράφουμε τις πραγματικές διαστάσεις της εικόνας. Ορισμένες φορές όμως θέλουμε να εμφανίσουμε την εικόνα με μικρότερες ή μεγαλύτερες από τις κανονικές διαστάσεις προσαρμόζοντας ανάλογα τις ιδιότητες width και height. Βέβαια αν οι διαστάσεις που ορίζουμε απέχουν πολύ από τις πραγματικές διαστάσεις της εικόνας, τότε αυτή εμφανίζεται αλλοιωμένη. Γι' αυτό είναι προτιμότερο να μικραίνουμε ή να μεγαλώνουμε την εικόνα μέσα σε κάποιο

πακέτο επεξεργασίας γραφικών γιατί εκεί χρησιμοποιούνται ειδικές συναρτήσεις που αλλάζουν το μέγεθος της εικόνας χωρίς να την αλλοιώνουν.
Επίσης, ορίζοντας αυτές τις ιδιότητες δημιουργείτε ένα πλαίσιο την ώρα που φορτώνει η ιστοσελίδα με αποτέλεσμα να μην «χαλάει» η εμφάνισή της έως ότου φορτώσει.
Τέλος, να επισημάνουμε ότι υπάρχουν κάποιες ιδιότητες οι δεν περιλαμβάνονται πλέον στην HTML 5 όπως είναι οι: align, border, hspace, vspace

3.5.8 Σύνδεσμοι

Ετικέτα <a>

Δημιουργούμε συνδέσμους με την ετικέτα <a> (a από την λέξη Anchor). Η πιο βασική ιδιότητα της ετικέτας είναι η href, η οποία περιέχει το url ενός αρχείου. Το κείμενο που γράφεται ανάμεσα στην ετικέτα αρχής και τέλους είναι το κείμενο που φαίνεται στην οθόνη και πατάει ο χρήστης επάνω σε αυτό για να μεταφερθεί στην σελίδα που δείχνει η ιδιότητα href.
Μια επίσης χρήσιμη ιδιότητα είναι η target. Με την ιδιότητα αυτή ορίζουμε σε ποιο παράθυρο ή μέρος του παραθύρου θα εμφανιστεί η σελίδα που ανοίγουμε πατώντας τον σύνδεσμο με πιο συνήθης τιμή την _blank που ανοίγει τη σελίδα σε νέο παράθυρο.

3.5.9 Πίνακες

Οι πίνακες είναι μια δομή της HTML η οποία μας επιτρέπει να εμφανίσουμε κείμενα και γραφικά στοιχισμένα μέσα σε γραμμές και στήλες.
Με την ετικέτα **<table>** ορίζουμε έναν πίνακα. Το ζεύγος των ετικετών **<tr>...</tr>** ορίζει μια γραμμή του πίνακα, ενώ το ζεύγος των ετικετών **<td>...</td>** ορίζουν ένα κελί στην γραμμή του πίνακα. Με την ετικέτα **<th>** ορίζουμε μια επικεφαλίδα σε μια στήλη του πίνακα. Το κείμενο που βρίσκετε μέσα στις ετικέτες <th> και </th> εμφανίζεται με bold χαρακτήρες. Η ετικέτες <th>...</th> τοποθετούνται μέσα στις ετικέτες <tr>...</tr> όπως οι ετικέτες <td>...</td>.
Είναι σημαντικό επίσης να αναφέρουμε ότι πριν την έλευση του CSS ο σχεδιασμός των ιστοσελίδων γινόταν κατεξοχήν με πίνακες κάτι που πλέον έχει περιοριστεί.

3.5.10 Φόρμες

Αν και ο κύριος σκοπός μιας ιστοσελίδας είναι η δημοσίευση πληροφοριών στο Internet, δεν είναι ο μοναδικός. Όλο και περισσότερα sites χρησιμοποιούν την αμφίδρομη επικοινωνία, δηλαδή δεν παρέχουν απλά πληροφορίες προς τους επισκέπτες, αλλά ζητούν από αυτούς να επιλέγουν ή να πληκτρολογούν στοιχεία.
Ο επισκέπτης, αφού συμπληρώσει τα πεδία πατάει το κουμπί.
Με το πάτημα του κουμπιού τα δεδομένα που συμπλήρωσε ο επισκέπτης στέλνονται στον Server μέσω του πρωτοκόλλου HTTP. Έπειτα ο Server επεξεργάζεται με κάποιο πρόγραμμα script τις τιμές των πεδίων και επιστρέφει στον browser του επισκέπτη μια HTML σελίδα. Τα περιεχόμενα της σελίδας αυτής μπορεί να είναι οτιδήποτε, από μια απλή απάντηση μέχρι επιστροφή αποτελέσματος σε ερώτηση προς μια περίπλοκη Βάση Δεδομένων.

Η ετικέτα <form>

Για να δημιουργήσουμε μια φόρμα χρησιμοποιούμε τις ετικέτες <form> και </form>. Οι κυριότερες ιδιότητες της ετικέτας είναι οι name, method και action.

Με την ιδιότητα name δίνουμε ένα όνομα στην φόρμα ώστε να μπορούμε να αναφερόμαστε σε αυτή μέσα από τον κώδικα ενός script. Καλό είναι πάντα να δίνουμε όνομα στις Φόρμες που κατασκευάζουμε, ιδιαίτερα στις περιπτώσεις που έχουμε περισσότερες από μια Φόρμες στην ίδια σελίδα.

Η ιδιότητα action περιέχει το URL του αρχείου script στον Server το οποίο θα επεξεργαστεί τα στοιχεία της Φόρμας. Το script είναι ένα πρόγραμμα το οποίο τρέχει στον Server και το οποίο μεταξύ των άλλων μπορεί να δέχεται σαν είσοδο δεδομένα τα οποία λαμβάνει ο Server από τον browser του επισκέπτη (συνήθως από μια Φόρμα). Το script , χρησιμοποιώντας μια script γλώσσα προγραμματισμού (π.χ. asp , php κτλ.) , επεξεργάζεται τα δεδομένα αυτά και έπειτα επιστρέφει στον browser μια HTML σελίδα. Η επεξεργασία αυτή μπορεί να είναι από απλές πράξεις μεταξύ των δεδομένων έως και αναζήτηση σε μια περίπλοκη Βάση Δεδομένων με βάση τα δεδομένα αυτά (ένα τέτοιο script περιέχει κατάλληλες εντολές ώστε να συνδεθεί σε μια Βάση Δεδομένων και να προσπελάσει αλλα και να προσθέσει και να διαγράψει εγγραφές)

Η method καθορίζει τον τρόπο με τον οποίο στέλνονται τα δεδομένα της Φόρμας στον Server που βρίσκεται το πρόγραμμα script που θα τα επεξεργαστεί και μπορεί να πάρει τις τιμές get ή post. Με την μέθοδο get τα δεδομένα προσθέτονται στο τέλος του URL που "δείχνει" η ιδιότητα action και χωρίζονται από το σύμβολο &. Με την μέθοδο post τα δεδομένα στέλνονται ξεχωριστά από το URL . Στην περίπτωση post το αρχείο script παίρνει τα δεδομένα της Φόρμας μέσω της στάνταρ εισόδου. Παρακάτω η ιδιότητα method αναφέρεται πιο αναλυτικά.

Μέσα στα όρια των ετικετών <form> και </form> εκτός απο τα στοιχεία της Φόρμας μπορούμε επίσης να προσθέσουμε κείμενο (συνήθως Λεζάντες των στοιχείων) μαζί με ετικέτες μορφοποίησης

Η ετικέτα <input>

Με την ετικέτα <input> εισάγουμε τα περισσότερα στοιχεία της φόρμας. Οι κυριότερες ιδιότητες της ετικέτας είναι η type η οποία καθορίζει τον τύπο του στοιχείου της Φόρμας (Πεδίο Κειμένου ή Περιοχή Κειμένου ή Κουμπί Επιλογών ή Κουτί Πολλαπλών Επιλογών ή Κουμπί), η name με την οποία δίνουμε ένα μοναδικό όνομα στο στοιχείο της φόρμας (δεν πρέπει να υπάρχουν στοιχεία φόρμας με τα ίδια ονόματα) και η value με την οποία δίνουμε μια αρχική τιμή στο στοιχείο της φόρμας. Η ετικέτα <input> δεν έχει ετικέτα τέλους.

Για να εισάγουμε ένα Πεδίο Κειμένου χρησιμοποιούμε την ετικέτα <input> ορίζοντας την τιμή text στην ιδιότητα type.

Η χρήση της ιδιότητας *value* σε αυτήν την περίπτωση εμφανίζει μέσα στο πλαίσιο κειμένου την τιμή της ιδιότητας. Μια άλλη ιδιότητα που μπορεί να πάρει η ετικέτα <input> είναι η size η οποία καθορίζει τον αριθμό των ορατών χαρακτήρων που μπορεί να χωρέσει το Πεδίο Κειμένου. Η ιδιότητα maxlength καθορίζει τον αριθμό χαρακτήρων που μπορεί να πληκτρολογήσει ο επισκέπτης.
Για να ένα πεδίο τύπου **password** έτσι ώστε να εμφανίζονται βουλίτσες ή αστεράκια κατά την πληκτρολόγηση, απλά βάζουμε την τιμή password στην ιδιότητα type της ετικέτας <input>.

Για να εισάγουμε **κουμπιά επιλο**γής χρησιμοποιούμε την ετικέτα <input> ορίζοντας την τιμή **radio** στην ιδιότητα type.

Για να εισάγουμε **Κουτιά Πολλαπλών Επιλογών** χρησιμοποιούμε την ετικέτα <input> ορίζοντας την τιμή **checkbox** στην ιδιότητα type.
Η ιδιότητα checked της ετικέτας <input> χρησιμοποιείται για να ορίσουμε ποια από τα ομαδοποιημένα κουμπιά θα είναι εξορισμού επιλεγμένο όταν φορτώνεται η σελίδα.

Για να εισάγουμε Περιοχή Κειμένου στην Φόρμα μας χρησιμοποιούμε την ετικέτα <textarea>.

Για να εισάγουμε **Κρυφά Πεδία** στην φόρμα μας χρησιμοποιούμε την ετικέτα <input> ορίζοντας την τιμή **hidden** στην ιδιότητα type.
Τα κρυφά πεδία δεν εμφανίζονται στον browser. Χρησιμοποιούμε κρυφά πεδία όταν θέλουμε να περάσουμε στο αρχείο script μια τιμή μαζί με τις υπόλοιπες τιμές που πληκτρολογεί ή επιλέγει ο επισκέπτης.

Η ετικέτα <SELECT>

Για να εισάγουμε Λίστα Επιλογών στην Φόρμα μας χρησιμοποιούμε την ετικέτα <select>.
Η ετικέτα <option> τοποθετείται μέσα στις ετικέτες <SELECT> και </select> και αντιστοιχεί σε μια επιλογή της λίστας. Όσες επιλογές επιθυμούμε να περιλαμβάνει η Λίστα Επιλογών τόσες ετικέτες <option> πρέπει να προσθέσουμε. Μεταξύ των ετικετών <option> και </option> γράφουμε το κείμενο που θέλουμε να εμφανίζεται στην λίστα
Με την ιδιότητα value της <option> ορίζουμε την τιμή που θα σταλεί στον Server.

Το κουμπί Υποβολής της Φόρμας

Το πάτημα του Κουμπιού Υποβολής της Φόρμας στέλνει στον Server τα δεδομένα που πληκτρολόγησε ο χρήστης. Για να εισάγουμε στην Φόρμα μας Κουμπί Υποβολής χρησιμοποιούμε την ετικέτα <input> ορίζοντας την τιμή Submit στην ιδιότητα type. Η ιδιότητα value ορίζει το κείμενο στο Κουμπί Υποβολής.

Αποστολή στοιχείων στον Server (Μέθοδος GET και POST)

Η αποστολή των τιμών από την φόρμα του browser προς το αρχείο script του Server που θα τα επεξεργαστεί, γίνεται με βάση την ιδιότητα method (μέθοδος) της ετικέτας <form> η οποία μπορεί να πάρει τις τιμές get και post.

Όταν η τιμή της ιδιότητας method είναι get, τα δεδομένα αποστέλλονται μέσω τις URL με τη μορφή string το οποίο απαρτίζεται από ζεύγη ονόματος / τιμής για κάθε δεδομένο. Όταν η τιμή της ιδιότητας method είναι post, τα δεδομένα αποστέλλονται απευθείας στον χειριστή φόρμας της εφαρμογής για επεξεργασία στον εξυπηρετητή.

Η μέθοδος get χρησιμοποιείται συνήθως όταν απαιτούνται στατικά δεδομένα από τον εξυπηρετητή για προσωρινή χρήση ή όταν αυτά θα χρησιμοποιηθούν ξανά σύντομα. Ο Server αποθηκεύει όλη την συμβολοσειρά του URL μετά το λατινικό ερωτηματικό στην μεταβλητή περιβάλλοντος QUERY_STRING .

Η μέθοδο post κάνει το ίδιο πράγμα με την μέθοδο get , με την διαφορά ότι στέλνει τα δεδομένα της φόρμας σε ξεχωριστή ροή δεδομένων (data stream) και όχι μαζί με το URL όπως με την μέθοδο GET. Ο Server δέχεται τα δεδομένα αυτά και τα αποθηκεύει σε ένα προσωρινό αρχείο και έπειτα τα περνάει μέσα στο script για να τα επεξεργαστεί. Ορισμένοι Server αντί να αποθηκεύουν τα δεδομένα σε κάποιο αρχείο χρησιμοποιούν για την είσοδο των δεδομένων την standard είσοδο (standard input).

3.6 Επίλογος

Το HTML δεν έχει αλλάξει τεχνικά όλα αυτά τα χρόνια. Ακόμα και η δημιουργία του νέου πρότυπου HTML που υπόσχεται πολλά στη σχεδίαση των ιστοσελίδων, βασικό στοιχείο μορφοποίησης της ιστοσελίδας είναι το CSS.

ΚΕΦΑΛΑΙΟ 4

CSS

4.1 Εισαγωγή

CSS σημαίνει *C*ascading *S*tyle *S*heets και *είναι στυλ που μπορούμε να ορίσουμε για τις HTML σελίδες*. Με τα στυλ ορίζουμε το χρώμα, το μέγεθος της γραμματοσειράς, την γραφή (bold, underline, κτλ.), το χρώμα του φόντου, τις διαστάσεις, την τιμή padding και μια σειρά από άλλες ιδιότητες των στοιχείων μιας ιστοσελίδας.

Στοιχείο είναι ένα οποιοδήποτε μέρος της HTML σελίδας, όπως: μια εικόνα, μια παράγραφος, μια λίστα, μια επιλογή μιας λίστας, μια επικεφαλίδα, ένα κείμενο ή μια λέξη που βρίσκεται μέσα σε ετικέτες διαμόρφωσης κτλ.

Κάθε στοιχείο λοιπόν, έχει 3 ιδιότητες που αφορούν την σχέση τους με τα υπόλοιπα γειτονικά στοιχεία, αλλά και την εμφάνιση τους:

- ο κενός χώρος μεταξύ του πλαισίου και των γειτονικών στοιχείων (margin)
- το πλαίσιο (border)
- ο κενός χώρος μεταξύ του περιεχομένου του στοιχείου και του πλαισίου του (padding)

Η w3.org, η οποία ανέπτυξε και διαχειρίζεται τα πρότυπα της CSS και της HTML, ομαδοποίησε αυτές τις ιδιότητες χρησιμοποιώντας τον ορισμό Box model, ο οποίος απεικονίζεται παρακάτω:

Εικόνα 1 - box-model

- Το margin καθορίζει την περιοχή έξω από το πλαίσιο (border) του στοιχείου. Ο χώρος αυτός δεν έχει χρώμα, καθώς είναι πάντα transparent (διαφανές)
- Το border ορίζει το πάχος και το χρώμα του πλαισίου γύρω απο το στοιχείο
- Το padding ορίζει την περιοχή μεταξύ του πλαισίου (border) και του περιεχομένου του στοιχείου. Η περιοχή αυτή μπορεί να έχει background ένα χρώμα ή μια εικόνα
- Το περιεχόμενο είναι το κείμενο ή η εικόνα

4.2 Πως βάζουμε CSS σε μια html σελίδα

Υπάρχουν 3 τρόποι να γίνει αυτό:

Τοποθέτηση στην ετικέτα head των CSS ενολών:

```
<head>
...
    <style type="text/css" media="screen">
    <!--
    --CSS εδώ--
    -->
    </style>
...
</head>
```

Τοποθέτηση μιας αναφοράς σε αρχείο που περιέχει τις εντολές:

Ο δεύτερος τρόπος είναι να δημιουργήσουμε ένα *εξωτερικό αρχείο στυλ* με επέκταση *.css* στο οποίο γράφουμε τα στυλ που θέλουμε το ένα κάτω απο το άλλο. Η σύνδεση του εξωτερικού αρχείου στυλ και της σελίδας HTML γίνεται με την χρήση της HTML ετικέτας <link> στο *τμήμα HEAD της HTML σελίδας*, όπως στον παρακάτω κώδικα. Η ετικέτα <link> εισάγει στην σελίδα τα στυλ που βρίσκονται στο εξωτερικό αρχείο .css

```
<link rel="stylesheet" type="text/css" href="path/styles.css" media="screen" />
```

Τοποθέτηση του στυλ μέσα στις ετικέτες

Μπορούμε να ορίσουμε στυλ στις ετικέτες που επιθυμούμε, χρησιμοποιώντας την κοινή ιδιότητα style της HTML. Η ιδιότητα *style* μπορεί να μπει σχεδόν σε όλες τις HTML ετικέτες της ενότητας <*body*>.

<p style=”font-color: 12px; color: #000000”;>κείμενο

4.3 Το συντακτικό των CSS

Τα CSS stylesheets χωρίζονται σε 3 μέρη:

- τον *Επιλογέα (selector)*,
- τις *Ιδιότητες (attributes)* του επιλογέα και
- τις *Τιμές (values)* των Ιδιοτήτων του επιλογέα

Ο επιλογέας καθορίζει τι αντικείμενο της ιστοσελίδας μας θα επηρεάσει η μορφοποίηση, η ιδιότητα καθορίζει τι χαρακτηριστικό του αντικειμένου θα επηρεαστεί και η τιμή είναι η τιμή που θα πάρει αυτό το χαρακτηριστικό. Ας δούμε όμως περισσότερες λεπτομέρειες:

Ο Επιλογέας (selector)

*****

Όταν ο επιλογέας είναι ένας χαρακτήρας αστερίσκου, τότε οι ιδιότητες που θα γράψουμε σε αυτόν τον κανόνα CSS εφαρμόζονται σε κάθε στοιχείο της σελίδας μας. Όπως είναι κατανοητό, συνήθως δεν είναι και πολύ χρήσιμος επιλογέας από μόνος του, και χρησιμοποιείται κυρίως σε συνδυασμό με άλλους.

στοιχείο

Όταν ο επιλογέας αποτελείται απλά από το όνομα ενός html tag, τότε οι ιδιότητες που θα γράψουμε σε αυτόν τον κανόνα CSS εφαρμόζονται σε κάθε τέτοιο στοιχείο html. Για παράδειγμα, ο επιλογέας p θα εφαρμοστεί σε οτιδήποτε στη σελίδα μας περιλαμβάνεται εντός των tags ... , ο επιλογέας table θα εφαρμοστεί σε όλους τους πίνακες στη σελίδα μας, ο επιλογέας img θα αφορά όλες τις εικόνες στη σελίδα κοκ. Προφανώς όταν θέλουμε να εφαρμόσουμε κάποιες ιδιότητες CSS σε ολόκληρη τη σελίδα, χρησιμοποιούμε ως επιλογέα body μιας και όλο το ορατό τμήμα της σελίδας περιέχεται εντός των tags <body>...</body>.

.όνομα_κλάσης

Όταν ο επιλογέας μας περιλαμβάνει μια τελεία (.) στην αρχή του, τότε ο browser ψάχνει όσα στοιχεία στη σελίδα μας περιλαμβάνουν την ιδιότητα *class* και εφαρμόζει τις ιδιότητες που θα γράψουμε στον κανόνα CSS αυτό σε οποιοδήποτε στοιχείο περιλαμβάνει την κλάση «όνομα_κλάσης» στην ιδιότητα class του. Φυσικά ως *όνομα_κλάσης* μπορούμε να γράψουμε οτιδήποτε αποτελείται από γράμματα, αριθμούς, παύλες και χαρακτήρες underscore (_) και να ξεκινάει με γράμμα. Αξίζει να σημειωθεί ότι μπορεί το ίδιο στοιχείο να ανήκει σε περισσότερες από μια κλάσεις, διαχωρισμένες με κενά μέσα στην *class* html attribute του. Πχ <p class=”emphasis bodytext”>... .

Για παράδειγμα, ο παρακάτω κανόνας CSS:

```
.emphasis
{
color: red;
}
```

θα κάνει κόκκινα τα γράμματα και στο στοιχείο blah blah , και στο στοιχείο <div class=”emphasis otherclass”>blah blah αλλά **όχι** στο στοιχείο <h1 class=”otherclass”>blah blah</h1>.

Οι κλάσεις γενικά χρησιμοποιούνται όταν θέλουμε να **ομαδοποιήσουμε** κάποια στοιχεία html για τα οποία δεν μπορούμε να βρούμε κάποιον άλλο επιλογέα που να αφορά **όλα αυτά** και **μόνον αυτά**, οπότε τους προσδίδουμε μια συγκεκριμένη κλάση, ώστε να μπορούμε στο CSS μας να αναφερθούμε **μόνο σε αυτά** και να τα μορφοποιήσουμε.

στοιχείο.όνομα_κλάσης

Αποτελεί ουσιαστικά συνδυασμό των δύο παραπάνω επιλογέων. Εφαρμόζεται σε όσα στοιχεία αποτελούνται από το html tag <*στοιχείο*> και ανήκουν στην κλάση *όνομα_κλάσης*. Πχ ο επιλογέας p.emphasis εφαρμόζεται σε ο,τι περιέχεται σε tags της μορφής ... . Ο επιλογέας αυτός είναι χρήσιμος όταν έχουμε πολλά **διαφορετικού τύπου** στοιχεία **με την ίδια κλάση** και επιθυμούμε να εφαρμόσουμε **διαφορετική μορφοποίηση** ανάλογα με τον **τύπο** του στοιχείου.

#όνομα_id

Όταν ο επιλογέας μας περιλαμβάνει ένα χαρακτήρα δίεσης (#) στην αρχή του, τότε ο browser εφαρμόζει τις ιδιότητες που θα γράψουμε στο στοιχείο το οποίο

περιλαμβάνει την ιδιότητα id=”όνομα_id”. **Δεν πρέπει να υπάρχουν δύο (ή περισσότερα) στοιχεία στη σελίδα μας με το ίδιο id.** Τα ids διέπονται από τους ίδιους κανόνες ονοματολογίας με τις κλάσεις. Ουσιαστικά, ο,τι μπορούμε να κάνουμε με τα ids μπορούμε να το κάνουμε και με τη χρήση κλάσεων, απλά όταν το στοιχείο που θέλουμε να μορφοποιήσουμε είναι **μοναδικό**, είναι γενικά καλύτερο να χρησιμοποιούμε ids.
στοιχείο[attribute=”value”] Αποτελεί ουσιαστικά μια «επέκταση» του επιλογέα *στοιχείο* που αναλύθηκε πρώτος. Ο εν λόγω επιλογέας, κάνει τον browser να εφαρμόζει τις ιδιότητες που θα γράψουμε σε αυτόν σε κάθε στοιχείο με tag <*στοιχείο*> το οποίο επιπροσθέτως έχει την τιμή *value* στην html ιδιότητα *attribute*. Παραδείγματος χάριν, ο επιλογέας input[type=”submit”] αφορά όλα τα κουμπιά υποβολής φόρμας που υπάρχουν στη σελίδα μας, χωρίς ωστόσο να εφαρμόζεται σε άλλα στοιχεία φορμών όπως τα πεδία κειμένου (στα οποία η ιδιότητα type είναι text). Άλλο ένα παράδειγμα: Έστω ότι θέλουμε να μορφοποιήσουμε μόνο όσους πίνακες στη σελίδα μας είναι κεντραρισμένοι. Αν χρησιμοποιούσαμε ως επιλογέα table, τότε οι ιδιότητες που θα γράφαμε σε αυτόν τον επιλογέα θα εφαρμόζονταν σε όλους τους πίνακες ανεξαιρέτως. Ενώ αν χρησιμοποιήσουμε τον επιλογέα table[align=”center”] τότε ο κανόνας CSS που θα γράψουμε θα εφαρμοστεί μόνο σε όσους πίνακες έχουν την ιδιότητα align=”center”.

4.4 Ψευδοκλάσεις (Pseudo-classes)

Οι *ψευδοκλάσεις* χρησιμοποιούνται για να καθορίζονται διαφορετικά στυλ σε επιλογείς σε διαφορετικές καταστάσεις.

Για να καθοριστεί το στυλ μιας ψευδοκλάσης χρησιμοποιούμε το παρακάτω συντακτικό:

επιλογέας:ψευδοκλάση { ιδιότητα: τιμή }

Οι ψευδοκλάσεις μπορούν να χρησιμοποιηθούν σε συνδυασμό με κανονικές κλάσεις:

επιλογέας.κλάση:ψευδοκλάση { ιδιότητα: τιμή }

Μερικές από τις ψευδοκλάσεις είναι οι :

- *:link*, εφαρμόζεται το στυλ με την ψευδοκλάση αυτή, όταν ο χρήστης **δεν** έχει επισκεφθεί παλαιότερα τον σύνδεσμο
- *:visited*, εφαρμόζεται το στυλ με την ψευδοκλάση αυτή, όταν ο χρήστης έχει επισκεφθεί παλαιότερα τον σύνδεσμο
- *:hover*, εφαρμόζεται το στυλ με την ψευδοκλάση αυτή, όταν ο χρήστης τοποθετεί τον δείκτη του ποντικιού του επάνω απο τον σύνδεσμο
- *:active*, εφαρμόζεται το στυλ με την ψευδοκλάση αυτή, όταν ο χρήστης έχει πατημένο το αριστερό κλικ του ποντικιού του επάνω στον σύνδεσμο

Πέρα από τις παραπάνω ψευδοκλάσεις υπάρχουν και οι:

- *:first-child*, εφαρμόζει το στυλ στην πρώτη εμφάνιση ενός επιλογέα σε μία σελίδα όπως για παράδειγμα την πρώτη επιλογή σε μία unsorted list.
- *:last-child*, εφαρμόζει το στυλ στην πρώτη εμφάνιση ενός επιλογέα σε μία σελίδα όπως για παράδειγμα την πρώτη επιλογή σε μία unsorted list.

4.5 Ψευδοστοιχεία (Pseudo-elements)

Τα *ψευδοστοιχεία* της CSS χρησιμοποιούνται για να προσθέσουν εφέ σε ορισμένα στοιχεία της σελίδας μας
Τα ψευδοστοιχεία είναι τα παρακάτω :

- *:first-line*, χρησιμοποιείται για να προσθέσει στυλ στην πρώτη γραμμή του κειμένου του στοιχείου που αναφέρετε ο επιλογέας του στυλ
- *:first-letter*, χρησιμοποιείται για να προσθέσει στυλ στον πρώτο γράμμα του κειμένου του στοιχείου που αναφέρετε ο επιλογέας του στυλ
- *:before*, χρησιμοποιείται για να τοποθετήσει περιεχόμενα (συνήθως μιας εικόνας) πριν ένα επιλογέα
- *:after*, χρησιμοποιείται για να τοποθετήσει περιεχόμενα (συνήθως μιας εικόνας) μετά από ένα επιλογέα

4.6 Τύποι μέσων εμφάνισης (Media Types)

Μια ακόμα χρήσιμη δυνατότητα του css είναι ότι μπορούμε να ορίζουμε διαφορετικά στυλ ανάλογα με το μέσο που θα εμφανιστεί μια ιστοσελίδα.

Τύποι μέσων που μπορούμε να χρησιμοποιήσουμε είναι οι παρακάτω:

- screen: για οθόνες υπολογιστών
- print: για εκτυπωτές
- handheld: για συσκευές χειρός
- projection: για παρουσιάσεις
- braille: for braille tactile feedback devices
- aural: for speech and sound synthesisers
- embossed: for paged braille printers
- tv: για οθόνες σαν της τηλεόρασης
- tty: για τερματικά
- all: για όλους τους τύπους εμφάνισης

4.7 Πλεονεκτήματα χρήσης CSS έναντι της μορφοποίσηςη μέσω HTML attribute

- Πολύ μεγαλύτερη ευελιξία. Το CSS κατέστησε εφικτές μορφοποιήσεις οι οποίες ήταν αδύνατες ή πολύ δύσκολες με την κλασσική HTML.
- Ευκολότερη συντήρηση των ιστοσελίδων. Η εμφάνιση ενός ολόκληρου site μπορεί να ελέγχεται από ένα μόνο εξωτερικό αρχείο CSS. Έτσι, κάθε αλλαγή στο στυλ της ιστοσελίδας μπορεί να γίνεται με μια μοναδική αλλαγή σε αυτό το αρχείο, αντί για την επεξεργασία πολλών σημείων σε κάθε σελίδα που υπάρχει στο site.
- Μικρότερο μέγεθος αρχείου, δεδομένου ότι ο κάθε κανόνας μορφοποίησης γράφεται μόνο μια φορά και όχι σε κάθε σημείο που εφαρμόζεται.
- Καλύτερο SEO (Search engine optimization). Οι μηχανές αναζήτησης δεν «μπερδεύονται» ανάμεσα σε περιεχόμενο και τη μορφοποίηση του, αλλά έχουν πρόσβαση στο περιεχόμενο σκέτο, οπότε είναι πολύ ευκολότερο να το καταγράψουν και να το αρχειοθετήσουν (indexing).
- Γρηγορότερες σελίδες. Όταν χρησιμοποιούμε εξωτερικό αρχείο CSS ο browser την πρώτη φορά που θα φορτώσει κάποια σελίδα του site μας το αποθηκεύει στην cache, οπότε δεν χρειάζεται να το κατεβάσει ξανά κάθε φορά που κατεβάζει ο χρήστης του κάποια άλλη σελίδα του site μας

4.8 Επίλογος

Όπως γράφτηκε πιο πάνω, η HTML χρησιμοποιείται για να δομήσει το περιεχόμενο μιας ιστοσελίδας και το CSS χρησιμοποιείται να τη διαμόρφωση ή μορφοποίηση του περιεχομένου. Το επόμενο που μπορούμε να προσθέσουμε σε μία ιστοσελίδα είναι η διαδραστικότητα η οποία και αναλύεται στο επόμενο κεφάλαιο.

ΚΕΦΑΛΑΙΟ 5

JAVASCRIPT και jQuery

5.1 Τι είναι Javascript

Η JavaScript είναι γλώσσα προγραμματισμού η οποία έχει σαν σκοπό την παραγωγή δυναμικού περιεχομένου και την εκτέλεση κώδικα στην πλευρά του πελάτη (client-side) σε ιστοσελίδες. Το πρότυπο της γλώσσας κατά τον οργανισμό τυποποίησης ECMA ονομάζεται ECMAscript.

5.1.1 Μοντέλο εκτέλεσης

Η αρχική έκδοση της Javascript βασίστηκε στη σύνταξη στη γλώσσα προγραμματισμού C, αν και έχει εξελιχθεί, ενσωματώνοντας πια χαρακτηριστικά από νεότερες γλώσσες.

Αρχικά χρησιμοποιήθηκε για προγραμματισμό από την πλευρά του πελάτη (client), που ήταν ο φυλλομετρητής (browser) του χρήστη, και χαρακτηρίστηκε σαν client-side γλώσσα προγραμματισμού. Αυτό σημαίνει ότι η επεξεργασία του κώδικα Javascript και η παραγωγή του τελικού περιεχομένου HTML δεν πραγματοποιείται στο διακομιστή, αλλά στο πρόγραμμα περιήγησης των επισκεπτών, ενώ μπορεί να ενσωματωθεί σε στατικές σελίδες HTML. Αντίθετα, άλλες γλώσσες όπως η PHP εκτελούνται στο διακομιστή (server-side γλώσσες προγραμματισμού).
Παρά την ευρεία χρήση της Javascript για συγγραφή προγραμμάτων σε περιβάλλον φυλλομετρητή, αξίζει να σημειωθεί ότι από την αρχή χρησιμοποιήθηκε και για τη συγγραφή κώδικα από την πλευρά του διακομιστή, από την ίδια τη Netscape στο προϊόν LiveWire, με μικρή επιτυχία. Η χρήση της Javascript στο διακομιστή εμφανίζεται πάλι σήμερα, με τη διάδοση του Node.js, ενός μοντέλου προγραμματισμού βασισμένο στα γεγονότα (events).

5.1.2 Javascript και Java

Η Javascript δεν θα πρέπει να συγχέεται με τη Java, που είναι διαφορετική γλώσσα προγραμματισμού και με διαφορετικές εφαρμογές. Η χρήση της λέξης "Java" στο όνομα της γλώσσας έχει περισσότερη σχέση με το προφίλ του

προϊόντος που έπρεπε να έχει και λιγότερο με κάποια πιθανή συμβατότητα ή άλλη στενή σχέση με τη Java. Ρόλο σε αυτήν τη σύγχυση έπαιξε και ότι η Java και η Javascript έχουν δεχτεί σημαντικές επιρροές από τη γλώσσα C, ειδικά στο συντακτικό, ενώ είναι και οι δύο αντικειμενοστρεφείς γλώσσες. Τονίζεται ότι ο σωστός τρόπος γραφής της είναι "Javascript" και όχι 'Java script' σαν δύο λέξεις, όπως λανθασμένα γράφεται ορισμένες φορές.

5.2 Χρήσεις της Javascript

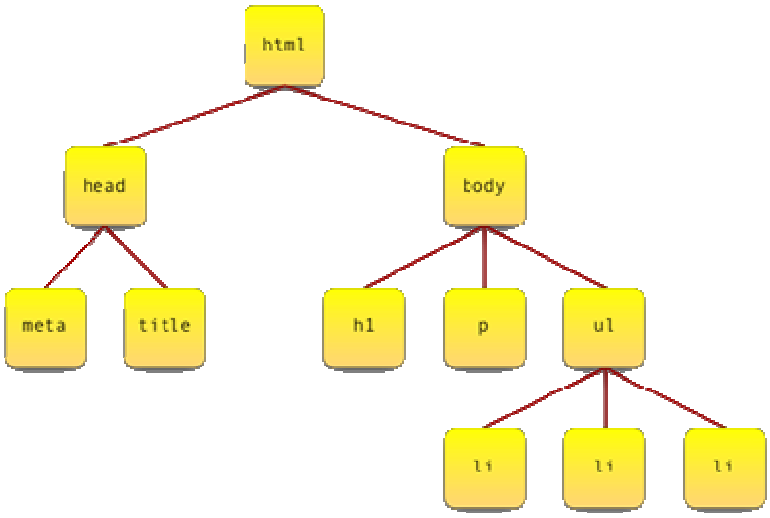
- Η χρήση της javascript χρησιμοποιείται κυρίως για να εξυπηρετήσει τους παρακάτω σκοπούς:
- Είναι κάτι παραπάνω από μία απλή browser scripting language.
- Πρόκειται για ένα πολύ δυνατό εργαλείο στο χώρο του WebDevelopment.
- Υποστηρίζει concepts όπως object oriented programming, recursion, lambdas και closure
- Μας επιτρέπει να μεταβάλουμε δυναμικά τον HTML κώδικα αφού έχει φτάσει το browser του χρήστη.
- Αντιδρά σε events.
- Μπορεί να κάνει validate τα data που έχει συμπληρώσει έναα χρήστης σε μια φόρμα.
- Επιτρέπει την υλοποίηση τεχνικών AJAX.

5.3 JavaScript και Document Object Model (DOM)

- το DOM είναι ένα ουδέτερο σε λειτουργικό και γλώσσα προγραμματισμού περιβάλλον το οποίο επιτρέπει σε προγράμματα και script να έχουν πρόσβαση και να ανανεώνουν το περιεχόμενο, τη δομή και το στυλ των εγγράφων. Το έγγραφο μπορεί να επεξεργαστεί περαιτέρω και τα αποτελέσματα αυτής της επεξεργασίας μπορούν να ενσωματωθούν στην σελίδα η οποία βρίσκεται υπό παρουσίαση.
- Το πιο σημαντικό στοιχείο το οποίο χρησιμοποιείται από το DOM είναι η
- αναπαράσταση του εγγράφου ως ένα δέντρο. Πιο συγκεκριμένα, ολόκληρο το
- έγγραφο αναπαριστάται ως ένα οικογενειακό δέντρο.

- Το μοντέλο του οικογενειακού δέντρου μπορεί να αναπαραστήσει ένα κείμενο γραμμένο σε (X)HTML. Για παράδειγμα ο παρακάτω κώδικάς αντικατοπτρίζει το δέντρο στο σχήμα II

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <title>Shopping list</title>
  </head>
  <body>
    <h1>What to buy</h1>
    <p title="a gentle reminder">Don't forget to buy this stuff.</p>
    <ul id="purchases">
      <li>A tin of beans</li>
      <li>Cheese</li>
      <li>Milk</li>
    </ul>
  </body>
</html>
```



Εικόνα 2 – Element nodes

5.3.1 Βασικά χαρακτηριστικά ενός DOM εγγράφου

5.3.1.1 Κόμβοι

Ο όρος κόμβος χρησιμοποιείται για να δείξει ένα σημείο σύνδεσης ανάμεσα σε δύο στοιχεία του δέντρου. Παρακάτω περιγράφονται οι τρεις τύποι κόμβων του DOM.

5.3.1.1.1 Element node

Όπως προειπώθηκε, το κύριο συστατικό του DOM είναι οι κόμβοι. Ένα είδος κόμβου είναι το στοιχείο (element). Τέτοια element είναι το <body>, το <p>, το <ul> κτλ. Η ετικέτα (tag) δίνει το όνομα των element. Τα element μπορούν να περιέχουν άλλα element όπως για παράδειγμα ενός <ul> που περιέχει <li>. Μόνο το στοιχείο <html> δεν περιέχεται σε άλλο element και για αυτό και ονομάζεται και ως στοιχείο ρίζα (root).

5.3.1.1.2 Text node

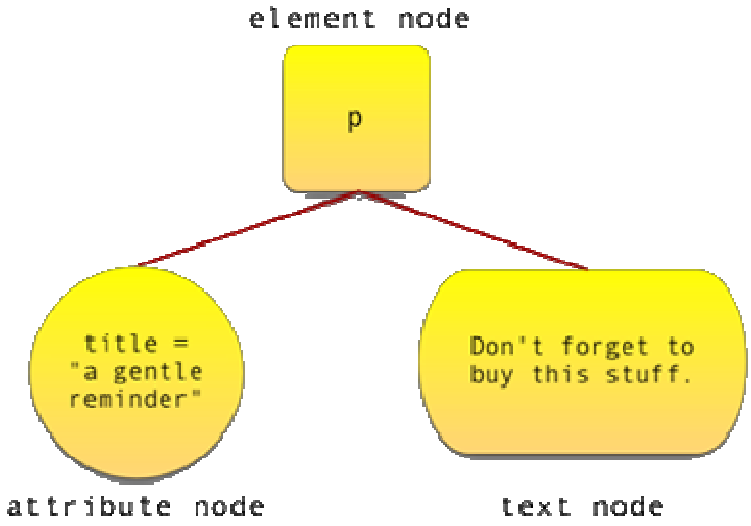
Άλλο είδος node είναι τα text node. Αν ένα έγγραφο αποτελούνταν μόνο από άδεια στοιχεία τότε αυτό θα είχε δομή αλλά καθόλου περιεχόμενο. Έτσι, χρησιμοποιούνται τα text node. Για παράδειγμα ένα element <p> περιέχει τον text node “Hello world”. Στην XHTML τα text nodes περιέχονται πάντα μέσα σε element nodes. Αλλά όλα τα element nodes δεν περιέχουν text nodes. Έτσι, ένα <ol> περιέχει <li> που αυτά περιέχουν text node.

5.3.1.1.3 Attribute node

Άλλο είδος node είναι το attribute node. Τέτοιο για παράδειγμα είναι το <title> που χρησιμοποιείται για να προσδώσει μία πιο συγκεκριμένη πληροφορία για το παρακάτω στοιχείο <p>.

Σημειώνεται ότι δεν περιέχουν όλα τα στοιχεία ιδιότητες, αλλά όλες οι ιδιότητες περιέχονται από στοιχεία.

Για να γίνει αυτό πιο κατανοητό παρατίθεται το Σχήμα ΙΙΙ:



Εικόνα 3: Elements και attributes

5.4 jQuery

Η jQuery είναι μια **βιβλιοθήκη (framework)** JavaScript που χρησιμοποιείται από προγραμματιστές για τη ταχεία ανάπτυξη ιστοσελίδων και διαδυκτιακών εφαρμογών που χρειάζονται μεγάλη **ευχρηστία** και **διαδραστικότητα** (interactivity). Η jQuery πρωτοεμφανίστηκε τον Ιανουάριο του 2006 στο BarCamp από τον John Resig. Πρόκειται για μια βιβλιοθήκη Javascript ανοιχτού κώδικα, υπό τις άδειες MIT License και την GNU General Public License.

Την βιβλιοθήκη jQuery μπορούμε να την κατεβάσουμε από την ιστοσελίδα http://www.jquery.com

5.4.1 Πλεονεκτήματα της jQuery

- Ακολουθεί την αρχή KISS (Keep It Simple Stupid): Η βιβλιοθήκη JQuery προσπαθεί να υπεραπλουστεύσει τον προγραμματισμό σε Javascript προσφέροντας πραγματικά απλούς μηχανισμούς και εντολές μέσω του framework της.
- Παρέχει πλήρη και αναλυτικότατη τεκμηρίωση που συμπληρώνεται από την εκτεταμένη παρουσία ηλεκτρονικών βοηθημάτων. Εκτός από την πολύ

καλοδουλεμένη τεκμηρίωσή του JQuery, οι ενδιαφερόμενοι μπορούν να ανατρέξουν και στην σελίδα του Visual JQuery όπου μπορούν να βρουν μία εναλλακτική αλλά πολύ βολική, από άποψη δομής, τεκμηρίωση.

- Υποστηρίζεται από μία πάρα πολύ ενεργή κοινότητα: Όπως για τα περισσότερα open source έργα λογισμικού, έτσι και για το JQuery η ύπαρξη μιας κατά το μέγιστο δυνατό ενεργής κοινότητας αποτελεί τον ακρογωνιαίο λίθο για την ανάπτυξη και ευημερία του.
- Μικρό μέγεθος: Το γεγονός ότι το βασικό πακέτο της JQuery είναι μόλις 20Kb αφενός επιβεβαιώνει την πρώτη παρατήρηση, ότι δηλαδή η φιλοσοφία της έγκειται στην απλότητα και αφετέρου κάνει πολύ εύκολη την κατανόηση της αρχιτεκτονικής της.
- Ποικιλία χαρακτηριστικών: Η JQuery δίνει τη δυνατότητα στον χρήστη να χρησιμοποιήσει σχεδόν το σύνολο των δυνατοτήτων που προσφέρει η γλώσσα JavaScript. Από απλά χαρακτηριστικά που σχετίζονται με βασικές λειτουργίες εμφάνισης / απόκρυψης, ως Ajax κλήσεις και σύνθετα εφέ.
- Επεκτασιμότητα: Η λογική με την οποία είναι φτιαγμένη η JQuery είναι απλή πράγμα που αντικατοπτρίζεται και στον ίδιο της τον κώδικα. Αυτό κάνει πολύ εύκολη την επέκτασή / τροποποίησή της.

5.4.2 Εισαγωγή στην ιστοσελίδα

Την jQuery μπορούμε να την ενσωματώσουμε στην ιστοσελίδα μας, εισάγοντας τον παρακάτω κώδικα στο head τμήμα μιας ιστοσελίδας HTML:

```
<script type="text/javascript" src="jquery.js"></script>
```

5.4.3 Σύνταξη jQuery

Η jQuery είναι σχεδιασμένη να **επιλέγει** στοιχεία HTML και να **εφαρμόζει** κάποιες ενέργειες στα επιλεγμένα στοιχεία. Η βασική σύνταξη είναι:

5.4.3.1 Η συνάρτηση Document Ready

Ίσως να έχετε παρατηρήσει πως όλες οι συναρτήσεις στην jQuery γράφονται μέσα σε μια συνάρτηση **document.ready()**:

```
$(document).ready(function(){
    // jQuery κώδικας
);
```

Αυτό γίνεται για να αποτρέψουμε των κώδικα μας να εκτελεστεί πριν ολοκληρωθεί το **φόρτωμα** της σελίδας. Αν δεν ολοκληρωθεί το φόρτωμα της σελίδας και προσπαθήσουμε να εκτελέσουμε κάποια κομμάτια κώδικα ίσως προκύψουν σφάλματα, για παράδειγμα φανταστείτε ο κώδικας μας να προσπαθεί να κρύψει ένα στοιχείο που δεν υπάρχει ακόμα στην σελίδα ή προσπαθεί να πάρει το ύψος από μια εικόνα που δεν έχει ακόμα φορτώσει.

5.4.3.2 jQuery Selectors

Ένα από τα κύρια **χαρακτηριστικά** και **πλεονεκτήματα** της jQuery είναι ο **τρόπος επιλογής** των στοιχείων σε μια ιστοσελίδα. Η jQuery μας παρέχει ένα σύνολο από selectors που μας επιτρέπουν να επιλέξουμε με **ακρίβεια** τα στοιχεία που μας ενδιαφέρουν.

(a) Element Selectors

Η jQuery χρησιμοποιεί τους selectors της **CSS** για να επιλέξει στοιχεία μέσα στο αρχείο HTML. Δείτε τα παρακάτω παραδείγματα:

```
// Επιλέγει όλα τα στοιχεία

$("p")

// Επιλέγει όλα τα στοιχεία
```

με κλάση intro

```
$("p.intro")

// Επιλέγει όλα τα στοιχεία
```

με id demo

```
$("p#demo")
```

(b) jQuery Attribute Selectors

Η jQuery χρησιμοποιεί εκφράσεις **XPath** για να επιλέξει στοιχεία με συγκεκριμένα **χαρακτηριστικά (attributes)**

```
// Επιλέγει όλα τα στοιχεία με href attribute
$("[href]")

// Επιλέγει όλα τα στοιχεία με href attribute που ισούται με "#"
$("[href='#']")

// Επιλέγει όλα τα στοιχεία με href attribute που ΔΕΝ ισούται με "#"
$("[href!='#']")

// Επιλέγει όλα τα στοιχεία με href attribute που τελειώνει σε ".jpg"
$("[href$='.jpg']")
```

(c) jQuery CSS Selectors

Η jQuery με την βοήθεια των CSS selectors μπορεί να αλλάξει τις ιδιότητες CSS των επιλεγμένων στοιχείων. Για παράδειγμα το παρακάτω απόσπασμά κώδικα θα αλλάξει το χρώμα του φόντου σε όλες τις παραγράφους σε κίτρινο:

```
$("p").css("background-color","yellow");
```

5.4.3.3 jQuery Event Μέθοδοι

Η διαχείριση των events είναι μια από τις κύριες λειτουργίες της jQuery. Τα event handlers είναι μέθοδοι που καλούνται όταν «συμβεί κάτι» στην html, για παράδειγμα όταν κάνουμε κλικ με το ποντίκι σε κάποιο σημείο της ιστοσελίδας τότε καλείτε και ενεργοποιείται ένα αντίστοιχο event handler. Δείτε το παρακάτω παράδειγμα:

```
$(document).ready(function(){
  $("button").click(function(){
    $("p").hide();
  });
});
```

Στο παραπάνω παράδειγμα μια συνάρτηση καλείτε όταν **πατηθεί** (γίνει κλικ) το κουμπί και η μέθοδος αποκρύπτει όλες τις παραγράφους της σελίδας.

Παρακάτω παραθέτουμε μερικά από τα πιο χρήσιμα event handlers της jQuery.

Πίνακας 1 – jQuery event handlers

Event Method	**Περιγραφή**
$(document).ready(function)	Η μέθοδος **ready()** ελέγχει εάν το document έχει τελειώσει την διαδικασία φόρτωσης.
$(*selector*).click(function)	Ενεργοποιεί μια συνάρτηση όταν γίνει **κλικ** στο επιλεγμένο στοιχείο.
$(*selector*).dblclick(function)	Ενεργοποιεί μια συνάρτηση όταν γίνει **διπλό κλικ** στο επιλεγμένο στοιχείο.
$(*selector*).focus(function)	Ενεργοποιεί μια συνάρτηση όταν γίνει **εστιάσουμε** στο επιλεγμένο στοιχείο.
$(*selector*).mouseover(function)	Ενεργοποιεί μια συνάρτηση όταν **περάσει το ποντίκι πάνω** από το επιλεγμένο στοιχείο.

5.4.3.4 jQuery Effects

Η jQuery υποστηρίζει συναρτήσεις που εκτελούν εφέ κίνησης, εμφάνισης κτλ στα στοιχεία html της ιστοσελίδας μας. Παρακάτω παραθέτουμε μερικές από τις βασικές συναρτήσεις για εφέ:

Πίνακας 2 – jQuery effect functions

Effect Method	**Περιγραφή**
hide()	Μέθοδος για την απόκρυψη ενός στοιχείου.
show()	Μέθοδος για την εμφάνιση ενός στοιχείου.
slideToggle()	Μέθοδος που εναλλάσσει την εμφάνιση και την απόκρυψη ενός στοιχείου.
fadeTo()	Μέθοδος για το ξεθώριασμα ενός στοιχείου.
animate()	Μέθοδος για την δημιουργία animation με ένα στοιχείο.

Για παράδειγμα, το παρακάτω απόσπασμα κώδικα, αποκρύπτει και εμφανίζει μια παράγραφό όταν κάνουμε κλικ στο στοιχείο με id #hide και με id #show αντίστοιχα:

```
$("#hide").click(function(){
  $("p").hide();
});
$("#show").click(function(){
  $("p").show();
});
```

Οι συναρτήσεις hide() και show(), δέχονται ως παράμετρο την ταχύτητα με την οποία εκτελείται η μέθοδος και μια κλήση προς άλλη συνάρτηση μετά την ολοκλήρωση της.

$(selector).hide(speed,callback)

$(selector).show(speed,callback)

Η παράμετρος **speed** μπορεί να έχει την τιμή **slow**, **normal**, **fast** ή έναν αριθμό σε **milliseconds**:

```
$("button").click(function(){
  $("p").hide(1000);
});
```

Στο παραπάνω απόσπασμα όταν κάνουμε κλικ στο κουμπί θα κρυφτούν όλες οι παράγραφοι της ιστοσελίδας και η κίνηση τους θα διαρκέσει 1.000 milliseconds (1 sec).

5.5 Επίλογος

Ολοκληρώνοντας την παρουσίαση της JavaScript του DOM και της jQuery έχει ολοκληρωθεί η παράθεση της τεχνολογίας από την πλευρά του browser. Στο επόμενο κεφάλαιο παρουσιάζεται η γλώσσα προγραμματισμού PHP η οποία εκτελείται στη μεριά του server και είναι ο πυρήνας της εφαρμογής που αναπτύσσεται. Επίσης στο δεύτερο μέρος θα επανέρθουμε στην jQuery για παρουσίαση των plugin που χρησιμοποιήθηκαν.

ΚΕΦΑΛΑΙΟ 6

PHP

6.1 Εισαγωγή

Η PHP είναι μια προκαθορισμένη γλώσσα προγραμματισμού που μπορεί να χρησιμοποιηθεί για τη δημιουργία ιστοσελίδων. Τα αρχικά PHP αντιστοιχούν σε “Personal home page Hypertext Preprocessor”. Η PHP είναι μια γλώσσα προγραμματισμού ανοικτού κώδικα ,που χρησιμοποιείται κυρίως για την ανάπτυξη server-side εφαρμογών και δυναμικού περιεχομένου στον Παγκόσμιο Ιστό.
Ο αρχικός σκοπός της PHP ήταν για να χρησιμοποιηθεί για τη δημιουργία δεσμών ενεργειών (scripts). Πιο πρόσφατα έχει ξεκινήσει η χρήση σαν πιο δομημένη γλώσσα προγραμματισμού, βασισμένη στο πρότυπου του object orientation. Κάτι το οποίο επιτρέπει στην δημιουργία ενός ευρύτερου φάσματος εφαρμογών λογισμικού. Η PHP (Hypertext Preprocessor) είναι μία ευρέως χρησιμοποιούμενη, ανοιχτού κώδικα και γενικού σκοπού γλώσσα σεναρίου που είναι ειδικά σχεδιασμένη για την ανάπτυξη εφαρμογών διαδικτύου και μπορεί να ενσωματωθεί μέσα σε κώδικα HTML και να εκτελείται κάθε φορά που ο χρήστης επισκέπτεται την σελίδα. Ο PHP κώδικας μεταφράζεται στον Web διακομιστή και δημιουργεί κώδικα HTML ή άλλη έξοδο που θα δει ο επισκέπτης.

Αυτό που διαχωρίζει την PHP από τα client-side JavaScripts είναι ότι ο κώδικας εκτελείται στον server (εξυπηρετητή). Αν υπήρχε ένα script PHP, ο browser θα έπαιρνε τα αποτελέσματα της εκτέλεσης αυτού του script, χωρίς να μπορεί να καταλάβει με κανένα τρόπο τι κώδικας υπάρχει από κάτω.
Μπορούμε ακόμα να ρυθμίσουμε τον Web Server ώστε να χειρίζεται όλα τα HTML αρχεία με την PHP. Αν και η ανάπτυξη της PHP εστιάζεται σε server-side scripting (scripting στην πλευρά του διακομιστή), μπορούν να γίνουν πολύ περισσότερα με αυτήν.

6.2 Βασικά χαρακτηριστικά

Η PHP ενσωματώνει την ισχύ και τη δυναμικότητα σχετικά παλαιότερων γλωσσών όπως η Perl αλλά καταργώντας τις αδυναμίες τους. Αναφέρουμε μερικά από τα βασικά χαρακτηριστικά της:

- . Ο συντακτικός αναλυτής της, καθώς και ο πηγαίος κώδικάς της διανέμεται ελεύθερα στο διαδίκτυο δίνοντας την δυνατότητα σε όποιον θέλει να κατασκευάζει και να διανέμει εφαρμογές για εμπορική και μη χρήση.
- Μπορεί να μεταφραστεί και να τρέξει στα περισσότερα λειτουργικά συστήματα που κυκλοφορούν στην αγορά (Microsoft Windows, Linux, BSD, Solaris, Macintosh OS X, και UNIX servers).
- Συνεργάζεται χωρίς προβλήματα με τους πιο δημοφιλείς Web Servers που κυκλοφορούν όπως τον Apache και τον Microsoft IIS.
- Διαθέτει ενσωματωμένες εντολές υποστήριξης για ένα μεγάλο αριθμό βάσεων δεδομένων όπωςMySQL, Sybase, Oracle, Ingres. Προσφέρει ένα σύνολο από Database API's τις ενοποιημένες ODBC συναρτήσεις (unified ODBC functions), που εξασφαλίζουν την προσπέλαση σε μια υποκείμενη βάση δεδομένων, χρησιμοποιώντας τις εγγενείς μεθόδους της εκάστοτε βάσης για να μεγιστοποιήσουν την απόδοση (IBM DB2).
- είναι πιο απλό να συντάξει κάποιος κώδικα PHP από ότι σε οποιαδήποτε άλλη γλώσσα σεναρίου.
- Μπορεί να χρησιμοποιηθεί στη δημιουργία εικόνων, ανάγνωση / εγγραφή σε αρχεία και για αποστολή email. Για να προσφέρει αυτές τις υπηρεσίες, η PHP επικοινωνεί με αρκετά πρωτόκολλα όπως: HTTP (Ιστοσελίδες), POP3 (e-mail), SNMP και LDAP.
- Υποστηρίζει τόσο τον διαδικαστικό προγραμματισμό όσο και τον αντικειμενοστραφή.

6.3 Αρχιτεκτονική Βάσης Δεδομένων με PHP – MySQL

Η βασική λειτουργία ενός Web server αποτελείται από δύο αντικείμενα από τα οποία το ένα είναι ο Web browser και το άλλο ο Web server. Απαιτείται μεταξύ τους μία σύνδεση επικοινωνίας. Ένας browser κάνει μία αίτηση στον server κι έπειτα ο server στέλνει πίσω μία απόκριση. Αυτή η αρχιτεκτονική εξυπηρετεί όταν ο διακομιστής παρέχει στατικές σελίδες.

Σημειώνεται ότι οι διακομιστές είναι τα μηχανήματα που προσφέρουν υπηρεσίες ενώ οι πελάτες είναι τα μηχανήματα που ζητούν και δέχονται τις υπηρεσίες αυτές. Ένα μηχάνημα μπορεί να είναι οποιουδήποτε τύπου, ακόμα και των δύο τύπων ταυτόχρονα.

Επίσης, η γλώσσα μορφοποίησης που χρησιμοποιείται για τη δημιουργία ιστοσελίδων, είναι η HTML (Hypertext Markup Language) και το πρωτόκολλο το οποίο χρησιμοποιείται για την μεταφορά των σελίδων από τον διακομιστή στον πελάτη είναι το HTTP (Hypertext Transfer Protocol).

Η αρχιτεκτονική που υποστηρίζει μία Web τοποθεσία με βάση δεδομένων είναι λίγο πιο περίπλοκη.
Μία τυπική Web συναλλαγή βάσεων δεδομένων αποτελείται από τις παρακάτω φάσεις:

- Ο browser ενός χρήστη κάνει μία HTTP αίτηση για μία συγκεκριμένη σελίδα.
- Ο server λαμβάνει την αίτηση για την συγκεκριμένη σελίδα, ανακαλεί το αρχείο και το περνά στην μηχανή PHP για επεξεργασία.
- Η PHP μηχανή αρχίζει την ανάλυση του script. Μέσα στον κώδικα, υπάρχει μία εντολή που κάνει την σύνδεση με την βάση δεδομένων και εκτελεί ένα ερώτημα. Η PHP ανοίγει μία σύνδεση με τον MySQL διακομιστή και στέλνει το κατάλληλο ερώτημα.
- Ο MySQL διακομιστής λαμβάνει το ερώτημα της βάσης δεδομένων, το επεξεργάζεται και στέλνει τα αποτελέσματα ξανά στην PHP μηχανή.
- Η PHP μηχανή σταματά την εκτέλεση του script, που συνήθως περιλαμβάνει την μορφοποίηση των αποτελεσμάτων του ερωτήματος σε HTML. Μετά, επιστρέφει την τελική HTML σελίδα στον server
- Ο server περνά την HTML σελίδα ξανά στον browser, όπου ο χρήστης μπορεί να δει τα αποτελέσματα.

Η διαδικασία είναι βασικά η ίδια, ανεξάρτητα από το ποια μηχανή script ή ποιος server βάσης δεδομένων χρησιμοποιείται. Συνήθως το πρόγραμμα του server, η PHP μηχανή και ο server της βάσης δεδομένων βρίσκονται στον ίδιο υπολογιστή. Ωστόσο, είναι πολύ συνηθισμένο ο server της βάσης δεδομένων να βρίσκεται σε διαφορετικό υπολογιστή. Αυτό μπορεί να γίνει για λόγους ασφάλειας, για μεγαλύτερη χωρητικότητα ή για κατανομή του φόρτου.

6.4 Γιατί PHP;

Παρακάτω είναι μια λίστα με τους λόγους που με έκαναν να επιλέξω την PHP ως την κύρια γλώσσα που χρησιμοποιείται για την κατασκευή του συστήματός μου:

- Η PHP είναι δωρεάν, δεν χρειάζεται τέλη αδειοδότησης ,τέλη υποστήριξης, τέλη συντήρησης, τέλη αναβάθμισης, ή άλλου είδους επιβάρυνση.
- Η PHP είναι cross-platform. Μπορεί να χρησιμοποιηθεί σε υπολογιστές web server που τρέχουν Windows, Mac OS X, Linux, Solaris, και πολλές άλλες μορφές του Unix.
- Η PHP χρησιμοποιείται ευρέως. Τον Μάρτιο του 2010, η PHP ήταν εγκατεστημένη σε περισσότερες από 15 εκατομμύρια ιστοσελίδες.
- Η PHP κρύβει την πολυπλοκότητα της. Μπορούμε να οικοδομήσουμε ισχυρές ιστοσελίδες με PHP οι οποίες διαχειρίζονται εκατομμύρια χρήστες.
- Η PHP είναι δημιουργημένη για τον προγραμματισμό ιστοσελίδων. Αντίθετα από τις περισσότερες άλλες γλώσσες προγραμματισμού, η PHP δημιουργήθηκε από το μηδέν για τη δημιουργία ιστοσελίδων. Αυτό σημαίνει ότι οι κοινές εργασίες προγραμματισμού web, όπως μιλώντας σε μια βάση δεδομένων, είναι συχνά ευκολότερο σε PHP. Η PHP έρχεται με τη συμβατότητα σε μορφή HTML και μπορεί να χρησιμοποιηθεί αλληλένδετη με την HTML.

6.5 Apache Web Server

Ο Apache Web Server είναι ένας πολύ δημοφιλής διακομιστής διαδικτύου που διανέμεται ελεύθερα στο διαδίκτυο. Αναπτύχθηκε και συντηρείται από μια ομάδα εθελοντών που ήθελαν να υλοποιήσουν έναν εύρωστο κώδικα για διακομιστή δικτύου, που να είναι εμπορικός και να έχει πολλά χαρακτηριστικά. Σήμερα ο Apache θεωρείται από τους πιο σταθερούς διακομιστές δικτύου που κυκλοφορούν και θα πρέπει να τονίσουμε ότι αρκετοί εμπορικοί διακομιστές διαδικτύου, όπως ο HTTP Server της IBM, χρησιμοποιούν τον πυρήνα του Apache.

6.6. HTTP

Το **Πρωτόκολλο Μεταφοράς Υπερκειμένου** (HyperText Transfer Protocol, HTTP) είναι η κύρια μέθοδος που χρησιμοποιούν τα πρωτόκολλα του Παγκοσμίου

Ιστού για να μεταφέρουν δεδομένα ανάμεσα σε έναν διακομιστή (server) και ένα πελάτη (client).
Η ανάπτυξη του HTTP έγινε υπό την εποπτεία του World Wide Web Consortium και του Internet Engineering Task Force (IETF).
Το HTTP είναι ο συνήθης για τη διεκπεραίωση αιτήσεων/απαντήσεων μεταξύ ενός υπολογιστή πελάτη (client) και ενός εξυπηρετητή (server). Πελάτης ονομάζεται ο τελικός χρήστης (που αλληλεπιδρά μέσω του φυλλομετρητή του), και ο εξυπηρέτητης είναι η εκάστοτε ιστοσελίδα.

6.7 Επίλογος

Ολοκληρώνοντας με την παρουσίαση και της γλώσσας προγραμματισμού που θα δουλέψουμε για την κατασκευή της εφαρμογής έχουμε ολοκληρώσει το μεγαλύτερο μέρος των τεχνολογιών που θα χρησιμοποιήσουμε. Για να μπούμε όμως στη φάση της υλοποίησης θα πρέπει να αναλυθεί και το σύστημα το οποίο θα χρησιμοποιήσουμε για την κατασκευή της, το οποίο είναι το dotCMS.

ΚΕΦΑΛΑΙΟ 7

dotCMS

7.1 Τι είναι το dotCMS

Το dotCMS είναι ένα σύστημα διαχείρισης περιεχομένου που αναπτύχθηκε από τις εταιρείες dotsoft και new media το 2008. Σκοπός τους ήταν να φτιαχτεί ένα online εργαλείο για τη διαχείριση και ανανέωση των περιεχομένων ενός website, εύκολα, γρήγορα και με ασφάλεια, χωρίς να απαιτούνται γνώσεις προγραμματισμού από τον τελικό χρήστη.

Αναπτύχθηκε επίσης με τέτοιο τρόπο ώστε η παραμετροποίησή του να μπορεί να γίνει από έναν απλό developer χωρίς πολλές γνώσεις mysql ή php μέχρι και τον πιο προχωρημένο προγραμματιστή.

7.2 Model – View – Controller (MVC)

Το dotCMS ακολουθεί το MVC πρότυπο σχεδιασμού λογισμικού. Σκοπός του είναι να διαρέσει την εφαρμογή σε τρία κύρια μέρη: μοντέλο (model), προβολή (view) και διαχείριση (controller) καθιστώντας τη διαδικασία τροποποίησης κάθε μέρους ευκολότερη.

Το πρότυπο αυτό, που δημιουργήθηκε αρχικά για να καλύψει εφαρμογές σε προγράμματα πελάτη (client-side), προωθεί το διαχωρισμό του κώδικα σε τρεις καλά διακριτές ενότητες:

- *Model (Μοντέλο):* η απεικόνιση των πληροφοριών τις οποίες διαχειρίζεται η εφαρμογή – διαφορετικού τύπου κάθε φορά, ανάλογα με το πεδίο δραστηριοποίησης (domain). Το μοντέλο είναι στην ουσία το κομμάτι εκείνο που ασχολείται με τη λογική της εφαρμογής (application logic). Η λογική της εφαρμογής είναι αυτή που δίνει νόημα σε ανεπεξέργαστα δεδομένα (π.χ. προσδιορίζει ότι ένα σύνολο αριθμών αποτελεί στην πραγματικότητα τα ποσά εσόδων και εξόδων μιας εταιρίας). Πολλές εφαρμογές χρησιμοποιούν έναν μόνιμο μηχανισμό αποθήκευσης, όπως συστήματα βάσεων δεδομένων ή αρχεία, για την αποθήκευση δεδομένων. Το MVC δεν προσδιορίζει ρητά τέτοιους διακριτούς μηχανισμούς διαχείρισης πόρων, επειδή θεωρείται ότι αποτελούν επί μέρους τμήματα που περικλείονται στο Μοντέλο.

- *View (Όψη)*: το κομμάτι που ασχολείται με το τι είναι ορατό στον τελικό χρήστη και αναλαμβάνει να δημιουργήσει τα στοιχεία διεπαφής (UI) που θα δώσουν στο χρήστη τη δυνατότητα να αλληλεπιδράσει με το Μοντέλο. Τυπικά, στις Web εφαρμογές, αυτά τα στοιχεία περιλαμβάνουν HTML σελίδες που περιέχουν φόρμες ή εκθέτουν πληροφορία.
- *Controller (Ελεγκτής)*: ο κώδικας που αναλαμβάνει να διασυνδέσει τα δύο παραπάνω επίπεδα, καθορίζοντας το ποιες λειτουργίες γίνονται στο επίπεδο του Μοντέλου ως απόκριση σε ενέργειες που προκαλεί ο χρήστης μέσω της διεπαφής. Τυπικά, όταν ένας χρήστης υποβάλλει τα στοιχεία μιας φόρμας στο επίπεδο View, ο Controller μεταφέρει τον έλεγχο εκτέλεσης σε κάποια κατάλληλη μέθοδο κάποιας κλάσης του επιπέδου Model που θα οδηγήσει στη μόνιμη αποθήκευσή τους.

Από τα παραπάνω είναι προφανές ότι ο κύριος διαχωρισμός που επιτυγχάνει το MVC είναι ανάμεσα στο ορατό στον τελικό χρήστη τμήμα της εφαρμογής και σε αυτό που περιέχει τη λογική της εφαρμογής.

7.3 Τα χαρακτηριστικά του dotCMS

- Περιέχει δικό του μηχανισμό διαχείρισης της βάσης δεδομένων του site. Μπορεί να προστεθεί οποιοδήποτε πεδίο στη βάση, σε οποιοδήποτε τμήμα (component) όποτε κριθεί απαραίτητο.
- Όλα τα τμήματα φτιάχνονται και διαμορφώνονται από τον προγραμματιστή και είναι πλήρως επεξεργάσιμα και εύχρηστα
- Είναι αυτόματα localized. Για να προστεθεί μία γλώσσα στην ιστοσελίδα αρκεί να δηλωθεί.
- Διαθέτει ένα πλήρως περιβάλλον διαχείρισης περιεχομένου ώστε ο τελικός χρήστης να μπορεί να διαχειρίζεται εύκολα και γρήγορα το περιεχόμενο της ιστοσελίδας του.
- Χρησιμοποιεί το smarty PHP template engine για την δημιουργία των templates της κάθε σελίδας.
- Τρέχει σε οποιοδήποτε λογισμικό που υπάρχουν εγκατεστημένα η PHP και η MySQL.

Κάποια από τα παραπάνω χαρακτηριστικά θα τα αναπτύξουμε στο δεύτερο μέρος μέσα από την εφαρμογή με ταυτόχρονα παραδείγματα.

7.4 Εκτεταμένη Διαχείριση και Δυνατότητες

- Δημοσιεύει απεριόριστες σελίδες και άρθρα χωρίς κανέναν απολύτως περιορισμό.
- Υπάρχει η δυνατότητα προσθήκης photo galleries, βιβλιοθήκες αρχείων.
- Εύκολη διαχείριση online των PNGs, PDFs, DOCs, XLSs, GIFs και JPEGs με τη βοήθεια του Image library.
- Υπάρχει επιλογή μη εμφάνισης κάποιου άρθρου, προϊόντος ή οποιασδήποτε εγγραφής έτσι ώστε παλιότερες εγγραφές να μη σβήνονται αλλά να κρατώνται σαν αρχείο.
- Ενσωματωμένος επεξεργαστής κειμένου αντίστοιχος του Word (Tiny MCE).
- Εμφάνιση και αισθητική την οποία διαμορφώνει ο χρήστης.
- Διαχείριση των Template (πρότυπα)
- Δυνατότητα προεπισκόπησης καθώς είναι εφικτή η προβολή αυτών των τμημάτων τοης ιστοσελίδας που έχουν δημιουργηθεί πριν παρουσιαστούν online.
- Προσαρμογή του σχεδιασμού των templates στις επιθυμίες του πελάτη, προσθήκη γραφικών, των **λογοτύπων** και των **σλόγκαν**.
- Εύκολη διαχείριση και διαμόρφωση του πρωτοσέλιδου με αναδιάταξη των άρθρων.
- Πολύ εύκολη κατασκευής πολυγλωσσικής ιστοσελίδας (**Multilanguage**)
- Δυνατότητα λήψης αντιγράφου ασφαλείας του site (**back up**)
- Όλα τα URL είναι **SEO Friendly**

- Η κατασκευή ενός site με το dotCMS μπορεί να καλύψει τις ανάγκες μιας απλής εταιρικής παρουσίασης μέχρι το πιο πολύπλοκο eshop ή οποιαδήποτε εφαρμογή μπορεί να ζητηθεί από κάποια επιχείρηση.
- Ένας developer με στοιχειώδης γνώσεις μπορεί να κατασκευάσει ένα site γρήγορα και εύκολα μετά από μια μικρή στοιχειώδη εκπαίδευση κατανόησης του τρόπου λειργουργίας του.
- Το περιβάλλον διαχείρισης είναι ασφαλές και ευέλικτο. Απο τη στιγμή που θα φτιαχτεί το site οποιοσδήποτε χρήστης του περιβάλλοντος διαχείρισης μπορεί να προσθαφαιρέσει οποιαδήποτε πληροφορία θελήσει στην ιστοσελίδα.

7.5 Εγκατάσταση dotCMS

Η εγκατάσταση του dotCMS προϋποθέτει πως η MySQL,ο Apache και η PHP θα πρέπει να έχουν εγκατασταθεί σωστά στον υπολογιστή μας. Αν έστω και ένα από τα συστατικά απουσιάζει δυστυχώς δε μπορούμε να προχωρήσουμε στην εγκατάσταση του dotCMS.

7.5.1 Εγκατάσταση εργαλείων

Έτσι, εγκαθιστούμε μία εφαρμογή όπως ο **WampServer**, η οποία είναι ένας τοπικός server που υποστηρίζει όλα τα παραπάνω και διατίθεται δωρεάν στο διαδίκτυο. Η εγκατάσταση αυτής της εφαρμογής είναι μία πολύ απλή διαδικασία και διαρκεί μόνο μερικά λεπτά.

Το **WAMP** είναι ένα αρκτικόλεξο για:

Windows

Apache HTTP Server

MySQL

PHP

Αφού «κατεβάσουμε» τον server από τη διεύθυνση **http://www.wampserver.com** τρέχουμε το αρχείο για την εγκατάστασή του.

Κατεβάζουμε λοιπόν την τελευταία έκδοση και το εγκαθιστάμε χρησιμοποιώντας τις κανονικές ρυθμίσεις που μας προτείνει (το κλασικό πλέον Next, Next, ..., Finish), και μετά πάμε στις ρυθμίσεις μας.

Για να τρέξει σωστά το dotCMS θα πρέπει να ενεργοποιηθούν κάποια modules του apache και κάποια extentions της PHP.

- Apache mod_rewrite
- php_curl (το χρειαζόμαστε για eshops που χρησιμοποιούν paypal)
- php_openssl (το χρειαζόμαστε για eshops που χρησιμοποιούν paypal)
- php_xsl (για το import/export to xml)

Μετά την ολοκλήρωση εγκατάστασης και των ρυθμίσεων του τοπικού server, πρέπει να εγκαταστήσουμε το dotCMS σε αυτόν. Το μόνο που πρέπει να κάνουμε είναι να δημιουργήσουμε έναν φάκελο με το όνομα του site (mvadeals) μέσα στον φάκελο websites του wamp server και να κάνουμε unzip τα αρχεία.

7.5.2 Η δομή των φακέλων

7.5.2.1 includes

περιέχει όλα τα αρχεία του framework. Περιέχει όλες τις κλάσεις, τα modules και τα plugins που χρησιμοποιούνται.
Στην περίπτωση που θα χρησιμοποιήσουμε δικό μας server μπορούμε να εγκαταστήσουμε αυτόν τον φάκελο εκτός site, σε κάποιο άλλο directory, πχ στο c:/wamp ή στο /var/www και να το προσθέσουμε στο include path του php.ini πχ. include_path = ".;c:\wamp\includes"
Με τον τρόπο αυτό, δεν χρειάζεται να βάζουμε αυτά τα αρχεία σε κάθε νέο site που θέλουμε να δημιουργήσουμε.
Αν ο server δεν είναι δικός μας τότε τον φάκελο τον αφήνουμε μέσα στον κεντρικο φάκελο του site.

7.5.2.1 Components

περιέχει τα components που θέλουμε να χρησιμοποιήσουμε στο site μας. Το κάθε ένα από αυτά το δημιουργεί ο προγραμματιστής/developer της εφαρμογής κάθε φορά σύμφωνα με την ανάλυση του έργου. Εκεί περιγράφεται η βάση δεδομένων, όλα τα πεδία καθώς και οι συσχετίσεις που υπάρχουν μεταξύ των πινάκων (πχ. Προϊόντα με κατηγορίες προϊόντων)

7.5.2.3 configs

υπάρχει το αρχείο config.php που εκεί γράφονται όλες οι απαραίτητες ρυθμίσεις όπως είναι τα στοιχεία σύνδεσης με την βάση, τα στοιχεία για τον mail server και τα στοιχεία σύνδεσης με την τράπεζα

7.5.2.4 resources

περιέχει τις εικόνες τα javascript και τα stylesheets (css) που καλούνται από τα templates

7.5.2.5 views

περιέχει όλα τα templates. Τα templates είναι γραμμένα σε html και χρησιμοποιούν το smarty PHP template Engine για να φέρουν τα δεδομένα.

7.5.2.6 locales

περιέχει τα αρχεία της κάθε γλώσσας. Το κάθε αρχείο έχει σαν όνομα το όνομα της γλώσσας (gr.php, en.php, de.php) και περιέχει τις μεταφράσεις όσων μεταβλητών χρησιμοποιούμε στα template και δεν μεταφράζονται στη βάση, όπως για παράδειγμα ένα κουμπί που στα ελληνικά γράφει “επιστροφή” στα αγγλικά γράφει “back”.

Όλες οι μεταβλητές αυτές είναι της μορφής

$l['ονομα μεταβλητής'] = 'κείμενο'

πχ. gr.php $l['availability'] = 'διαθεσιμότητα';

en.php $l['availability'] = 'availability';

7.5.2.7 uploads

Ο φάκελος αυτός είναι συνδεδεμένος με τον file manager που χρησιμοποιείτε και εκεί αποθηκεύονται όλα τα αρχεία που ανεβάζει ο χρήστης.

Τα αρχεία που χρησιμοποιούνται στους φακέλους components, locales, configs είναι php, σημαίνει ότι δεν μπορούμε να γράψουμε ελεύθερα php κώδικα αν αυτός είναι απαραίτητος.

7.6 Επίλογος

Τώρα που γνωρίσαμε τη δομή του dotCMS έχει ολοκληρωθεί το πρώτο μέρος της εργασίας αυτής. Στο δεύτερο μέρος θα ξεκινήσουμε με την μεθοδολογία που θα ακολουθήσουμε για την ανάπτυξη του έργου, θα περιγράψουμε τη βάση δεδομένων και την λειτουργία της εφαρμογής.

ΜΕΡΟΣ ΔΕΥΤΕΡΟ

ΚΕΦΑΛΑΙΟ 8

ΑΝΑΛΥΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

8.1 Η σημαντικότητα της ανάλυσης

Για να είναι επιτυχημένο ένα έργο, το πιο σημαντικό στάδιο είναι το στάδιο της ανάλυσης. Η ανάλυση είναι αυτή που θα καθορίσει τι components θα χρησιμοποιηθούν, τι πεδία θα περιέχει η βάση, ποιες θα είναι οι επιλογές που θα έχει η ιστοσελίδα, πως θα λειτουργεί η ιστοσελίδα.

Όσο πιο σωστή και ολοκληρωμένη είναι η ανάλυση τόσο λιγότερα λάθη θα γίνουν στο έργο. Είναι ο μόνος τρόπος για να ξέρει ο πελάτης τι είναι αυτό που θα του παραδοθεί και πως ακριβώς θα δουλεύει έτσι ώστε να αποφευχθούν λάθη παρανόησης των απαιτήσεων της κάθε εφαρμογής. Άλλο να θέλει δηλαδή ο πελάτης και άλλο να φτιάχνεται. Αυτό είναι συνηθισμένο γιατί ο πελάτης πολλές φορές δεν ξέρει τι είναι ακριβώς αυτό που θέλει. Θέλει να κάνει διάφορα αλλά δεν ξέρει τι ακριβώς. Ο πελάτης δεν έχει τις γνώσεις, ακούει διάφορες λέξεις από τον περίγυρό του χωρίς να τις κατανοεί.

Εκεί έρχεται ο αναλυτής / project manager. Θα πρέπει να βοηθήσει αρχικά τον πελάτη και να τον καθοδηγήσει ώστε να κατανοήσει τί είναι αυτό που θέλει από την εφαρμογή και πως θα πρέπει τελικά να λειτουργεί.

Από τη στιγμή που ο project manager ολοκληρώσει την ανάλυση, θα πρέπει να καταγράψει όλες τις προδιαγραφές και να τις δώσει στον developer που καλείτε να φτιάξει το έργο.

8.2 Τι είναι το MVA Deals

Το MVADeals είναι μία ιστοσελίδα προσφορών (deal site). Ο στόχος της ιστοσελίδας είναι μέσα από τις κατάλληλες συνεργασίες να εξασφαλίσει εκπτώσεις σε προϊόντα και υπηρεσίες που φτάνουν μέχρι 90%. Οι προσφορές γίνονται από τις επιχειρήσεις και ισχύουν για συγκεκριμένο αριθμό παραγγελιών ή για συγκεκριμένες μόνο ημερομηνίες.

8.3 Προδιαγραφές

Η εφαρμογή έχει τρεις κατηγορίες χρηστών.

- Τους απλούς χρήστες,
- τους εγγεγραμμένους χρήστες – πελάτες (customers)
- και τον χρήστη διαχειριστή (administrator).

Ο απλός επισκέπτης του site, θα μπορεί να πλοηγηθεί σε όλο το site, να δει όλες τις πληροφορίες που αφορούν την ιστοσελίδα και τις προσφορές που παρουσιάζει. Θα μπορεί να βάζει προϊόντα στο καλάθι αγορών αλλά για να συνεχίσει στην αγορά θα πρέπει να είναι εγγεγραμμένος χρήστης.

Ένας εγγεγραμμένος χρήστης, εκτός από τη δυνατότητα αγοράς των προσφορών, του παρέχεται η δυνατότητα αναδρομής στις αγορές που έχει κάνει και στα κουπόνια που έχουν εκδοθεί.

Αντίθετα από τους παραπάνω χρήστες, ο χρήστης administrator έχει πλήρη πρόσβαση στο περιβάλλον διαχείρισης. Από το σύστημα διαχείρισης μπορεί να προσθέτει προϊόντα, προσφορές, να διαχειρίζεται το περιεχόμενο όλης της ιστοσελίδας, να παρακολουθεί τις παραγγελίες που έχουν γίνει από τους πελάτες, τα κουπόνια που έχουν εκδοθεί, να φτιάχνει και να στέλνει newsletter.

Εκτός από τους χρήστες, η ιστοσελίδα θα περιέχει διάφορα κείμενα, προϊόντα – προσφορές, συνεργαζόμενες εταιρείες, φόρμα επικοινωνίας, newsletter.

Τέλος, οι προσφορές αφορούν συγκεκριμένες πόλεις της Ελλάδος γιατί η επιχείρηση που δίνει την προσφορά βρίσκετε σε συγκεκριμένη πόλη. Αυτό φυσικά δεν απαγορεύει κάποιον να το αγοράσει, η πόλη υπάρχει για διαχωρισμό των προσφορών στην εμφάνιση.

8.4 Σχεδίαση του layout

Από το στάδιο της ανάλυσης έχοντας καταλήξει στις προδιαγραφές της ιστοσελίδας θα πρέπει να σχεδιαστεί το layout και να εγκριθεί από τον πελάτη. Το στάδιο της εικαστικής προβολής μιας ιστοσελίδας είναι πολύ σημαντικό για το τελικό αποτέλεσμα.

Το εικαστικό θα πρέπει να καλύπτει όλες τις προδιαγραφές που αναφέρθηκαν στον κεφάλαιο 2 και θα πρέπει επίσης να οριστούν και να σχεδιαστούν οι αλληλεπιδράσεις που θα έχει η ιστοσελίδα με τον χρήστη. Τι είδους μενού επιλογών τα επιλεγεί, ποια θα είναι τα στοιχεία που θα συμπληρώνει ο επισκέπτης κατά της εγγραφή του κ.ο.κ. Κάποια στοιχεία από αυτά θα πρέπει να σχεδιαστούν σωστά γιατί η μετέπειτα αλλαγή τους θα είναι δύσκολη.
Για την υλοποίηση της συγκεκριμένης εργασίας χρησιμοποιήθηκε το πρόγραμμα Photoshop της Adobe.
Σχεδιάστηκε αρχικά το λογότυπο

Εικόνα 3 - Λογότυπο εφαρμογής

Με βάση το λογότυπο, επιλέχτηκαν και τα χρώματα που θα χρησιμοποιηθούν και έπειτα σχεδιάστηκαν όλες οι σελίδες που θα βλέπει ο επισκέπτης.

8.5 Λειτουργίες της εφαρμογής

Παρακάτω παρατίθεται η λίστα με της κυριότερες διεπαφές της εφαρμογής
Σταθερά στοιχεία σε όλες τις σελίδες

- Header
- Καλάθι αγορών
- Επιλογή πόλης
- Σημερινές προσφορές
- Προηγούμενες προσφορές
- Άλλες προσφορές
- Εγγραφή στο newsletter
- Footer
- Λογότυπα από διάφορα social media

Επιλογές στο header πριν τη σύνδεση ή εγγραφή

- Είσοδος
- Εγγραφείτε!

Επιλογές μετά τη σύνδεση

- Αποσύνδεση
- Μενού χρήστη

Αρχική σελίδα

Στην αρχική σελίδα θα υπάρχει το κεντρικό μενού επιλογών και από κάτω θα υπάρχει ένα κυλιόμενο carousel με όλες τις τρέχουσες προσφορές.
Κάτω από αυτά θα εμφανίζεται αρχικά το κείμενο της πρώτης προσφοράς, Τα στοιχεία της προσφορά, η τιμή, η έκπτωση και το ρολόι που θα μετράει την ώρα μέχρι τη λήξη της προσφοράς.
Δεξιά θα εμφανίζεται λίστα με τις υπόλοιπες προσφορές που υπάρχουν.

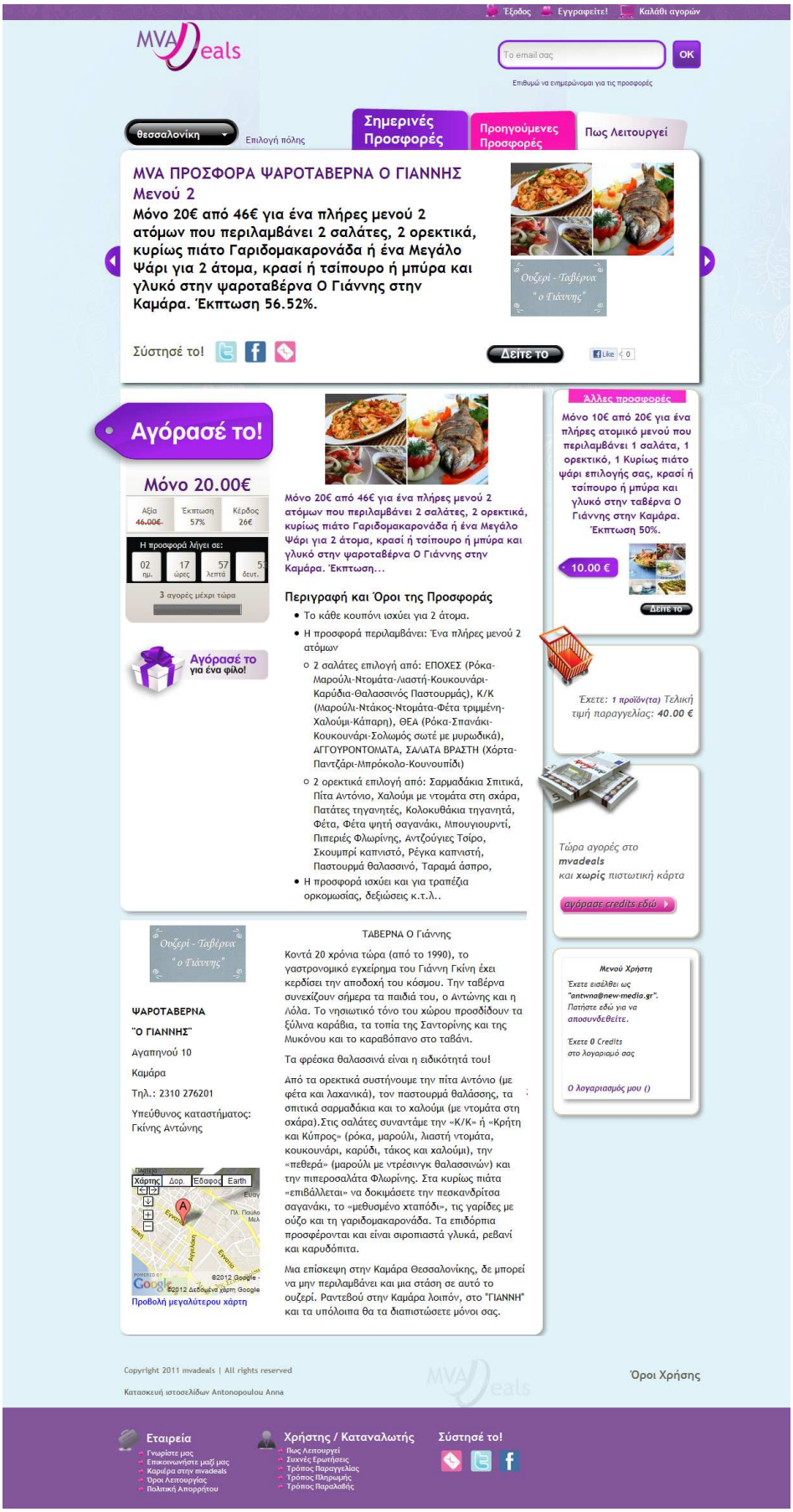
Στην επιλογή Προηγούμενες προσφορές, δεν θα υπάρχει η δεξιά στήλη και θα γεμίζει η σελίδα με τις προσφορές που έχουν λήξη.

Αντίστοιχη συμπεριφορά, χωρίς δηλαδή δεξιά στήλη θα έχουν όλες οι σελίδες κειμένου.

Καλάθι αγορών

Θα εμφανίζει έναν πίνακα με όλες τις προσφορές που έχει βάλει στο καλάθι του για αγορά ο επισκέπτης και για να ολοκληρώσει τις αγορές θα πρέπει να πατήσει την αντίστοιχη επιλογή.

Έπειτα από τον σχεδιασμό, το layout που προέκυψε είναι αυτό που φαίνεται στο σχήμα V



Εικόνα 5 – Η αρχική σελίδα του site

ΚΕΦΑΛΑΙΟ 9

ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

Στο κεφάλαιο αυτό θα περιγράψουμε όλα τα component που θα πρέπει να φτιάξουμε και από αυτά θα προκύψει και το σχήμα της βάσης.
Όπως είπαμε και στο κεφάλαιο 7 που αναφερθήκαμε στο dotCMS, για να κατασκευάσουμε τη βάση χρειάζεται μόνο να περιγράψουμε τα components με συγκεκριμένες εντολές, και το dotCMS θα δημιουργήσει μόνο του τη βάση δεδομένων.

9.1 Components και ρύθμιση στηλών

9.1.1 Components

Το αρχείο που καθορίζει τον τύπο των κάθε πεδίων στη βάση είναι το system/db.php. Επίσης ταυτόχρονα με τη δημιουργία της βάσης φτιάχνεται αυτόματα και το admin με βασική συνάρτηση την formview η οποία μετατρέπει τα δηλωμένα πεδία στα components σε πεδία φόρμας. Ταυτόχρονα με τα πεδία που θα έχει ο κάθε πίνακας, θα παρατίθενται και κομμάτια κώδικα php για το πώς δηλώνονται στο αντίστοιχο component και την αντίστοιχη sql για το table create.

Όλα τα components έχουνε κοινά τα βασικά configuration options τους. Κάθε component διαφορετικού είδους έχει επιπλέον options που πρέπει να τα δείτε στα επιμέρους components.

```
$componentCoreName = 'component_name';

$columns[$componentCoreName][] = array(
    ...
);

$components[$componentCoreName] = array(
    'uses'          => 'module_type',
    'title'         => 'Title for admin panel',
    'url'   => '/xxx/%id',
```

```
    'name'        => $componentCoreName,
    'columns'     => $columns[$componentCoreName],
    'table'       => $componentCoreName,
    'restrictions'        => array(
        'all' => 'admin',
        'new'     => 'admin',
        'edit'      => 'admin',
        'delete' =>'admin'
    )
);
```

componentCoreName

Το όνομα του component με το οποιο θα αναφερόμαστε σε αυτό αργότερα. Πρέπει να είναι με λατινικούς χαρακτήρες, χωρίς κενά η άλλους χαρακτήρες.

uses

Το είδος του component που θα χρησιμοποιήσουμε (pages, categories, contact, newsletter, etc).

title

Το όνομα με το οποιο θα εμφανίζεται στο admin panel. Πρέπει να είναι localized.

url

Χρησιμοποιείται στο helper για search για να δημιουργεί το url στα results

table

Το όνομα του table που θα δημιουργηθεί στην βάση για αυτό το component, αν δεν υπάρχει λόγος καλύτερα να μείνει το ίδιο με το componentName.

restrictions

Ποια groups θα έχουνε δικαιώματα για view, add, edit και delete στο admin. *all* - αν θα μπορεί ο χρηστης να δει το grid στο admin *add* / *edit* / *delete* - η πρόσβαση του χρηστη σε αυτά τα actions.

adminMenu

Το κάνουμε false αν θέλουμε να μην εμφανίζεται το component στο menu του admin.

admingroup

Μπορούμε να εμφανίζουμε τα components ομαδοποιημένα στο menu του admin. Στο admingroup δηλώνουμε το όνομα του group μέσα στο οποίο θα εμφανίζεται το component . πχ για να βάλουμε το component ‘News categories’ σε ενα group που θα λέγεται ‘News’ γραφουμε στο config του news categories:

‘admingroup’ => ‘News’

groupOrder

Μπορούμε να αλλάξουμε τη σειρά των groups στο menu, δίνοντας συγκεκριμένα τη θέση που θα έχουν (αριθμητικά) πχ ‘groupOrder’ => 1. Το groupOrder το χρησιμοποιώ και για να ορισω θεση σε components που δεν ανηκουν *σε κανενα group*.

Αν σε ένα group ανήκουν πολλά components δεν χρειάζεται να το δηλώσουμε σε όλα, αρκεί σε ένα. Επίσης δεν χρειάζεται να βάλουμε groupOrder σε όλα τα groups, μόνο σε όσα χρειάζεται. Πχ αν θέλω ενα συγκεκριμένο group να εμφανιζεται 1ο και τα υπολοιπα να μεινουν οπως ειναι απλα βάζω ‘groupOrder’ => 1 στο συγκεκριμένο.

componentOrder

Μπορούμε επίσης να ορίσουμε τη θέση των components *μέσα στο ιδιο group*. Και πάλι δεν ειναι απαραίτητο να το δηλώσουμε για όλα τα components του group.

9.1.2 Τύποι module

9.1.2.1 Pages

Κάθε website αποτελείται από διαφορετικούς τύπους δυναμικών σελίδων που παρουσιάζουν περιεχόμενο με συγκεκριμένα χαρακτηριστικά. Για παράδειγμα ένα

απλό εταιρικό site έχει κάποιες σελίδες που παρουσιάζουν τα νέα της και κάποιες άλλες που παρουσιάζουν τα έργα της με διαφορετικά χαρακτηριστικά. Κατά την ανάλυση απαιτήσεων του site καθορίζεται ποιοι διαφορετικοί τύποι περιεχομένου-σελίδες θα υπάρχουν στο site και τα χαρακτηριστικά-metadata που τους περιγράφουν. Υπάρχουν κάποιοι προκαθορισμένοι τύποι περιεχομένου που περιγράφονται παρακάτω, αλλά ο developer έχει τη δυνατότητα να τροποποιήσει τα metadata τους ή να δημιουργήσει ένα νέο τύπο με βάση τις ανάγκες του site. Τέλος υπάρχει η δυνατότητα να συσχετισθεί ένα page με ένα ή περισσότερα page του ίδιου ή διαφορετικού τύπου. Έτσι για παράδειγμα μπορούμε να έχουμε κατασκευαστικές εταιρίες που συνδέονται με τα έργα τους ή άρθρα που σχετίζονται μεταξύ τους.

Παραδείγματα χρήσης pages.

Νέα, Blog posts, Αρχεία, Gallery φωτογραφιών, Συνδέσεις, Διαφημίσεις κτλ.

Γενικά είναι για “αντικείμενα” που δεν έχουνε ιεραρχία από μονα τους (δεν ανήκει το ένα σε κάποιο άλλο του ιδιου είδους).

Configuration options

Τα pages δεν έχουνε αλλα options εκτος από τα κοινά όλων των components.

9.1.2.2 Categories

Παραδείγματα χρήσης categories.

Κατηγοριες, Βιβλίο, Συχνές ερωτήσεις, Μενού πλοήγησης, κτλ

Η διαφορα των categories από τα pages είναι ότι τα categories έχουνε ιεραρχία, η μια εγγραφη ανήκει (είναι παιδί) σε μια άλλη του ιδιου component.

Configuration options

adminRender (*grid*, *categories*)
Ο τρόπος που θέλουμε να εμφανίζεται όταν ο χρηστης επιλεγει το component από το menu.

9.1.2.3 Contact

Παραδείγματα χρήσης contact.

Χρησιμοποιείται για την δημιουργία φορμών επικοινωνίας. Αναλαμβάνει να στείλει email σε ένα προκαθορισμένο email και επίσης καταχωρεί τα στοιχεια στην βάση ώστε να μπορεί ο χρηστης να τα δει από το admin.

Configuration options.

email
Η διεύθυνση email που θέλουμε να πηγαίνουνε τα emails.

Μπορεί να είναι ενα ή περισσότερα email addresses χωρισμένα μεταξύ τους με semicolon.

from
Το όνομα που θέλουμε να εμφανίζεται ως αποστολέας του email. Μπορεί να είναι email address ή το όνομα ενός column σε braces πχ {email}

subject
Το θέμα του email που θα στέλνουμε. Μπορούμε να χρησιμοποιήσουμε επίσης ονόματα από columns στο subject, πχ

'subject' => 'Ο χρήστης {name} {surname} σας προτείνει μια σελίδα στο site…'

after
Το relative στο BASEURL url που θέλουμε να πηγαίνει ο χρηστης αφού έχει αποσταλεί το email.

emailSender (*true*, *false*)

Αν θέλουμε ένα αντίγραφο του email να αποσταλεί στο email του χρηστη που συμπλήρωσε τη φόρμα. Πρέπει να υπάρχει πεδίο email στη φορμα.

body

Το template που θέλουμε να εμφανίζεται στο email που θα σταλεί. Μπορεί να είναι το όνομα ενός tpl αρχείου ή smarty code κατευθείαν.

9.1.3 Columns

Τα columns είναι τα πεδία που θα δημιουργηθούν μέσα στον πίνακα του component στη βάση δεδομένων.

Παράδειγμα column:

```
$componentCoreName = 'links';

$columns[$componentCoreName][] = array(

'name'          => 'title',

'title'         => $l['modules']['columns']['question'],

'formControl'   => 'textbox',

'validation'    => array('required'),

'localized'     => true

);
```

ΤΥΠΟΙ METADATA

Κείμενο: Απλό textbox που δέχεται κείμενο και χρησιμοποιείται για τίτλους κλπ

AutoComplete: textbox που καθώς ο χρήστης πληκτρολογεί του κάνει autocomplete με τις τιμές που έχει βάλει στο παρελθόν στο ίδιο πεδίο. Αν θέλει γράφει κάτι τελείως διαφορετικό – δεν περιορίζεται στη λίστα δηλαδή.

Νούμερο: Δέχεται μόνο αριθμητικές μόνο τιμές όπως είναι η χρονιά και η τιμή ενός προϊόντος

Ημερομηνία: Εμφανίζει ημερολογιάκι για την επιλογή της ημερομηνίας. checkbox: για binary τιμές

Editor: Για το βασικό περιεχόμενο μιας σελίδας. Έχουμε 2 εκδόσεις: την basic που έχει μόνο εντολές μορφοποίησης και την advanced που είναι πλήρης editor με δυνατότητα εισαγωγής εικόνων και άλλων media αρχείων. Η επιλογή τύπου και μεγέθους γραμματοσειράς, χρωμάτων, layoyt κλπ γίνεται μέσω προεπιλεγμένων – συμφωνημένων στυλ για να διατηρείται συνέπεια στην εμφάνιση του περιεχόμενου.

Σύνδεση Αρχείων: επιτρέπει τη σύνδεση ενός ή περισσότερων αρχείων με την εγγραφή και καθορισμό της σειράς εμφάνισής τους. Τα αρχεία μπορούν να ελέγχονται για τον τύπο ή το μέγεθος τους

Tree: Αντιστοίχηση της εγγραφής σε μία ή περισσότερες κατηγορίες ιεραρχικά οργανωμένες. Αν είναι επιθυμητό το δέντρο μπορεί να περιοριστεί σε βάθος και να «κλειδωθούν» συγκεκριμένες κατηγορίες. Επιτρέπει reordering των εγγραφών με drag’ n drop.

Combobox/Listbox: Επιλογή μίας ή πολλών τιμών από κάποιες προκαθορισμένες που μπορούν να τροποποιηθούν ή όχι από το διαχειριστή. Π.χ. (νέα/ανακοινώσεις/δελτία τύπου/εκδηλωσεις)

GooglePoint: Αντιστοίχηση της εγγραφής με σημείο σε χάρτη (χ,ψ συντεταγμένες) Positioning: Πεδίο για τον καθορισμό της σειράς εμφάνισης των εγγραφών. Ο καθορισμός του γίνεται με έυκολο τρόπο: drag n drop στο grid των εγγραφών.

LinkToPage: Συσχέτιση με ένα ή περισσότερα page του ίδιου ή διαφορετικού τύπου και καθορισμός προτεραιότητας-σειράς.

Keywors-tags: μπορεί να εισάγει όσα θέλει με ελεύθερο κείμενο ή/και να επιλέξει όσα θέλει από α0υτά που έχουν ήδη χρησιμοποιηθεί

9.2 Περιγραφή των components και της βάσης

Όλοι οι πίνακες έχουν τα πεδία:
Id (auto increment)
Tag (textbox)
Το πεδίο tag το χρησιμοποιούμε πιο πολύ για SEO, για να περνάμε στο url αυτή την τιμή και όχι το id

9.2.1 Πόλεις (cities)

Αρχικά θα πρέπει να δηλώσουμε τις πόλεις που θα εμφανίζονται στο μενού και σε αυτές θα ανήκουν οι συνεργάτες και οι προσφορές. Εκτός από το όνομα της πόλης θα υπάρχει και ένα checkbox (0,1) αν η πόλη αυτή είναι κεντρική ή όχι. Αν είναι κεντρική θα εμφανίζεται στο μενού επιλογής πόλης από τον χρήστη.
Πεδία που πρέπει να έχει:
Τίτλος (textbox)
Central (checkbox)

Αρχείο components/cities.php

```
<?php {
$componentCoreName = 'cities';
$columns[$componentCoreName][] = array(
'name'              => 'title',
'title'             => $l['modules']['columns']['title'],
'formControl'       => 'textbox',
'validation'        => array('required'),
'localized'         => true
);

 $columns[$componentCoreName][] = array(
'name'              => 'tag',
'title'             => $l['modules']['columns']['tag'],
'formControl'       => 'textbox',
'validation'        => array('required','unique'),
'localized'         => false,
'showingrid'        => false
);
```

```
19.  $columns[$componentCoreName][] = array(
20. 'name'            => 'central',
21. 'title'           => $l['modules']['columns']['central'],
22. 'formControl'     => 'checkbox',
23. 'validation'      => array()
24.  );
25.  ?>
26.
```

9.2.2 Περιοχές (states)

Σε κάθε πόλη η συνεργαζόμενη εταιρεία βρίσκεται και σε μία περιοχή ή Δήμο, οπότε θα πρέπει να αντιστοιχίσουμε περιοχές σε πόλεις

Πεδία:

Τίτλος (textbox)

BelongsTo -> Cities

Με το **belongsTo** δείχνουμε την συσχέτιση 1 προς πολλά μεταξύ 2 πινάκων. Μία πόλη δηλαδή έχει πολλές περιοχές. Στη συσχετιζόμενη στήλη μπαίνει το id της εγγραφής του «απέναντι» πίνακα.

Components/states.php

```
1.  <?php {
2.  $componentCoreName = states;
3.  $columns[$componentCoreName][] = array(
4.  'name'            => 'title',
5.  'title'           => $l['modules']['columns']['title'],
6.  'formControl'     => 'textbox',
7.  'validation'      => array('required'),
8.  'localized'       => true
9.  );
10.
11.  $columns[$componentCoreName][] = array(
12. 'name'            => 'tag',
13. 'title'           => $l['modules']['columns']['tag'],
14. 'formControl'     => 'textbox',
15. 'validation'      => array('required','unique'),
16. 'localized'       => false,
17. 'showingrid'      => false
```

```
18. );
19. $columns[$componentCoreName][] = array(
20. 'name'                    => 'city',
21. 'title'                   => $l['modules']['columns']['city'],
22. 'formControl'     => 'relation',
23. 'belongsTo'       => 'cities',
24. order'         => true,
25. 'validation'    => array('required'),
26. 'localized'     => false
27.  );
28.  ?>
```

9.2.3 Συνεργάτες (partners)

Ο πίνακας συνεργάτες αφορά τις συνεργαζόμενες επιχειρήσεις. Εδώ καταγράφονται όλα τα στοιχεία παρουσίασης της επιχείρησης.

Πεδία:

Τίτλος (textbox)

Κωδικός συνεργάτη (textbox)

Περιγραφή (editor)

Λογότυπο (image)

Has -> Cities

Όνομα συνεργάτη (textbox)

Διεύθυνση (editor)

Email (textbox)

URL (textbox)

Google Map script (textarea)

Διάφορες επιπλέον πληροφορίες (textarea)

Με την Has δείχνουμε μία συσχέτιση πολλά προς πολλά μεταξύ 2 πινάκων. Μία επιχείρηση μπορεί να είναι σε πολλές πόλεις και το αντίστροφο. Όταν έχουμε μια τέτοια συσχέτιση τότε δημιουργείτε ένας ενδιάμεσος πίνακας όπου περιέχει σαν πεδία τα 2 id των δυο συσχετιζόμενων πινάκων.

Για να μην παραθέτουμε όλο τον κώδικα από τα components, θα παραθέτω μόνο αυτά που είναι διαφορετικά από κάποιο άλλο.

Components/partners.php

```
1.  <?php {
2.  …
3.  ….
4.  $columns[$componentCoreName][] = array(
5.  'name'                  => $componentCoreName . '_cities',
6.  'title'                 => $l['modules']['columns']['partners_cities'],
7.  'formControl'     => 'relation',
8.  'has'                   => 'cities',
9.  'foreign_table'   => array(
10. 'table'                 => 'cities' . $componentCoreName . '_bonds',
11. 'master_field'    => 'master_id',
12. 'slave_field'     => 'slave_id'
13. ),
14. 'validation'    => array(),
15. 'localized'     => false
16. );
17. …
18. …
19. ?>
```

9.2.4 Καταστήματα (shops)

Κάθε συνεργάτης όμως, πέρα από τα στοιχεία του, μπορεί να έχει ένα ή και περισσότερα υποκαταστήματα, οπότε θα πρέπει να καταχωρηθούν και αυτά σε σχέση τον συνεργάτη.

Πεδία:

Τίτλος (textbox)

Διεύθυνση (textbox)

Ταχυδρομικός κώδικας (textbox)

Τηλέφωνο (textbox)

BelongsTo -> Cities

BelongsTo -> Partners

BelongsTo -> States

9.2.5 Προσφορές (offers)

Κάθε προσφορά έχει τον κεντρικό της τίτλο, την επιχείρηση που κάνει την προσφορά και 2 σετ ημερομηνιών. Το 1^{ο} σετ αφορά την έναρξη και λήξη της προσφοράς και το 2^{ο} σετ την έναρξη και λήξη παραλαβής / εξαργύρωσης κουπονιών.

Πεδία:

Τίτλος (textbox)

Ημερομηνία έναρξης προσφοράς (date)

Ημερομηνία λήξης προσφοράς (date)

Ημερομηνία έναρξης παραλαβής (date)

Ημερομηνία λήξης παραλαβής (date)

BelongsTo -> Partners

Όροι παραλαβής (ediror)

9.2.6 Προϊόντα προσφοράς (offer_products)

Κάθε προσφορά μπορεί να έχει παραπάνω από ένα προϊόντα, επίσης κάθε προϊόν προσφοράς χαρακτηρίζεται από: τον κωδικό του, ένα barcode το οποίο μπορεί να χρησιμοποιηθεί μελλοντικά αν θέλουμε να τυπώνουμε barcodes πάνω στα κουπόνια ώστε να περνάνε από μηχανές αναγνώρισης πχ σε supermarket, την τιμή του, το ποσοστό έκπτωσης, την τιμή με έκπτωση, την διαθεσιμότητα που υπάρχει προς πώληση. Κάποιες προσφορές έχουν περιορισμένο αριθμό προϊόντων τα οποία μπορεί να εξαντληθούν πριν την ημερομηνία λήξης της προσφοράς και θα πρέπει να ελέγχετε αυτή η διαθεσιμότητα, οπότε είναι αναγκαίο ένα checkbox πεδίο για το αν θα ελέγχουμε τη διαθεσιμότητα ή όχι. Επίσης, κάποια προϊόντα έχουν περιορισμό στον αριθμό κουπονιών που μπορεί να αγοράσει ένας χρήστης. Τέλος για καθαρά θέμα μάρκετινγκ είναι αναγκαία η ύπαρξη ενός πεδίου «fake» πωλήσεων έτσι ώστε αν κάποιο προϊόν δεν έχει πολλές αγορές, να μπαίνει εκεί μία τιμή και να εμφανίζει αντί για τις πραγματικές πωλήσεις.

Πεδία:

Κείμενο προσφοράς (editor)

Κωδικός προϊόντος (textbox)

Barcode (textbox)

Τιμή (number)
Τιμή με έκπτωση (number)
Ποσοστό έκπτωσης (number)
Διαθεσιμότητα (textbox)
Περιορισμός ανά χρήστη (textbox)
Check Availability (checkbox)
Πωλήσεις (textbox)
BelongsTo -> Offers

9.2.7 Προϊόντα

Για τον λόγο ότι τα προϊόντα μπορεί να επαναλαμβάνονται αλλά σε διαφορετικές προσφορές και διαφορετικές τιμές κάθε φορά για τον λόγο αυτό χρησιμοποιούμε το προϊόν σαν διαφορετική οντότητα και το συνδέουμε κάθε φορά με τα προϊόντα προσφοράς.

Πεδία:
Τίτλος (textbox)
Περιγραφή (editor)
Εικόνα (image)

9.2.8 Πόλεις προσφοράς

Όπως αναφέραμε και παραπάνω, οι προσφορές παίζουν σε κάποιες πόλεις και σε κάποια καταστήματα. Μπορεί μία επιχείρηση να έχει πολλά καταστήματα, αλλά η συγκεκριμένη προσφορά ισχύει για συγκεκριμένα καταστήματα. Είναι πολύ σημαντικό το κατάστημα για να μπορεί ο ιδιοκτήτης μετά το τέλος της προσφοράς να τροφοδοτήσει τα καταστήματα που πρέπει να τις ποσότητες που πρέπει καθώς επίσης να στείλει και τα αντίστοιχα ονόματα των πελατών. Επίσης αν μια επιχείρηση έχει πολλά καταστήματα καταχωρημένα και θέλει να βάλει την προσφορά σε όλα, για να μην τα επιλέγει όλα ένα-ένα χρησιμοποιούμε ένα checkbox «όλα τα καταστήματα». Αν επίσης η επιχείρηση έχει μόνο ένα σημείο παραλαβής, τότε δίνουμε ένα ελεύθερο πεδίο να καταγραφεί η διεύθυνση αυτή.

Πεδία:
BelongsTo -> Cities
Σε όλα τα υποκαταστήματα (checkbox)

Has -> Shops
Παραλαβή από κατάστημα (textbox)
BelongsTo -> Offers

9.2.9 Credits

Τα credits είναι ένα προϊόν το οποίο κοστίζει 1 ευρώ. Μπορεί κάποιος πελάτης να αγοράσει πχ 100 credits τα οποία κοστίζουν 100 ευρώ και μπορεί να τα χρησιμοποιήσει για να κάνει τις συναλλαγές του με αυτά. Επίσης τα credits μπορεί να δίνονται στους πελάτες είτε σαν δώρο είτε σαν επιστροφή από την MVAdeals.

Πεδία:

Τίτλος (textbox)
Τιμή (number)
Περιγραφή (textbox)

9.2.10 Πελάτες

Στον πίνακα πελάτες έχουμε όλα τα απαραίτητα στοιχεία για τον κάθε πελάτη/χρήστη της ιστοσελίδας μας.

Πεδία:

email -> (textbox)
κωδικός -> (textbox)
Όνομα -> (textbox)
Επώνυμο -> (textbox)
Διεύθυνση -> (textbox)
ΤΚ -> (textbox)
Πόλη -> (textbox)
Χώρα -> (textbox)
Τηλέφωνο -> (textbox)
Has -> Customer_categories *
newsletter -> (checkbox)
Credits **

* Δημιουργούμε ένα πίνακα με κατηγορίες πελατών. Τις κατηγορίες αυτές δεν τις χρησιμοποιούμε προς το παρόν πουθενά. Είναι για μελλοντική χρήση για την

περίπτωση που θέλουμε να δώσουμε πρόσβαση σε κάποιους χρήστες με παραπάνω δικαιώματα. Πχ να δώσουμε κάποια στοιχεία παρακολούθησης στις συνεργαζόμενες εταιρείες

** Στο πεδίο αυτό, όταν κάποιος πελάτης αγοράσει credits, αφού θα ελέγχεται πρώτα από τον διαχειριστή η αγορά, θα προστίθενται τα credits που έχει αγοράσει ο πελάτης σε αυτό το πεδίο και θα μειώνονται αυτόματα με τις αγορές.

9.2.11 Καλάθι αγορών

Το καλάθι αγορών είναι ταυτόχρονα και ο πίνακας των παραγγελιών, οπότε θα πρέπει να κρατάμε τα εξής στοιχεία του χρήστη, user_id, το status της παραγγελίας, αν έχει ολοκληρωθεί ή όχι, τη δημιουργία που δημιουργήθηκε το καλάθι και την ημερομηνία που έγινε submit το καλάθι και την τελική τιμή της παραγγελίας

<u>πεδία</u>

title (textbox)

user_id (belongsTo -> customers)

status (selectone -> status)

date_created (date)

date_submitted (date)

products_price (number)

9.2.12 Προϊόντα καλαθιού

Το κάθε καλάθι, έχει πολλά προϊόντα και γι’ αυτό υπάρχει ο πίνακας με τα προϊόντα καλαθιού, όπου εκεί θα πρέπει να κρατάμε το basket_id φυσικά που είναι το πεδίο που το συνδέει με το basket. Την τιμή μονάδας, την ποσότητα που έχει βάλει στο καλάθι καθώς και την ημερομηνία που το έχει προσθέσει.

<u>Πεδία</u>

basket_id, 'belongsTo' => 'baskets',

price => 'number',

quantity => ‘textbox’

date_added => ‘date’

9.2.13 Κουπόνια

Για κάθε ένα τεμάχιο που μπαίνει στο καλάθι, θα πρέπει να φτιάχνεται και το αντίστοιχο κουπόνι. Τα στοιχεία που θα πρέπει να έχει το κουπόνι είναι τα:

πεδία

customer, ‘belongsTo' => 'customers',

barcode => ‘textbox’

quantity_no = ‘textbox’, ο αριθμός κουπονιού, αν δηλαδή κάποιος αγοράσει 5 κουπόνια από μια προσφορά, σε ποιο κουπόνι αντιστοιχεί η εγγραφή

owner = ‘textbox’, τα στοιχεία του κατόχου του κουπονιού

basket_id, 'belongsTo' => 'baskets', σε ποιο καλάθι ανήκει

basket_items_id, 'belongsTo' => 'basket_items', σε ποιο προϊόν καλαθιού ανήκει

active = ‘checkbox’, αν είναι ενεργό ή όχι.

random_code, ένας τυχαίος αριθμός που τυπώνεται πάνω στο κουπόνι

offer, 'belongsTo' => 'offers', σε ποιο προσφορά ανήκει το προϊόν

9.2.14 Υπόλοιποι πίνακες

Εκτός από τους πίνακες που αφορούν τα προϊόντα υπάρχουν και οι πίνακες που αφορούν τα κείμενα της ιστοσελίδας, τα faq, τα στατικά κείμενα και τις φόρμες επικοινωνίας.

Για τα κείμενα της ιστοσελίδας θα δημιουργήσουμε το component pages το οποίο και θα έχει τα στοιχεία:

Τίτλος, κατηγορία και κείμενο, το FAQ θα έχει τα πεδία Ερώτηση, Απάντηση.

9.3 Localization

Για να γίνει η εφαρμογή πολύγλωσση δηλώνουμε σε κάθε component την τιμή Localized = true ή false ανάλογα με το αν το συγκεκριμένο πεδίο πρέπει να μεταφραστεί ή όχι. Για παράδειγμα ο τίτλος μιας προσφοράς πρέπει να μεταφραστεί, ενώ αντίστοιχα ένα checkbox ή η τιμή δεν χρειάζεται, παραμένει σταθερό άσχετα με τη γλώσσα της ιστοσελίδας.

Με τον τρόπο αυτό, χωρίζουμε το κάθε component σε δυο διαφορετικά tables τα:

«component name»_core και «component name»_locale

Ο «component name»_core περιέχει τα μοναδικά πεδία ενώ αντίστοιχα ο «component name»_locale τα πεδία που πρέπει να μεταφραστούν. Εκτός από τα

πεδία αυτά, για να έχει σύνδεση με τον _core περιέχει το πεδίο που αναφέρετε στο id του καθώς επίσης και το locale_id για τη γλώσσα που αναφέρετε.
Με αυτόν τον τρόπο, η βάση μας είναι κανονικοποιημένη.
Για λόγους ευκολίας και ταχύτητας, ώστε να μην κάνουμε joing τους πίνακες αυτούς την ώρα που ζητάμε δεδομένα, δημιουργούμε ένα view τους περιέχει τη σύνδεσή τους.

9.4 Επίλογος

Αφού ορίσαμε και φτιάξαμε τα components το μένει τώρα να ετοιμάσουμε τη δομή του site (model) και την εμφάνισή του μέσα από τα templates.
Στο *παράρτημα Α* υπάρχει όλος το κώδικας MySQL που προκύπτει κατά την επεξεργασία των component

ΚΕΦΑΛΑΙΟ 10

MODEL και VIEW

10.1 MODEL

Το model.php περιγραφει την δομή του site και βοηθάει στην παραμετροποίηση του μέσω παραμέτρων από το URL.

Ένα κενό model.

```
1.  class model extends modules
2.  {
3.
4.  }
```

Για κάθε ξεχωριστή “σελίδα” στο site χρειάζεται ένα method/function μέσα στο model class.

Για παράδειγμα αν θέλουμε μια “σελίδα” main (http://www.test.gr/main/).

```
1.  class model extends modules
2.  {
3.  public function main()
4.  {
5.  /* main page logic */
6.  }
7.  }
```

Το κάθε function αποτελεί τη λογική της κάθε σελίδας την οποια μπορούμε να αλλάξουμε χρησιμοποιώντας παραμέτρους.

Η παράμετροι δηλώνονται στο url ως συνεχεια του function name.

```
1.  class model extends modules
2.  {
3.  public function main()
4.  {
5.  /* main page logic */
6.  }
7.
8.  public function about($about_what='company')
9.  {
10. /* about page logic */
11. }
12. }
```

Στο παραπάνω παράδειγμα έχουμε δημιουργήσει μια επιπλέον function about, την οποια μπορούμε να δούμε μέσω του URL http://www.test.gr/about/

Το $about_what=’company’ δηλώνει την default τιμή της μεταβλητής όταν στο URL δεν υπάρχει κάτι.

Αν ο χρηστης πληκτρολογήσει το url http://www.test.gr/about/ τότε η τιμή τις μεταβλητής $about_what θα είναι company.

Ενώ στο http://www.test.gr/about/hosting/ η τιμή τις μεταβλητής είναι hosting.

Σε περίπτωση που έχουμε παραπάνω από μια παραμέτρους οι προηγούμενες από αυτήν που θέλουμε είναι υποχρεωτικές.

```
1. public function about($param, $other_param, $third_param)
2. {
3. /* about page logic */
4. }
```

Μερικά παραδείγματα…
http://www.test.gr/param_1_value/param_2_value/param_3_value/

```
1. $param = 'param_1_value';
2. $other_param = 'param_2_value';
3. $third_param = 'param_3_value';
```

http://www.test.gr/param_x_value/

```
1. $param = 'param_x_value';
2. $other_param = '';
3. $third_param = '';
```

Η σελίδα (template) που θα δει ο χρηστης ανάλογα με το url που έχει επιλέξει καθορίζεται από το κάθε method.

Ένα site μπορεί να έχει πολλά templates, χωρισμένα ανάλογα με το site η τον developer ανά κατηγορια, μέρος του site κτλ.

Σε κάθε method του model μπορούμε να εμφανίσουμε ένα η περισσότερα templates ανάλογα με το αποτέλεσμα που θέλουμε και τον τρόπο που δουλεύουμε.
-Τα παραδείγματα παρακάτω είναι απλά για να δείξουνε τον τρόπο λειτουργίας των functions και δεν είναι best practice.-

```
1. class model extends modules
2. {
3. public function main()
4. {
5. append('header.tpl');
6. append('content.tpl');
7. append('footer.tpl');
```

```
8.  }
9.
```

Στο παράδειγμα αυτό η σελίδα που θα δει ο χρηστης στο /main θα εμφανίσει την html από τα templates με τη σειρά που τα καλέσαμε.

```
1.  class model extends modules
2.  {
3.  public function main()
4.  {
5.  assign('page_title', 'Kentriki selida');
6.  append('header.tpl');
7.  }
8.  }
```

Σε περίπτωση που θέλουμε να περάσουμε μια μεταβλητή σε ένα από τα templates χρησιμοποιούμε την function assign.

Στο παραπάνω παράδειγμα δηλώνουμε μια μεταβλητή με όνομα “page_title” και τιμή “Κεντρική σελίδα” για χρήση μέσα στα templates που κάνουμε append. Σε όλα τα templates που καλούμε τώρα μπορούμε να χρησιμοποιήσουμε τι μεταβλητή $page_title.

Στις μεταβλητές που κάνουμε assign μπορούμε να βάλουμε ως τιμή ένα string, array η ακόμα και object.

Μια χρήση του assign είναι να μεταφέρουμε στο template μια παράμετρο του method.

```
20. class model extends modules
21. {
22. public function main($page_name='test')
23. {
24. assign('page_name', $page_name);
25. append('header.tpl');
26. }
27. }
```

10.2 Εμφάνιση δεδομένων

Έχοντας φτιάξει τη δομή του site μέσα στο model αυτό που μένει τέλος να κάνουμε είναι αφού «κόψουμε» το layout στις κατάλληλες εικόνες, να φτιάξουμε το template βάση του οποίου θα εμφανίζεται η ιστοσελίδα μας.

10.2.1 Smarty

Το smarty είναι ένα template engine το οποίο βοηθάει στο διαχωριστεί η λογική ενός site απο την εμφάνισή του. Η λογική του site είναι ουσιαστικά οι συναρτήσεις που γράφουμε στο model ενώ η εμφάνισή του είναι τα templates (αρχεία .tpl) που δημιουργούμε στα views.

Στη συνέχεια περιγράφουμε κάποιες default συναρτήσεις του smarty που θεωρούμε χρήσιμες, καθώς και κάποιες που έχουμε δημιουργήσει μέσα στο framework.

Το documentation του smarty υπάρχει online στη διεύθυνση: http://www.smarty.net/manual/en/

10.2.2 Χρήσιμες smarty συναρτήσεις και μεταβλητές

10.2.2.1 title

Η {title} κάνει την αλλαγή του <title> του site απο οποιοδήποτε μέρος της σελίδας χωρίς να χρειάζεται να ξέρουμε τι θέλουμε να μπει στο title πριν εμφανιστεί το βασικό περιεχόμενο της σελίδας.

Οποιαδήποτε στιγμή θελήσουμε να αλλάξουμε τον τίτλο της σελίδας, ή να προσθέσουμε καποιο κείμενο απλά καλούμε την {title} μέσα απο το template.

Παίρνει δυο παραμέτρους.
insert (**after**, before, over)
text (το κείμενο που θέλουμε να εμφανιστεί)

πχ.
//title: Normal title
{title text=”Just a title” insert=”over”}
//title: Just a title
{title text=” - Home” insert=”after”}
//title: Just a title - Home
{title text=”Welcome to: ” insert=”before”}
//title: Welcome to: Just a title - Home

10.2.2.2 breadcrumb

Η {breadcrumb} χρησιμοποιείται για τη δημιουργία του breadcrumb του site. Μπορούμε να την καλέσουμε απο οποιοδήποτε μέρος της σελίδας για να δημιουργήσουμε ή να αλλάξουμε το breadcrumb.

Θα πρέπει στο βασικό template (αυτό που είναι κοινό για όλες τις σελίδες) να δημιουργήσουμε το

** **

Οποιαδήποτε στιγμή θελήσουμε να αλλάξουμε τον τίτλο της σελίδας, ή να προσθέσουμε καποιο κείμενο απλά καλούμε την {breadcrumb} μέσα απο το template.

Παίρνει δυο παραμέτρους.

text (το κείμενο που θέλουμε να εμφανιστεί)

insert (after, before, over)

Συνήθως βολεύει να βάλουμε το αρχικό τμήμα του breadcrumb στο βασικό template πχ
** Home **

και μετά να προσθέτουμε σε αυτό αυτό που θέλουμε να δείξουμε στην κάθε σελίδα πχ.
{breadcrumb text=”news” insert=”after”}
Εμφανίζει: Home » news

Για να δείξουμε την τιμή μιας μεταβλητής μέσα στο breadcrumb όπως και μέσα σε οποιαδήποτε άλλη συνάρτηση smarty χρησιμοποιούμε “

10.2.2.3 Switch

Η λειτουργία της είναι η ίδια με αυτήν της Php (**php-switch**) .

Παράδειγμα 1:

```
1. {switch $action}
```

```
2. {case 'category'}
3. //show category
4. {case 'product'}
5. //show product
6. {default}
7. //show other
8. {/switch}
```

Παράδειγμα2:

```
1. {foreach item=$debugItem from=$debugData}
2. // Switch on $debugItem.type
3. {switch $debugItem.type}
4. {case 1}
5. {case "invalid_field"}
6. // Case checks for string and numbers.
7. {/case}
8.
9. {case $postError}
10. {case $getError|cat:"_ajax"|lower}
11. // Case checks can also use variables and modifiers.
12. {break}
13.
14. {default}
15. // Default case is supported.
16. {/switch}
17. {/foreach}
```

Developer notes:

Το plugin είναι στο element/plugins/smarty/compiler.switch.php και φορτώνει αυτόματα.

10.2.2.4 Thumb

Δημιουργεί το thumbnail μιας εικόνας.

πχ.

```
1. <img src="{thumb src=$image w=100 h=100 zc=1}" title="image"/>
```

Configurations

src

Το path της εικόνας χωρίς το basedir πχ uploads/images/image.jpg

w

Width σε px

h

Height σε px

zc (0/1)

Ρυθμίζει το αν ο αλγόριθμος του thumbnail θα ζουμάρει την εικόνα ή όχι (χρησιμοποιείται για να εμφανίζονται καλύτερα κάποιες εικόνες που δεν είναι απαραίτητο να φαίνονται ολόκληρες)

10.2.2.5 array

Συνάρτηση για να δημιουργούμε associative arrays

Πχ

```
1. {array key="name" value=$name}
2. {array key="surname" value=$surname}
3. {array key="email" value=$email}
```

Το παραπάνω δημιουργεί ένα array που περιέχει τις τιμές $name, $surname, $email που έχουμε πάρει από τη βάση.

10.2.2.6 krumo

Χρησιμοποιείται για να βλέπουμε nested τα περιεχόμενα ενός array.

πχ.

```
1. {krumo value=$data}
```

10.2.3 Inject

Η πιο βασική από της συναρτήσεις. Είναι η συνάρτηση που χρησιμοποιείται για να κάνουμε ερωτήματα στη βάση.

```
1. {inject module="news" action="one" id=5 assign="news_item_5"}
```

Στο παραπάνω παράδειγμα χρησιμοποιούμε την inject για να πάρουμε το row με id 5 από το component news.

Η inject επιτρέπει να καλούμε methods από τα components που έχουμε δημιουργησει ωστε μετα να χρησιμοποιουμε το αποτελεσμα απο τα methods στο template.
Ανάλογα με το component υπάρχουν διαφορετικά methods που μπορούμε να καλέσουμε.

10.2.3.1 formview

Χρησιμοποιείται όταν θέλουμε να εμφανίσουμε μια φόρμα όπου ο χρήστης θα εισάγει στοιχεία για να τα αποθηκεύσει στη βάση ή να τα στείλει με mail.

Υποτίθεται ότι έχουμε δημιουργήσει το component με όλα τα πεδία της φόρμας. Αν θέλουμε η φόρμα απλά να αποθηκευτεί στη βάση τότε την κάνουμε τύπου **page** ενώ αν θέλουμε να σταλεί και με email την κάνουμε τύπου **contact**.

Καλούμε την formView ως εξής:

```
1. {inject module="contact" action="formView"}
```

Επιπλέον options:
data
Με την παράμετρο data μπορούμε να περάσουμε στη φόρμα κάποια default στοιχεία. Τα στοιχεία αυτά τα δίνουμε ως array πχ.

```
1. {array var="data" key="name" value=$name}
2. {array var="data" key="surname" value=$surname}
3. {array var="data" key="email" value=$email}
```

Δημιουργεί ένα array $data με default τιμές για τα πεδία name, surname, email. Για να τα περάσουμε στη φόρμα:

```
1. {inject module="contact" action="formView" data=$data}
```

idsFromNames (0/1, default:0)
Προσθέτει το attribute id σε κάθε πεδίο της φόρμας, το οποίο το παίρνει από το attribute name.

idsPrepend

Το χρησιμοποιούμε για να προσθέσουμε μια έξτρα λέξη πρίν από κάθε id.

πχ

```
1. {inject module="contact_form" action="formView" idsFromNames="1" idsPrepend="form_"}
```

formName (string / optional)

Το χρησιμοποιούμε για να δώσουμε μια συγκεκριμένη τιμή στο **name** & το **id** της φόρμας.

Αν δεν το χρησιμοποιήσουμε, η φόρμα θα πάρει ως id το όνομα του component & ως name το form_ και το όνομα του component.

10.2.3.2 One

Συνάρτηση που επιστρέφει ένα row από τη βάση δεδομένων.

Παράδειγμα:

```
1. {inject module="news" action="one" id=5 assign="news_item_5"}
```

Στο παραπάνω παράδειγμα παίρνουμε το row με id 5 από το component news και το αποθηκεύουμε στη μεταβλητή news_item_5.

10.2.3.3 all

Συνάρτηση που επιστρέφει τα rows ενός component ανάλογα με τα configurations που κάνουμε.

Απλό παράδειγμα:

```
1. {inject module="projects" action="all" assign="projects"}
```

Επιστρέφει όλα τα rows του component projects και τα αποθηκεύει στη μεταβλητή projects.

10.2.3.4 Επιπλέον options

where

Μπορούμε να βάλουμε κώδικα sql που θα εκτελεστεί στο τμήμα WHERE του query πχ.

```
1. {inject module="projects" action="all" where=" visible = '1'  " assign="projects"}
```

Επιστρέφει όλα τα projects που έχουμε κάνει visible.

```
1. {inject module="news" action="all" where="title LIKE '%announcement%'" assign="news"}
```

Επιστρέφει όλα τα news στα οποία ο τίτλος περιέχει τη λέξη ‘announcement’.

limit

Μπορούμε να επιλέξουμε πόσες εγγραφές θέλουμε να φέρει η inject από τη βάση. πχ

```
1. {inject module="news" action="all" limit="10" assign="news"}
```

Επιστρέφει τις εγγραφές από 0 μέχρι 10

```
1. {inject module="news" action="all" limit="10,20" assign="news"}
```

Επιστρέφει τις εγγραφές από 10 μέχρι 20

pagelimit

Μπορούμε να χωρίσουμε τις εγγραφές που επιστρέφει η all για να εμφανίζονται σε ξεχωριστές σελίδες.

Χρησιμοποιείται σε συνδυασμό με τη συνάρτηση {paginate} που δημιουργεί αυτόματα τη σελιδοποίηση (περιγράφεται στο κεφάλαιο smarty). πχ

```
1. {inject module="news" action="all" pagelimit="10" assign="news"}
```

Επιστρέφει όλες τις εγγραφές χωρισμένες σε 10άδες

groupBy

Μπορούμε να κάνουμε group τις εγγραφές με βάση ένα ή περισσότερα columns. Πχ

```
1. {inject module="products" action="all" groupBy="code" assign="products"}
```

Κάνει group τα προιόντα με βάση τον κωδικό τους. Αν υπάρχουν πολλά προιόντα με τον ίδιο κωδικό επιστρέφεται μόνο το ένα.

orderBy
Επιστρέφει τα results ταξινομημένα με βάση κάποιο column. Μπορούμε να χρησιμοποιήσουμε επιπλέον το orderDir για να ορίσουμε την σειρά ταξινόμησης (αύξουσα/φθίνουσα - ascending/descending), με default ascending. Πχ

```
1. {inject module="news" action="all" orderBy="date" assign="news"}
```

Επιστρέφει όλα τα news ταξινομημένα με βάση την ημερομηνία. Πρώτα τα παλαιότερα και μετά τα πιο πρόσφατα νέα.

orderDir (asc/desc, default: asc)
Πχ

```
1. {inject module="news" action="all" orderBy="date" orderDir="desc" assign="news"}
```

Επιστρέφει όλα τα news ταξινομημένα με βάση την ημερομηνία, από το πιο πρόσφατο μέχρι το πιο παλιο.

10.2.3.5 Children

Η children χρησιμοποιείται μόνο για components τύπου **categories**. Επιστρέφει τα παιδιά ενός συγκεκριμένου node με βάση την ιεραρχία που έχει δημιουργηθεί στο tree από το admin.

Configuration options

parent
Το **id** της κατηγορίας-γονέα (parent).

parentTag
Το **tag** της κατηγορίας-γονέα (parent).

*Χρησιμοποιούμε το **parent** ή το **parentTag**, ανάλογα με την περίπτωση.*

*Το root του δέντρου έχει πάντα **id = 1** και **tag = root**.*

depth (default:1)
Με το depth ορίζουμε σε πόσα επίπεδα του δένδρου θα ψάξει η inject **σε σχέση με το parent node** . Πχ για depth = 1 θα ψάξει τις υποκατηγορίες του parent node, για depth = 2 θα ψάξει και τις υποκατηγορίες αυτών κτλ.

orderBy
Επιστρέφει τα results ταξινομημένα με βάση κάποιο column. Μπορούμε να χρησιμοποιήσουμε επιπλέον το orderDir για να ορίσουμε την κατεύθυνση ταξινόμησης (ascending/descending)

orderDir , optional (asc/disc, default: asc)
Η κατεύθυνση ταξινόμησης (ascending/descending). Σε συνδυασμό με το orderBy.

Παραδείγματα

Εστω ότι δημιουργούμε ένα component κατηγορίες προιόντων και προσθέτουμε τις κατηγορίες παπούτσια, τσάντες. Μέσα στην κατηγορία παπούτσια προσθέτουμε τις υποκατηγορίες μπότες, αθλητικά, γόβες.
Αν θέλουμε να δείξουμε όλες τις κατηγορίες προιόντων κάνουμε το εξής:

```
1. {inject module="product_cats" action="children" parent="1" assign="product_cats"}
```

ή

```
1. {inject module="product_cats" action="children" parentTag= 'root' " assign="produc
   t_cats"}
```

Επιστρέφει: παπούτσια, τσάντες.

```
1. {inject module="product_cats" action="children" parentTag= 'root' " depth="2" assig
   n="product_cats"}
```

Επιστρέφει: παπούτσια, μπότες, αθλητικά, γόβες, τσάντες.

```
1. {inject module="product_cats" action="children" parent= '5' " assign="produ
   ct_cats"}
```

Επιστρέφει τα παιδιά της κατηγορίας με id = 5 (παπούτσια): μπότες, αθλητικά, γόβες.

10.2.3.6 getParent

Η getParent χρησιμοποιείται μόνο για components τύπου **categories**. Επιστρέφει τον γονέα ενός συγκεκριμένου node (στο αμέσως προηγούμενο επίπεδο) με βάση την ιεραρχία που έχει δημιουργηθεί στο tree από το admin μέσα σε ένα array.

id

Το id του node

Article II. Παράδειγμα:

Εστω ότι έχουμε φτιάξει ένα δέντρο με τις εξής κατηγορίες:

Root

- Παπούτσια

- Γυναικεία παπούτσια

- Μπότες

- Αθλητικά

και θέλουμε να πάρουμε την κατηγορία στην οποια ανηκουν οι ‘Μπότες’ (id=5).

```
1. {inject module="products_cats" action="getParent" id=5 assign="parent"}
```

Επιστρέφει την κατηγορία ‘Γυναικεία παπούτσια’

10.2.4 Modifiers

strip_html_tags

Βελτίωση της strip_tags. Η strip_html_tags μπορεί να μην κάνει strip κάποια συγκεκριμένα tags.

Παραδείγματα:

```
1. {$var|strip_html_tags:'<b><p>'}
```

Θα κάνει strip όλα τα tags εκτός των <b> και <p>. Τα tags γίνονται replace με κενό.

```
1. {$var|strip_html_tags:false:'<b><p>'}
```

Χρησιμοποιώντας :false τα tags δεν γίνονται replace με κενό, αφαιρούνται τελείως.

strip_selected_tags

Διαγράφει συγκεκριμένα tags από ένα string πχ

```
1. {$content|strip_selected_tags:'<p><span>'}
```

διαγράφει όλα τα <p> και tags.

Αν θέλουμε να διαγράψουμε και το περιεχόμενό τους δινουμε τιμή true στην παράμετρο skipContent:

```
1. {$content|strip_selected_tags:'<p><span>':true}
```

strip_tags_except

Διαγράφει όλα τα html tags από ένα string εκτός από κάποια που ορίζουμε εμείς πχ

```
1. {$content|strip_tags_except:'<b><strong>'}
```

Διαγράφει όλα τα tags εκτός από <b>,

10.3 jQuery plugins

Χρησιμοποιήθηκαν διάφορα jQuery plugins για να εμφανιστεί η σελίδα στη μορφή που είναι.
Τα πιο βασικά από αυτά είναι:

JCAROUSEL

Δημιουργεί ένα carousel με content μέσα και το χρησιμοποιουμε για να εμφανίζουμε τις προσφορές ημέρας.

Lightbox

Το lightbox είναι το σημείο στο οποίο προβάλλονται οι φωτογραφίες όταν κάνετε κλικ σε οποιαδήποτε μικρογραφία φωτογραφίας. Από την προβολή lightbox, μπορείτε να πλοηγηθείτε στο λεύκωμα και να εκτελέσετε διάφορες ενέργειες στη φωτογραφία.

simpleFAQ

Εμφανίζει το FAQ με τη μορφή ερωτήσεις και πατώντας πάνω σε ένα εικονίδιο κανει slide προς τα κάτω και εμφανίζεται η απάντηση.

PrintArea

Ορίζει μια εκτυπώσιμη περιοχή και την εμφανίζει με το print.css όταν πατηθεί η αντίστοιχη επιλογή.

Progressbar

Δημιουργεί την μπάρα προόδου πωλήσεων κάτω από το χρονόμετρο που δείχνει τη διαθεσιμότητα

Στο παράρτημα Β υπάρχουν στοιχεία κώδικα των templates που δημιουργήθηκαν για την εφαρμογή καθώς επίσης και στο *παράρτημα Γ* κομμάτια PHP κώδικα.

ΚΕΦΑΛΑΙΟ 11

ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

Επέκταση της διαχείρισης στις συνεργαζόμενες εταιρείες

Η εφαρμογή θα μπορούσε να επεκταθεί ακόμα παραπάνω δίνοντας κάποιες δυνατότητες στις συνεργαζόμενες εταιρείες. Ήδη υπάρχει η υποδομή σε κάποια από τα components και μένει να γραφεί ο ανάλογος κώδικας που θα δίνει τις επιπλέον δυνατότητες. Θα πρέπει όμως να οριστούν πρώτα οι επιπλέον λειτουργίες και πόσο πολύ πρέπει να επεκταθούν.

Παρακολούθηση προσφορών

Να δωθεί η δυνατότητα σε μία συνεργαζόμενη εταιρεία να μπει στο σύστημα και να μπορεί να παρακολουθεί τις προσφορές που έχει αναρτήσει είτε είναι τρέχουσες είτε είναι παλιότερες. Να μπορεί επίσης να εκτυπώνει τη λίστα με τα κουπόνια. Να μπορεί επίσης το σύστημα να κάνει αυτόματη εκκαθάριση των ποσοστών που πρέπει να λάβει η κάθε μεριά. Για να γίνει αυτό θα πρέπει να προστεθεί ένα πεδίο ποσοστού στην κάθε προσφορά για να γίνεται αυτός ο υπολογισμός.

Προσθήκη προσφορών

Να δωθεί η δυνατότητα μπαίνοντας η επιχείρηση να βάζει μόνη της τις προσφορές στο σύστημα.

Αποστολή newsletter

Να μπορεί η επιχείρηση να στέλνει newsletter στους πελάτες του. Στους χρήστες δηλαδή που έχουν αγοράσει κάποια προσφορά τους και όχι σε όλους.

Βιβλιογραφία

- http://www.w3schools.com
- http://www.php.net
- http://www.mysql.com
- http://www.w3c.com
- http://domscripting.com
- http://www.web-resources.eu
- http://www.smarty.com
- http://el.wikiversity.org
- http://www.observatory.gr
- http://www.wlearn.gr
- http://www.new-media.gr
- http://www.in2life.gr
- http://www.anyware.gr
- http://www.groupon.gr
- http://www.goldendeals.gr

- Keith, J., (2005), *Web Design with JavaScript and the Document Object Model*, Apress, Berkeley, USA.

ΠΑΡΑΡΤΗΜΑ Α

Κώδικάς MySQL

```
-- ----------------------------------------------------------------------------------------
-- Table structure for `basket_items_core`
-- ----------------------------------------------------------------------------------------

DROP TABLE IF EXISTS `basket_items_core`;

CREATE TABLE `basket_items_core` (
`id` int(11) NOT NULL AUTO_INCREMENT,
`basket_id` int(11) DEFAULT NULL,
`component_name` longtext COLLATE utf8_unicode_ci,
`component_id` longtext COLLATE utf8_unicode_ci,
`price` float(10,3) DEFAULT NULL,
`quantity` longtext COLLATE utf8_unicode_ci,
`date_added` date DEFAULT NULL,
`color` longtext COLLATE utf8_unicode_ci,
`branch` longtext COLLATE utf8_unicode_ci,
`size` longtext COLLATE utf8_unicode_ci,
`meta_visible` tinyint(1) NOT NULL DEFAULT '0',
PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=4 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `basket_items_locale`
-- ----------------------------------------------------------------------------------------
```

1. DROP TABLE IF EXISTS `basket_items_locale`;
2.
3. CREATE TABLE `basket_items_locale` (
4. `basket_items_id` int(11) NOT NULL DEFAULT '0',
5. `locale_id` int(3) NOT NULL,
6. PRIMARY KEY (`basket_items_id`,`locale_id`)
7.) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

```
-- ----------------------------------------------------------------------------------------
-- Table structure for `baskets_core`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `baskets_core`;
CREATE TABLE `baskets_core` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `basketid` longtext COLLATE utf8_unicode_ci,
```

```
  `ref` longtext COLLATE utf8_unicode_ci,
  `lang` longtext COLLATE utf8_unicode_ci,
  `title` longtext COLLATE utf8_unicode_ci,
  `user_id` int(11) DEFAULT NULL,
  `status` longtext COLLATE utf8_unicode_ci,
  `status_admin` longtext COLLATE utf8_unicode_ci,
  `date_created` date DEFAULT NULL,
  `date_submitted` date DEFAULT NULL,
  `date_ready` date DEFAULT NULL,
  `date_delivered` date DEFAULT NULL,
  `products_price` float(10,3) DEFAULT NULL,
  `products_price_fpa` float(10,3) DEFAULT NULL,
  `shipping_price` float(10,3) DEFAULT NULL,
  `total_price` float(10,3) DEFAULT NULL,
  `barcode` longtext COLLATE utf8_unicode_ci,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=4 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `baskets_locale`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `baskets_locale`;
CREATE TABLE `baskets_locale` (
  `baskets_id` int(11) NOT NULL DEFAULT '0',
  `locale_id` int(3) NOT NULL,
  PRIMARY KEY (`baskets_id`,`locale_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `characteristics_cats_core`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `characteristics_cats_core`;
CREATE TABLE `characteristics_cats_core` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `lft` int(11) DEFAULT NULL,
  `level` int(11) DEFAULT NULL,
  `rght` int(11) DEFAULT NULL,
  `tag` longtext COLLATE utf8_unicode_ci,
  `meta_visible` tinyint(1) DEFAULT '0',
```

```
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=2 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------
-- Table structure for `characteristics_cats_locale`
-- ----------------------------------------------------------------------------
DROP TABLE IF EXISTS `characteristics_cats_locale`;
CREATE TABLE `characteristics_cats_locale` (
  `characteristics_cats_id` int(11) NOT NULL DEFAULT '0',
  `locale_id` int(3) NOT NULL,
  `title` longtext COLLATE utf8_unicode_ci,
  PRIMARY KEY (`characteristics_cats_id`,`locale_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------
-- Table structure for `cities_core`
-- ----------------------------------------------------------------------------
DROP TABLE IF EXISTS `cities_core`;
CREATE TABLE `cities_core` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `tag` longtext COLLATE utf8_unicode_ci,
  `xx__order` int(11) DEFAULT NULL,
  `meta_visible` tinyint(1) DEFAULT '0',
  `central` tinyint(1) NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=5 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------
-- Table structure for `cities_locale`
-- ----------------------------------------------------------------------------
DROP TABLE IF EXISTS `cities_locale`;
CREATE TABLE `cities_locale` (
  `cities_id` int(11) NOT NULL DEFAULT '0',
  `locale_id` int(3) NOT NULL,
  `title` longtext COLLATE utf8_unicode_ci,
  PRIMARY KEY (`cities_id`,`locale_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

```
-- ----------------------------------------------------------------------------------------
-- Table structure for `cities_offers_bonds`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `cities_offers_bonds`;
CREATE TABLE `cities_offers_bonds` (
  `master_id` int(11) NOT NULL,
  `slave_id` int(11) NOT NULL,
  `xx__order` int(11) DEFAULT NULL,
  `re__order` int(11) DEFAULT NULL,
  PRIMARY KEY (`master_id`,`slave_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `citiespartners_bonds`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `citiespartners_bonds`;
CREATE TABLE `citiespartners_bonds` (
  `master_id` int(11) NOT NULL,
  `slave_id` int(11) NOT NULL,
  `xx__order` int(11) DEFAULT NULL,
  `re__order` int(11) DEFAULT NULL,
  PRIMARY KEY (`master_id`,`slave_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `contact_core`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `contact_core`;
CREATE TABLE `contact_core` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `title` longtext COLLATE utf8_unicode_ci,
  `name` longtext COLLATE utf8_unicode_ci,
  `surname` longtext COLLATE utf8_unicode_ci,
  `address` longtext COLLATE utf8_unicode_ci,
  `email` longtext COLLATE utf8_unicode_ci,
  `phone` longtext COLLATE utf8_unicode_ci,
  `subject` longtext COLLATE utf8_unicode_ci,
  `comments` longtext COLLATE utf8_unicode_ci,
  `captcha` longtext COLLATE utf8_unicode_ci,
  PRIMARY KEY (`id`)
```

```
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `contact_locale`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `contact_locale`;
CREATE TABLE `contact_locale` (
  `contact_id` int(11) NOT NULL DEFAULT '0',
  `locale_id` int(3) NOT NULL,
  PRIMARY KEY (`contact_id`,`locale_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `coupons_core`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `coupons_core`;
CREATE TABLE `coupons_core` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `title` date DEFAULT NULL,
  `customer` int(11) DEFAULT NULL,
  `customer__order` int(11) DEFAULT NULL,
  `barcode` longtext COLLATE utf8_unicode_ci,
  `quantity_no` longtext COLLATE utf8_unicode_ci,
  `owner` longtext COLLATE utf8_unicode_ci,
  `basket_id` int(11) DEFAULT NULL,
  `basket_items_id` int(11) DEFAULT NULL,
  `active` tinyint(1) NOT NULL DEFAULT '0',
  `random_code` longtext COLLATE utf8_unicode_ci,
  `offer` int(11) DEFAULT NULL,
  `xx__order` int(11) DEFAULT NULL,
  `meta_visible` tinyint(1) DEFAULT '0',
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=5 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;
```

```
-- ----------------------------------------------------------------------------------------
-- Table structure for `coupons_locale`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `coupons_locale`;
CREATE TABLE `coupons_locale` (
  `coupons_id` int(11) NOT NULL DEFAULT '0',
  `locale_id` int(3) NOT NULL,
  PRIMARY KEY (`coupons_id`,`locale_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `credits_core`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `credits_core`;
CREATE TABLE `credits_core` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `tag` longtext COLLATE utf8_unicode_ci,
  `price` float(10,3) DEFAULT NULL,
  `xx__order` int(11) DEFAULT NULL,
  `meta_visible` tinyint(1) DEFAULT '0',
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=2 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `credits_locale`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `credits_locale`;
CREATE TABLE `credits_locale` (
  `credits_id` int(11) NOT NULL DEFAULT '0',
  `locale_id` int(3) NOT NULL,
  `title` longtext COLLATE utf8_unicode_ci,
  `description` longtext COLLATE utf8_unicode_ci,
  PRIMARY KEY (`credits_id`,`locale_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

```
-- ----------------------------------------------------------------------------------------
-- Table structure for `customers_bonds`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `customers_bonds`;
CREATE TABLE `customers_bonds` (
  `master_id` int(11) NOT NULL,
  `slave_id` int(11) NOT NULL,
  `xx__order` int(11) DEFAULT NULL,
  `re__order` int(11) DEFAULT NULL,
  PRIMARY KEY (`master_id`,`slave_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `customers_cats_core`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `customers_cats_core`;
CREATE TABLE `customers_cats_core` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `lft` int(11) DEFAULT NULL,
  `level` int(11) DEFAULT NULL,
  `rght` int(11) DEFAULT NULL,
  `tag` longtext COLLATE utf8_unicode_ci,
  `meta_visible` tinyint(1) DEFAULT '0',
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=6 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `customers_cats_locale`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `customers_cats_locale`;
CREATE TABLE `customers_cats_locale` (
  `customers_cats_id` int(11) NOT NULL DEFAULT '0',
  `locale_id` int(3) NOT NULL,
  `title` longtext COLLATE utf8_unicode_ci,
  PRIMARY KEY (`customers_cats_id`,`locale_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

```
-- ----------------------------------------------------------------------------
-- Table structure for `customers_core`
-- ----------------------------------------------------------------------------
DROP TABLE IF EXISTS `customers_core`;
CREATE TABLE `customers_core` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `title` longtext COLLATE utf8_unicode_ci,
  `email` longtext COLLATE utf8_unicode_ci,
  `password` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  `name` longtext COLLATE utf8_unicode_ci,
  `surname` longtext COLLATE utf8_unicode_ci,
  `address` longtext COLLATE utf8_unicode_ci,
  `zipcode` longtext COLLATE utf8_unicode_ci,
  `city` longtext COLLATE utf8_unicode_ci,
  `country` int(11) DEFAULT NULL,
  `phone` longtext COLLATE utf8_unicode_ci,
  `customers_categories` int(11) DEFAULT NULL,
  `newsletter` tinyint(1) NOT NULL DEFAULT '0',
  `captcha` longtext COLLATE utf8_unicode_ci,
  `credits` float(10,3) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=2 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------
-- Table structure for `customers_locale`
-- ----------------------------------------------------------------------------
DROP TABLE IF EXISTS `customers_locale`;
CREATE TABLE `customers_locale` (
  `customers_id` int(11) NOT NULL DEFAULT '0',
  `locale_id` int(3) NOT NULL,
  PRIMARY KEY (`customers_id`,`locale_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------
-- Table structure for `emailfriend_core`
-- ----------------------------------------------------------------------------
DROP TABLE IF EXISTS `emailfriend_core`;
CREATE TABLE `emailfriend_core` (
```

```
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `title` longtext COLLATE utf8_unicode_ci,
  `email` longtext COLLATE utf8_unicode_ci,
  `myemail` longtext COLLATE utf8_unicode_ci,
  `name` longtext COLLATE utf8_unicode_ci,
  `comments` longtext COLLATE utf8_unicode_ci,
  `link` longtext COLLATE utf8_unicode_ci,
  `captcha` longtext COLLATE utf8_unicode_ci,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------
-- Table structure for `emailfriend_locale`
-- ----------------------------------------------------------------------------
DROP TABLE IF EXISTS `emailfriend_locale`;
CREATE TABLE `emailfriend_locale` (
  `emailfriend_id` int(11) NOT NULL DEFAULT '0',
  `locale_id` int(3) NOT NULL,
  PRIMARY KEY (`emailfriend_id`,`locale_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------
-- Table structure for `faqs_cats_core`
-- ----------------------------------------------------------------------------
DROP TABLE IF EXISTS `faqs_cats_core`;
CREATE TABLE `faqs_cats_core` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `lft` int(11) DEFAULT NULL,
  `level` int(11) DEFAULT NULL,
  `rght` int(11) DEFAULT NULL,
  `tag` longtext COLLATE utf8_unicode_ci,
  `meta_visible` tinyint(1) DEFAULT '0',
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=3 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;
```

```
-- ----------------------------------------------------------------------------------------
-- Table structure for `faqs_cats_locale`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `faqs_cats_locale`;
CREATE TABLE `faqs_cats_locale` (
  `faqs_cats_id` int(11) NOT NULL DEFAULT '0',
  `locale_id` int(3) NOT NULL,
  `title` longtext COLLATE utf8_unicode_ci,
  `body` longtext COLLATE utf8_unicode_ci,
  PRIMARY KEY (`faqs_cats_id`,`locale_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `faqs_core`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `faqs_core`;
CREATE TABLE `faqs_core` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `tag` longtext COLLATE utf8_unicode_ci,
  `faqs_categories` int(11) DEFAULT NULL,
  `faqs_categories__order` int(11) DEFAULT NULL,
  `xx__order` int(11) DEFAULT NULL,
  `meta_visible` tinyint(1) DEFAULT '0',
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=19 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `faqs_locale`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `faqs_locale`;
CREATE TABLE `faqs_locale` (
  `faqs_id` int(11) NOT NULL DEFAULT '0',
  `locale_id` int(3) NOT NULL,
  `title` longtext COLLATE utf8_unicode_ci,
  `body` longtext COLLATE utf8_unicode_ci,
  PRIMARY KEY (`faqs_id`,`locale_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

```
-- ----------------------------------------------------------------------------------------
-- Table structure for `locales`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `locales`;
CREATE TABLE `locales` (
  `id` int(5) NOT NULL AUTO_INCREMENT,
  `locale` varchar(2) COLLATE utf8_unicode_ci NOT NULL,
  `isdefault` tinyint(1) NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`,`locale`)
) ENGINE=MyISAM AUTO_INCREMENT=3 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `mailing_lists_queue_core`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `mailing_lists_queue_core`;
CREATE TABLE `mailing_lists_queue_core` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `newsletter_tag` longtext COLLATE utf8_unicode_ci,
  `email` longtext COLLATE utf8_unicode_ci,
  `name` longtext COLLATE utf8_unicode_ci,
  `surname` longtext COLLATE utf8_unicode_ci,
  `activation_code` longtext COLLATE utf8_unicode_ci,
  `xx__order` int(11) DEFAULT NULL,
  `meta_visible` tinyint(1) DEFAULT '0',
  PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `mailing_lists_queue_locale`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `mailing_lists_queue_locale`;
CREATE TABLE `mailing_lists_queue_locale` (
  `mailing_lists_queue_id` int(11) NOT NULL DEFAULT '0',
  `locale_id` int(3) NOT NULL,
  PRIMARY KEY (`mailing_lists_queue_id`,`locale_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

```
-- ----------------------------------------------------------------------------------------
-- Table structure for `news_core`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `news_core`;
CREATE TABLE `news_core` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `datePublished` date DEFAULT NULL,
  `category` longtext COLLATE utf8_unicode_ci,
  `image` longtext COLLATE utf8_unicode_ci,
  `document` longtext COLLATE utf8_unicode_ci,
  `meta_visible` tinyint(1) DEFAULT '0',
  PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `news_locale`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `news_locale`;
CREATE TABLE `news_locale` (
  `news_id` int(11) NOT NULL DEFAULT '0',
  `locale_id` int(3) NOT NULL,
  `title` longtext COLLATE utf8_unicode_ci,
  `body` longtext COLLATE utf8_unicode_ci,
  PRIMARY KEY (`news_id`,`locale_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `newsletter_users_core`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `newsletter_users_core`;
CREATE TABLE `newsletter_users_core` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `title` longtext COLLATE utf8_unicode_ci,
  `email` longtext COLLATE utf8_unicode_ci,
  `name` longtext COLLATE utf8_unicode_ci,
  `surname` longtext COLLATE utf8_unicode_ci,
  `city` longtext COLLATE utf8_unicode_ci,
  `agree_with_terms` tinyint(1) NOT NULL DEFAULT '0',
  `interests` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  `active` tinyint(1) NOT NULL DEFAULT '0',
```

```
  `activation_code` longtext COLLATE utf8_unicode_ci,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------
-- Table structure for `newsletter_users_locale`
-- ----------------------------------------------------------------------------
DROP TABLE IF EXISTS `newsletter_users_locale`;
CREATE TABLE `newsletter_users_locale` (
  `newsletter_users_id` int(11) NOT NULL DEFAULT '0',
  `locale_id` int(3) NOT NULL,
  PRIMARY KEY (`newsletter_users_id`,`locale_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------
-- Table structure for `newsletters_core`
-- ----------------------------------------------------------------------------
DROP TABLE IF EXISTS `newsletters_core`;
CREATE TABLE `newsletters_core` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `tag` longtext COLLATE utf8_unicode_ci,
  `dateAdded` date DEFAULT NULL,
  `partner` int(11) DEFAULT NULL,
  `partner__order` int(11) DEFAULT NULL,
  `xx__order` int(11) DEFAULT NULL,
  `meta_visible` tinyint(1) DEFAULT '0',
  PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------
-- Table structure for `newsletters_locale`
-- ----------------------------------------------------------------------------
DROP TABLE IF EXISTS `newsletters_locale`;
CREATE TABLE `newsletters_locale` (
  `newsletters_id` int(11) NOT NULL DEFAULT '0',
  `locale_id` int(3) NOT NULL,
  `title` longtext COLLATE utf8_unicode_ci,
  `body` longtext COLLATE utf8_unicode_ci,
  PRIMARY KEY (`newsletters_id`,`locale_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

```
-- ----------------------------------------------------------------------------------------
-- Table structure for `offer_cities_core`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `offer_cities_core`;
CREATE TABLE `offer_cities_core` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `city` int(11) DEFAULT NULL,
  `city__order` int(11) DEFAULT NULL,
  `all_sub_shops` tinyint(1) NOT NULL DEFAULT '0',
  `offer_cities_shops` int(11) DEFAULT NULL,
  `offer` int(11) DEFAULT NULL,
  `offer__order` int(11) DEFAULT NULL,
  `xx__order` int(11) DEFAULT NULL,
  `meta_visible` tinyint(1) DEFAULT '0',
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=23 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `offer_cities_locale`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `offer_cities_locale`;
CREATE TABLE `offer_cities_locale` (
  `offer_cities_id` int(11) NOT NULL DEFAULT '0',
  `locale_id` int(3) NOT NULL,
  `receive_one_branch` longtext COLLATE utf8_unicode_ci,
  PRIMARY KEY (`offer_cities_id`,`locale_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `offer_products_core`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `offer_products_core`;
CREATE TABLE `offer_products_core` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `tag` longtext COLLATE utf8_unicode_ci,
  `code` longtext COLLATE utf8_unicode_ci,
  `images` longtext COLLATE utf8_unicode_ci,
  `offer` int(11) DEFAULT NULL,
```

```
  `offer__order` int(11) DEFAULT NULL,
  `xx__order` int(11) DEFAULT NULL,
  `meta_visible` tinyint(1) DEFAULT '0',
  `barcode` longtext COLLATE utf8_unicode_ci,
  `price` float(10,3) DEFAULT NULL,
  `discount` float(10,3) DEFAULT NULL,
  `availability` float(10,3) DEFAULT NULL,
  `limit_per_user` float(10,3) DEFAULT NULL,
  `checkAvailability` tinyint(1) NOT NULL DEFAULT '0',
  `price_disc` float(10,3) DEFAULT NULL,
  `moufa_sold` float(10,3) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=29 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `offer_products_locale`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `offer_products_locale`;
CREATE TABLE `offer_products_locale` (
  `offer_products_id` int(11) NOT NULL DEFAULT '0',
  `locale_id` int(3) NOT NULL,
  `title` longtext COLLATE utf8_unicode_ci,
  `description` longtext COLLATE utf8_unicode_ci,
  `body` longtext COLLATE utf8_unicode_ci,
  PRIMARY KEY (`offer_products_id`,`locale_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `offers_core`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `offers_core`;
CREATE TABLE `offers_core` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `tag` longtext COLLATE utf8_unicode_ci,
  `price` float(10,3) DEFAULT NULL,
  `discount` float(10,3) DEFAULT NULL,
  `dateOfferStarted` date DEFAULT NULL,
  `dateOfferFinished` date DEFAULT NULL,
  `dateReceiveStarted` date DEFAULT NULL,
```

```
  `dateOReceiveFinished` date DEFAULT NULL,
  `checkAvailability` tinyint(1) NOT NULL DEFAULT '0',
  `availability` float(10,3) DEFAULT NULL,
  `position` longtext COLLATE utf8_unicode_ci,
  `offerType` longtext COLLATE utf8_unicode_ci,
  `receiveTerms__6` longtext COLLATE utf8_unicode_ci,
  `receiveTerms__order` int(11) DEFAULT NULL,
  `xx__order` int(11) DEFAULT NULL,
  `meta_visible` tinyint(1) DEFAULT '0',
  `offers_city` int(11) DEFAULT NULL,
  `code` longtext COLLATE utf8_unicode_ci,
  `partner` int(11) DEFAULT NULL,
  `partner__order` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=29 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `offers_locale`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `offers_locale`;
CREATE TABLE `offers_locale` (
  `offers_id` int(11) NOT NULL DEFAULT '0',
  `locale_id` int(3) NOT NULL,
  `title` longtext COLLATE utf8_unicode_ci,
  `receiveTerms` longtext COLLATE utf8_unicode_ci,
  PRIMARY KEY (`offers_id`,`locale_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `offers_partners_core`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `offers_partners_core`;
CREATE TABLE `offers_partners_core` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `customer` int(11) DEFAULT NULL,
  `customer__order` int(11) DEFAULT NULL,
  `offers_partners_shops` int(11) DEFAULT NULL,
  `offer` int(11) DEFAULT NULL,
  `offer__order` int(11) DEFAULT NULL,
```

```
  `xx__order` int(11) DEFAULT NULL,
  `meta_visible` tinyint(1) DEFAULT '0',
  `has_sub_shops` tinyint(1) NOT NULL DEFAULT '0',
  `city` int(11) DEFAULT NULL,
  `city__order` int(11) DEFAULT NULL,
  `offers_partners_city` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=14 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `offers_partners_locale`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `offers_partners_locale`;
CREATE TABLE `offers_partners_locale` (
  `offers_partners_id` int(11) NOT NULL DEFAULT '0',
  `locale_id` int(3) NOT NULL,
  PRIMARY KEY (`offers_partners_id`,`locale_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `pages_cats_core`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `pages_cats_core`;
CREATE TABLE `pages_cats_core` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `lft` int(11) DEFAULT NULL,
  `level` int(11) DEFAULT NULL,
  `rght` int(11) DEFAULT NULL,
  `tag` longtext COLLATE utf8_unicode_ci,
  `meta_visible` tinyint(1) DEFAULT '0',
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=3 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;
```

```
-- ----------------------------------------------------------------------------------------
-- Table structure for `pages_cats_locale`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `pages_cats_locale`;
CREATE TABLE `pages_cats_locale` (
  `pages_cats_id` int(11) NOT NULL DEFAULT '0',
  `locale_id` int(3) NOT NULL,
  `title` longtext COLLATE utf8_unicode_ci,
  `body` longtext COLLATE utf8_unicode_ci,
  PRIMARY KEY (`pages_cats_id`,`locale_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `pages_core`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `pages_core`;
CREATE TABLE `pages_core` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `tag` longtext COLLATE utf8_unicode_ci,
  `pages_categories` int(11) DEFAULT NULL,
  `pages_categories__order` int(11) DEFAULT NULL,
  `first_page` tinyint(1) NOT NULL DEFAULT '0',
  `images` longtext COLLATE utf8_unicode_ci,
  `xx__order` int(11) DEFAULT NULL,
  `meta_visible` tinyint(1) DEFAULT '0',
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=11 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `pages_locale`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `pages_locale`;
CREATE TABLE `pages_locale` (
  `pages_id` int(11) NOT NULL DEFAULT '0',
  `locale_id` int(3) NOT NULL,
  `title` longtext COLLATE utf8_unicode_ci,
  `body` longtext COLLATE utf8_unicode_ci,
  PRIMARY KEY (`pages_id`,`locale_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

```
-- ----------------------------------------------------------------------------
-- Table structure for `partners_core`
-- ----------------------------------------------------------------------------
DROP TABLE IF EXISTS `partners_core`;
CREATE TABLE `partners_core` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `tag` longtext COLLATE utf8_unicode_ci,
  `code` longtext COLLATE utf8_unicode_ci,
  `logotype` longtext COLLATE utf8_unicode_ci,
  `user` int(11) DEFAULT NULL,
  `user__order` int(11) DEFAULT NULL,
  `companyType` longtext COLLATE utf8_unicode_ci,
  `partners_cities` int(11) DEFAULT NULL,
  `partnerName` longtext COLLATE utf8_unicode_ci,
  `address` longtext COLLATE utf8_unicode_ci,
  `phone` longtext COLLATE utf8_unicode_ci,
  `email` longtext COLLATE utf8_unicode_ci,
  `url` longtext COLLATE utf8_unicode_ci,
  `xx__order` int(11) DEFAULT NULL,
  `meta_visible` tinyint(1) DEFAULT '0',
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=19 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------
-- Table structure for `partners_locale`
-- ----------------------------------------------------------------------------
DROP TABLE IF EXISTS `partners_locale`;
CREATE TABLE `partners_locale` (
  `partners_id` int(11) NOT NULL DEFAULT '0',
  `locale_id` int(3) NOT NULL,
  `title` longtext COLLATE utf8_unicode_ci,
  `description` longtext COLLATE utf8_unicode_ci,
  `map_script` longtext COLLATE utf8_unicode_ci,
  `tips` longtext COLLATE utf8_unicode_ci,
  PRIMARY KEY (`partners_id`,`locale_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

```
-- ----------------------------------------------------------------------------------------
-- Table structure for `payment_methods_core`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `payment_methods_core`;
CREATE TABLE `payment_methods_core` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `baskets_id` int(11) DEFAULT NULL,
  `payment_method` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  `meta_visible` tinyint(1) NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=2 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `payment_methods_locale`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `payment_methods_locale`;
CREATE TABLE `payment_methods_locale` (
  `payment_methods_id` int(11) NOT NULL DEFAULT '0',
  `locale_id` int(3) NOT NULL,
  PRIMARY KEY (`payment_methods_id`,`locale_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `products_cats_core`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `products_cats_core`;
CREATE TABLE `products_cats_core` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `lft` int(11) DEFAULT NULL,
  `level` int(11) DEFAULT NULL,
  `rght` int(11) DEFAULT NULL,
  `tag` longtext COLLATE utf8_unicode_ci,
  `image` longtext COLLATE utf8_unicode_ci,
  `meta_visible` tinyint(1) DEFAULT '0',
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=3 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;
```

```
-- ----------------------------------------------------------------------------------------
-- Table structure for `products_cats_locale`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `products_cats_locale`;
CREATE TABLE `products_cats_locale` (
  `products_cats_id` int(11) NOT NULL DEFAULT '0',
  `locale_id` int(3) NOT NULL,
  `title` longtext COLLATE utf8_unicode_ci,
  PRIMARY KEY (`products_cats_id`,`locale_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
-- ----------------------------------------------------------------------------------------
-- Table structure for `products_core`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `products_core`;
CREATE TABLE `products_core` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `tag` longtext COLLATE utf8_unicode_ci,
  `code` longtext COLLATE utf8_unicode_ci,
  `products_categories` int(11) DEFAULT NULL,
  `products_categories__order` int(11) DEFAULT NULL,
  `price` float(10,3) DEFAULT NULL,
  `price_disc` float(10,3) DEFAULT NULL,
  `images` longtext COLLATE utf8_unicode_ci,
  `newproduct` tinyint(1) NOT NULL DEFAULT '0',
  `availability` float(10,3) DEFAULT NULL,
  `xx__order` int(11) DEFAULT NULL,
  `meta_visible` tinyint(1) DEFAULT '0',
  `position` longtext COLLATE utf8_unicode_ci,
  `offer_product` int(11) DEFAULT NULL,
  `offer_product__order` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=25 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;
```

```
-- ----------------------------------------------------------------------------------------
-- Table structure for `products_locale`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `products_locale`;
CREATE TABLE `products_locale` (
  `products_id` int(11) NOT NULL DEFAULT '0',
  `locale_id` int(3) NOT NULL,
  `title` longtext COLLATE utf8_unicode_ci,
  `description` longtext COLLATE utf8_unicode_ci,
  `tags` longtext COLLATE utf8_unicode_ci,
  `body` longtext COLLATE utf8_unicode_ci,
  PRIMARY KEY (`products_id`,`locale_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `shops_core`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `shops_core`;
CREATE TABLE `shops_core` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `lft` int(11) DEFAULT NULL,
  `level` int(11) DEFAULT NULL,
  `rght` int(11) DEFAULT NULL,
  `address` longtext COLLATE utf8_unicode_ci,
  `zip_code` longtext COLLATE utf8_unicode_ci,
  `phone` longtext COLLATE utf8_unicode_ci,
  `city` int(11) DEFAULT NULL,
  `city__order` int(11) DEFAULT NULL,
  `partner` int(11) DEFAULT NULL,
  `partner__order` int(11) DEFAULT NULL,
  `state` int(11) DEFAULT NULL,
  `state__order` int(11) DEFAULT NULL,
  `meta_visible` tinyint(1) DEFAULT '0',
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=23 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;
```

```
-- ----------------------------------------------------------------------------------------
-- Table structure for `shops_locale`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `shops_locale`;
CREATE TABLE `shops_locale` (
  `shops_id` int(11) NOT NULL DEFAULT '0',
  `locale_id` int(3) NOT NULL,
  `title` longtext COLLATE utf8_unicode_ci,
  PRIMARY KEY (`shops_id`,`locale_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `shops_offer_cities_bonds`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `shops_offer_cities_bonds`;
CREATE TABLE `shops_offer_cities_bonds` (
  `master_id` int(11) NOT NULL,
  `slave_id` int(11) NOT NULL,
  `xx__order` int(11) DEFAULT NULL,
  `re__order` int(11) DEFAULT NULL,
  PRIMARY KEY (`master_id`,`slave_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `shopsoffers_partners_bonds`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `shopsoffers_partners_bonds`;
CREATE TABLE `shopsoffers_partners_bonds` (
  `master_id` int(11) NOT NULL,
  `slave_id` int(11) NOT NULL,
  `xx__order` int(11) DEFAULT NULL,
  `re__order` int(11) DEFAULT NULL,
  PRIMARY KEY (`master_id`,`slave_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

```
-- ----------------------------------------------------------------------------------------
-- Table structure for `states_core`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `states_core`;
CREATE TABLE `states_core` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `tag` longtext COLLATE utf8_unicode_ci,
  `xx__order` int(11) DEFAULT NULL,
  `meta_visible` tinyint(1) DEFAULT '0',
  `city` int(11) DEFAULT NULL,
  `city__order` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=15 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `states_locale`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `states_locale`;
CREATE TABLE `states_locale` (
  `states_id` int(11) NOT NULL DEFAULT '0',
  `locale_id` int(3) NOT NULL,
  `title` longtext COLLATE utf8_unicode_ci,
  PRIMARY KEY (`states_id`,`locale_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `statesshops_bonds`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `statesshops_bonds`;
CREATE TABLE `statesshops_bonds` (
  `master_id` int(11) NOT NULL,
  `slave_id` int(11) NOT NULL,
  `xx__order` int(11) DEFAULT NULL,
  `re__order` int(11) DEFAULT NULL,
  PRIMARY KEY (`master_id`,`slave_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

```
-- ----------------------------------------------------------------------------------------
-- Table structure for `static_core`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `static_core`;
CREATE TABLE `static_core` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `tag` longtext COLLATE utf8_unicode_ci,
  `images` longtext COLLATE utf8_unicode_ci,
  `search_url` longtext COLLATE utf8_unicode_ci,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=17 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `static_locale`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `static_locale`;
CREATE TABLE `static_locale` (
  `static_id` int(11) NOT NULL DEFAULT '0',
  `locale_id` int(3) NOT NULL,
  `title` longtext COLLATE utf8_unicode_ci,
  `body` longtext COLLATE utf8_unicode_ci,
  PRIMARY KEY (`static_id`,`locale_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `users_bonds`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `users_bonds`;
CREATE TABLE `users_bonds` (
  `master_id` int(11) NOT NULL,
  `slave_id` int(11) NOT NULL,
  `xx__order` int(11) DEFAULT NULL,
  `re__order` int(11) DEFAULT NULL,
  PRIMARY KEY (`master_id`,`slave_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

```
-- ----------------------------------------------------------------------------------------
-- Table structure for `users_cats_core`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `users_cats_core`;
CREATE TABLE `users_cats_core` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `lft` int(11) DEFAULT NULL,
  `level` int(11) DEFAULT NULL,
  `rght` int(11) DEFAULT NULL,
  `title` longtext COLLATE utf8_unicode_ci,
  `tag` longtext COLLATE utf8_unicode_ci,
  `meta_visible` tinyint(1) DEFAULT '0',
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=3 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `users_cats_locale`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `users_cats_locale`;
CREATE TABLE `users_cats_locale` (
  `users_cats_id` int(11) NOT NULL DEFAULT '0',
  `locale_id` int(3) NOT NULL,
  PRIMARY KEY (`users_cats_id`,`locale_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `users_core`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `users_core`;
CREATE TABLE `users_core` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `title` longtext COLLATE utf8_unicode_ci,
  `username` longtext COLLATE utf8_unicode_ci,
  `email` longtext COLLATE utf8_unicode_ci,
  `password` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  `name` longtext COLLATE utf8_unicode_ci,
  `surname` longtext COLLATE utf8_unicode_ci,
  `company` longtext COLLATE utf8_unicode_ci,
  `address` longtext COLLATE utf8_unicode_ci,
```

```
  `phone` longtext COLLATE utf8_unicode_ci,
  `fax` longtext COLLATE utf8_unicode_ci,
  `url` longtext COLLATE utf8_unicode_ci,
  `users_categories` int(11) DEFAULT NULL,
  `password_two` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=2 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;

-- ----------------------------------------------------------------------------------------
-- Table structure for `users_locale`
-- ----------------------------------------------------------------------------------------
DROP TABLE IF EXISTS `users_locale`;
CREATE TABLE `users_locale` (
  `users_id` int(11) NOT NULL DEFAULT '0',
  `locale_id` int(3) NOT NULL,
  PRIMARY KEY (`users_id`,`locale_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

ΠΑΡΑΡΤΗΜΑ Β

Κώδικάς PHP

Αποσπασμα κώδικα από το model.php

```
public function offers($position_tag=null, $offer_tag=null, $extra_field=null, $counter=null,
$all_sub_shops)
	{
		global $config;
		$params = array( 'position_tag' => $position_tag,  'offer_tag' => $offer_tag,
		check_availability'=> false,  showQuantity' => false, 'extra_field'=> $extra_field,
		'city'	=> $_SESSION[APPHASH]['city_selected'], 'specific_offer_tag'	=>
		$offer_tag, 'counter'=> $counter, 'all_sub_shops' => $all_sub_shops
	);
		require_once('resources/models/MyOffers.php');
		$site_offers = new MyOffers();
		if($params['position_tag']=='showStateBranches')
		{
			$site_offers->displayStateBranches($params);
		}
		elseif($params['position_tag']=='previous' && !$params['offer_tag'])
		{
			$site_offers->displayOtherOffers($params);
		}
		elseif($params['offer_tag'])
		{
			$site_offers->displayCentralOffer($params);
			$site_offers->displayOtherOffers($params);
		}
		elseif($params['position_tag'])
		{
			$site_offers->displayOffersInCategoryWithBelongTo($params);
		}
		else
		{
			$site_offers->displayAllPositions($params);
		}

		if($params['position_tag']!='showStateBranches')
		{
			assign('center', $_SESSION[APPHASH]['viewsFolder'].'offers.tpl');
			append($_SESSION[APPHASH]['viewsFolder'].'layout.tpl');
		}
	}
```

Απόσπασμα από το MyOffers.php

```
<?php
class MyOffers
{

  /* this function maybe not used in most sites */
        public function displayAllOffers($paramss) {
                global $l;
                $offers = db()->adodb->getArray("SELECT `title`, `tag` FROM offers");
                if($params['showbreadcrumb'])
                {
                        /* CUSTOM BREADCRUMB CALCULATION */
                        $trail = new Breadcrumb();
                        $trail->add($l['offers'], BASEURL.'/offers');
                        assign('myBreadcrumb', $trail->output());
                }

                assign('alloffers', $offers);
  }

        /* when somebody asks for the controller */
         public function displayAllPositions($params) {
                global $l;

                $parent_category = db()->adodb->getRow("SELECT id ,level, lft, rght FROM x
                x__typed WHERE ( tag = 'position' ) AND
                locale_id=".$_SESSION[APPHASH][locale][id]);

                $temp_query = "SELECT title, tag FROM xx__typed WHERE ( ( ( lft BETWEEN
                '".$parent_category['lft']."' AND '".$parent_category['rght']."' ) AND
                ( level-".$parent_category['level']."<=1 ) AND ( level>".$parent_category['level']." ) )
                AND locale_id= ".$_SESSION[APPHASH][locale][id]." ) AND locale_id=
                ".$_SESSION[APPHASH][locale][id]." ORDER BY lft asc";

                $offers_positions = db()->adodb->getArray($final_query);

                assign('alloffersPositions', $offers_positions);
  }
```

```
/* when somebody asks for a specific offer */
public function displayOffer($params) {
        global $l;
        $GeneralHelper = new GeneralHelpClass();
        global $modules;

        if($params['check_availability']){$check_availability = ' AND p.`availability`>0 ';}

        $offer =  db()->adodb->getRow("SELECT p.`id`, p.`title`, p.`tag`, p.`code`, p.`price`,
        p.`price_disc`, p.`discount` FROM offers p where (p.`tag`='".$params['offer_tag']."')
        and  p.`locale_id`=".$_SESSION[APPHASH][locale][id].$check_availability);

        /* RETREIVES THE PRODUCT FIELDS OF THE OFFER WICH ARE INTO THE
        THE TABLE offer_products */
        $offer_product =  db()->adodb->getRow("SELECT p.`tag`, p.`images`,
        p.`description` FROM offer_products p where (p.`offer`='".$offer['id']."') and
        p.`locale_id`=".$_SESSION[APPHASH][locale][id]);
        $offer_product = $GeneralHelper->relateResults(array($offer_product));
         /* TO MAKE ENCAPSULATED TABLES FOR IMAGES AND FILES */
        $offer['offer_product']=$offer_product;

        /* RETREIVES THE FIELDS FOR THE COMPANIES ARE WORKING
        TOGETHER FOR THE OFFER WICH ARE INTO THE THE TABLE
        offers_partners */
        //$offer_partners =  db()->adodb->getArray("SELECT * FROM offers_partners p
        where (p.`offer`='".$offer['id']."') and
        p.`locale_id`=".$_SESSION[APPHASH][locale][id]);

        $offer_partners = $modules['offers_partners']->all(array("where" => "offer =
        ".$offer['id'] , "getRelated" => "once"));

        $offer['offer_partners']=$offer_partners;

        $offer = $GeneralHelper->relateResults(array($offer));
        assign('offer_product', $offer_product);

        $offer = $GeneralHelper->relateResults(array($offer));
        assign('offer', $offer);

        $category = db()->adodb->getRow("SELECT `title`, `tag` FROM xx__typed where
```

```
        `tag`='".$params['position_tag']."' AND locale_id=".$_SESSION[APPHASH][locale][id]);

                assign('params', $params);

                if($params['showRelatedoffers']==1)
                 /* I HAVE IT SEPERATE FROM THE offer BECAUSE IT IS IMPEMENTED WITH
                HAS */
                {
                        $related_offers = db()->adodb->getArray("SELECT p.id, p.title, p.images,
                        p.tag, rp.offer_id FROM related_offers rp, offers p where rp.offer_id = p.id
                        and rp.related_offer_id=".$offer['id'] ." GROUP BY rp.offer_id");

                        foreach($related_offers as $related_offer)
                        {
                        $related_offers_array[] =
                        $GeneralHelper->relateResults(array($related_offer));
                        }
                        assign('related_offers', $related_offers_array);
                }
                assign ('params', $params);

    }
}
?>
```

ΠΑΡΑΡΤΗΜΑ Γ

Κώδικάς HTML, JAVASCRIPT μέσα στα template

Απόσπασμα από το offers.tpl

```
{if $previousOffers}
{include file='app/help_files/main_block_header.tpl'}
{*krumo value=$previousOffers*}
	{title text="`$l.previous_offers` - " insert="before"}
		<h1 class="article_title">{$l.offers}</h1>
		<h2 class="art-postheader">{$l.previous_offers} ({$l.offers_that_expired})</h2>
		<table id="offer-table" width="100%" cellpadding="0" cellspacing="1" border="0">
		</tr>
		 {foreach from=$previousOffers item="offer"}
		<div class="previousDeal">
		{if $offer.offer_products|@count>1}
		{foreach from=$offer.offer_products item="offer_product"}
		<div style="float:left; width:33%;text-align:center;">
		<div class="previousDealDescriprion_triple">
		<a href="{$BASEURL}/offers/past/{$offer.tag}" title="{$offer_product.title}">
		{$offer_product.title|mb_truncate:160|strip_tags}</a>
		</div>
		<div style="min-height:100px;">
		<a href="{$BASEURL}/offers/past/{$offer.tag}">
		<img src="{thumb src=$offer_product.product.images.0 propx=90 propy=90 zc=0}"
		alt="{$offer_product.title}" title="{$offer_product.title}"></a>
		</div>
		<div class="bought"><span class="bought_how_many">
		{assign var="solds" value=$offer.offer_products.0.moufa_sold +
		$offer.offer_products.0.sold.items_sold}
		{if $solds}{$solds|string_format:"%.0f"}{else}0{/if}</span><br /> {$l.bought}
		</div>
		<div class="our_price">{$l.our_price}:
		<span class="our_price_price">{$offer_product.price|string_format:"%.0f"}
		&euro;</span>
		</div>
		{/foreach}

		{else}
		<div class="previousDealDescriprion">
		<a href="{$BASEURL}/offers/past/{$offer.tag}" title="{$offer.offer_products.0.title}">
		{$offer.offer_products.0.body|mb_truncate:160|strip_tags}</a></div>
		<br />
```

```
<div style="float:left;">
<div class="cleared"></div>
<div class="bought"><span class="bought_how_many">
{assign var="solds"
value=$offer.offer_products.0.moufa_sold+$offer.offer_products.0.sold.items_sold}
{if $solds}{$solds|string_format:"%.0f"}{else}0{/if}</span><br /> {$l.bought}
</div>
<div class="our_price">
{if $offer.offer_products.0.price_disc && $offer.offer_products.0.price_disc>0}
{assign var="discount" value=$offer.offer_products.0.price_disc}
{else}
{assign var="discount" value=$offer.offer_products.0.price-
$offer.offer_products.0.price*$offer.offer_products.0.discount/100}
{/if}
{$l.our_price}: <span class="our_price_price">{$discount|string_format:"%.2f"}&euro;</a>
</div>
{if $offer.offer_products.0.price_disc && $offer.offer_products.0.price_disc>0 &&
$offer.offer_products.0.discount && $offer.offer_products.0.discount>0}
{if $offer.offer_products.0.price_disc && $offer.offer_products.0.price_disc>0}
<div class="our_price">{$l.aksia}: <span class="price_value"><del><b><font
color="#524A37">{$offer.offer_products.0.price|string_format:"%.2f"}&euro; </font></b>
</del></span></div>
<div class="our_price">{$l.discount}: <b><span
class="discount_value">{$offer.offer_products.0.discount|string_format:"%.0f"}%</span></b></div>
<div class="our_price">{$l.win}: <b><span class="win_value">{$offer.offer_products.0.price-
$offer.offer_products.0.price_disc}&euro;</span></b></div>
{else}
<div class="our_price">{$l.aksia}: <span class="price_value"><del><b><font
color="#524A37">{$offer.offer_products.0.price|string_format:"%.2f"}&euro; </font></b></del
></span></div>
<div class="our_price">{$l.discount}: <b><span
class="discount_value">{$offer.offer_products.0.discount|string_format:"%.0f"}%</span></b></div>
<div class="our_price">{$l.win}: <b><span
class="win_value">{$offer.offer_products.0.price*$offer.offer_products.0.discount/100}&euro;</spa
n></b></div>
{/if}

{/if}
</div>
```

```
<div class="previous_offer_img">
a href="{$BASEURL}/offers/past/{$offer.tag}"><img src="{thumb
src=$offer.offer_products.0.product.images.0 propx=162 propy=162 zc=0}" alt="{$offer.title}"
title="{$offer.title}"></a>
</div>
{/if}
</div>
{/foreach}
</table>

{include file='app/help_files/main_block_footer.tpl'}
{elseif $centralOffer.offer_products}
        {if $centralOffer.offer_products|@count>1}
        {assign var="counter" value=0}
        <div class="three_products_container">
        {assign var="only_once" value=0}
        {foreach from=$centralOffer.offer_products item="offer"}
        {if $offer.availability>0 && $offer.checkAvailability && $smarty.now|date_format:"%Y-%m-
%d" < $centralOffer.dateOfferFinished && $centralOffer.dateOfferStarted <
$centralOffer.dateOfferFinished && !$only_once}
{assign var="show_timer_from_this_offer" value=$offer.id} {* AN ESTO MIA PROSFORA DEN
EXEI LIKSEI DIKSE TON TIMER APO AUTIN TIN PROSFORA - OI ALLES AN EXOUN LIKSEI
TOTE THA DIKSEI MESA STIN KATHEMIA PSOSFORA TO MINIMA TOU SOLD OUT I TOU
ELIKSE *}
{assign var="only_once" value=1}
{else}
{assign var="show_timer_from_this_offer" value=$centralOffer.offer_products.0.id}
{/if}
{assign var="counter" value=$counter+1}
{include file='app/help_files/each_offer_timer.tpl'}
                        {include file='app/help_files/each_offer.tpl'}
                {/foreach}
        {elseif $centralOffer.offer_products|@count==1}
                {assign var="counter" value=1}
                {assign var="only_once" value=1}
                {include file='app/help_files/each_offer_curvy_header.tpl'}
                        {assign var="offer" value=$centralOffer.offer_products.0}
                        {assign var="show_timer_from_this_offer" value=$offer.id}
                        {include file='app/help_files/each_offer.tpl'}
                {include file='app/help_files/each_offer_curvy_footer.tpl'}
```

```
		{else}

		{/if}

		{if $centralOffer.offer_products|@count>1}
			{assign var="offer" value=$centralOffer.offer_products.0}
		{/if}

		<div style="float:right; width: 230px;">
			{include file='app/help_files/other_offers.tpl'}
			{include file='app/right_column.tpl'}
		</div>

		{include file='app/help_files/each_offer_curvy_header.tpl'}
			{include file='app/help_files/common_to_offers.tpl'}
		{include file='app/help_files/each_offer_curvy_footer.tpl'}

		</div>
{else}
		{include file='app/help_files/main_block_header.tpl'}
		{$l.no_access_or_no_content}
		{include file='app/help_files/main_block_footer.tpl'}
{/if}
```

each_offer_timer.tpl

```
{* I IMERA LIKSIS NA EINAI MEGALYTERI APO TIN SIMERINI IMERA KAI TIN IMERA ENARKSIS TIS PROSFORAS *}

{if $offer.id==$show_timer_from_this_offer}

{* ELEGXO KAI TO availability GIATI AN EXEI EKSANTLITHEI TO AVAILABILITY PREPEI NA THEOREITAI LHKSASA I PROSFORA *}

{if $offer.availability>0 && $offer.checkAvailability && $smarty.now|date_format:"%Y-%m-%d" < $centralOffer.dateOfferFinished && $centralOffer.dateOfferStarted < centralOffer.dateOfferFinished}
<div class="timer_container">

<div id="countdown_text"></div>
<span style="position:relative; top:-25px; color:#544937; font-size:11px;">
        {$l.days}
        {$l.hours}
        {$l.minutes}
        {$l.seconds}</span>
{literal}
<script type="text/javascript">
function calcage(secs) /* calculates secs given in hours, minutes, seconds format */
{
days = ((Math.floor(secs/86400))%100000).toString(); if (days.length < 2) days = "0" + days;
hours = ((Math.floor(secs/3600))%24).toString(); if (hours.length < 2) hours = "0" + hours;
min = ((Math.floor(secs/60))%60).toString(); if (min.length < 2) min = "0" + min;
sec = ((Math.floor(secs/1))%60).toString(); if (sec.length < 2) sec = "0" + sec;
//alert(min);
return days  + hours + +min+sec;
}

var countdown;
var countdown_number;

function countdown_init() {
countdown_number = {/literal}{makeTimeStamp date="`$centralOffer.dateOfferFinished`-00-00-00"}{literal};
countdown_trigger();
}
```

```
        function countdown_trigger()
        {
        if(countdown_number > 0)
        {
        countdown_number--;
        document.getElementById('countdown_text').innerHTML = '{/literal}<div
        class="timer_bg"><div style="font-size:13px; margin: 0 0 11px 0;
        color:#ffffff;">{$l.time_remaining}:</div>{literal}' + calcage(countdown_number) +
        '{/literal}</div>{literal}';
        if(countdown_number > 0)
        {
        countdown = setTimeout('countdown_trigger()', 1000);
        }}}
        countdown_init();
        </script>
        {/literal}
        </div>
        {else}
        <div class="timer_container" style=" padding: 20px;">{$l.offer_is_closed}</div>
        {/if}
{/if}
```