«Σύγχρονες μηχανές αναζήτησης μικρών αγγελιών»

Μελέτη για την ανταλλαγή δεδομένων και στην διασύνδεση συστημάτων μέσω internet.

Τμήμα: Πληροφορικής και Επικοινωνιών

Τ.Ε.Ι. Σερρών 2012

Σπουδαστής :Θεοδωρόπουλος Αλέξανδρος

Αριθμός Μητρώου: 252

Τμήμα Πληροφορικής και Επικοινωνιών

Ημ/νία : 2012

Υπεύθυνος Καθηγητής : Λάντζος Θεόδωρος

Περιεχόμενα.

1 Εισαγωγή

Η εργασία αυτή γίνεται στα πλαίσια πτυχιακής εργασίας του Τμήματος Πληροφορικής και Επικοινωνιών του Τεχνολογικού Εκπαιδευτικού Ιδρύματος Σερρών. Το αντικείμενο της ασχολείται με το προγραμματιστικό πεδίο της σχολής και κυρίως με το πεδίο του μαθήματος «Προγραμματιστικές Εφαρμογές στο Διαδίκτυο» αλλά περιέχει και στοιχεία του «δομημένου» και «αντικειμενοστραφούς προγραμματισμού» στη γλώσσα C# σαν μετεξέλιξη της C++. Χρησιμοποιούνται τεχνολογίες οι οποίες έχουν διδαχτεί στα πλαίσιο των παραπάνω μαθημάτων αλλά εμπεριέχει και άλλες σχετικές οι οποίες μελετήθηκαν από τον σπουδαστή με σκοπό την ολοκλήρωση του ζητούμενου θέματος. Οι τεχνολογίες αυτές, γνώστες και μη, αναφέρονται και αναλύονται σε ικανοποιητικό βάθος σε παρακάτω κεφάλαιο όσον αφορά τον τρόπο χρήσης τους και λειτουργίας τους καθώς και φυσικά όλες οι πηγές οι οποίες χρησιμοποιήθηκαν.

1.1)Σκοπός – Αντικείμενο Εργασίας

Το θέμα με το οποίο ασχολούμαστε είναι η πραγματοποίηση μιας σύγχρονης μηχανής αναζήτησης μικρών αγγελιών συγκεκριμένου αγαθού, στην περίπτωση μας αυτοκίνητα, η οποία θα λαμβάνει δεδομένα από πολλούς διαδικτυακούς τόπους σχετικούς φυσικά με το αντικείμενο. Σε ανάλυση και επέκταση του παραπάνω θέματος καταλήξαμε να μπούμε στον κόσμο της «Διασύνδεσης Συστημάτων» και της ανταλλαγής δεδομένων, άλλες φορές σε πραγματικό χρόνο και άλλες μετά από ανθρώπινη επέμβαση. Πρόβλημα χρόνιο αλλά και πολύ σύγχρονο όσον αφορά την επικοινωνία της πληθώρας των διαφορετικών τεχνολογιών και συστημάτων που χρησιμοποιούνται από διάφορους οργανισμούς εμπορικούς και μη.

1.2)Διασύνδεση Συστημάτων και Σύγχρονες ανάγκες 1.2

Φλέγων θέμα τα τελευταία χρόνια στον κόσμο του διαδικτύου αλλά και όχι μόνο είναι η κωδικοποίηση και διασύνδεση των διαφορετικών συστημάτων που υπάρχουν και ο τρόπος ανταλλαγής δεδομένων μεταξύ τους. Η επικοινωνία αυτή έχει επιτευχτεί με την ευρύτερη χρήση του διαδίκτυο και των δικτύων υπολογιστών με χρήση των πρωτοκόλλων TCP/IP και την μεταφορά δεδομένων μέσω HTTP. Όμως η μεταφορά δεδομένων αλλά και η κοινή τους κωδικοποίηση ώστε πολλαπλά συστήματα να μπορούν να τα «καταλάβουν» ήταν αλλά και είναι ένα μείζον πρόβλημα.

Για παράδειγμα υπάρχουν οργανισμοί που χρησιμοποιούν ακόμα πεπαλαιωμένα συστήματα εφόσον δεν έχουν ιδιαίτερες απαιτήσεις για να καλύψουν τις ανάγκες τους αλλά και το κόστος αναβάθμισης των συστημάτων αυτών είναι τεράστιο και ασύμφορο. Τι γίνεται όμως όταν οι σύγχρονες ανάγκες απαιτούν την ανταλλαγή δεδομένων με συστήματα άλλων οργανισμών ,ποιο σύγχρονα ,διαφορετικής αρχιτεκτονικής και διαφορετικού κατασκευαστή; Αν ακόμα θέσουμε και το θέμα της ασφάλειας τα πράγματα δυσκολεύουν ακόμα περισσότερο.

1.3)Έως τώρα

Μέχρι τώρα η ανταλλαγή δεδομένων πάντα ήταν ένα μείζον πρόβλημα και οι λύσεις οι οποίες χρησιμοποιούνταν ήταν βαριές αλλά και προγραμματιστικά ασύμφορες. Μεγάλοι όγκοι δεδομένων εξαγόντουσαν σε μορφή κειμένου (txt) που ανταλλασσόντουσαν μέσω **ftp** και η ακόμα και **mail** και κάθε φορά σχεδόν υπήρχε απαίτηση συγκεκριμένου και εξειδικευμένου κώδικα για την ολοκλήρωση αυτών των διαδικασιών, ώστε να παρθούν τα ζητούμενα δεδομένα, να ανακατασκευαστούν στην εκάστοτε απαιτούμενη μορφή και τέλος να χρησιμοποιηθούν από το κάθε σύστημα.

Ακόμα και στον κόσμο του Internet που αποτελεί παράδειγμα πολύ ποιο ελεύθερης αρχιτεκτονικής, οι ιστοσελίδες εμφανίζονται σε κάθε **browser** ανεξαρτήτως συσκευής – λειτουργικού συστήματος κλπ, η ανταλλαγή δεδομένων και η λεγόμενη «συλλογή δεδομένων» «Data mining» γινότανε με λύσεις μη αναγνωρισμένες ακόμα και με χρήση ανορθόδοξων τρόπων που έφταναν και στο όριο μη εξουσιοδοτημένης συλλογής και χρήσης δεδομένων όπως το κατέβασμα χιλιάδων σελίδων HTML και προσπέλαση τους μέχρι να βρεθούν τα απαραίτητα δεδομένα. Ακόμα και με συγκατάθεση του πνευματικού ιδιοκτήτη των δεδομένων αυτών κάθε πελάτης **Client** έπρεπε να κάνει τη δουλεία ξεχωριστά βρίσκοντας διάφορα κόλπα και τεχνάσματα ανάλογα με τις απαιτήσεις του. Σε περιπτώσεις βέβαια μην συγκατάθεση θα μπορούσε ο ιδιοκτήτης του **Server** να φιλτράρει τα **request** για την σελίδα του μέσω αποκλεισμού ολόκληρου block από **IP** διευθύνσεις, κάτι το οποίο θα ήταν και από τις λίγες λύσεις φιλτραρίσματος της πρόσβασης από τον ιδιοκτήτη του Server ώστε να σταματήσει η μη εξουσιοδοτημένη συλλογή δεδομένων από τον συγκεκριμένο ιστότοπο.

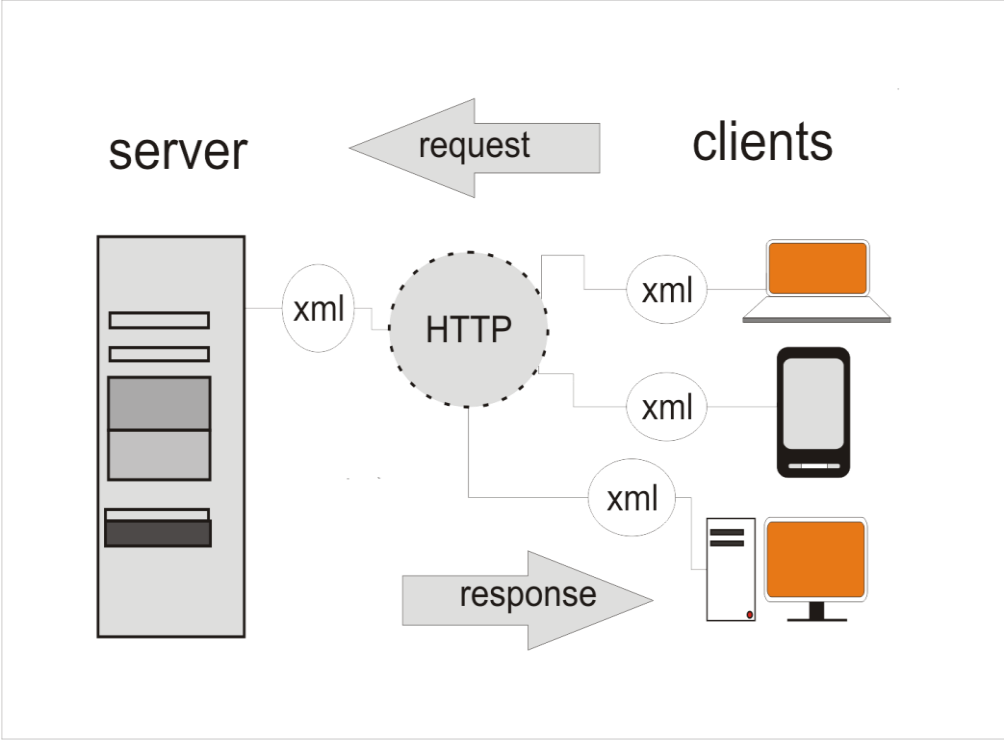
1.4)Η λύση 1.4

Από τα μέσα της δεκαετίας του 90 άρχισαν να γίνονται προσπάθειες για ενοποίηση των συστημάτων και εύρεση ενός κοινού τρόπου επικοινωνίας μεταξύ τους. Υπήρξαν προσπάθειες από πολλές μεριές να γίνει κάτι τέτοιο από τις μεγάλες εταιρίες. Αναφορικά οι προσπάθειες που έγιναν από IBM Microsoft και SUN είναι οι τεχνολογίες DCOM CORBA και JAVA RMI. Όμως υπήρξαν τελικά άλυτα προβλήματα τα οποία εμπόδισαν την ευρύτερη διάδοση τους. Κάθε εταιρία εστίασε στην χρήση τους από τα δικά τους συστήματα και γενικότερα θεωρούνταν πολύ δύσκολες στην χρήση και κατασκευή τους, επίσης χρονικά έπεσαν πάνω στην ευρεία διάδοση του διαδικτύου το οποίο στηριζόταν σε πρωτόκολλα ανοιχτού λογισμικού και ποιο χαλαρές αρχιτεκτονικές. Η ανάγκη λοιπόν παρέμεινε.

Για να επιτευχτεί η λύση σε αυτό το πρόβλημα έπρεπε να βρεθεί μια «γλώσσα» την οποία θα μιλάγανε τα περισσότερα, αν όχι άλλα τα συστήματα. Το ρόλο αυτό φαίνετε ότι επωμίστηκε η **XML.** Η XML κατάφερε να γίνει μια γλώσσα παραδεκτή από όλα τα συστήματα και τους μεγάλους κατασκευαστές λογισμικού και αν μας επιτρέπεται η έκφραση *«Παίζει το ρόλο στις γλώσσες προγραμματισμού που παίζουν τα αγγλικά στις ανθρώπινες γλώσσες».*

Οπότε χοντρικά το μόνο που θα χρειαζόταν πια κάθε σύστημα, ανεξαρτήτως κατασκευαστή η παλαιότητας είναι απλά να μπορεί να διαβάζει και να γράφει σε μορφή XML ώστε να μπορεί να ανταλλάξει δεδομένα με άλλα συστήματα.

Η συντομογραφία XMLπροέρχεται από τις λέξεις : e**X**tensive **M**arkup **L**anguage. Σε μετάφραση «επεκτάσιμη γλώσσα σήμανσης». Η XML σχεδιάστηκε δίνοντας έμφαση στην απλότητα, τη γενικότητα και τη χρησιμότητα στο διαδίκτυο. Ορίζεται κυρίως στην προδιαγραφή **XML 1.0** που δημιούργησε ο Διεθνής Οργανισμός Προτύπων **W3C (World Wide Web Consortium)** Θυμίζει αρκετά την **HTML** που είναι επίσης γλώσσα σήμανσης αλλά είναι ποιο δομημένη, είναι αναγνώσιμη και από συστήματα αλλά και από ανθρώπους και τέλος είναι επεκτάσιμη από τον χρήστη όπως αναφέρεται και στον ορισμό της. Περισσότερα στοιχεία για την XML και την χρήση της θα δούμε σε παρακάτω κεφάλαιο αφιερωμένο ειδικά στην XML και τις δυνατότητες της. Ακόμα στην XML στηρίζονται και οι **«Υπηρεσίες Ιστού» , «Web Services»** τις οποίες θα δούμε λεπτομερώς επίσης παρακάτω.



Εικόνα 1

2 Σύγχρονες Τεχνολογίες Διαδικτύου.

Παλαιότερα το μόνο που χρειαζόταν κανείς για να ασχοληθεί γενικά με το διαδικτύου ήταν η κλασική HTML και σε ανεβασμένο επίπεδο για δυναμικότητα βασικές γνώσεις PHP και MySQL. Κυρίως όμως την τελευταία δεκαετία η ευρύτερη χρήση του διαδικτύου ανέβασαν τις απαιτήσεις όσον αφορά τη λειτουργικότητα, την διαπεραστικότητα αλλά και την μορφολογία τον εφαρμογών διαδικτύου. Αυτό αυτόματος σήμαινε και την ανάπτυξη περισσότερων τεχνολογιών για την κάλυψη των στόχων αυτών.

Σε αυτό το κεφάλαιο θα αναφέρουμε με κάποιες λεπτομέρειες τις τεχνολογίες διαδικτύου που χρησιμοποιήσαμε για την επίτευξη του στόχου μας. Κρίνεται αυτό απαραίτητο εφόσον πολλές από αυτές τις τεχνολογίες μας ήταν άγνωστες πριν από το πέρας της εργασίας κάτι το οποίο μπορεί κάλλιστα να σημαίνει ότι θα είναι άγνωστες και σε μέρος των αναγνωστών του συγκεκριμένου βιβλίου πτυχιακής. Ο τρόπος ο όποιος θα αναφερθούμε σε αυτές τις τεχνολογίες είναι ονομαστικά, περιγραφικά ως προς την χρήση τους και σε ορισμένες φορές με μικρά παραδείγματα κώδικα. Σίγουρα οι αναφορές μας δεν είναι εκτενείς και ολοκληρωμένες μιας και κάτι τέτοιο δεν αποτελεί στόχο της εργασίας μας.

2.1) HTML 4

Η πλέων γνωστή «γλώσσα» σε ολόκληρο τον κόσμο του Διαδικτύου είναι η HTML και η τυποποίηση που χρησιμοποιείται ευρέως αυτήν την στιγμή είναι της HTML 4. Καθοδόν όμως βρίσκεται η ολοκλήρωση της τυποποίησης της HTML 5 που δίνει πολύ περισσότερες δυνατότητες στον χρηστή για ποιο δομημένο, εμφανίσιμο και ελαφρύ διαδίκτυο.

HTML είναι η συντομογραφία για τον ορισμό: **Hyper Text Markup Language** που θα μπορούσε να μεταφραστεί σαν «Γλώσσα σήμανσης Υπερκειμένου». Αποτελείται από δομικά στοιχεία που αναφέρονται με ετικέτες **Tags** τα οποία περιγράφουν τα βασικά στοιχεία μιας ιστοσελίδας. Όπως τις επικεφαλίδες , το κυρίως σώμα το περιεχόμενο , κλάσεις , πίνακες κλπ. Κάθε δομικό στοιχείο ανοίγει με μία ετικέτα έναρξης πχ **<h1>** και κλείνει με μια ετικέτα τέλους **</h1>.** Οι ετικέτες μπορούν να είναι η μια μέσα στην άλλη και ανάμεσα τους να τοποθετείτε το κείμενο ή η εικόνα η οποιοδήποτε άλλο στοιχείο θέλουμε να εμφανίζεται στον Browser (πρόγραμμα ανάγνωσης και αναπαράστασης ιστοσελίδων) ανάλογα με τις ιδιότητες που του προσδίδει η κάθε ετικέτα.

Παράδειγμα

```
<html><head></head>
<body>
<h1> προγραμματιστικές Εφαρμογές Στο Διαδίκτυο </h1>
<p>Οι προγραμματιστικές εφαρμογές στο διαδίκτυο είναι ένα μάθημα……</p>
</body></html>
```

Επίσης για την μορφοποίηση των ιστοσελίδων χρησιμοποιούνται και αρχεία που περιγράφουν διάφορα στυλ μορφοποίησης. Τα επονομαζόμενα και CSS.

2.2) CSS 2.0

Ευρέως για την μορφοποίηση ιστοσελίδων χρησιμοποιούνται τα λεγόμενα CSS συντομογραφία για το Cascading Style Sheets «Φύλλα Διαδοχικών Στυλ».

Αυτά τα «Φύλλα» λοιπόν είναι ξεχωριστά αρχεία με κατάληξη (*.css) τα οποία συνήθως με ένα σύνδεσμο στην σελίδα της γλώσσας σήμανσης (HTML , XML κλπ) καθορίζουν τον τρόπο εμφάνισης των στοιχείων των διαφόρων ετικετών και των περιεχομένων τους. Έτσι με ένα μόνο αρχείο Css μπορούμε να διαμορφώσουμε πάνω από μία σελίδες. Μπορούν να είναι γραμμένα και μέσα στην κάθε σελίδα αλλά γενικότερα αυτό είναι ασύμφορο όσον αφορά την επαναχρησιμοποίηση τους.

Η σύνταξη των CSS είναι απλή.

Ξεκάνει με τον επιλογέα **Selector** της ετικέτας της γλώσσας μορφοποίησης, και ακλουθούν οι δηλώσεις. Κάθε δήλωση αποτελείται από ένα **property (**ιδιότητα**)** και από μία αξία αυτής (**value**). Ιδιότητες υπάρχουν πολλές όσον αφορά την εμφάνιση του εκάστου στοιχείου. Αυτές οι ιδιότητες θα μπορούσαν να χωριστούν σε ομάδες ανάλογα με το τι επηρεάζει η κάθε μια. Υπάρχουν οι **ιδιότητες κειμένου**(ορίζουν γραμματοσειρά, μέγεθος γραμμάτων, εμφάνιση γραμμάτων κλπ) οι **ιδιότητες φόντου** (ορίζουν το χρώμα του φόντου, κάποια εικόνα, που θα εμφανίζεται κλπ) οι **ιδιότητες στοίχισης** (απόσταση μεταξύ γραμμάτων κατεύθυνση κειμένου, απόσταση γραμμών κ.α) οι **ιδιότητες θέσης** (στοίχιση ετικέτας αριστερά δεξιά, απόσταση από άλλες ετικέτες, εσωτερική περιθώριο κ.α) **ιδιότητες ορίων** (χρώμα ορίου ετικέτας, πάχος, στυλ κλπ) **ιδιότητες λιστών** (θέση λίστας , εικονίδιο λίστας κλπ) **ιδιότητες τοποθέτησης στο έγγραφο** (συντεταγμένες, κλείδωμα θέσης κλπ).

Ο επιλογέας αυτός μπορεί να περιγραφεί μια υπάρχουσα ετικέτα της HTML πχ **body{},** να δίνει μια ταυτότητα επιλογέα πχ **#main{}** που χρησιμοποιείται συνήθως από μια ετικέτα της html που την ονομάζουμε **device** πχ**. ** αντιπροσωπεύει ένα μοναδικό αντικείμενο, η μία κλάση η οποία μπορεί να χρησιμοποιηθεί αρκετές φορές μέσα σε ένα έγγραφο ορίζοντας παραπάνω από ένα στοιχεία. Που ορίζεται σαν πχ **.red{color:red;}** και χρησιμοποιείτε σαν ****

Παράδειγμα

```
h2
{
float:right;
color:white;
line-height:2ex;
}



#wrap
{
width: 1030px;
background: white;
```

```
padding-bottom:20px;
}
.gray
{
font-size:12pt;
border:black thin solid;
}
```

Ο ευρύτερος ορισμός των χρησιμοποιούμενων CSS 2.0 και των selectors (επιλογέων) του αποτελεί άλλο ένα από τα standard του W3 consortium. Ακόμα και τώρα βέβαια άσχετος αν η χρήση τους είναι δεδομένη και θα μπορούσαμε να πούμε απαραίτητη για τα σύγχρονα στάνταρ του ιντερνέτ προβλήματα ποικίλουν ανάμεσα στην εμφάνιση τους από διάφορους Browsers. Σαν ποιο συνεπείς Browsers όσον αφορά τουλάχιστον τα CSS 2.0 φαίνονται να είναι ο Chrome της Google και Ο Firefox του ιδρύματος Mozilla. Βραβείο φυσικά ασυμβατότητας παίρνει ο Internet Explorer της Microsoft και ειδικά όσον αφορά παλαιότερες εκδόσεις του.

Τέλος καινούργιοι Selectors που βγαίνουν όσον αφορά την τυποποίηση CSS 3.0 δίνουν ακόμα περισσότερες δυνατότητες για μορφοποίηση των ιστολογιών από τους χρήστες (όπως στρογγυλοποιημένες γωνιές, σκιές κλπ) αλλά δεν είναι ακόμα σε ώριμο σημείο για ευρύτερη χρήση ειδικά όσον αφορά το Θέμα “Browser Independent” δλδ να υπάρχει ίδια μετάφραση από όλους τους Browsers, θέμα που φαίνετε έχει επωμιστεί ο οργανισμός W3C.

2.3) *AMP(Apache , MySql, Php)

Πριν μιλήσουμε εκτενώς για την γλώσσα PHP σαν απαραίτητο εργαλείο του διαδικτύου και σαν κύριο εργαλείο υλοποίησης της εργασίας αυτής αναγκαία είναι μια σύντομη αναφορά στα ανοιχτό πακέτα λογισμικού τύπου *AMP και στον Apache HTTP Server και στην MySql.

Το πρώτο γράμμα αυτού του πακέτου λογισμικού ποικίλει ανάλογα με το λειτουργικό σύστημα για το οποίο προορίζεται. Έτσι έχουμε Lamp για Linux, Mamp για Macintosh, Wamp για Windows και Xamp για όλα τα παραπάνω. Το πακέτο αυτό δεν είναι κάτι περισσότερο από μια ενοποιημένη λύση που περιέχει τα τρία αυτά λογισμικά ανοιχτού κώδικα ώστε να λειτουργήσει ένας Server. Φυσικά μπορούν να εγκατασταθούν και ξεχωριστά αλλά αυτό είναι πολύ ποιο χρονοβόρο ανάλογα φυσικά με την χρήση για την οποία προορίζεται ο κάθε Server.

Ο **Apache HTTP Server** που χάριν συντομίας αναφέρεται σαν Apache είναι από τα ποιο διαδεδομένα λογισμικά Server αυτήν την στιγμή στο Internet. Είναι λογισμικό ανοιχτού κώδικα που δημιουργήθηκε και συντηρείτε από μια «ανοιχτή κοινωνία» προγραμματιστών. Ρόλος ενός Server (εξυπηρετητής) είναι να αναμένει αιτήσεις από διάφορα προγράμματα χρήστες (Clients) όπως είναι ένας browser και να εξυπηρετεί (σερβίρει) τις ζητούμενες από τις αιτήσεις σελίδες σύμφωνα με τα πρότυπα που ορίζει το πρωτόκολλο HTTP (Hyper Text Transfer Protocol). Είναι λογισμικό απαραίτητο για την

λειτουργιά **server side** γλωσσών προγραμματισμού όπως η **Php** η **Python** και **Perl** απαραίτητες για την επικοινωνία ιστοσελίδων με διεργασίες, από μεριάς Server όπως θα μπορούσαμε να πούμε, (πχ επικοινωνία με βάσεις δεδομένων, εξαγωγή και εισαγωγή στοιχείων από αυτές κλπ)

Η **MySQL** είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων που παρέχει πολλαπλή πρόσβαση σε διάφορους χρήστες και χρησιμοποιείτε ευρέως στο χώρο του διαδικτύου. Είναι επίσης λογισμικό ανοιχτού κώδικα που υπόκειται στο GNU General Public License που είναι μια ευρύτερη άδεια χρήσης για λογισμικά ανοιχτού κώδικα ακολουθώντας μια συγκεκριμένη φιλοσοφία για την απόκτηση χρήση και τροποποίηση αυτών.

2.4) Η ΓΛΩΣΣΑ PHP

Τι είναι η php; Η **php** είναι μία γλώσσα προγραμματισμού που χρησιμοποιείται ευρέος για την δημιουργία ιστοσελίδων με δυναμικό περιεχόμενο και είναι γνωστή σαν **hypertext preprocessor** και έχει σαν βάση την γλώσσα C**.** Είναι επίσης ανοιχτού κώδικα γλώσσα και έχει γίνει απαραίτητο εργαλείο για κάθε web προγραμματιστή. Επίσης έχει αποδεδειχθεί και σαν γλώσσα «κόλα» (glue language) για την διασύνδεση συστημάτων μέσω WEB. Μπορεί να τοποθετηθεί μέσα σε σελίδες HTML (αλλάζοντας όμως την κατάληξη από *.html σε *.php) παράγει περιεχόμενο μέσω αλληλεπίδρασης με τον server.

Ο κώδικας php ανοίγει με τα σύμβολα “**<?php”** και κλείνει με το σύμβολο “**?>”.**

Παραδειγμα αρχείου .php

```
<html>
        <body>
                <?php echo “this is a page”; ?>
        </body>
</html>
```

Δυναμικό περιεχόμενο σε αντίθεση με το στατικό ορίζεται αυτό το οποίο παράγεται μετά από επεξεργασία μιας php σελίδας από ένα διακομιστή **(server)** όπως είναι ο **Apache** και επιστρέφει μετά από τις ζητούμενες εργασίες στο πρόγραμμα περιήγησης **(browser)** έξοδο σε μορφή html. Η γλώσσα PHP είναι γλώσσα “**Server Side Προγραμματισμού”** δλδ παράγει αποτελέσματα μετά από αλληλεπίδραση με τον server. Γι αυτό και είναι απαραίτητη σε αρκετά σημεία να γίνει αποστολή στο server κάποιον στοιχείων για να πάρουμε αποτελέσματα. Αυτό θα δούμε ότι είναι ένα πρόβλημα το οποίο λύνεται με χρήση άλλων τεχνολογιών που είναι γλώσσες **Browser Side Προγραμματισμού.**

Ο Apache web server υποστηρίζει εξορισμού τον κώδικα php έτσι και η php διαθέτει συναρτήσεις εξορισμού για αλληλεπίδραση με βάσεις δεδομένων **Mysql** , χρήση της **XML** , επεξεργασία αρχείων(**files**),συναρτήσεις για αλληλεπίδραση φορμών που χρησιμοποιούνται από τον χρήστη και παρέχει πληθώρα άλλων λειτουργιών.

Παράδειγμα χρήσης με φόρμες.

```
<form action="action.php" method="post">
 <p>Your name: <input type="text" name="name" /></p>
 <p>Your age: <input type="text" name="age" /></p>
```

```
 <p><input type="submit" /></p>
</form>
```

Και διάβασμα δεδομένων από φόρμες

```
Hi <?php echo htmlspecialchars($_POST['name']); ?>.
You are <?php echo (int)$_POST['age']; ?> years old.
```

Περισσότερα παραδείγματα χρήσης PHP θα δούμε στην πορεία όταν μελετάμε την κατασκευή των ιστοσελίδων μας.

Αναφορικά (σαν παράδειγμα της διαδεδομένης χρήσης της php) τα περισσότερα ευρέως διαδεδομένα «έτοιμα» εργαλεία λογισμικού διαδικτύου και διαχείρισης περιεχομένου χρησιμοποιούν σαν κυρίως μέρος την PHP. Τέτοια εργαλεία είναι Το Joomla, Wordpress, Prestashop κ.α

2.5) JavaScript, J query, Ajax ,J son

2.5.1) JavaScript

Καταρχάς όπως συνηθίζεται να λέγεται σε κάθε tutorial για την javascript **"Η javascript δεν είναι java".** Είναι μια εντελώς καινούργια γλώσσα που κατασκευάστηκε από την Netscape και αρχικά ονομαζόταν LiveScript. Μετονομάστηκε σε Javascript καθαρά για θέματα marketing –και απ' ότι φαίνεται πέτυχε μιας και δεν είναι απλά διαδεδομένη στον χώρο ρου διαδικτύου αλλά αποτελεί ένα από τα στάνταρ του αυτής της εποχής.

Η javascript είναι μια γλώσσα προγραμματισμού η όπως την ονομάζουν περισσότερο μια "scripting language" η οποία χρησιμοποιείτε κυρίως για να προσθέσει διαπεραστικότητα σε μια σελίδα όσον αφορά την αλληλεπίδραση με τον χρήστη. Σε αντίθεση με την PHP και άλλες παρόμοιες γλώσσες είναι "browser side" και όχι "server side" που σημαίνει ότι οι αλλαγές που γίνονται γίνονται στον browser του χρήστη και δεν απαιτείται αλληλεπίδραση με τον server.

Κομμάτια κώδικα της javascript μπορούν να συμπεριλαμβάνονται σε μια σελιδα html η να «φορτώνονται από ξεχωριστό αρχείο με ένα σύνδεσμο. Προτιμάται κυρίως η εξωτερική τους τοποθέτηση για λόγους δομής και επαναχρησιμοποίησης κώδικα.

Η javascript είναι στην κατηγορία γλωσσών που ονομάζουμε object oriented language. Αυτό σημαίνει ότι επεξεργάζεται όλα τα στοιχεία σαν αντικείμενα ακολουθώντας το DOM. Το DOM στην ουσία του είναι μια φιλοσοφία ερμηνείας μιας ιστοσελίδας , ενός αρχείου, ενός έγγραφου XML κ.ο.κ. η οποία βλέπει τα πάντα σαν αντικείμενα. Τα αντικείμενα αυτά φυσικά μπορούν να περιέχουν αλλά αντικείμενα, να έχουν γονείς και παιδιά πολλές φορές περισσότερα από ένα και κάθε ένα από αυτά να είναι ένα ξεχωριστό αντικείμενο. Για να το καταλάβουμε καλύτερα όλο αυτό το μοντέλο πρέπει να ξαναφέρουμε στο μυαλό μας την δομή του δέντρου που χρησιμοποιούμε στα συστήματα φακέλων. Από την μια και

μοναδική ρίζα μερί το τελευταίο φύλλο. Στην javascript η προσπέλαση των αντικειμένων γίνεται με τον τελεστή «.» της τελείας.

Παράδειγμα

```
document.body.p
document.title
document.form.submit
```

Κάθε αντικείμενο ακολουθεί με την σειρά του το μοντέλο PME –Properties, Methods, Events- Δηλαδή –Ιδιότητες , Μέθοδοι, Συμβάντα- . Το μοντέλο αυτό μας είναι ήδη γνωστό από τον Οπτικό Προγραμματισμό της Γλώσσας C++. Εν ολίγης κάθε αντικείμενο όπως και στον πραγματικό κόσμο έχει ιδιότητες, έχει κάποιες «λειτουργίες» και μπορούν να γίνουν με αυτό κάποια συμβάντα. Μερικά από αυτά τα συμβάντα προέρχονται και από τον χρήστη και είναι αυτά που εκμεταλλευόμαστε για να κάνουμε τις αλλαγές στα στοιχεία της ιστοσελίδας μας και να της δώσουμε αυτό που ονομάζουμε σήμερα «διαπεραστικότητα» η «αλληλεπίδραση με τον χρήστη», αλλά πάλι προέρχονται από διαδικασίες.

Παράδειγμα

```
Button1.click();
Window.open();
```

Κυρίως λοιπόν η JavaScript χρησιμοποιείται για: -επαλήθευση δεδομένων που εισάγονται σε φόρμες, -κατασκευή διαδραστικών μενού επιλογών “popup menu”, -δυναμική διαμόρφωση και μορφοποίηση της Html μας αλληλεπίδραση με τον χρήστη.

Υπάρχουν και βέβαια πράγματα τα οποία η JavaScript δεν μπορεί να κάνει. Δεν μπορεί να έχει αλληλεπίδραση με βάση δεδομένων, δεν μπορεί να γράψει σε αρχεία, δεν μπορεί να δημιουργήσει και να χρησιμοποιήσει κάτι παραπάνω από Cookies. Για όλες αυτές τις λειτουργίες χρειαζόμαστε μια “server side” γλώσσα όπως είναι η PHP.

Θα μπορούσαμε να αναφερθούμε εκτενώς στην JavaScript άλλα θα χάναμε την ουσία της παρούσας εργασίας. Αυτό το οποίο θα προσθέσουμε ακόμα είναι ότι έκτος από τα στοιχεία της έκαστης ιστοσελίδας και τον τρόπο της εμφάνισης τους στον Browser μπορεί να διαχειριστεί μεταβλητές, πράξεις, βρόγχους, ακόμα και να ελέγξει κατά κάποιο τρόπο τους browsers που καλούνται να εμφανίσουν την σελίδα.

2.5.2) J query

Υπάρχει ένα μπέρδεμα όσον αφορά το J query. Το J query δεν είναι γλώσσα προγραμματισμού. Το J query είναι μια ανοιχτού κώδικα βιβλιοθήκη γραμμένη σε Javascript. Και όπως κάθε βιβλιοθήκη πρέπει να συμπεριληφθεί με έναν σύνδεσμο στο αρχείο μας (εδώ στην Html σελίδα μας) προκειμένου να μπορέσουμε να χρησιμοποιήσουμε τις συναρτήσεις της. Αυτό το οποίο όμως έχει κάνει το jquery τόσο γνωστό στους κύκλους του διαδικτύου είναι η μεγάλη του απλότητα και χρηστικότητα όσον αφορά κοινές,

επαναλαμβανόμενες προγραμματιστικές εργασίες, εύκολο συντακτικό όσον αφορά την χειραγώγηση CSS, δίνει ευκολία στην επεξεργασία πολλών στοιχείων ταυτόχρονα και να παράγει «αλυσίδα λειτουργιών».

Ένα από τα κυριότερα οφέλη είναι ότι έχει πάρα πολλά έτοιμα plugins για την προσθήκη εφέ αλλά και άλλων λειτουργιών τα οποία μπορούν να τροποποιηθούν κατά βούληση και να χρησιμοποιηθούν σε κάθε κώδικα. Ένα από αυτά θα δούμε παρακάτω στην υλοποίηση της εργασίας μας το οποίο μας βοήθησε να λύσουμε ένα αρκετά «ενοχλητικό» θέμα που συναντήσαμε.

2.5.3) Ajax

Η λέξη Ajax προέρχεται από τα αρχικά Asynchronous JavaScript and XML. Δλδ Ασύγχρονη JavaScript και XML. Στην ουσία είναι ένα σύνολο μεθόδων το όποιο χρησιμοποιείται από τον Client (και όχι από το Server) για την κατασκευή ασύγχρονων εφαρμογών ιστού. Εν ολίγοις οι εφαρμογές αυτές με την χρήση της τεχνολογίας Ajax (στην ούσα σύνολο τεχνολογιών) μπορούν να λάβουν και να στείλουν δεδομένα σε ένα Server ασύγχρονα, δηλαδή στο παρασκήνιο, χωρίς να επηρεάζουν την εμφάνιση και την συμπεριφορά της υπάρχουσας σελίδας. Τα δεδομένα συνήθως λαμβάνονται χρησιμοποιώντας το XMLHttpRequest αντικείμενο. Βέβαια παρά το όνομα εδώ η χρήση της XML δεν είναι απαραίτητη μιας και χρησιμοποιείται ποιο συχνά το **J son** όπως επίσης και η αλληλεπίδραση με τον Server δεν χρειάζεται πάντα να γίνεται ασύγχρονα.

Με την χρήση λοιπόν του συνόλου των τεχνολογιών αυτών (Javascript XML JSON HTML CSS ακόμα και PHP) και πάντα με την χρήση του DOM τα δεδομένα εμφανίζονται μέσω JavaScript και ο χρήστης μπορεί να αλληλεπιδρά με αυτά χωρίς να είναι απαραίτητη η επαναφώρτοση ολόκληρης της σελίδας (όπως γίνεται με την PHP και την λειτουργία Submit τον φορμών)

Υπάρχουν πολλά σύνολα βιβλιοθηκών, η καλύτερα JavaScript Frameworks τα οποία χρησιμοποιούνται για την παραγωγή εφαρμογών Ajax. Τα ποιο γνωστά από αυτά είναι το Spry Framework της Adobe που συμπεριλαμβάνεται στο Dreamweaver και χρησιμοποιήσαμε εκτενώς για επιβεβαίωση φορμών και άλλα controls στην εφαρμογή μας, Το jQuery που εκτός από Javascript framework παρέχει και Ajax Framework το οποίο χρησιμοποιήσαμε στην μηχανή αναζήτησης μας για λήψη δεδομένων χωρίς την ανάγκη να φορτωθεί όλη η σελίδα από την αρχή. Επίσης υπάρχουν προεκτάσεις για Ajax στο .NET framework το οποίο χρησιμοποιεί το Visual Studio της Microsoft και χρησιμοποιήσαμε για την κατασκευή του Desktop άλλα και του Mobile Application.

2.5.4) J SON

Το J SON είναι ένας τρόπος αναπαράστασης δεδομένων το οποίο σχεδιάστηκε για την αναπαράσταση δεδομένων σε μορφή αναγνώσιμη από μηχανές αλλά και από ανθρώπους. Παράγεται με την Javascript για την αναπαράσταση απλών δομών δεδομένων και συσχετισμένων πινάκων τα οποία ονομάζονται αντικείμενα. Παρ' όλη την σχέση του με την Javascript είναι ανεξάρτητο γλώσσας μιας και οι περισσότερες γλώσσες διαθέτουν την δυνατότητα γραφής και ανάγνωσης σε μορφή JSON, μια από αυτές είναι και η PHP.

Χρησιμοποιείται συνήθως για την μετάδοση μορφών δεδομένων σε περιβάλλοντα δικτύου και κυρίως για την ανταλλαγή δεδομένων ανάμεσα σε εφαρμογές διαδικτύου και Servers σαν εναλλακτική της XML.

Παραδειγμα

```
{
"employees": [
{ "firstName":"John" , "lastName":"Doe" },
{ "firstName":"Anna" , "lastName":"Smith" },
{ "firstName":"Peter" , "lastName":"Jones" }
]
}
```

2.6 Η γλώσσα XML

2. 6.1) Τι είναι η XML;

Όταν μιλάμε ποια για ενοποιημένες εφαρμογές και ανταλλαγή δεδομένων είτε μέσω διαδικτύου είτε γενικότερα μεταξύ διαφόρων εφαρμογών, εταιριών, οργανισμών, μιλάμε για την γλώσσα XML.

Η XML είναι μια σημασιολογική γλώσσα η οποία μοιάζει αρκετά με την HTML. Δεν είναι γλώσσα προγραμματισμού και δεν «κάνει» κάτι. Είναι απλά μια μορφή περιγραφής και αποθήκευσης δεδομένων. Επίκεντρο της XML είναι η μεταφορά η δομή και αποθήκευση δεδομένων με βαρύτητα στο τι είναι αυτά τα δεδομένα αντίθετα η HTML ασχολείται με το πώς εμφανίζονται τα δεδομένα αυτά στο browser. Βέβαια εφόσον η SQL και η χρήση της είναι πανταχού παρόν δεν φαίνεται ότι μπορεί να παίξει το κύριο μέρος ειδικά στην αποθήκευση δεδομένων αλλά κυρίως στην ενοποίηση και μεταφορά τους.

Χρησιμοποιείται σε πολλές μορφές από κάθε τύπου WEB εφαρμογές ή και Desktop και απλοποιεί την αναπαράσταση, το διαμοιρασμό και την μεταφορά δεδομένων μεταξύ εφαρμογών. Το ζήτημα της πολυπλοκότητας λοιπόν της ανταλλαγής δεδομένων μεταξύ εφαρμογών που είναι ασύμβατες μπορεί να λυθεί με την αναπαράσταση δεδομένων σε μορφή XML. Εκατοντάδες εφαρμογές ανεξαρτήτως λειτουργικού και κατασκευαστή, και όλες οι γλώσσες προγραμματισμού έχουν συναρτήσεις για την ανάγνωση και εγγραφή σε αρχεία μορφής XML. Μελλοντικά λοιπόν φαίνεται ότι η XML θα είναι μέσα σε κάθε εφαρμογή, ιστού και μη για αναπαράσταση και ανταλλαγή δεδομένων χωρίς πολλές ενδιάμεσες λειτουργιές.(αν και ειδή ισχύει κάτι τέτοιο). Επίσης στην XML βασίζονται και οι **Υπηρεσίες Ιστού** που είναι και το κύριο θέμα της Εργασίας μας και θα δούμε αναλυτικα στο 3ο μέρος. Ακόμα και πολλές γλώσσες για το χώρο του internet δημιουργούνται με την χρήση της XML, μία από αυτές θα δούμε παρακάτω όταν μιλήσουμε για το SOAP και το WSDL.

2.6.2) Πώς συντάσσεται ένα XML έγγραφο;

Το X στην XML προέρχεται από την λέξη e**X**tensible που σημαίνει επεκτάσιμη σημασιολογική γλώσσα. Τι σημαίνει επεκτάσιμη. Στην XML δεν υπάρχουν έτοιμες ετικέτες να θυμάται ο χρήστης ανάλογα με το τι θέλει να αποθηκεύσει. Κάθε χρήστης μπορεί να δημιουργήσει τις δικιές του ετικέτες ακολουθώντας ένα πολύ απλό τρόπο σύνταξης. Με το γνωστό σύμβολο ανοίγει μια ετικέτα «<>» και κλείνει «</>» ακριβώς σαν την HTML (μόνο που στην XML οι κανόνες σύνταξης είναι ποιο αυστηροί). Ανάμεσα μπαίνει και το επιθυμητό κείμενο, ημερομηνία, αλφαριθμητικό νούμερο κοινώς οποιαδήποτε μορφή δεδομένου.

```
<person>
        <name>Alex</name>
        <surname>Jameson</surname>
        <phone>6955654854</phone>
        <born>25/12/1985</born>
</person>
```

Το μόνο που είναι απαραίτητο είναι ένα αρχικό στοιχείο που περιλαμβάνει όλα τα υπόλοιπα στοιχεία. Το root element όπως ονομάζεται.

```
<people>
   <person>
        <name>Alex</name>
        <surname>Jameson</surname>
   </person>
   <person>
        <name>John</name>
        <surname>Doe</surname>
   </person>

   <person>
        <name>Nick</name>
        <surname>Thomson</surname>
   </person>
</people>
```

Οι ετικέτες της XML μπορούν επίσης να περιέχουν και χαρακτηριστικά “attributes” τα οποία παρέχουν περισσότερες πληροφορίες για τα δεδομένα μπορούν να περιέχουν και διάφορες τιμές. Οι τιμές αυτές πάντα πρέπει να είναι εντός διπλού εισαγωγικού. «**””**».

```
<toys>
   <toy id=”2554”>Lego Castle </product>
   <toy id=”3854”>Pokémon balloon</product>
   <toy id=”9684”>Blue Racing Car</product>
</toys>
```

Τα χαρακτηριστικά στην XML δεν παίζουν τον ρόλο που παίζουν στην HTML και από πολλές απόψεις δεν θεωρούνται σημαντικά αφού ότι έχει να πει κάποιος μπορεί να το πει με ένα παραπάνω στοιχείο.

```
<toys>
 <toy>
   <toy_id>2554</toy_id>
   <toy_description>Lego Castle</toy_description>
 </toy>
 <toy>
   <toy_id>3854</toy_id>
   <toy_description> Pokémon balloon </toy_description>
 </toy>
 <toy>
   <toy_id>9684</toy_id>
   <toy_description> Blue Racing Car </toy_description>
 </toy>
</toys>
```

Τα XML έγγραφα όσον αφορά την κωδικοποίηση και την έκδοση τους δηλώνονται εσωτερικά με μια «επικεφαλίδα».

```
<?xml version=”1.0” encoding=”UTF-8”?>
```

Για παράδειγμα ένα ολοκληρωμένο έγγραφο XML έχει την παρακάτω μορφή.

```
<?xml version=”1.0” encoding=”UTF-8”?>

<!-- Comments are like HTML -->
<!-- This file describes people -->

<people>
   <person>
        <name>Alex</name>
        <surname>Jameson</surname>
        <phone num=”1”>2105458569</phone>
        <phone num=”2”>6956969658</phone>
   </person>
   <person>
        <name>John</name>
        <surname>Doe</surname>
        <phone num=”1”>2105658548</phone>
   </person>

   <person>
        <name>Nick</name>
        <surname>Thomson</surname>
   </person>
</people>
```

2.6.3) XML namespaces

Τι είναι τα “namespaces”; Πριν πούμε τι είναι τα namespaces πρεπει να αναφερθούμε σε ένα θέμα που προκύπτει από την «ελευθερία» της XML που παρέχει στον χρήστη για την δημιουργία δικών του ετικετών, την πιθανότητα ίδιας ονομασίας σε μια ετικέτα διαφορετικής σημασίας. Παράδειγμα η ετικέτα <id/> σε ένα κατασκευαστή που ανταλλάσει δεδομένα με έναν πολίτη ο οποίος αναφέρεται σε μια ετικέτα <id/>. Σε

συνδυασμό αυτών τον εγγράφων θα υπάρχει θέμα διένεξης στο ίδιο όνομα ετικέτας και ένας **“parser”** (αναλυτής εγγράφου) δεν θα μπορεί να καταλάβει την διαφορά. Αν αυτό το σκεφτούμε στην κλίμακα χιλιάδων εφαρμογών και ελευθέρων XML εγγράφων ανά τον κόσμο μιλάμε για ένα χάος.

Η λύση σε αυτό το θέμα είναι τα «προθέματα» πχ. <maker:id> και <sales:id>. Όταν χρησιμοποιούμε προθέματα στην XML πρέπει να ορίσουμε για κάθε πρόθεμα ένα **namespace.** Το namespace ορίζεται από ένα χαρακτηριστικό το οποίο ονομάζεται **xmlns** και ορίζεται στην αρχική ετικέτα ενός στοιχείου. Η δήλωση ενός namespace ακολουθεί την ακόλουθη σύνταξη **«xmlns:(πρόθεμα)=”URI”»**

```
<maker:id xmlns:maker="http://www.furnituremakers.org">
<sales:id xmlns:sales="http://www.megamarket.com/furniture">
```

Δηλώνοντας φυσικά στην αρχή του εγγράφου τα namespaces δεν χρειάζεται να τα επαναλαμβάνουμε για κάθε στοιχείο. Αν και τα URI τα οποία δηλώνουμε δεν διαβάζονται από τους parsers αλλα χρησιμοποιούνται περισσότερο για λόγους ταυτοποίησης υπάρχουν σελίδες που πραγματικά παρέχουν πληροφορίες για την μορφολογία η και την δομή ενός XML εγγράφου. Στην πραγματικότητα χρησιμοποιούνται για να καθορίσουν κάποια μορφοποίηση του εγγράφου XSLT και στην κατασκευή XML Schema η XSD.

Αν και είναι πιθανή η μορφοποίηση ενός εγγράφου XML με τα γνωστά μας CSS Το XSLT είναι αυτό το οποίο προτείνεται και χρησιμοποιείται ευρέος για την μορφοποίηση XML. Αναφορικά είναι και αυτό γραμμένο σε XML αλλά δεν θα σταθούμε περισσότερο στο θέμα αυτό.

2.6.4)XML Schema Definition - XSD και επικύρωση ενός XML έγγραφου.

Το XML Schema Definition είναι αυτό που θα λέγαμε ο διάδοχος τον DTD (Document Type Definition) όσον αφορά την XML. Τα DTD στην ουσία είναι έγγραφα τα οποία καθορίζουν τα «νόμιμα» στοιχεία και την «νόμιμη» δομή ενός XML. Μπορούν να είναι εσωτερικά μέσα σε ένα XML η να αναφέρονται εξωτερικά. Υπάρχουν όμως κάποια μειονεκτήματα ή καλύτερα κάποια πλεονεκτήματα στην χρήση των XSD αντί των DTD τα οποία τα καθιστάν ποιο σύγχρονα , ακριβή, και κατάλληλα για την χρήση τους με XML. Για αρχή είναι γραμμένα σε XML, δεύτερον υποστηρίζουν περιορισμούς αλλά και τύπους δεδομένων, (αν το περιεχόμενο είναι ακέραιος ή αριθμός κινητής υποδιαστολής, αν είναι αλφαριθμητικό η bool η uri, επίσης αν τα ψηφία η οι χαρακτήρες έχουν περιορισμού στο πλήθος τους). Αν αναλογιστούμε πως είναι δομημένες και κατασκευασμένες οι βάσεις δεδομένων τότε σίγουρα τα XSD είναι το εργαλείο που χρειαζόμαστε ειδικά στην περίπτωση που τα δεδομένα ενός XML αρχείου πρόκειται να αποθηκευθούν σε μια βάση δεδομένων η να αλληλεπιδράσουν με στοιχεία της.

Χαρακτηριστικά δίνουμε το XSD που χρησιμοποιήσαμε στην εφαρμογή μας για να ορίσουμε την μορφή των XML αρχείων τα οποία δέχεται η μηχανή αναζήτησης μας.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <!-- required elements -->
```

```
<!-- add_code-->
<xs:element name="add_code">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:maxLength value="10"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

<!--product url on your site-->
<xs:element name="url">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="200"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

<!--product's image url on your site-->
<xs:element name="imgurl">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="300"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

<!--maker of the car-->
<xs:element name="maker">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="20"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

<!--car Model-->
<xs:element name="model">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="20"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

<!--price in euro-->
<xs:element name="price">
  <xs:simpleType>
    <xs:restriction base="xs:float"/>
  </xs:simpleType>
</xs:element>

<!--engine size-->
<xs:element name="cc">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:maxLength value="6"/>
    </xs:restriction>
  </xs:simpleType>
```

```
  </xs:element>

  <!--horse power of the engine-->
  <xs:element name="bhp">
    <xs:simpleType>
      <xs:restriction base="xs:integer">
        <xs:maxLength value="4"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>

  <!--killometers runned-->
  <xs:element name="km">
    <xs:simpleType>
      <xs:restriction base="xs:int">
        <xs:maxLength value="10"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <!--year of first cirqulation -->
  <xs:element name="year">
    <xs:simpleType>
      <xs:restriction base="xs:gYear">
        <xs:maxLength value="4"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>

  <!--color of the car -->
  <xs:element name="color">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="20"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <!---->
  <!-- Description Of The Car -->
  <!--The Description Must Be in Order of
  maker, model, price, cc, bhp, km, year, color-->
  <xs:element name="description">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="200"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>

</xs:schema>
```

Ένα παράδειγμα ενός XML αρχείου το οποίο ακολουθεί τους παραπάνω κανόνες.

```
  <?xml version="1.0" encoding="UTF-8"?>
<cars>
<car>
<add_code>69</add_code>
<imgurl>http://localhost/carz/images/noimage.png</imgurl>
<url>http://localhost/carz/public/searched.php?add=69</url>
```

```
<maker>Alfa Romeo</maker>
<model>Alfa 145</model>
<price>5600</price>
<cc>2000</cc>
<bhp>98</bhp>
<km>20000</km>
<year>2008</year>
<color>Brown</color>
<description>Alfa Romeo Alfa 145 654 2000 100 20000 2008
Brown</description>
</car>
<car>
<add_code>70</add_code>
<imgurl>http://localhost/carz/images/bmw.png</imgurl>
<url>http://localhost/carz/public/searched.php?add=70</url>
<maker>Alfa Romeo</maker>
<model>Alfa 147</model>
<price>2500</price>
<cc>1800</cc>
<bhp>105</bhp>
<km>20000</km>
<year>2001</year>
<color>Black</color>
<description>Alfa Romeo Alfa 147 2500 1800 105 20000 2001
Black</description>
</car>
</cars>
```

Με την χρήση ενός X S D μπορεί ένας οργανισμός η μια εταιρία η μια εφαρμογή μπορεί να καθορίσει την δομή αλλά και τα χαρακτηριστικά των δεδομένων τα οποία θα αναγράφονται στα XML τα οποία δέχεται και μπορεί πολύ απλά να το τοποθετήσει σε μια δημόσια θέση ώστε να το κοινοποιήσει στους μελλοντικούς η και υπάρχοντες συνεργάτες της. Επίσης μπορεί κάποιος να επικυρώσει και το XML που παράγει σε σχέση με ένα XSD με διάφορα εργαλεία ακόμα και online.

2.6.5) PHP και XML

Η PHP έρχεται με ήδη κατασκευασμένη με επεκτάσεις για χρήση XML οι οποίες συντάσσονται αυτομάτως κατά την εγκατάσταση της PHP, οπότε δεν χρειάζεται περεταίρω ρυθμίσεις. Αυτό μπορούμε να το δούμε και μέσα από την συνάρτηση **phpinfo();**. Με τις επεκτάσεις της PHP για την XML μπορούμε να διαβάσουμε ένα αρχείο XML και να κάνουμε διαδικασίες στους κόμβους και τα δεδομένα του εγγραφου. Στην PHP 5.0 θα βρούμε τρία ήδη εγκατεστημένα API (Application Protocol Interface) για να χρησιμοποιήσουμε και να «αναλύσουμε» ένα έγγραφο XML. Το **DOM**, το **SAX** και το **Simple XML** parser. Φυσικά κάθε επέκταση έχει τα μειονεκτήματα αλλά και τα πλεονεκτήματα της και ενδείκνυται για διαφορετικές χρήσης όσον αφορά ένα XML έγγραφο(δημιουργία, ανάγνωση, αλλαγή).

Το **DOM** API για την PHP είναι το ήδη γνωστό μας **D**ocument **O**bject **M**odel το οποίο διαβάζει ένα αρχείο XML και δημιουργεί στην μνήμη ένα αντικείμενο σε μορφή δέντρου το οποίο μπορεί να προσπελαστεί διαβάζοντας τους κόμβους του και τα πεδία καθώς και το

κείμενο που περικλείουν οι ετικέτες αυτές. Το χρησιμοποιούμε όταν έχουμε ολόκληρο το έγγραφο XML και όχι όταν το λαμβάνουμε από μία απομακρυσμένη τοποθεσία γιατί πρέπει να είναι ολοκληρωμένο για να αναγνωστεί. Κατά κύριο λόγο το χρησιμοποιούμε όταν θέλουμε να κατασκευάσουμε ένα XML μέσω PHP κώδικα αυτούσιο και όχι να το γράψουμε με το χέρι, η όταν θέλουμε προχωρημένη «πλοήγηση» στο έγγραφο αυτό, μετακίνηση σε προηγούμενους κόμβους κλπ. Το μειονέκτημα του DOM είναι ότι επειδή αναγνώθει ένα ολόκληρο έγγραφο και το τοποθετεί στην μνήμη μπορεί να προκαλέσει πολύ μεγάλη επιβάρυνση στο σύστημα.

Το **SAX** **S**imple **A**PI for **X**ML είναι πολύ ελαφρύ και εύκολο να χρησιμοποιηθεί αλλά στην ουσία συμπεριφέρεται σαν να διαβάζει όλο το XML έγγραφο σαν ένα μεγάλο string. Ενδείκνυται για «σειριακό» διάβασμα ενός XML και για επαναλαμβανόμενες διαδικασίες στα στοιχεία του (πχ αντικατάσταση ενός στοιχείου που εμφανίζεται επανειλημμένα) και εμφάνιση του σε μορφή HTML αλλά δεν είναι δυνατή η κατά βούληση προσπέλαση του(προηγούμενα στοιχεία, ανώτεροι κόμβοι κλπ).

Το **Simple XML API** είναι πολύ εύκολο στο να ανοίξει γρήγορα ένα έγγραφο, να μετατρέψει κάποια από τα στοιχεία του σε μεταβλητές, πίνακες ή αντικείμενα της PHP και να διαχειριστεί αυτές τις μεταβλητές όπως θα μεταχειριζόταν οποιεσδήποτε php μεταβλητές. Είναι ο ποιο εύκολος τρόπος και με λιγότερες κλήσεις σε συναρτήσεις όπως στο SAX και στο DOM και είναι πολύ ποιο ελαφρύ όσον αφορά την κατανάλωση μνήμης του συστήματος. Έχει βέβαια κάποιους περιορισμούς όσον αφορά τα χαρακτηριστικά των ετικετών και κάποια στοιχεία τα οποία είναι κατά πολύ εμφωλευμένα.

Στην εργασία μας για την μετασκευή των ιστότοπων χρησιμοποιούμε δύο από αυτά τα API της PHP. Το **DOM** όσον αφορά την δημιουργία των XML αρχείων μας μέσω PHP κώδικα και το **Simple XML** όσον αφορά την ανάγνωση και την διαχείριση των στοιχείων τους.

3 Υπηρεσίες Ιστού (Web Services)

Οι Υπηρεσίες κατά τον W3C () είναι ένα σύστημα για την αλληλεπίδραση μηχανών σε ένα Δίκτυο βασισμένο στην γλώσσα δεδομένων XML και στην μεταφορά τους μέσω HTTP. Για πολλούς είναι η απάντηση η τουλάχιστον ένας φωτεινός φάρος όσον αφορά την ενοποίηση και την ανταλλαγή δεδομένων ανάμεσα σε απομακρυσμένα συστήματα.

Ποια είναι η χρήση τους; «Η εύκολη ενοποίηση ετερόκλητων συστημάτων.» Σας δίνω ένα αρκετά κατατοπιστικό παράδειγμα από τον τίτλο «Programming PHP» από τον εκδοτικό O'REILLY.

«Ας πούμε ότι έχουμε ένα πολύπλοκη σύστημα το οποίο διαχειρίζεται πάρα πολλά tables βάσεων δεδομένων με πολύπλοκους συσχετισμούς και πράξεις και είναι κατασκευασμένο σε C#, C++, VB και γενικά σε κάποιον γνωστό κώδικα για Desktop εφαρμογές. Ας πούμε τώρα ότι μερικά από αυτά τα στοιχεία θέλουμε να είναι προσβάσιμα από το διαδίκτυο και θέλουμε να φτιάξουμε και μια φόρμα για την εισαγωγή νέων στοιχείων σε αυτό. Υπάρχουν δυο λύσεις η να κάτσουμε και να μεταφέρουμε όλο το πολύπλοκο σύστημα διαχείρισης πινάκων στην PHP και να γράψουμε άπειρο κωδικό για την σωστή διαχείριση των tables ξα να γράψουμε ελάχιστο κώδικα σε C# η οποία γλώσσα και να είναι ώστε να «δημοσιοποιήσουμε» τις απαραίτητες διεργασίες σαν ένα Web Service. Έτσι το μόνο που πρέπει να κάνουμε τώρα σε PHP είναι να διαχειριστούμε το μέρος του διαδικτύου και να αφήσουμε το Service του συστήματος να κάνει όλη την υπόλοιπη δουλεία.»

Ποια είναι όμως τα χαρακτηριστικά με τα οποία θα πρέπει να λειτούργει ένα πρωτόκολλο για τις Υπηρεσίες Ιστού; Το πρώτο κοινός παραδεκτό χαρακτηριστικό στην ενοποίηση συστημάτων είναι ότι όλα τα συστήματα θα πρέπει να μιλάν μια κοινή γλώσσα. Χωρίς αμφιβολία τώρα ποια αυτή είναι η XML. Το δεύτερο είναι, ότι θα πρέπει να υπάρχει ένας κοινός μηχανισμός μεταφοράς. Το ρόλο αυτό αναλαμβάνει το είδη υπάρχον HTTP πρωτόκολλο το οποίο είναι αρκετά βολικό μιας και μπορεί και παίρνει από τα περισσότερα firewalls αφού χρησιμοποιεί προκαθορισμένες πόρτες (σε αντίθεση με τα Sockets). Εφόσον μπορούμε να παράγουμε υπηρεσίες ιστού με γλώσσες σαν την PHP λύνεται και το θέμα κόστους. Αυτά όμως που ακόμα παραμένουν είναι κάποια θέματα περί απλότητας και ευχρηστίας, γι αυτό και έχουμε παραπάνω από ένα πρωτόκολλα όσον αφορά τις υπηρεσίες ιστού. Αυτά είναι τα **REST, XML-RPC, και SOAP.**

3.1)REST

Rest σημαίνει **R**epresentational **S**tate **T**ransfer (αντιπροσωπευτική κατάσταση μεταφοράς). Η φιλοσοφία της αρχιτεκτονικής αυτής είναι ότι υπάρχει ήδη ότι χρειαζόμαστε για να εφαρμόσουμε τις υπηρεσίες ιστού μέσα από το πρωτόκολλο HTTP. Κατ' επέκταση όλες οι υπηρεσίες ιστού πρέπει να είναι προσβάσιμες από κοινά URI (Uniform Resource Identifier) χρησιμοποιώντας την μέθοδο GET του HTTP και να επιστρέφουν δεδομένα σε

μορφή XML χωρίς κανένα ιδιαίτερο περίβλημα κώδικα. Κοινός μια REST υπηρεσία είναι απλά μία XML σελίδα στο διαδίκτυο. Έτσι δεν υπάρχει κάποιος κώδικας για την δημιουργία της, κάποια υποστήριξη για περίπλοκους τύπους, η κάποιο εξειδικευμένο λεξιλόγιο. Είναι χρήσιμη όμως λόγο της απλότητας της για τους λιγότερο τεχνικά καταρτισμένους και χρησιμοποιείται κυρίως για διαμοιρασμό περιεχομένου. Το μόνο λοιπόν που χρειάζεται είναι να ανεβάσουμε στο διαδίκτυο τις πληροφορίες σε μορφή XML και να δημοσιεύσουμε στους χρήστες μας το URI που βρίσκεται.

3.2) XML – RPC (XML Remote Procedure Call)

Η XML- RPC αναφέρεται σε μια προδιαγραφή για να γίνονται απομακρυσμένες κλήσεις διεργασιών μέσω του HTTP πρωτοκόλλου χρησιμοποιώντας XML. Αποτελείται από ένα Server και ένα Client. Ο server δέχεται μία κλήση κωδικοποιημένη σε xml που αποστέλλεται μέσω HTTP POST για μία συγκεκριμένη δημοσιευμένη διεργασία. Αφού γίνει η προσπέλαση της κλήσης και μεταφερθούν οι παράμετροι δεδομένα στην διεργασία επιστρέφεται η απάντηση κωδικοποιημένη σε XML με παρόμοιο τρόπο. Εδώ βλέπουμε ότι αντιθέτως με την REST δεν ζητάω απλά να πάρω δεδομένα αλλά καλώ μια συγκεκριμένη συνάρτηση σε ένα άλλο μηχάνημα χρησιμοποιώντας συγκεκριμένους τύπους και περνώντας παραμέτρους. Εν ολίγοις οποιαδήποτε συνάρτηση διαθέτει ένας server μπορεί να δημοσιοποιηθεί σαν service είτε αποστέλλει δεδομένα σαν απάντηση είτε όχι. Επίσης η XML- RPC υποστηρίζει όλους τους τύπους δεδομένων της PHP εκτός από αντικείμενα και μερικούς ακόμα που δεν διαθέτει η PHP όπως δομές, ημερομηνία, κ.α . Είναι λοιπόν κάτι ενδιάμεσο ανάμεσα στην υπερβολική απλότητα της αρχιτεκτονικής REST και της πολυπλοκότητας του SOAP.

3.3) SOAP (Simple Object Access Protocol)

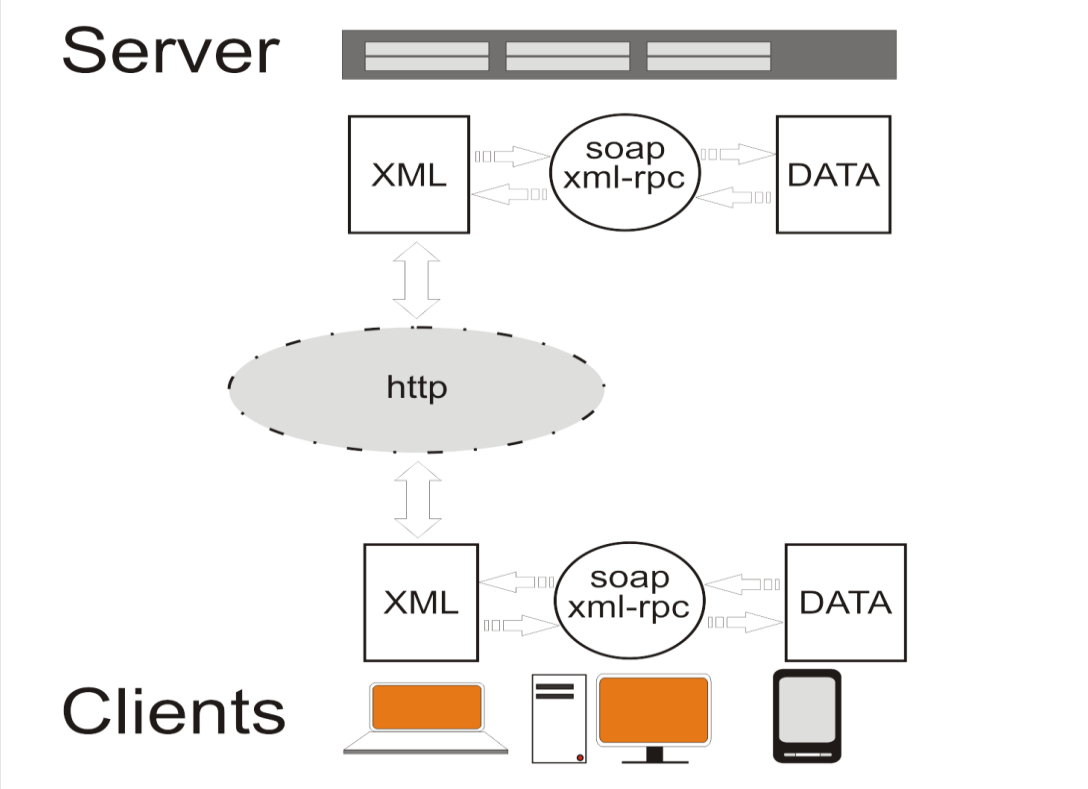
Το SOAP είναι ένα πρωτόκολλο το οποίο είναι προτεινόμενο από τον W3C και είναι γραμμένο από μια επιτροπή η απαρτίζεται κατά μεγάλο ποσοστό από τους «μεγάλους» κατασκευαστές λογισμικού κυρίως από την Microsoft και την IBM. Αυτό είναι μεν καλό γιατί η υποστήριξη ήταν ραγδαία εφόσον το υιοθέτησαν κατευθείαν οι μεγάλες εταιρίες λογισμικού, από την άλλη είναι ένα open standard το οποίο μπορεί να χρησιμοποιηθεί σαν εργαλείο πωλήσεων για τις εταιρίες αυτές κάτι το οποίο είναι εναντίον της φιλοσοφίας του Open Software.

Όπως και η XML RPC το SOAP στέλνει μηνύματα (requests) σε ένα XML περίβλημα αλλά με πολύ ποιο αυστηρό λεξιλόγιο στο οποίο γίνεται εκτεταμένη χρήση namespaces.

Το SOAP προσφέρει ακόμα περισσότερους τύπους δεδομένων από την XML – RPC αλλά πρέπει να προσδιοριστούν πολλές περισσότεροι παράμετροι. Κατά τα άλλα οι λειτουργία τους είναι παρόμοια. Αιτήσεις με δεδομένα σαν παράμετροι για δημοσιοποιημένες συναρτήσεις (Requests) και απαντήσεις (Reply) σαν έξοδος από τις συναρτήσεις αυτές και όλα αυτά σε ένα XML περίβλημα.

Το SOAP βέβαια είναι το ποιο περίπλοκο και βαρύ εργαλείο για την κατασκευή υπηρεσιών ιστού αλλά είναι και το ποιο συγκεκριμένο όσον αφορά τους τύπους δεδομένων που ανταλλάσσονται και τις συναρτήσεις που εκτείνονται. Σαν βέβαια το ποιο

βαρύ εργαλείο εμφανίζει μειονεκτήματα ειδικά σε περιοχές που ακόμα και τώρα ο χώρος και η μνήμη αποτελούν ακόμα ζήτημα όπως τα κινητά τηλεφωνά η καλύτερα smart phones. Ακόμα και το περίβλημα του που αποτελείται από ένα φάκελο, επικεφαλίδες, περιεχόμενο κ.α το κάνουν ιδιαιτέρως δυσανάγνωστο οπότε και δύσκολο για τον απλό χρήστη να καταλάβει το τι γίνεται στην ουσία του. Η εκτεταμένη όμως υποστήριξη που δίνεται από τις μεγάλες εταιρίες και κυρίως από την Microsoft το κάνουν ιδιαιτέρως εύχρηστο όσον αφορά εφαρμογές με .NET framework εφόσον παρέχονται αρκετές βιβλιοθήκες και συναρτήσεις. Ευτυχώς για μας (τους απλούς κατασκευαστές) υπάρχουν διάφορες βιβλιοθήκες και για SOAP αλλά και για XML – RPC και στην PHP οι οποίες μας δίνουν την δυνατότητα να κατασκευάσουμε και να καταναλώσουμε υπηρεσίες συγκρατώντας μόνο τα δεδομένα που χρειαζόμαστε.



Εικόνα 2

3.4) WSDL (Web Service Description Language)

Ένα πάρα πολύ χρήσιμο εργαλείο και ιδιαίτερα όσον αφορά την χρήση της αρχιτεκτονικής SOAP είναι το WSDL. Είναι ένα έγγραφο γραμμένο και αυτό σε XML και είναι σύσταση του W3C, το οποίο περιγράφει μία υπηρεσία ιστού, τα δεδομένα της και τις μεθόδους της. Ένα WSDL περιγράφει μια υπηρεσία ιστού χρησιμοποιώντας τέσσερις κύριες ετικέτες (tags). Την ετικέτα <types> η οποία περιγράφει τους τύπους δεδομένων που χρησιμοποιεί η υπηρεσία και για περισσότερη ουδετερότητα χρησιμοποιεί XSD σύνταξη για να ορίσει τους τύπους δεδομένων. Την ετικέτα <message> η οποία περιγράφει τα μηνύματα της υπηρεσίας και τις ανάλογες «παραμέτρους» τους δηλαδή τα μέρη του μηνύματος άμα είναι πάνω από ένα. Η ετικέτα <portType> που περιγράφει της λειτουργίες της υπηρεσίας είναι και οι ποιο σημαντική ετικέτα του WSDL γιατί περιγράφει την υπηρεσία, τις μεθόδους και τα μηνύματα που εμπλέκονται στην μέθοδε. Τέλος η ετικέτα

<binding> περιγράφει τα πρωτόκολλα που χρησιμοποιούνται από την υπηρεσία και τις λεπτομέρειες τους και την μορφολογία του μηνύματος.

3.5) XML-RPC EPI και PHP

Η PHP έρχεται στις τελευταίες εκδόσεις με έτοιμες επεκτάσεις για προγραμματισμό υπηρεσιών με X M L - R P C. Οι επεκτάσεις αυτές είναι αντίγραφο της XML – RPC EPI επέκτασης που είναι γραμμένη σε C και προσφέρουν συναρτήσεις για την δημιουργία XML κωδικοποιημένων δεδομένων για την ανταλλαγή τους με XML – RPC.

Εμείς χρησιμοποιούμε το xmlrpc-epi-php-0.51 που περιγράφεται στην διεύθυνση http://xmlrpc-epi.sourceforge.net/. Το αρχείο μπορούμε να το βρούμε στην παρακάτω διεύθυνση http://sourceforge.net/projects/xmlrpc-epi/files/xmlrpc-epi-php/

Απλά θα πρέπει να εισάγουμε πριν το χρησιμοποιήσουμε το αρχείο «utils.php» ώστε να χρησιμοποιήσουμε τις διαθέσιμες συναρτήσεις της βιβλιοθήκης.

```
include("utils/utils.php"); // from xmlrpc-epi utils
```

Δημιουργία ενός XML – RPC SERVER

```
<?php

//Το πρώτο πράγμα που πρέπει να καθορίσουμε είναι οι συναρτήσεις οι οποίες //θα δημοσιευτούν

function say_hi($method name, $params, $app_data)
{
$name=$params[0];
return “Hello “.$name;
}

//Τώρα δημιουργούμε ένα server και δίνουμε την διαχείρισης του στην //μεταβλητή $xmlrpc_server

$xmlrpc_server = xmlrpc_server_create();

//Πρέπει μια PHP συνάρτηση να την δηλώνει σαν μέθοδο XML – RPC //χρησιμοποιώντας //την συνάρτηση xmlrpc_server_register_method() η οποία //παίρνει τρεις παραμέτρους. //Την μεταβλητή που αναθέσαμε το server , το //όνομα με το οποίο θέλουμε να //καταχωρήσουμε την συνάρτηση και το όνομα //της συνάρτησης.

xmlrpc_server_register_method($xmlrpc_server, “hello”, “say_hi”);

//Όταν ένα request στέλνεται στο script αυτό το βρίσκουμε στο raw post data

$request_xml = $HTTP_RAW_POST_DATA;

//Η συνάρτηση xmlrpc_server_call_method() στέλνει το request στο server και //επιστρέφει την απάντηση σε XML. Παίρνει τρεις παραμέτρους. Την μεταβλητή //που αναθέσαμε στο server , το request που έχουμε λάβει, εδώ την μεταβλητή //$request_xml, και σαν Τρίτη παράμετρο δεδομένα για την εφαρμογή. Ότι //δίνουμε σε αυτήν την παράμετρο περνά στο $app_data της δήλωσης της //συναρτησης.
```

```
$response= xmlrpc_server_call_method($xmlrpc_server, $request_xml, ‘’);

//εκτυπώνουμε την απάντηση για να την δει ο client

Print $response;
//και ελευθερώνουμε τους πόρους που χρησιμοποιήσαμε για την κατασκευή του server.
xmlrpc_server_destroy($xmlrpc_server);
?>
```

original example : devshed.com

Δημιουργία ενός XML – RPC CLIENT

```
<?php

// πρώτα κάνουμε include την βιβλιοθήκη η οποία θα μας βοηθήσει να κάνουμε εύκολα
XML- RPC requests.
include("utils/utils.php");

// εδώ δίνουμε τα στοιχεία του host που έχουμε το service και το uri που βρίσκεται
//πχ αν το url μας είναι http://www.alekz.eu/xmlrpc_server.php τότε το $host μας
//είναι www.alekz.eu και το uri ειναι /xmlrpc_server.php

$host=”yourhost.yourdoamin.com”;
$uri = “/xmlrpc_server.php”;

// εδώ κάνουμε και το request στην μέθοδο με την συνάρτηση xu_rpc_http_concise η
//οποία παίρνει σαν παράμετρο έναν πίνακα που περιέχει όλα τα στοιχεία που
//χρειάζονται για το request και επιστρέφουμε αυτά τα δεδομένα στην μεταβλητή
//$result

$result = xu_rpc_http_concise(
array(
‘method’ => “hello”, //Το όνομα της μεθόδου
‘args’ => array($name), //Τις παραμέτρους που περνάμε
‘host’ => $host, //Το host που βρίσκεται η μέθοδος
‘uri’ => $uri, //Το uri που βρίσκεται η μέθοδος
‘port’ => 80 //η πόρτα που επιθυμούμε μέσω να γίνει η επικοινωνία.
)
);

Print $result;  //Τέλος εκτυπώνουμε το αποτέλεσμα που πήραμε από την μέθοδο
?>
```

original example : devshed.com

Μέσα από αυτά τα παραδείγματα και χρησιμοποιώντας τα σαν ένα βασικό οδηγό μπορούμε να υλοποιήσουμε πολύ απλά με PHP και το XML – RPC EPI servers και clients και αναπτύσσοντας τα να φτιάξουμε το service που επιθυμούμε. Σίγουρα βέβαια δεν έχουμε καλύψει όλες τις πτυχές της XML RPC. Περισσότερα μπορούν να βρεθούν στην διεύθυνση: http://xmlrpc.scripting.com/spec.html

3.6) SOAP και PHP

Για να κατασκευάσουμε μια SOAP PHP εφαρμογή χρησιμοποιούμε την βιβλιοθήκη **NuSOAP**. Η βιβλιοθήκη αυτή είναι ένα σύνολο κλάσεων που βοηθούν στην δημιουργία αλλά και κατανάλωση υπηρεσιών ιστού που είναι βασισμένες στο πρωτόκολλο SOAP χωρίς καμία άλλη επέκταση. Υποστηρίζει την προδιαγραφή του SOAP 1.1. Μπορεί να δημιουργήσει έγγραφα WSDL 1.1 όπως και να τα χρησιμοποιήσει. Περισσότερες πληροφορίες μπορούμε να βρούμε http://www.scottnichol.com/nusoapintro.htm και μπορούμε να κατεβάσουμε την επέκταση από την παρακάτω διεύθυνση http://sourceforge.net/projects/nusoap/ Απλά πρέπει να συμπεριλάβουμε το ανάλογο αρχείο βιβλιοθήκης για να κάνουμε χρήση των συναρτήσεων.

```
<?php require_once('nusoap.php'); ?>
```

Παράδειγμα ενός SOAP Server

```
// Ένα απλοϊκό παράδειγμα ενός SOAP server

<?php
// Ζητάμε τον nusoap κώδικα
require_once('nusoap.php');

// Δημιουργούμε μια μεταβλητή που της αναθέτουμε το server
$server = new soap_server;

// Καταχωρούμε την συνάρτηση που θα δημοσιεύσουμε
$server->register('hello');

// Δηλώνουμε η κατασκευάζουμε την συνάρτηση
function hello($name) {
    return 'Hello, ' . $name;
}

// Χρησιμοποιούμε το request για να προσπαθήσουμε να ενεργοποιήσουμε την
//υπηρεσία
$HTTP_RAW_POST_DATA = isset($HTTP_RAW_POST_DATA) ? $HTTP_RAW_POST_DATA :
'';
$server->service($HTTP_RAW_POST_DATA);
?>
                              original example : http://www.scottnichol.com/nusoapintro.htm
```

Παράδειγμα ενός SOAP Client

```
<?php
// Ζητάμε τον nusoap κώδικα
require_once('nusoap.php');

// Δημιουργούμε μια μεταβλητή που της αναθέτουμε τον client
//δίνοντας το url στο οποίο βρίσκεται ο SOAP server.
$client = new soapclient('http://localhost/phphack/helloworld.php');
```

```
//Κάνουμε κλήση της μεθόδου που θέλουμε να χρησιμοποιήσουμε
//με το όνομα της και δίνουμε τις παραμέτρους που θέλουμε
//να περάσουμε στην συνάρτηση
$result = $client->call('hello', array('name' => 'Scott'));

// Αν όλα πάνε όπως πρέπει εμφανίζουμε το αποτέλεσμα.
print_r($result);
?>
                                original example : http://www.scottnichol.com/nusoapintro.htm
```

Με την χρήση των συναρτήσεων response και request μπορούμε να εκτυπώσουμε και να δούμε τα soap μηνύματα τα οποία αποστέλλονται.

```
echo '<pre>' . htmlspecialchars($client->request, ENT_QUOTES) . '</pre>';
echo '<pre>' . htmlspecialchars($client->response, ENT_QUOTES) . '</pre>';
```

Σε περίπτωση που χρειαζόμαστε αποσφαλμάτωση του κώδικα μας, αν τα μηνύματα μας δεν εμφανίζονται όπως θα περιμέναμε μπορούμε να εκτυπώσουμε το debugging μήνυμα. Η ερμηνεία βέβαια μπορεί να είναι αρκετά δύσκολη λόγο της περίπλοκης δομής του Soap αλλά μερικές φορές είναι και η μόνη λύση που έχουμε.

```
echo '<pre>' . htmlspecialchars($client->debug_str, ENT_QUOTES) . '</pre>';
```

Σας παραθέτουμε για λόγους αναφοράς τώρα περισσότερο πως είναι στην πραγματική τους δομή τα μηνύματα ζήτησης(request) και απάντησης(response) του SOAP.

```
                                    //REQUEST

POST /phphack/helloworld2.php HTTP/1.0
Host: localhost
User-Agent: NuSOAP/0.6.8 (1.81)
Content-Type: text/xml; charset=ISO-8859-1
SOAPAction: ""
Content-Length: 538

<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:si="http://soapinterop.org/xsd">
 <SOAP-ENV:Body>
  <ns1:hello xmlns:ns1="http://testuri.org">
   <name xsi:type="xsd:string">Scott</name>
  </ns1:hello>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```
                                    //RESPONSE

HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Wed, 03 Nov 2004 21:32:34 GMT
X-Powered-By: ASP.NET
X-Powered-By: PHP/4.3.4
Server: NuSOAP Server v0.6.8
X-SOAP-Server: NuSOAP/0.6.8 (1.81)
Content-Type: text/xml; charset=ISO-8859-1
Content-Length: 556

<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:si="http://soapinterop.org/xsd">
 <SOAP-ENV:Body>
  <ns1:helloResponse xmlns:ns1="http://tempuri.org">
   <return xsi:type="xsd:string">Hello, Scott</return>
  </helloResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Τα δύο παραπάνω παραδείγματα είναι ότι ποιο απλό σε σχέση με μία πραγματική εφαρμογή αλλά δίνουν την ιδέα και τον σκελετό της λειτουργίας ενός SOAP server και client. Ποιο εκτεταμένα παραδείγματα μπορούμε να βρούμε : http://www.scottnichol.com/nusoapprog.htm

3.7) WS vs. Sockets

Πριν από διάδοση των WS λύσεις με παρόμοια λειτουργία και αποτελέσματα ήταν η χρήση των Sockets. Τα sockets είναι σχετικά μία απλή σχετικά έννοια αλλά όχι και τόσο απλή στην υλοποίηση τους. Ας δούμε μία πολύ απλοποιημένη ανάλυση των sockets. Τα sockets λειτουργούν σαν «βύσματα» που στο κάθε ένα «κουμπώνουμε» την λειτουργία μιας πόρτας “port”. Με αυτόν τον τρόπο κάθε server ανοίγει και «ακούει» αυτήν την πόρτα και όταν έρθει κάποιο μήνυμα το «ακούει» και μετά «απαντά». Τα μηνύματα ανταλλάσσονται συνήθως με την χρήση μεγάλων strings τα οποία και αναλύονται στον server και στον client.

Οι ομοιότητες είναι πολλές με την χρήση των Υπηρεσιών Ιστού αλλά υπάρχουν σημαντικές διαφορές. Κάθε τεχνολογία έχει τα πλεονεκτήματα και τα μειονεκτήματα της όπως:

- Τα sockets χρησιμοποιούν το πρωτόκολλο TCP «Transfer Control Protocol» και το UDP «User Datagram Protocol» ενώ τα WS χρησιμοποιούν το HTTP «Hypertext

Transfer Protocol». Από αυτήν την άποψη τα sockets θεωρούνται γρηγορότερα και με λιγότερο όγκο σε κεφαλίδες και άλλα στοιχειά του πρωτοκόλλου.

- Τα WS είναι πολύ ποιο εύκολα στην μάθηση και στην χρήση και γνωρίζουν αυξανόμενη αποδοχή και χρήση.

- Τα WS χρησιμοποιούν την ήδη ανοιχτή πόρτα 80 που είναι καθιερωμένη για την χρήση διαδικτύου και τα περισσότερα εμπορικά firewall έχουν ήδη αυτό που αποκαλούμε τρύπες σε αυτά για την συγκεκριμένη πόρτα. Έτσι η γενικότερη χρήση των WS είναι ποιο εύκολη από άποψη αποδοχής σε διάφορα συστήματα.

- Τα WS χρησιμοποιούν μηχανισμούς κωδικοποίησης και μπορούν να λειτουργήσουν και με SSL “secure socket layer” ενώ στην χρήση socket πρέπει να χρησιμοποιήσουμε δικούς μας μηχανισμούς κρυπτογράφησης δεδομένων.

Αυτά είναι και μερικά από τα πλεονεκτήματα και μειονεκτήματα των τεχνολογιών αυτών. Αναλύοντας τα και κάνοντας μια μικρή έρευνα στο διαδίκτυο μπορούμε να πούμε με ασφάλεια ότι «Εάν δεν υπάρχουν συγκεκριμένοι λόγοι και απαιτήσεις από την μεριά των εφαρμογών είναι πολύ ποιο εύκολη και ίσως αποτελεσματική η χρήση WS σε σχέση με τα sockets». Η ευκολία λοιπόν αλλά και οι διαπερατότητα της λειτουργίας των υπηρεσιών ιστού κάνει την χρήση τους ποιο διαδεδομένη και ποιο «κοινή».

4 Χτίζοντας τις Εφαρμογές

Για να μελετήσουμε την διασύνδεση και την ανταλλαγή δεδομένων μέσω διαφορετικών συστημάτων έπρεπε να κατασκευάσουμε συστήματα – παραδείγματα σε διαφορετικούς «χώρους»(servers) και με διαφορετική πολιτική διαχείρισης των δεδομένων τους. Φυσικά τα συστήματα αυτά ασχολούνται με την ανταλλαγή δεδομένων για μικρές αγγελίες και στην περίπτωση μας ειδικά για αυτοκίνητα. Το πρώτο που κρίθηκε απαραίτητο είναι η κατασκευή ενός site το οποίο θα δημοσιεύει μικρές αγγελίες , ώστε να μελετηθούν οι ιδιαιτερότητες που εμφανίζονται σε ένα τέτοιο χώρο.

4.1) Χτίζοντας το «Carz» website

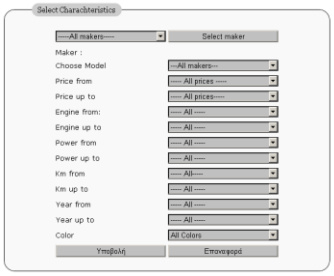
Το carz είναι ένα site το οποίο ασχολείται με την εύρεση και δημοσίευση μικρών αγγελιών για μεταχειρισμένα αυτοκίνητα. Λειτουργεί με τοπική βάση mysql που είναι κατασκευασμένη στο server και εκεί αποθηκεύονται όλα τα δεδομένα του. Οι τεχνολογίες που χρησιμοποιήσαμε είναι καθαρά php, html, css, και mySql.

Το πρώτο βήμα ήταν φυσικά να ασχοληθούμε με το αντικείμενο πώλησης δηλαδή τα αυτοκίνητα και τα τεχνικά χαρακτηριστικά τους και την συλλογή δεδομένων. Μελετήθηκαν διαφορά site παρόμοιας λειτουργίας όσον αφορά την κατασκευή και την δομή τους, την παρουσίαση των δεδομένων τους και τα στοιχεία αλληλεπίδρασης με τον χρήστη. Τα βασικά λοιπόν χαρακτηριστικά όσον αφορά την λειτουργία ενός τέτοιου χώρου είναι: Η αναζήτηση μικρών αγγελιών, η δυνατότητα δημοσίευσης αγγελίας και η διαχείριση μιας αγγελίας από τον ιδιοκτήτη της. Παρακάτω αναφέρουμε τις κάποια προβλήματα αλλά και θέματα τα οποία συναντήσαμε κατά την υλοποίηση.

Εικόνα 3

- **Η αναζήτηση μικρής αγγελίας.**

Μεγάλο θέμα όσον αφορά την αναζήτηση είναι η μορφή της φόρμας αναζήτησης ώστε να εξυπηρετεί τον χρήστη. Αυτό οδηγεί στην χρήση στοιχείων select για τα κριτήρια αναζήτησης, τα οποία έχουν προκαθορισμένες τιμές και ο χρήστης διαλέγει μια από αυτές. Φυσικά υπάρχει η απαραίτητη προϋπόθεση οι τιμές αυτές να προϋπάρχουν. Εδώ συναντήσαμε και τις πρώτες δυσκολίες.



- **Κατασκευαστές – μοντέλα και συσχετισμός δεδομένων**

Μετά από έρευνα καταφέραμε να συγκεντρώσουμε έγκυρα στοιχεία για όλους τους κατασκευαστές αυτοκινήτων του χώρου και τα μοντέλα τα οποία διαθέτει ο καθένας. Αυτά τοποθετήθηκαν σε μια βάση δεδομένων σε δύο διαφορετικά tables. Στο ένα μπήκαν οι κατασκευαστές και οι κωδικοί που τους δώσαμε (δύο πεδία) και στον αλλά τα μοντέλα με τον κωδικό τους και τον αντίστοιχο κωδικό κατασκευαστή για τον καθένα (τρία πεδία). Το σκεπτικό λοιπόν είναι ένας χρήστης να επιλεγεί πρώτα τον κατασκευαστή που ψάχνει και μετά το μοντέλο αυτού. Αυτά είναι και τα μόνα αλληλεξαρτώμενα δεδομένα που έχουμε. Το πρόβλημα λοιπόν ήταν με ποια μέθοδο θα φορτώνονται σαν επιλογές τα μοντέλα με την επιλογή του κατασκευαστή τους.

Χρησιμοποιώντας σε αυτό το παράδειγμα μόνο php και MySQL η μόνη λύση που έχουμε είναι ο χρήστης να επιλέγει τον κατασκευαστή από ένα πεδίο και να κάνεις μετά submit στην βάση για να πάρει πίσω τα δεδομένα για τα μοντέλα του κατασκευαστή τα οποία φορτώνονται δυναμικά σε ένα 2^{ο} select σαν options και να επιλέξει το επιθυμητό και μετά να συνεχίσει την αναζήτηση του επιλέγοντας τιμές στα άλλα πεδία τα οποία είναι ανεξάρτητα όπως τιμή, κυβικά, άλογα και άλλα χαρακτηριστικά αυτοκινήτων και τέλος να κάνει πάλι submit το ολοκληρωμένο ερώτημα στην βάση και να πάρει τα επιθυμητά αποτελέσματα. Το θέμα αυτό με το διπλό submit είναι φυσικά κάτι το οποίο απαιτεί μια κάποια λύση γιατί δεν έχει και πολύ μεγάλη λειτουργικότητα. Την λύση αυτού του θέματος θα την δούμε στην κατασκευή της γενικευμένης μηχανής αναζήτησης, προς το παρόν λοιπόν βλέπουμε ακόμα την κατασκευή του site “carz”.

- **Επιλογές αναζήτησης και δυναμικά ερωτήματα sql.**

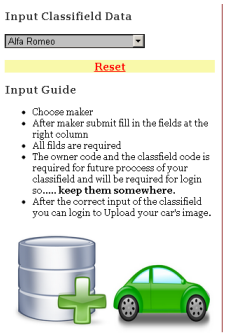
Ακόμα ένα αρκετά σημαντικό θέμα το οποίο βρήκαμε μπροστά μας είναι η παράγωγη των sql ερωτημάτων δυναμικά. Δηλαδή αυτά τα ερωτήματα να παράγονται φυσικά από τον χρήστη αλλά μόνο για τα δεδομένα τα οποία επιθυμεί αυτός. Πχ να επιλέγει μόνο τιμά σαν κριτήριο αναζήτησης και τίποτε άλλο. Αυτό φυσικά αποτελούσε πρόβλημα εφόσον στα ερωτήματα πρέπει να δώσουμε εμείς κριτήρια και το μόνο που αλλάζει στην ουσία είναι οι τιμές αυτόν τον κριτηρίων, τιμές τις οποίες επιλέγει φυσικά ο χρήστης. Εδώ όμως πρέπει και τα κριτήρια να επιλέγονται από τον χρήστη. Μία λύση

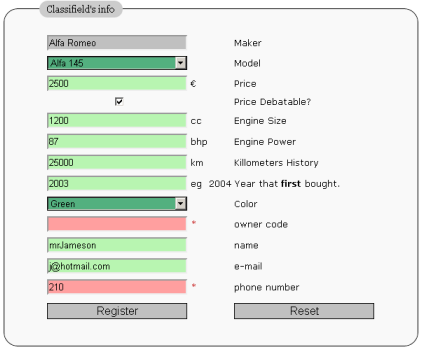
φυσικά θα ήταν να υποχρεώνουμε με κάποιο μηχανισμό τον χρήστη να επιλέξει τιμές σε όλα τα παιδιά αλλά αυτό είναι γενικότερα μη λειτουργικό, εφόσον περιορίζει κατά πολύ το εύρος αναζήτησης του χρήστη και είναι γενικότερα μια «*μεσοβέζικη*» τεχνική. Η λύση η οποία βρέθηκε σε αυτό το θέμα είναι η μετασκευή μιας συνάρτησης η οποία θα παρασκευάζει δυναμικά sql ανάλογα με τον αν έχει οριστεί μια τιμή η όχι. Έτσι τα μόνα κριτήρια που θα υπάρχουν είναι αυτά τα οποία θα περιέχουν κάποια τιμή, αλλιώς θα επιστρέφει όλα τα στοιχεία της βάσης με τις αναρτημένες αγγελίες. Η συνάρτηση αυτή δίνεται παρακάτω και εξηγείται όπου θεωρείται απαραίτητο.

```
//ορίζουμε την συνάρτηση και τις περνάμε τα δεδομένα από τα κριτήρια που έχει
επιλέγει ο χρήστης τα οποία παίρνουμε από μια «get» η μία «post».

function  dynamicSql($var1, $var2, $var3, $var4, $var5, $var6, $var7,
$var8, $var9, $var10, $var11, $var12, $var13)
{

$where = array();  //ορίζουμε έναν πίνακα για τα κριτήρια
$filter=null;  //και το φίλτρο αναζήτησης

//Εδώ ελέγχουμε αν οι τιμές που έχουμε περάσει μέσα στην συνάρτηση έχουν η όχι
//τιμή. Αν έχουν σημαίνει ότι ο χρήστης έχει επιλέξει μια τιμή σε ένα κριτήριο αν
//όχι το έχει αφήσει κενό όποτε δεν τον ενδιαφέρει. Η συνθήκη λοιπόν είναι η !empty
//που σημαίνει αν έχει τιμή τότε παράγουμε το κριτήριο με την τιμή σαν πεδίο του
//πινάκα αν όχι προχωράμε παρακάτω και αυτό γίνεται για όλα τα πεδία.

if(!empty($var1)) $where[] = "maker_code = '$var1'";
if(!empty($var2)) $where[] = "model_code = '$var2'";
if(!empty($var3)) $where[] = "price >= '$var3'";
if(!empty($var4)) $where[] = "price <= '$var4'";
if(!empty($var5)) $where[] = "cc >= '$var5";
if(!empty($var6)) $where[] = "cc <= '$var6'";
if(!empty($var7)) $where[] = "bhp >= '$var7'";
if(!empty($var8)) $where[] = "bhp <= '$var8'";
if(!empty($var9)) $where[] = "km <= '$var9";
if(!empty($var10)) $where[] = "km <= '$var10'";
if(!empty($var11)) $where[] = "add_year <= '$var11'";
if(!empty($var12)) $where[] = "add_year <= '$var12";
if(!empty($var13)) $where[] = "add_color = '$var13'";

//Εφόσον όλα τα παιδιά ελεγχτούν ορίζουμε το φίλτρο αν υπάρχει
if(count($where)) $filter ="WHERE ".implode(' AND ', $where);

//Δίνουμε το στάνταρ ερωτήματα και μετά το ενώνουμε με το υποψήφιο φίλτρο
$sql="SELECT * FROM `adds` ".$filter."";

//Το δυναμικά παραγμένο sql ερώτημα το δίνουμε σαν έξοδο της συνάρτησης.
return $sql;
}
```

- **Προσδιορισμός νέας αγγελίας και επικύρωση στοιχείων φόρμας**

Η υλοποίηση της εισαγωγής νέας αγγελίας υλοποιήθηκε ως εξής. Ο ιδιοκτήτης επιλεγεί την φόρμα εισαγωγής νέας αγγελίας και καλείται να δώσει συγκεκριμένα στοιχεία για το αυτοκίνητο προς πώληση και κάποια στοιχεία ώστε να επικοινωνεί ο κάθε ενδιαφερόμενος. Εδώ είναι ίσος και το μόνο σημείο που χρησιμοποιήσαμε κομμάτια διαφορετικής τεχνολογίας εκτός από MYSQL, και PHP για την επιβεβαίωση της φόρμας, κάτι που κρίνεται απαραίτητο για αποφυγή κακόβουλων, λανθασμένων και ψευδών δεδομένων, όσο φυσικά αυτό είναι εφικτό.





Η τεχνολογία που χρησιμοποιήθηκε είναι το Spry framework για Ajax εφαρμογές, και είναι ένα αρκετά βολικό και εύκολο εργαλείο που μπορούμε να βρούμε σε εφαρμογές της Adobe όπως το Dreamweaver CS5 που χρησιμοποιήθηκε στην περίπτωση αυτή. Με την χρήση της τεχνολογίας αυτής μπορούμε να κάνουμε μια φόρμα όχι μόνο να επικυρώνει η όχι την ύπαρξη τιμών σε κάποιο πεδίο αλλά και την μορφή της τιμάς αυτής, (ακέραιος, δεκαδικός, αλφαριθμητικό, mail κα) καθώς και το εύρος των χαρακτήρων άλλα και το μέγιστο και ελάχιστο τον υποχρεωτικών τιμών. Στην αποτυχία των κριτηρίων αυτών επιστρέφει στον χρήστη ένα ανάλογο κατατοπιστικό μήνυμα για την ορθή εισαγωγή δεδομένων.

- **Διαχείριση και διαγραφή αγγελίας**

Οι χρήστες εφόσον κάνουν εισαγωγή της αγγελίας τους τους δίνεται ένας κωδικός που παράγεται με στοιχεία της επιλογής τους και ο σειριακός αριθμός της αγγελίας που εισήγαν. Με αυτά τα στοιχεία μπορούν να κάνουν login και να διαχειριστούν την αγγελία τους. Το login δουλεύει με sessions και οι επιλογές του κάθε χρήστη είναι να διαγράψει την αγγελία του εφόσον δεν ισχύει πια και να ανεβάσεις κάποια φωτογραφία για την αγγελία του. Οι δυνατότητες διαχείρισης περιορίζονται σε αυτές τις λειτουργίες μιας και σκοπός μας είναι ο διαμοιρασμός των δεδομένων κρίθηκε να μην αφιερώσουμε περισσότερο χρόνο στην τοπική τους διαχείριση.

- **Εικόνες αγγελίας και script για ανέβασμα αρχείων.**

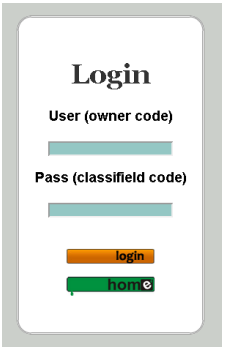
Σε κάθε site το οποίο περιέχει μικρές αγγελίες, ειδικά όσον αφορά αυτοκίνητα υπάρχει τουλάχιστον μια φωτογραφία που απεικονίζει το προς πώληση αυτοκίνητο. Εδώ το θέμα που παρουσιάζεται λοιπόν είναι Που θα είναι αποθηκευμένη η φωτογραφία αυτή, με τι όνομα, τι τύπος αρχείου θα είναι αυτή η φωτογραφία, τι μέγεθος, ποιες διαστάσεις θα έχει και κα.

Στην περίπτωση μας αποφασίσαμε ο χρήστης να μπορεί να εισάγει μια φωτογραφία εφόσον έχει κάνει επιτυχείς δήλωση της αγγελίας του με την χρήση τον

κωδικών και του σειριακού αριθμού της αγγελίας που τον προμηθεύουμε μετά την εισαγωγή. Ο χρήστης μπορεί να επιλέξει να ανεβάσει μια φωτογραφία συγκεκριμένου χωρητικότητας με προτεινόμενες διαστάσεις τριών διαφορετικών τύπων: Jpg, gif και png. Σε περίπτωση που το αρχείο δεν είναι ανάλογης μορφής η είναι μεγαλύτερη από το επιτρεπτό μέγεθος η διαδικασία διακόπτεται και επιστρέφει μηνύματα σφάλματος. Η εικόνα – αρχείο που κάνει upload ο χρήστης αποθηκεύεται σε έναν φάκελο συγκεκριμένο στο server και μετονομάζεται παίρνοντας το όνομα του μοναδικού κωδικού της κάθε αγγελίας. Παράλληλα στην βάση που είναι αποθηκευμένα τα στοιχεία της αγγελίας γίνεται update στο πεδίο της τοποθεσίας της εικόνας.

- **Το διαχειριστικό κομμάτι administration interface.**

Σε ένα τέτοιο χώρο διαχείρισης αγγελιών θα μπορούσε να υπάρχει ένα πολύ εκτεταμένο σύστημα διαχείρισης των αγγελιών και των χρηστών από την μεριά του διαχειριστή, που διαχειρίζονται. Τέτοια διαχείριση θυμίζουν πολλές έτοιμες λύσεις για e-shop και e_commerce site όπως το prestashop η και άλλα εργαλεία για Joomla και wordpress. Το μόνο όμως διαχειριστικό κομμάτι που μας ενδιαφέρει εμάς σε αυτήν εμάς είναι να εξάγει τα δεδομένα των μικρών αγγελιών το site σε μία διεύθυνση εντός sever για να κοινοποιήσει σε κάποια μηχανή αναζήτησης όπως θα δούμε και παρακάτω. Οπότε όταν εισάγουμε συγκεκριμένο login και pass στο σύστημα μπαίνουμε στο διαχειριστικό του κομμάτι και από εκεί το μόνο που υλοποιήσαμε είναι να μπορούμε να εξάγουμε τα δεδομένα μας σε μορφή XML. Το script αυτό χρησιμοποιεί το μοντέλο DOM για να δημιουργήσει το XML έγγραφο.



4.1.1) Χτίζοντας τον Soap Server για το «Carz».

Κάθε client και κάθε υπηρεσία που ανταλλάσει δεδομένα χρειάζεται έναν εξυπηρετητή δεδομένων. Αυτός ο εξυπηρετητή είναι και αυτός που διαθέτει και διανέμει τις πληροφορίες. Τον ρόλο αυτόν εδώ παίζει και το CARZ website που έχει και όλες τις πληροφορίες για τις μικρές αγγελίες για τα αυτοκίνητα. Για να κατασκευάσουμε λοιπόν μια υπηρεσία ιστού SOAP με server το carz θα πρέπει να κατασκευάσουμε και να δημοσιεύσουμε μια συνάρτηση που θα κάνει αναζήτηση και να την δημοσιοποιήσουμε σε κάποιο url του server μας. Επίσης θα κατασκευάσουμε και το ανάλογο WSDL το οποίο θα περιγράφει την υπηρεσία, τα δεδομένα αλλά και τους τύπους τους που θα εμπλέκονται σε αυτήν την υπηρεσία.

Θα περιγράψουμε το πώς γίνεται αυτό με απλά βήματα και όπου χρειάζεται με κάποιες γραμμές του κώδικα μας και την επεξήγηση που χρειάζεται.

```
<?php
// πρώτα απ’ όλα εισάγουμε την βιβλιοθήκη που μας δίνει τις συναρτήσεις που
//χρησιμοποιούμε
require_once('nusoap.php');
```

```
// Δημιουργούμε έναν καινούργιο server στιγμιότυπο και δίνουμε το αποτέλεσμα στην
μεταβλητή  $server που θα χρησιμοποιήσουμε σαν «χερούλι»
$server = new soap_server();

//Εδώ καθορίζουμε ότι έχουμε υποστηριξη wsdl εγγραφου για την υπηρεσία μας.
$server->configureWSDL('carws', 'urn:carws');


//Οριζουμε τους τυπους δεδομένων που θα χρησιμοποιούνται από την υπηρεσία μας
$server->wsdl->addComplexType(  //εδώ ορίζεται η μεταβλητή Car
    'Car',              //Η οποία περνάει στα request για την υπηρεσία
    'complexType', //και περιεχει τις πληροφοριες για το ζητουμενο
    'struct',   //αυτοκίνητο
    'all',
    '',
    array(
        "maker"=>array("name"=>"maker", "type"=>"xsd:int"),
                "model"=>array("name"=>"model", "type"=>"xsd:int"),
                "min_price"=>array("name"=>"min_price",
"type"=>"xsd:int"),
                "max_price"=>array("name"=>"max_price",
"type"=>"xsd:int"),
                "min_engine"=>array("name"=>"min_engine",
"type"=>"xsd:int"),
                "max_engine"=>array("name"=>"max_engine",
"type"=>"xsd:int"),
                "min_power"=>array("name"=>"min_power",
"type"=>"xsd:int"),
                "max_power"=>array("name"=>"max_power",
"type"=>"xsd:int"),
                "min_km"=>array("name"=>"min_km", "type"=>"xsd:int"),
                "max_km"=>array("name"=>"max_km", "type"=>"xsd:int"),
                "min_year"=>array("name"=>"min_year", "type"=>"xsd:int"),
        "max_year"=>array("name"=>"max_year", "type"=>"xsd:int"),
                "color"=>array("name"=>"color", "type"=>"xsd:string")
    )
);

$server->wsdl->addComplexType(  //Εδώ ορίζεται η μεταβλητή reply η
    'reply',    //οποία περνά στα response της υπηρεσίας
    'complexType', //και επιστρέφει αποτελέσματα από την αναζήτηση
    'struct',
    'all',
    '',
        array( 'reply0' => array('name' => 'reply', 'type' =>
'xsd:string'),
                //'reply1' => array('name' => 'reply1', 'type' => 'xsd:string'),
                //'reply2' => array('name' => 'reply2', 'type' => 'xsd:string'),
                //'reply3' => array('name' => 'reply3', 'type' => 'xsd:string'),
                //'reply4' => array('name' => 'reply4', 'type' => 'xsd:string')
    )
); //Εδώ ορίζουμε τα στοιχεία της απάντησης τα οποία πρέπει να είναι συγκεκριμένα
//επειδή δεν ξέρουμε την ποσότητα τον αποτελεσμάτων η επιστρέφουμε έναν
συγκεκριμένο αριθμό αποτελεσμάτων η τα συνδέουμε όλα σε ένα string και το περνάμε
σε μια μεταβλητή.


//Καταχωρούμε την μέθοδο την οποία αποφασίζουμε να δημοσιεύσουμε
$server->register('car',    // δίνουμε το όνομα της μεθόδου
// και καθορίζουμε τις παραμέτρους εισόδου και εξόδου
    array('Car' => 'tns:Car'),
```

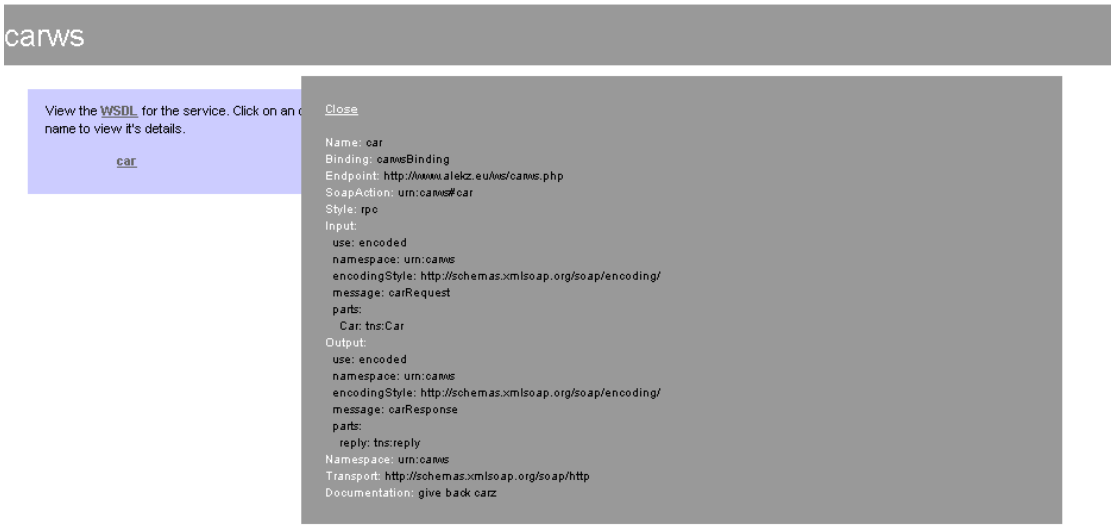
```
    array('reply' => 'tns:reply'),
        // output parameters
    'urn:carws',
        //καθορίζουμε και το namespace
    'urn:carws#car',
          //καθορίζουμε την ενέργεια soap
    'rpc',
        // και το στυλ της οποίας αν θα είναι κωδικοποιημένη η όχι
    'encoded',
// use
    'give back carz'   //Εδώ δίνουμε σαν τίτλο της υπηρεσίας η μια σύντομη
        //περιγραφή.
);

// Define the method as a PHP function

function car($Car) {
//εδώ κατασκευάζουμε την συνάρτηση περνώντας την παράμετρο $Car η οποία έρχεται
μέσα από το request της υπηρεσίας και μπορούμε να προσπελάσουμε τα δεδομένα σαν
στοιχεία πίνακα «παράδειγμα :».

$var1=$Car['maker'];
/*
Κατασκευή συνάρτησης

Και έξοδος αποτελεσμάτων
*/
$rep1["reply0"]=implode(' AND ', $rep);
        return $rep1;
}

// Εδώ κοιτάει άμα υπάρχει request μέσα στην μεταβλητή $HTTP_RAW_POST_DATA
//ώστε να επικαλεσθεί την υπηρεσία

$HTTP_RAW_POST_DATA = isset($HTTP_RAW_POST_DATA) ? $HTTP_RAW_POST_DATA :
'';
$server->service($HTTP_RAW_POST_DATA);
```

Με αυτόν τον κώδικα έχουμε ολοκληρώσει τον SOAP server και παράγουμε το ανάλογο WSDL το οποίο βρίσκεται στην παρακάτω διεύθυνση και έχει την παρακάτω μορφή:

```
<definitions targetNamespace="urn:carws"><types><xsd:schema
targetNamespace="urn:carws"><xsd:import
namespace="http://schemas.xmlsoap.org/soap/encoding/"/><xsd:import
namespace="http://schemas.xmlsoap.org/wsdl/"/><xsd:complexType
name="Car"><xsd:all><xsd:element name="maker" type="xsd:int"/><xsd:element name="model"
type="xsd:string"/><xsd:element name="min_price" type="xsd:int"/><xsd:element name="max_price"
type="xsd:int"/><xsd:element name="min_engine" type="xsd:int"/><xsd:element name="max_engine"
type="xsd:int"/><xsd:element name="min_power" type="xsd:int"/><xsd:element name="max_power"
type="xsd:int"/><xsd:element name="min_km" type="xsd:int"/><xsd:element name="max_km"
type="xsd:int"/><xsd:element name="min_year" type="xsd:int"/><xsd:element name="max_year"
type="xsd:int"/><xsd:element name="color"
type="xsd:string"/></xsd:all></xsd:complexType><xsd:complexType
name="reply"><xsd:all><xsd:element name="reply0"
type="xsd:string"/></xsd:all></xsd:complexType></xsd:schema></types><message
name="carRequest"><part name="Car" type="tns:Car"/></message><message name="carResponse"><part
name="reply" type="tns:reply"/></message><portType name="carwsPortType"><operation
name="car"><documentation>give back carz</documentation><input
message="tns:carRequest"/><output message="tns:carResponse"/></operation></portType><binding
name="carwsBinding" type="tns:carwsPortType"><soap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/><operation name="car"><soap:operation
soapAction="urn:carws#car" style="rpc"/><input><soap:body use="encoded" namespace="urn:carws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input><output><soap:body
use="encoded" namespace="urn:carws"
```

```
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output></operation></binding><ser
vice name="carws"><port name="carwsPort" binding="tns:carwsBinding"><soap:address
location="http://www.alekz.eu/ws/carws.php"/></port></service></definitions>
```

Το οποίο μπορούμε να δούμε από την αρχική σελίδα που ακολουθεί πατώντας το link που αναγράφει WSDL.



Εικόνα 4

Ο soap server είναι έτοιμος και αναμένει requests.

4.1.2)Χτίζοντας τον XML – RPC Server για το «Carz»

Η φιλοσοφία για έναν XML – RPC server είναι κατά πολύ παρόμοια με τον Soap εκτός από την αυστηρή δομή των δεδομένων και την υποστήριξη WSDL εγγράφου. Ο κώδικας κατασκευής ακόμα είναι αρκετά ποιο εύκολος και με λιγότερους ορισμούς. Παρομοίως με το προηγούμενο παράδειγμα δίνουμε κάποιες συναρτήσεις του κώδικα για την υλοποίηση με την ανάλογη επεξήγηση.

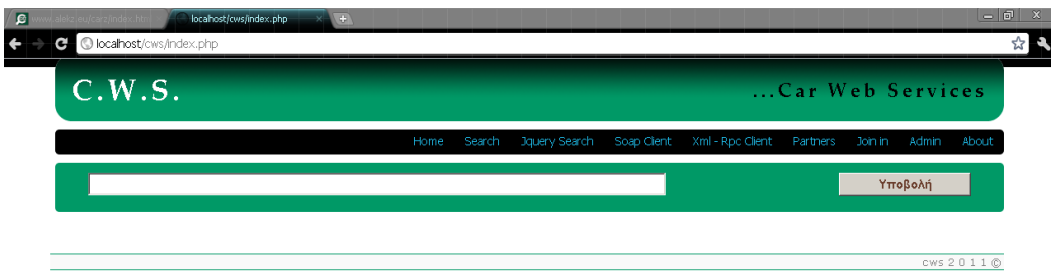
```
//===Κατασκευάζουμε την συνάρτηση
function car_func($method_name,$Car,$app_data) {
                        $var1=$Car["maker"];
/*
Κορμός συνάρτησης και εξαγωγή αποτελεσμάτων

*/
return $reply;
}
```

```
//========Δημιουργία server και επιστροφή «χερουλιού» για την Διαχείριση
$xmlrpc_server = xmlrpc_server_create();

//========Δημοσίευση Συνάρτησης με επωνυμία
xmlrpc_server_register_method($xmlrpc_server, "car_func", "car_func");

//========παίρνουμε το request  από το $HTTP_RAW_POST_DATA;
$request_xml = $HTTP_RAW_POST_DATA;

//======= παραγουμε την απάντηση σαν αποτέλεσμα της συνάρτησης
$response = xmlrpc_server_call_method($xmlrpc_server, $request_xml, '');

//===και εξάγουμε την απάντηση για τον χρήστη
print $response;

//===καταστρέφουμε το server και ελευθερώνουμε τους πόρους.
xmlrpc_server_destroy($xmlrpc_server);

?>
```

Βλέπουμε λοιπόν ότι η δομή είναι η ίδια σχεδόν με το SOAP server αλλά πολύ ποιο «χαλαρή» σε σχέση με την δημιουργία των δομών δεδομένων. Βέβαια υπάρχει το μειονέκτημα ότι δεν παράγεται κάποιο WSDL το οποίο μπορεί να χρησιμεύσει σαν οδηγός για την κατασκευή Client για την συγκεκριμένη υπηρεσία, κάτι το οποίο θα δούμε παρακάτω ότι μπορεί να παίξει πολύ σημαντικό ρόλο σε ορισμένες εφαρμογές των υπηρεσιών Ιστού.

4.2) Χτίζοντας την μηχανή αναζήτησης «C.W.S»

Το όνομα της η εφαρμογή το παίρνει από την ονομασία **C**ars **W**eb **S**ervices και παίζει το ρόλο μιας μηχανής αναζήτησης η οποία συνεργάζεται με διάφορα site που ασχολούνται με την δημοσίευση αγγελιών για μεταχειρισμένα αυτοκίνητα. Η μηχανή αυτή διαθέτει τέσσερις διαφορετικούς τρόπους αναζήτησης κάθε μία υποστηριζόμενη από διαφορετική τεχνολογία, και είναι στημένη στον τοπικό μας server. Οι τεχνολογίες αυτές θεωρητικά είναι όλες υπηρεσίες ιστού. Άλλες απαιτούν την παρέμβαση του χρήστη και άλλες ακολουθούν τεχνολογία «ζωντανής» αναζήτησης.



Εικόνα 5

- **Χειροκίνητη ανταλλαγή και αναβάθμιση δεδομένων.**

Οι αγγελίες οι οποίες βρίσκονται στην βάση δεδομένων της μηχανής αυτής είναι αποτέλεσμα ανανέωσης της βάσης με την παρέμβαση του διαχειριστή του site τραβώντας μικρές αγγελίες από αποθηκευμένες διευθύνσεις συνεργατών που δημοσιοποιούν σε αυτές τα δεδομένα τους. Οι ανάλογοι διαχειριστές των site αυτών με την σειρά τους είναι υπεύθυνοι για την ανανέωση των δικών τους XML αρχείων ώστε να ανταποκρίνεται στην υπάρχουσα κατάσταση του site τους. Οι αγγελίες αυτές αποθηκεύονται σε μια τοπική βάση όλες μαζί και το σύστημα από κει τις διαχειρίζεται ανάλογα με τις απαιτήσεις του.

Τα XML έγγραφα των συνεργατών πρέπει να ακλουθούν ένα XSD το οποίο έχουμε επίσης δημοσιευμένο σε μια θέση μέσα στο site για οδηγό προς τους συνεργάτες. Αυτό το έγγραφο μαζί με ένα παράδειγμα ενός XML αρχείου που περιγράφει αγγελίες αυτοκινήτων, και με έναν ενδεικτικό κώδικα PHP ο οποίος ακλουθεί το μοντέλο DOM για εξαγωγή σε αρχείο XML τα περιεχόμενα της ανάλογης βάσης, δίνονται στην σελίδα των οδηγιών προς τους υποψήφιους συνεργάτες.

Το μόνο λοιπόν που χρειάζεται για μια τέτοια συνεργασία είναι η ύπαρξη ενός συγκεκριμένου αρχείου XML που περιγράφει αυτοκίνητα με οδηγό το συγκεκριμένο XSD, το αντίστοιχο site με αγγελίες και η συμπλήρωση μιας φόρμας με τα απαραίτητα στοιχεία και τις διευθύνσεις των XML ώστε να αυτοματοποιείται η διαδικασία της ανανέωσης από όλους τους συνεργάτες. Παράδειγμα τέτοιας εφαρμογής είναι και το Skroutz.gr

- **Αναζήτηση και Jquery**

Παραπάνω όταν κατασκευάζαμε την φόρμα αναζήτησης για το site “CARZ” συναντήσαμε το πρόβλημα της εξάρτησης των δεδομένων στο ζήτημα κατασκευαστή – μοντέλων. Η λύση η οποία βρέθηκε μόνο με χρήση της PHP είναι να κάνουμε πρώτα submit στην βάση και μετά να φορτώνουμε τα μοντέλα στο select σαν options και να επιλέγει μετά ο χρήστης το μοντέλο που τον ενδιαφέρει. Αυτό το ζήτημα το λύσαμε με ένα jQuery plugin που μπορούμε να βρούμε παραδείγματα και ορισμό στην διεύθυνση http://www.texotela.co.uk/code/jquery/select/.

Το plug-in αυτό μας εφόσον το κάνουμε include μας παρέχει κάποιες συναρτήσεις για την εισαγωγή, ταξινόμηση και εύρεση διαγραφή επιλογών σε select box,

την παρακάτω :

```
function populateModels() {

  var mkr=$('#maker option:selected').val();

$("#model").ajaxAddOption("Gmodels.php",{"maker" : mkr}, false);
}
```

Με την συνάρτηση **ajaxAddOption** περνάμε σαν παράμετρο ένα αρχείο php, και την μεταβλητή **mkr** η οποία έχει πάρει την τιμή του **select box** το οποίο είναι επιλεγμένος ο κατασκευαστής. Το php αρχείο αφού βρει στην βάση τα μοντέλα του κατασκευαστή

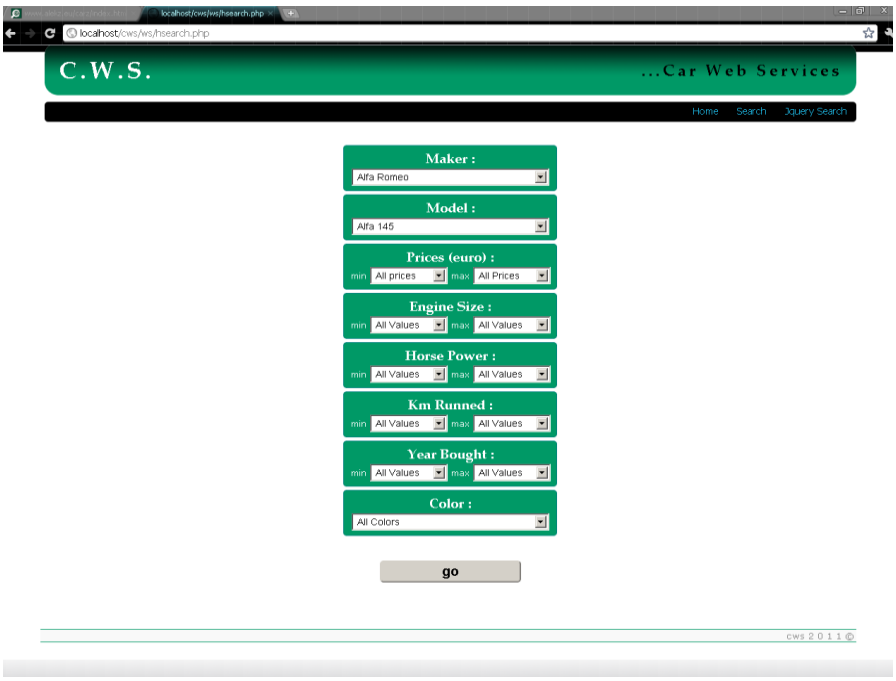
επιστρέφει σε μορφή jSon τα μοντέλα ώστε να εισαχθούν σαν options στο select box που προορίζεται για τα μοντέλα με χρήση του παρακάτω κώδικα και την συνάρτηση **json_encode();**.

```
$jsonString = json_encode($rows);

header('Content-type: application/json');

header("Content-Disposition: inline; filename=model-varities.json");

        echo $jsonString;
```

Η **ajaxAddOption()** παίρνει σαν παράμετρο το κωδικοποιημένο jSon αρχείο και τοποθετεί τις αξίες αλλά και τα labels στα options. Αναφορικά το jSon αρχείο που διαβάζει έχει την ακόλουθη μορφή (Τα μοντέλα που εμφανίζονται αφορούν τον κατασκευαστή Alfa Romeo):

```
[{"model_name":"Alfa 145"},{"model_name":"Alfa
146"},{"model_name":"Alfa 147"},{"model_name":"Alfa
155"},{"model_name":"Alfa 156"},{"model_name":"Alfa
159"},{"model_name":"Alfa 164"},{"model_name":"Alfa
166"},{"model_name":"Alfa 33"},{"model_name":"Alfa
75"},{"model_name":"Alfa
90"},{"model_name":"Alfasud"},{"model_name":"Brera"},{"model_name":"C
rossWagon"},{"model_name":"Giulia"},{"model_name":"Giulietta"},{"mode
l_name":"GT"},{"model_name":"GTV"},{"model_name":"Mito"},{"model_name
":"RZ\/SZ"},{"model_name":"Spider"},{"model_name":"SportWagon"},{"mod
el_name":"Sprint"}]
```

Με αυτόν τον τρόπο λύνουμε και το θέμα των αλληλοεξαρτώμενων δεδομένων κάνοντας με εφαρμογή τεχνολογιών τύπου **Ajax**, που όπως είδαμε και κατά την επεξήγηση τους χρησιμοποιούν περισσότερη πια κωδικοποίηση jSon παρά XML.



Εικόνα 6

4.2.1)Χτίζοντας έναν SOAP Client για τον Soap Server

Η μηχανή CWS υποστηρίζει δύο εφαρμογές live αναζήτησης με χρήση των υπηρεσιών ιστού. Μία από αυτές είναι ένας client έναν SOAP server ο οποίος παρέχει υπηρεσίες αναζήτησης στην βάση δεδομένων των αγγελιών του site «carz». Κάνοντας αναζήτηση λοιπόν χρησιμοποιώντας αυτήν την λειτουργία μας εμφανίζονται αποτελέσματα της βάσης σε πραγματικό χρόνο.

Ας δούμε λίγο τον κώδικα ενός τέτοιου client με κάποιες επεξηγήσεις.

```
//πρώτα παίρνουμε την βιβλιοθήκη για να χρησιμοποιήσουμε τις συναρτήσεις της.

<?php
require_once('nusoap.php');

//Δημιουργούμε τον πελάτη client του soap server που περιγράφεται από την
//Διεύθυνση που περνάμε σαν παράμετρο.
$client = new  soapclient('http://alekz.eu/ws/carws.php?wsdl', true);

//Κάνουμε έλεγχο για το αν δημιουργήθηκε ομαλά ο client
$err = $client->getError();
if ($err) {
    //Αν υπάρχει λοιπόν τέτοιο λάθος το εμφανίζουμε στην οθόνη
    echo '<h2>Constructor error</h2><pre>' . $err . '</pre>';
    //Σε αυτό το σημείο ξέρουμε ότι το call θα αποτύχει
}

//Μεταφέρουμε τα δεδομένα στην μεταβλητή $Car που είναι ορισμένη ότι περνάει σαν
//παράμετρος για την υπηρεσία ιστού.
//Στην συγκεκριμένη περίπτωση της παίρνουμε από την Global μεταβλητή POST
if(!empty($_POST["maker"])) { $make=$_POST["maker"]; } else { $make=""; }

$Car = array('maker' =>$make,
/*
*
*
*
*/
 "color"=>$color
);

//Εδώ καλούμε και την δημοσιευμένη συνάρτηση

$results = $client->call('car', array('Car' =>$Car));

//Ελέγχουμε αν υπάρχουν μηνύματα fault
if ($client->fault) {
    echo '<h2>Fault</h2><pre>';
    print_r($results);
    echo '</pre>';
} else {
    //Αν παρουσιάστηκαν λάθη και αν υπάρχουν τα εμφανίζουμε στην οθόνη.
    $err = $client->getError();
    if ($err) {
        echo '<h2>Error</h2><pre>' . $err . '</pre>';
    } else {
        //Αλλιώς εκτυπώνουμε τα αποτελέσματα της υπηρεσίας
        echo '<h2>Result</h2><pre>';
                print_r($results);
```

```
//Εδώ μπορούμε να δούμε και τα request και response στην κάθε αυτού της μορφή.
echo '<h2>Request</h2>';
echo '<pre>' . htmlspecialchars($client->request, ENT_QUOTES) . '</pre>';
echo '<h2>Response</h2>';
echo '<pre>' . htmlspecialchars($client->response, ENT_QUOTES) . '</pre>';

//Η και τα μηνύματα αποσφαλμάτωσης
echo '<h2>Debug</h2>';
echo '<pre>' . htmlspecialchars($client->debug_str, ENT_QUOTES) . '</pre>';
*/
?>
```

Ο client εδώ αναφέρεται σε κάποιο WSDL και δημιουργείται βάση αυτού. Από κει και πέρα το μόνο που πρέπει να κάνουμε είναι να κατασκευάσουμε την δομή $Car και να της διοχετεύσουμε τις επιθυμητές τιμές, εδώ μέσα από μία μεταβλητή $POST από κάποια φόρμα. Έπειτα την περνάμε σαν παράμετρο στην δημοσιευμένη συνάρτηση που καλούμε και εκτυπώνουμε τα αποτελέσματα που μας επιστρέφει. Σε περίπτωση ασυμβατότητας η σφαλμάτων παίρνουμε μηνύματα λάθους, ζήτησης και απάντησης τα οποία μπορούμε να αναλύσουμε και να δούμε κατά πώς είναι μέσα στην μορφολογία που τα συσκευάζει το SOAP με XML ετικέτες, καθώς και μηνύματα αποσφαλμάτωσης, μέσω των μεταβλητών $client->request, $client->response, $client->debug_str. Βέβαια η ανάγνωση τους και η ερμηνεία τους είναι αρκετά πολύπλοκα αλλά σε πολλές περιπτώσεις είναι και η μόνη λύση για τον κατασκευαστή.

4.2.2)Χτίζοντας έναν XML – RPC Client για τον XML – RPC Server

Η δεύτερη εφαρμογή που υποστηρίζει live αναζήτηση στο site CWS είναι ένας Client σε έναν XML- RPC server ο οποίος βρίσκεται επίσης στο site «CARZ». Η λειτουργία του είναι σχεδόν ίδια με τον SOAP client αλλά βλέπουμε ότι έχει πολύ ποιο χαλαρή σύνταξη. Βέβαια παραμένει το μειονέκτημα ότι δεν αναφέρεται σε κάποιο έγγραφο «περιγραφής» της υπηρεσίας «WSDL» οπότε η σύνταξη και οι τύποι δεδομένων είναι θέμα συνεννόησης των κατασκευαστών.

```
<?php
//Πρέπει να κάνουμε φυσικά include τις ανάλογες βιβλιοθήκες για να χρησιμοποιήσουμε
τις ανάλογες συναρτήσεις.
if (include("utils.php"))  {echo "TRUE :utils.php";} else { echo "FALSE
:utils.php";} echo "<br/>";

//Ορίζουμε μεταβλητές για την τοποθεσία και το $host που βρίσκεται η υπηρεσία
$host = "localhost";
$uri = "/ws-rpc/server_rpc.php";
```

```
//ελέγχουμε τις τιμές και τις αποθηκεύουμε στην μεταβλητή $Car που περνάμε σαν
παράμετρο στην κλήση της απομακρυσμένης συνάρτησης
if(!empty($_POST["maker"])) { $make=$_POST["maker"]; } else { $make=""; }

$Car = array('maker' =>$make,
/*
*/
 "color"=>$color
);

//Κάνουμε κλήση της απομακρυσμένης συνάρτησης και εκτυπώνουμε τα αποτελέσματα που
αποθηκεύονται στην μεταβλητή $results τα οποία και επιστρέφουμε
$results = xu_rpc_http_concise(
array(
'method' => "carz",
'args'  => $Car,
'host'  => $host,
'uri'  => $uri,
'port'  => 80
)
);

echo $results
?>
```

4.3) XML Client - Server εφαρμογές και η χρησιμότητα τους

Με τις εφαρμογές των τεχνολογιών υπηρεσιών ιστού καταφέρνουμε: Την ανταλλαγή δεδομένων μεταξύ συστημάτων μέσω μηχανισμών της γλώσσας XML είτε με παρέμβαση χρηστή είτε με δημοσίευση συναρτήσεων και εγγράφων περιγραφής των υπηρεσιών. Η τεχνολογία αυτή εξυπηρετεί Sites τα οποία λειτουργούν σαν portal παρόμοιων site τα οποία δημοσιεύουν τα δεδομένα τους σε μορφή XML, αλλά και Sites τα οποία λειτουργούν σαν Server και παρέχουν γενικότερα υπηρεσίες. Κάτι τέτοιο φυσικά ειδικά όσον αφορά την live αναζήτηση μπορούμε να καταλάβουμε ότι είναι πάρα πολύ πρακτικό και χρήσιμο ειδικά για υπηρεσίες για τις οποίες το real time είναι και το στοιχείο λειτουργιάς τους όπως η έρευνα και κράτηση εισιτηρίων, άλλων υπηρεσιών όπως ενοικίαση αυτοκινήτων ακόμα και περίπλοκες ίσως τραπεζικές εφαρμογές με την απαραίτητη ασφάλεια βέβαια.

Με την χρήση τους εκτός από την εύκολη ανταλλαγή δεδομένων από το ένα site στο άλλο οι υπηρεσίες ιστού μπορούν να υποσχεθούν και αρκετή ευκολία σε εφαρμογές ώστε να έχουν και αυτές με την σειρά τους πρόσβαση σε δεδομένα τα οποία βρίσκονται σε απομακρυσμένους servers χρησιμοποιώντας για άλλη μια φορά την ευκολία που μπορούν να δώσουν οι υπηρεσίες ιστού και η μεταφορά μέσω του standard πρωτοκόλλου HTTP. Τέτοιες εφαρμογές θα δούμε παρακάτω.

5 Το Μέλλον στις Εφαρμογές και στις Εφαρμογές Διαδικτύου.

Σε αυτό το κομμάτι θα ασχοληθούμε πολύ σύντομα και περιληπτικά στην εξέλιξη των υπολογιστικών συστημάτων όσον αφορά συσκευές τηλεφώνων τα οποία δείχνουν την τάση να αντικαταστήσουν βαθμιαία τα απλά κινητά τηλέφωνα εφόσον παρέχουν πάρα πολλές εφαρμογές και λειτουργίες.

Παρακολουθώντας κάποιος την εξέλιξη των συστημάτων τα τελευταία χρόνια αντιλαμβάνεται ότι μεγάλη ανάπτυξη υπάρχει στις εφαρμογές για «έξυπνα κινητά» η αλλιώς smartphones.

Οι περισσότερες από αυτές τις εφαρμογές φυσικά χρησιμοποιούν το internet και απαιτούν πρόσβαση στα δεδομένα του. Φυσικά μπορούμε και έχουμε και πρόσβαση στις διάφορες ιστοσελίδες μέσω των browsers που έχουν και πολλά site έχουν διαφορετικό στήσιμο ώστε να είναι ποιο ευανάγνωστο σε browsers κινητών τηλεφώνων. Και όμως η κατασκευή ανεξάρτητων εφαρμογών που προϋποθέτει επικοινωνία με το ίντερνετ για αυτές τις συσκευές δεν έχει σταματήσει να αναπτύσσεται και δείχνει ότι θα συνεχίσει να κερδίζει χώρο. Ολοένα και περισσότερα μεγάλα site αναζήτησης δημοσιεύουν τέτοιες εφαρμογές που μπορούμε να κατεβάσουμε και να τρέξουμε από τα κινητά μας. Amazon, eBay, XO, tvxs ακόμα και μικρότερα site που έχουν την δυνατότητα να αναπτύξουν τέτοιες εφαρμογές. Οι εφαρμογές αυτές συνήθως ανεβαίνουν στα ανάλογα App Store κάθε εταιρίας και είναι downloadable από τους χρήστες.

Σε αυτές βέβαια τις εφαρμογές οι κατασκευαστές καλούνται να λύσουν προβλήματα τα οποία όσο αναπτύσσεται η τεχνολογία έχουν ξεπεραστεί για τις Desktop εφαρμογές. Προβλήματα τέτοια είναι η μνήμη, η χωρητικότητα δεδομένων, η ταχύτητα και άλλα προβλήματα τα οποία φυσικά παρουσιάζονται (προς το παρόν) από την έλλειψη χώρου για ποιο ισχυρό hardware, αλλά η εξέλιξη μας έχει δείξει ότι αυτό το πρόβλημα είναι μάλλον προσωρινό, υπαρκτό όμως προς το παρόν.

5.1) Λειτουργικά Συστήματα και Smart phones

Τα λεγόμενα έξυπνα κινητά η και smartphones όπως αποκαλούνται είναι η μετεξέλιξη και ο συνδυασμός μεταξύ των κινητών τηλεφώνων και των PDA (Personal Digital Assistant), συνήθως βιομηχανικές συσκευές που λειτουργούν για απομακρυσμένη σύνδεση σε βιομηχανικές εφαρμογές. Συνδυάζουν τις αυξανόμενες υπολογιστικές δυνατότητες και την συνδεσιμότητα και από τους δύο αυτούς τύπους συσκευών. Υπάρχουν πολλές συσκευές άλλα και πολλά λειτουργικά συστήματα τα οποία είναι κατασκευασμένα για τέτοιου τύπου συσκευές. Τα ποιο γνωστά από αυτά είναι τα: BlackBerry, Windows Mobile, Symbian, Android, and iPhone OS.

Εμείς θα αναφερθούμε περισσότερο σε εφαρμογές οι οποίες υποστηρίζονται από τα λειτουργικά συστήματα της Microsoft όπως τα Windows Mobile και τα Windows Phone.

Ιστορικά λοιπόν η Microsoft ξεκινάει με τα Windows Mobile τα οποία είναι λειτουργικά συστήματα για κινητά αλλά και για συσκευές βιομηχανικών εφαρμογών. Από τα ποιο καθιερωμένα είναι τα Windows Mobile 5.0 και 5.5 τα οποία και θεωρούνται και τα ποιο ελαφριά από όλα τα άλλα (κάτι το οποίο αναφέραμε ότι είναι μείζον ζήτημα στον τομέα αυτό) και για το λόγο αυτό συνεχίζουν να χρησιμοποιούνται ακόμα από αρκετές επιχειρήσεις. Οι εκδώσεις αυτές μέχρι σήμερα φτάνουν και τα Windows Mobile 6.0 και 6.5 όπου ανακοινώνει και η Microsoft ότι κατασκευάζει τα Windows Phone. Βέβαια πιστή με την πολιτική που ακολουθεί η εταιρία αυτή εδώ και χρόνια σταματάει και την υποστήριξη της για όλες τις προηγούμενες αλλά και κόβει και την συμβατότητα. Μια πολιτική η οποία ανά τα έτη έχει προκαλέσει μεγάλη δυσαρέσκεια σε κύκλους καταναλωτών αλλά και κατασκευαστών εφαρμογών.

Φαίνεται καθαρά από τις τάσεις στην αγορά ότι η κυριαρχία της Microsoft, ειδικά σε αυτόν τον τομέα δεν θα έχει την ίδια μοίρα με τους προσωπικούς υπολογιστές στους οποίους καθόταν μόνη στην κορυφή για χρόνια. Το Android της Google και το iPhone OS της Apple κερδίζουν ολοένα και περισσότερους φανατικούς οπαδούς αλλά και κατασκευαστές κινητών τηλεφώνων. Αυτό ενισχύεται και με τις εφαρμογές τους στα App Store της κάθε εταιρίας που χαρακτηρίζουν κατά πολύ και το κάθε λογισμικό.

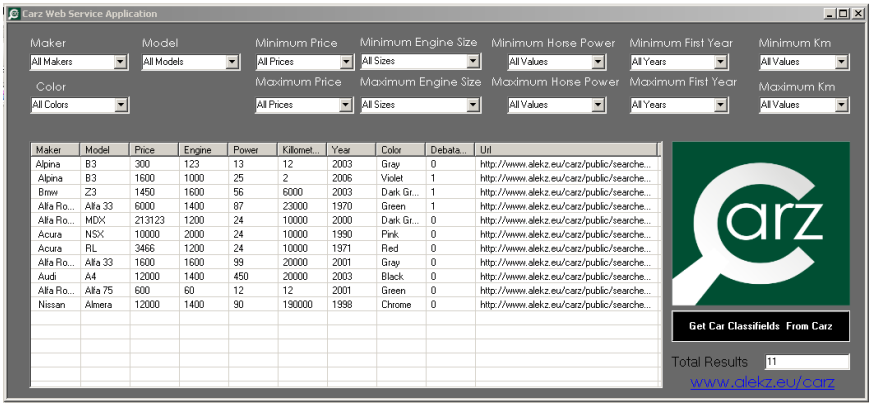
Το λογισμικό όμως με το οποίο θα ασχοληθούμε στο επόμενο μέρος είναι τα windows mobile. Βέβαια έγιναν απόπειρες και πειραματισμοί με άλλα λειτουργικά συστήματα, αλλά όπως προαναφέραμε η εκτεταμένη υποστήριξη της πλατφόρμας SOAP από την Microsoft, οι γνώσεις μας στην γλώσσα C++ και η οικειότητα με το Visual Studio μας έδωσε την δυνατότητα να υλοποιήσουμε μια εφαρμογή αναζήτησης μικρών αγγελιών η οποία είναι Client στο Carz website.

5.2) Χτίζοντας το App για Windows Mobile.

- **Περιβάλλον εργασίας και γλώσσα υλοποίησης.**

Το περιβάλλον το οποίο εργαστήκαμε για την υλοποίηση της εφαρμογής είναι το «Microsoft Visual Studio 2008» και η εφαρμογή σχεδιάστηκε για λειτουργικά συστήματα «Windows Mobile». Η γλώσσα υλοποίησης που επιλέξαμε να χρησιμοποιήσουμε είναι η C# σαν μετεξέλιξη της C++ την οποία και έχουμε διδαχτεί θεωρώντας την σαν ποιο σύγχρονη και λειτουργική. Το πλαίσιο υλοποίησης «framework» το όποιο χρησιμοποιήσαμε είναι το «Microsoft .NET framework 3.5 Compact» το οποίο και χρησιμοποιείται για mobile εφαρμογές σαν ποιο ελαφρύ από τα «μη – compact» πλαίσια τα οποία και χρησιμοποιούνται για Desktop εφαρμογές όπως το Microsoft .NET framework 4.

- **Mobile και Desktop εφαρμογή.**



Εικόνα 7

Η εφαρμογή πρώτα κτίστηκε σαν Desktop εφαρμογή (Η οποία διατίθεται επίσης από το site) για κάποιους πολύ απλούς λόγους. Ο πρώτος λόγος ήταν η έλλειψη εξοικείωσης με την όλη mobile φιλοσοφία, θέμα το οποίο απαιτούσε χρόνο και γινόταν παράλληλα. Ο δεύτερος λόγος είναι ότι οι εξομοιωτές συσκευών mobile είναι πάρα πολύ αργοί, κάτι το οποίο κάνει το debugging που είναι πάρα πολύ αργό και δύσκολο να εφαρμοστεί με συχνές αλλαγές στον κώδικα. Προϋποθέσεις για την πραγματοποίηση μιας τέτοιας εφαρμογής είναι: εξομοιωτές mobile συσκευών καθώς και σύνδεση με Active Sync ώστε να έχει ο εξομοιωτής σύνδεση μέσω του υπολογιστή με το internet για να μπορεί να λειτουργήσει και η Υπηρεσία Ιστού και να επιστρέψει δεδομένα. Βέβαια κατά την μετάφραση από το ένα framework στο άλλο έπρεπε να αλλάξουν κάποιες συναρτήσεις οι οποίες δεν υποστηρίζονταν από το compact framework.

- **Σχεδιάζοντας την εφαρμογή.**

Η φιλοσοφία της εφαρμογής είναι να επιλέγει ο χρήστης κάποια κριτήρια αναζήτησης για ένα αυτοκίνητο και αυτά τα κριτήρια να στέλνονται σαν αίτημα προς τον SOAP server. Ο server θα επεξεργάζεται αυτά τα στοιχεία και θα αποστέλλει σαν απάντηση τα στοιχεία των αγγελιών οι οποίες ικανοποιούν τα κριτήρια. Τα δεδομένα των αγγελιών αυτών θα εμφανίζονται σε μία λίστα στην εφαρμογή. Να επισημάνουμε ότι ο σκοπός μας εδώ δεν η εμφάνιση των δεδομένων αλλά η μελέτη του μηχανισμού ανταλλαγής τους.

Τα κριτήρια αναζήτησης θα επιλέγονται από τον χρήστη όπως ακριβώς και στην online μηχανή αναζήτησης. Τα δεδομένα θα βρίσκονται τοποθετημένα σε Combo Boxes (αντίστοιχα των SelectBoxes της HTML) απ' όπου και θα γίνεται επιλογή τους.

- **Combo Boxes και Select Boxes**

Το πρώτο πρόβλημα το οποίο συναντήσαμε ήταν με την απόδοση τιμών στα combo boxes. Η κλάση των combo boxes πρέπει να πάρει σαν όρισμα αντικείμενα. Εμείς όμως θέλουμε να τους δώσουμε αξίες και μάλιστα άλλη αξία να επιλέγεται και άλλη ετικέτα να φαίνεται στον χρήστη (name & value). Αυτό λύθηκε με την δημιουργία μίας κλάσης η οποία

φτιάχνει αντικείμενα τύπου ComboBoxItems και μας επιτρέπει να μεταχειριζόμαστε τα combo boxes ακριβώς όπως και τα select boxes στην HTML. Ο κώδικας που φαίνεται παρακάτω είναι και η υλοποίηση της κλάσης αυτής, αλλά δίνεται και παράδειγμα ορισμού των δεδομένων του Item.

```
public class CustomComboBoxItem
        {
    //Κατασκευάζουμε τα δεδομένα να της «ετικέτας» του αντικειμένου.
            private string _contents;
            public string contents { get { return _contents; } set {
_contents = value; } }
 // Εδώ τοποθετούμε την κρυφή ετικέτα του αντικείμενου
            private object _tag;
            public object tag { get { return _tag; } set { _tag = value; }
}
            public CustomComboBoxItem(string contents, object tag)
            {
                this._contents = contents;
                this._tag =tag;
            }
         // Εδώ ορίζουμε ότι θα εμφανίζεται μόνο η ετικέτα του αντικειμένου κατά
την κλήση του.
                 public override string ToString() { return _contents; }
        }

// Εδώ φαίνεται η πρόσθεση ενός καινούργιου αντικειμένου που δίνεται η ετικέτα και
// η αξία του. Συγκεκριμένα στο Combo Box που καθορίζουμε τους κατασκευαστές.

this.comboBox1.Items.Add(new CustomComboBoxItem("Alfa Romeo", 1112));
this.comboBox1.Items.Add(new CustomComboBoxItem("Bmw", 1120));

 Original examplehttp://www.techerator.com/2011/04/how-to-create-a-html-style-combobox-in-wpf-
                                                                                and-c/
```

- **Κατασκευαστές – Μοντέλα, εξάρτηση και αποθήκευση δεδομένων.**

Όπως και στις προηγούμενες μηχανές αναζήτησης που υλοποιήσαμε συναντήσαμε και δω το θέμα με την επιλογή των μοντέλων ανάλογα με τους κατασκευαστές. Οι κατασκευαστές μπαίνουν σαν options μέσα στον κώδικα. Τα μοντέλα όμως είναι εξαρτημένα ανάλογα με τον κατασκευαστή. Βέβαια στην C# έχουμε πολύ μεγαλύτερη ευκολία και ευελιξία κινήσεων για δυναμική φόρτωση δεδομένων. Το μόνο που πρέπει να κάνουμε είναι να γράψουμε τον ανάλογο κώδικα στο on-change event του combo box των κατασκευαστών και να φορτώσουμε τα μοντέλα. Αυτό όμως προϋποθέτει κάποιο μηχανισμό αποθήκευσης των δεδομένων των μοντέλων όπως μία βάση δεδομένων η ένα αρχείο XML τα οποία όμως θα έπρεπε να «κατεβούν» μαζί με την εφαρμογή για να δουλέψει και επίσης παρουσιάζουν κάποια μειονεκτήματα όπως: Όγκος εφαρμογής, τεχνικές γνώσεις εκτός αντικειμένου, έλλειψη ενημέρωσης για καινούργια μοντέλα και ανάγκη για update. Η λύση σε αυτά είναι για άλλη μια φορά μία υπηρεσία ιστού.

- **Υπηρεσία Ιστού για φόρτωση δεδομένων.**

Δημιουργήσαμε λοιπόν μία ακόμα υπηρεσία ιστού την οποία καλούμε μετά την επιλογή του κατασκευαστή από τον χρήστη. Αυτή η υπηρεσία ιστού αποστέλλει σαν παράμετρο τον κατασκευαστή στον server και λαμβάνει όλα τα μοντέλα σαν ένα string. Το string επιλέχτηκε σαν μορφή διότι δεν ξέρουμε τον εξαρχής αριθμό τον μοντέλων. Τα μοντέλα διαχωρίζονται από έναν ειδικό χαρακτήρα και με μια συνάρτηση ανάλογη της «explode» την «Split» το string διαχωρίζεται και μπαίνουν τα μοντέλα σαν options στο combo box που επιλέγονται μοντέλα. Η υπηρεσία αυτή καλείται κάθε φορά που αλλάζει η επιλογή του κατασκευαστή, και η ανάλογη αξία της επιλογής αυτής περνιέται σαν παράμετρος στην υπηρεσία.

- **Αναφορά και αναγνώριση υπηρεσίας Ιστού.**

Οι υπηρεσίες ιστού που προαναφέραμε είναι κατασκευασμένες με την SOAP αρχιτεκτονική και υποστηρίζονται από WSDL. Πώς όμως μπορούμε να τις εμπλέξουμε στην εφαρμογή μας και στο Visual Studio;

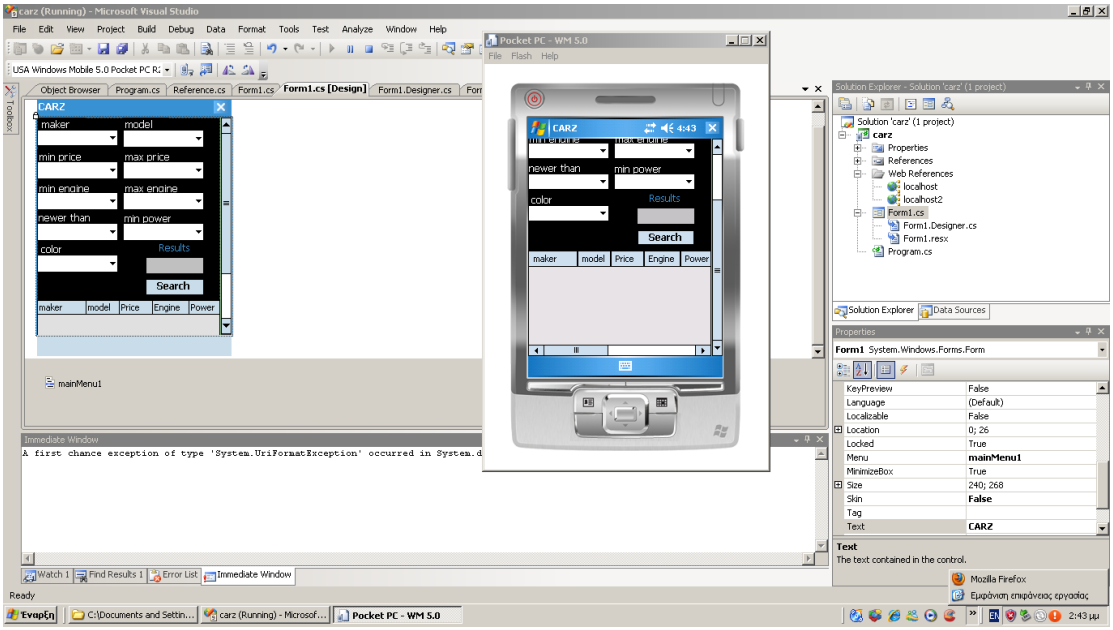
Όταν ανοίγουμε μια καινούργια εφαρμογή αφού επιλέξουμε την πλατφόρμα και τα εισαγωγικά στοιχειά τα οποία θα αφορούν την εφαρμογή μας βλέπουμε στο δεξί μας χέρι το «Solution Explorer». Στο «Solution Explorer» βλέπουμε την εφαρμογή σε δομή δένδρου. Στην δομή αυτή βλέπουμε την ετικέτα «References». Με δεξί κλικ στην ετικέτα αυτή επιλέγουμε «Add Web Reference» και μας εμφανίζεται ένα παράθυρο διαλόγου. Στο παράθυρο διαλόγου αυτό πρέπει να δώσουμε και την διεύθυνση του WSDL πχ.

```
http://www.alekz.eu/ws/carws.php?wsdl
```

Αυτομάτως το «Visual Studio» διαβάζει το WSDL και αναγνωρίζει τις μεταβλητές οι οποίες ορίζονται από το WSDL και εφόσον ορίσουμε το service τις μεταβλητές και το reply της υπηρεσίας δουλεύουμε κανονικά σαν να είναι η υπηρεσία κλάση μέσα στον κώδικα μας. Καθορίζουμε τα δεδομένα των μεταβλητών πχ Car.maker καλούμε την υπηρεσία με την συνάρτηση .reply();

Το «Visual Studio» παρέχει πολύ μεγάλη ευκολία (απ' ότι είδαμε) τουλάχιστον όσον αφορά υπηρεσίες ιστού και την κατανάλωση τους εφόσον όμως είναι χτισμένες σε Soap και έχουν υποστήριξη WSDL. Σε αυτό αναφερθήκαμε και πρωτίστως και αναφέραμε ότι η Microsoft υποστηρίζει πάρα πολύ την SOAP αρχιτεκτονική, παρέχοντας μεγάλη ευκολία στους κατασκευαστές της μέσω των εργαλείων που προσφέρει.

- **Αποτέλεσμα – λειτουργία της εφαρμογής**



Εικόνα 8

Η εφαρμογή στην τελική της μορφή αποτελείται από Combo Boxes από τα οποία ο χρήστης επιλέγει κριτήρια, ένα list view στο οποίο εμφανίζονται τα αποτελέσματα της αναζήτησης και ένα κουμπί υποβολής του ερωτήματος, και υλοποιείται με την «κατανάλωση» δύο υπηρεσιών ιστού. Η μία καλείται όταν επιλέξουμε κατασκευαστή, η αξία του οποίου αποστέλλεται σαν παράμετρος και μας επιστρέφει σαν αποτελέσματα τα μοντέλα του εκάστοτε κατασκευαστή. Η δεύτερη καλείται όταν πατάμε το κουμπί αναζήτησης και περνά σαν παράμετρο έναν πίνακα με τα κριτήρια αναζήτησης. Τα αποτελέσματα εμφανίζονται στο list view και αναγράφεται η ποσότητα τους σε ένα text box. Όπως και στις προηγούμενες μηχανές αναζήτησης αν κάποιο κριτήριο δεν επιλεγεί τότε επιλέγονται όλες οι τιμές του κριτηρίου αυτού. Το μόνο προβλήτα που μπορεί να συναντήσουν στην λειτουργία τους εφαρμογές τέτοιου είδους είναι η πρόσβαση τους στο διαδίκτυο για το λόγο αυτό η εφαρμογή υλοποιήθηκε με την λειτουργία Try Catch ώστε όταν μια διαδικασία κόβεται από κάποιο σφάλμα να εμφανίζει και το ανάλογο μήνυμα

Εν κατακλείδι η χρησιμοποίηση Υπηρεσιών Ιστού βασισμένες στην XML είναι ένα πάρα πολύ χρήσιμο εργαλείο για ζωντανή ανταλλαγή δεδομένων και για εφαρμογές οι οποίες αναζητούν η και καταχωρούν δεδομένα σε κάποιο server και μπορούν να είναι στημένες σε άλλες ποικίλες πλατφόρμες για διάφορες συσκευές και λειτουργικά συστήματα.

6 Πηγές – Αναφορές

- http://www.lynda.com/
- Php & My Sql Bible
- O'Reilly Media
- Jquery select boxes http://www.texotela.co.uk/code/jquery/select/
- http://www.scottnichol.com/soap.htm
- http://www.techerator.com/2011/04/how-to-create-a-html-style-combobox-in-wpf-and-c/
- http://msdn.microsoft.com
- http://www.w3schools.com/
- http://www.php.net/