ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΔΙΑΣΥΝΔΕΣΗ ΧΑΡΤΩΝ GOOGLE ΜΕ ΤΗΝ ΤΕΧΝΟΛΟΓΙΑ .NET

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΠΑΝΑΓΙΩΤΗ ΚΑΡΒΟΥΝΗ

Επιβλέπων : Γεώργιος Στασινόπουλος
Καθηγητής Ε.Μ.Π.

Αθήνα, Απρίλιος 2009

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

ΔΙΑΣΥΝΔΕΣΗ ΧΑΡΤΩΝ GOOGLE ΜΕ ΤΗΝ ΤΕΧΝΟΛΟΓΙΑ .NET

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΠΑΝΑΓΙΩΤΗ ΚΑΡΒΟΥΝΗ

Επιβλέπων : Γεώργιος Στασινόπουλος
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την …

.....	.....	.....
Γ. Στασινόπουλος Καθηγητής Ε.Μ.Π.	Μ. Θεολόγου Καθηγητής Ε.Μ.Π.	Ε. Συκάς Καθηγητής Ε.Μ.Π.

Αθήνα, Απρίλιος 2009

.....

Παναγιώτης Α. Καρβούνης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Η παρούσα διπλωματική εργασία αναπτύσσει την μεθοδολογία διασύνδεσης των χαρτών Google με την τεχνολογία ASP.NET.

Αρχικά γίνεται ανάγνωση των πληροφοριών μιας βάσης δεδομένων Microsoft SQL Server που περιέχει και γεωγραφικά δεδομένα και στη συνέχεια με τη χρήση του Google Maps API παρουσιάζονται σε έναν χάρτη Google που έχει ενσωματωθεί σε μία ιστοσελίδα.

Για την επίτευξη του σκοπού αυτού γίνεται χρήση της τεχνολογίας ASP.NET AJAX που επιτρέπει την κλήση μεθόδων ASP.NET που βρίσκονται στον εξυπηρετητή από τη γλώσσα προγραμματισμού JavaScript, στην οποία βασίζεται και το Google Maps API.

Στην παρούσα διπλωματική εργασία η βάση δεδομένων που χρησιμοποιήθηκε περιέχει πληροφορίες για σταθμούς ηλεκτρικής ενέργειας. Με την κατάλληλη επεξεργασία των δεδομένων της βάσης και τη χρήση μίας διαδικτυακής υπηρεσίας γίνεται εφικτή η εμφάνιση περιοχών με ηλεκτρικούς σταθμούς σε ένα χάρτη και επιπρόσθετα στοιχεία για τους σταθμούς κάθε περιοχής.

Για την καλύτερη εμφάνιση του τελικού αποτελέσματος γίνεται επίσης χρήση της τεχνολογίας CSS.

Το αποτέλεσμα της εργασίας είναι μία σύγχρονη ιστοσελίδα που ανταποκρίνεται γρήγορα στις απαιτήσεις του χρήστη.

Το τεχνικό μέρος αποτελείται από το Λειτουργικό Microsoft Windows, το .NET Framework 3.5 καθώς και τον εξυπηρετητή της Microsoft IIS. Για την εμφάνιση της ιστοσελίδας είναι απαραίτητος περιηγητής που υποστηρίζει JavaScript και Χάρτες Google.

Συνοψίζοντας παρούσα διπλωματική εργασία αποτελεί εισαγωγή στη διασύνδεση ASP.NET και Χαρτών Google και μπορεί να αποτελέσει χρήσιμο εργαλείο για πιο προχωρημένες υλοποιήσεις στο συγκεκριμένο θέμα.

Λέξεις Κλειδιά: Χάρτες Google, ASP.NET, .NET Framework, Περιηγητής, AJAX, C#, JavaScript, Διασύνδεση, Επέκταση, Γεωγραφικές Συντεταγμένες, Microsoft

Abstract

This thesis presents a methodology for the integration between Google Maps and the ASP.NET technology.

Initially using ASP.NET we read data from a given Microsoft SQL Server database that contains also geographic data and then with the use Google Maps API they are presented in a Google map that has been incorporated in a web page.

For the achievement of this aim we use of technology ASP. NET AJAX that allows the call of ASP.NET methods that are located in the server using the JavaScript programming language, on which the Google Maps API is based.

In the current thesis the database that was used contains information on stations of electric energy. With the suitable treatment of data of the database and the use of a web service is possible the presentation of regions with electric stations in map and additional information on the stations of each region.

CSS technology is used in order to have better site appearance.

The result of thesis is a modern web page that corresponds fast in the requirements of user.

The technical part is constituted by the Operating System Microsoft Windows, the .NET Framework 3.5 as well as the Microsoft IIS Server. To view the page is needed a Web Browser that supports JavaScript and Google Maps API.

To summarize, current thesis is an introduction of Google Maps and ASP.NET integration and can be used as a useful tool to achieve harder tasks on this particular subject.

Keywords: Google Maps, ASP. NET, .Net Framework, Browser, AJAX, C#, JavaScript, Integration, Geographic Coordinates , Microsoft

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω τον καθηγητή κ. Γεώργιο Στασινόπουλο που μου έδωσε την ευκαιρία να ασχοληθώ με το συγκεκριμένο αντικείμενο καθώς και για την πολύτιμη συνεισφορά του στην ολοκλήρωση της εργασίας.

Επίσης θα ήθελα να ευχαριστήσω τον μεταπτυχιακό κ. Βίκτορ Ματεεβίτσι για την βοήθεια και τη συνδρομή του στην υλοποίηση της εργασίας.

Κυρίως όμως θα ήθελα να αφιερώσω αυτή την εργασία στην οικογένεια μου για τη στήριξή της, υλική και πνευματική για τη διεκπεραίωση των σπουδών μου.

Πίνακας περιεχομένων

1 Εισαγωγή – Τεχνολογίες

Στο συγκεκριμένο κεφάλαιο γίνεται μία παρουσίαση των βασικών τεχνολογιών που χρησιμοποιήθηκαν για την υλοποίηση του Περιηγητή Σταθμών Ηλεκτρικής Ενέργειας. Συγκεκριμένα οι τεχνολογίες που θα παρουσιαστούν είναι οι παρακάτω:

- Microsoft .NET Framework και ειδικότερα ASP.NET
- Η γλώσσα προγραμματισμού C#
- Η γλώσσα προγραμματισμού JavaScript
- Η τεχνολογία ASP.NET AJAX
- Microsoft SQL Server
- Microsoft IIS Server

1.1 Microsoft .NET Framework και ASP.NET

1.1.1 Microsoft .NET Framework

Το Microsoft.NET Framework είναι ένα πλαίσιο ανάπτυξης λογισμικού για την διαχείριση των πλέον συνηθισμένων προβλημάτων προγραμματισμού. Περιλαμβάνει μια εικονική μηχανή που διαχειρίζεται την εκτέλεση των προγραμμάτων που γράφονται συγκεκριμένα για το Framework. Το Framework είναι μια βασική προσφορά της Microsoft και προορίζεται να χρησιμοποιηθεί για τις περισσότερες νέες εφαρμογές που δημιουργούνται για το λειτουργικό σύστημα της Microsoft, τα Windows.

Η βασική βιβλιοθήκη παρέχει μια μεγάλη ποικιλία χαρακτηριστικών που αφορούν:

- Διασύνδεση χρήστη
- Διαχείριση Δεδομένων
- Διασύνδεση με Βάσεις Δεδομένων
- Κρυπτογράφηση
- Ανάπτυξη Διαδικτυακών Εφαρμογών
- Αριθμητικούς Αλγορίθμους
- Επικοινωνία Δικτύων

Όλα τα παραπάνω χρησιμοποιούνται από τους προγραμματιστές για τη δημιουργία πανίσχυρων εφαρμογών.

Τα προγράμματα που γράφονται με τη χρήση του Framework εκτελούνται σε ένα περιβάλλον λογισμικού που διαχειρίζεται τις απαιτήσεις χρόνου εκτέλεσης.

Επίσης μέρος του Framework, αυτό το περιβάλλον χρόνου εκτέλεσης είναι γνωστό ως Common Language Runtime (CLR). Το CLR παρέχει μια εικονική μηχανή εκτέλεσης εφαρμογών έτσι ώστε οι προγραμματιστές δεν χρειάζονται να εξετάσουν τις ικανότητες της συγκεκριμένης ΚΜΕ που θα εκτελέσει το πρόγραμμα. Το CLR παρέχει επίσης άλλες σημαντικές υπηρεσίες όπως η ασφάλεια, η διαχείριση μνήμης, και ο χειρισμός εξαιρέσεων. Η βασική βιβλιοθήκη και το CLR συνθέτουν μαζί το .net Framework. Οι βασικές γλώσσες που χρησιμοποιούνται είναι οι VB.NET, C#, C++/CLI, J#.

Η έκδοση 3.0 του Framework συμπεριλαμβάνεται στα Windows Server 2008 και στα Windows Vista. Η τρέχουσα έκδοση του Framework μπορεί να εγκατασταθεί στα Windows XP και η οικογένεια λειτουργικών συστημάτων Windows Server 2003. μειωμένης . Μία πιο "ελαφριά" έκδοση είναι διαθέσιμη για τα κινητά τηλέφωνα που χρησιμοποιούν Windows Mobile.

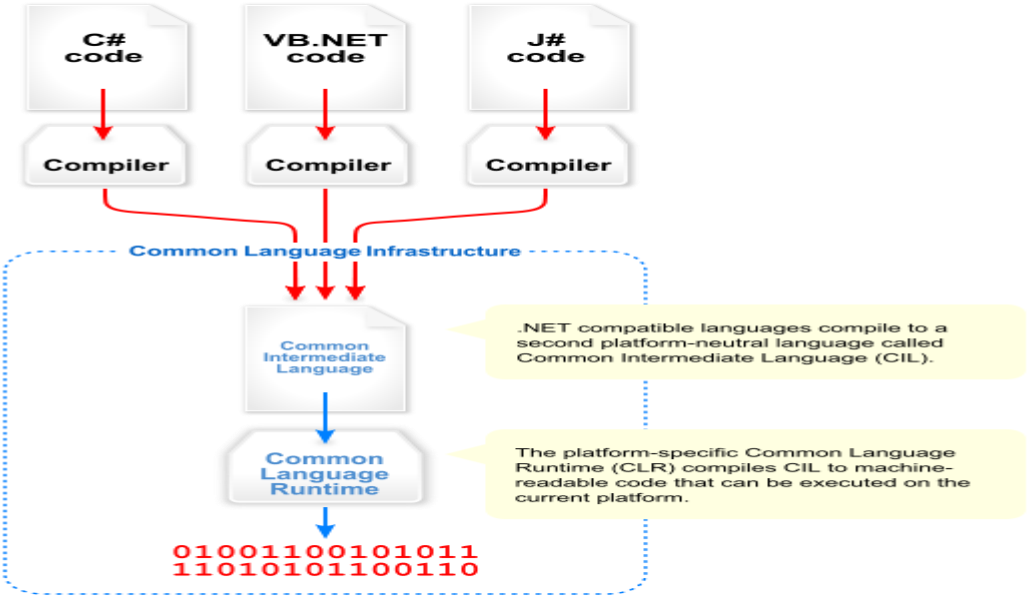
Τα βασικά χαρακτηριστικά του .NET Framework είναι :

- **Διαλειτουργικότητα :** Επειδή συνήθως απαιτείται η αλληλεπίδραση μεταξύ των νέων και παλαιότερων εφαρμογών το Framework παρέχει τα μέσα για να προσεγγιστεί η λειτουργία που εφαρμόζεται στα προγράμματα που εκτελούνται έξω από αυτό. Η επικοινωνία την παλαιότερη τεχνολογία COM παρέχεται μέσω των namespaces System.Runtime.InteropServices και EnterpriseServices του Framework
- **Κοινή μηχανή χρόνου εκτέλεσης :** Όλα τα προγράμματα εκτελούνται υπό την επίβλεψη του CLR, το οποίο εξασφαλίζει σωστή λειτουργία
- **Ανεξαρτησία Γλώσσας Προγραμματισμού :** Περιέχεται κοινό σύστημα τύπων και μπορεί να χρησιμοποιηθεί οποιαδήποτε από της γλώσσες του Framework
- **Βασική Βιβλιοθήκη (BCL) :** Η βασική βιβλιοθήκη του Framework είναι κοινή για όλες τις γλώσσες του. Προσφέρει οτιδήποτε χρειάζεται για λειτουργίες όπως ανάγνωση - εγγραφή σε αρχεία, χειρισμός γραφικών, χρήση βάσεων δεδομένων και XML
- **Εύκολη Εγκατάσταση Προγραμμάτων :** Η εγκατάσταση των υλοποιημένων εφαρμογών είναι πολύ απλή διαδικασία για κάθε υπολογιστή.

- **Ασφάλεια :** Οι εφαρμογές είναι ασφαλείς από χαρακτηριστικά κακόβουλου λογισμικού

- **Φορητότητα :** Οι εφαρμογές που υλοποιούνται με τη χρήση του Framework είναι δυνατό να εκτελεστούν σε κάθε σύστημα που το έχει εγκατεστημένο ανεξάρτητα από τα χαρακτηριστικά του συστήματος

Ορισμένα στοιχεία της αρχιτεκτονικής του .NET Framework φαίνονται στις παρακάτω εικόνες:



Εικόνα 1.1 : CLI- Σύστημα Γλωσσών

Namespaces in the BCL[9]
System
System. CodeDom
System. Collections
System. Diagnostics
System. Globalization
System. IO
System. Resources
System. Text
System. Text.RegularExpressions

Εικόνα 1.2 : Τα βασικά Namespaces

1.1.2 ASP.NET

Το ASP.NET είναι ένα Framework το οποίο βασίζεται στο Microsoft.NET Framework και χρησιμοποιείται για τη δημιουργία δικτυακών τόπων, δικτυακών εφαρμογών και διαδικτυακών υπηρεσιών. Κυκλοφόρησε από τη Microsoft τον Ιανουάριο του 2002 και αποτελεί διάδοχο της παλαιότερης τεχνολογίας ASP. Προφανώς βασίζεται και αυτό στο CLR και οι προγραμματιστές μπορούν να χρησιμοποιούν οποιαδήποτε από τις γλώσσες του .NET Framework. Το ASP.NET χρειάζεται τον Internet Information Services Server (IIS) της Microsoft.

Μερικά από τα βασικά χαρακτηριστικά του ASP.NET είναι τα παρακάτω:

- **Pages – Σελίδες :** Αποτελούν το βασικό κομμάτι κάθε δικτυακού τόπου. Έχουν επέκταση aspx και περιέχουν κώδικα κυρίως (X)HTML με τα στατικά και δυναμικά δεδομένα που επιθυμούν οι προγραμματιστές. Δυναμικός κώδικας περιέχεται μεταξύ <% και %>
- **Code Behind Model :** Συνίσταται ο δυναμικός κώδικας του προγράμματος να τοποθετείται σε ξεχωριστό αρχείο από τη σελίδα. Η επέκταση αυτών των σελίδων είναι aspx.vb, aspx.cs ανάλογα με τη γλώσσα προγραμματισμού που χρησιμοποιείται. Με τη συγκεκριμένη τεχνική ο προγραμματιστής χειρίζεται γεγονότα όπως η φόρτωση της σελίδας, το πάτημα ενός κουμπιού και άλλα σχετικά με πιο αποδοτικό και εύχρηστο τρόπο. Το Code Behind Model αποτελεί βασική διαφορά από την παλαιότερη τεχνολογία της Microsoft, τις σελίδες ASP. Με το συγκεκριμένο μοντέλο μπορεί ο σχεδιαστής να ασχολείται μόνο με το σχεδιασμό της σελίδας και το οπτικό κομμάτι, ενώ ο προγραμματιστής μόνο με το χειρισμό των γεγονότων και τη λειτουργία της σελίδας
- **User Controls – Στοιχεία Ελέγχου :** Είναι δυνατή η δημιουργία επαναχρησιμοποιούμενων στοιχείων έλεγχου για το χειρισμό της ίδιας λειτουργίας. Τα User Controls κληρονομούν την κλάση WEB.UI.Controls και ο προγραμματιστής μπορεί να ορίζει σε αυτά ιδιότητες, μεθόδους και γεγονότα. Τα User Controls μπορούν να ενσωματώνονται και σε βιβλιοθήκες DLL
- **Rendering Technique – Τεχνική Απόδοσης :** Τα πραγματικά αιτήματα για τη σελίδα από τους χρήστες υποβάλλονται σε επεξεργασία μέσω διάφορων βημάτων. Κατ' αρχάς, κατά την έναρξη, εκτελείται ο κώδικας έναρξης. Αυτό παράγει το αρχικό δέντρο ελέγχου που τώρα χαρακτηριστικά χειρίζεται τις μεθόδους της σελίδας στα ακόλουθα βήματα. Ο παραγόμενος κώδικας HTML

αποστέλλεται στον πελάτη. Μετά το πέρας της επεξεργασίας του αιτήματος το δέντρο ελέγχου διαγράφεται.

- **Template Engine - Μηχανή Προτύπων :** Αρχικά οι προγραμματιστές ήταν υποχρεωμένοι να κληρονομούν οι σελίδες τους το System.Web.UI.Page και στη συνέχεια σχεδίαζαν κάθε σελίδα ξεχωριστά. Από την έκδοση 2.0 και μετά έγινε η εισαγωγή των Master Pages, από τις οποίες κληρονομούν όσες σελίδες θέλει ο προγραμματιστής κάποια κοινά χαρακτηριστικά, τόσο οπτικά όσο και λειτουργικά.
- **State Management :** Οι εφαρμογές ASP.NET φιλοξενούνται σε έναν κεντρικό υπολογιστή δικτύου (Web Server) και η προσπέλασή τους γίνεται με τη χρήση του Πρωτοκόλλου HTTP. Το ASP.NET προσφέρει πολλές λειτουργίες State Management.
    - **Application State – Κατάσταση Εφαρμογής :** Το Application State αποτελεί μια συλλογή από μεταβλητές ορισμένες από το χρήστη που είναι διαθέσιμες στην εφαρμογή. Αρχικοποιούνται όταν πυροδοτείται το γεγονός Applcation_OnStart, όταν φορτώνεται για πρώτη φορά η σελίδα και είναι διαθέσιμες συνεχώς ώσπου να σταματήσει και η τελευταία εκτέλεση. Μπορούν να προσπελαστούν από τη συλλογή Application. Ταυτοποιούνται από το όνομά τους.
    - **Session State – Κατάσταση Χρήστη :** Το Session State αποτελεί μια συλλογή από μεταβλητές ορισμένες από το χρήστη που είναι διαθέσιμες όσο λειτουργεί το Session του επισκέπτη της εφαρμογής. Μπορούν να προσπελαστούν από τη συλλογή Session.
    - **View State – Κατάσταση Σελίδας :** Το View State αποτελεί ένα σύνολο μεταβλητών που αφορούν την ίδια τη σελίδα. Μπορούν να προσπελαστούν από τη συλλογή ViewState.
- Επίσης υπάρχουν οι κλασσικές λειτουργίες των **cookies, caching και querystring.**

Η συνηθισμένη διάρθρωση των καταλόγων μιας εφαρμογής ASP.NET είναι η παρακάτω:

App_Browsers : Περιέχει αρχεία σχετικά με τον περιηγητή - Browser

App_Code : Περιέχει αρχεία για το Business Model της εφαρμογής.

App_Date : Περιέχει αρχεία βάσεων δεδομένων

App_LocalResources : Περιέχει μεταφράσεις σε διάφορες γλώσσες

App_GlobalResources : Περιέχει τα αρχεία resx που είναι διαθέσιμα για όλες τις σελίδες τις εφαρμογής.

App_Themes : Περιέχει διαφορετικά οπτικά θέματα της εφαρμογής.

App_WebReferences: Περιέχει αναφορές σε διαδικτυακές υπηρεσίες – web services .

Bin : Περιέχει μεταγλωττισμένα αρχεία.

1.2 Η γλώσσα προγραμματισμού C#

1.2.1 Γενικά Στοιχεία - Ιστορικά

Η C# (εκφωνείται ΣΙ ΣΑΡΠ) είναι μία γλώσσα προγραμματισμού η οποία συνδυάζει τον συναρτησιακό προγραμματισμό, τον προστακτικό προγραμματισμό και τον αντικειμενοστρεφή προγραμματισμό. Αναπτύχθηκε από τη Microsoft ως τμήμα του .NET Framework και στη συνέχεια κανονικοποιήθηκε από ECMA και ISO. Είναι μια από τις γλώσσες που υποστηρίζονται από το CRL.

Η C# ως γλώσσα έχει ως στόχο να είναι απλή, μοντέρνα, γενικού σκοπού και αντικειμενοστρεφής. Η ομάδα ανάπτυξής της έχει ως επικεφαλή τον Anders Hejlsberg, τον δημιουργό της Borland Object Pascal. Η σύνταξη της γλώσσας είναι αντικειμενοστραφής βασισμένη στη C++ και είναι πολύ επηρεασμένη από τη Java. Αρχικά το όνομά της ήταν Cool, που σημαίνει "C- Like Object Oriented Language" δηλαδή "Αντικειμενοστρεφής Γλώσσα που μοιάζει με την C". Ωστόσο τον Ιούλιο του 2000 όταν δημοσιοποίησε τη γλώσσα η Microsoft, πηρέ τη σημερινή της ονομασία. Η πιο πρόσφατη έκδοσή της είναι η 3.0 η οποία κυκλοφόρησε το 2007 μαζί το .NET Framework 3.5.Η έκδοση 4.0 είναι ήδη υπό υλοποίηση.

1.2.2 Χαρακτηριστικά

Παρακάτω γίνεται απλή παρουσίαση ορισμένων βασικών χαρακτηριστικών της γλώσσας C# :

- Μια component-oriented γλώσσα
- Βασίζεται στην εμπειρία του COM+
- Εγγενής υποστήριξη για
    - Namespaces
    - Versioning
    - Attribute-driven ανάπτυξη
- Ελάχιστη καμπύλη εκμάθησης για όλους
- Πιο σαφής από την C++
- Πιο δομημένη από τη Visual Basic

Αναλυτικότερα :

Χώροι Ονομάτων (Namespaces)

- Κάθε ορισμός πρέπει να περιέχεται σε ένα Namespace
- Αποφεύγεται η σύγκρουση ονομάτων
- Επιβάλλεται η τάξη
- Τα API γίνονται ευκολότερα στην κατανόηση
- Μπορούν και πρέπει να εμφωλιάζονται
- Ομάδες κλάσεων και τύπων κατά σημασιολογία
- Δηλώνονται με τη λέξη-κλειδί **namespace**
- Shortcut με **using**

Κλάσεις

- Υλοποίηση κώδικα και δεδομένων
  - Αναπαριστά μια σημασιολογική ενότητα
- Υλοποιεί interfaces
- Κληρονομεί από μία βασική κλάση (base class)
- Οι κλάσεις περιέχουν:
  - Πεδία (Fields): μεταβλητές μελών
  - Ιδιότητες (Properties): τιμές στις οποίες γίνεται πρόσβαση με ζευγάρια μεθόδων **get**/**set**
  - Μέθοδοι (Methods): λειτουργικότητα για το αντικείμενο ή την κλάση
  - Άλλα: events, indexers, delegates

Μορφή – Δομή

- Δεν υπάρχουν header files
- Η C# χρησιμοποιεί το μοντέλο «ορισμός μαζί με τη δήλωση”
  - Όπως περίπου Visual Basic, Pascal, Modula, Java
  - Παρόμοιο με την υλοποίηση του "inline” στη C++
- Όλος ο κώδικας και οι δηλώσεις σε ένα μέρος
  - Ο κώδικας διατηρείται συνεπής και συντηρήσιμος
  - Πιο εύκολη κατανόηση για συνεργασία με ομάδες
  - Δηλώσεις προσβάσιμες μέσω metadata
- Μεταγλώττιση υπό συνθήκη – αλλά **όχι** macros

Σύστημα Τύπων

- Βασίζεται στο κοινό σύστημα τύπων του .NET Framework
- Εγγενής πρόσβασή στο σύστημα τύπων του .NET
  - C# γεννήθηκε από το .NET
- Βασικές έννοιες:
  - Τα πάντα είναι objects
  - Τα πάντα κληρονομούν το **System.Object**
  - Ξεκάθαρη διάκριση ανάμεσα στους τύπους τιμής και αναφοράς
    - Με τιμή: απλοί τύποι, enums, structs
    - Με αναφορά: Interfaces, Κλάσεις, Πίνακες

Απλοί Τύποι

- Τύποι Ακεραίων
  - **byte**, **sbyte** (8bit), **short**, **ushort** (16bit)
  - **int**, **uint** (32bit), **long**, **ulong** (64bit)
- Τύποι κινητής υποδιαστολής (IEEE)
  - **float** (ακρίβεια 7 ψηφίων)
  - **double** (ακρίβεια 15–16 dψηφίων)
- Exact Numeric Type
  - **decimal** (28 σημαντικά ψηφία)
- Τύποι Χαρακτήρων
  - **char** (ένας χαρακτήρας)
  - **string** (πλούσια λειτουργικότητα, τύπος αναφοράς)
- Τύποι Boolean
  - **bool** (ξεχωριστός τύπος, διαφορετικός από τον **int**)

Απαριθμητοί Τύποι

- Χρησιμοποιούνται στοιχεία με όνομα αντί για αριθμητικές επιλογές
- Ισχυρός τύπος, όχι αυτόματη μετατροπή σε int
- Καλύτερο το "Color.Blue" από μια αριθμητική έκφραση
  - Πιο κατανοητό, ευκολότερη συντήρηση
  - Εξίσου ελαφρύ με το απλό int

Πίνακες

- Zero based, type bound
- Βασίζονται στην .NET κλάση **System.Array**
- Δηλώνονται με τύπο και μορφή, αλλά χωρίς όρια
- Δημιουργούνται με χρήση του **new** με όρια ή initializers

Interfaces

- Συμβόλαια δήλωσης σημασιολογίας μεταξύ δύο μερών
  - Επιτρέπουν το component orientation
- Δηλώνουν τη δομή και τη σημασιολογία για καθορισμένους σκοπούς
- Αφηρημένοι ορισμοί μεθόδων και ιδιοτήτων
- Υποστηρίζουν (πολλαπλή) κληρονομικότητα

Δομές

- Ομάδες δεδομένων και κώδικα
  - Σαν τις κλάσεις classes, αλλά:
    - Δεν υποστηρίζεται κληρονομικότητα
    - Περνιούνται πάντα κατά τιμή
  - Κλάσεις vs. Structs
    - Struct ⇨ "ελαφρύ" data container, τύπος τιμής
    - Class ⇨ Πλούσιο αντικείμενο με αναφορές, τύπος αναφοράς

Προστασία Πρόσβασης

- Υιοθετεί το μοντέλο της C++
  - **public** ⇨ Όλοι μπορούν να καλέσουν ή να έχουν πρόσβαση
  - **protected** ⇨ Μόνο τα μέλη έχουν πρόσβαση
  - **private** ⇨ Μόνο τα μέλη και μόνο αυτής της κλάσης
- Επεκτείνει το μοντέλο της C++
  - **sealed** ⇨ Δεν μπορεί να χρησιμοποιηθεί σαν βασική κλάση
  - **internal** ⇨ Public πρόσβαση μόνο μέσα στην assembly
  - **protected internal** ⇨ Protected στην assembly

Ιδιότητες

- Κάτι ανάμεσα στα πεδία και στις μεθόδους
- Χρησιμοποιούμε properties για:
  - Υλοποίηση read-only μελών
  - Επικύρωση κατά την ανάθεση
  - Παράγωγες ή σύνθετες τιμές
  - Έκθεση τιμών στα interfaces

Indexers

- Συνεπής τρόπος να χτίζεις containers
- Βασίζονται στην ιδέα των properties
- Επιτρέπουν δεικτοδοτούμενη (indexed) πρόσβαση στα αντικείμενα που περιέχουν
- Το προσδιοριστικό του δείκτη (index) μπορεί να είναι οποιουδήποτε τύπου

Αντιπρόσωποι – Delegates

- Κάτι σαν τους δείκτες σε συναρτήσεις στη C/C++
- Ισχυροί τύποι, όχι σύγχυση στο type-casting ή σφάλματα.
- Η δήλωση δημιουργεί μια typed method signature:
  - **delegate void Clicked(Element e, Point p);**
- Ο πραγματικός αντιπρόσωπος είναι ένα στιγμιότυπο αυτού του τύπου

  Το όρισμα περνιέται στον constructor του αντιπροσώπου:
  - Αναφορά στο στιγμιότυπο του αντικειμένου και στη μέθοδο.
  - Η μέθοδος πρέπει να έχει την ίδια ακριβώς υπογραφή

Χαρακτηριστικά – Attributes

- Παρόμοια με τα attributes που γνωρίζουμε από την IDL (Interface Description Language)
- Δηλωτική πρόσβαση στη λειτουργικότητα
- Επεκτάσιμη μέσω custom attributes
- Επιτρέπει την επαύξηση κώδικα με:
  - Υποδείξεις προς το runtime environment
  - Δηλωτική σημασιολογία

1.3 Η γλώσσα προγραμματισμού JavaScript

1.3.1 Γενικά Στοιχεία – Ιστορικά

Η γλώσσα προγραμματισμού JavaScript αναπτύχθηκε από την Netscape ως LiveScript. Όταν έγινε κοινό project της Netscape και της Sun το 1995 μετονομάσθηκε σε JavaScript. Τυποποιήθηκε από την European Computer Manufacturers Association ως ECMA-262 (και ως ISO 16262).

Η γλώσσα JavaScript είναι μια γλώσσα σεναρίων (scripting language) που βοηθά τον προγραμματιστή στην κατασκευή διαδραστικών και λειτουργικών εγγράφων στο Internet. Τα σενάρια γραμμένα σε JavaScript βασικά εκτελούνται στον Browser του χρήστη (client-side processing). Για να μπορέσει ένας Browser να εκτελέσει ένα σενάριο γραμμένο σε JavaScript θα πρέπει να διαθέτει έναν διερμηνευτή JavaScript (interpreter). Το σενάριο της JavaScript ενσωματώνεται στον κώδικα HTML. Οι γνωστότεροι Browsers, IE, Firefox, Chrome, Opera ενσωματώνουν ερμηνευτή της JavaScript.

1.3.2 Χαρακτηριστικά

Μερικές από τις δυνατότητες της γλώσσας είναι :

- Έλεγχος στο περιεχόμενο και στην εμφάνιση των ιστοσελίδων
- Έλεγχος της συμπεριφοράς και των λειτουργιών του Browser
- Αλληλεπίδραση με φόρμες HTML
- Αλληλεπίδραση με το χρήστη με τη βοήθεια γεγονότων
- Ανάγνωση ή καταγραφή του Η/Υ του χρήστη μέσω cookies
- Δυνατότητα για εναλλαγή εικόνων

Ωστόσο υπάρχουν και περιορισμοί :

- Για λόγους ασφαλείας δεν επιτρέπει προσπέλαση ή χειρισμό αρχείων στον ΗΥ του χρήστη
- Δεν υποστηρίζει δυνατότητες δικτύωσης
- Ένα σενάριο που εκτελείται στο Browser του χρήστη δεν μπορεί να ενημερώσει αρχεία που βρίσκονται στο Web Server

Μερικές βασικές έννοιες της JavaScript είναι:

- Η JavaScript περιέχει αντικειμενοστρεφή στοιχεία

- Οι βασικές έννοιες του αντικειμενοστρεφούς προγραμματισμού είναι: κλάση, αντικείμενο, μέθοδοι, ιδιότητες
- Ως αντικείμενο θεωρούμε μια δομή δεδομένων που διαθέτει ιδιότητες και μεθόδους
- Τα αντικείμενα που χρησιμοποιεί η JavaScript είτε είναι στοιχεία του εγγράφου πχ το έγγραφο που φορτώνεται ή ένα frame, είτε αποτελούν στοιχεία της ίδιας της γλώσσας, πχ, ο πίνακας, η ημερομηνία κτλ
- Οι ιδιότητες (properties) περιγράφουν την κατάσταση ενός αντικειμένου πχ η ιδιότητα closed του αντικειμένου window προσδιορίζει αν ένα παράθυρο είναι ανοικτό ή κλειστό
- Οι μέθοδοι καθορίζουν πως συμπεριφέρεται ή ενεργεί ένα αντικείμενο. πχ η μέθοδος alert() όταν κληθεί για το αντικείμενο window εμφανίζει ένα προειδοποιητικό μήνυμα. Σύνταξη μεθόδου: αντικείμενο.μέθοδος() Πχ. Window.alert("Καλημέρα");
- Η έννοια της κλάσης είναι η κεντρική ιδέα του αντικειμενοστρεφούς προγραμματισμού και λειτουργεί ως φόρμα κατασκευής αντικειμένων που κληρονομούν τις μεθόδους και ιδιότητες της κλάσης
- Η JavaScript δεν διαθέτει τη δεσμευμένη λέξη class όπως η Java & C++ που επιτρέπουν στον προγραμματιστή να δημιουργήσει δικές του κλάσεις αλλά μπορεί να χρησιμοποιεί αντικείμενα και δημιουργεί νέα με τη βοήθεια συναρτήσεων

Δεδομένα Μεταβλητές

- Η JavaScript υποστηρίζει αριθμητικά και αλφαριθμητικά δεδομένα
- Αριθμητικά: αριθμοί ακέραιοι, κινητής υποδιαστολής
- Αλφαριθμητικά: ομάδες χαρακτήρων, τοποθετούνται μέσα σε απλά ή διπλά εισαγωγικά
- Λογικές τιμές: true, false
- Μεταβλητές. Ορίζονται με τη χρήση της δεσμευμένης λέξης var

Εμβέλεια Μεταβλητών

- Καθολική (global) Μια μεταβλητή που δηλώνεται εκτός συναρτήσεων και αναγνωρίζεται σε όλα τα σημεία του εγγράφου. Μια καθολική μεταβλητή μπορεί να δηλωθεί με τη χρήση της var ή όχι
- Τοπική (local) μια μεταβλητή που δηλώνεται μέσα σε συνάρτηση και αναγνωρίζεται μόνο μέσα στο σώμα της συνάρτησης. Μια μεταβλητή που δηλώνεται μέσα σε μια συνάρτηση χωρίς τη var τότε θεωρείται καθολική!

- Αν μια καθολική και μια τοπική μεταβλητή έχουν το ίδιο όνομα μέσα στη συνάρτηση ισχύει η τοπική

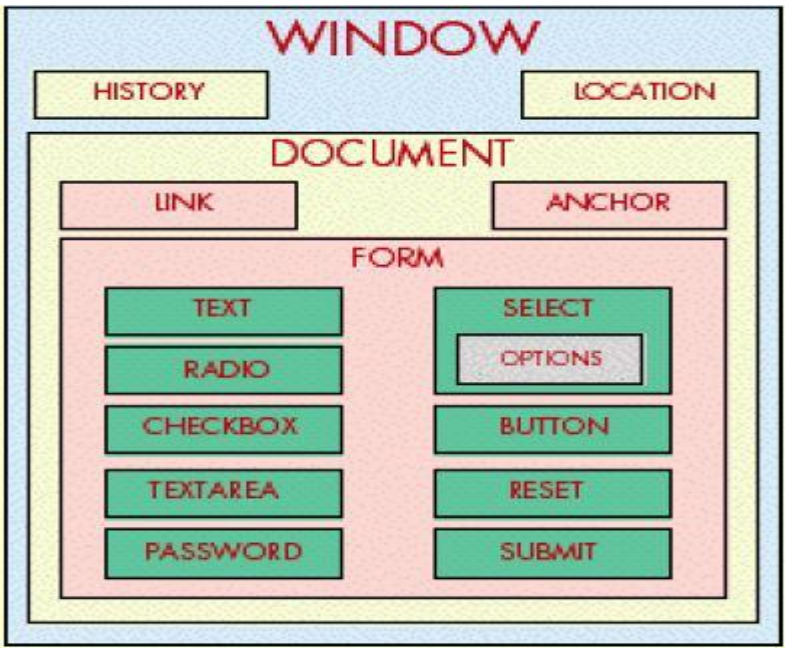
JavaScript και Browser

Τρία βασικά θέματα για τον τρόπο δράσης του κώδικα JavaScript σε ένα Browser είναι τα παρακάτω:

- **Το αντικείμενο window** λειτουργεί ως υποδοχέας άλλων αντικειμένων και ως χώρος εκτέλεσης του κώδικα JavaSccript στον Η/Υ του χρήστη. Ο κύριος σκοπός ενός Browser είναι να εμφανίσει τα HTML έγγραφα σ' ένα παράθυρο. Το αντικείμενο window περικλείει το αντικείμενο document που αντιστοιχεί στο έγγραφο HTML
- **Το ιεραρχικό μοντέλο αντικειμένων DOM (Document Object Model)** που δημιουργούνται σε ένα Browser, όπου η βάση είναι το αντικείμενο window. Ολοι οι Browser δεν χρησιμοποιούν το ίδιο DOM

Ιεραρχικό μοντέλο δημιουργίας των αντικειμένων

Το DOM είναι ένας εσωτερικός «χάρτης» όλων των αντικειμένων ενός HTML εγγράφου. Στην κορυφή της ιεραρχίας βρίσκεται το παράθυρο του Browser (window) και ακολουθεί το document. Για να αναφερθούμε στην τιμή (value) ενός πεδίου κειμένου (Text) με όνομα visitor, που βρίσκεται στη φόρμα myform θα πρέπει να δώσουμε την πλήρη «διαδρομή».



Εικόνα 1.3 : Ιεραρχικό Μοντέλο DOM

Φόρμες και JavaScript

- Το αντικείμενο form της JavaScript αντιστοιχεί σε μια φόρμα HTML που τοποθετείται στο έγγραφο με το tag <form>
- Το αντικείμενο document έχει την ιδιότητα forms[], ένα πίνακα με τις φόρμες που περιέχει το έγγραφο
- Η ιδιότητα length του πίνακα forms[]επιστρέφει το πλήθος των φορμών που υπάρχουν στο έγγραφο
- Με την ιδιότητα elements[] (πίνακας) της forms[] έχουμε πρόσβαση στα διάφορα συστατικά της φόρμας (κουμπιά, πλαίσια κειμένου κτλ). πχ με την document.forms[2].elements[3] μπορούμε να προσπελάσουμε το 4ο συστατικό της 3ης φόρμας.

JSON (JavaScript Object Notation)

Το JSON – JavaScript Object Notation είναι μία μορφή δεδομένων βασισμένη σε κείμενο που διευκολύνει την προσπέλαση από τη JavaScript των δεδομένων που αποτελούν απάντηση σε μία Ajax κλήση μιας μεθόδου που βρίσκεται στον εξυπηρετητή. Τα δεδομένα προσπελαύνονται ως αντικείμενα JavaScript.

1.4 Η τεχνολογία ASP.NET AJAX

Η τεχνολογία ASP.NET AJAX, αρχικά η ονομασία της ήταν Atlas, είναι ένα πλαίσιο ανάπτυξης της Microsoft, το οποίο μεταφέρει την τεχνολογία AJAX στην πλατφόρμα ASP.NET.

Με το ASP.NET AJAX είναι δυνατή η γρήγορη δημιουργία ιστοσελίδων, οι οποίες είναι πιο φιλικές προς το χρήστη. Αυτό επιτυγχάνεται κυρίως με τη συνεργασία JavaScript, DHTML και Server-Side κώδικα.

Ορισμένα χαρακτηριστικά του ASP.NET AJAX είναι:

- Σημαντική βελτίωση της αποτελεσματικότητας, διότι σημαντικά τμήματα της ιστοσελίδας εκτελούνται από τον Web Browser σε επίπεδο client
- Εν – μέρει ανανεώσεις της ιστοσελίδας – μόνο των τμημάτων της που έχουν υποστεί κάποια αλλαγή

- Συνεργασία του client με τα ASP.NET services για τον έλεγχο χρήστη και θεώρηση φορμών
- Εύκολη κλήση web services μέσω JavaScript σε επίπεδο client και εύκολη προσπέλαση των δεδομένων μέσω JSON serializer
- Ύπαρξη πλαισίου ανάπτυξης για την προσαρμογή των διαφόρων στοιχείων έλεγχου ώστε να ανταποκρίνονται στις ανάγκες του χρήστη
- Υποστήριξη για όλους του δημοφιλείς Web Browsers όπως Internet Explorer, Mozilla Firefox, Google Chrome , Opera.

Η αρχιτεκτονική του ASP.NET AJAX φαίνεται στην παρακάτω εικόνα:

Εικόνα 1.4 : ASP.NET AJAX

1.5 Microsoft SQL Server

Ο Microsoft SQL Server είναι ένα σύστημα διαχείρισης σχεσιακής βάσης δεδομένων το οποίο κατασκευάστηκε από την Microsoft. Οι βασικές γλώσσες ερωτήσεών του είναι οι MS-SQL και T-SQL.

1.5.1 Αρχιτεκτονική MsSql Server

Το επίπεδο πρωτοκόλλου εφαρμόζει το εξωτερικό interface στον SQL Server. Όλες οι λειτουργίες οι οποίες μπορούν να επικαλεστούν (αναζητηθούν/επιζητηθούν) στον SQL Server συνδέονται με αυτόν μέσω μιας διάταξης καθορισμένης απο την Microsoft που αποκαλείται Tabular Data Stream (TDS). Η TDS είναι μια εφαρμογή επιπέδου πρωτοκόλλου , η οποία χρησιμοποιείται για να μεταφερθούν πληροφορίες ανάμεσα σε ένα Server βάσης (database Server) δεδομένων και ένα client. Αρχικά αναπτύχθηκε από την εταιρεία Sybase για να εξυπηρετήσει τους σκοπούς της βάσης δεδομένων Sybase SQL Server το 1984, και αργότερα απο την Microsoft για τον SQL Server, TDS πακέτα μπορούν να συμπεριληφθούν σε άλλα physical transport dependent πρωτόκολλα, συμπεριλαμβανομένων TCP/IP, Named Pipes και Shared Memory. Συνεπώς, πρόσβαση στον SQL Server είναι δυνατή μέσω αυτών των πρωτόκολλων.

1.5.2 Αποθήκευση δεδομένων

Ο SQL Server υποστηρίζει διαφορετικούς τύπους δεδομένων, συμπεριλαμβανομένων, μεταξύ άλλων, πρωτευόντων τύπων όπως ακέραιους αριθμούς, κινητούς δεκαδικούς, δεκαδικούς, char (συμπεριλαμβανομένων σειρών χαρακτήρων / character strings), μεταβλητές σειρές χαρακτήρων / variable length character strings), δυαδικούς (για μη κατασκευασμένες ποσότητες δεδομένων), κείμενο (για δεδομένα που αναφέρονται σε κείμενο). Επίσης επιτρέπει σε σύνθετους τύπους οι οποίοι έχουν οριστεί από τον χρήστη (user-defined composite types - UDTs) να οριστούν και να χρησιμοποιηθούν.

1.5.3 Υπηρεσίες

Ο SQL Server επίσης εμπεριέχει μια ποικιλία υπηρεσιών add-on. Παρότι όλα αυτά δεν είναι ουσιώδη για την λειτουργία του συστήματος βάσης δεδομένων, παρέχονται ως επιπρόσθετες υπηρεσίες σε αυτές της διαχείρισης βάσεων δεδομένων. Αυτές οι υπηρεσίες εκτελούνται είτε ως συνιστούντα μέρη κάποιου SQL Server είτε ως αυτοτελή μέσω κάποιας υπηρεσίας των windows και παρουσιάζουν την δική τους

API (application programming interface) για να ελέγχουν και να αλληλεπιδρούν με αυτά.

- **Service Broker (μεσάζων):** Ο μεσάζων, ο οποίος εκτελείτε ως μέρος της μηχανής βάσης δεδομένων, παρέχει μια αξιόπιστη πλατφόρμα αποστολής μηνυμάτων και μηνυμάτων προς εκτέλεση για εφαρμογές του SQL Server. Οταν χρησιμοποιείτε μέσα σε ένα παράδειγμα παρέχει ένα ασύγχρονο προγραμματιστικό περιβάλλον. Για εφαρμογές που χρησιμοποιούν πολλαπλά παραδείγματα, ο μεσάζων επικοινωνεί μέσω TCP/IP και επιτρέπει τα διάφορα μέρη να συγχρονιστούν μεταξύ τους, μέσω της ανταλλαγής μηνυμάτων.
- **Υπηρεσία δημιουργίας αντιγράφων:** Οι SQL Server Replication Services χρησιμοποιούνται απο τον SQL Server για να αντιγράψει και να συγχρονίσει τα αντικείμενα της βάσης δεδομένων, είτε στην ολότητά τους ή ένα υποσύνολο των παρόντων αντικειμένων, διαμέσου συντελεστών αντιγραφής (replication agents), οι οποίοι μπορεί να είναι άλλοι Server βάσεων δεδομένων σε όλη την έκταση του δικτύου, ή caches βάσεων δεδομένων απο την πλευρά του client. Η αντιγραφή ακολουθεί ένα μοντέλο εκδότη/συνδρομητή (publisher/subscriber), π.χ. οι αλλαγές αποστέλλονται από έναν Server βάσης (publisher) δεδομένων και λαμβάνονται από άλλους (subscriber).
- **Υπηρεσίες ανάλυσης :** Οι υπηρεσίες ανάλυσης του SQL Server προσφέρουν τις δυνατότητες των OLAP και συλλογής δεδομένων (data mining) για τις βάσεις δεδομένων SQL. Η μηχανή OLAP υποστηρίζει τις εξής μεθόδους αποθήκευσης δεδομένων: MOLAP, ROLAP και HOLAP. Οι υπηρεσίες ανάλυσης υποστηρίζουν την XML για πρότυπη ανάλυση ως το θεμελιώδες πρωτόκολλο επικοινωνίας. Τα κυβικά δεδομένα (cube data) μπορούν να προσπελαστούν με την χρήση MDX ερωτημάτων. Η καθοριστική λειτουργικότητα της συλλογής δεδομένων εκφράζεται μέσω της γλώσσας ερωτημάτων DMX. Οι υπηρεσίες ανάλυσης περιλαμβάνουν ποικίλους αλγόριθμους – δέντρα αποφάσεων, clustering algorithm, Naive Bayes αλγόριθμους, ανάλυση χρονολογικών σειρών, αλληλουχίες clustering αλγορίθμων, ανάλυση γραμμικών και logistic παλινδρομήσεων και νευρωνικά δίκτυα – για χρήση στην συλλογή δεδομένων.
- **Υπηρεσίες αναφοράς :** Οι υπηρεσίες αναφοράς του SQL Server είναι ένα περιβάλλον δημιουργίας αναφορών για δεδομένα που συγκεντρώνονται από τις βάσεις δεδομένων των SQL Server. Διαχειρίζεται μέσω ενός web interface. Οι

υπηρεσίες αναφοράς εμπεριέχουν ένα web service interface για να υποστηρίξουν την ανάπτυξη των συνηθισμένων εφαρμογών αναφοράς. Οι αναφορές δημιουργούνται ως RDL αρχεία. Οι αναφορές μπορούν να κατασκευαστούν με την χρήση της Microsoft Visual Studio (συμπεριλαμβανομένης και της Visual Studio.NET 2003) καθώς και με την βοήθεια της Business Intelligence Development Studio, είτε εγκατεστημένες είτε με τον εμπεριεχόμενο Report Builder. Αφότου κατασκευαστούν τα RDL αρχεία μπορούν να αποδοθούν σε ποικίλες μορφές όπως Excel, PDF, CSV, XML, TIFF (και άλλα είδη αρχείων εικόνας) και HTML web αρχείο.

- **Υπηρεσίες ειδοποίησης :** Εμφανίστηκε για πρώτη φορά με τον SQL Server 2005 (και είναι ο μοναδικός που έχει αυτή την υπηρεσία μέχρι σήμερα). Η υπηρεσία αυτή του SQL Server αποτελεί μια πλατφόρμα δημιουργίας ειδοποιήσεων, οι οποίες αποστέλλονται στους συνδρομητές της υπηρεσίας αυτής. Ένας συνδρομητής εγγράφεται σε ένα συγκεκριμένο γεγονός ή συναλλαγή (η οποία σημειώνεται στον Server της βάσης δεδομένων ως ένα trigger). Όταν το γεγονός συμβεί, η υπηρεσία ειδοποίησης αποστέλλει ένα μήνυμα στον εγγεγραμμένο συνδρομητή, με την χρήση ενός service broker, που τον πληροφορεί για την πραγματοποίηση του γεγονότος.
- **Υπηρεσίες ενσωμάτωσης :** Η υπηρεσία ενσωμάτωσης του SQL Server χρησιμοποιείται για να ενσωματωθούν δεδομένα από διάφορες πηγές.. Χρησιμοποιείται για τις ETL δυνατότητες του SQL Server για τις ανάγκες αποθήκευσης δεδομένων. Η υπηρεσία αυτή εμπεριέχει εργαλεία GUI για να κατασκευάζει κανείς ποικίλα πρακτικά workflows ενσωμάτωσης τα οποία εξάγουν δεδομένα από διάφορες πηγές, υποβάλλουν δεδομένα, μετατρέπουν δεδομένα συμπεριλαμβανομένης και της συγκέντρωσής τους, αντιγραφή και συγχώνευση δεδομένων και μετά φορτώνει τα αλλαγμένα δεδομένα σε άλλες πηγές ή στέλνει e-mails πληροφορώντας για την κατάσταση της διεργασίας.
- **Υπηρεσία Αναζήτησης Κειμένου :** Η υπηρεσία αναζήτησης κειμένου του SQL Server είναι μια ειδικευμένη υπηρεσία indexing και querying για unstructured κείμενο αποθηκευμένο στις βάσεις δεδομένων του SQL Server. Το ευρετήριο της πλήρους αναζήτησης κειμένου μπορεί να κατασκευαστεί σε οποιαδήποτε στήλη με text-based δεδομένα. Επιτρέπει να αναζητηθούν λέξεις στις στήλες κειμένου (text-columns). Ενώ κάτι τέτοιο μπορεί να πραγματοποιηθεί με έναν SQL `LIKE` operator, η χρήση της υπηρεσίας αναζήτησης κειμένου του SQL Server μπορεί να αποδειχθεί πιο αποδοτική. Η Full Text Search (FTS) - πλήρης αναζήτηση κειμένου – επιτρέπει την μη ακριβή αντιστοίχηση των συμβόλων της πηγής, με την χρήση αριθμού ιεραρχίας, ο οποίος μπορεί να κυμαίνεται από 0 έως 1000 – μεγαλύτερος αριθμός σημαίνει πιο ακριβής αντιστοίχιση. Επίσης επιτρέπει την γλωσσική αντιστοίχηση («inflectional search»),

π.χ. γλωσσολογικές παραλλαγές μια λέξης (όπως διαφορετικός χρόνος για ένα ρήμα) θα δοθεί ως αντίστοιχο για μια δοσμένη λέξη (αλλά με μικρότερη ιεραρχία σε σχέση με την ακριβές αντίστοιχο).

1.5.4 SQL Server 2005

Ο SQL Server 2005, κυκλοφόρησε τον Οκτώβριο του 2005, ως διάδοχος του SQL Server 2000. Εμπεριείχε δική του ενσωματωμένη υποστήριξη διαχείρισης δεδομένων XML, επιπρόσθετα των σχεσιακών δεδομένων. Γι' αυτό τον σκοπό, όρισε ένα τύπο δεδομένων XML ο οποίος μπορούσε να χρησιμοποιηθεί είτε ως τύπος δεδομένου σε στήλες βάσεων δεδομένων ή ως literals στα ερωτήματα. Οι στήλες XML μπορούν να συσχετιστούν με XSD παραστάσεις. Τα αποθηκευμένα XML δεδομένα επαληθεύονται μεσω της παράστασης. Ο XML μετατρέπεται σε ένα εσωτερικό δυαδικό τύπο δεδομένων προτού αποθηκευτεί στην βάση δεδομένων. Εξειδικευμένες μέθοδοι για indexing έγιναν διαθέσιμες για τα XML δεδομένα. Τα XML δεδομένα καλούνται χρησιμοποιώντας την XQuery – ο SQL Server 2005 πρόσθεσε κάποιες επεκτάσεις στην γλώσσα T-SQL προκειμένου να επιτρέψει την ενσωμάτωση των XQuery ερωτημάτων στην T-SQL. Επιπλέον, καθορίζει μια νέα επέκταση στην XQuery, την αποκαλούμενη XML DML, η οποία επιτρέπει μετατροπές σε query-based XML δεδομένα. Ο SQL Server 2005 επίσης επιτρέπει σε κάποιο Server βάσης δεδομένων να δημοσιευτεί (exposed) μέσω web υπηρεσιών χρησιμοποιώντας πακέτα TDS «κρυμμένα» μέσα σε SOAP ερωτήματα. Όταν τα δεδομένα προσπελαύνονται από υπηρεσίες web, τα αποτελέσματα μετατρέπονται σε XML.

Για σχεσιακά δεδομένα, η T-SQL έχει εμπλουτιστεί με διάφορα χαρακτηριστικά (features) για αντιμετώπιση λαθών και υποστήριξη για recursive ερωτήματα. Ο SQL Server 2005 έχει επίσης εμπλουτιστεί με νέους αλγορίθμους indexing και καλύτερα συστήματα ανάκτησης λαθών. Οι σελίδες δεδομένων για καλύτερη προσαρμοστικότητα στα λάθη επαληθεύεται, και μια αισιόδοξη σύγχρονη υποστήριξη (optimistic concurrency support) έχει προστεθεί για καλύτερη απόδοση. Η ρύθμιση των αδειών και της πρόσβασης έχουν γίνει πιο λεπτομερειακά και ο επεξεργαστής των ερωτημάτων χειρίζεται την ταυτόχρονη εκτέλεση ερωτημάτων με ένα πιο αποδοτικό τρόπο. Οι διαμερίσεις/μερισμοί/διχοτομήσεις (partitions) στους πίνακες και τους δείκτες υποστηρίζονται εσωτερικά (natively), έτσι η κλιμάκωση (scaling) μιας βάσης δεδομένων σε ένα cluster είναι ευκολότερη. Ο SQL CLR εμφανίστηκε με τον SQL Server 2005 και επέτρεψε την ενοποίηση με την .NET Framework.

1.5.5 SQL Server Management Studio

Το περιβάλλον διαχείρισης SQL Server είναι ένα εργαλείο GUI που εμπεριέχεται στον SQL Server και χρησιμοποιείται για διαμόρφωση, διαχείριση και χρήση όλων των συστατικών στοιχείων του Microsoft SQL Server. Το εργαλείο αυτό περιλαμβάνει τόσο script editors και εργαλεία γραφικών τα οποία δουλεύουν αρμονικά με τα αντικείμενα και τα χαρακτηριστικά του Server. Το περιβάλλον διαχείρισης SQL Server ,από την εμφάνιση του SQL Server2005, αντικαθιστά τον Enterprise Manager ως το βασικό εργαλείο (Interface) διαχείρισης για τον Microsoft SQL Server. Επίσης υπάρχει και μια έκδοση του περιβάλλοντος διαχείρισης SQL Server προσαρμοσμένη για τον SQL Server Express Studio, η οποία είναι γνωστή ως SQL Server Management Studio Express (SSMSE).

Ένα βασικό/κεντρικό χαρακτηριστικό του περιβάλλοντος διαχείρισης SQL Server είναι ο Object Explorer, ο οποίος επιτρέπει στον χρήστη να αναζητήσει, επιλέξει και να ενεργήσει σε οποιοδήποτε αντικείμενο που υπάρχει μέσα στον Server. Μπορεί επίσης να χρησιμοποιηθεί ώστε να παρατηρηθούν και αναλυθούν οπτικά σχέδια ερωτημάτων και να βελτιστοποιηθεί, μεταξύ άλλων, η απόδοση της βάσης δεδομένων. Το περιβάλλοντος διαχείρισης SQL Server μπορεί επιπλέον να χρησιμοποιηθεί για να δημιουργήσουμε μια νέα βάση δεδομένων, να τροποποιήσουμε οποιοδήποτε σχεδιάγραμμα υπάρχουσας βάσης δεδομένων προσθέτοντας ή μετατρέποντας πίνακες και δείκτες, ή να αναλύσουμε την απόδοση. Τέλος συμπεριλαμβάνει τα παράθυρα ερωτήσεων τα οποία παρέχουν ένα GUI-based interface για καταχώρηση/γραφή και εκτέλεση ερωτημάτων.

1.6 Microsoft IIS Server

Ο IIS (Internet Information Server) είναι ένα σύνολο από εξυπηρετητές (Servers) διαδικτύου ανεπτυγμένος από την Microsoft και αποτέλεσε την πλέον ανταγωνιστική απάντηση στον πολύ δημοφιλή apache Server. Στο σύνολο των εξυπηρετητών που εμπεριέχονται στον IIS, προς το παρόν, συμπεριλαμβάνονται οι FTP, SMTP, NNTP και HTTP/HTTPS εξυπηρετητές.

Ο IIS προσφέρει επιπρόσθετες δυνατότητες για τα λειτουργικά συστήματα των windows, στα οποία η Microsoft τον παρέχει ενσωματωμένο.

Μαζί με τον IIS η Microsoft παρέχει και ένα σύνολο από προγράμματα και εργαλεία για την κατασκευή και την διαχείριση ιστοσελίδων καθώς και για την επικοινωνία διαδικτυακών εφαρμογών με βάσεις δεδομένων.

Στα μειονεκτήματα του IIS αναφέρεται η ευπάθειά του στις επιθέσεις ιών. Ωστόσο αυτά οι νεότερες εκδόσεις έχουν αποδειχθεί ιδιαίτερα ασφαλείς.

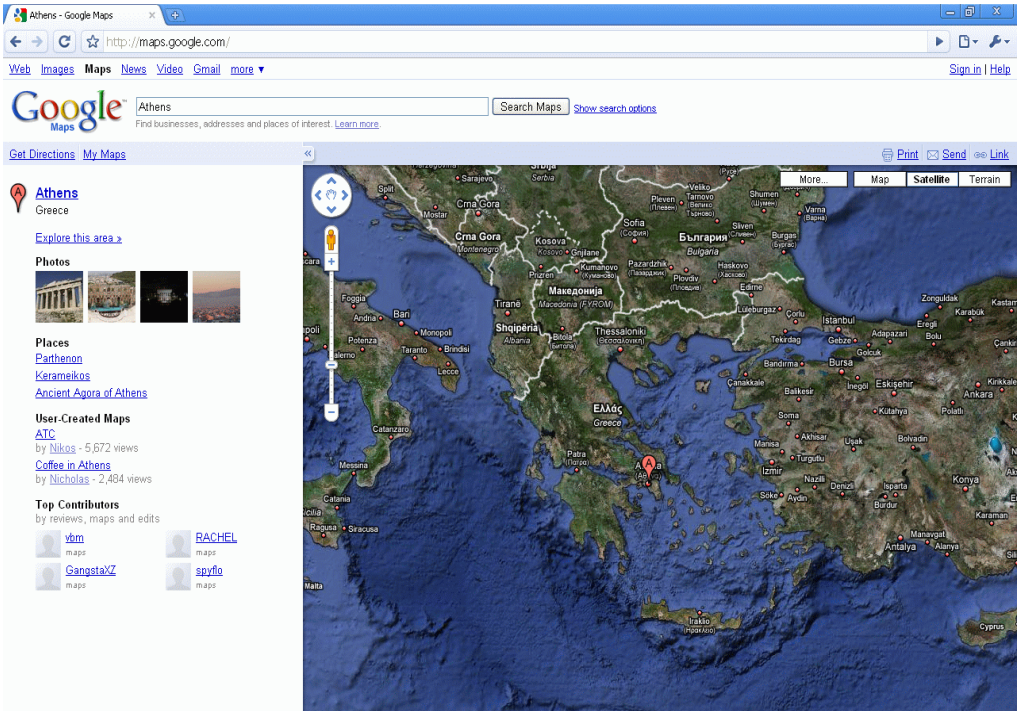
2 Χάρτες Google – Google Maps API

Η εμφάνιση των περιοχών σε χάρτη θα γίνει με τη χρήση των Google Maps και συγκεκριμένα με το Google Maps API. Στο συγκεκριμένο κεφάλαιο γίνεται μία παρουσίαση της συγκεκριμένης τεχνολογίας, δίνοντας έμφαση στο Google Maps API και τα προγραμματιστικά εργαλεία που αυτό προσφέρει.

2.1 Γενικά Στοιχεία

Οι **Χάρτες Google - Google Maps** (η προηγούμενη ονομασία τους ήταν **Google Local)** είναι μια δωρεάν διαδικτυακή υπηρεσία χαρτογράφησης και εφαρμογή τεχνολογίας που παρέχεται από την Google. Η κεντρική σελίδα των χαρτών είναι η παρακάτω:

http://maps.google.com/



Εικόνα 2.1 : Google Maps

Επιπλέον δίνεται δυνατότητα ενσωμάτωσης χαρτών σε δικτυακούς τόπους τρίτων μέσω του Google Maps API. Επίσης υπάρχει και η αντίστοιχη εφαρμογή που ονομάζεται Google Earth, η οποία είναι διαθέσιμη για Microsoft Windows , Mac OS X, Linux , SymbianOS και iPhone OS και προσφέρει περισσότερα οπτικά χαρακτηριστικά.

Η υπηρεσία προσφέρει οδικούς χάρτες, σχεδιασμό διαδρομής για ταξίδια με τα πόδια, ποδήλατο, αυτοκίνητο ή τις δημόσιες συγκοινωνίες καθώς και διάφορους άλλους χάρτες. Επίσης δίνει τη δυνατότητα εύρεσης επιχειρήσεων.

Όπως και πολλές άλλες εφαρμογές της Google, οι Google Maps χρησιμοποιούν εκτεταμένα JavaScript. Όταν ο χρήστης μετακινεί το χάρτη τα κομμάτια του χάρτη κατεβαίνουν από τον εξυπηρετητή και τοποθετούνται στη σελίδα. Όταν ο χρήστης κάνει κάποια έρευνα, τα αποτελέσματα εμφανίζονται αριστερά από το χάρτη και στη συνέχεια η σελίδα ξαναφορτώνεται. Οι διάφορες τοποθεσίες ζωγραφίζονται τοποθετώντας μία κόκκινη κουκίδα στις εικόνες του χάρτη.

Ένα κρυμμένο αντικείμενο JFrame χρησιμοποιείται για τη διατήρηση του ιστορικού. Επιπλέον γίνεται χρήση του JSON για μεταφορά δεδομένων αντί για XML για λόγους απόδοσης. Οι χρησιμοποιούμενες τεχνικές βασίζονται στην τεχνολογία AJAX.

Το γεωγραφικό πληροφοριακό σύστημα δεδομένων που χρησιμοποιείται παρέχεται από την Tele Atlas.

2.2 Google Maps API

2.2.1 Γενικές Πληροφορίες

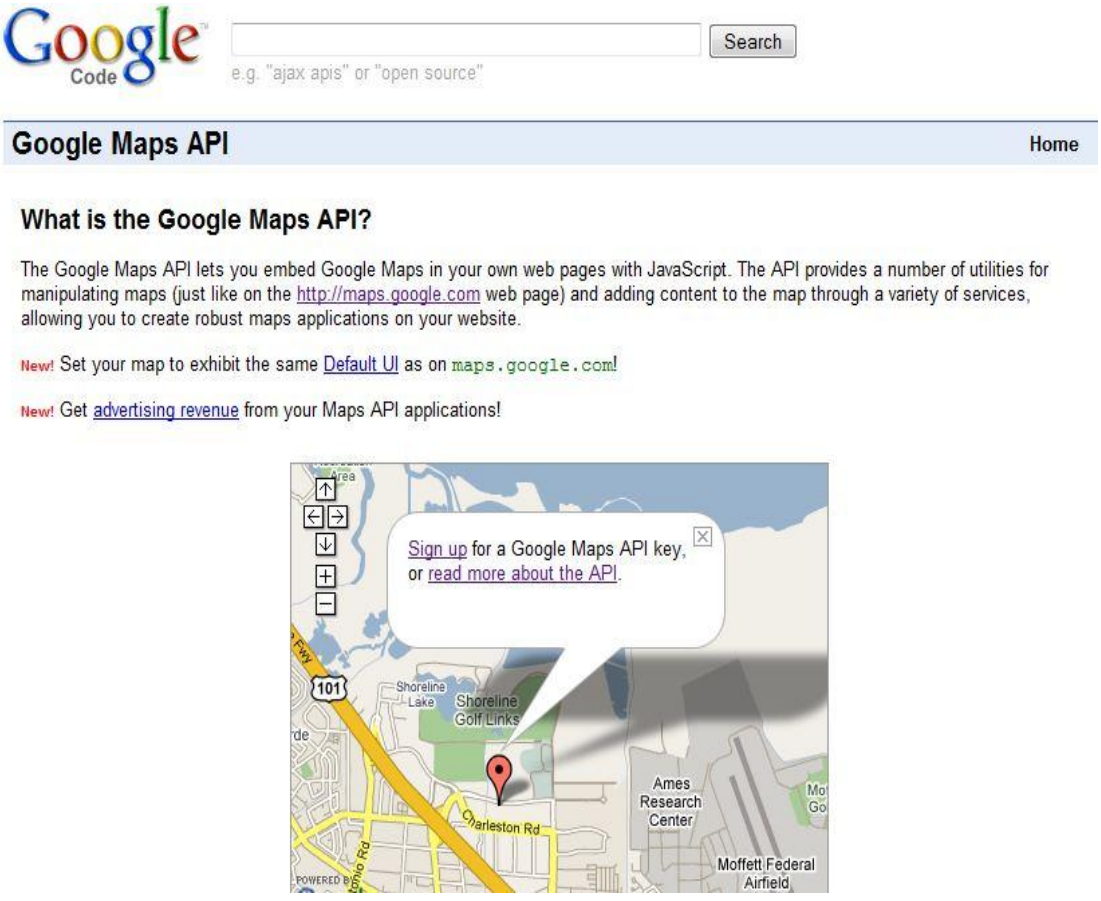
. Η Google δημιούργησε το Google Maps API για να διευκολύνει την ενσωμάτωση των χαρτών Google στους ιστοτόπους από τον καθένα και αποτελεί ελεύθερη υπηρεσία. Με τη χρησιμοποίηση των Google Maps API μπορούμε να ενσωματώσουμε τους πλήρεις χάρτες Google σε έναν εξωτερικό ιστοτόπο. Με την προσθήκη κώδικα JavaScript σε μια ιστοσελίδα μπορούμε να δημιουργήσουμε το δικό μας Google Map προσαρμοσμένο στις ανάγκες μας.

Μπορεί κανείς με τις λειτουργίες που παρέχονται να σχεδιάσει δείκτες πάνω στο χάρτη, ή να αναπτύξει ακόμα πιο πολύπλοκες εφαρμογές. Προς το παρόν όμως οι υπηρεσίες αυτές που παρέχονται από την Google είναι διαθέσιμες μόνο για Web σελίδες και δεν μπορούν να χρησιμοποιηθούν από άλλη εφαρμογή. Δε βασίζονται σε κάποιο ανοιχτό πρότυπο, όπως το SOAP/XML αλλά χρησιμοποιεί JavaScript όπως

προαναφέραμε. Για το λόγο αυτό, ο μόνος τρόπος ενσωμάτωσης τους είναι σε ιστοσελίδες.

Το πλεονέκτημα από τη χρήση των υπηρεσιών αυτών είναι ότι παρέχονται δωρεάν (τουλάχιστον για Web Sites με μέγιστο αριθμό επισκέψεων 50000/ημέρα). Το μόνο που χρειάζεται να κάνει κάποιος για να χρησιμοποιήσει τις υπηρεσίες αυτές είναι να εγγραφεί στο σύστημα της Google για να του παραχωρηθεί ένας κωδικός πρόσβασης στα API της Google (API key). Εφόσον όμως οι υπηρεσίες παρέχονται δωρεάν από τη Google θα πρέπει όποιος τις χρησιμοποιεί να μην τις διαθέτει στο site του επί πληρωμή αλλά επίσης δωρεάν προς τους επισκέπτες του ιστοτόπου του. Όταν το API προώθηηκε αρχικά, δεν είχε τη δυνατότητα να γεοκωδικοποιεί διευθύνσεις, και απαιτούσε από τον χρήστη να προσθέσει με το χέρι τα σημεία (γεωγραφικό πλάτος, γεωγραφικό μήκος) το σχήμα. Ο διαδικτυακός τόπος του Google Maps API είναι ο παρακάτω:

http://code.google.com/apis/maps/



Εικόνα 2.2 : Google Maps API

2.2.2 Προσφερόμενες υπηρεσίες

Στη συνέχεια παρουσιάζουμε συνοπτικά τα χαρακτηριστικά των υπηρεσιών που παρέχονται στο Google Maps API:

- **GMap Class :** Πρόκειται για την βασική κλάση. Ένα αντικείμενο της κλάσης GMap αντιστοιχεί σε ένα χάρτη στην σελίδα μας. Μπορεί κανείς να δημιουργήσει όσα στιγμιότυπα αυτής της κλάσης επιθυμεί (ένα για κάθε χάρτη στη σελίδα). Όταν δημιουργούμε ένα νέο στιγμιότυπο χάρτη, καθορίζουμε ένα στοιχείο στη σελίδα το οποίο θα περιέχει το χάρτη. Ο χάρτης στη συνέχεια, σαν μέγεθος του χρησιμοποιεί το μέγεθος του στοιχείου που τον περιλαμβάνει εκτός και αν το ορίσουμε διαφορετικά. Η κλάση GMap παρέχει μεθόδους χειρισμού του κέντρου του χάρτη και του επιπέδου zoom, καθώς και μεθόδους για προσθήκη ή αφαίρεση διαφόρων overlays (όπως για παράδειγμα στιγμιότυπα των κλάσεων GMarker και GPolyline όπως θα δούμε). Επιπλέον, παρέχει μεθόδους που μας δίνουν τη δυνατότητα να ανοίξουμε ένα «παράθυρο πληροφοριών» το οποίο θα περιέχει διάφορες πληροφορίες πάνω στο χάρτη.
- **Events :** Με τη χρήση των event listeners μπορούμε να εισάγουμε δυναμικά στοιχεία στην εφαρμογή μας. Ένα αντικείμενο της κλάσης αυτής παρέχει έναν αριθμό από γεγονότα (events) και η εφαρμογή μας μπορεί να τα «ακούει» χρησιμοποιώντας τις στατικές μεθόδους GEvent.addListener ή GEvent.bind. Έτσι μπορεί το πρόγραμμα μας για παράδειγμα να εμφανίζει ένα μήνυμα ανάλογα με το κλικ ενός χρήστη πάνω στο χάρτη.
- **Info Window :** Κάθε χάρτης έχει ένα μόνο ‘‘παράθυρο πληροφοριών’’, το οποίο εμφανίζει HTML περιεχόμενο σε ένα παράθυρο πάνω από το χάρτη. Το παράθυρο πληροφοριών μοιάζει με ένα «συννεφάκι». Αποτελείται από μια περιοχή με το περιεχόμενο των πληροφοριών η οποία λεπταίνει στην άκρη και γίνεται σαν δείκτης που δείχνει σε ένα καθορισμένο σημείο πάνω στο χάρτη. Αν κάποιος έχει χρησιμοποιήσει το Google Maps, τότε πολύ πιθανό να έχει δει ένα «παράθυρο πληροφοριών» όταν κάνει κλικ σε ένα εικονίδιο (marker). Άλλο χαρακτηριστικό των παραθύρων αυτών είναι ότι δεν μπορεί κανείς να εμφανίσει περισσότερα από ένα ταυτόχρονα σε ένα δοσμένο χάρτη αλλά μπορεί να μετακινήσει το παράθυρο και να αλλάξει τα περιεχόμενά του αν αυτό είναι επιθυμητό. Η βασική μέθοδος για ένα παράθυρο πληροφοριών είναι η openInfoWindow, η οποία παίρνει σαν είσοδο ένα σημείο και ένα HTML DOM element. Το παράθυρο πληροφοριών εμφανίζεται με το κείμενό του στο δοσμένο σημείο του χάρτη και εμφανίζει το DOM element στην περιοχή που περιλαμβάνει. Η μέθοδος openInfoWindowHtml είναι παρόμοια, αλλά παίρνει ένα HTML string σαν δεύτερο όρισμα αντί για ένα DOM element. Ομοίως, η openInfoWindowXslt

παίρνει ένα σημείο, ένα XML DOM element, και το URL ενός XSLT αρχείου. Στη συνέχεια εφαρμόζει το μετασχηματισμό XSLT στο XML για να παράγει τα περιεχόμενα του παραθύρου. Η μέθοδος αυτή μεταφέρει το XSLT ασύγχρονα αν δεν έχει ήδη μεταφερθεί από τον Browser του χρήστη. Εκτός από τα παραπάνω, μπορούμε επίσης να εμφανίσουμε ένα παράθυρο πληροφοριών πάνω από ένα overlay όπως για παράδειγμα ένα εικονίδιο (marker). Για να το κάνουμε αυτό περνάμε ως τρίτο όρισμα ένα pixel offset μεταξύ του καθορισμένου σημείου και του κειμένου του παραθύρου πληροφοριών. Η κλάση GMarker (την οποία θα δούμε παρακάτω) επιτρέπει μεθόδους openInfoWindow οι οποίες χειρίζονται τα pixel offsets αυτόματα βασισμένες στο μέγεθος και σχήμα του εικονιδίου, και συνεπώς δε χρειάζεται να ανησυχεί ο προγραμματιστής για τον υπολογισμό των offsets στην εφαρμογή του.

- **Overlays :** Τα overlays είναι αντικείμενα πάνω στο χάρτη τα οποία είναι «συνδεδεμένα» σε γεωγραφικές συντεταγμένες, και έτσι μετακινούνται όταν ο χρήστης μετακινεί το χάρτη ή κάνει zoom σε αυτόν ή όταν αλλάζει ο τρόπος προβολής (η Google παρέχει διάφορους τρόπους προβολής ενός χάρτη, για παράδειγμα δορυφορική προβολή κτλ). Το Google Maps API περιλαμβάνει δύο τύπους overlays, τα markers που είναι εικονίδια πάνω στο χάρτη και τα polylines που είναι γραμμές που κατασκευάζονται από μία σειρά σημείων.
    - **Markers and Icons :** Ο GMarker constructor παίρνει σαν είσοδο ένα εικονίδιο και ένα σημείο και παράγει ένα μικρό σύνολο από events όπως για παράδειγμα το «κλικ» το οποίο μπορούμε να χειριστούμε στη συνέχεια στον κώδικά μας. Το δυσκολότερο κομμάτι στη δημιουργία ενός marker είναι ο καθορισμός του εικονιδίου. Αυτό είναι δύσκολο διότι είναι μεγάλος ο αριθμός των διαφορετικών εικόνων οι οποίες συνθέτουν ένα απλό εικονίδιο στο Maps API. Παρόλα αυτά, αν θέλει κανείς ένα γενικό εικονίδιο μπορεί να δημιουργήσει ένα GMarker χωρίς να καθορίσει ένα εικονίδιο. Τα εικονίδια είναι συνήθως της μορφής πινέζας, με ένα tip το οποίο εμφανίζεται στη θέση που καθορίζεται από τον GMarker constructor. Κάθε εικονίδιο έχει (τουλάχιστον) μια εικόνα στο προσκήνιο και μία εικόνα σαν σκιά. Υπάρχουν και περισσότερες λεπτομέρειες που αφορούν το πώς πρέπει να καθορίζονται τα εικονίδια και μπορούν να αναζητηθούν στο documentation του Google Maps API .
    - **Polylines :** Ο GPolyline constructor παίρνει σαν είσοδο έναν πίνακα σημείων και δημιουργεί μια σειρά από ευθύγραμμα τμήματα τα οποία ενώνουν αυτά τα σημεία με τη σειρά που δίνονται. Μπορεί κανείς επίσης να καθορίσει το χρώμα, το πάχος και τη φωτεινότητα της γραμμής. Το χρώμα πρέπει να είναι σε δεκαεξαδική μορφή όπως στην HTML.

Περισσότερες λεπτομέρειες για τα polylines μπορούν να αναζητηθούν στο documentation του Google Maps API.

- **Controls :** Για να χρησιμοποιήσουμε έλεγχο πάνω στο χάρτη, όπως μετακίνηση ή zoom ή οποιονδήποτε άλλο έλεγχο, υπάρχει η μέθοδος addControl. Το Maps API έχει ενσωματωμένους τους εξής ελέγχους που μπορεί κανείς να χρησιμοποιήσει στο χάρτη του:
  - GLargeMapControl: Ένας μεγάλου εύρους κίνησης/zoom έλεγχος που χρησιμοποιείται στο Google Maps.
  - GSmallMapControl: Ένας μικρότερος έλεγχος κίνησης/zoom που χρησιμοποιείται στο Google Local.
  - GSmallZoomControl: Ένας μικρός έλεγχος zoom που χρησιμοποιείται στο Google Maps για να εμφανίσει οδηγίες καθοδήγησης.
  - GMapTypeControl: Έλεγχος για να μπορεί ο χρήστης να αλλάζει τους διάφορους τύπους χαρτών (για παράδειγμα Map και Satellite).
- **XML and RPC :** Το Google Maps API προσφέρει μια «factory μέθοδο» για τη δημιουργία XmlHttpRequest αντικειμένων τα οποία «δουλεύουν» σε πρόσφατες εκδόσεις των Internet Explorer, Firefox, Chrome, Opera και Safari. Δίνεται επίσης η δυνατότητα να κάνουμε parsing σε ένα XML έγγραφο με τη στατική μέθοδο GXml.parse, η οποία παίρνει σαν είσοδο ένα XML string. Να σημειώσουμε μόνο ότι το Google Maps API δεν απαιτεί τη χρήση XML ή XmlHttpRequest για να λειτουργήσει καθώς βασίζεται αποκλειστικά σε ένα «καθαρό» JavaScript/DHTML API.

2.2.3 Εύρεση συντεταγμένων περιοχής

Ένας πολύ εύχρηστος ιστοτόπος για την εύρεση γεωγραφικών συντεταγμένων κάποιας περιοχής είναι ο παρακάτω:

http://www.satsig.net/maps/lat-long-finder.htm

3 Διαδικασία Υλοποίησης

3.1 Γενικά Στοιχεία

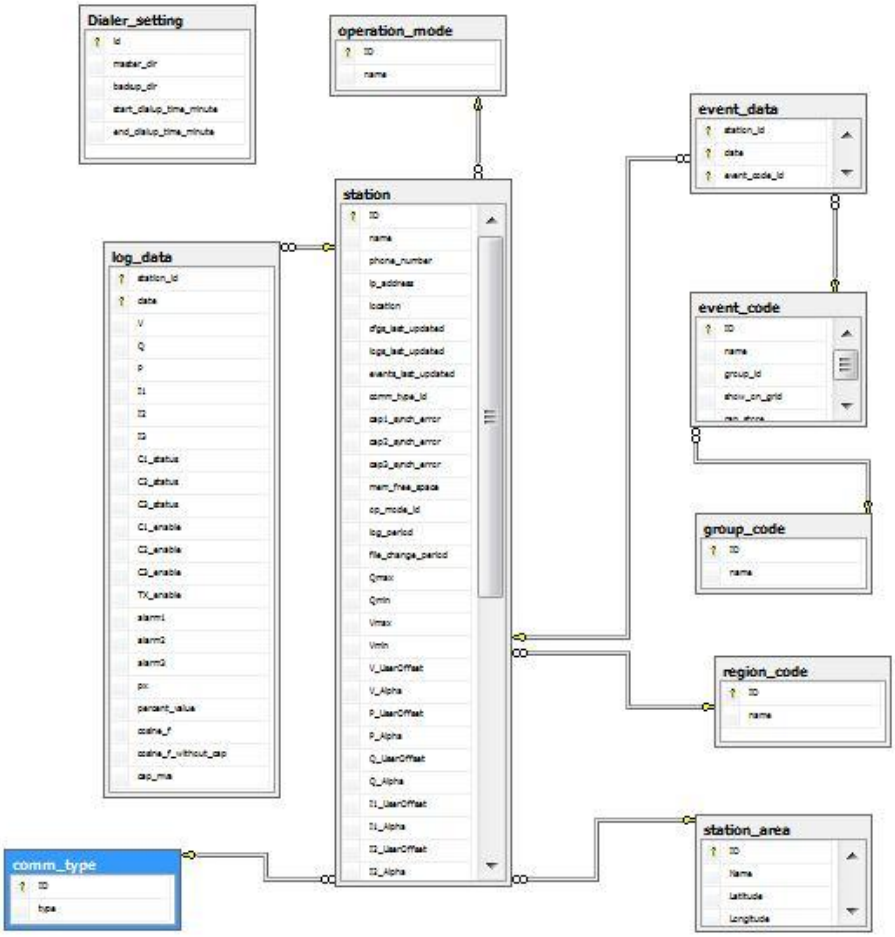
Στόχος της διπλωματικής εργασίας είναι η υλοποίηση ενός ιστοτόπου ο οποίος θα διαβάζει δεδομένα που σχετίζονται με ηλεκτρικούς σταθμούς από μία βάση δεδομένων και παρουσιάζει στο χάρτη τους σταθμούς με τις πληροφορίες που μας ενδιαφέρουν από τη βάση.

Για την υλοποίηση είναι απαραίτητα τα παρακάτω:

- Λειτουργικό σύστημα Microsoft Windows – Κατά την υλοποίηση έγινε χρήση των Windows Vista Home Premium
- Microsoft IIS Server
- Microsoft .NET 3.5 SP1
- Microsoft Visual Studio 2005 με υποστήριξη C#
- Microsoft SQL Server 2005
- Web Browser με υποστήριξη JavaScript – Κατά την υλοποίηση χρησιμοποιήθηκαν οι Internet Explorer 7 και 8, Firefox 2 , Google Chrome
- Για τη χρήση JSON Serializer είναι απαραίτητο το Microsoft.Web.Preview.dll

3.2 Η Βάση Δεδομένων

Η βάση δεδομένων cbcMonGMAP που χρησιμοποιήθηκε περιέχει στοιχεία για ηλετρικούς σταθμούς ανά την Ελλάδα. Χρησιμοποιήσαμε τον SQL Server 2005 και για εργαλείο το Microsoft SQL Server Management Studio 2005. Το σχήμα της βάσης είναι το παρακάτω:



Εικόνα 3.1 : Σχήμα Βάσης

Οι πίνακες της βάσης είναι οι παρακάτω:

- **Station :** Περιέχει δεδομένα για κάθε ηλεκτρικό σταθμό
- **Dialer_Setting:** Περιέχει δεδομένα για ρυθμίσεις επικοινωνιών
- **Operation_mode :** Περιέχει δεδομένα για τις καταστάσεις λειτουργίας των σταθμών
- **event_data :** Περιέχει δεδομένα για τα διάφορα γεγονότα
- **event_code :** Περιέχει δεδομένα κωδικών για τα διάφορα γεγονότα
- **log_data :** Περιέχει δεδομένα για τη λειτουργία των σταθμών
- **group_code :** Περιέχει δεδομένα ομαδοποίησης των σταθμών
- **region_code :** Περιέχει δεδομένα για γεωγραφικές περιοχές που ταξινομούνται οι σταθμοί
- **station_area :** Περιέχει δεδομένα για τις περιοχές που βρίσκονται σταθμοί

- **comm._type :** Περιέχει δεδομένα για τους τρόπους επικοινωνίας

Τέλος πολύ σημαντικό για την υλοποίηση είναι το view GMAP_VIEW που περιέχει όσα δεδομένα χρειαζόμαστε για την απεικόνιση στο χάρτη.

3.3 Δημιουργία Solution στο Visual Studio

Το πρώτο βήμα για την υλοποίηση είναι η δημιουργία ενός solution με τα απαραίτητα projects.

Ανοίγουμε το Visual Studio και δημιουργούμε ένα νέο web site (File -> New Web Site) δίνοντας την ονομασία StationLocator. Στη συνέχεια προσθέτουμε ένα καινούριο project (File -> Add -> New Project -> Visual C# -> Class Library) και δίνουμε την ονομασία StationLibrary.

Το project StationLocator θα αποτελεί τον ιστοτόπο, ενώ από το project StationLibrary θα επιτυγχάνεται η επικοινωνία με τη βάση δεδομένων.

Στη συνέχεια προχωρώντας στην υλοποίηση θα προσθέτουμε καινούρια αρχεία και φακέλους. Αρχικά δημιουργούμε τρεις υποφακέλους στο φάκελο StationLocator:

- **images :** Φάκελος που θα περιέχει αρχεία γραφικών
- **scripts :** Φάκελος που θα περιέχει αρχεία JavaScript
- **style :** Φάκελος που θα περιέχει αρχεία CSS

Τέλος αποθηκεύουμε το solution με την ονομασία StationLocator.

3.4 Υλοποίηση βιβλιοθήκης επικοινωνίας με τη βάση

Για να επιτύχουμε επικοινωνία με τα δεδομένα της βάσης δημιουργούμε ένα αρχείο βιβλιοθήκης των Windows – DLL. Το project το έχουμε δημιουργήσει ήδη και είναι το StationLibrary.

Το StationLibrary θα περιέχει τις απαραίτητες κλάσεις για την ανάγνωση των δεδομένων της βάσης. Για κάθε πίνακα της βάσης δημιουργούμε και την αντίστοιχη κλάση(StationLibrary -> Add -> Class). Η μορφή των κλάσεων είναι σχετικά απλή.

Συγκεκριμένα, η κάθε κλάση:

- Ανήκει στο namespace StationLibrary
- Περιέχει τους αντίστοιχους Constructors
- Περιέχει πεδία και ιδιότητες – properties που αντιστοιχίζονται στα πεδία των πινάκων της βάσης. Τα properties είναι ReadOnly αφού μόνο διαβάζουμε από τη βάση
- Περιέχει μία μέθοδο Load που γεμίζει αντικείμενα της κλάσης
- Περιέχει μία μέθοδο Collect που επιστρέφει πλήθος εγγραφών του πίνακα που εξυπηρετούν ορισμένα κριτήρια
- Περιέχει πρόσθετες μεθόδους για λειτουργίες που αφορούν το συγκεκριμένο πίνακα
- Όλες οι διεργασίες με τη βάση, όπως σύνδεση και ερωτήματα γίνονται μέσω της κλάσης SQLManager

Για παράδειγμα θα δούμε την κλάση CCommType. Πρώτα δηλώνουμε το namespace με τον παρακάτω κώδικα

```
namespace StaionLibrary
```

στη συνέχεια έχουμε τους constructors

```
public CCommType() {
        m_SQLManager = new SQLManager();
 }

public CCommType(long id, string type) {
        m_Id = id;
        m_Type = type;
        m_SQLManager = new SQLManager();
  }
```

στη συνέχεια έχουμε τα πεδία του πίνακα που αντιστοιχίζονται σε πεδία και ιδιότητες της κλάσης

```
private long m_Id;
private string m_Type;


public long ID{
     get {
           return m_Id;
     }
 }

 public string Type{
```

```
        get {
              return m_Type;
        }
  }
```

Έπειτα ακολουθεί η μέθοδος Load

```
public void Load(long id) {
      string sSQL;
      OleDbConnection oConn = new OleDbConnection();
      DataSet oDs = new DataSet();

      try {
            sSQL = "SELECT * FROM comm_type WHERE ID =
            Convert.ToString(id);
            oDs = m_SQLManager.ExecuteSQL(sSQL, ref oConn);

                if (oDs.Tables[0].Rows.Count > 0) {
            m_Id = Convert.ToInt64(oDs.Tables[0].Rows[0]["ID"]);
            m_Type = Convert.ToString(oDs.Tables[0].Rows[0]["type"]);
                }
            } catch {
      }
 }
```

Και τέλος έχουμε τη μέθοδο Collect

```
public DataSet Collect(string fields,string ids, string type) {
            string sSQL;
      OleDbConnection oConn = new OleDbConnection();
      DataSet oDs = new DataSet();
      try {
            sSQL = "SELECT " + fields + " FROM comm_type";
            if (ids != "")
            m_SQLManager.AddCriteria(ref sSQL, "ID IN (" + ids + ")",
      true);      if (type != "")
                        m_SQLManager.AddCriteria(ref sSQL,
                        m_SQLManager.SQLLike("type", type), !
                        (sSQL.Contains("WHERE")));
                        oDs = m_SQLManager.ExecuteSQL(sSQL, ref
                        oConn);
                        return oDs;
      }
      catch {

            return null;
      }
}
```

Όλες οι κλάσεις του StationLibrary ακολουθούν την παραπάνω μορφή.

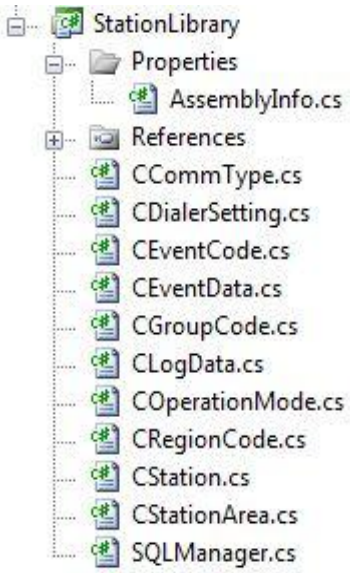
Η κλάση SQLManager

Η συγκεκριμένη κλάση ανήκει στο StationLibrary και προσφέρει βοηθητικές μεθόδους για την επικοινωνία με μία βάση δεδομένων.

Πιο συγκεκριμένα περιέχει τις μεθόδους:

- **EncryptConnectionString :** Κρυπτογράφιση του ConectionString
- **DecryptConnectionString :** Απόκρυπτογράφιση του ConectionString
- **ConnectSQL :** Σύνδεση σε βάση SQL Server
- **ExecuteSQL :** Εκτέλεση ερωτήματος στη βάση
- **AddCriteria :** Προσθήκη WHERE clause σε ερώτημα
- **EnQuote :** Τοποθέτηση αλφαριθμητικου μεταξύ ‘ ‘
- **SQLLike :** Προσθήκη κριτηρίου LIKE
- **SQLDate :** Προσθήκη κριτηρίου ημερομηνίας

Η μορφή του StationLibrary είναι η παρακάτω:



Εικόνα 3.2 : Μορφή StationLibrary

Κάνοντας Build έχουμε έτοιμο το dll μας και μπορούμε να το χρησιμοποιούμε όπου θέλουμε.

3.5 Υλοποίηση της βασικής σελίδας

Το επόμενο βήμα είναι η δημιουργία της σελίδας που θα βλέπει ο χρήστης στον Browser. Προσθέτουμε στο StationLocator μία καινούρια Web Form με την ονομασία Default.aspx(StationLocator -> Add New Item -> Web Form).

Θέλουμε να προσθέσουμε ένα Google Map στη συγκεκριμένη φόρμα. Για το λόγο αυτό πρέπει να πάρουμε ένα κλειδί από τη Google που μας δίνει το δικαίωμα να χρησιμοποιούμε το χάρτη. Στη σελίδα

http://code.google.com/intl/el-GR/apis/maps/signup.html

δίνουμε για My web site URL το http://localhost/StationLocator και με το Generate API Key παίρνουμε το κλειδί μας.

Ο σχεδιασμός της σελίδας βασίζεται σε tables και divs. Θα αναφερθούμε σε βασικά σημεία.

Επειδή θέλουμε να χρησιμοποιήσουμε Google Maps στο head της σελίδας προσθέτουμε

```
<script
src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=ABQIAAAAUuk
M5G9wy93VqONz3LwvWhQ6ychhZpw5khizo-KZI37zlHpfPBTM6dLNtr2tA-
ZRsWac88nTJsuAYg" type="text/javascript"></script>
```

Χρησιμοποιούμε το κλειδί που πήραμε πριν. Επίσης προσθέτουμε σύνδεση με StyleSheet CSS.

```
<link type="text/css" href="style/main.css" rel="STYLESHEET"/>
```

Θα χρησιμοποιήσουμε ASP.NET AJAX γι'αυτό στο body προσθέτουμε

```
 <asp:ScriptManager ID="ScriptManager1" runat="server">
  <Services><asp:ServiceReference Path="StationAreasWS.asmx"
/></Services>
  <Scripts><asp:ScriptReference Path="~/scripts/MapFunctions.js"
/></Scripts>
  </asp:ScriptManager>
```

Βλέπουμε ότι έχουμε προσθέσει αναφορά σε δύο αρχεία.

- **StationAreasWS.asmx :** Είναι web service για την επικοινωνία των google maps – βάσης στο server με ASP.NET AJAX
- **MapFunctions.js :** Περιέχει τις λειτουργίες του χάρτη σε JavaScript

Θα παρουσιαστούν αναλυτικότερα αργότερα.

Επιπλέον προσθέσαμε τρία κουμπιά που θα κάνουν κάποιες βασικές λειτουργίες στο χάρτη. Για να μην κάνουν πλήρες PostBack, τα τοποθετούμε μέσα σε ASP:UpdatePanel.

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
 <ContentTemplate>
       <div id = "buttons">
            <table align="left">
      <tr><td><asp:Button ID="btnRefresh" runat="server"
     Text="ΑΝΑΝΕΩΣΗ       ΧΑΡΤΗ" OnClientClick="DrawMap()"
     width="175px"/></td></tr>
     <tr><td><asp:Button ID="btnClear" runat="server"
     Text="ΚΑΘΑΡΙΣΜΟΣ ΧΑΡΤΗ" OnClientClick="ClearMarkers()"
     Width="175px" /></td></tr>
    <tr><td><asp:Button ID="btnMapFocusOnCenter" runat="server"
    Text="ΕΠΑΝΑΦΟΡΑ ΧΑΡΤΗ" OnClientClick="MapFocusOnCenter()"
    Width="175px" /></td></tr>
          </table>
      </div>
</ContentTemplate>
</asp:UpdatePanel>
```

Τέλος προσθέτουμε το div που θα περιέχει το χάρτη Google.

```
<td><div id="map"></div></td>
```

Οι λεπτομέρειες του χάρτη βρίσκονται στο StyleSheet

```
#map{
      width:600px;
      height:500px;
}
```

Η κάθε ιστοσελίδα ASP.NET έχει και το αντίστοιχο αρχείο κώδικο με κατάληξη aspx.cs. Προφανώς το ίδιο ισχύει και για τη σελίδα μας.

Στο αρχείο Default.aspx.cs γίνεται η αρχικοποίηση των drop down menus που υπάρχουν και αναφέρονται στη γεωγραφική περιοχή, την κατάσταση και τον τρόπο επικοινωνιάς. Αυτό επιτυγχάνεται με τρείς μεθόδους που καλούνται κατά τη φόρτωση της σελίδας:

- **FillRegions :** Διαβάζει τις περιοχές από τον πίνακα region_code με τη βοήθεια της μεθόδου Collect της κλάσης CRegionCode
- **FillCommTypes :** Διαβάζει τους τρόπους επικοινωνίας από τον πίνακα comm_type με τη βοήθεια της μεθόδου Collect της κλάσης CCommType

- **FillOperationModes :** Διαβάζει τις καταστάσεις από τον πίνακα operation_mode με τη βοήθεια της μεθόδου Collect της κλάσης COperationMode

Είναι προφανές ότι η σελίδα μας χρησιμοποιεί το StationLibrary.dll . Για να είναι δυνατό αυτό πρέπει να προσθέσουμε ως reference το dll στο Web Site StationLocator (StationLocator -> Add Reference -> Projects -> StationLibrary)

Κατά τη φόρτωση της σελίδας με τη μέθοδο Load θέτουμε και τη σύνδεση με τη βάση.

3.6 Το Web Service StationAreasWS

Για την επικοινωνία των Google Maps με το Server είναι απαραίτητη η χρήση ASP.NET AJAX. Με τον τρόπο αυτό ο κώδικας JavaScript των Google Maps έχει τη δυνατότητα κλήσης μεθόδων που βρίσκονται στο Server. Αυτό επιτυγχάνεται με τη χρήση ενός Web Service.

Στη συγκεκριμένη περίπτωση δημιουργούμε ένα Web Service στο StationLocator (StationLocator -> Add New Item -> Web Service) με την ονομασία StationAreasWS.

Προκύπτουν δύο αρχεία. Το StationAreasWS.asmx και το StationAreasWS.cs. Το StationAreasWS.asmx είναι το αρχείο περιγραφής των μεθόδων που προσφέρει το Web Service ενώ το StationAreasWS.cs περιέχει την υλοποίηση.

Το συγκεκριμένο Web Service προσφέρει τις δύο μεθόδους που χρειαζόμαστε για τον Περιηγητή Ηλεκτρικών Σταθμών

Πιο αναλυτικά, η κλάση StationAreasWS.cs κληρονομεί την System.Web.Services.WebService και περιέχει δύο μεθόδους.

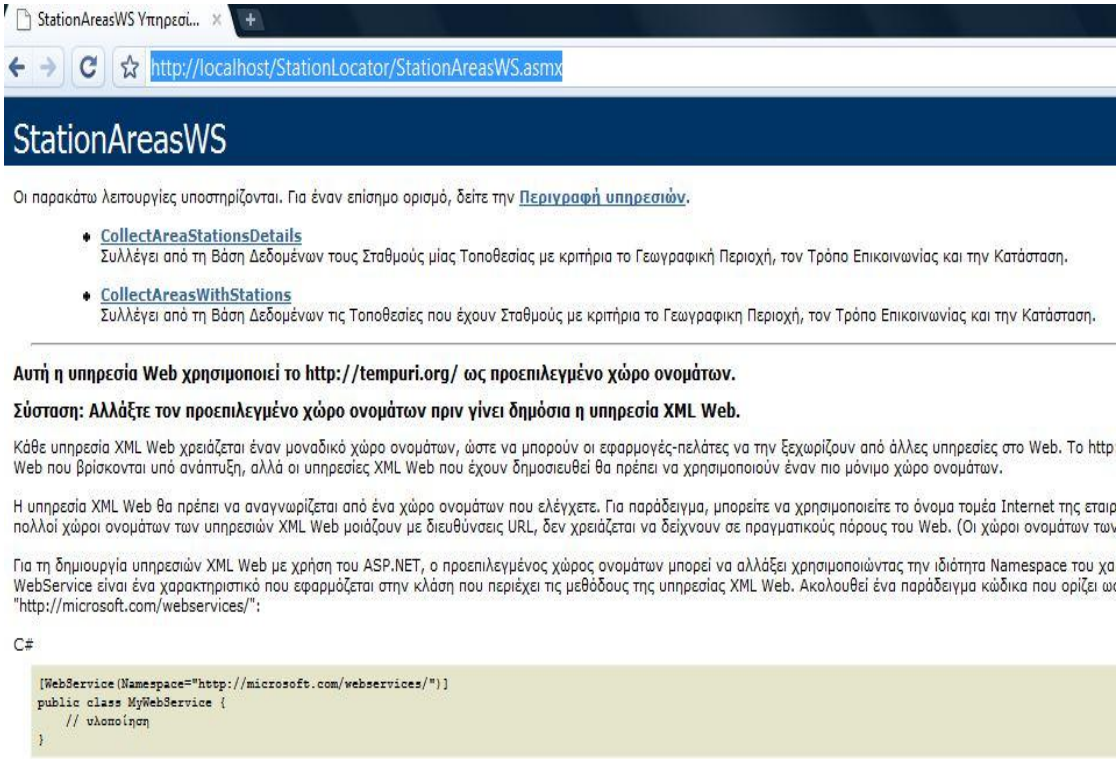
- **CollectAreasWithStations :** Συλλέγει από τη Βάση Δεδομένων τις Τοποθεσίες που έχουν Σταθμούς με κριτήρια το Γεωγραφικη Περιοχή, τον Τρόπο Επικοινωνίας και την Κατάσταση. Αυτό επιτυγχάνεται με την κληση της μεθόδου CollectStationAreas της κλάσσης CStation που ανήκει στο StationLibrary.
- **CollectAreaStationsDetails :** Συλλέγει από τη Βάση Δεδομένων τους Σταθμούς μίας Τοποθεσίας με κριτήρια το Γεωγραφική Περιοχή,τον Τρόπο Επικοινωνίας και την Κατάσταση. Αυτό επιτυγχάνεται με την κληση της μεθόδου CollectAreaStationsDetails της κλάσσης CStationArea που ανήκει στο StationLibrary.

Οι παραπάνω μέθοδοι θα κληθούν από τη JavaScript και γι’αυτό το λόγο πρέπει να έχουν την ιδιότητα [ScriptService]. Οι δύο μέθοδοι επιστρέφουν απάντηση σε μορφή XML που περιγράφει τα δεδομένα που επιστρέφονται από την κλήση τους.

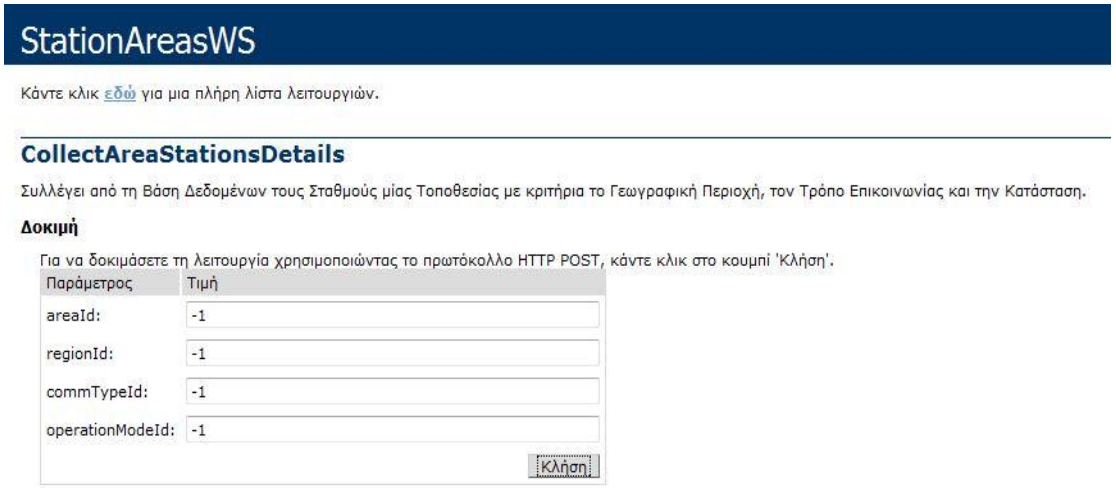
Το StationAreasWS μπορούμε να το δοκιμάσουμε μέσω της διεύθυνσης

http://localhost/StationLocator/StationAreasWS.asmx

και από εκεί να καλέσουμε τις μεθόδους και να δούμε τις απαντήσεις του WebService.



Εικόνα 3.3 : Η σελίδα του Web Service



Εικόνα 3.4 : Κλήση Μεθόδου

```
−<diffgr:diffgram>
  −<NewDataSet>
    −<sql_DATA diffgr:id="sql_DATA1" msdata:rowOrder="0">
       <AreaName>ΚΑΛΑΜΟΣ</AreaName>
       <StationName>ΚΑΛΑΜΟΣ</StationName>
       <StationLastLogEntry>12/02/2009 23:59</StationLastLogEntry>
       <C1>ON</C1>
       <C2>OUT</C2>
       <C3>OFF</C3>
       <V>21.91</V>
       <P>0.17</P>
       <Q>0.06</Q>
       <alarm1>false</alarm1>
     </sql_DATA>
    −<sql_DATA diffgr:id="sql_DATA2" msdata:rowOrder="1">
       <AreaName>ΧΑΛΚΙΔΑ</AreaName>
       <StationName>ΧΑΛΚΙΔΑ</StationName>
       <StationLastLogEntry>12/02/2009 23:57</StationLastLogEntry>
       <C1>ON</C1>
       <C2>OFF</C2>
       <C3>OFF</C3>
       <V>20.83</V>
       <P>9.54</P>
       <Q>-1.48</Q>
       <alarm1>false</alarm1>
     </sql_DATA>
```

Εικόνα 3.5 : Τμήμα XML της απάντησης

Τα δεδομένα αυτά είναι πολύ εύκολα προσπελάσιμα από το Google Maps API.

3.7 Χρήση Google Maps API

Το αρχείο στο οποίο γίνεται η χρήση του Google Maps API είναι το MapFunctions.js το οποίο προσθέτουμε στο φάκελο scripts.

Παρακάτω γίνεται μία παρουσίαση της λειτουργικότητας του αρχείου εξηγώντας τα διάφορα τμήματα κώδικα του αρχείου.

Όπως είχαμε αναφέρει και στην υλοποίηση του Default.aspx, το αρχείο MapFunctions.js δηλώνεται στο συγκεκριμένο αρχείο ώστε να μπορούν να προσπελαστούν από τη σελίδα οι διάφορες συναρτήσεις του MapFunctions.js

Αναλυτικότερα έχουμε:

- Αρχικοποιούμε το χάρτη δίνοντας το zoom και τις συντεταγμένες κέντρου του.

```
var centerLatitude = 37.9853;
var centerLongitude = 23.7191;
var centerDescription = 'ΑΘΗΝΑ';
var startZoom = 6;
```

- Δηλώνουμε τις καθολικές μεταβλητές που περιγράφουν την κατάσταση του χάρτη

```
var map;
var region;
var commType;
var operationMode;
var currentMarker;
```

Η μεταβλητή map είναι ο χάρτης, η region είναι η επιλεγμένη γεωγραφική περιοχή, η commType είναι ο τρόπος επικοινωνίας ,η operationMode είναι η κατάσταση και η currentMarker είναι η τρέχουσα επιλεγμένη περιοχή.

- Ορίζουμε τα εικονίδια που θα φαίνονται στις περιοχές. Οι εικόνες είναι τοποθετημένες στο φάκελο images.

```
var customIcon = new GIcon(G_DEFAULT_ICON);
customIcon.iconSize = new GSize(20,34);
customIcon.shadowSize = new GSize(0, 0);

var customIcons = [];
customIcons['green'] = new GIcon(customIcon,
'/StationLocator/images/marker_green.png');
customIcons['red'] = new GIcon(customIcon,
'/StationLocator/images/marker_red.png');
```

- Ακολουθούν βοηθητικές συναρτήσεις που χρωματίζουν κελιά σε πίνακες πληροφοριών με βάση τα δεδομένα του σταθμού.
- Υλοποιούμε τη συνάρτηση Init που φορτώνει το χάρτη στη σελίδα και τον αρχικοποιεί δίνοντας τιμές στις καθολικές μεταβλητές. Ελέγχουμε αν ο Browser υποστηρίζει Google Maps και αν δεν υποστηρίζει εμφανίζεται το αντίστοιχο μήνυμα.

```
function Init() {
    if (GBrowserIsCompatible()) {
        region = -1;
        commType = -1;
        operationMode = -1;
        currentMarker = null;
        map = new GMap2($get("map"));
    map.addControl(new GSmallMapControl());
       map.setCenter(new GLatLng(centerLatitude,
centerLongitude), startZoom);
    }
    else {

   $get('formStationLocator').innerHTML = '<h1>Ο Browser δεν
υποστηρίζει Google Maps!</h1>';
    }
}
```

- Στη συνέχεια προχωρούμε στη σχεδίαση των περιοχών στο χάρτη. Το συγκεκριμένο σημείο είναι το πλέον σημαντικό γιατί γίνεται και η επικοινωνία με τη βάση δεδομένων.

  Με τη συνάρτηση ClearMarkers σβήνονται όλες οι περιοχές που φαίνονται στο χάρτη.

```
function ClearMarkers() {
  map.clearOverlays();
}
```

  Η συνάρτηση MapFocusOnCenter επαναφέρει το χάρτη στο κέντρο του.

```
function MapFocusOnCenter() {
    if (currentMarker != null) {
                currentMarker.closeInfoWindow();
                 currentMarker = null;
   }
    map.setCenter(new GLatLng(centerLatitude,
centerLongitude), startZoom);
}
```

  Η συνάρτηση DrawMap συλλέγει από τη βάση τις περιοχές που ικανοποιούν τα κριτήρια γεωγραφικής περιοχής, τρόπου επικοινωνίας και κατάστασεις. Στη συνέχεια τις εμφανίζει στο χάρτη.

  Η συλλογή γίνεται με ASP.NET AJAX. Καλείται η μέθοδος StationLocator.StationAreasWS.CollectAreasWithStations του Web Service.

```
function DrawMap() {
  ClearMarkers();
   region = $get('ddlRegion').value;
  commType = $get('ddlCommType').value;
```

```
   operationMode = $get('ddlOperationMode').value;

   StationLocator.StationAreasWS.CollectAreasWithStations(re
   gion,commType,operationMode,OnRequestAreasSuccess,
   OnRequestAreasFail);
}
```

Στην κλήση AJAX υπάρχουν δύο περιπτώσεις. Είτε να είναι επιτυχής και καλείται μετά η συνάρτηση OnRequestAreaSuccess, είτε αποτυγχάνει και καλείται η OnRequestAreasFail.

Η OnRequestAreasFail ειδοποιεί το χρήστη ότι η ενημέρωση του χάρτη απέτυχε.

```
function OnRequestAreasFail() {
    alert('Η ενημέρωση του χάρτη απέτυχε!');
}
```

Η OnRequestAreasSuccess συλλέγει τα δεδομένα της βάσης στη μεταβλητή results και μέσω του JSON γίνεται εύκολα η προσπέλασή τους σαν να είναι αντικείμενα χωρίς καθόλου επεξεργασία XML. Ανάλογα με τα δεδομένα προσθέτει τις περιοχές στο χάρτη με την addMarker.

```
function OnRequestAreasSuccess(results) {
   var areaId;
   var areaName;
   var areaLatitude;
   var areaLongitude;
   var areaHasProblem;
   var i;

   if (results != null) {
   for (i = 0; i < results.tables[0].rows.length; i++) {
   areaId = results.tables[0].rows[i].AreaId;
   areaName = results.tables[0].rows[i].AreaName;
   areaLatitude = results.tables[0].rows[i].AreaLatitude;
   areaLongitude = results.tables[0].rows[i].AreaLongitude;
   areaHasProblem =
   results.tables[0].rows[i].AreaHasProblem;
         addMarker(areaId, areaName, areaLatitude,
         areaLongitude, areaHasProblem);
         }
   }
}
```

Η addMarker με βάση τα δεδομένα από τη βάση ζωγραφίζει μία περιοχή με πράσινο αν δεν έχει πρόβλημα και με κόκκινο αν έχει πρόβλημα. Στη συνέχεια ορίζει ένα event σε κάθε σύμβολο περιοχής, έτσι ώστε όταν γίνει κλικ πάνω στην περιοχή να εμφανίζεται το παράθυρο πληροφοριών κάθε περιοχής με δεδομένα για κάθε σταθμό αυτής.

```
function addMarker(areaId, areaName, areaLatitude,
areaLongitude,areaHasProblem) {
   var markerOptions;
if (areaHasProblem)
            markerOptions = { icon: customIcons['red'] , title
: areaName };

 else
            markerOptions = { icon: customIcons['green'],
title: areaName };

 var marker = new GMarker(new GLatLng(areaLatitude,
areaLongitude),markerOptions);
GEvent.addListener(marker, "click", function(){
ShowInfoWindow(marker,areaId); });
       map.addOverlay(marker);
}
```

Η εμφάνιση του πίνακα σταθμών μίας περιοχής με κλικ γίνεται με τη βοήθεια της συνάρτησης ShowInfoWindow. Η συγκεκριμένη συνάρτηση κάνει επίσης κλήση AJAX στο server. Καλεί την CollectAreaStationDetails του Web Service. Αντίστοιχα σε περίπτωση επιτυχούς κλήσεως καλείται η OnRequestAreaStationDetailsSuccess, ενώ σε περίπτωσης αποτυχίας καλείται η OnRequestAreaStationsDetailsFail.

```
function ShowInfoWindow(marker, areaId) {
      currentMarker = marker;

      StationLocator.StationAreasWS.CollectAreaStationsDetails(
areaId, region, commType, operationMode,
          OnRequestAreaStationsDetailsSuccess,
OnRequestAreaStationsDetailsFail);
}
```

Η OnRequestAreaStationsDetailsFail ενημερώνει το χρήστη ότι η συλλογή των σταθμών της περιοχής απέτυχε.

```
function OnRequestAreaStationsDetailsFail(results) {
  alert('Η συλλογή των Σταθμών της Τοποθεσίας απέτυχε!');
}
```

Η OnRequestAreaStationsDetailsSuccess παίρνει τα δεδομένα από την κλήση στη μεταβλητή results και τα διαβάζει διαδοχικά. Στη συνέχεια εμφανίζει το infoWindow των Google Maps για την περιοχή που έγινε κλικ. Στο παράθυρο περιέχεται πίνακας με τους σταθμούς και τα δεδομένα τους για την αντίστοιχη

περιοχή. Σημειώνουμε ότι και εδώ γίνεται χρήση του JSON. Για οπτικούς λόγους σε περίπτωση που ο αριθμός των σταθμών είναι μεγαλύτερος των τριών προσθέτουμε μπάρα κύλισης στο παράθυρο πληροφοριών ώστε να μην γίνεται αυτό υπερβολικά μεγάλο. Επιπλέον πρέπει να τονιστεί ότι το περιεχόμενου του παραθύρου πληροφοριών είναι HTML κώδικας που χτίζεται δυναμικά από τη συγκεκριμένη συνάρτηση. Το παράθυρο εμφανίζεται με την OpenInfoWindowsHTML του Google Maps API.

```
function OnRequestAreaStationsDetailsSuccess(results) {
	var areaName;
	var stationName;
	var stationLastLogEntry;
	var stationC1;
	var stationC2;
	var stationC3;
	var stationV;
	var stationP;
	var stationQ;
	var stationAlarm;
	var i;
	var stationsNumber;
	var html;

	if (results!=null) {
	areaName = results.tables[0].rows[0].AreaName;

	stationsNumber = results.tables[0].rows.length;
	if(stationsNumber > 3)
	html ='<div style=width:470px;height:110px><div
	style=overflow:auto;width:470px;height:110px>';
	else
	html = '';

	html += '<div><b>' + 'Τοποθεσία Σταθμών: ' + areaName +
	'</b></div>' +
	'<table border="1" style="font-size:small"><tr
	align="center">' +
	'<td>Ονομασία</td><td>Τελ.Εγγραφή</td><td>C1</td>' +
	'<td>C2</td><td>C3</td><td>V</td><td>P</td>' +
	'<td>Q</td><td>Συν/μός</td></tr>';

	for (i = 0; i < results.tables[0].rows.length; i++) {
	stationName = results.tables[0].rows[i].StationName;
	stationLastLogEntry =
	results.tables[0].rows[i].StationLastLogEntry;
	stationC1 = results.tables[0].rows[i].C1;
	stationC2 = results.tables[0].rows[i].C2;
	stationC3 = results.tables[0].rows[i].C3;
	stationV = results.tables[0].rows[i].V;
	stationP = results.tables[0].rows[i].P;
	stationQ = results.tables[0].rows[i].Q;
	stationAlarm = results.tables[0].rows[i].alarm1;
	html += '<tr><td align="left">' + stationName + '</td>' +
	'<td align="right">' + stationLastLogEntry + '</td>' +
```

```
        '<td align="center" style="background-color:' +
        getColourForC(stationC1) + '">' + stationC1 +
        '</td>' +


            '<td align="center" style="background-color:' +
        getColourForC(stationC2) + '">' + stationC2 +
        '</td>' +
            '<td align="center" style="background-color:' +
    getColourForC(stationC3) + '">' + stationC3 + '</td>'
+

'<td align="right">' + stationV + '</td>' +
'<td align="right">' + stationP + '</td>' +
'<td align="right">' + stationQ + '</td>' +
      '<td align="center" style="background-color:' +
      getColourForAlarm(stationAlarm) + '">' +
getTextForAlarm(stationAlarm)+'</td></tr>';

}
if(stationsNumber > 3)
html += '</div></div>';

html += '</table>';
currentMarker.openInfoWindowHtml(html);
}
}
```

- Στο τέλος του MapFunctions.js δηλώνουμε ότι όταν γίνετια φόρτωση του παραθύρου εκτελείται η init, ενώ με το κλείσιμο εκτελείται η GUnload για απελευθέρωση του χάρτη και αποσυμφόριση του συστήματος.

3.8 Το αρχείο web.config

Το αρχείο web.config αποτελεί το αρχείο ρυθμίσεων κάθε ASP.NET εφαρμογής. Η βασική του μορφή δημιουργείται άμεσα κατά τη δημιουργία του project της εφαρμογής από το Visual Studio.

Στη συνέχεια ο προγραμματιστής προσθέτει επιπλέον στοιχεία στο αρχείο.

Όσον αφορά την εφαρμογή του Περιηγητή Ηλεκτρικών Σταθμών οι παρεμβάσεις μας στο web.config έγιναν σε δύο σημεία. Αρχικά θέσαμε το connectionString στο κατάλληλο σημείο του αρχείου. Η ρύθμιση αυτή καθορίζει και τη σύνδεση του ιστοτόπου με τη βάση δεδομένων. Προφανώς λανθασμένη ή ελλιπής πληροφορία έχει ως αποτέλεσμα τη μη σύνδεση με τη βάση δεδομένων. Για το λόγο αυτό προσθέσαμε στο section με τα connectionStrings του web.config τον παρακάτω κώδικα :

```
<add name="StationLocatorConnectionString"
connectionString="Provider=SQLOLEDB.1;Persist Security
Info=False;user id=sa;password=sapk;Initial Catalog=cbcMonGMAP;Data
Source=PANAGIOTIS-PC\SQL2005;"/>
```

Η δεύτερη και πολύ σημαντική αλλαγή είναι η προσθήκη των παρακάτω γραμμών :

```
<system.web.extensions>
      <scripting>
            <webServices>
                  <jsonSerialization>
                        <converters>
                              <add name="DataSetConverter"
type="Microsoft.Web.Preview.Script.Serialization.Converters.DataSetCo
nverter, Microsoft.Web.Preview"/>
                              <add name="DataRowConverter"
type="Microsoft.Web.Preview.Script.Serialization.Converters.DataRowCo
nverter, Microsoft.Web.Preview"/>
                              <add name="DataTableConverter"
type="Microsoft.Web.Preview.Script.Serialization.Converters.DataTable
Converter, Microsoft.Web.Preview"/>
                        </converters>
                  </jsonSerialization>
            </webServices>
      </scripting>
</system.web.extensions>
```

Οι παραπάνω γραμμές έπρεπε να προστεθούν σε συνδυασμό με την προσθήκη reference στο Microsoft.Web.Preview.dll, ώστε τα dataset που προέρχονταν από τις κλήσεις AJAX στο Web Service να μπορούν να διαβαστούν από τη JavaScript.

Χωρίς αυτή την προσθήκη πιθανότητα οι κλήσεις AJAX θα αποτύγχαναν λόγω σφάλματος κυκλικής αναφοράς.

3.9 Χρήση StyleSheet CSS

Για την καλύτερη εμφάνιση του ιστοτόπου προστέθηκε το αρχείο main.css στο φάκελο style με σκοπό τη βελτίωση της εμφάνισης του ιστοτόπου. Το συγκεκριμένο αρχείο ακολουθεί επακριβώς τους κανόνες του css.

3.10 Μεταγλώττιση και Έκδοση του Περιηγητή

Πλέον ο Περιηγητής Ηλεκτρικών Σταθμών έχει ολοκληρωθεί και απομένει η μεταγλώττισή του και η έκδοση του στον IIS.

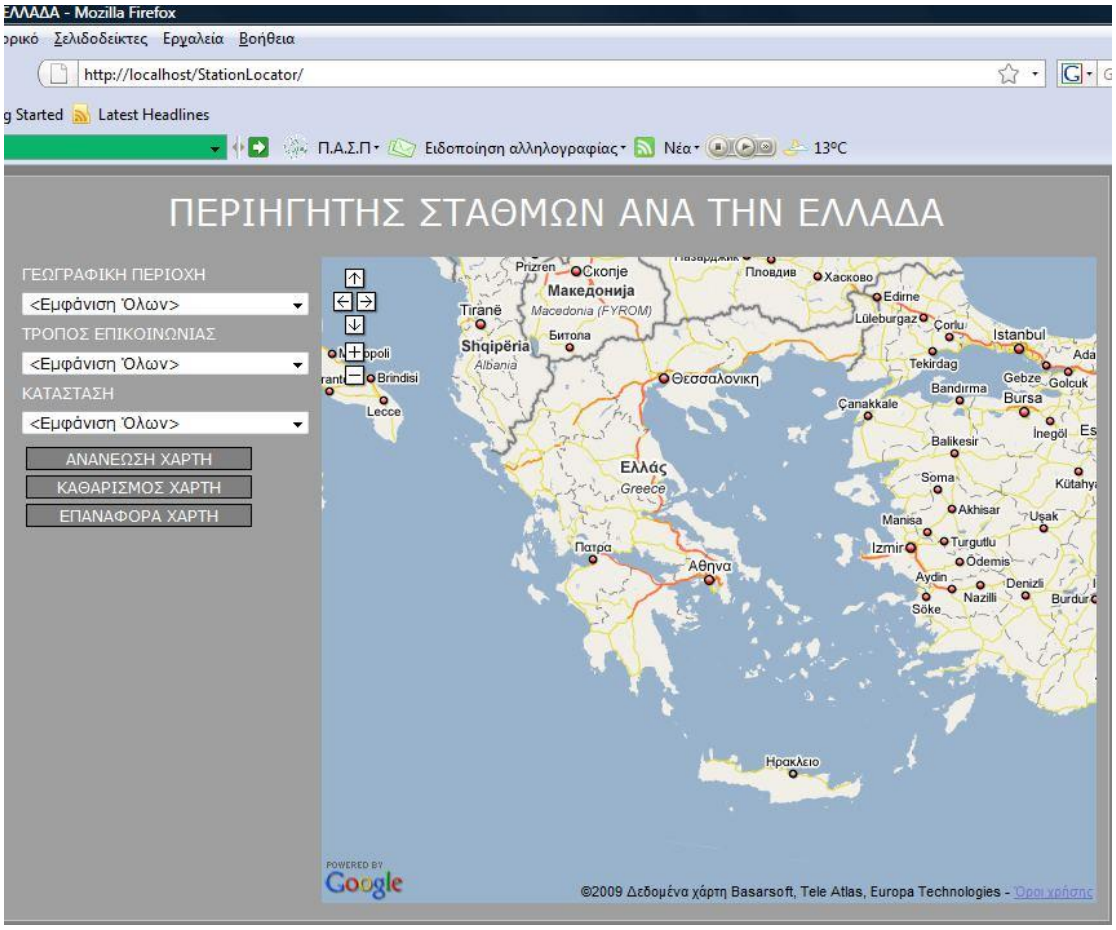
Αυτό γίνεται πολύ εύκολα μέσα από το Visual Studio.Η μεταγλώττιση γίνεται με τη χρήση των πλήκτρων Ctrl + Shift + B.

Για την έκδοση απαιτείται δεξί κλικ στο StationLocator και στη συνέχεια Publish επιλέγοντας σε ποια τοποθεσία θα γίνει η έκδοση. Είναι πολύ βασικό αυτή η τοποθεσία να είναι η ίδια με αυτή που χρησιμοποιήσαμε για να πάρουμε το κλειδί για τη χρήση των Google Maps.

Στην περίπτωσή μας η διεύθυνση είναι η

http://localhost/StationLocator

Με την πληκτρολόγηση της στον Browser ανοίγει και η σελίδα!



Εικόνα 3.6 : Ο Περιηγητής

4 Παρουσίαση του Περιηγητή Ηλεκτρικών Σταθμών

Ο Περιηγητής Ηλεκτρικών Σταθμών είναι μία ιστοσελίδα που ενσωματώνει ένα χάρτη Google που επικεντρώνεται στον Ελλαδικό χώρο και εμφανίζει τους Ηλεκτρινούς Σταθμούς ανά την Ελλάδα σύμφωνα με δεδομένα που διαβάζει από μία βάση δεδομένων.

Το γραφικό περιβάλλον αποτελείται από έναν χάρτη. Επιπλέον εμφανίζονται τρεις dropdown λίστες που περιέχουν τα κριτήρια εμφάνισης ηλεκτρικών σταθμών και είναι η γεωγραφική περιοχή, ο τρόπος επικοινωνίας και η κατάσταση. Επιπλέον περιέχονται τρία κουμπιά με λειτουργίες την ανανέωση χάρτη, τον καθαρισμό χάρτη και την επαναφορά χάρτη.

Με τη χρήση του κουμπιού της ανανέωσης χάρτη εμφανίζονται οι περιοχές που περιέχουν ηλεκτρικούς σταθμούς και πληρούν τα κριτήρια των τριών λιστών. Αν η περιοχή περιέχει σταθμό με πρόβλημα εμφανίζεται ως κόκκινη, διαφορετικά εμφανίζεται ως πράσινη. Στη συνέχεια με κλικ πάνω σε περιοχή, αναδύεται παράθυρο πληροφοριών των Google Maps που περιέχει έναν πίνακα με τους σταθμούς της περιοχής και ορισμένα χαρακτηριστικά στοιχεία κάθε σταθμού, χρησιμοποιώντας διάφορους χρωματισμούς.

Με τη χρήση του κουμπιού καθαρισμού του χάρτη γίνεται το αυτονόητο. Εξαφανίζονται όλες οι περιοχές από το χάρτη. Με τη χρήση του κουμπιού επαναφορά χάρτη επανατοποθετείται ο χάρτης στο αρχικό του κέντρο.

Σε περίπτωση σφάλματος κατά την ανάγνωση από τη βάση δεδομένων εμφανίζεται αντίστοιχο μήνυμα προς το χρήστη.

Αν ο Web Browser δεν υποστηρίζει Google Maps εμφανίζεται μία κενή σελίδα με το αντίστοιχο μήνυμα.

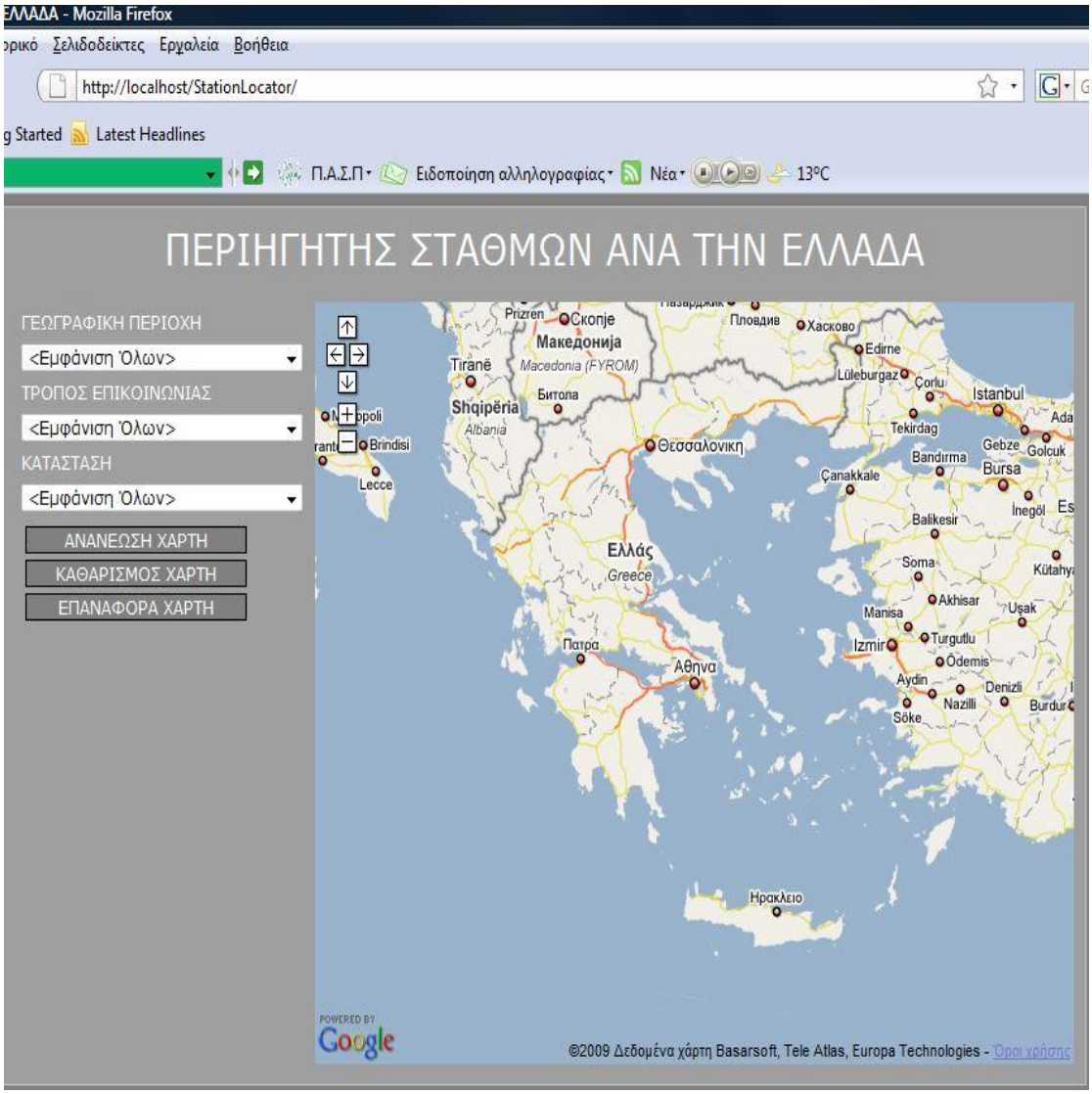
Ο χάρτης υποστηρίζει zoom in , zoom out , μετακίνηση.

Τέλος πρέπει να σημειωθεί ότι ο Περιηγητής ηλεκτρικών σταθμών δοκιμάστηκε σε περιβάλλοντα

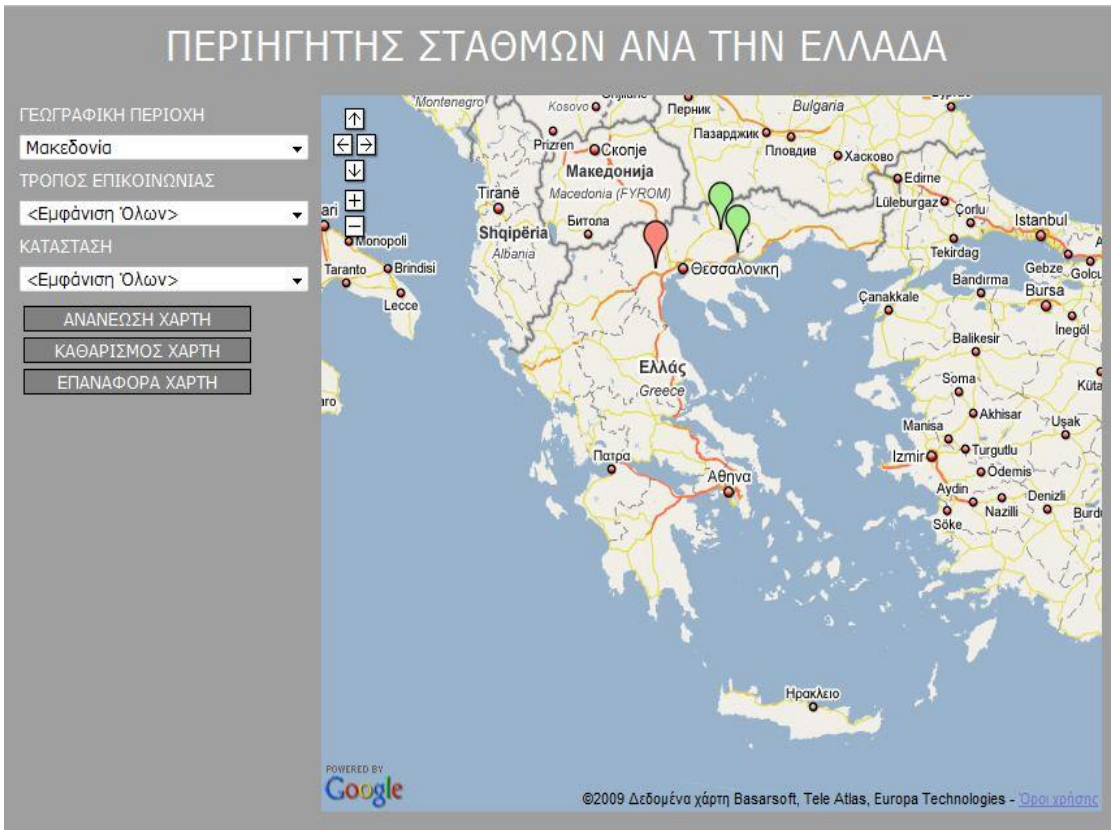
- Microsoft Internet Explorer 7 και 8
- Firefox
- Chrome

- Opera
- Safari

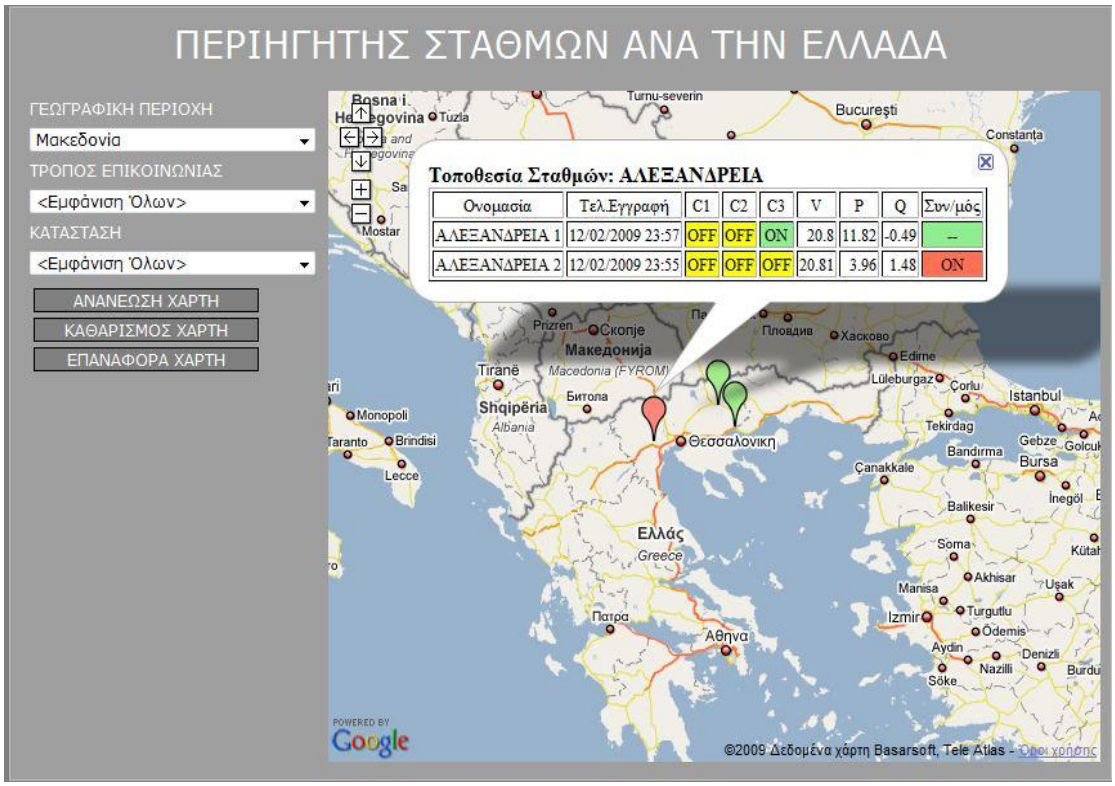
Παρακάτω ακολουθούν οθόνες από τη λειτουργία του Περιηγητή Ηλεκτρικών Σταθμών.



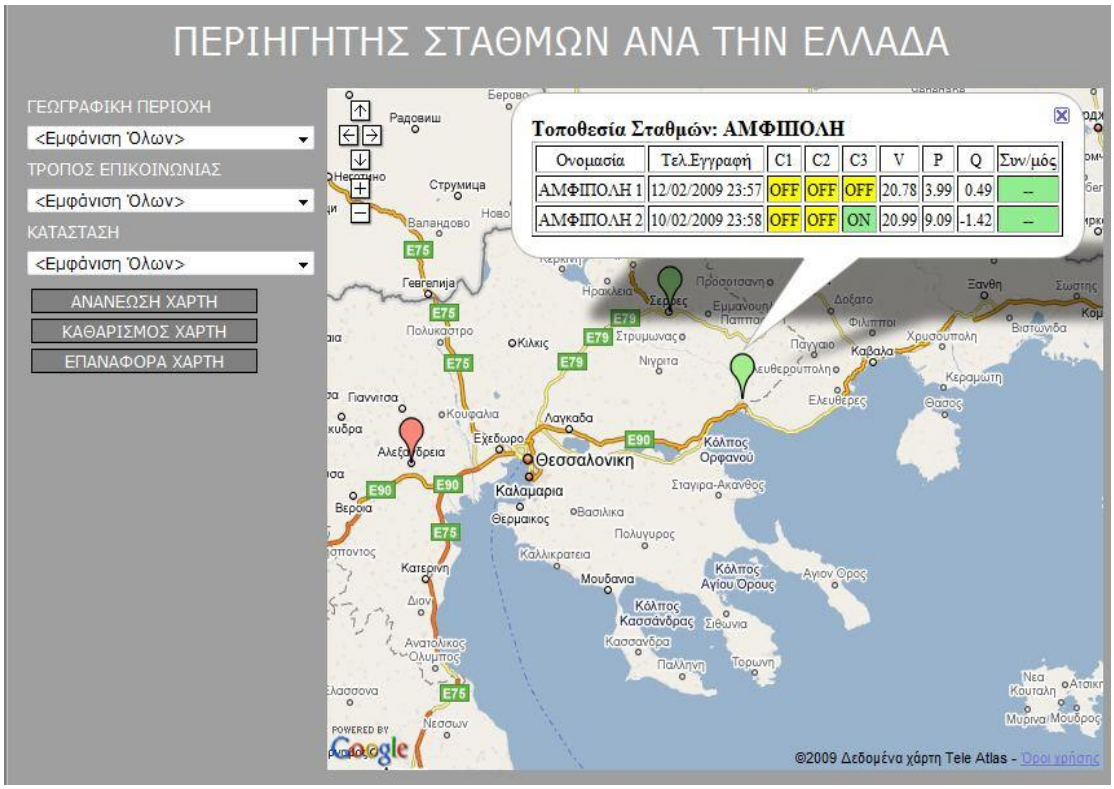
Εικόνα 4.1 : Αρχική Σελίδα



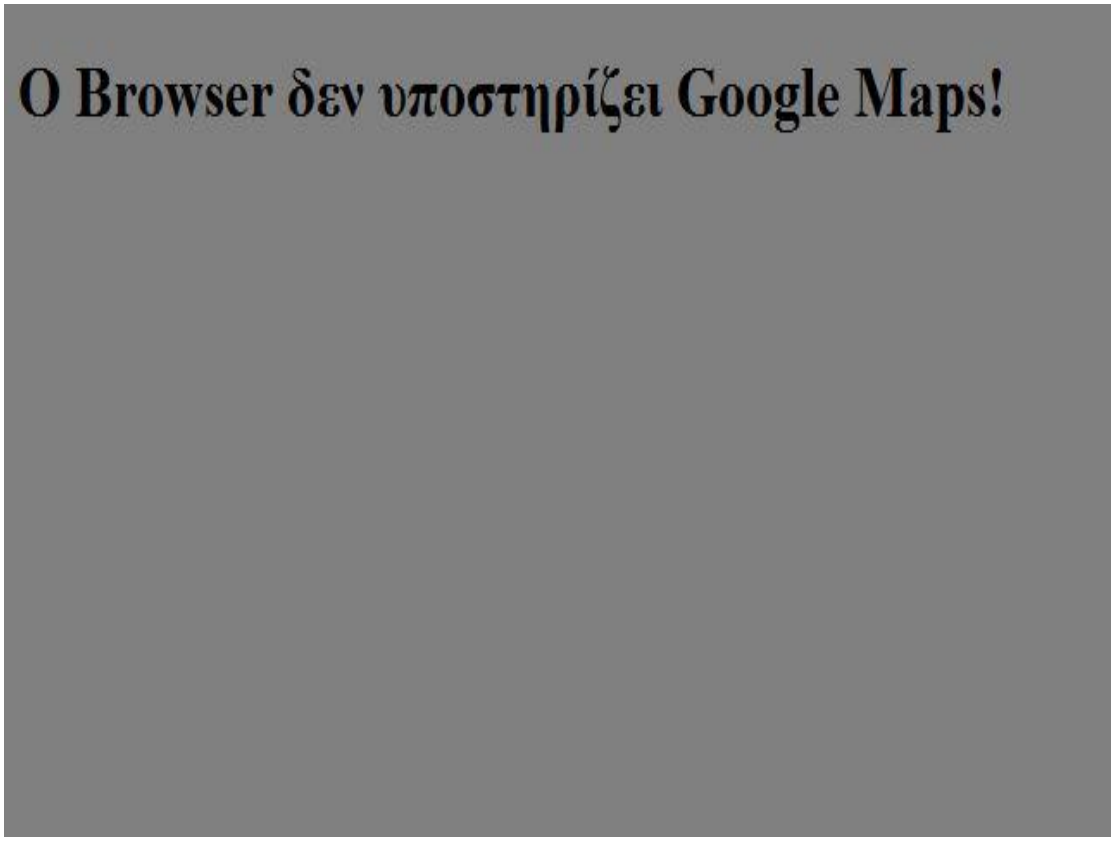
Εικόνα 4.2 : Εμφάνιση Περιοχών



Εικόνα 4.3 : Σταθμοί Περιοχής



Εικόνα 4.4 : Χάρτης με Zoom



Εικόνα 4.5 : Χωρίς Υποστήριξη

5 Πλήρης Κώδικας

Στο κεφάλαιο αυτό παρατίθεται ο πλήρης κώδικας του Περιηγητή Ηλεκτρικών Σταθμών

5.1 StationLibrary

5.1.1 CCommType.cs

```
using System;
using System.Data;
using System.Data.OleDb;

namespace StationLibrary {

public class CCommType {

private long m_Id;
private string m_Type;

private SQLManager m_SQLManager;

public CCommType() {
m_SQLManager = new SQLManager();
}

public CCommType(long id, string type) {
m_Id = id;
m_Type = type;
m_SQLManager = new SQLManager();
}

public long ID{
get {
return m_Id;
}
}

public string Type{
get {
return m_Type;
}
}


public void Load(long id) {
string sSQL;
OleDbConnection oConn = new OleDbConnection();
DataSet oDs = new DataSet();

try {
sSQL = "SELECT * FROM comm_type WHERE ID = " + Convert.ToString(id);
```

```
oDs = m_SQLManager.ExecuteSQL(sSQL, ref oConn);

if (oDs.Tables[0].Rows.Count > 0) {
m_Id = Convert.ToInt64(oDs.Tables[0].Rows[0]["ID"]);
m_Type = Convert.ToString(oDs.Tables[0].Rows[0]["type"]);
}
} catch {
}
}

public DataSet Collect(string fields,string ids, string type) {
string sSQL;
OleDbConnection oConn = new OleDbConnection();
DataSet oDs = new DataSet();

try {
sSQL = "SELECT " + fields + " FROM comm_type";

if (ids != "")
m_SQLManager.AddCriteria(ref sSQL, "ID IN (" + ids + ")", true);
if (type != "")
m_SQLManager.AddCriteria(ref sSQL, m_SQLManager.SQLLike("type",
type), ! (sSQL.Contains("WHERE")));

oDs = m_SQLManager.ExecuteSQL(sSQL, ref oConn);
return oDs;
}
catch {
return null;
}
}

}

}
```

5.1.2 CDialerSetting.cs

```
using System;
using System.Data;
using System.Data.OleDb;


namespace StationLibrary {

public class CDialerSetting {

private long m_Id;
private string m_MasterDir;
private string m_BackupDir;
private long m_StartDialupTimeMinute;
private long m_EndDialupTimeMinute;

private SQLManager m_SQLManager;

public CDialerSetting() {
m_SQLManager = new SQLManager();
}
```

```
public CDialerSetting(long id, string masterDir, string backupDir,
long startDialupTimeMinute, long endDialupTimeMinute) {
m_Id = id;
m_MasterDir = masterDir;
m_BackupDir = backupDir;
m_StartDialupTimeMinute = startDialupTimeMinute;
m_EndDialupTimeMinute = endDialupTimeMinute;
m_SQLManager = new SQLManager();
}

public long ID {
get {
return m_Id;
}
}

public string MasterDir {
get {
return m_MasterDir;
}
}

public string BackupDir {
get {
return m_BackupDir;
}
}

public long StartDialupTimeMinute {
get {
return m_StartDialupTimeMinute;
}
}

public long EndDialupTimeMinute {
get {
return m_EndDialupTimeMinute;
}
}

public void Load(long id) {
string sSQL;
OleDbConnection oConn = new OleDbConnection();
DataSet oDs = new DataSet();

try {
sSQL = "SELECT * FROM Dialer_setting WHERE id = " +
Convert.ToString(id);
oDs = m_SQLManager.ExecuteSQL(sSQL, ref oConn);

if (oDs.Tables[0].Rows.Count > 0) {
m_Id = Convert.ToInt64(oDs.Tables[0].Rows[0]["id"]);
m_MasterDir = Convert.ToString(oDs.Tables[0].Rows[0]["master_dir"]);
m_BackupDir = Convert.ToString(oDs.Tables[0].Rows[0]["backup_dir"]);
m_StartDialupTimeMinute =
Convert.ToInt64(oDs.Tables[0].Rows[0]["start_dialup_time_minute"]);
m_EndDialupTimeMinute =
Convert.ToInt64(oDs.Tables[0].Rows[0]["end_dialup_time_minute"]);
}
} catch {
```

```
}
}

public DataSet Collect(string fields, string ids, string masterDir,
string backupDir,
long startDialupTimeMinuteFrom, long startDialupTimeMinuteTo,
long endDialupTimeMinuteFrom, long endDialupTimeMinuteTo) {
string sSQL;
OleDbConnection oConn = new OleDbConnection();
DataSet oDs = new DataSet();

try {
sSQL = "SELECT " + fields + " FROM Dialer_setting";

if (ids != "")
m_SQLManager.AddCriteria(ref sSQL, "id IN (" + ids + ")", true);
if (masterDir != "")
m_SQLManager.AddCriteria(ref sSQL, m_SQLManager.SQLLike("master_dir",
masterDir), !(sSQL.Contains("WHERE")));
if (backupDir != "")
m_SQLManager.AddCriteria(ref sSQL, m_SQLManager.SQLLike("backup_dir",
backupDir), !(sSQL.Contains("WHERE")));
if (startDialupTimeMinuteFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "start_dialup_time_minute >= " +
Convert.ToString(startDialupTimeMinuteFrom),
!(sSQL.Contains("WHERE")));

if (startDialupTimeMinuteTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "start_dialup_time_minute <= " +
Convert.ToString(startDialupTimeMinuteTo),
!(sSQL.Contains("WHERE")));
if (endDialupTimeMinuteFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "end_dialup_time_minute >= " +
Convert.ToString(endDialupTimeMinuteFrom),
!(sSQL.Contains("WHERE")));
if (endDialupTimeMinuteTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "end_dialup_time_minute <= " +
Convert.ToString(endDialupTimeMinuteTo), !(sSQL.Contains("WHERE")));

oDs = m_SQLManager.ExecuteSQL(sSQL, ref oConn);
return oDs;
} catch {
return null;
}
}

}

}
```

5.1.3 CEventCode.cs

```
using System;
using System.Data;
using System.Data.OleDb;

namespace StationLibrary {

public class CEventCode {
```

```
private long m_Id;

private string m_Name;
private long m_GroupId;
private bool m_ShowOnGrid;
private bool m_CanStore;

private SQLManager m_SQLManager;

public CEventCode() {
m_SQLManager = new SQLManager();
}

public CEventCode(long id, string name, long groupId,
bool showOnGrid, bool canStore) {
m_Id = id;
m_Name = name;
m_GroupId = groupId;
m_ShowOnGrid = showOnGrid;

m_CanStore = canStore;
m_SQLManager = new SQLManager();
}

public long ID {
get {
return m_Id;
}
}

public string Name {
get {
return m_Name;
}
}

public long GroupId {
get {
return m_GroupId;
}
}

public string GroupName {
get {
try {
CGroupCode oGroupCode = new CGroupCode();
oGroupCode.Load(m_GroupId);
return oGroupCode.Name;
} catch {
return "";
}
}
}

public bool ShowOnGrid {
get {
return m_ShowOnGrid;
}
}
```

```
public bool CanStore {
get {
return m_CanStore;
}
}

public void Load(long id) {
string sSQL;
OleDbConnection oConn = new OleDbConnection();

DataSet oDs = new DataSet();

try {
sSQL = "SELECT * FROM event_code WHERE ID = " + Convert.ToString(id);
oDs = m_SQLManager.ExecuteSQL(sSQL, ref oConn);

if (oDs.Tables[0].Rows.Count > 0) {
m_Id = Convert.ToInt64(oDs.Tables[0].Rows[0]["ID"]);
m_Name = Convert.ToString(oDs.Tables[0].Rows[0]["name"]);
m_GroupId = Convert.ToInt64(oDs.Tables[0].Rows[0]["group_id"]);
m_ShowOnGrid =
Convert.ToBoolean(oDs.Tables[0].Rows[0]["show_on_grid"]);
m_CanStore = Convert.ToBoolean(oDs.Tables[0].Rows[0]["can_store"]);
}
} catch {
}
}

public DataSet Collect(string fields, string ids, string name, long
groupId,
long showOnGrid, long canStore) {
string sSQL;
OleDbConnection oConn = new OleDbConnection();
DataSet oDs = new DataSet();

try {
sSQL = "SELECT " + fields + " FROM event_code ";

if (ids != "")
m_SQLManager.AddCriteria(ref sSQL, "id IN (" + ids + ")", true);
if (name != "")
m_SQLManager.AddCriteria(ref sSQL, m_SQLManager.SQLLike("name",
name), !(sSQL.Contains("WHERE")));
if (groupId > -1)
m_SQLManager.AddCriteria(ref sSQL, "group_id = " +
Convert.ToString(groupId), !(sSQL.Contains("WHERE")));
if (showOnGrid > -1)
m_SQLManager.AddCriteria(ref sSQL, "show_on_grid = " +
Convert.ToString(showOnGrid), !(sSQL.Contains("WHERE")));
if (canStore > -1)
m_SQLManager.AddCriteria(ref sSQL, "can_store = " +
Convert.ToString(canStore), !(sSQL.Contains("WHERE")));

oDs = m_SQLManager.ExecuteSQL(sSQL, ref oConn);
return oDs;
} catch {
return null;
}
}
```

```
}

}
```

5.1.4 CEventData.cs

```
using System;
using System.Data;
using System.Data.OleDb;

namespace StationLibrary {

public class CEventData {

private long m_StationId;
private DateTime m_Date;
private long m_EventCodeId;

private SQLManager m_SQLManager;

public CEventData() {
m_SQLManager = new SQLManager();
}

public CEventData(long stationId, DateTime date, long eventCodeId) {
m_StationId = stationId;
m_Date = date;
m_EventCodeId = eventCodeId;
m_SQLManager = new SQLManager();
}

public long StationID {
get {
return m_StationId;
}
}

public string StationName {
get {
try {
CStation oStation = new CStation();
oStation.Load(m_StationId);
return oStation.Name;
} catch {
return "";
}
}
}

public DateTime Date {
get {

return m_Date;
}
}

public long EventCodeID {
get {
return m_EventCodeId;
```

```
}
}

public string EventCodeName {
get {
try {
CEventCode oEventCode = new CEventCode();
oEventCode.Load(m_EventCodeId);
return oEventCode.Name;
} catch {
return "";
}
}
}

public void Load(long stationId, DateTime date, long eventCodeId) {
string sSQL;
OleDbConnection oConn = new OleDbConnection();
DataSet oDs = new DataSet();

try {
sSQL = "SELECT * FROM event_data WHERE " +
"station_id = " + Convert.ToString(stationId) + " AND " +
"date = CONVERT(DATETIME , '" + String.Format("{0:yyyy-MM-dd
HH:mm:ss}",date) + "' , 120)" +
" AND event_code_id = " + Convert.ToString(eventCodeId);
oDs = m_SQLManager.ExecuteSQL(sSQL, ref oConn);

if (oDs.Tables[0].Rows.Count > 0) {
m_StationId = Convert.ToInt64(oDs.Tables[0].Rows[0]["station_id"]);
m_Date = Convert.ToDateTime(oDs.Tables[0].Rows[0]["date"]);
m_EventCodeId =
Convert.ToInt64(oDs.Tables[0].Rows[0]["event_code_id"]);
}
} catch {
}
}

public DataSet Collect(string fields, long stationId, DateTime
dateFrom,
DateTime dateTo, long eventCodeId) {
string sSQL;
OleDbConnection oConn = new OleDbConnection();
DataSet oDs = new DataSet();

try {
sSQL = "SELECT " + fields + " FROM event_data ";

if (stationId > -1)
m_SQLManager.AddCriteria(ref sSQL, "station_id = " +
Convert.ToString(stationId), !(sSQL.Contains("WHERE")));
if (dateFrom != null)
m_SQLManager.AddCriteria(ref sSQL,
m_SQLManager.SQLDate(">=","date",dateFrom),
!(sSQL.Contains("WHERE")));
if (dateTo != null)
m_SQLManager.AddCriteria(ref sSQL,
m_SQLManager.SQLDate("<=","date",dateTo), !(sSQL.Contains("WHERE")));
if (eventCodeId > -1)
```

```
m_SQLManager.AddCriteria(ref sSQL, "event_code_id = " +
Convert.ToString(eventCodeId), !(sSQL.Contains("WHERE")));

oDs = m_SQLManager.ExecuteSQL(sSQL, ref oConn);
return oDs;
} catch {
return null;
}
}

}

}
```

5.1.5 CGroupCode.cs

```
using System;
using System.Data;
using System.Data.OleDb;

namespace StationLibrary {

public class CGroupCode {

private long m_Id;
private string m_Name;

private SQLManager m_SQLManager;

public CGroupCode() {
m_SQLManager = new SQLManager();
}

public CGroupCode(long id, string name) {
m_Id = id;

m_Name = name;
m_SQLManager = new SQLManager();
}

public long ID {
get {
return m_Id;
}
}

public string Name {
get {
return m_Name;
}
}

public void Load(long id) {
string sSQL;
OleDbConnection oConn = new OleDbConnection();
DataSet oDs = new DataSet();

try {
```

```
sSQL = "SELECT * FROM group_code WHERE ID = " + Convert.ToString(id);
oDs = m_SQLManager.ExecuteSQL(sSQL, ref oConn);

if (oDs.Tables[0].Rows.Count > 0) {
m_Id = Convert.ToInt64(oDs.Tables[0].Rows[0]["ID"]);
m_Name = Convert.ToString(oDs.Tables[0].Rows[0]["name"]);
}

} catch {
}
}

public DataSet Collect(string fields, string ids, string name) {
string sSQL;
OleDbConnection oConn = new OleDbConnection();
DataSet oDs = new DataSet();

try {
sSQL = "SELECT " + fields + " FROM group_code";

if (ids != "")
m_SQLManager.AddCriteria(ref sSQL, "ID IN (" + ids + ")", true);
if (name != "")
m_SQLManager.AddCriteria(ref sSQL, m_SQLManager.SQLLike("name",
name), !(sSQL.Contains("WHERE")));

oDs = m_SQLManager.ExecuteSQL(sSQL, ref oConn);
return oDs;

} catch {
return null;
}
}

}

}
```

5.1.6 CLogData.cs

```
using System;
using System.Data;
using System.Data.OleDb;

namespace StationLibrary {

public class CLogData {

private long m_StationId;
private DateTime m_Date;
private double m_V;
private double m_Q;
private double m_P;
private double m_I1;
private double m_I2;
private double m_I3;
private bool m_C1Status;
private bool m_C2Status;
private bool m_C3Status;
```

```
private bool m_C1Enable;
private bool m_C2Enable;
private bool m_C3Enable;
private bool m_TXEnable;

private bool m_Alarm1;
private bool m_Alarm2;
private bool m_Alarm3;
private double m_Px;
private double m_PercentValue;
private double m_CosineF;
private double m_CosineFWithoutCap;
private double m_CapMva;

private SQLManager m_SQLManager;

public CLogData() {
m_SQLManager = new SQLManager();
}


public CLogData(long stationId, DateTime date, double v, double q,
double p, double i1, double i2, double i3, bool c1Status,
bool c2Status, bool c3Status, bool c1Enable, bool c2Enable,
bool c3Enable, bool tXEnable, bool alarm1, bool alarm2,
bool alarm3, double px, double percentValue, double cosineF,
double cosineFWithoutCap, double capMva) {
m_StationId = stationId;
m_Date = date;
m_V = v;
m_Q = q;
m_P = p;
m_I1 = i1;
m_I2 = i2;
m_I3 = i3;
m_C1Status = c1Status;
m_C2Status = c2Status;
m_C3Status = c3Status;
m_C1Enable = c1Enable;
m_C2Enable = c2Enable;
m_C3Enable = c3Enable;
m_TXEnable = tXEnable;
m_Alarm1 = alarm1;
m_Alarm2 = alarm2;
m_Alarm3 = alarm3;
m_Px = px;
m_PercentValue = percentValue;
m_CosineF = cosineF;
m_CosineFWithoutCap = cosineFWithoutCap;
m_CapMva = capMva;
m_SQLManager = new SQLManager();
}

public long StationId {
get {
return m_StationId;
}
}

public DateTime Date {
```

```
get {
return m_Date;
}
}

public double V {
get {
return m_V;
}
}

public double Q {
get {
return m_Q;
}
}

public double P {
get {
return m_P;
}
}

public double I1 {
get {
return m_I1;
}
}

public double I2 {
get {
return m_I2;
}
}

public double I3 {
get {
return m_I3;
}
}

public bool C1Status {
get {
return m_C1Status;
}
}

public bool C2Status {
get {
return m_C2Status;
}
}

public bool C3Status {
get {
return m_C3Status;
}
}

public bool C1Enable {
```

```
get {
return m_C1Enable;
}
}

public bool C2Enable {
get {
return m_C2Enable;
}
}

public bool C3Enable {
get {
return m_C3Enable;
}
}

public bool TXEnable {
get {
return m_TXEnable;
}
}

public bool Alarm1 {
get {
return m_Alarm1;
}
}

public bool Alarm2 {
get {
return m_Alarm2;
}
}

public bool Alarm3 {
get {
return m_Alarm3;
}
}

public double Px {
get {
return m_Px;
}
}

public double PercentValue {
get {

return m_PercentValue;
}
}

public double CosineF {
get {
return m_CosineF;
}
}
```

```
public double CosineFWithoutCap {
get {
return m_CosineFWithoutCap;
}
}

public double CapMva {
get {
return m_CapMva;
}
}

public void Load(long stationId, DateTime date) {
string sSQL;
OleDbConnection oConn = new OleDbConnection();
DataSet oDs = new DataSet();

try {
sSQL = "SELECT * FROM log_data WHERE station_id = " +
Convert.ToString(stationId) +
" AND date = CONVERT(DATETIME , '" + String.Format("{0:yyyy-MM-dd
HH:mm:ss}", date) + "' , 120)";
oDs = m_SQLManager.ExecuteSQL(sSQL, ref oConn);

if (oDs.Tables[0].Rows.Count > 0) {
m_StationId = Convert.ToInt64(oDs.Tables[0].Rows[0]["station_id"]);
m_Date = Convert.ToDateTime(oDs.Tables[0].Rows[0]["date"]);
m_V = Convert.ToDouble(oDs.Tables[0].Rows[0]["V"]);
m_Q = Convert.ToDouble(oDs.Tables[0].Rows[0]["Q"]);
m_P = Convert.ToDouble(oDs.Tables[0].Rows[0]["P"]);
m_I1 = Convert.ToDouble(oDs.Tables[0].Rows[0]["I1"]);
m_I2 = Convert.ToDouble(oDs.Tables[0].Rows[0]["I2"]);
m_I3 = Convert.ToDouble(oDs.Tables[0].Rows[0]["I3"]);
m_C1Status = Convert.ToBoolean(oDs.Tables[0].Rows[0]["C1_status"]);
m_C2Status = Convert.ToBoolean(oDs.Tables[0].Rows[0]["C2_status"]);
m_C3Status = Convert.ToBoolean(oDs.Tables[0].Rows[0]["C3_status"]);
m_C1Enable = Convert.ToBoolean(oDs.Tables[0].Rows[0]["C1_enable"]);
m_C2Enable = Convert.ToBoolean(oDs.Tables[0].Rows[0]["C2_enable"]);
m_C3Enable = Convert.ToBoolean(oDs.Tables[0].Rows[0]["C3_enable"]);

m_TXEnable = Convert.ToBoolean(oDs.Tables[0].Rows[0]["TX_enable"]);
m_Alarm1 = Convert.ToBoolean(oDs.Tables[0].Rows[0]["alarm1"]);
m_Alarm2 = Convert.ToBoolean(oDs.Tables[0].Rows[0]["alarm2"]);
m_Alarm3 = Convert.ToBoolean(oDs.Tables[0].Rows[0]["alarm3"]);
m_Px = Convert.ToDouble(oDs.Tables[0].Rows[0]["px"]);
m_PercentValue =
Convert.ToDouble(oDs.Tables[0].Rows[0]["percent_value"]);
m_CosineF = Convert.ToDouble(oDs.Tables[0].Rows[0]["cosine_f"]);
m_CosineFWithoutCap =
Convert.ToDouble(oDs.Tables[0].Rows[0]["cosine_f_without_cap"]);
m_CapMva = Convert.ToDouble(oDs.Tables[0].Rows[0]["cap_mva"]);
}
} catch {
}
}

public DataSet Collect(string fields, long stationId, DateTime
dateFrom, DateTime dateTo,
long vFrom , long vTo , long pFrom, long pTo, long qFrom, long qTo,
```

```
long i1From, long i1To, long i2From, long i2To, long i3From, long
i3To,
long c1Status, long c2Status, long c3Status, long c1Enable, long
c2Enable,
long c3Enable, long tXEnable, long alarm1, long alarm2, long alarm3,
long pxFrom, long pxTo, long percentValueFrom, long percentValueTo,
long cosineFFrom, long cosineFTo, long cosineFWithoutCapFrom,
long cosineFWithoutCapTo, long capMvaFrom, long capMvaTo) {
string sSQL;
OleDbConnection oConn = new OleDbConnection();
DataSet oDs = new DataSet();

try {
sSQL = "SELECT " + fields + " FROM log_data ";

if (stationId > -1)
m_SQLManager.AddCriteria(ref sSQL, "station_id = " +
Convert.ToString(stationId), !(sSQL.Contains("WHERE")));
if (dateFrom != null)
m_SQLManager.AddCriteria(ref sSQL,
m_SQLManager.SQLDate(">=","date",dateFrom),
!(sSQL.Contains("WHERE")));
if (dateTo != null)
m_SQLManager.AddCriteria(ref sSQL,
m_SQLManager.SQLDate("<=","date",dateTo), !(sSQL.Contains("WHERE")));
if (vFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "V >= " + Convert.ToString(vFrom),
!(sSQL.Contains("WHERE")));
if (vTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "V <= " + Convert.ToString(vTo),
!(sSQL.Contains("WHERE")));
if (pFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "P >= " + Convert.ToString(pFrom),
!(sSQL.Contains("WHERE")));

if (pTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "P <= " + Convert.ToString(pTo),
!(sSQL.Contains("WHERE")));
if (qFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "Q >= " + Convert.ToString(qFrom),
!(sSQL.Contains("WHERE")));
if (qTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "Q <= " + Convert.ToString(qTo),
!(sSQL.Contains("WHERE")));
if (i1From > 0)
m_SQLManager.AddCriteria(ref sSQL, "I1 >= " +
Convert.ToString(i1From), !(sSQL.Contains("WHERE")));
if (i1To > 0)
m_SQLManager.AddCriteria(ref sSQL, "I1 <= " + Convert.ToString(i1To),
!(sSQL.Contains("WHERE")));
if (i2From > 0)
m_SQLManager.AddCriteria(ref sSQL, "I2 >= " +
Convert.ToString(i2From), !(sSQL.Contains("WHERE")));
if (i2To > 0)
m_SQLManager.AddCriteria(ref sSQL, "I2 <= " + Convert.ToString(i2To),
!(sSQL.Contains("WHERE")));
if (i3From > 0)
m_SQLManager.AddCriteria(ref sSQL, "I3 >= " +
Convert.ToString(i3From), !(sSQL.Contains("WHERE")));
if (i3To > 0)
```

```
m_SQLManager.AddCriteria(ref sSQL, "I3 <= " + Convert.ToString(i3To),
!(sSQL.Contains("WHERE")));
if (c1Status > -1)
m_SQLManager.AddCriteria(ref sSQL, "C1_status = " +
Convert.ToString(c1Status), !(sSQL.Contains("WHERE")));
if (c2Status > -1)
m_SQLManager.AddCriteria(ref sSQL, "C2_status = " +
Convert.ToString(c2Status), !(sSQL.Contains("WHERE")));
if (c3Status > -1)
m_SQLManager.AddCriteria(ref sSQL, "C3_status = " +
Convert.ToString(c3Status), !(sSQL.Contains("WHERE")));
if (c1Enable > -1)
m_SQLManager.AddCriteria(ref sSQL, "C1_enable = " +
Convert.ToString(c1Enable), !(sSQL.Contains("WHERE")));
if (c2Enable > -1)
m_SQLManager.AddCriteria(ref sSQL, "C2_enable = " +
Convert.ToString(c2Enable), !(sSQL.Contains("WHERE")));
if (c3Enable > -1)
m_SQLManager.AddCriteria(ref sSQL, "C3_enable = " +
Convert.ToString(c3Enable), !(sSQL.Contains("WHERE")));
if (tXEnable > -1)
m_SQLManager.AddCriteria(ref sSQL, "TX_enable = " +
Convert.ToString(tXEnable), !(sSQL.Contains("WHERE")));
if (alarm1 > -1)

m_SQLManager.AddCriteria(ref sSQL, "alarm1 = " +
Convert.ToString(alarm1), !(sSQL.Contains("WHERE")));
if (alarm2 > -1)
m_SQLManager.AddCriteria(ref sSQL, "alarm2 = " +
Convert.ToString(alarm2), !(sSQL.Contains("WHERE")));
if (alarm3 > -1)
m_SQLManager.AddCriteria(ref sSQL, "alarm3 = " +
Convert.ToString(alarm3), !(sSQL.Contains("WHERE")));
if (pxFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "px >= " +
Convert.ToString(pxFrom), !(sSQL.Contains("WHERE")));
if (pxTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "px <= " + Convert.ToString(pxTo),
!(sSQL.Contains("WHERE")));
if (percentValueFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "percent_value >= " +
Convert.ToString(percentValueFrom), !(sSQL.Contains("WHERE")));
if (percentValueTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "percent_value <= " +
Convert.ToString(percentValueTo), !(sSQL.Contains("WHERE")));
if (cosineFFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "cosine_f >= " +
Convert.ToString(cosineFFrom), !(sSQL.Contains("WHERE")));
if (cosineFTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "cosine_f <= " +
Convert.ToString(cosineFTo), !(sSQL.Contains("WHERE")));
if (cosineFWithoutCapFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "cosine_f_without_cap >= " +
Convert.ToString(cosineFWithoutCapFrom), !(sSQL.Contains("WHERE")));
if (cosineFWithoutCapTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "cosine_f_without_cap <= " +
Convert.ToString(cosineFWithoutCapTo), !(sSQL.Contains("WHERE")));
if (capMvaFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "cap_mva >= " +
Convert.ToString(capMvaFrom), !(sSQL.Contains("WHERE")));
```

```
if (capMvaTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "cap_mva <= " +
Convert.ToString(capMvaTo), !(sSQL.Contains("WHERE")));


oDs = m_SQLManager.ExecuteSQL(sSQL, ref oConn);
return oDs;
} catch {
return null;
}
}

}

}
```

5.1.7 COperationMode.cs

```
using System;
using System.Data;
using System.Data.OleDb;

namespace StationLibrary {

public class COperationMode {

private long m_Id;
private string m_Name;

private SQLManager m_SQLManager;

public COperationMode() {
m_SQLManager = new SQLManager();
}

public COperationMode(long id, string name) {
m_Id = id;
m_Name = name;
m_SQLManager = new SQLManager();
}

public long ID {
get {
return m_Id;
}
}

public string Name {
get {
return m_Name;
}
}

public void Load(long id) {
string sSQL;
OleDbConnection oConn = new OleDbConnection();
DataSet oDs = new DataSet();
```

```
try {
sSQL = "SELECT * FROM operation_mode WHERE ID = " +
Convert.ToString(id);
oDs = m_SQLManager.ExecuteSQL(sSQL, ref oConn);

if (oDs.Tables[0].Rows.Count > 0) {

m_Id = Convert.ToInt64(oDs.Tables[0].Rows[0]["ID"]);
m_Name = Convert.ToString(oDs.Tables[0].Rows[0]["name"]);
}
} catch {
}
}

public DataSet Collect(string fields, string ids, string name) {
string sSQL;
OleDbConnection oConn = new OleDbConnection();
DataSet oDs = new DataSet();

try {
sSQL = "SELECT " + fields + " FROM operation_mode";

if (ids != "")
m_SQLManager.AddCriteria(ref sSQL, "ID IN (" + ids + ")", true);
if (name != "")
m_SQLManager.AddCriteria(ref sSQL, m_SQLManager.SQLLike("name",
name), !(sSQL.Contains("WHERE")));

oDs = m_SQLManager.ExecuteSQL(sSQL, ref oConn);
return oDs;
} catch {
return null;
}
}

}

}
```

5.1.8 CRegionCode.cs

```
using System;
using System.Data;
using System.Data.OleDb;

namespace StationLibrary {

public class CRegionCode {

private long m_Id;
private string m_Name;

private SQLManager m_SQLManager;

public CRegionCode() {

m_SQLManager = new SQLManager();
}
```

```
public CRegionCode(long id, string name) {
m_Id = id;
m_Name = name;
m_SQLManager = new SQLManager();
}

public long ID {
get {
return m_Id;
}
}

public string Name {
get {
return m_Name;
}
}

public void Load(long id) {
string sSQL;
OleDbConnection oConn = new OleDbConnection();
DataSet oDs = new DataSet();

try {
sSQL = "SELECT * FROM region_code WHERE ID = " +
Convert.ToString(id);
oDs = m_SQLManager.ExecuteSQL(sSQL, ref oConn);

if (oDs.Tables[0].Rows.Count > 0) {
m_Id = Convert.ToInt64(oDs.Tables[0].Rows[0]["ID"]);
m_Name = Convert.ToString(oDs.Tables[0].Rows[0]["name"]);
}
} catch {
}
}

public DataSet Collect(string fields, string ids, string name) {
string sSQL;
OleDbConnection oConn = new OleDbConnection();
DataSet oDs = new DataSet();

try {
sSQL = "SELECT " + fields + " FROM region_code";

if (ids != "")
m_SQLManager.AddCriteria(ref sSQL, "ID IN (" + ids + ")", true);
if (name != "")

m_SQLManager.AddCriteria(ref sSQL, m_SQLManager.SQLLike("name",
name), !(sSQL.Contains("WHERE")));

oDs = m_SQLManager.ExecuteSQL(sSQL, ref oConn);
return oDs;
} catch {
return null;
}
}

}
```

```
    }
```

5.1.9 CStation.cs

```
using System;
using System.Data;
using System.Data.OleDb;

namespace StationLibrary {

public class CStation {

private long m_Id;
private string m_Name;
private string m_PhoneNumber;
private string m_IpAddress;
private string m_Location;
private DateTime m_CfgsLastUpdated;
private DateTime m_LogsLastUpdated;
private DateTime m_EventsLastUpdated;
private long m_CommTypeId;
private bool m_Cap1SynchError;
private bool m_Cap2SynchError;
private bool m_Cap3SynchError;
private long m_MemFreeSpace;
private long m_OpModeId;
private long m_LogPeriod;
private long m_FileChangePeriod;
private double m_Qmax;
private double m_Qmin;
private double m_Vmax;
private double m_Vmin;
private double m_VUserOffset;
private double m_VAlpha;
private double m_PUserOffset;
private double m_PAlpha;
private double m_QUserOffset;
private double m_QAlpha;

private double m_I1UserOffset;
private double m_I1Alpha;
private double m_I2UserOffset;
private double m_I2Alpha;
private double m_I3UserOffset;
private double m_I3Alpha;
private double m_VOffset;
private double m_VGain;
private double m_POffset;
private double m_PGain;
private double m_QOffset;
private double m_QGain;
private double m_I1Offset;
private double m_I1Gain;
private double m_I2Offset;
private double m_I2Gain;
private double m_I3Offset;
private double m_I3Gain;
private long m_RegionId;
private bool m_CommunationEnable;
```

```
private long m_CommunationInterval;
private long m_CommunationTry;

private long m_StationAreaId;

private SQLManager m_SQLManager;

public CStation() {
m_SQLManager = new SQLManager();
}

public CStation(long id ,string name, string phoneNumber, string
ipAddress, string location,
DateTime cfgsLastUpdated, DateTime logsLastUpdated, DateTime
eventsLastUpdated,
long commTypeId, bool cap1SynchError, bool cap2SynchError, bool
cap3SynchError,
long memFreeSpace, long opModeId, long logPeriod, long
fileChangePeriod, double qmax,
double qmin, double vmax, double vmin, double vUserOffset, double
vAlpha,
double pUserOffset, double pAlpha, double qUserOffset, double qAlpha
,double i1UserOffset,
double i1Alpha, double i2UserOffset, double i2Alpha, double
i3UserOffset, double i3Alpha,
double vOffset, double vGain, double pOffset, double pGain, double
qOffset, double qGain,
double i1Offset, double i1Gain, double i2Offset, double i2Gain,
double i3Offset, double i3Gain,
long regionId, bool communationEnable, long communationInterval, long
communationTry,

long stationAreaId) {

m_Id = id;
m_Name = name;
m_PhoneNumber = phoneNumber;
m_IpAddress = ipAddress;
m_Location = location;
m_CfgsLastUpdated = cfgsLastUpdated;
m_LogsLastUpdated = logsLastUpdated;
m_EventsLastUpdated = eventsLastUpdated;
m_CommTypeId = commTypeId;
m_Cap1SynchError = cap1SynchError;
m_Cap2SynchError = cap2SynchError;
m_Cap3SynchError = cap3SynchError;
m_MemFreeSpace = memFreeSpace;
m_OpModeId = opModeId;
m_LogPeriod = logPeriod;
m_FileChangePeriod = fileChangePeriod;
m_Qmax = qmax;
m_Qmin = qmin;
m_Vmax = vmax;
m_Vmin = vmin;
m_VUserOffset = vUserOffset;
m_VAlpha = vAlpha;
m_PUserOffset = pUserOffset;
m_PAlpha = pAlpha;
m_QUserOffset = qUserOffset;
m_QAlpha = qAlpha;
m_I1UserOffset = i1UserOffset;
```

```
m_I1Alpha = i1Alpha;
m_I2UserOffset = i2UserOffset;

m_I2Alpha = i2Alpha;
m_I3UserOffset = i3UserOffset;
m_I3Alpha = i3Alpha;
m_VOffset = vOffset;
m_VGain = vGain;
m_POffset = pOffset;
m_PGain = pGain;
m_QOffset = qOffset;
m_QGain = qGain;
m_I1Offset = i1Offset;
m_I1Gain = i1Gain;
m_I2Offset = i2Offset;
m_I2Gain = i2Gain;
m_I3Offset = i3Offset;
m_I3Gain = i3Gain;
m_RegionId = regionId;
m_CommunationEnable = communationEnable;
m_CommunationInterval = communationInterval;

m_CommunationTry = communationTry;
m_StationAreaId = stationAreaId;

m_SQLManager = new SQLManager();
}


public long ID {
get {
return m_Id;
}
}

public string Name {
get {
return m_Name;
}
}

public string PhoneNumber {
get {
return m_PhoneNumber;
}
}

public string IpAddress {
get {
return m_IpAddress;
}
}

public string Location {
get {
return m_Location;
}
}

public DateTime CfgsLastUpdated {
```

```
get {

return m_CfgsLastUpdated;
}

}

public DateTime LogsLastUpdated {
get {
return m_LogsLastUpdated;
}
}


public DateTime EventsLastUpdated {
get {
return m_EventsLastUpdated;
}
}

public long CommTypeId {
get {
return m_CommTypeId;
}
}

public string CommType {
get {
try {
CCommType oCommType = new CCommType();
oCommType.Load(m_CommTypeId);
return oCommType.Type;
} catch {
return "";
}
}
}

public bool Cap1SynchError {
get {
return m_Cap1SynchError;
}
}

public bool Cap2SynchError {
get {
return m_Cap2SynchError;
}
}

public bool Cap3SynchError {
get {
return m_Cap3SynchError;
}
}

public long MemFreeSpace {
get {
return m_MemFreeSpace;
}
}
```

```
public long OperationModeId {

get {

return m_OpModeId;
}
}

public string OperationModeName {
get {
try {
COperationMode oOperationMode = new COperationMode();
oOperationMode.Load(m_OpModeId);
return oOperationMode.Name;
} catch {
return "";
}
}
}

public long LogPeriod {
get {
return m_LogPeriod;
}
}

public long FileChangePeriod {
get {
return m_FileChangePeriod;
}
}

public double Qmax {
get {
return m_Qmax;
}
}

public double Qmin {
get {
return m_Qmin;
}
}

public double Vmax {
get {
return m_Vmax;
}
}

public double Vmin {
get {

return m_Vmin;
}
}

public double VUserOffset {
get {
return m_VUserOffset;
```

```
}
}

public double VAlpha {

get {
return m_VAlpha;
}
}

public double PUserOffset {
get {
return m_PUserOffset;
}
}

public double PAlpha {
get {
return m_PAlpha;
}
}

public double QUserOffset {
get {
return m_QUserOffset;
}
}

public double QAlpha {
get {
return m_QAlpha;
}
}

public double I1UserOffset {
get {
return m_I1UserOffset;
}
}

public double I1Alpha {
get {
return m_I1Alpha;

}
}

public double I2UserOffset {
get {
return m_I2UserOffset;
}
}

public double I2Alpha {
get {
return m_I2Alpha;
}
}

public double I3UserOffset {
get {
```

```
return m_I3UserOffset;
}

}


public double I3Alpha {
get {
return m_I3Alpha;
}
}

public double VOffset {
get {
return m_VOffset;
}
}

public double VGain {
get {
return m_VGain;
}
}

public double POffset {
get {
return m_POffset;
}
}

public double PGain {
get {
return m_PGain;
}

}

public double QOffset {
get {
return m_QOffset;
}
}

public double QGain {
get {
return m_QGain;
}
}

public double I1Offset {
get {
return m_I1Offset;
}
}

public double I1Gain {
get {
return m_I1Gain;
}
}
```

```
public double I2Offset {
get {
return m_I2Offset;

}
}


public double I2Gain {
get {
return m_I2Gain;
}
}

public double I3Offset {
get {
return m_I3Offset;
}
}

public double I3Gain {
get {
return m_I3Gain;
}
}

public long RegionId {
get {
return m_RegionId;
}
}

public string RegionName {
get {
try {
CRegionCode oRegionCode = new CRegionCode();
oRegionCode.Load(m_RegionId);
return oRegionCode.Name;
} catch {
return "";
}
}
}

public bool CommunationEnable {
get {
return m_CommunationEnable;
}
}

public long CommunationInterval {
get {
return m_CommunationInterval;
}
}

public long CommunationTry {
get {
return m_CommunationTry;
}
}
```

```
public long StationAreaId {
get {
return m_StationAreaId;
}
}

public string StationAreaName {
get {
try {
CStationArea oStationArea = new CStationArea();
oStationArea.Load(m_StationAreaId);
return oStationArea.Name;
} catch {

return "";
}
}
}

public void Load(long id) {
string sSQL;
OleDbConnection oConn = new OleDbConnection();
DataSet oDs = new DataSet();

try {
sSQL = "SELECT * FROM station WHERE ID = " + Convert.ToString(id);
oDs = m_SQLManager.ExecuteSQL(sSQL, ref oConn);

if (oDs.Tables[0].Rows.Count > 0) {
m_Id = Convert.ToInt64(oDs.Tables[0].Rows[0]["ID"]);
m_Name = Convert.ToString(oDs.Tables[0].Rows[0]["name"]);
m_PhoneNumber =
Convert.ToString(oDs.Tables[0].Rows[0]["phone_number"]);
m_IpAddress = Convert.ToString(oDs.Tables[0].Rows[0]["ip_address"]);
m_Location = Convert.ToString(oDs.Tables[0].Rows[0]["location"]);
m_CfgsLastUpdated =
Convert.ToDateTime(oDs.Tables[0].Rows[0]["cfgs_last_updated"]);
m_LogsLastUpdated =
Convert.ToDateTime(oDs.Tables[0].Rows[0]["logs_last_updated"]);
m_EventsLastUpdated =
Convert.ToDateTime(oDs.Tables[0].Rows[0]["events_last_updated"]);
m_CommTypeId =
Convert.ToInt64(oDs.Tables[0].Rows[0]["comm_type_id"]);
m_Cap1SynchError =
Convert.ToBoolean(oDs.Tables[0].Rows[0]["cap1_synch_error"]);
m_Cap2SynchError =
Convert.ToBoolean(oDs.Tables[0].Rows[0]["cap2_synch_error"]);
m_Cap3SynchError =
Convert.ToBoolean(oDs.Tables[0].Rows[0]["cap3_synch_error"]);
m_MemFreeSpace =
Convert.ToInt64(oDs.Tables[0].Rows[0]["memm_free_space"]);
m_OpModeId = Convert.ToInt64(oDs.Tables[0].Rows[0]["op_mode_id"]);
m_LogPeriod = Convert.ToInt64(oDs.Tables[0].Rows[0]["log_period"]);
m_FileChangePeriod =
Convert.ToInt64(oDs.Tables[0].Rows[0]["file_change"]);
m_Qmax = Convert.ToDouble(oDs.Tables[0].Rows[0]["QMax"]);
m_Qmin = Convert.ToDouble(oDs.Tables[0].Rows[0]["QMin"]);
m_Vmax = Convert.ToDouble(oDs.Tables[0].Rows[0]["VMax"]);
m_Vmin = Convert.ToDouble(oDs.Tables[0].Rows[0]["VMin"]);
```

```
m_VUserOffset =
Convert.ToDouble(oDs.Tables[0].Rows[0]["V_UserOffset"]);
m_VAlpha = Convert.ToDouble(oDs.Tables[0].Rows[0]["V_Alpha"]);
m_PUserOffset =
Convert.ToDouble(oDs.Tables[0].Rows[0]["P_UserOffset"]);

m_PAlpha = Convert.ToDouble(oDs.Tables[0].Rows[0]["P_Alpha"]);


m_QUserOffset =
Convert.ToDouble(oDs.Tables[0].Rows[0]["Q_UserOffset"]);
m_QAlpha = Convert.ToDouble(oDs.Tables[0].Rows[0]["P_Alpha"]);
m_I1UserOffset =
Convert.ToDouble(oDs.Tables[0].Rows[0]["I1_UserOffset"]);

m_I1Alpha = Convert.ToDouble(oDs.Tables[0].Rows[0]["I1_Alpha"]);
m_I2UserOffset =
Convert.ToDouble(oDs.Tables[0].Rows[0]["I2_UserOffset"]);
m_I2Alpha = Convert.ToDouble(oDs.Tables[0].Rows[0]["I2_Alpha"]);
m_I3UserOffset =
Convert.ToDouble(oDs.Tables[0].Rows[0]["I3_UserOffset"]);
m_I3Alpha = Convert.ToDouble(oDs.Tables[0].Rows[0]["I3_Alpha"]);
m_VOffset = Convert.ToDouble(oDs.Tables[0].Rows[0]["V_Offset"]);
m_VGain = Convert.ToDouble(oDs.Tables[0].Rows[0]["V_Gain"]);
m_POffset = Convert.ToDouble(oDs.Tables[0].Rows[0]["P_Offset"]);
m_PGain = Convert.ToDouble(oDs.Tables[0].Rows[0]["P_Gain"]);
m_QOffset = Convert.ToDouble(oDs.Tables[0].Rows[0]["Q_Offset"]);
m_QGain = Convert.ToDouble(oDs.Tables[0].Rows[0]["Q_Gain"]);
m_I1Offset = Convert.ToDouble(oDs.Tables[0].Rows[0]["I1_Offset"]);
m_I1Gain = Convert.ToDouble(oDs.Tables[0].Rows[0]["I1_Gain"]);
m_I2Offset = Convert.ToDouble(oDs.Tables[0].Rows[0]["I2_Offset"]);
m_I2Gain = Convert.ToDouble(oDs.Tables[0].Rows[0]["I2_Gain"]);
m_I3Offset = Convert.ToDouble(oDs.Tables[0].Rows[0]["I3_Offset"]);
m_I3Gain = Convert.ToDouble(oDs.Tables[0].Rows[0]["I3_Gain"]);
m_RegionId = Convert.ToInt64(oDs.Tables[0].Rows[0]["region_id"]);
m_CommunationEnable =
Convert.ToBoolean(oDs.Tables[0].Rows[0]["communation_enable"]);
m_CommunationInterval =
Convert.ToInt64(oDs.Tables[0].Rows[0]["communation_interval"]);
m_CommunationTry =
Convert.ToInt64(oDs.Tables[0].Rows[0]["communation_try"]);
m_StationAreaId =
Convert.ToInt64(oDs.Tables[0].Rows[0]["station_area_id"]);
}
}
catch {
}
}

public DataSet Collect(string fields, string ids, string name, string
phoneNumber, string ipAddress,
string location, DateTime cfgsLastUpdatedFrom, DateTime
cfgsLastUpdatedTo,
DateTime logsLastUpdatedFrom, DateTime logsLastUpdatedTo, DateTime
eventsLastUpdatedFrom,
DateTime eventsLastUpdatedTo, long commTypeId, long cap1SynchError,
long cap2SynchError,
long cap3SynchError, long memFreeSpaceFrom, long memFreeSpaceTo, long
opModeId,
long logPeriodFrom, long logPeriodTo, long fileChangePeriodFrom, long
fileChangePeriodTo,
```

```
double qmaxFrom, double qmaxTo, double qminFrom, double qminTo,
double vmaxFrom,
double vmaxTo, double vminFrom, double vminTo, double
vUserOffsetFrom, double vUserOffsetTo,
double vAlphaFrom, double vAlphaTo, double pUserOffsetFrom, double
pUserOffsetTo, double pAlphaFrom,

double pAlphaTo, double qUserOffsetFrom, double qUserOffsetTo ,double
qAlphaFrom, double qAlphaTo,
double i1UserOffsetFrom, double i1UserOffsetTo, double i1AlphaFrom,
double i1AlphaTo,
double i2UserOffsetFrom, double i2UserOffsetTo, double i2AlphaFrom,
double i2AlphaTo,
double i3UserOffsetFrom, double i3UserOffsetTo, double i3AlphaFrom,
double i3AlphaTo,
double vOffsetFrom, double vOffsetTo, double vGainFrom, double
vGainTo,
double pOffsetFrom, double pOffsetTo, double pGainFrom, double
pGainTo,
double qOffsetFrom, double qOffsetTo, double qGainFrom, double
qGainTo,
double i1OffsetFrom, double i1OffsetTo, double i1GainFrom, double
i1GainTo,
double i2OffsetFrom, double i2OffsetTo, double i2GainFrom, double
i2GainTo,
double i3OffsetFrom, double i3OffsetTo, double i3GainFrom, double
i3GainTo,
long regionId, long communationEnable, long communationIntervalFrom,
long communationIntervalTo,
long communationTryFrom, long communationTryTo, long stationAreaId) {
string sSQL;
OleDbConnection oConn = new OleDbConnection();
DataSet oDs = new DataSet();

try
{
sSQL = "SELECT " + fields + " FROM station ";

if (ids != "")
m_SQLManager.AddCriteria(ref sSQL, "ID IN (" + ids + ")", true);
if (name != "")
m_SQLManager.AddCriteria(ref sSQL, m_SQLManager.SQLLike("name",
name), !(sSQL.Contains("WHERE")));
if (phoneNumber != "")
m_SQLManager.AddCriteria(ref sSQL,
m_SQLManager.SQLLike("phone_number", phoneNumber),
!(sSQL.Contains("WHERE")));
if (ipAddress != "")
m_SQLManager.AddCriteria(ref sSQL, m_SQLManager.SQLLike("ip_address",
ipAddress), !(sSQL.Contains("WHERE")));
if (location != "")
m_SQLManager.AddCriteria(ref sSQL, m_SQLManager.SQLLike("location",
location), !(sSQL.Contains("WHERE")));
if (cfgsLastUpdatedFrom != null)
m_SQLManager.AddCriteria(ref sSQL,
m_SQLManager.SQLDate(">=","cfgs_last_updated",cfgsLastUpdatedFrom),
!(sSQL.Contains("WHERE")));
if (cfgsLastUpdatedTo != null)
```

```
m_SQLManager.AddCriteria(ref sSQL,
m_SQLManager.SQLDate("<=","cfgs_last_updated",cfgsLastUpdatedTo),
!(sSQL.Contains("WHERE")));
if (logsLastUpdatedFrom != null)

m_SQLManager.AddCriteria(ref sSQL,
m_SQLManager.SQLDate(">=","logs_last_updated",logsLastUpdatedFrom),
!(sSQL.Contains("WHERE")));

if (logsLastUpdatedTo != null)
m_SQLManager.AddCriteria(ref sSQL,
m_SQLManager.SQLDate("<=","logs_last_updated",logsLastUpdatedTo),
!(sSQL.Contains("WHERE")));
if (eventsLastUpdatedFrom != null)
m_SQLManager.AddCriteria(ref sSQL,
m_SQLManager.SQLDate(">=","events_last_updated",eventsLastUpdatedFrom
), !(sSQL.Contains("WHERE")));
if (eventsLastUpdatedTo != null)
m_SQLManager.AddCriteria(ref sSQL,
m_SQLManager.SQLDate("<=","events_last_updated",logsLastUpdatedTo),
!(sSQL.Contains("WHERE")));
if (commTypeId > -1)
m_SQLManager.AddCriteria(ref sSQL, "comm_type_id = " +
Convert.ToString(commTypeId), !(sSQL.Contains("WHERE")));
if (cap1SynchError > -1)
m_SQLManager.AddCriteria(ref sSQL, "cap1_synch_error = " +
Convert.ToString(cap1SynchError), !(sSQL.Contains("WHERE")));
if (cap2SynchError > -1)
m_SQLManager.AddCriteria(ref sSQL, "cap2_synch_error = " +
Convert.ToString(cap2SynchError), !(sSQL.Contains("WHERE")));
if (cap3SynchError > -1)
m_SQLManager.AddCriteria(ref sSQL, "cap3_synch_error = " +
Convert.ToString(cap3SynchError), !(sSQL.Contains("WHERE")));
if (memFreeSpaceFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "mem_free_space >= " +
Convert.ToString(memFreeSpaceFrom), !(sSQL.Contains("WHERE")));
if (memFreeSpaceTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "mem_free_space <= " +
Convert.ToString(memFreeSpaceTo), !(sSQL.Contains("WHERE")));
if (opModeId > -1)
m_SQLManager.AddCriteria(ref sSQL, "op_mode_id = " +
Convert.ToString(opModeId), !(sSQL.Contains("WHERE")));
if (logPeriodFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "log_period >= " +
Convert.ToString(logPeriodFrom), !(sSQL.Contains("WHERE")));
if (logPeriodTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "log_period <= " +
Convert.ToString(logPeriodTo), !(sSQL.Contains("WHERE")));
if (fileChangePeriodFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "file_change_period >= " +
Convert.ToString(fileChangePeriodFrom), !(sSQL.Contains("WHERE")));
if (fileChangePeriodTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "file_change_period <= " +
Convert.ToString(fileChangePeriodTo), !(sSQL.Contains("WHERE")));
if (qmaxFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "Qmax >= " +
Convert.ToString(qmaxFrom), !(sSQL.Contains("WHERE")));
if (qmaxTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "Qmax <= " +
Convert.ToString(qmaxTo), !(sSQL.Contains("WHERE")));
```

```
if (qminFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "Qmin >= " +
Convert.ToString(qminFrom), !(sSQL.Contains("WHERE")));
if (qminTo > 0)

m_SQLManager.AddCriteria(ref sSQL, "Qmin <= " +
Convert.ToString(qminTo), !(sSQL.Contains("WHERE")));
if (vmaxFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "Vmax >= " +
Convert.ToString(vmaxFrom), !(sSQL.Contains("WHERE")));
if (vmaxTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "Vmax <= " +
Convert.ToString(vmaxTo), !(sSQL.Contains("WHERE")));
if (vminFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "Vmin >= " +
Convert.ToString(vminFrom), !(sSQL.Contains("WHERE")));
if (vminTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "Vmin <= " +
Convert.ToString(vminTo), !(sSQL.Contains("WHERE")));
if (vUserOffsetFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "V_UserOffset >= " +
Convert.ToString(vUserOffsetFrom), !(sSQL.Contains("WHERE")));
if (vUserOffsetTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "V_UserOffset <= " +
Convert.ToString(vUserOffsetTo), !(sSQL.Contains("WHERE")));
if (vAlphaFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "V_Alpha >= " +
Convert.ToString(vAlphaFrom), !(sSQL.Contains("WHERE")));
if (vAlphaTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "V_Alpha <= " +
Convert.ToString(vAlphaTo), !(sSQL.Contains("WHERE")));
if (pUserOffsetFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "P_UserOffset >= " +
Convert.ToString(pUserOffsetFrom), !(sSQL.Contains("WHERE")));
if (pUserOffsetTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "P_UserOffset <= " +
Convert.ToString(pUserOffsetTo), !(sSQL.Contains("WHERE")));
if (pAlphaFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "P_Alpha >= " +
Convert.ToString(pAlphaFrom), !(sSQL.Contains("WHERE")));
if (pAlphaTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "P_Alpha <= " +
Convert.ToString(pAlphaTo), !(sSQL.Contains("WHERE")));
if (qUserOffsetFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "Q_UserOffset >= " +
Convert.ToString(qUserOffsetFrom), !(sSQL.Contains("WHERE")));
if (qUserOffsetTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "Q_UserOffset <= " +
Convert.ToString(qUserOffsetTo), !(sSQL.Contains("WHERE")));
if (qAlphaFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "Q_Alpha >= " +
Convert.ToString(qAlphaFrom), !(sSQL.Contains("WHERE")));
if (qAlphaTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "Q_Alpha <= " +
Convert.ToString(qAlphaTo), !(sSQL.Contains("WHERE")));
if (i1UserOffsetFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "I1_UserOffset >= " +
Convert.ToString(i1UserOffsetFrom), !(sSQL.Contains("WHERE")));
if (i1UserOffsetTo > 0)
```

```
m_SQLManager.AddCriteria(ref sSQL, "I1_UserOffset <= " +
Convert.ToString(i1UserOffsetTo), !(sSQL.Contains("WHERE")));
if (i1AlphaFrom > 0)

m_SQLManager.AddCriteria(ref sSQL, "I1_Alpha >= " +
Convert.ToString(i1AlphaFrom), !(sSQL.Contains("WHERE")));
if (i1AlphaTo > 0)


m_SQLManager.AddCriteria(ref sSQL, "I1_Alpha <= " +
Convert.ToString(i1AlphaTo), !(sSQL.Contains("WHERE")));
if (i2UserOffsetFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "I2_UserOffset >= " +
Convert.ToString(i2UserOffsetFrom), !(sSQL.Contains("WHERE")));
if (i2UserOffsetTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "I2_UserOffset <= " +
Convert.ToString(i2UserOffsetTo), !(sSQL.Contains("WHERE")));
if (i2AlphaFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "I2_Alpha >= " +
Convert.ToString(i2AlphaFrom), !(sSQL.Contains("WHERE")));
if (i2AlphaTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "I2_Alpha <= " +
Convert.ToString(i2AlphaTo), !(sSQL.Contains("WHERE")));
if (i3UserOffsetFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "I3_UserOffset >= " +
Convert.ToString(i3UserOffsetFrom), !(sSQL.Contains("WHERE")));
if (i3UserOffsetTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "I3_UserOffset <= " +
Convert.ToString(i3UserOffsetTo), !(sSQL.Contains("WHERE")));
if (i3AlphaFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "I3_Alpha >= " +
Convert.ToString(i3AlphaFrom), !(sSQL.Contains("WHERE")));
if (i3AlphaTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "I3_Alpha <= " +
Convert.ToString(i3AlphaTo), !(sSQL.Contains("WHERE")));
if (vOffsetFrom > 0)


m_SQLManager.AddCriteria(ref sSQL, "V_Offset >= " +
Convert.ToString(vOffsetFrom), !(sSQL.Contains("WHERE")));
if (vOffsetTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "V_Offset <= " +
Convert.ToString(vOffsetTo), !(sSQL.Contains("WHERE")));
if (vGainFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "V_Gain >= " +
Convert.ToString(vGainFrom), !(sSQL.Contains("WHERE")));
if (vGainTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "V_Gain <= " +
Convert.ToString(vGainTo), !(sSQL.Contains("WHERE")));
if (pOffsetFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "P_Offset >= " +
Convert.ToString(pOffsetFrom), !(sSQL.Contains("WHERE")));
if (pOffsetTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "P_Offset <= " +
Convert.ToString(pOffsetTo), !(sSQL.Contains("WHERE")));
if (pGainFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "P_Gain >= " +
Convert.ToString(pGainFrom), !(sSQL.Contains("WHERE")));
if (pGainTo > 0)
```

```
m_SQLManager.AddCriteria(ref sSQL, "P_Gain <= " +
Convert.ToString(pGainTo), !(sSQL.Contains("WHERE")));
if (qOffsetFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "Q_Offset >= " +
Convert.ToString(qOffsetFrom), !(sSQL.Contains("WHERE")));
if (qOffsetTo > 0)

m_SQLManager.AddCriteria(ref sSQL, "Q_Offset <= " +
Convert.ToString(qOffsetTo), !(sSQL.Contains("WHERE")));
if (qGainFrom > 0)


m_SQLManager.AddCriteria(ref sSQL, "Q_Gain >= " +
Convert.ToString(qGainFrom), !(sSQL.Contains("WHERE")));
if (qGainTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "Q_Gain <= " +
Convert.ToString(qGainTo), !(sSQL.Contains("WHERE")));
if (i1OffsetFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "I1_Offset >= " +
Convert.ToString(i1OffsetFrom), !(sSQL.Contains("WHERE")));
if (i1OffsetTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "I1_Offset <= " +
Convert.ToString(i1OffsetTo), !(sSQL.Contains("WHERE")));
if (i1GainFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "I1_Gain >= " +
Convert.ToString(i1GainFrom), !(sSQL.Contains("WHERE")));
if (i1GainTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "I1_Gain <= " +
Convert.ToString(i1GainTo), !(sSQL.Contains("WHERE")));
if (i2OffsetFrom > 0)


m_SQLManager.AddCriteria(ref sSQL, "I2_Offset >= " +
Convert.ToString(i2OffsetFrom), !(sSQL.Contains("WHERE")));
if (i2OffsetTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "I2_Offset <= " +
Convert.ToString(i2OffsetTo), !(sSQL.Contains("WHERE")));
if (i2GainFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "I2_Gain >= " +
Convert.ToString(i2GainFrom), !(sSQL.Contains("WHERE")));
if (i2GainTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "I2_Gain <= " +
Convert.ToString(i2GainTo), !(sSQL.Contains("WHERE")));
if (i3OffsetFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "I3_Offset >= " +
Convert.ToString(i3OffsetFrom), !(sSQL.Contains("WHERE")));
if (i3OffsetTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "I3_Offset <= " +
Convert.ToString(i3OffsetTo), !(sSQL.Contains("WHERE")));
if (i3GainFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "I3_Gain >= " +
Convert.ToString(i3GainFrom), !(sSQL.Contains("WHERE")));
if (i3GainTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "I3_Gain <= " +
Convert.ToString(i3GainTo), !(sSQL.Contains("WHERE")));
if (regionId > -1)
m_SQLManager.AddCriteria(ref sSQL, "region_id = " +
Convert.ToString(regionId), !(sSQL.Contains("WHERE")));
if (communationEnable > -1)
```

```
m_SQLManager.AddCriteria(ref sSQL, "communation_enable = " +
Convert.ToString(communationEnable), !(sSQL.Contains("WHERE")));
if (communationIntervalFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "communation_interval >= " +
Convert.ToString(communationIntervalFrom),
!(sSQL.Contains("WHERE")));
if (communationIntervalTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "communation_interval <= " +
Convert.ToString(communationIntervalTo), !(sSQL.Contains("WHERE")));

if (communationTryFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "communation_try >= " +
Convert.ToString(communationTryFrom), !(sSQL.Contains("WHERE")));
if (communationTryTo > 0)

m_SQLManager.AddCriteria(ref sSQL, "communation_try <= " +
Convert.ToString(communationTryTo), !(sSQL.Contains("WHERE")));
if (stationAreaId > -1)
m_SQLManager.AddCriteria(ref sSQL, "station_area_id = " +
Convert.ToString(stationAreaId), !(sSQL.Contains("WHERE")));

oDs = m_SQLManager.ExecuteSQL(sSQL, ref oConn);
return oDs;
}
catch{

return null;
}
}

public DataSet CollectStationAreas(long regionId, long commTypeId,
long opModeId){
string sSQL;
string sCriteria="";
OleDbConnection oConn = new OleDbConnection();
DataSet oDs = new DataSet();

try{
if (regionId > -1)
m_SQLManager.AddCriteria(ref sCriteria, "Region = " +
Convert.ToString(regionId), true);
if (commTypeId > -1)
m_SQLManager.AddCriteria(ref sCriteria,"CommType = " +
Convert.ToString(commTypeId),!(sCriteria.Contains("WHERE")));
if (opModeId > -1)
m_SQLManager.AddCriteria(ref sCriteria, "OpMode = " +
Convert.ToString(opModeId), !(sCriteria.Contains("WHERE")));

sSQL = "SELECT station_area.ID AS AreaId,station_area.Name AS
AreaName," +
"MAX(station_area.Latitude) AS
AreaLatitude,MAX(station_area.Longitude) " +
"AS AreaLongitude,MAX(CAST(alarm1 AS INT)) AS AreaHasProblem " +
"FROM station_area INNER JOIN " +
"(SELECT alarm1,Area FROM GMAP_VIEW" + sCriteria +
") AS StationAreas ON station_area.id = Area " +
"GROUP BY station_area.ID,station_area.name";

oDs = m_SQLManager.ExecuteSQL(sSQL, ref oConn);
return oDs;
```

```
}
catch{
return null;
}
}

}

}
```

5.1.10 CStationArea.cs

```
using System;
using System.Data;
using System.Data.OleDb;


namespace StationLibrary {

public class CStationArea {

private long m_Id;
private string m_Name;
private double m_Latitude;
private double m_Longitude;

private SQLManager m_SQLManager;

public CStationArea() {
m_SQLManager = new SQLManager();
}

public CStationArea(long id, string name, double latitude, double
longitude) {
m_Id = id;
m_Name = name;
m_Latitude = latitude;
m_Longitude = longitude;
m_SQLManager = new SQLManager();
}

public long ID {
get {
return m_Id;
}
}

public string Name {
get {
return m_Name;
}
}

public double Latitude {
get {
return m_Latitude;
}
}
```

```
public double Longitude {
get {
return m_Longitude;
}
}


public void Load(long id) {
string sSQL;


OleDbConnection oConn = new OleDbConnection();

DataSet oDs = new DataSet();

try {
sSQL = "SELECT * FROM station_area WHERE ID = " +
Convert.ToString(id);
oDs = m_SQLManager.ExecuteSQL(sSQL, ref oConn);

if (oDs.Tables[0].Rows.Count > 0) {
m_Id = Convert.ToInt64(oDs.Tables[0].Rows[0]["ID"]);
m_Name = Convert.ToString(oDs.Tables[0].Rows[0]["Name"]);
m_Latitude = Convert.ToDouble(oDs.Tables[0].Rows[0]["Latitude"]);
m_Longitude = Convert.ToDouble(oDs.Tables[0].Rows[0]["Longitude"]);
}
} catch {
}
}

public DataSet Collect(string fields, string ids, string name, string
description,
double latitudeFrom, double latitudeTo, double longitudeFrom,
double longitudeTo) {
string sSQL;
OleDbConnection oConn = new OleDbConnection();
DataSet oDs = new DataSet();

try {
sSQL = "SELECT " + fields + " FROM station_area";

if (ids != "")
m_SQLManager.AddCriteria(ref sSQL, "ID IN (" + ids + ")", true);
if (name != "")
m_SQLManager.AddCriteria(ref sSQL, m_SQLManager.SQLLike("Name",
name), !(sSQL.Contains("WHERE")));
if (latitudeFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "Latitude >= " +
Convert.ToString(latitudeFrom), !(sSQL.Contains("WHERE")));
if (latitudeTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "Latitude <= " +
Convert.ToString(latitudeTo), !(sSQL.Contains("WHERE")));
if (longitudeFrom > 0)
m_SQLManager.AddCriteria(ref sSQL, "Longitude >= " +
Convert.ToString(longitudeFrom), !(sSQL.Contains("WHERE")));
if (longitudeTo > 0)
m_SQLManager.AddCriteria(ref sSQL, "Longitude <= " +
Convert.ToString(longitudeTo), !(sSQL.Contains("WHERE")));

oDs = m_SQLManager.ExecuteSQL(sSQL, ref oConn);
```

```
return oDs;
} catch {
return null;

}
}

public DataSet CollectAreaStationsDetails(long areaId, long regionId,
long commTypeId, long opModeId) {

string sSQL;
OleDbConnection oConn = new OleDbConnection();
DataSet oDs = new DataSet();


try {

sSQL = "SELECT station_area.name AS AreaName,GMAP_VIEW.name AS
StationName," +
"CONVERT(varchar,Last_log_entry,103) + ' '" +"+
SUBSTRING(CONVERT(varchar,Last_log_entry,108),1,5)" +
" AS
StationLastLogEntry,GMAP_VIEW.C1,GMAP_VIEW.C2,GMAP_VIEW.C3,GMAP_VIEW.
V,GMAP_VIEW.P," +
"GMAP_VIEW.Q,GMAP_VIEW.alarm1" +
" FROM GMAP_VIEW INNER JOIN station_area ON station_area.ID =
GMAP_VIEW.Area";

if (areaId > -1)
m_SQLManager.AddCriteria(ref sSQL, "Area = " +
Convert.ToString(areaId), true);
if (commTypeId > -1)
m_SQLManager.AddCriteria(ref sSQL, "CommType = " +
Convert.ToString(commTypeId), !(sSQL.Contains("WHERE")));
if (opModeId > -1)
m_SQLManager.AddCriteria(ref sSQL, "OpMode = " +
Convert.ToString(opModeId), !(sSQL.Contains("WHERE")));
if (regionId > -1)
m_SQLManager.AddCriteria(ref sSQL, "Region = " +
Convert.ToString(regionId), !(sSQL.Contains("WHERE")));

oDs = m_SQLManager.ExecuteSQL(sSQL, ref oConn);
return oDs;
} catch {
return null;
}
}

}

}
```

5.1.11 SQLManager.cs

```
using System;

using System.Data;
using System.Data.OleDb;
```

```
namespace StationLibrary {

public class SQLManager {

public static string ConnectionString;

public SQLManager() { }

public string EncryptConnectionString(string connectionString) {

byte[] b =
System.Text.ASCIIEncoding.ASCII.GetBytes(connectionString);
string encryptedConnectionString = Convert.ToBase64String(b);
return encryptedConnectionString;

}

public string DecryptConnectionString(string connectionString) {
byte[] b = Convert.FromBase64String(connectionString);
string decryptedConnectionString =
System.Text.ASCIIEncoding.ASCII.GetString(b);
return decryptedConnectionString;
}

public bool ConnectSQL(ref OleDbConnection connection, string
connectionString) {
try {
if (connection == null)
connection = new OleDbConnection();

if (!(connection.State == ConnectionState.Closed))
connection.Close();

connection.ConnectionString = connectionString;
connection.Open();
return true;
} catch {
return false;
}
}

public bool ConnectSQL(ref OleDbConnection connection, string server,
string database, string username, string password) {
string sConnectionString = "Provider=sqlOLEDB.1;Persist Security
Info=False;" +
"user id=" + username + ";" +
"password=" + password + ";" +
"Initial Catalog=" + database + ";" +
"Data Source=" + server + ";";

return ConnectSQL(ref connection, sConnectionString);

}

public bool ConnectSQL(ref OleDbConnection connection) {
return ConnectSQL(ref connection, ConnectionString);
}
```

```
public DataSet ExecuteSQL(string sql, ref OleDbConnection connection,
string connectionString) {
DataSet oDs = new DataSet();

try {
if (ConnectSQL(ref connection, connectionString)) {
OleDbDataAdapter dbAdapt = new OleDbDataAdapter(sql, connection);
dbAdapt.Fill(oDs, "sql_DATA");
}
} catch {
} finally {
connection.Close();
}

return oDs;
}



public DataSet ExecuteSQL(string sql, ref OleDbConnection connection)
{
return ExecuteSQL(sql, ref connection, ConnectionString);
}

public void AddCriteria(ref string sql, string criteria, bool
addWhere) {
if (!addWhere)
sql = sql + " AND " + criteria;
else
sql = sql + " WHERE " + criteria;
}

public string EnQuote(string text) {
return "'" + text + "'";
}

public string SQLLike(string field, string text) {
return field + " LIKE " + "'%" + text + "%'";
}

public string SQLDate(string compOperator , string field, DateTime
dateTime) {
return field + " " + compOperator + " CONVERT(DATETIME , '" +
String.Format("{0:yyyy-MM-dd HH:mm:ss}",dateTime) + "' , 120)";
}

}

}
```

5.2 StationLocator

5.2.1 Default.aspx

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="Default.aspx.cs" Inherits="_Default"%>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="headStationLocator" runat="server">
    <title>ΠΕΡΙΗΓΗΤΗΣ ΣΤΑΘΜΩΝ ΑΝΑ ΤΗΝ ΕΛΛΑΔΑ</title>
    <script
src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=ABQIAAAAUuk
M5G9wy93VqONz3LwvWhQ6ychhZpw5khizo-KZI37zlHpfPBTM6dLNtr2tA-
ZRsWac88nTJsuAYg"
    type="text/javascript"></script>
    <link type="text/css" href="style/main.css" rel="STYLESHEET"/>
</head>
<body>

    <form id="formStationLocator" runat="server">
        <asp:ScriptManager ID="ScriptManager1" runat="server">
            <Services><asp:ServiceReference
Path="StationAreasWS.asmx" /></Services>

            <Scripts><asp:ScriptReference
Path="~/scripts/MapFunctions.js" /></Scripts>
        </asp:ScriptManager>
        <div align="center">
            <div class="mainHolder">
            <div id="myTitle">
                    ΠΕΡΙΗΓΗΤΗΣ ΣΤΑΘΜΩΝ ΑΝΑ ΤΗΝ ΕΛΛΑΔΑ
            </div>
            <table>
                <tr>
                    <td valign="top">
                        <table cellspacing="5" class="myTable">
                            <tr>
                                <td align="left">ΓΕΩΓΡΑΦΙΚΗ
ΠΕΡΙΟΧΗ</td>
                            </tr>
                            <tr>
                                <td>
                                    <asp:DropDownList ID="ddlRegion"
runat="server" CssClass="myDropDownList" ></asp:DropDownList>
                                </td>
                            </tr>
                            <tr>

                             <td align="left">ΤΡΟΠΟΣ
ΕΠΙΚΟΙΝΩΝΙΑΣ</td>
                            </tr>
                            <tr>
                                <td>
                                    <asp:DropDownList
ID="ddlCommType" runat="server" CssClass="myDropDownList"
></asp:DropDownList>
                                </td>
                            </tr>
                            <tr>
                                <td align="left">ΚΑΤΑΣΤΑΣΗ</td>
                            </tr>
                            <tr>
                                <td>
```

```
<asp:DropDownList ID="ddlOperationMode" runat="server"
CssClass="myDropDownList" ></asp:DropDownList>
                                    </td>
                                </tr>
                                <tr>
                                    <td>
                                        <asp:UpdatePanel
ID="UpdatePanel1" runat="server">
                                            <ContentTemplate>
                                                <div id = "buttons">
                                                    <table align="left">

<tr><td><asp:Button ID="btnRefresh" runat="server" Text="ΑΝΑΝΕΩΣΗ
ΧΑΡΤΗ" OnClientClick="DrawMap()" width="175px"/></td></tr>

<tr><td><asp:Button ID="btnClear" runat="server" Text="ΚΑΘΑΡΙΣΜΟΣ
ΧΑΡΤΗ" OnClientClick="ClearMarkers()" Width="175px" /></td></tr>

<tr><td><asp:Button ID="btnMapFocusOnCenter" runat="server"


Text="ΕΠΑΝΑΦΟΡΑ ΧΑΡΤΗ" OnClientClick="MapFocusOnCenter()"
Width="175px" /></td></tr>
                                                    </table>
                                                </div>
                                            </ContentTemplate>
                                        </asp:UpdatePanel>
                                    </td>
                                </tr>
                                <tr>
                                </tr>
                            </table>
                        </td>
                        <td><div id="map"></div></td>
                    </tr>
                </table>
                </div>
            </div>

        </form>

</body>
</html>
```

5.2.2 Default.aspx.cs

```
using System;
using System.Configuration;
using System.Data;
using System.Web.UI;
using System.Web.UI.WebControls;
using StationLibrary;

public partial class _Default : System.Web.UI.Page {

    protected void Page_Load(object sender, EventArgs e) {
        SQLManager.ConnectionString =
ConfigurationManager.ConnectionStrings["StationLocatorConnectionStrin
g"].ConnectionString;
```

```
        FillRegions();
        FillCommTypes();
        FillOperationModes();
    }

    //ΓΕΜΙΣΜΑ ΛΙΣΤΑΣ ΓΕΩΓΡΑΦΙΚΩΝ ΔΙΑΜΕΡΙΣΜΑΤΩΝ
    protected void FillRegions() {
        DataSet oDs;
        CRegionCode oRegionCode;

        ddlRegion.Items.Clear();
        ddlRegion.Items.Add(new ListItem("<Εμφάνιση Όλων>", "-1"));
        try {
            oRegionCode = new CRegionCode();
            oDs = oRegionCode.Collect("ID,name","","");

      if ((oDs.Tables[0] != null)) {
                foreach (DataRow Row in oDs.Tables[0].Rows)
                    ddlRegion.Items.Add(new
ListItem(Convert.ToString(Row["name"]),Convert.ToString(Row["ID"])));
            }
            ddlRegion.SelectedValue = "-1";

       }
        catch {
        }
    }

    //ΓΕΜΙΣΜΑ ΛΙΣΤΑΣ ΤΡΟΠΩΝ ΕΠΙΚΟΙΝΩΝΙΑΣ
    protected void FillCommTypes() {
        DataSet oDs;
        CCommType oCommType;


    ddlCommType.Items.Clear();
        ddlCommType.Items.Add(new ListItem("<Εμφάνιση Όλων>", "-1"));
        try {
            oCommType = new CCommType();
            oDs = oCommType.Collect("ID,type", "", "");
            if ((oDs.Tables[0] != null)) {
                foreach (DataRow Row in oDs.Tables[0].Rows)
                    ddlCommType.Items.Add(new
ListItem(Convert.ToString(Row["type"]),
Convert.ToString(Row["ID"])));
            }
            ddlCommType.SelectedValue = "-1";
        } catch {
        }
    }

    //ΓΕΜΙΣΜΑ ΛΙΣΤΑΣ ΚΑΤΑΣΤΑΣΕΩΝ
    protected void FillOperationModes() {
        DataSet oDs;
        COperationMode oOperationMode;

        ddlOperationMode.Items.Clear();
        ddlOperationMode.Items.Add(new ListItem("<Εμφάνιση Όλων>", "-
1"));
        try {
            oOperationMode = new COperationMode();
```

```
            oDs = oOperationMode.Collect("ID,name", "", "");
            if ((oDs.Tables[0] != null)) {
                foreach (DataRow Row in oDs.Tables[0].Rows)
                    ddlOperationMode.Items.Add(new
ListItem(Convert.ToString(Row["name"]),
Convert.ToString(Row["ID"])));
            }
            ddlOperationMode.SelectedValue = "-1";
        } catch {
        }
    }

}
```

5.2.3 StationAreasWS.asmx

```
<%@ WebService Language="C#"
CodeBehind="~/App_Code/StationAreasWS.cs"
Class="StationLocator.StationAreasWS"%>
```

5.2.4 StationAreasWS.cs

```
using System;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;

using System.Data;
using System.Configuration;
using StationLibrary;

namespace StationLocator {

    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [System.Web.Script.Services.ScriptService()]
    public class StationAreasWS : System.Web.Services.WebService {

        public StationAreasWS() {
            SQLManager.ConnectionString =
ConfigurationManager.ConnectionStrings["StationLocatorConnectionStrin
g"].ConnectionString;
        }

        [WebMethod(true, Description = "Συλλέγει από τη Βάση
Δεδομένων τις Τοποθεσίες που έχουν Σταθμούς με κριτήρια το Γεωγραφικη
Περιοχή," +
                                       " τον Τρόπο Επικοινωνίας και
την Κατάσταση.")]
        public DataSet CollectAreasWithStations(long regionId, long
commTypeId, long operationModeId) {
            try {
                CStation oStation = new CStation();
                return oStation.CollectStationAreas(regionId,
commTypeId, operationModeId);
```

```
             }
            catch {
                return null;
            }
        }

        [WebMethod(true, Description = "Συλλέγει από τη Βάση
Δεδομένων τους Σταθμούς μίας Τοποθεσίας με κριτήρια το Γεωγραφική
Περιοχή," +
                                       " τον Τρόπο Επικοινωνίας και
την Κατάσταση.")]
        public DataSet CollectAreaStationsDetails(long areaId, long
regionId, long commTypeId, long operationModeId) {
            try {
                CStationArea oStationArea = new CStationArea();
                return
oStationArea.CollectAreaStationsDetails(areaId, regionId, commTypeId,
operationModeId);
            }
            catch {
                return null;
            }
        }


    }

}
```

5.2.5 MapFunctions.js

```
//*********************************************************************
// ΣΧΕΔΙΑΣΗ ΚΑΙ ΛΕΙΤΟΥΡΓΙΕΣ ΧΑΡΤΗ
//*********************************************************************

//----- ΑΡΧΙΚΟΠΟΙΗΣΗ -ΣΤΟΙΧΕΙΑ ΚΕΝΤΡΟΥ ΧΑΡΤΗ ----------
//size: 600 - 500
var centerLatitude = 37.9853;
var centerLongitude = 23.7191;
var centerDescription = 'ΑΘΗΝΑ';

//size 1000-900
//var centerLatitude = 38.38831;
//var centerLongitude = 24.42051;
//var centerDescription = 'ΚΕΝΤΡΟ ΧΑΡΤΗ';
//var startZoom = 7;


var startZoom = 6;

var map;
var region;
var commType;
var operationMode;
var currentMarker;
//-----------------------------------------------------

//--- ΕΙΚΟΝΙΔΙΑ ΓΙΑ ΤΙΣ ΠΕΡΙΟΧΕΣ----------------------
```

```
var customIcon = new GIcon(G_DEFAULT_ICON);
customIcon.iconSize = new GSize(20,34);
customIcon.shadowSize = new GSize(0, 0);


var customIcons = [];
customIcons['green'] = new GIcon(customIcon,
'/StationLocator/images/marker_green.png');
customIcons['red'] = new GIcon(customIcon,
'/StationLocator/images/marker_red.png');
//-----------------------------------------------------

//--- ΣΥΝΑΡΤΗΣΗ ΓΙΑ ΝΑ ΠΑΙΡΝΟΥΜΕ ΤΟ ΧΡΩΜΑ ΤΟΥ ΚΕΛΙΟΥ--
//--- ΓΙΑ ΤΙΣ ΚΑΤΑΣΤΑΣΕΙΣ ΤΩΝ C1,C2,C3----------------
function getColourForC(cellState) {
    switch(cellState) {
        case 'ON':
            return '#90EE90';
        case 'OFF':
            return '#FFFF00';

      case 'OUT':
            return '#FFFFFF';
        case 'FAULT':
            return '#FF0000';
        case 'FLT/OUT':

           return '#FFC0CB';
    }
}
//-----------------------------------------------------

//--- ΣΥΝΑΡΤΗΣΕΙΣ ΓΙΑ ΝΑ ΠΑΙΡΝΟΥΜΕ ΧΡΩΜΑ,ΚΕΙΜΕΝΟ------
//---ΤΟΥ ΚΕΛΙΟΥ ΓΙΑ ΤΗΝ ΥΠΑΡΞΗ ΠΡΟΒΛΗΜΑΤΟΣ------------
function getColourForAlarm(alarm) {
    if(alarm)
        return '#FF7055';
    else
        return '#90EE90';
}

function getTextForAlarm(alarm) {
    if(alarm)
        return 'ON';
    else
        return '--';
}
//-----------------------------------------------------

function Init() {
    if (GBrowserIsCompatible()) {
        region = -1;
        commType = -1;
        operationMode = -1;
        currentMarker = null;
        map = new GMap2($get("map"));
        map.addControl(new GSmallMapControl());
        map.setCenter(new GLatLng(centerLatitude, centerLongitude),
startZoom);
    }
```

```
      else {
        $get('formStationLocator').innerHTML = '<h1>Ο Browser δεν
υποστηρίζει Google Maps!</h1>';
    }
}

//*****************************************************************
******


//*****************************************************************
******
// ΕΜΦΑΝΙΣΗ ΤΟΠΟΘΕΣΙΩΝ ΜΕ ΣΤΑΘΜΟΥΣ ΣΤΟ ΧΑΡΤΗ
//*****************************************************************
******
function ClearMarkers() {
    map.clearOverlays();
}

function MapFocusOnCenter() {
    if (currentMarker != null) {
        currentMarker.closeInfoWindow();

     currentMarker = null;
    }
    map.setCenter(new GLatLng(centerLatitude, centerLongitude),
startZoom);
}

function DrawMap() {
    ClearMarkers();
    region = $get('ddlRegion').value;
    commType = $get('ddlCommType').value;
    operationMode = $get('ddlOperationMode').value;

StationLocator.StationAreasWS.CollectAreasWithStations(region,commTyp
e,operationMode,OnRequestAreasSuccess, OnRequestAreasFail);
}

function OnRequestAreasSuccess(results) {
    var areaId;
    var areaName;
    var areaLatitude;
    var areaLongitude;
    var areaHasProblem;
    var i;

    if (results != null) {
        for (i = 0; i < results.tables[0].rows.length; i++) {
            areaId = results.tables[0].rows[i].AreaId;
            areaName = results.tables[0].rows[i].AreaName;
            areaLatitude = results.tables[0].rows[i].AreaLatitude;
            areaLongitude = results.tables[0].rows[i].AreaLongitude;
            areaHasProblem =
results.tables[0].rows[i].AreaHasProblem;
            addMarker(areaId, areaName, areaLatitude, areaLongitude,
areaHasProblem);
        }
    }
}
```

```
function OnRequestAreasFail() {

 alert('Η ενημέρωση του χάρτη απέτυχε!');
}

function addMarker(areaId, areaName, areaLatitude, areaLongitude,
areaHasProblem) {
    var markerOptions;

    if (areaHasProblem)
        markerOptions = { icon: customIcons['red'] , title : areaName
};
    else
        markerOptions = { icon: customIcons['green'], title: areaName
};

    var marker = new GMarker(new GLatLng(areaLatitude,
areaLongitude),markerOptions);
    GEvent.addListener(marker, "click", function(){
ShowInfoWindow(marker,areaId); });
    map.addOverlay(marker);
}


//--------ΔΗΜΙΟΥΡΓΙΑ ΠΑΡΑΘΥΡΩΝ ΠΛΗΡΟΦΟΡΙΩΝ------------

function ShowInfoWindow(marker, areaId) {
    currentMarker = marker;



    StationLocator.StationAreasWS.CollectAreaStationsDetails(areaId,
region, commType, operationMode,
                                OnRequestAreaStationsDetailsSuccess,
OnRequestAreaStationsDetailsFail);
}

function OnRequestAreaStationsDetailsSuccess(results) {
    var areaName;
    var stationName;
    var stationLastLogEntry;
    var stationC1;
    var stationC2;
    var stationC3;
    var stationV;
    var stationP;
    var stationQ;
    var stationAlarm;
    var i;
    var stationsNumber;
    var html;

    if (results!=null) {
        areaName = results.tables[0].rows[0].AreaName;

        stationsNumber = results.tables[0].rows.length;
        if(stationsNumber > 3)

     html ='<div style=width:470px;height:110px><div
style=overflow:auto;width:470px;height:110px>';
```

```
        else
             html = '';

         html += '<div><b>' + 'Τοποθεσία Σταθμών: ' + areaName +
'</b></div>' +
                 '<table border="1" style="font-size:small"><tr
align="center">' +
                 '<td>Ονομασία</td><td>Τελ.Εγγραφή</td><td>C1</td>' +
                 '<td>C2</td><td>C3</td><td>V</td><td>P</td>' +
                 '<td>Q</td><td>Συν/μός</td></tr>';

         for (i = 0; i < results.tables[0].rows.length; i++) {
             stationName = results.tables[0].rows[i].StationName;
             stationLastLogEntry =
results.tables[0].rows[i].StationLastLogEntry;
             stationC1 = results.tables[0].rows[i].C1;
             stationC2 = results.tables[0].rows[i].C2;
             stationC3 = results.tables[0].rows[i].C3;
             stationV = results.tables[0].rows[i].V;
             stationP = results.tables[0].rows[i].P;
             stationQ = results.tables[0].rows[i].Q;
             stationAlarm = results.tables[0].rows[i].alarm1;
             html += '<tr><td align="left">' + stationName + '</td>' +
                     '<td align="right">' + stationLastLogEntry +
'</td>' +

                  '<td align="center" style="background-color:' +
getColourForC(stationC1) + '">' + stationC1 + '</td>' +
                     '<td align="center" style="background-color:' +
getColourForC(stationC2) + '">' + stationC2 + '</td>' +


                   '<td align="center" style="background-color:' +
getColourForC(stationC3) + '">' + stationC3 + '</td>' +
                     '<td align="right">' + stationV + '</td>' +
                     '<td align="right">' + stationP + '</td>' +
                     '<td align="right">' + stationQ + '</td>' +
                     '<td align="center" style="background-color:' +
getColourForAlarm(stationAlarm) + '">' +
                    getTextForAlarm(stationAlarm)+'</td></tr>';

         }
         if(stationsNumber > 3)
             html += '</div></div>';

         html += '</table>';
         currentMarker.openInfoWindowHtml(html);
     }
}

function OnRequestAreaStationsDetailsFail(results) {
    alert('Η συλλογή των Σταθμών της Τοποθεσίας απέτυχε!');
}
//-------------------------------------------------

//*****************************************************************
******
```

```
//*********************************************************************
******
// ΓΕΝΙΚΕΣ ΛΕΙΤΟΥΡΓΙΕΣ
//*********************************************************************
******
window.onload = Init;
window.onunload = GUnload;

//*********************************************************************
******
```

5.2.6 main.css

```
body {
	background-color: #808080;
}

.mainHolder{
	border: 1px solid #C0C0C0;
	width: 855px;
	padding-bottom: 10px;
	background-color: #9F9F9F;
}


#myTitle{
	color: White;
	font-family: Verdana;
	font-size: 28px;
	margin-bottom: 15px;

	margin-top: 10px;
}

.myTable input {
	border: 1px solid #000000;
	background-color: Gray;
	color: White;
	font-family: Verdana;
	font-size: 12px;
}

.myTable input:hover {
	background-color: #000000;
	border: 1px solid Silver;
}

.myTable td {
	color: White;
	font-family: Verdana;
	font-size: 12px;
}

#map{
	width:600px;
	height:500px;
```

```
}

.myDropDownList{
      font-family:Verdana;
      font-size:12px;
      width:222px;
}
```

5.2.7 web.config

```
<?xml version="1.0"?>
<!--
    Note: As an alternative to hand editing this file you can use the
    web admin tool to configure settings for your application. Use
    the Website->Asp.Net Configuration option in Visual Studio.
    A full list of settings and comments can be found in
    machine.config.comments usually located in
    \Windows\Microsoft.Net\Framework\v2.x\Config
-->
<configuration>
      <configSections>
            <sectionGroup name="system.web.extensions"
type="System.Web.Configuration.SystemWebExtensionsSectionGroup,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35">


<sectionGroup name="scripting"
type="System.Web.Configuration.ScriptingSectionGroup,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35">


                        <section name="scriptResourceHandler"
type="System.Web.Configuration.ScriptingScriptResourceHandlerSection,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35" requirePermission="false"
allowDefinition="MachineToApplication"/>
                        <sectionGroup name="webServices"
type="System.Web.Configuration.ScriptingWebServicesSectionGroup,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35">
                              <section name="jsonSerialization"
type="System.Web.Configuration.ScriptingJsonSerializationSection,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35" requirePermission="false"
allowDefinition="Everywhere"/>

                              <section name="profileService"
type="System.Web.Configuration.ScriptingProfileServiceSection,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35" requirePermission="false"
allowDefinition="MachineToApplication"/>
                              <section name="authenticationService"
type="System.Web.Configuration.ScriptingAuthenticationServiceSection,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
```

```
PublicKeyToken=31BF3856AD364E35" requirePermission="false"
allowDefinition="MachineToApplication"/>
                              <section name="roleService"
type="System.Web.Configuration.ScriptingRoleServiceSection,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35" requirePermission="false"
allowDefinition="MachineToApplication"/>
                        </sectionGroup>
                  </sectionGroup>
            </sectionGroup>
      </configSections>
      <appSettings/>
      <connectionStrings>
            <add name="StationLocatorConnectionString"
connectionString="Provider=SQLOLEDB.1;Persist Security
Info=False;user id=sa;password=sapk;Initial Catalog=cbcMonGMAP;Data
Source=PANAGIOTIS-PC\SQL2005;"/>
    <!--<add name="StationLocatorConnectionString"
connectionString="Provider=SQLOLEDB.1;Integrated
Security=SSPI;Persist Security Info=False;Initial
Catalog=cbcMonGMAP;Data Source=PANAGIOTIS-PC\SQL2005;"/>-->
      </connectionStrings>
      <system.web>
            <!--
            Set compilation debug="true" to insert debugging
            symbols into the compiled page. Because this
            affects performance, set this value to true only
            during development.

            Visual Basic options:
            Set strict="true" to disallow all data type conversions
            where data loss can occur.

     Set explicit="true" to force declaration of all variables.
        -->
            <compilation debug="true" strict="false" explicit="true">
                  <assemblies>

                  <add assembly="System.Core, Version=3.5.0.0,
Culture=neutral, PublicKeyToken=B77A5C561934E089"/>
                        <add assembly="System.Web.Extensions,
Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>


                        <add assembly="System.Xml.Linq,
Version=3.5.0.0, Culture=neutral, PublicKeyToken=B77A5C561934E089"/>
                        <add assembly="System.Data.DataSetExtensions,
Version=3.5.0.0, Culture=neutral, PublicKeyToken=B77A5C561934E089"/>
                        <add assembly="System.Design,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=B03F5F7F11D50A3A"/>
                        <add assembly="System.Windows.Forms,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=B77A5C561934E089"/>
                        <add assembly="System.Web.Extensions,
Version=1.0.61025.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"/></assemblies>
            </compilation>
            <pages>
                  <namespaces>
                        <clear/>
                        <add namespace="System"/>
```

```
                        <add namespace="System.Collections"/>
                        <add namespace="System.Collections.Generic"/>
                        <add
namespace="System.Collections.Specialized"/>
                        <add namespace="System.Configuration"/>
                        <add namespace="System.Text"/>
                        <add
namespace="System.Text.RegularExpressions"/>
                        <add namespace="System.Linq"/>
                        <add namespace="System.Xml.Linq"/>
                        <add namespace="System.Web"/>
                        <add namespace="System.Web.Caching"/>
                        <add namespace="System.Web.SessionState"/>
                        <add namespace="System.Web.Security"/>
                        <add namespace="System.Web.Profile"/>
                        <add namespace="System.Web.UI"/>
                        <add namespace="System.Web.UI.WebControls"/>
                        <add
namespace="System.Web.UI.WebControls.WebParts"/>
                        <add namespace="System.Web.UI.HtmlControls"/>
                    </namespaces>
                    <controls>
                        <add tagPrefix="asp"
namespace="System.Web.UI" assembly="System.Web.Extensions,
Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
                        <add tagPrefix="asp"
namespace="System.Web.UI.WebControls"
assembly="System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"/>
                    </controls>
            </pages>
            <!--
            The <authentication> section enables configuration

   of the security authentication mode used by
            ASP.NET to identify an incoming user.
        -->

            <authentication mode="Windows"/>

      <!--
            The <customErrors> section enables configuration
            of what to do if/when an unhandled error occurs
            during the execution of a request. Specifically,
            it enables developers to configure html error pages
            to be displayed in place of a error stack trace.

        <customErrors mode="RemoteOnly"
defaultRedirect="GenericErrorPage.htm">
            <error statusCode="403" redirect="NoAccess.htm" />
            <error statusCode="404" redirect="FileNotFound.htm" />
        </customErrors>
        -->
            <httpHandlers>
                    <remove verb="*" path="*.asmx"/>
                    <add verb="*" path="*.asmx" validate="false"
type="System.Web.Script.Services.ScriptHandlerFactory,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"/>
```

```
<add verb="*" path="*_AppService.axd" validate="false"
type="System.Web.Script.Services.ScriptHandlerFactory,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"/>
                  <add verb="GET,HEAD" path="ScriptResource.axd"
validate="false" type="System.Web.Handlers.ScriptResourceHandler,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"/>
            </httpHandlers>
            <httpModules>
                  <add name="ScriptModule"
type="System.Web.Handlers.ScriptModule, System.Web.Extensions,
Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
            </httpModules>
      </system.web>
      <system.codedom>
            <compilers>
                  <compiler language="c#;cs;csharp" extension=".cs"
type="Microsoft.CSharp.CSharpCodeProvider,System, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089" warningLevel="4">
                        <providerOption name="CompilerVersion"
value="v3.5"/>
                        <providerOption name="WarnAsError"
value="false"/>
                  </compiler>
                  <compiler language="vb;vbs;visualbasic;vbscript"
extension=".vb" type="Microsoft.VisualBasic.VBCodeProvider, System,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"
warningLevel="4">
                        <providerOption name="CompilerVersion"
value="v3.5"/>

                        <providerOption name="OptionInfer"
value="true"/>


                  <providerOption name="WarnAsError" value="false"/>
                  </compiler>
            </compilers>
      </system.codedom>
      <!--

 The system.webServer section is required for running ASP.NET AJAX
under Internet
        Information Services 7.0.  It is not necessary for previous
version of IIS.
    -->
      <system.webServer>
            <validation validateIntegratedModeConfiguration="false"/>
            <modules>
                  <remove name="ScriptModule"/>
                  <add name="ScriptModule"
preCondition="managedHandler" type="System.Web.Handlers.ScriptModule,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"/>
            </modules>
            <handlers>
                  <remove name="WebServiceHandlerFactory-
Integrated"/>
                  <remove name="ScriptHandlerFactory"/>
```

```
                  <remove name="ScriptHandlerFactoryAppServices"/>
                  <remove name="ScriptResource"/>
                  <add name="ScriptHandlerFactory" verb="*"
path="*.asmx" preCondition="integratedMode"
type="System.Web.Script.Services.ScriptHandlerFactory,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"/>
                  <add name="ScriptHandlerFactoryAppServices"
verb="*" path="*_AppService.axd" preCondition="integratedMode"
type="System.Web.Script.Services.ScriptHandlerFactory,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"/>
                  <add name="ScriptResource" verb="GET,HEAD"
path="ScriptResource.axd" preCondition="integratedMode"
type="System.Web.Handlers.ScriptResourceHandler,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"/>
            </handlers>
      </system.webServer>
      <runtime>
            <assemblyBinding xmlns="urn:schemas-microsoft-
com:asm.v1">
                  <dependentAssembly>
                        <assemblyIdentity
name="System.Web.Extensions" publicKeyToken="31bf3856ad364e35"/>
                        <bindingRedirect oldVersion="1.0.0.0-1.1.0.0"
newVersion="3.5.0.0"/>
                  </dependentAssembly>
                  <dependentAssembly>
                        <assemblyIdentity
name="System.Web.Extensions.Design"
publicKeyToken="31bf3856ad364e35"/>
                        <bindingRedirect oldVersion="1.0.0.0-1.1.0.0"
newVersion="3.5.0.0"/>

      </dependentAssembly>
            </assemblyBinding>
      </runtime>
      <system.web.extensions>
            <scripting>
                  <webServices>

                  <jsonSerialization>
                              <converters>
                                    <add name="DataSetConverter"
type="Microsoft.Web.Preview.Script.Serialization.Converters.DataSetCo
nverter, Microsoft.Web.Preview"/>
                                    <add name="DataRowConverter"
type="Microsoft.Web.Preview.Script.Serialization.Converters.DataRowCo
nverter, Microsoft.Web.Preview"/>
                                    <add name="DataTableConverter"
type="Microsoft.Web.Preview.Script.Serialization.Converters.DataTable
Converter, Microsoft.Web.Preview"/>
                              </converters>
                        </jsonSerialization>
                  </webServices>
            </scripting>
      </system.web.extensions>
</configuration
```

6 Βιβλιογραφία – Ιστοσυνδέσεις

6.1 Βιβλιογραφία

6.1.1 Google maps

- Google Maps Hacks by Rich Gibson (Author), Schuyler Erle (Author)
- Google Maps API by Scott Davis
- Google Maps Hacks by Jens & Lars Rasmussen, Google Maps Tech Leads
- Beginning Google Maps Mashups with Mapplets, KML, and GeoRSS: From Novice to Professional by Sterling Udell
- Beginning Google Maps Applications with PHP and Ajax: From Novice to Professional

6.1.2 C#

- C# 3.0 in a Nutshell, Third Edition By Joseph Albahari, Ben Albahari
- C# in depth by Jon Skeet
- A Programmer's Introduction to C# by Eric Gunnerson
- Programming C# by Jesse Liberty
- Effective C#—50 Specific Ways to Improve Your C# by Bill Wagner

6.1.3 JavaScript

- JavaScript: The Definitive Guide by David Flanagan
- JavaScript for Dummies by Emily A. Vander Veer
- The Book of JavaScript: A Practical Guide to Interactive Web Pages by Dave Thau

6.1.4 ASP.NET

- ASP.NET Unleashed by Stephen Walther
- Professional ASP.NET 2.0 (Programmer to Programmer) by Bill Evjen (Author), Scott Hanselman (Author), Farhan Muhammad (Author), Srinivasa Sivakumar (Author), Devin Rader (Author)
- ASP.NET 2.0: A Developer's Notebook By Wei-Meng Lee

6.1.5 ASP.NET AJAX

- Beginning Ajax with ASP.NET by Wallace B. McClure (Author), Scott Cate (Author), Paul Glavich (Author), Craig Shoemaker (Author)
- Professional ASP.NET 2.0 AJAX (Programmer to Programmer)

  by Matt Gibbs (Author), Dan Wahlin (Author)
- ASP.NET AJAX in Action by Alessandro Gallo, David Barkol, and Rama Krishna Vavilala

6.1.6 Microsoft SQL Server

- Microsoft SQL Server 2005 :A Beginner"s Guide
- Microsoft SQL Server 2005 For Dummies by Andrew Watt (Author)

6.2 Ιστοσυνδέσεις

6.2.1 Wikipedia

- http://en.wikipedia.org/wiki/Google_Maps
- http://en.wikipedia.org/wiki/C_Sharp
- http://en.wikipedia.org/wiki/ASP.NET
- http://en.wikipedia.org/wiki/ASP.NET_AJAX
- http://en.wikipedia.org/wiki/JavaScript
- http://en.wikipedia.org/wiki/Microsoft_SQL_Server

- http://en.wikipedia.org/wiki/Internet_Information_Services
- http://en.wikipedia.org/wiki/JSON

6.2.2 Google Maps

- http://maps.google.com/
- http://code.google.com/apis/maps/
- http://msdn.microsoft.com/el-gr/default.aspx
- http://dotnet.sys-con.com/node/121828
- http://www.codeproject.com/KB/scripting/Use_of_Google_Map.aspx

6.2.3 C#

- http://www.csharp-station.com/
- http://www.ecma-international.org/publications/standards/Ecma-334.htm
- http://www.csharphelp.com/
- http://genamics.com/developer/csharp_comparative.htm
- http://www.csharpfriends.com/

6.2.4 JavaScript

- http://javascript.internet.com/
- http://www.javascript.com/
- http://www.w3schools.com/JS/default.asp
- http://www.ecma-international.org/publications/standards/Ecma-262.htm
- http://www.w3schools.com/JS/default.asp
- http://www.webteacher.com/javatour/

6.2.5 ASP.NET

- http://www.asp.net/
- http://www.w3schools.com/ASPNET/default.asp
- http://quickstarts.asp.net/QuickstartV20/aspnet/
- http://wiki.asp.net/
- http://www.aspnetfaq.com/

6.2.6 ASP.NET AJAX

- http://www.asp.net/ajax/
- http://www.codeproject.com/KB/ajax/aspnetajaxtips.aspx
- http://www.webreference.com/programming/asp/ajax_frameworks/
- http://www.ajaxpro.info/
- http://products.secureserver.net/guides/hostingaspnetajax.pdf

6.2.7 Microsoft SQL Server

- http://www.microsoft.com/SQL/default.mspx
- http://www.sqlservercentral.com/
- http://www.devguru.com/technologies/t-sql/home.asp
- http://searchsqlserver.techtarget.com/sDefinition/0,,sid87_gci817547,00.html
- http://msdn.microsoft.com/en-us/library/aa260642(SQL.80).aspx

6.2.8 IIS

- http://www.iis.net/
- http://www.petri.co.il/install_iis_on_windows_xp.htm

6.2.9 JSON

- http://www.json.org/
- http://www.json.org/js.html