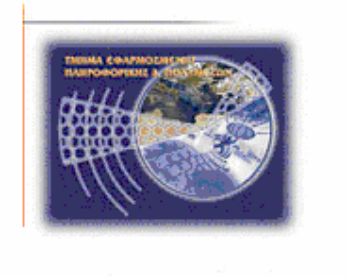
Ανώτατο Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων



Πτυχιακή εργασία

Υλοποίηση εφαρμογών Client - Server για online αποθήκευση αρχείων με χρήση C# .Net Framework και TCP/IP socket programming

Θεοδωράκος Κωνσταντίνος (ΑΜ:1279)

Επιβλέπων καθηγητής : Δρ Ιωάννης Παχουλάκης
Ημερομηνία παρουσίασης : Δευτέρα 17 Οκτωβρίου 2011

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον κύριο Δρ Ιωάννη Παχουλάκη για την δυνατότητα που μου έδωσε για να ασχοληθώ με τη συγκεκριμένη πτυχιακή εργασία. Θα ήθελα επίσης να ευχαριστήσω την οικογένειά μου για την υποστήριξη που μου παρείχε σε όλη τη διάρκεια των σπουδών μου.

ΣΥΝΟΨΗ

Το αντικείμενο της πτυχιακής εργασίας είναι η υλοποίηση δύο εφαρμογών, Client και Server. Μέσω ενός φιλικού προς το χρήστη περιβάλλον, θα παρέχεται δυνατότητα αποθήκευσης, συγχρονισμού αρχείων και λήψης αντιγράφων ασφαλείας σε online δικτυακό χώρο.

Οι Client και Server εφαρμογές θα αναπτυχθούν μέσω της πλατφόρμας .Net, με βάση δεδομένων Microsoft SQL Server, σε γλώσσα προγραμματισμού Visual C# 4.0. Θα γίνει η χρήση TCP/IP Network Sockets για τη μεταφορά των αρχείων. Τα αρχεία έπειτα θα μεταφέρονται και θα αποθηκεύονται μόνιμα σε έναν File Server.

Ο χρήστης θα κάνει εγγραφή σε μία Asp.net ιστοσελίδα η οποία θα είναι hosted σε ένα IIS server. Τα στοιχεία του κάθε χρήστη θα καταγράφονται στη βάση δεδομένων SQL.

Το login και authentication θα γίνεται μέσω της client εφαρμογής που θα τρέχει από τον Η/Υ του χρήστη. Η client εφαρμογή θα αποστέλλει τα επιλεγμένα αρχεία μέσω asynchronous file transfer στην Server εφαρμογή.

Η Server εφαρμογή θα είναι εγκατεστημένη σε Η/Υ με Windows Server 2008. Τα αρχεία αφού αποθηκευτούν προσωρινά τοπικά, θα αποστέλλονται σε έναν FTP File Server με λειτουργικό Debian Linux.

Η server εφαρμογή θα έχει επίσης τη δυνατότητα καταγραφής στατιστικών χρήσης, διαδικτυακής κίνησης και κατάστασης των αρχείων.

Λέξεις – Κλειδιά: Visual C# 4.0, .Net Framework 3.5, Tcp/Ip Network socket programming, αντικειμενοστραφής προγραμματισμός, προγραμματισμός διαδικτύου, σχεδιασμός διεπαφής, Visual Studio .Net 2010, διαχείριση βάσεων δεδομένων, Microsoft SQL 2008, data streaming, data manipulation.

ABSTRACT

The object of this diploma thesis is the implementation of two applications, Client and Server.

Through a friendly user interface, there will be provided the capability of storing, file synchronizing and the ability to back up files through an online system.

For the implementation of the Client and the Server applications I used the Integrated Development Environment (IDE) Microsoft Visual Studio 2010, the programming language Visual C# 4.0, the Database Management System (DBMS) Microsoft SQL Server 2008 and the IIS (Internet Information Services) web server. I also used the technologies such as Asp.net and CSS in order to achieve the best interaction between the user and the system.

The user will register through an Asp.net web page which will be hosted on an IIS server. The data of every user will be recorded at the SQL database.

The login and the authentication will be fulfilled through the client application which will run on the client's terminal. The client application will send the selected files through an asynchronous file transfer system to the server application.

The server application will be installed in a computer system with Windows Server 2008 operating system. The files will be stored locally and would be then sent to a Debian Linux Ftp file server.

The server application will also have the ability to record usage data, online traffic statistics and file status.

Keywords: Database, Visual C# 4.0, .Net Framework 3.5, Tcp/Ip Network socket programming, object-oriented programming, network programming, user interface design, Visual Studio .Net 2010, data base management, Microsoft SQL 2008, data streaming, data manipulation.

ΠΕΡΙΕΧΟΜΕΝΑ

ΛΙΣΤΑ ΕΝΟΤΗΤΩΝ

ΛΙΣΤΑ ΕΙΚΟΝΩΝ

ΛΙΣΤΑ ΠΙΝΑΚΩΝ

ΚΕΦΑΛΑΙΟ 1ο

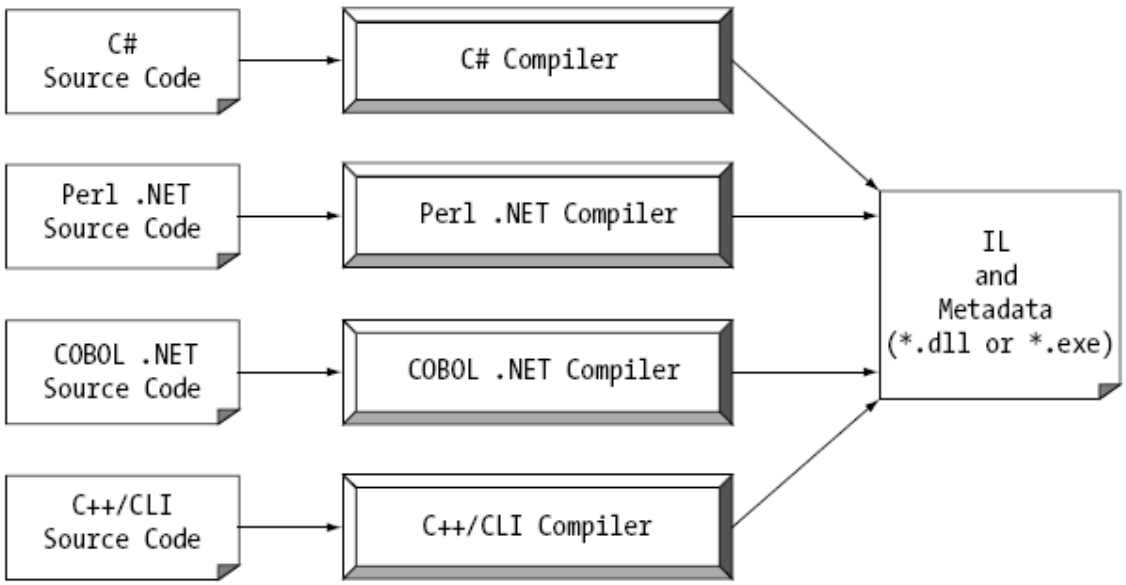
ΠΛΑΤΦΟΡΜΑ ΥΛΟΠΟΙΗΣΗΣ

1.1 Γιατί προγραμματισμός δικτύων μέσω .NET?

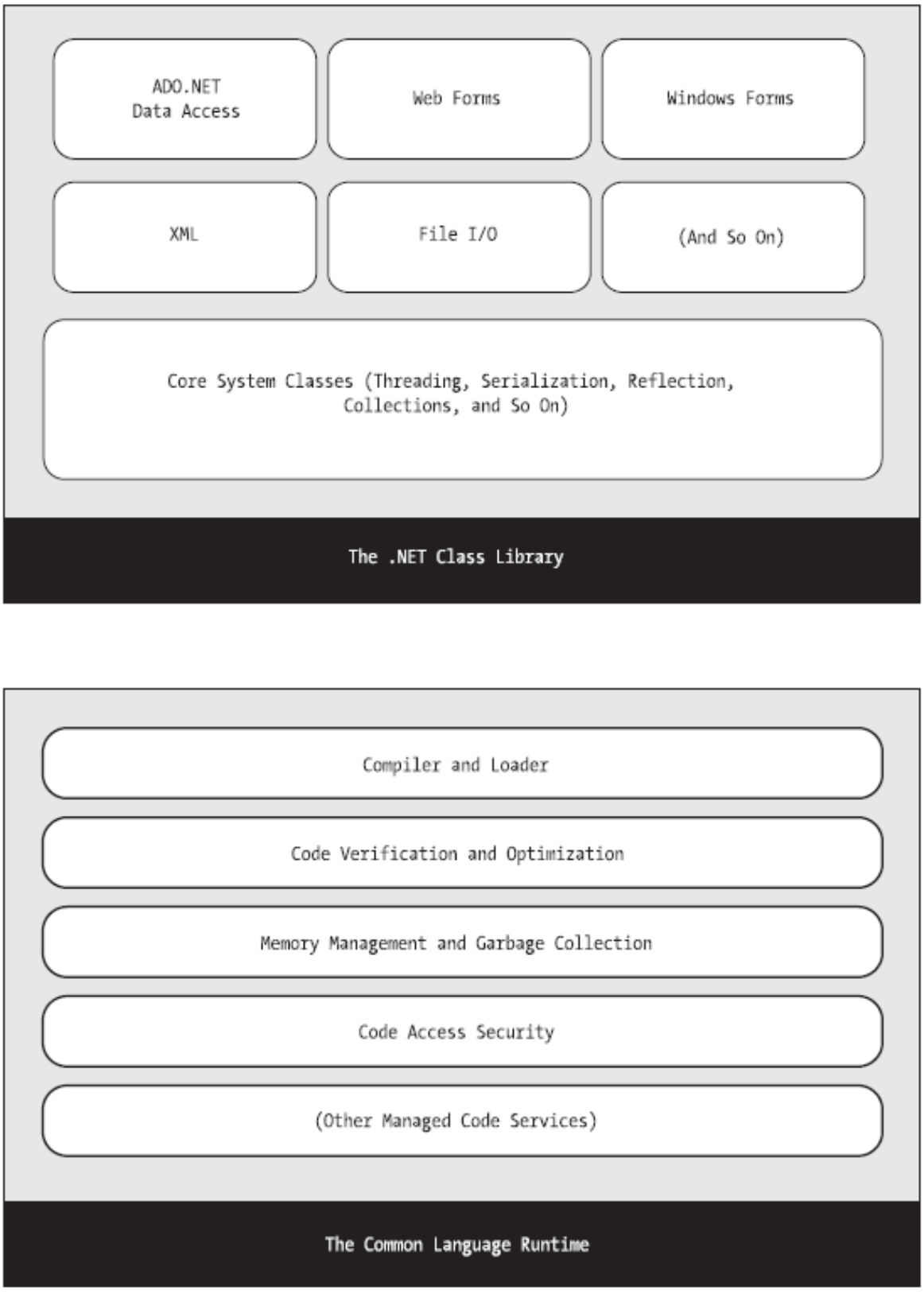
Ένα από το πρώτα ζητήματα που θα χρειαστεί να αποφασιστούν πριν απο την εκπόνηση ενός νέου project είναι το "ποιά γλώσσα προγραμματισμού θα χρησιμοποιηθεί;". Η .Net πλατφόρμα είναι ικανή να υλοποιήσει σχεδόν κάθε λύση και να προσφέρει επαρκή υποστήριξη για προγραμματισμό δικτυακών εφαρμογών. Στην ουσία, η .Net έχει πιο ουσιαστική υποστήριξη για δικτυακές εφαρμογές από κάθε άλλη πλατφόρμα που έχει αναπτυχθεί απο τη Microsoft. Η .Net δεν σημαίνει ότι είναι η αποκλειστική λύση για network - programming applications. Εάν η εφαρμογή θα έπρεπε να τρέχει σε περιβάλλον UNIX για την επικοινωνιακή υποδομή μόνο, τότε η σωστή επιλογή θα ήταν η χρήση της Java Remote Method Invocation (RMI). Στις περισσότερες περιπτώσεις όμως η .Net είναι ικανή να χειριστεί οποιοδήποτε θέμα αφορά δικτυακή επικοινωνία.

Χαρακτηριστικά που προσφέρει η .Net:

- Κοινό run time για όλες τις .Net γλώσσες προγραμματισμού.
- Δια-λειτουργικότητα υπάρχοντα κώδικα. Ήδη υπάρχοντα binaries μπορούν να συνεργαστούν με binaries γραμμένα σε μεταγενέστερες εκδόσεις .Net.
- Υποστήριξη για πολλαπλές γλώσσες προγραμματισμού. Εφαρμογές σε .Net μπορούν να δημιουργηθούν μέσα από μια πληθώρα γλωσσών προγραμματισμού (π.χ. C#, Visual Basic, F#, S# κτλ)

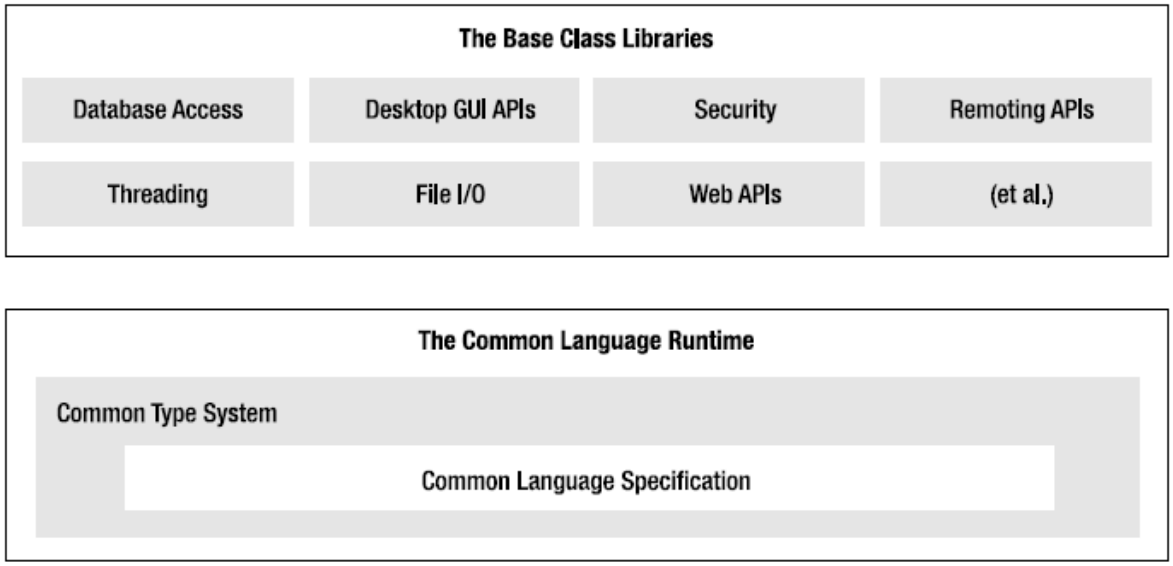


Εικόνα 1. Όλοι οι .NET Compilers.



Εικόνα 2. Το .NET Framework.

Το CTS ή Common Type System, αναλαμβάνει να περιγράψει όλους τους πιθανούς τύπους δεδομένων και προγραμματιστικών blocks που υποστηρίζονται από το runtime. Περιγράφει το πώς αυτές οι οντότητες μπορούν να αλληλεπιδράσουν μεταξύ τους και λεπτομέρειες όπως για το πώς εμφανίζονται στο format των .NET metadata.

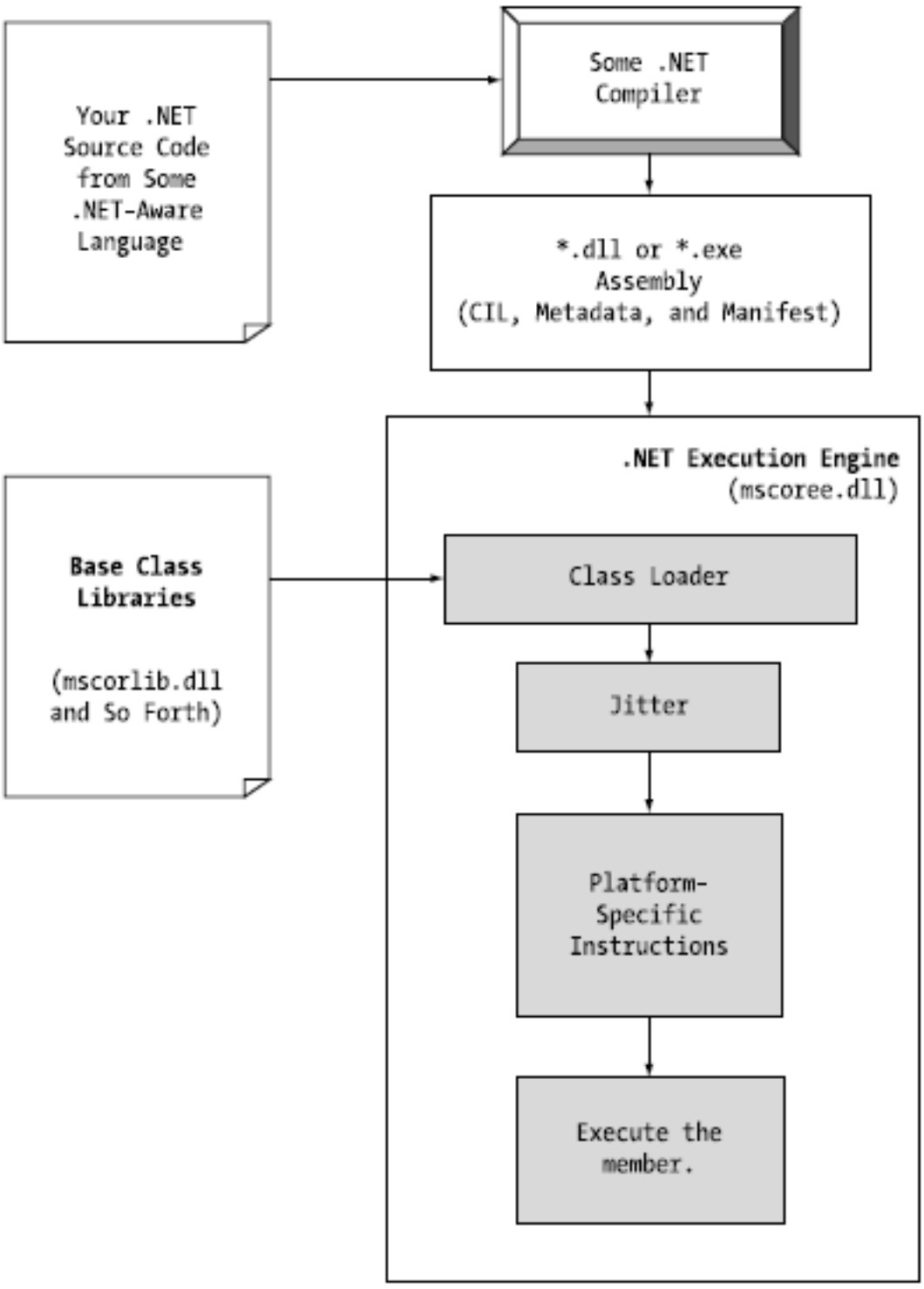


Εικόνα 3. Η σχέση μεταξύ CLR, CTS, CLS και της βασικής βιβλιοθήκης κλάσεων.

Το CLS ή Common Language Specification, καθορίζει ένα υποσέτ από κοινούς τύπους δεδομένων και προγραμματιστικών block στο οποίο όλες οι .NET γλώσσες μπορούν να «συμφωνήσουν».

1.2 Ο ρόλος των βασικών βιβλιοθηκών κλάσεων.

Η πλατφόρμα .NET, μαζί με προδιαγραφές των CLR, CTS/CLS, παρέχει μια βιβλιοθήκη βασικών κλάσεων, που είναι διαθέσιμη για όλες τις .NET γλώσσες προγραμματισμού. Αυτή η base class library, όχι μόνο ενθυλακώνει βασικά πρωτεύοντα στοιχεία όπως νήματα, είσοδος/έξοδος αρχείων, rendering γραφικών και αλληλεπίδραση μεταξύ διαφόρων συσκευών hardware, αλλά παρέχει μια σειρά από υπηρεσίες που χρειάζονται στις περισσότερες εφαρμογές του πραγματικού κόσμου.

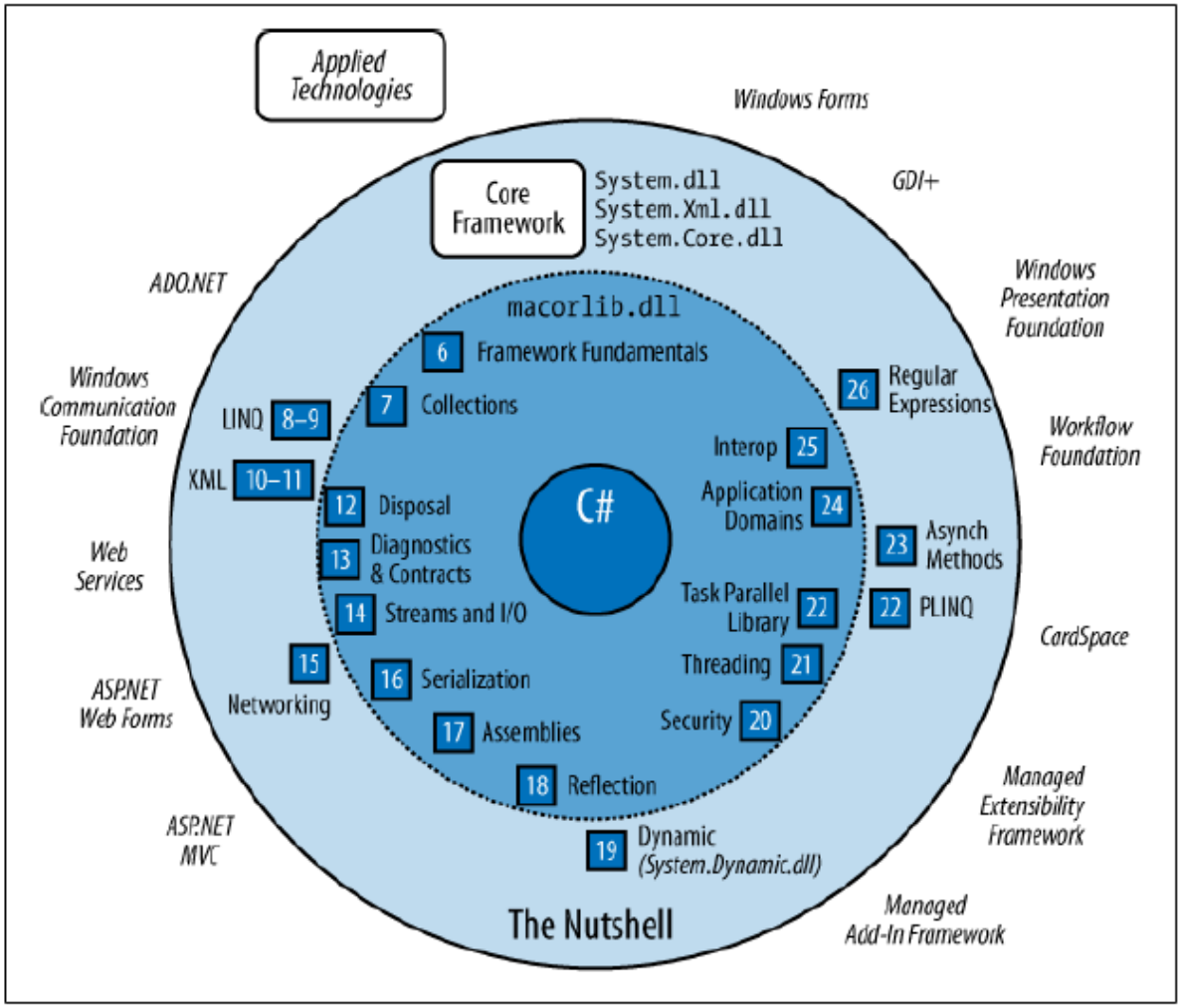


Εικόνα 4. Οι Compiler και βασικές βιβλιοθήκες κλάσεων.

Για παράδειγμα, οι base class libraries, καθορίζουν τους τύπους που έχουν πρόσβαση σε βάσεις δεδομένων, τη διαχείριση αρχείων XML, προγραμματιστική ασφάλεια και τη δημιουργία εφαρμογών web, παραδοσιακών desktop αλλά και απλών console applications.

1.3 Το CLR ή Common Language Runtime.

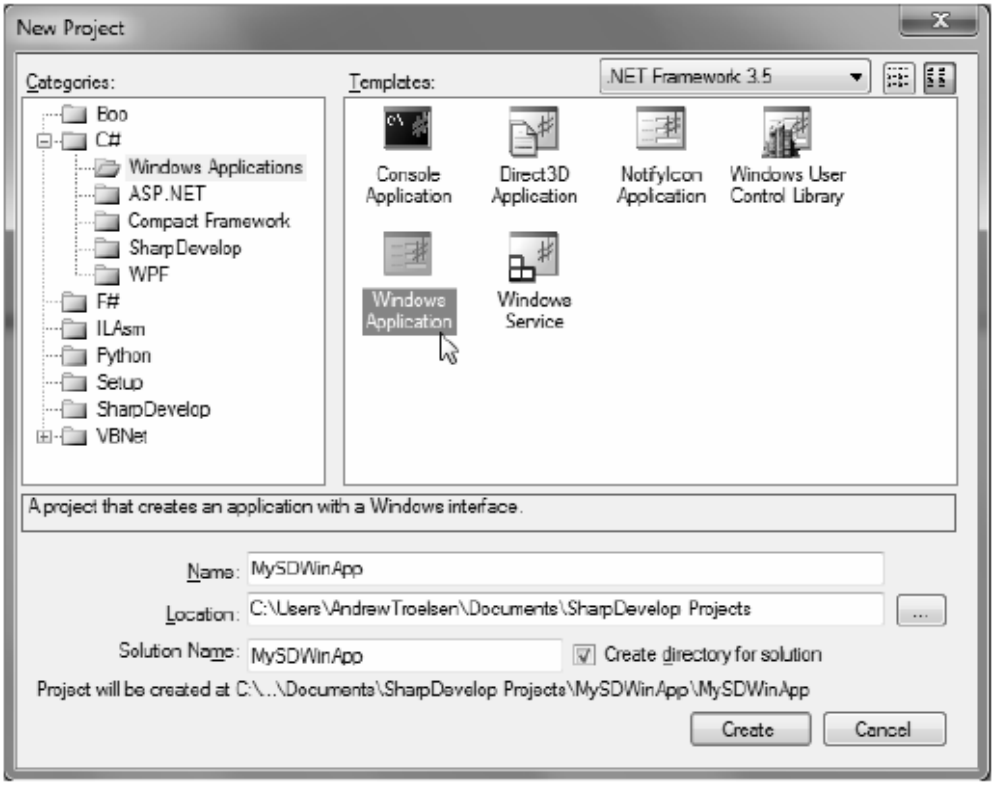
Το CLR ή Common Language Runtime, είναι το επίπεδο στο οποίο εμπεριέχεται το περιβάλλον εκτέλεσης του προγράμματος. Ο πρωταρχικός σκοπός του CLR είναι ο εντοπισμός, το loading και η διαχείριση των διάφορων data type της .NET. Το CLR επίσης αναλαμβάνει τη διαχείριση διαφόρων εργασιών χαμηλού επιπέδου, δηλαδή κοντά στη γλώσσα μηχανής, όπως διαχείριση μνήμης, hosting εφαρμογών, διαχείριση των threads και διάφορες λειτουργίες ασφάλειας.



Εικόνα 5. C# και .NET.

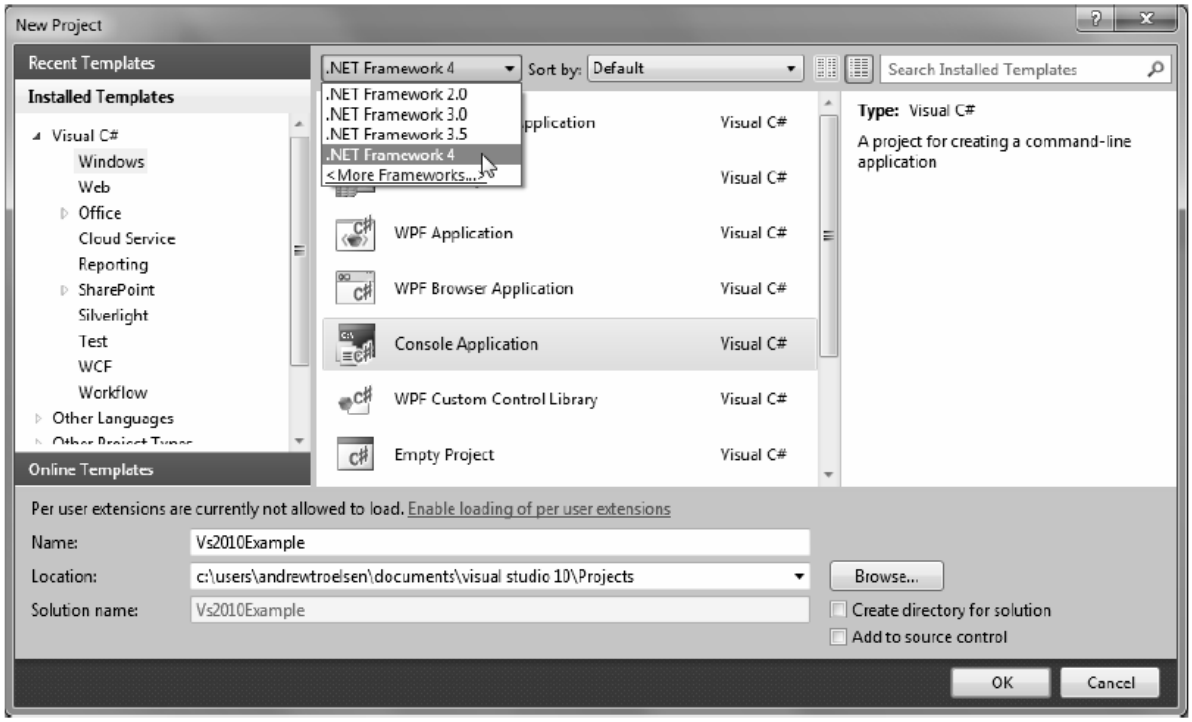
1.4 Τι παρέχει η C#

Η C# είναι μια γλώσσα προγραμματισμού, της οποίας η βασική σύνταξη μοιάζει αρκετά με τη σύνταξη της Java. Και οι δυο γλώσσες έχουν κοινές ρίζες (C, Objective C, C++).



Εικόνα 6. Το παράθυρο δημιουργίας νέου project του Visual Studio C# 2010.

Στην πραγματικότητα, πολλά από τα συντακτικά μέλη της C# προέρχονται από την Visual Basic 6.0 και την C++. Επειδή λοιπόν η C# είναι ένα υβρίδιο από πολλές γλώσσες, το τελικό αποτέλεσμα είναι ένα προϊόν το οποίο έχει σχεδόν την ίδια ευελιξία με την C++ και την απλότητα της Visual Basic.

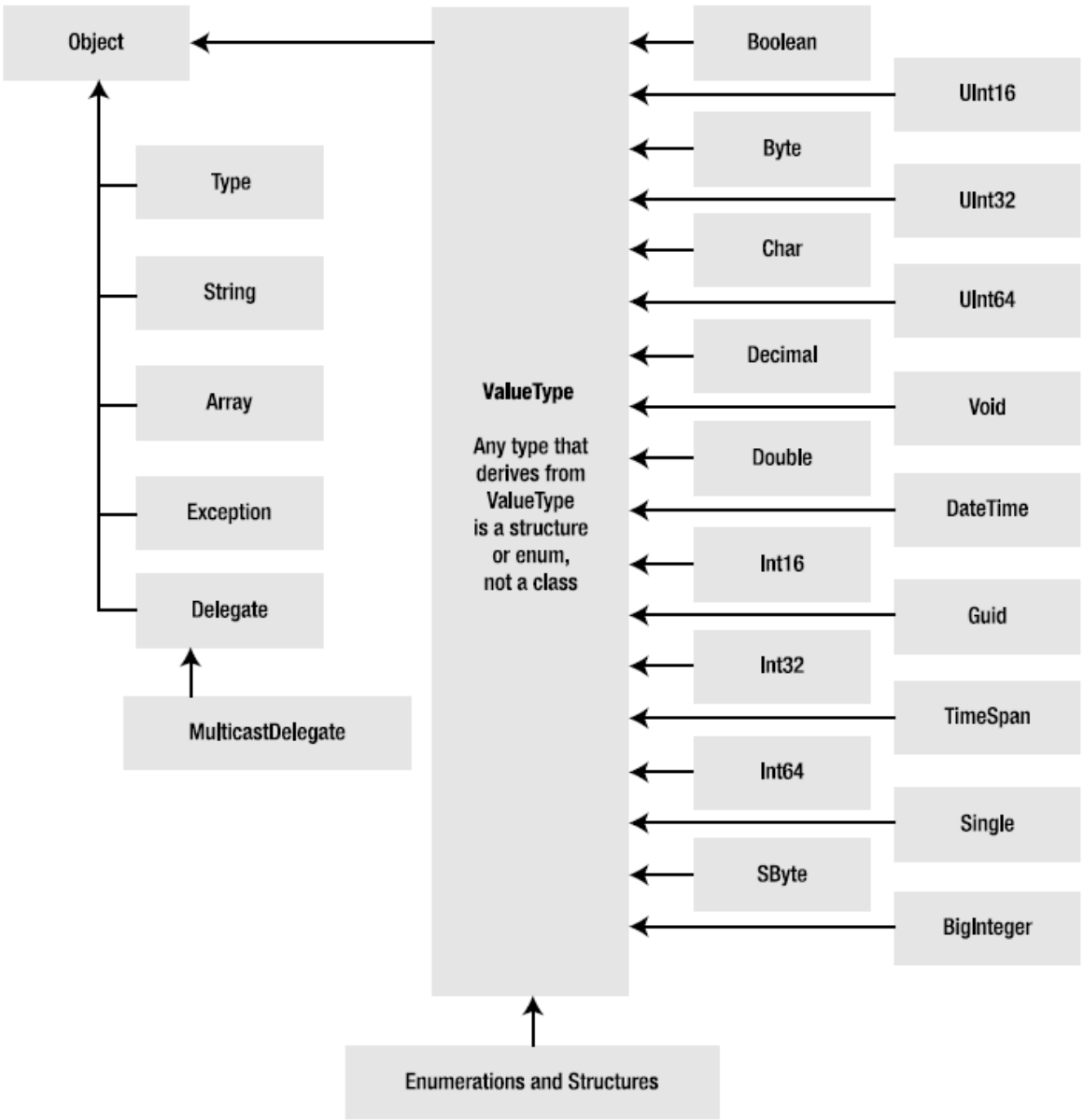


Εικόνα 7. Το Visual Studio 2010 επιτρέπει επιλογή συγκεκριμένου .NET Framework.

Χαρακτηριστικά της C# για όλες τις υπάρχουσες εκδόσεις της:

- Δεν απαιτείται η χρήση pointers. Τυπικά δεν είναι απαραίτητη η χρήση τους αν και υπάρχει η δυνατότητα χρησιμοποίησής τους.
- Αυτόματη διαχείριση της μνήμης μέσω του Garbage Collection.
- Επίσημες συντακτικές μορφές για classes, interfaces, structures, enumerations και delegates.
- Η ικανότητα της να κάνει overload custom operators χωρίς την πολυπλοκότητα της C++.

1.5 Τύποι Δεδομένων



Εικόνα 8. Η ιεραρχία κλάσεων των τύπων συστήματος.

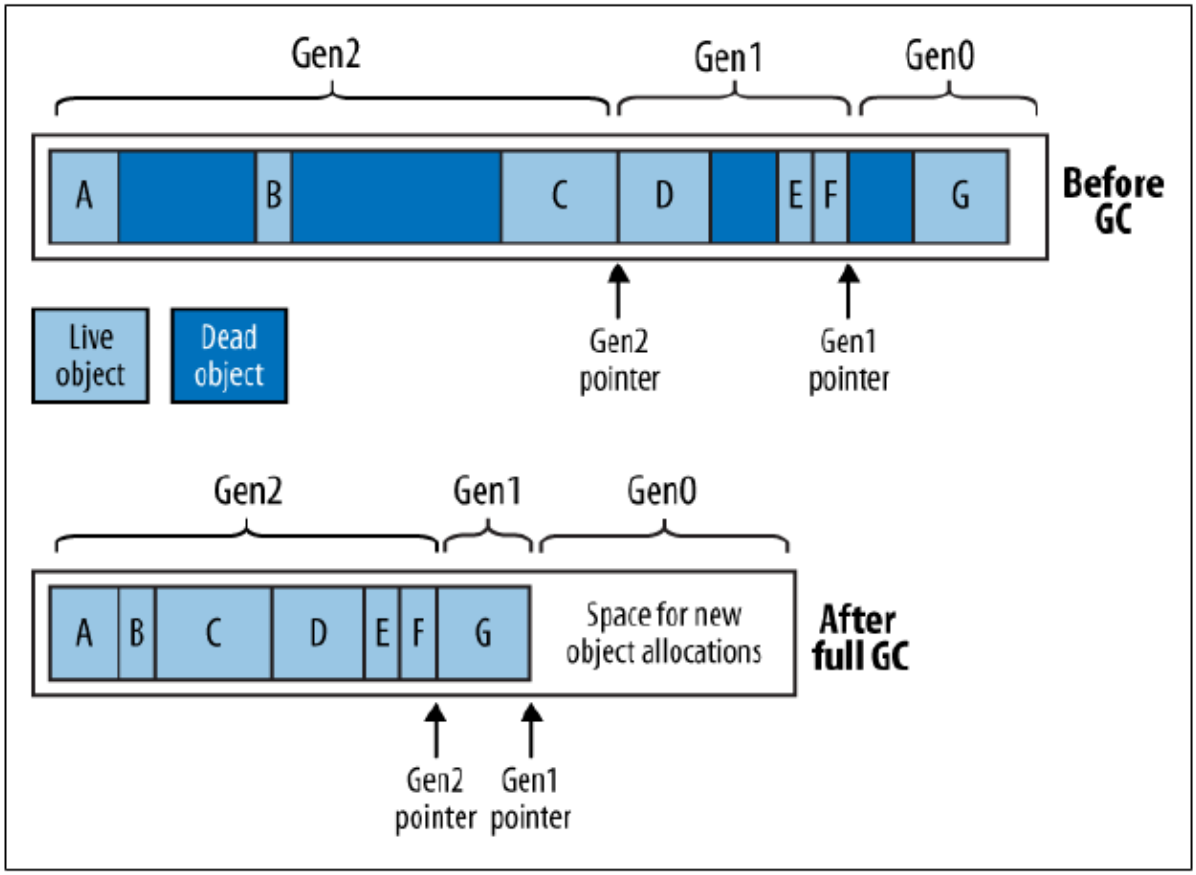
1.6 Χρησιμοποιώντας τον Garbage Collector

Κατά το runtime, μπορούμε να δηλώσουμε ένα αντικείμενο όσες φορές θέλουμε. Αυτό όμως έχει επίδραση στη ζωή του αντικειμένου. Θα παραμείνει στη μνήμη μέχρις ότου όλες οι αναφορές σε αυτό εξαφανιστούν. Για αυτό τον λόγο υπάρχει ο garbage collector.

- Ο Garbage Collector είναι υπεύθυνος για την καταστροφή των αντικειμένων.
- Κάθε αντικείμενο και οι destructors τους φορτώνονται στην μνήμη κατά το run time. Όταν τελειώσει η εκτέλεση του προγράμματος, όλα τα αντικείμενα θα καταστραφούν.
- Κάθε αντικείμενο θα καταστραφεί μόνο μια φορά.
- Κάθε αντικείμενο καταστρέφεται όταν δεν υπάρχουν πλέον αναφορές προς το αντικείμενο αυτό.

1.7 Πότε συμβαίνει το Garbage Collection?

Επειδή το Garbage Collection είναι απαιτητική διαδικασία, συμβαίνει μόνο όταν χρειάζεται, π.χ. όταν διαπιστώνεται ότι η διαθέσιμη μνήμη αρχίζει και περιορίζεται και τότε απελευθερώνει όση περισσότερη μνήμη μπορεί. Παρ' όλα αυτά μπορούμε να καλέσουμε τον Garbage collector με την μέθοδο Collect που υπάρχει στο System namespace. Η μέθοδος System.GC.Collect ξεκινάει τον Garbage Collector αλλά δεν περιμένει να τελειώσει η διαδικασία διότι η μέθοδος αυτή δουλεύει ασύγχρονα και αποφασίζεται από το runtime ο καλύτερος χρόνος για την συλλογή της μνήμης.



Εικόνα 9. Γενιές heap και ο garbage collector.

1.8 Πως λειτουργεί ο garbage collector?

Ο Garbage Collector τρέχει στο δικό του thread και μπορεί να εκτελεστεί μόνο κατά συγκεκριμένες χρονικές στιγμές, τυπικά όταν η εφαρμογή φτάνει στο τέλος μιας μεθόδου. Ενώ τρέχει, άλλες εφαρμογές πιθανό να σταματήσουν προσωρινά. Αυτό γίνεται διότι ο Garbage Collector ίσως χρειαστεί να μετακινήσει objects στη μνήμη και να αναβαθμίσει αναφορές αντικειμένων, κάτι που δεν μπορεί να γίνει εάν τα αντικείμενα χρησιμοποιούνται.

Τα βήματα που ακολουθεί ο Garbage Collector είναι:

- Δημιουργία ενός χάρτη με όλα τα reachable αντικείμενα. Αυτό γίνεται ακολουθώντας τα πεδία αναφορών μέσα στα αντικείμενα. Ο Garbage Collector χτίζει αυτό το χάρτη πολύ προσεκτικά και σιγουρεύει ότι οι κυκλικές αναφορές δεν προκαλούν loop επ' άπειρον. Κάθε αντικείμενο που δεν περιέχεται σε αυτό τον χάρτη θεωρείται μη προσβάσιμο.
- Ελέγχει αν οποιαδήποτε από τα μη προσβάσιμα αντικείμενα έχουν destructor που θα πρέπει να εκτελεστεί (μια διαδικασία που συνήθως ονομάζεται finalization). Κάθε unreachable object που χρειάζεται finalization τοποθετείται σε μια ειδική ουρά που ονομάζεται f-reachable queue.
- Ανακατανέμει τα εναπομείναντα μη προσβάσιμα αντικείμενα (αυτά που δεν χρειάζονται finalization) μεταφέροντας τα προσβάσιμα αντικείμενα στο τέλος της σειράς στη μνήμη, προκαλώντας έτσι την ανασυγκρότηση την μνήμης και την ελευθέρωσή της στην κορυφή της στοίβας. Επίσης όταν ο Garbage Collector μετακινεί ένα reachable αντικείμενο, επίσης αναβαθμίζει όποιες αναφορές σε αυτό.
- Σε αυτό το σημείο, ο Garbage Collector επιτρέπει τα άλλα νήματα να συνεχίσουν την εκτέλεσή τους.
- Προκαλεί finalization στα unreachable αντικείμενα που χρειάζονται ελευθέρωση (αυτά που βρίσκονται στην f-reachable queue), μέσω του δικού του thread.

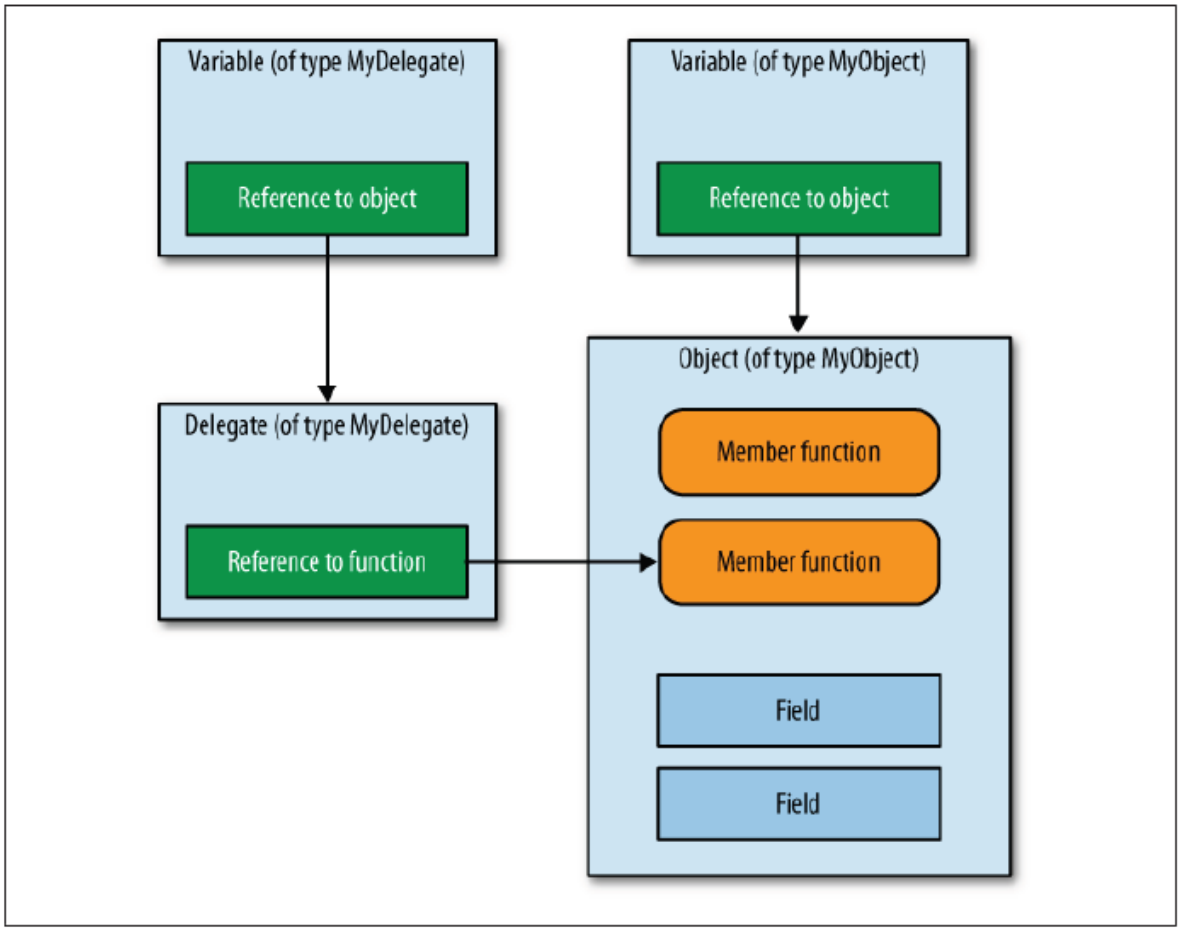
Πίνακας 1. System.GC

Μέλος System.GC	Περιγραφή
AddMemoryPressure() **RemoveMemoryPressure()**	Επιτρέπει στον καθορισμό μιας αριθμητικής αξίας που αναπαριστά το πόσο επείγον είναι το κάθε στοιχείο κατά το Garbage Collection
Collect()	Αναγκάζει τον Garbage Collector να κάνει το collection. Αυτή η μέθοδος μπορεί επίσης να καθορίζει και τον τρόπο περισυλλογής (μέσω του GC.CollectionMode)
CollectionCount()	Επιστρέφει μια αριθμητική αξία που αναπαριστά το πόσες φορές έχει «καθαριστεί» μια συγκεκριμένη γενιά από objects.
GetGeneration()	Επιστρέφει τη γενιά στην οποία ανήκει τη συγκεκριμένη χρονική στιγμή ένα αντικείμενο
GetTotalMemory()	Επιστρέφει την κατά προσέγγιση ποσότητα μνήμης (σε

	byte) που εμπεριέχεται την τρέχουσα στιγμή στη στοίβα μνήμης που είναι υπό διαχείριση. Μια παράμετρος τύπου Bool, καθορίζει το αν η επιστροφή της κλήσης θα γίνει ή μετά το Garbage Collection
MaxGeneration	Επιστρέφει τον αριθμό των μέγιστων γενιών που υποστηρίζονται από το συγκεκριμένο σύστημα. Υπό την .NET version 4 της Microsoft, υποστηρίζονται έως 3 γενιές: 0, 1 και 2.
SupressFinalize()	Θέτει μια «σημαία» η οποία υποδεικνύει ότι το συγκεκριμένο αντικείμενο δεν πρέπει να κληθεί από την μέθοδο finalize() .
WaitForPendingFinalizers()	Αναστέλλει το συγκεκριμένο thread μέχρι όλα τα finalizable αντικείμενα να έχουν γίνει finalized. Αυτή η μέθοδος τυπικά καλείται ακριβώς μετά την κλήση της GC.Collect().

1.9 Delegates

Delegate είναι οι αντίστοιχοι type-safe C-style δείκτες αλλά για την .NET. Η κύρια διαφορά τους είναι ότι το delegate είναι μια κλάση που πηγάζει από το System.MulticastDelagate, παρά απο ένα απλό δείκτη σε μια θέση μνήμης. Στην C# χρησιμοποιούνται μέσω της λέξης κλειδί “delegate”.Πχ *delegate int BinaryOp(int x, int y);*

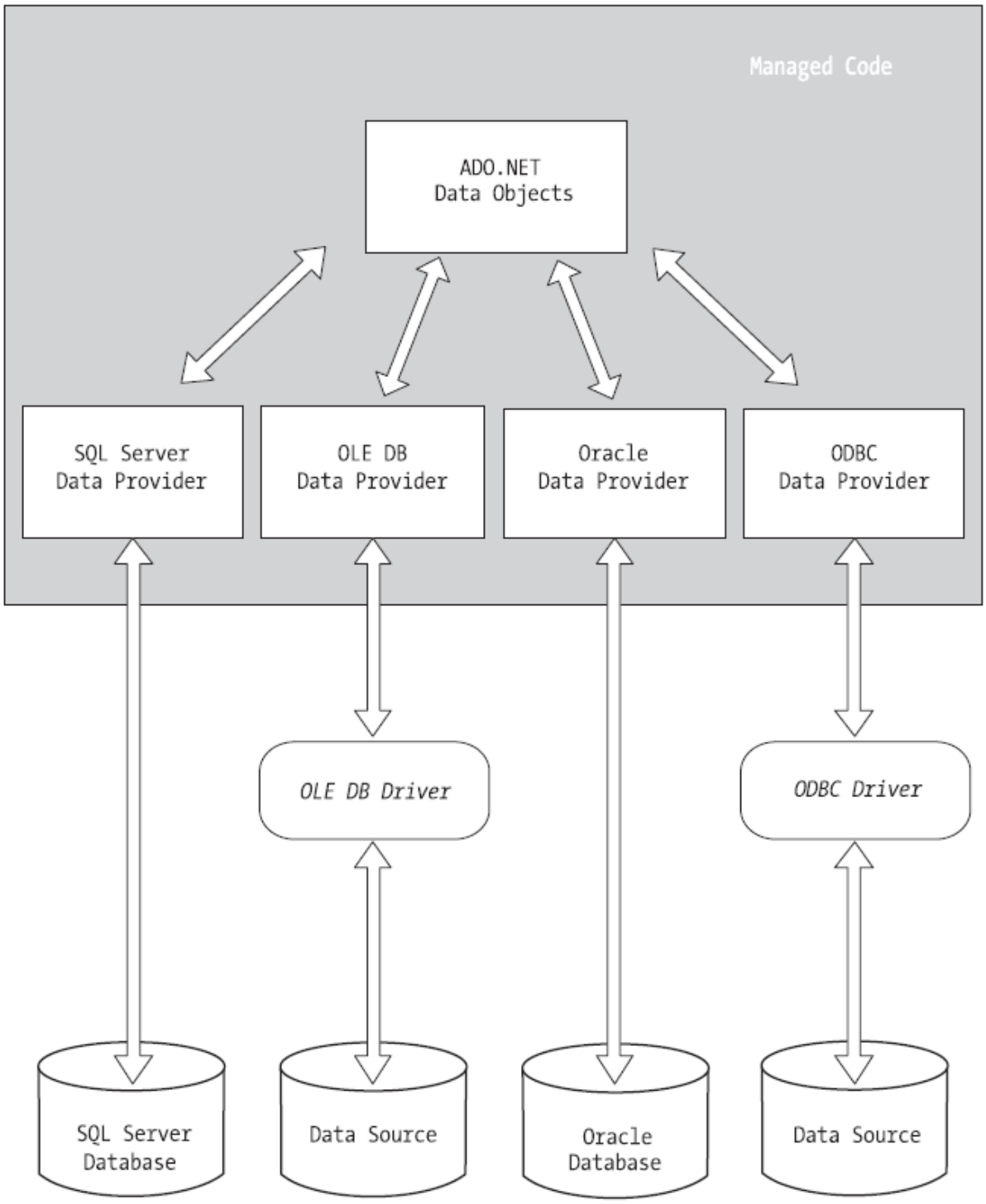


Εικόνα 10. Delegates και μεταβλητές.

Τα delegates είναι χρήσιμα για να παρέχουν ένα τρόπο επικοινωνίας μεταξύ δύο διαφορετικών οντοτήτων και παρέχουν τα θεμέλια για την αλληλο-επικοινωνία τους μέσα στην .NET αρχιτεκτονική. Συνήθως παρέχουν υποστήριξη για multicasting και για κλήση ασύγχρονων μεθόδων.

1.10 Βάσεις Δεδομένων

Η C# παρέχει την δυνατότητα για σύνδεση με πολλές βάσεις δεδομένων. Θα εξετάσουμε μια συγκεκριμένη περίπτωση όμως.

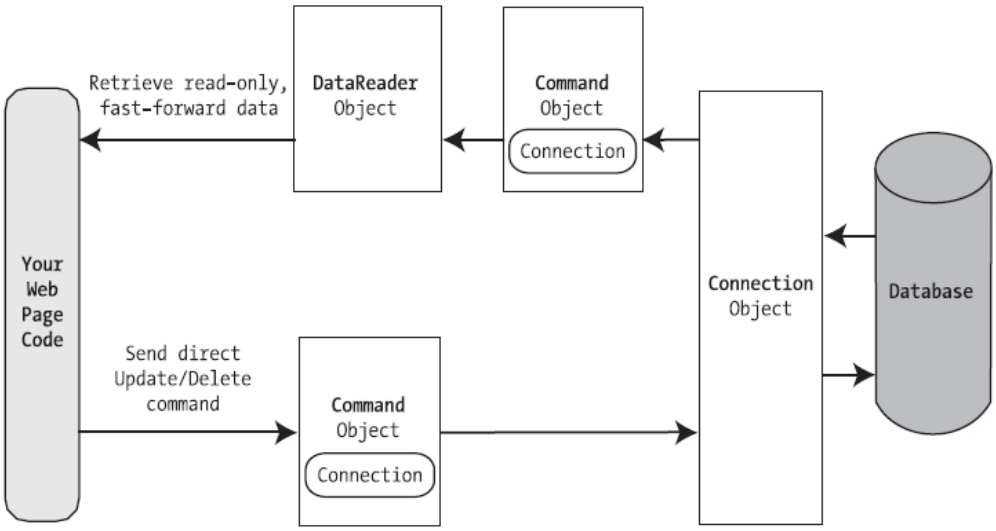


Εικόνα 11. Τα επίπεδα μεταξύ του κώδικα και των βάσεων δεδομένων.

Για την σύνδεση σε έναν Microsoft SQL server 2008, χρειάζεται να γίνει χρήση των SQLClient libraries:

- Πρώτα απ'όλα θα πρέπει να προστεθεί στο namespace του προγράμματος: Using System.Data.SqlClient;

- Μετά θα χρειαστεί να δημιουργηθεί ένα νέο instance του SqlConnection object: SqlConnection myConnection = new SqlConnection();



Εικόνα 12. Άμεση πρόσβαση στα δεδομένα με ADO.NET.

- Το SqlConnection παίρνει ως argument ένα string, το connection string, το οποίο περιγράφει τις ιδιότητες του SQL Server. Δηλαδή, το όνομα του server, το είδος της σύνδεσης, όνομα χρήστη, κωδικός, ονομασία της προεπιλεγμένης βάσης δεδομένων καθώς και το connection timeout. Στη συγκεκριμένη περίπτωση:

```
SqlConnection myConnection = new SqlConnection("user=" + dbUserName + ";" +
"password=" + dbPassword + ";server=" + dbIp + ";" + "Trusted_Connection=no;" +
"database=test_db; " + "connection timeout=3");
```

- Ορίζουμε ένα query command π.χ.:

```
SqlCommand myCommand = new SqlCommand("INSERT INTO test_db.dbo.test (test1,
test2) VALUES ('1', '2')", myConnection);
```

- Η σύνδεση με τη βάση δεδομένων θα επιτευχθεί με την εξής εντολή τελικά:

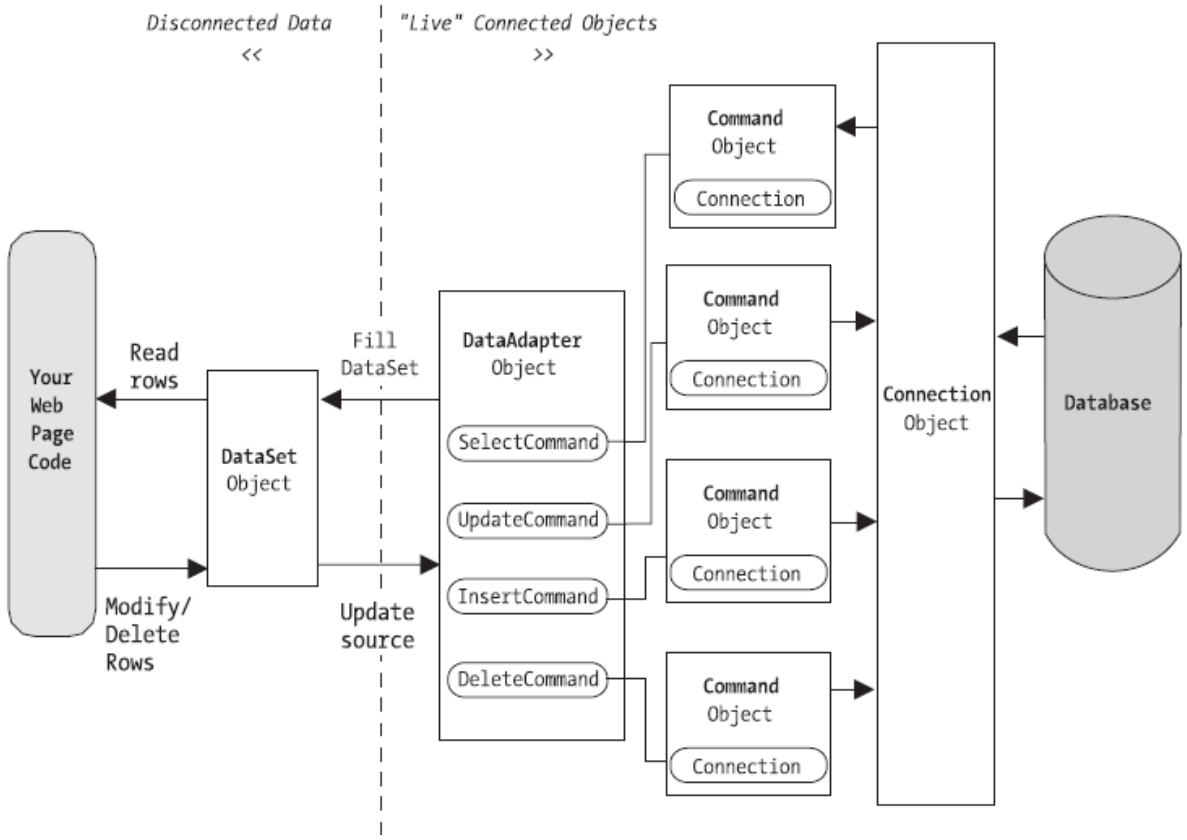
```
myCommand.Connection.Open();
```

- Έπειτα εκτελούμε το query:

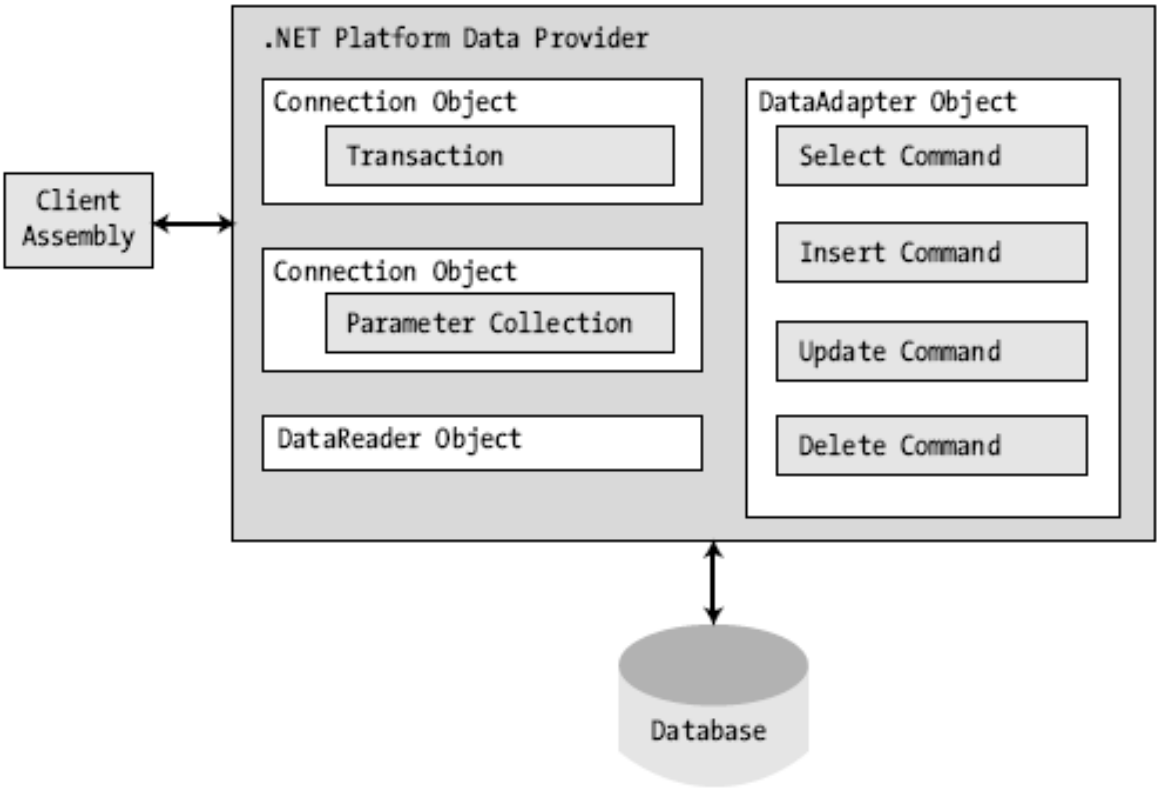
```
myCommand.ExecuteNonQuery();
```

- Αφού λάβουμε τα επιθυμητά αποτελέσματα από τη βάση δεδομένων, θα χρειαστεί να κλείσει το connection με τη βάση δεδομένων:

```
myConnection.Close();
```



Εικόνα 13. Χρησιμοποιώντας DataSet με ADO.NET.



Εικόνα 14. Οι ADO.NET παροχείς δεδομένων για πρόσβαση σε συγκεκριμένο DBMS.

ΚΕΦΑΛΑΙΟ 2ο

ΔΙΚΤΥΑ ΚΑΙ SOCKETS

2.1 Εισαγωγή στα Networks – sockets

Εκατομμύρια υπολογιστών, σε όλο τον κόσμο, είναι αυτή την στιγμή συνδεμένοι στο παγκόσμιο δίκτυο, γνωστό ως Internet. Το Internet επιτρέπει την εκτέλεση προγραμμάτων σε υπολογιστές που απέχουν χιλιάδες χιλιόμετρα μεταξύ τους και μπορούν να επικοινωνούν και να ανταλλάσουν πληροφορίες απρόσκοπτα. Αν έχετε έναν υπολογιστή συνδεδεμένο σε ένα δίκτυο, αναμφισβήτητα έχετε χρησιμοποιήσει έναν Web Browser - ένα τυπικό πρόγραμμα που κάνει χρήση του Internet. Τι κάνει ένα τέτοιο πρόγραμμα ώστε να επικοινωνεί μέσω ενός δικτύου? Η απάντηση εξαρτάται από την εφαρμογή και απο το λειτουργικό σύστημα (Operating System), αλλά ένα μεγάλο πλήθος προγραμμάτων αποκτούν πρόσβαση σε υπηρεσίες δικτυακής επικοινωνίας μέσω της διασύνδεσης προγραμματισμού εφαρμογών των "sockets" (Application Programming Interface).

Πριν πάμε στις λεπτομέρειες του Socket API, αξίζει να ρίξουμε μια ματιά στην ολική εικόνα των δικτύων και πρωτοκόλλων και να δούμε που ταιριάζει το Application Programming Interface του TCP/IP. Ο στόχος είναι η εισαγωγή σε βασικούς όρους και έννοιες.

2.2 Τι είναι δίκτυο

Δίκτυο είναι ένα σύνολο συσκευών (υπολογιστών, εκτυπωτών, τερματικών, δορυφόρων,…) συνδεδεμένων μεταξύ τους με κανάλια επικοινωνίας (φυσικές συνδέσεις) τα οποία μπορούν να παράγουν, να στέλνουν, να προωθούν και να λαμβάνουν πληροφορίες (απλά δεδομένα, ήχος, βίντεο, εικόνα…).

Σκοποί:

1. Καταμερισμός πόρων (υλικού – λογισμικού)
2. Υψηλή αξιοπιστία (ασφάλεια)
3. Ταχύτητα επίλυσης προβλημάτων
4. Επικοινωνία – Συνεργασία χρηστών
5. Περιορισμός κόστους

Εφαρμογές Δικτύων:

1. Τηλεπικοινωνίες (σταθερή/κινητή τηλεφωνία, καλωδιακή τηλεόραση, ηλεκτρονικό ταχυδρομείο, τηλεγραφία, τηλεδιάσκεψη)
2. Οικονομικές Υπηρεσίες
3. Βιομηχανική παραγωγή (π.χ. κατασκευές)
4. Πωλήσεις – Marketing –Διαφήμιση
5. Υπηρεσίες καταλόγου …

2.3 Δομή δικτύων

Όπως έχει αναφερθεί ένα δίκτυο αποτελείται από διάφορες συσκευές (hardware – δορυφόροι, δρομολογητές, τερματικά, γέφυρες, …). Οι συσκευές αυτές συνδέονται

μεταξύ τους με γραμμές μετάδοσης (links). Υπάρχουν στη βιβλιογραφία πολλές ορολογίες που αφορούν τα παραπάνω στοιχεία υλικού με αποτέλεσμα να υπάρχει κάποια σύγχυση. Παρακάτω γίνεται μια προσπάθεια ταξινόμησης του hardware σε διάφορες κατηγορίες:

- Κεντρικοί Υπολογιστές (hosts): Παίζουν το ρόλο του πομπού ή του δέκτη. Στη βιβλιογραφία αναφέρονται και ως τερματικά συστήματα (end systems). Τέτοιες συσκευές μπορεί να είναι είτε προσωπικοί υπολογιστές (PCs), είτε διάφορα κεντρικά, ισχυρά υπολογιστικά συστήματα (Mainframes) που λειτουργούν ως εξυπηρετητές (servers) πολλών τελικών χρηστών. Πολλά τέτοια συστήματα είναι γνωστά και ως: Τερματικές Συσκευές Ευρείας Ζώνης Εκπομπής (Broadband Terminal Equipment).
- Γραμμές Μετάδοσης (Transmission Lines, Links): Πρόκειται για τα φυσικά μονοπάτια επικοινωνίας διαμέσου των οποίων μεταφέρονται τα δεδομένα από τη μια συσκευή στην άλλη. Ένα τέτοιο μονοπάτι συνδέει δύο (point-to-point line configuration) ή περισσότερες (multipoint line configuration) συσκευές και μπορεί να παρέχει σε αυτές είτε μονόδρομη (simplex), είτε αμφίδρομη όχι ταυτόχρονη (half-duplex), είτε αμφίδρομη και ταυτόχρονη (full-duplex) επικοινωνία.
- Στοιχεία Μεταγωγής (Switching Elements): Πρόκειται για τις ενδιάμεσες συσκευές που συνδέουν τις γραμμές μετάδοσης και επιφορτίζονται με το έργο της δρομολόγησης των δεδομένων από τη μια γραμμή στην άλλη ή από το ένα δίκτυο στο άλλο. Αυτά τα στοιχεία είναι γνωστά με διάφορες ορολογίες όπως: Επεξεργαστές Μηνύματος Διασύνδεσης (Interface Message Processors - IMPs), Κόμβοι Μεταγωγής Πακέτων (Packet Switch Nodes), Ενδιάμεσα Συστήματα (Intermediate Systems) και Κέντρα Μεταγωγής Δεδομένων (Data Switching Exchange). Σε αυτήν την κατηγορία ανήκουν οι Γέφυρες (Bridges), οι μεταγωγείς ATM, οι Δρομολογητές (Routers) για διαδικτυακή επικοινωνία, οι Κυψέλες (cells) κινητής τηλεφωνίας και οι δορυφόροι.

Οι κυριότερες συσκευές θα αναλυθούν σε ξεχωριστή παράγραφο, αφού προηγουμένως γίνει η μελέτη της αρχιτεκτονικής δικτύων, έτσι ώστε να δοθεί μια γενική εικόνα για το πώς (δηλαδή σε τι επίπεδο) λειτουργούν αυτές.

Το λογισμικό (Software) που χρησιμοποιείται στα δίκτυα, πραγματοποιεί τις απαιτούμενες λειτουργίες (όπως δρομολόγηση, έλεγχο σφαλμάτων, τμηματοποίηση πληροφορίας, κρυπτογράφηση πληροφορίας,…). Το σύνολο του λογισμικό μαζί με τους κανόνες λειτουργίας των δικτύων αποτελούν τα πρωτόκολλα. Τα πρωτόκολλα είναι ιεραρχημένα σύμφωνα με την αρχιτεκτονική των δικτύων.

2.4 Το πρότυπο OSI της ISO

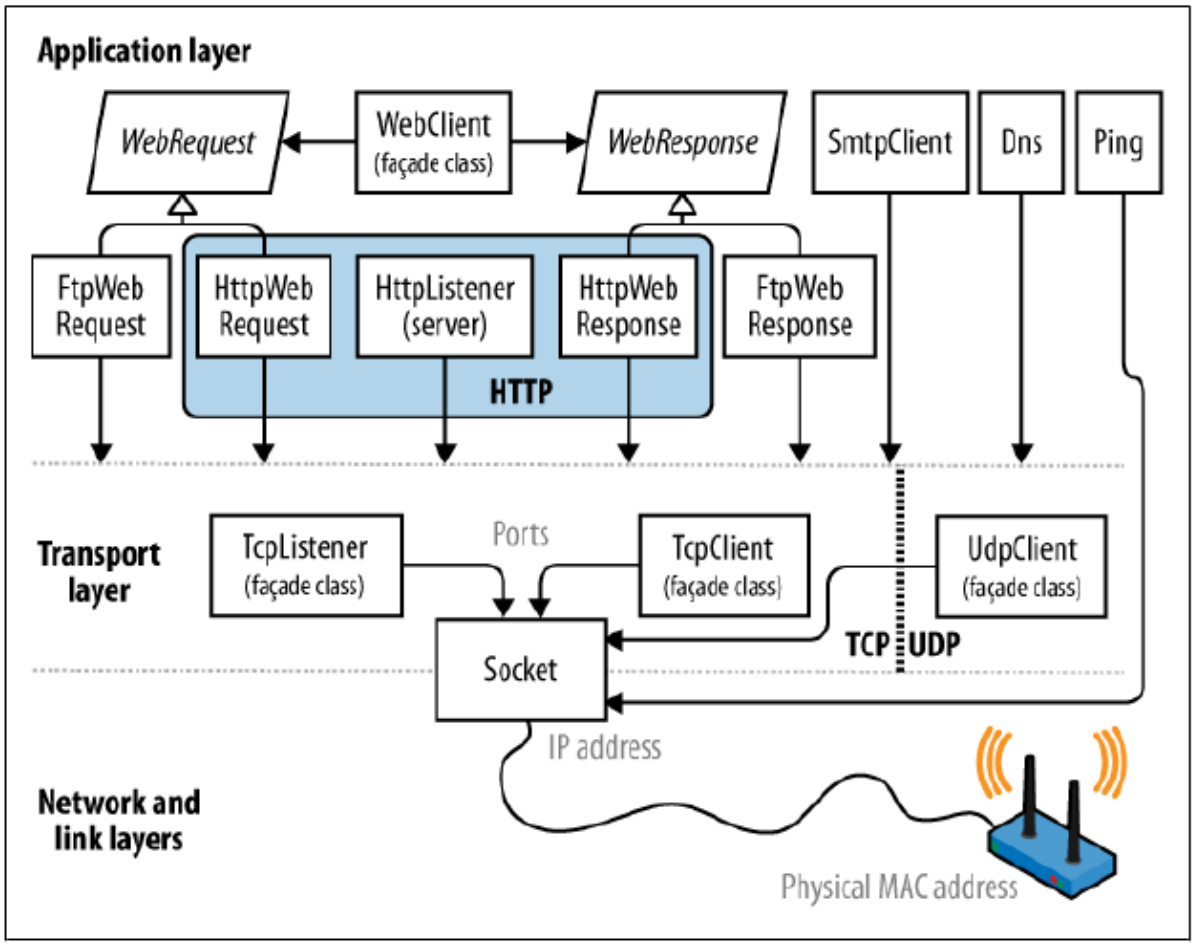
Η προσπάθεια για τη μοντελοποίηση των δικτύων οδήγησε στην ανάπτυξη του προτύπου OSI (Open Systems Interconnection) από τον οργανισμό ISO, το οποίο θεωρείται ως το πλέον αντιπροσωπευτικό μοντέλο δικτύου. Η βασική ιδέα που οδήγησε στην ανάπτυξη του μοντέλου αυτού είναι η ομαδοποίηση και ταξινόμηση όλων των λειτουργιών που εκτελούνται σε ένα δίκτυο σε διάφορα επίπεδα, ανάλογα με τη φύση και το είδος της εξυπηρέτησης που προσφέρουν οι λειτουργίες αυτές. Τα επίπεδα αυτά είναι 7 και παρουσιάζονται παρακάτω (σε μια top-down προσέγγιση). Το κάθε ένα από

τα 6 κατώτερα επίπεδα προσφέρει τις υπηρεσίες του στο ανώτερό του. Οι υπηρεσίες και οι πληροφορίες που παρέχονται στο ανώτερο επίπεδο προσδιορίζονται από διασυνδέσεις (Interfaces) ανάμεσα στα επίπεδα.

Οι συσκευές του δικτύου εκτελούν τις λειτουργίες ενός αριθμού επιπέδων του OSI (π.χ. οι περισσότεροι ενδιάμεσοι κόμβοι των τριών πρώτων επιπέδων). Ανάμεσα στις διάφορες συσκευές, κάποιες διεργασίες από αντίστοιχα επίπεδα επικοινωνούν μεταξύ τους (peer-to-peer processes) με βάση κάποιους κανόνες και διατάξεις. Τα σύνολα αυτών των κανόνων σχηματίζουν τα πρωτόκολλα των επιπέδων.

2.5 Αρχιτεκτονική TCP/IP

Η αρχιτεκτονική TCP/IP χρησιμοποιήθηκε σε πειραματικό στάδιο σε δίκτυα μεταγωγής πακέτων (packet-switching networks) και η πιο σημαντική εφαρμογή της ήταν στο ARPANET. Σήμερα, έχει εξελιχθεί σε μεγάλο βαθμό και χρησιμοποιείται στις περισσότερες εφαρμογές δικτύων. Η λειτουργία του σημερινού Internet στηρίζεται πάνω σε αυτή την αρχιτεκτονική, ενώ η ανάπτυξη μιας σειράς επιπλέον συμβατών πρωτοκόλλων για δίκτυα (όπως τα δίκτυα ATM και MPLS) έχει οδηγήσει σε ακόμα καλύτερα αποτελέσματα όσον αφορά τις επιδόσεις του Internet.



Εικόνα 15. Αρχιτεκτονική δικτύων.

Συγκριτικά με το πρότυπο OSI, η αρχιτεκτονική TCP/IP είναι απλούστερη όσο αφορά τον αριθμό των επιπέδων που χρησιμοποιεί. Περιλαμβάνει τα παρακάτω 5 επίπεδα λειτουργίας:

- Επίπεδο Εφαρμογής (Application Layer)
- Επίπεδο Μεταφοράς (Host-to-host, ή Transport Layer)
- Επίπεδο Διαδικτύου (Internet Layer ή Internetwork Layer)
- Επίπεδο Πρόσβασης Δικτύου (Network Access Layer)
- Φυσικό Επίπεδο (Physical Layer)

Από τον αριθμό των επιπέδων διαπιστώνεται ότι, κάποια από τα επίπεδα του TCP/IP καλύπτουν τις λειτουργίες των πλεοναζόντων επιπέδων του OSI. Πιο συγκεκριμένα το επίπεδο εφαρμογής του TCP/IP καλύπτει τόσο το αντίστοιχό του όσο και τα επίπεδα παρουσίασης και συνόδου, στο OSI. Κάποιες από τις λειτουργίες του επιπέδου συνόδου πραγματοποιούνται στο επίπεδο μεταφοράς του TCP/IP, στο οποίο κυριαρχεί η παρουσία του πρωτοκόλλου TCP (Transmission Control Protocol).

Κάτι που πρέπει να τονιστεί είναι ότι, η αρχιτεκτονική TCP/IP διαχωρίζει τη διαδικτυακή λειτουργία από τη λειτουργία σε επίπεδο δικτύου. Η πρώτη εκτελείται στο επίπεδο διαδικτύου, όπου το κυρίαρχο πρωτόκολλο είναι το IP (Internet Protocol). Το επίπεδο διαδικτύου επιφορτίζεται με το έργο της δρομολόγησης ανάμεσα στα διάφορα δίκτυα που παρεμβάλλονται ανάμεσα στην πηγή και τον προορισμό. Με το πώς θα κινηθούν τα πακέτα μέσα στα δίκτυα αυτά ασχολείται το επίπεδο πρόσβασης δικτύου, στο οποίο δεσπόζει η παρουσία του πρωτοκόλλου που κάθε φορά χρησιμοποιείται, ανάλογα με το δίκτυο (π.χ. ATM, Ethernet, X.25,…). Το επίπεδο πρόσβασης δικτύου ασχολείται και με τη μετακίνηση από το ένα άκρο της γραμμής στο άλλο (Node-to-node delivery).

Τέλος, το φυσικό επίπεδο του TCP/IP καλύπτει τις ίδιες λειτουργίες όπως και στο OSI.

2.6 Διευθύνσεις και Ports

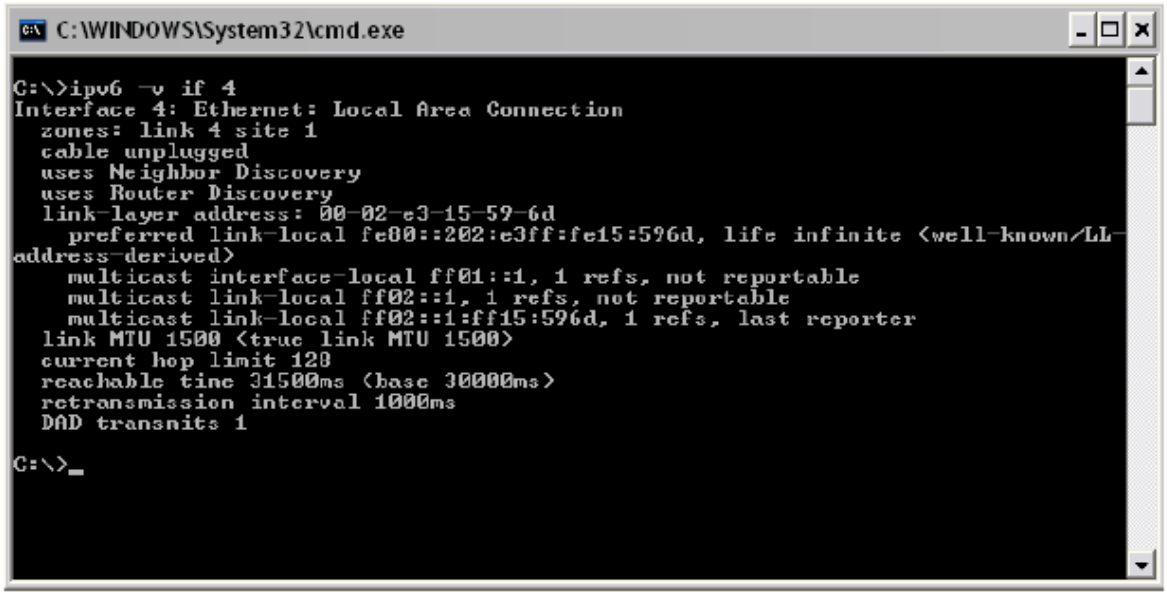
Για να λειτουργήσουν οι τηλεπικοινωνίες, ένας υπολογιστής ή μια συσκευή απαιτείται να έχει μια διεύθυνση. Στο internet χρησιμοποιούνται κυρίως δύο συστήματα διευθυνσιοδότησης:

- **IPv4**. Το κυρίαρχο σύστημα διευθυνσιοδότησης προς το παρόν. Οι διευθύνσεις IPv4 είναι μήκους 32bit. Σε μορφή string γράφονται ως 4 decimal αριθμούς από το 0-255, χωριζόμενοι από τελείες (πχ 102.101.234.252). Μια διεύθυνση μπορεί να είναι μοναδική στον κόσμο ή μοναδική σε ένα συγκεκριμένο subnet (τοπικό υποδίκτυο συνήθως)
- **IPv6.** Το καινούργιο σύστημα 128-bit διευθυνσιοδότησης. Οι διευθύνσεις σε μορφή string είναι σε μορφή hexadecimal και διαχωρίζονται από ":" (πχ. 3EA0:FFFF:E4A3:54F0:8D31:198A:41BC:FFF3)

IP Address Range	Number of Distinct Addresses
10.0.0.0 to 10.255.255.255	Up to 16 million computers (Class A)
172.16.0.0 to 172.31.255.255	900,000 computers (Class B)
192.168.0.0 to 192.168.255.255	65,000 computers (Class C)

Εικόνα 16. Οικογένειες ιδιωτικών IP.

Η κλάση IPAddress στο System.Net namespace αναπαριστά μια διεύθυνση και για τα δύο πρωτόκολλα. Έχει έναν constructor ο οποίος αποδέχεται πίνακα από byte και μια στατική Parse μέθοδο για να αποδέχεται string.



Εικόνα 17. Το εργαλείο IPv6 του MsDos.

Χρησιμοποιώντας το Dns

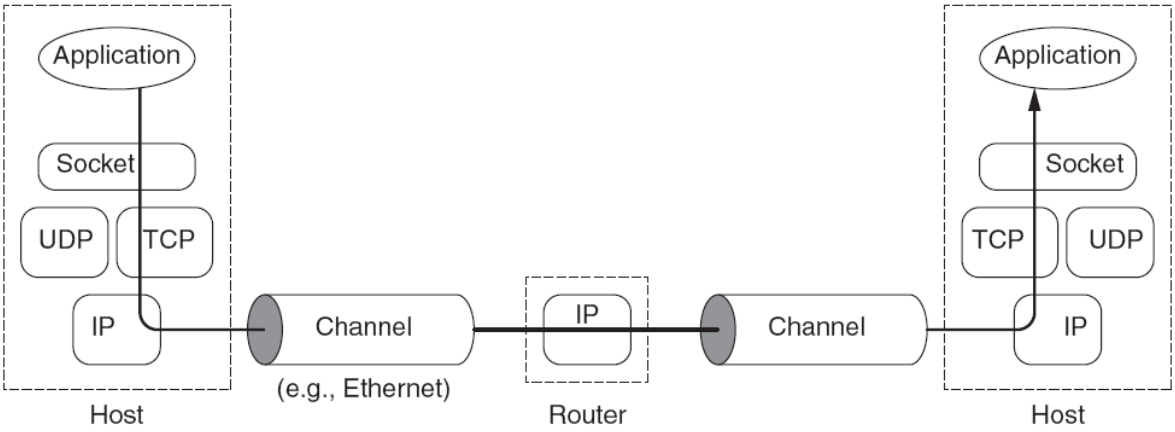
Η στατική κλάση Dns, ενθυλακώνει το Domain Name Service, το οποίο μετατρέπει «ωμές» Ip διευθύνσεις, όπως το 66.135.192.87 σε φιλικά προς τον άνθρωπο domain names, όπως ebay.com και αντίστροφα.

2.7 Δίκτυα, πακέτα και πρωτόκολλα

Ένα δίκτυο υπολογιστών αποτελείται από συσκευές που διασυνδέονται μεταξύ τους μέσω καναλιών επικοινωνίας. Αυτές συνήθως είναι hosts, routers και switches. Hosts είναι οι υπολογιστές που τρέχουν εφαρμογές όπως ο Web Browser, στην ουσία τα προγράμματα που τρέχουν στους hosts είναι και οι ίδιοι οι χρήστες του δικτύου.

Τα Routers είναι συσκευές που η δουλειά τους είναι να αναμεταδίδουν ή να προωθούν πληροφορίες από το ένα κανάλι επικοινωνίας στο άλλο. Μπορεί να εκτελούν προγράμματα και υπηρεσίες αλλά συνήθως δεν εκτελούν προγράμματα εφαρμογών/χρηστών. Το κανάλι επικοινωνίας (communication channel) είναι το μέσο

απο το οποίο μεταφέρονται οι ακολουθίες των byte από το ένα host στον άλλο. Μπορεί να είναι μια τεχνολογία broadcast-αναμετάδοσης όπως το ethernet, μια σύνδεση μέσω modem ή κάτι ποιο εξελιγμένο. Τα Routers είναι σημαντικές συσκευές διότι δεν είναι πρακτικό να συνδέεται κάθε host απευθείας με κάθε άλλο host. Γι' αυτό, μερικοί host συνδεδεμένοι σε ένα router, το οποίο συνδέεται με άλλα router σχηματίζουν ένα δίκτυο (network). Προγράμματα τα οποία ανταλλάσσουν πληροφορίες μέσα σε ένα δίκτυο, παρ' όλα αυτά, δεν επικοινωνούν κατευθείαν με τα router και γενικά δεν γνωρίζουν την ύπαρξη τους.



Εικόνα 18. Ένα TCP/IP δίκτυο.

2.8 Πληροφορίες - Πρωτόκολλα

Χρησιμοποιώντας την λέξη Πληροφορία, εννοούμε τις ακολουθίες των byte, που δημιουργούνται και ερμηνεύονται από τα προγράμματα. Στο πλαίσιο των δικτύων υπολογιστών, αυτές οι ακολουθίες από byte γενικά αναφέρονται ως πακέτα. Ένα πακέτο περιέχει πληροφορίες ελέγχου που τις χρησιμοποιεί το δίκτυο για να ελέγχει την κίνηση τους και κάποιες φορές περιέχει δεδομένα χρηστών. Ένα παράδειγμα πληροφορίας είναι ο προορισμός ενός πακέτου. Τα router χρησιμοποιούν τέτοιου είδους πληροφορίες για να βρουν τον τρόπο με τον οποίο θα προωθήσουν το κάθε πακέτο. Το πρωτόκολλό (protocol) είναι μια "συμφωνία" για τα πακέτα που ανταλλάσσονται από τα προγράμματα επικοινωνίας και το τι σημαίνουν. Το πρωτόκολλο αναφέρει για το πώς τα πακέτα δομούνται.

2.9 Τι είναι socket

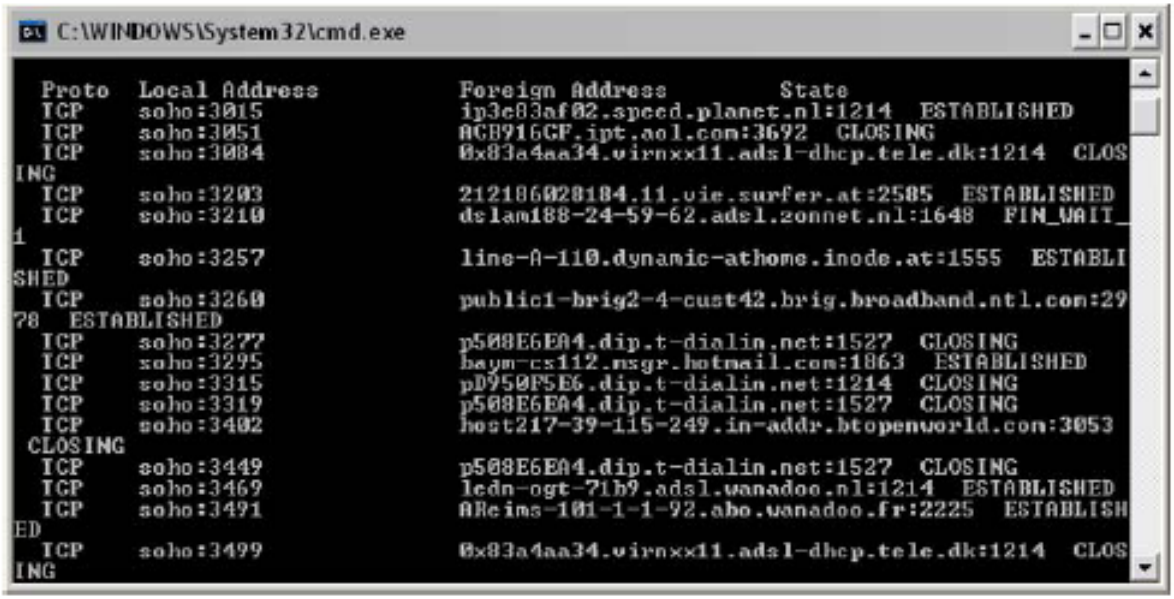
Το socket είναι μια αφαιρετική έννοια, μέσω της οποίας μια εφαρμογή μπορεί να στείλει και να λάβει δεδομένα, με τον ίδιο τρόπο που ένα ανοιχτό αρχείο επιτρέπει σε μια εφαρμογή να διαβάζει και να γράφει δεδομένα σε ένα σταθερό μέσο αποθήκευσης. Το Socket επιτρέπει σε μια εφαρμογή να "συνδεθεί" σε ένα δίκτυο και να επικοινωνήσει με άλλες εφαρμογές οι οποίες επίσης είναι συνδεδεμένες στο ίδιο δίκτυο. Πληροφορίες που γράφονται στο socket από μια εφαρμογή σε έναν υπολογιστή, μπορούν να αναγνωστούν από μια εφαρμογή σε ένα διαφορετικό μηχάνημα και αντιστρόφως. Οι διαφορετικοί τύποι των socket ανταποκρίνονται στις διαφορετικές σουίτες πρωτοκόλλων που θεμελιώνονται καθώς και στις διαφορετικές στοίβες πρωτοκόλλων μέσα στην ίδια σουίτα. Τα sockets στη σημερινή εποχή χρησιμοποιούν συνήθως τη σουίτα πρωτοκόλλων του TCP/IP. Οι κύριοι τύποι τους

είναι stream sockets (ροής) και datagram sockets. Τα stream sockets χρησιμοποιούν το TCP/IP ως πρωτόκολλό end-to-end (από άκρο σε άκρο) και με αυτό τον τρόπο παρέχουν υπηρεσίες σύνδεσης - αξιόπιστης μεταφοράς δεδομένων. Τα Datagram Sockets χρησιμοποιούν UDP (User Datagram Protocol) και παρέχουν υπηρεσίες μεταφοράς "καλύτερης προσπάθειας" - best effort.

Πίνακας 2. Γνωστά Port Numbers

Αριθμός Port	Πρωτόκολλο
20	FTP Δεδομένα
21	FTP Έλεγχος
25	SMTP (αποστολή email)
53	DNS
80	HTTP
110	POP3 (αποστολή email)
143	IMAP (λήψη email)

Όλα τα packets που ταξιδεύουν στο Internet, θα πρέπει να χρησιμοποιούν το Internet Protocol. Αυτό σημαίνει ότι η πηγαία IP διεύθυνση και η IP διεύθυνση προορισμού θα πρέπει να περιλαμβάνονται στο πακέτο. Τα περισσότερα πακέτα θα πρέπει επίσης να περιλαμβάνουν και ένα αριθμό port. Το Port είναι απλά ένας αριθμός μεταξύ του 1 και του 65,535 και χρησιμοποιείται για να διαφοροποιήσει πρωτόκολλα υψηλότερου επιπέδου όπως FTP ή SMTP.

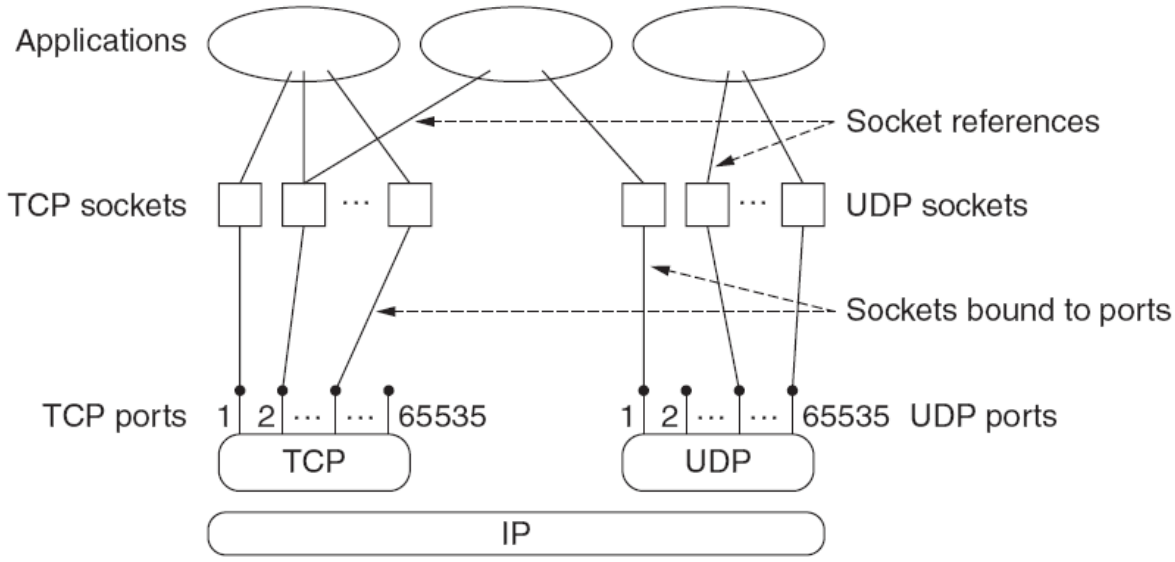


Εικόνα 19. Το εργαλείο NetStat.

Η σωστή χρήση των port είναι σημαντική στην περίπτωση του network programming, διότι δεν μπορούν 2 προγράμματα να χρησιμοποιούν την ίδια port. Προτείνεται για πειραματικούς σκοπούς να χρησιμοποιούνται ports μεγαλύτερες από την 1024.

Πακέτα που περιέχουν ports είναι συνήθως δυο ειδών: UDP και TCP/IP. Τα UDP έχουν χαμηλότερα latency συνήθως, ειδικά κατά την εκκίνηση. Σε περιπτώσεις

που η ακεραιότητα των δεδομένων δεν είναι πρωτίστης σημασίας, το UDP μπορεί να αποδειχθεί ευκολότερο στην χρήση. Στην πράξη όμως τα πακέτα UDP δεν φτάνουν με τη σωστή σειρά τους και κατ' επέκταση θεωρούνται «άχρηστα». Το TCP/IP είναι περισσότερο περίπλοκο από το UDP και υψηλότερο latency αλλά το πρωτόκολλο αυτό εγγυάται ότι τα δεδομένα δεν θα αλλοιωθούν ενώ ταξιδεύουν στο internet.



Εικόνα 20.Τα Sockets, protocols και οι ports.

Το TCP είναι περισσότερο χρήσιμο στη μεταφορά αρχείων, όπου μια extra καθυστέρηση δεν επηρεάζει και τόσο όσο ένα corrupted πακέτο και το UDP ταιριάζει περισσότερο στις περιπτώσεις Internet Radio ή streaming, όπου ένα χαμένο καρέ ή ένας περίεργος θόρυβος είναι περισσότερο αποδεκτός από ένα εκτεταμένο χρόνο μη υπηρεσίας.

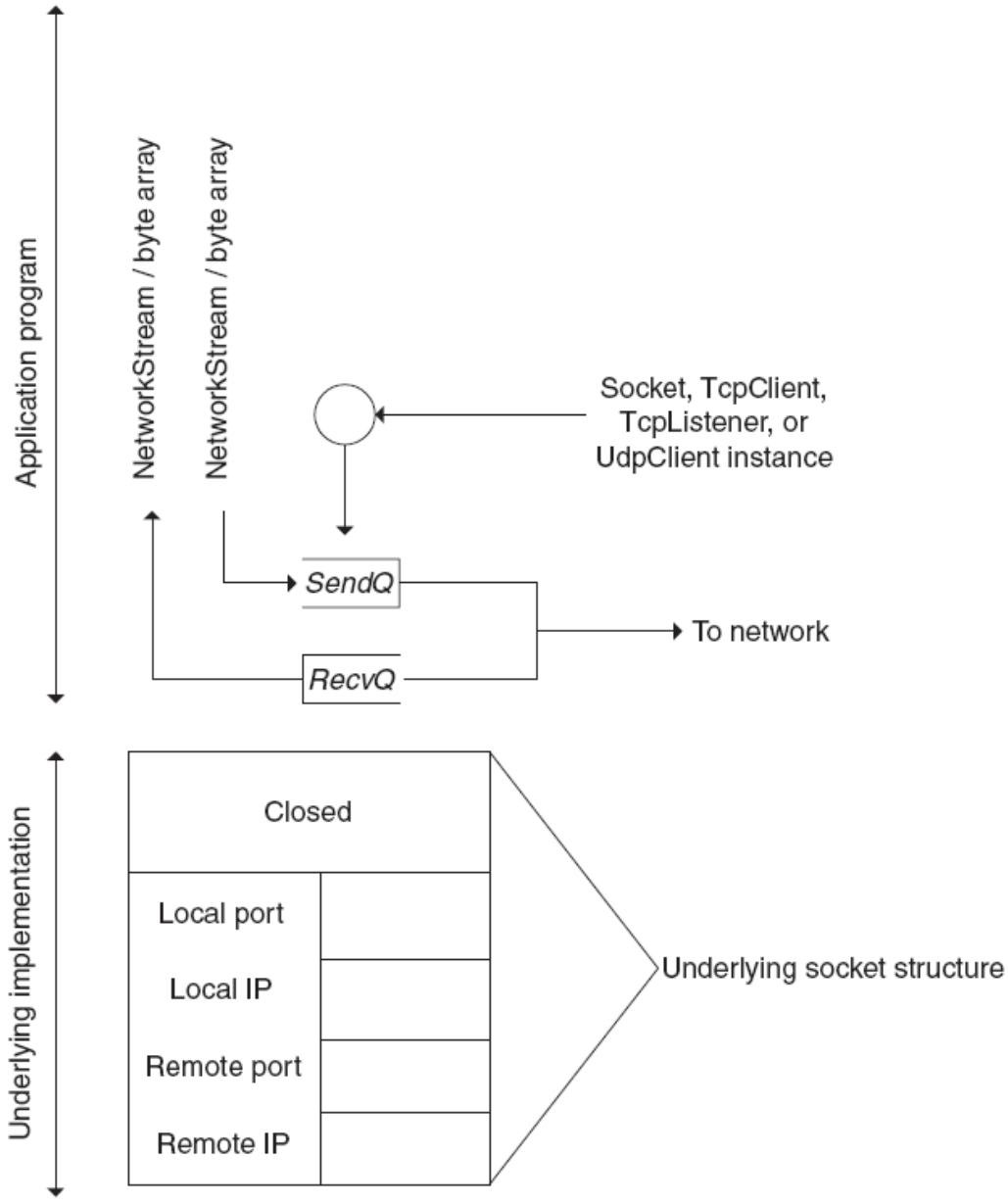
2.10 Οι Clients και οι Servers

Σε αναλογία με το ταχυδρομείο, κάθε τηλεπικοινωνία αποτελείται από δύο μέλη. Το ένα μέλος είναι αυτό που στέλνει το γράμμα αρχικώς και το άλλο μέλος είναι αυτό που στέλνει πίσω ένα γράμμα απάντησης. Οι όροι client και Server έχουν παρόμοιους ρόλους: Η client εφαρμογή ξεκινάει την επικοινωνία, ενώ ο server περιμένει παθητικά και ανταποκρίνεται όταν κάποιος client προσπαθεί να επικοινωνήσει μαζί του. Μαζί, ο client και ο server αποτελούν την εφαρμογή.

Το αν ένα πρόγραμμα ενεργεί ως client ή ως server καθορίζει τη γενική μορφή του socket api που θα χρησιμοποιήσει. Ο client-server διαχωρισμός αυτός είναι σημαντικός διότι ο client πρέπει να γνωρίζει τη διεύθυνση και την port του server αρχικά και όχι το αντίθετο. Με το api των socket, ο server μπορεί, εάν αυτό είναι απαραίτητο, να μάθει τη διεύθυνση του client όταν λάβει την πρώτη αίτηση επικοινωνίας από τον client. Αυτό είναι ανάλογο μιας τηλεφωνικής κλήσης, από τη στιγμή που πραγματοποιηθεί η σύνδεση, ο διαχωρισμός μεταξύ client και server εξαφανίζεται.

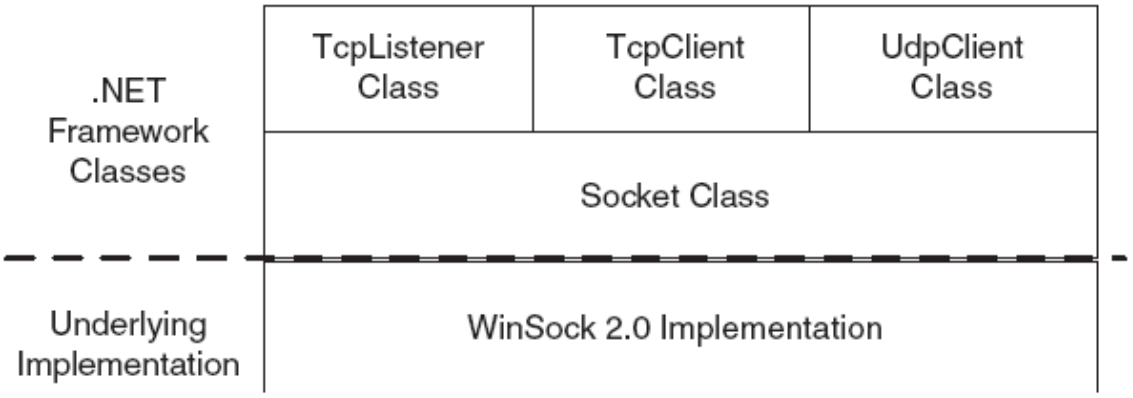
2.11 TCP Sockets

Στο .NET framework υπάρχουν δυο κλάσεις ειδικά για το TCP Protocol: TcpClient και TcpListener. Αυτές οι κλάσεις παρέχουν υψηλού επιπέδου αφαιρετικότητα της Socket κλάσης, αν και υπάρχουν περιπτώσεις που εκτεταμένη λειτουργικότητα επιτυγχάνεται μόνο μέσω της απ' ευθείας χρήσης της κλάσης Socket.



Εικόνα 21. Συσχετισμοί socket με δομές δεδομένων.

Ένα TcpConnection είναι ένα αφαιρετικός ορισμός ενός καναλιού δυο δρόμων το οποίου το κάθε άκρο αναγνωρίζεται από την IP και το αριθμό της port. Η .NET χρησιμοποιεί επίσης την κλάση EndPoint και την υποκλάση IPEndPoint για την αφαιρετική μορφή αυτής της έννοιας.



Εικόνα 22. Σχέσεις μεταξύ των κλάσεων Socket.

Μια instance του TcpListener «ακούει» για αιτήσεις TCP σύνδεσης και δημιουργεί ένα νέο socket (στη μορφή του TcpClient ή Socket) για να διαχειριστεί την κάθε εισερχόμενη σύνδεση.

2.12 Επικοινωνία με File Servers

Το FTP protocol επικοινωνεί με 2 ports, 21, το socket ελέγχου και ένα socket δεδομένων στην port 20 ή σε κάποια άλλη μεγαλύτερη port.

Όπως και στα email protocols, η επικοινωνία μεταξύ client και server είναι αρκετά κοντά σε μια τυπική ανθρώπινη επικοινωνία και διαχωρίζεται σε γραμμές όπως ένα οποιοδήποτε κείμενο σε μια φυσική γλώσσα.

2.13 Πώς το FTP χρησιμοποιεί ports

Η port 21 χρησιμοποιείται για την αποστολή και λήψη εντολών και απαντήσεων, κάθε μια από αυτές θεωρείται ότι τελειώνει από ένα <enter>. Όταν αποστέλλονται δεδομένα μεταβλητού μήκους μεταξύ client και server, όπως αρχεία και λίστες φακέλων, μια προσωρινή σύνδεση ανοίγει στην port 20, τα δεδομένα μεταφέρονται και τότε η port αυτή κλείνει ξανά. Παρ' όλα αυτά στον πραγματικό κόσμο κάτι τέτοιο καθίσταται μη εφικτό κάποιες φορές λόγω του ότι ο client μπορεί να βρίσκεται πίσω από ένα firewall.

Το Passive-mode FTP υπάρχει όταν ο client ζητάει από τον server να «ακούσει» από μια port διαφορετική από την προεπιλεγμένη. Ο client τότε θα συνδεθεί σε εκείνη την port και θα την χρησιμοποιήσει για την αποστολή και λήψη δεδομένων.

2.14 Μετακίνηση στους φακέλους

Όπως και με την πλοήγηση στους φακέλους ενός τυπικού file system, απαιτείται η γνώση των αρχείων και των φακέλων που υπάρχουν σε κάθε directory.

Η συνήθης διαδικασία είναι η εξής:

- Ο Client αποστέλλει την εντολή "LIST"
- Ο Server αναμένει για την δημιουργία data socket.
- Server μεταφέρει δεδομένα.

- Ο Server κλείνει το connection.

Πίνακας 3. Τυπικές FTP εντολές

FTP Εντολή	Ενέργεια
RETR	Download
STOR	Upload
STOU	Upload, όπου ο server επιλέγει το όνομα του τελικού αρχείου.
APPE	Appends
REST	Επανεκκίνηση της μεταφοράς σε ένα συγκεκριμένο σημείο
RNFR	Μετονομασία αρχείου From
RNTO	Μετονομασία αρχείου To
ABOR	Διακοπή μεταφοράς
DELE	Διαγραφή συγκεκριμένου αρχείου
RMD	Διαγραφή φακέλου
MKD	Δημιουργία φακέλου
PWD	Επιστρέφει το Preset Work Directory
LIST	Επιστρέφει τη λίστα με τα περιεχόμενα του τρέχοντος φακέλου σε μορφή αναγνώσιμη από τον άνθρωπο
NLST	Λίστα όλων των αρχείων στο συγκεκριμένο φάκελο
SITE	Παρέχει τη λίστα των παρεχόμενων υπηρεσιών
SYST	Επιστρέφει το όνομα του λειτουργικού συστήματος
STAT	Επιστρέφει την κατάσταση μεταφοράς
HELP	Επιστρέφει πληροφορίες για τον server σε μορφή αναγνώσιμη από τον άνθρωπο
NOOP	No effect
USER	Καθορίζει το username του χρήστη
PASS	Καθορίζει το password του χρήστη
TYPE	Υποδεικνύει τη μορφή των δεδομένων, A για ASCII, E για EBCDIC, I για BINARY, Ln για προσαρμοσμένο μέγεθος byte (n μήκος του byte)

2.15 Τι μπορεί να κάνει ένα network program?

Network program είναι μια εφαρμογή η οποία χρησιμοποιεί ένα δίκτυο ηλεκτρονικών υπολογιστών για να μεταφέρει πληροφορίες μεταξύ εφαρμογών αμφίδρομα. Το εύρος τέτοιων εφαρμογών μπορεί να είναι πολύ μεγάλο, απο έναν συνηθισμένο Web Browser όπως ο Internet Explorer, η εφαρμογή αποστολής και λήψης e-mail, μέχρι και το λογισμικό που ελέγχει διαστημόπλοια απο τη NASA. Όλες αυτές οι εφαρμογές μοιράζονται την ικανότητα να επικοινωνούν με άλλους υπολογιστές, παρέχοντας έτσι τη δυνατότητα αυτόματα να είναι πιο χρήσιμες προς τον τελικό χρήστη. Στην περίπτωση του Web Browser όπως ο Internet Explorer, κάθε ιστοσελίδα που επισκέπτεστε είναι στην ουσία αρχεία αποθηκευμένα σε έναν υπολογιστή κάπου αλλού στο Internet. Με το πρόγραμμα αποστολής/λήψης e-mail, υπάρχει επικοινωνία με τον ηλεκτρονικό υπολογιστή του ISP (Internet Service Provider) ή με τους υπολογιστές της εταιρίας που ελέγχει τους εξυπηρετητές email. Η συγκεκριμένη πτυχιακή αφορά τη δημιουργία κομματιών λογισμικού που θα παρέχουν τη δυνατότητα μεταφοράς και αποθήκευσης αρχείων. Παρόλο που πλέον οι ικανότητες των ιστοσελίδων και των δικτυακών εφαρμογών αρχίζουν πλέον να

συμπτύσσονται, είναι σημαντικό να καταλάβουμε τα προτερήματα και τα μειονεκτήματα του κάθε συστήματος. Μια υπηρεσία μέσω ενός Web site, είναι άμεσα προσβάσιμη στους χρήστες μέσω πληθώρας διαφορετικών εφαρμογών και πλατφόρμων και ολόκληρη η δικτυακή αρχιτεκτονική είναι ήδη έτοιμη αλλά υπάρχει ένα σημείο όπου οι δυνατότητες που παρέχονται μέσω των ιστοσελίδων είναι πραγματικά περιορισμένες. Σε αυτή την περίπτωση θα χρειαστεί να απευθυνθούμε σε δικτυακές εφαρμογές.

Οι χρήστες γενικά εμπιστεύονται τις δικτυακές εφαρμογές, οπότε αυτά τα προγράμματα έχουν μεγαλύτερο έλεγχο στον ηλεκτρονικό υπολογιστή στον οποίο τρέχουν παρά τις δυνατότητες που έχει ένα web site στους υπολογιστές που το προβάλλουν μέσω του browser τους. Αυτό επιτρέπει, σε μια δικτυακή εφαρμογή, να διαχειρίζεται αρχεία στον τοπικό υπολογιστή, σε αντίθεση με ένα web site, στην πράξη δεν μπορεί να το κάνει αυτό. Κυρίως, από διαδικτυακή άποψη, μια εφαρμογή έχει μεγαλύτερο έλεγχο σχετικά με τον τρόπο επικοινωνίας με τους άλλους υπολογιστές στο Internet.

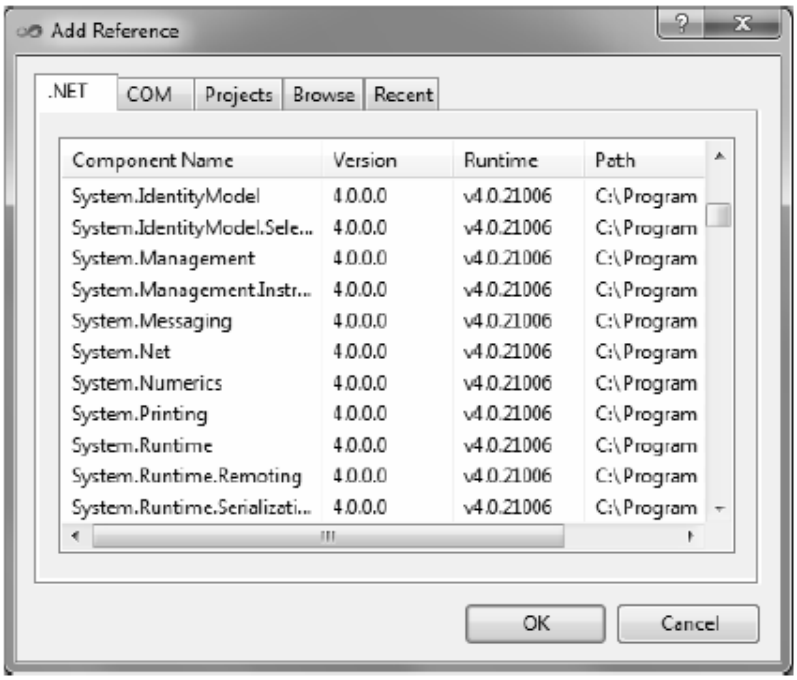
Για παράδειγμα, μια ιστοσελίδα δεν μπορεί να επιτρέψει σε ένα υπολογιστή που προβάλλει το web site, να ανοίξει μια μόνιμη σύνδεση με έναν άλλο υπολογιστή (εκτός από τον εξυπηρετητή που περιέχει αποθηκευμένη την ιστοσελίδα). Αυτό συμβαίνει ακόμα και όταν το web site περιέχει ενσωματωμένο περιεχόμενο όπως π.χ. ένα Java Applet ή ένα flash clip. Υπάρχει βέβαια μια εξαίρεση στον κανόνα, μόνο όταν υπάρχει ένα εκτελέσιμο κομμάτι κώδικα (όπως ένα ActiveX control στοιχείο) που εμπεριέχεται στη σελίδα. Σε αυτή την περίπτωση, η ιστοσελίδα μπορεί να έχει αρκετά από τα χαρακτηριστικά που θα είχε μια δικτυακή εφαρμογή αλλά οι περισσότεροι φυλλομετρητές και συστήματα antivirus, θα προειδοποιήσουν για αυτές τις ενέργειες ή ακόμα και θα αρνηθούν την εκτέλεση τέτοιου περιεχομένου.

ΚΕΦΑΛΑΙΟ 3ο

NAMESPACES, ΒΙΒΛΙΟΘΗΚΕΣ ΚΑΙ ΤΕΧΝΙΚΕΣ

3.1 Multithreading και Παράλληλος Προγραμματισμός

Είναι προφανές ότι δεν είναι καλό μια εφαρμογή να είναι αργή και να κλειδώνει άλλες διαδικασίες κατά την εκτέλεσή της. Γι' αυτό το λόγο η πλατφόρμα .NET παρέχει μια πληθώρα τρόπων για την δημιουργία λογισμικού που θα μπορεί να εκτελεί πολλαπλές και πολύπλοκες διεργασίες χωρίς να προκαλεί το προσωρινό κλείδωμα του προγράμματος.



Εικόνα 23. Το παράθυρο προσθήκης αναφοράς (Add Reference).

Ένας από τους τρόπους αυτούς είναι η χρήση τύπων delegate για την υποστήριξη κλήσεων ασύγχρονων μεθόδων. Γίνεται η χρήση του System.Threads namespace που παρέχει τη δυνατότητα κλήσης και διαχείρισης πολλαπλών νημάτων κατά το runtime.

Πίνακας 4. System.Threading Namespace

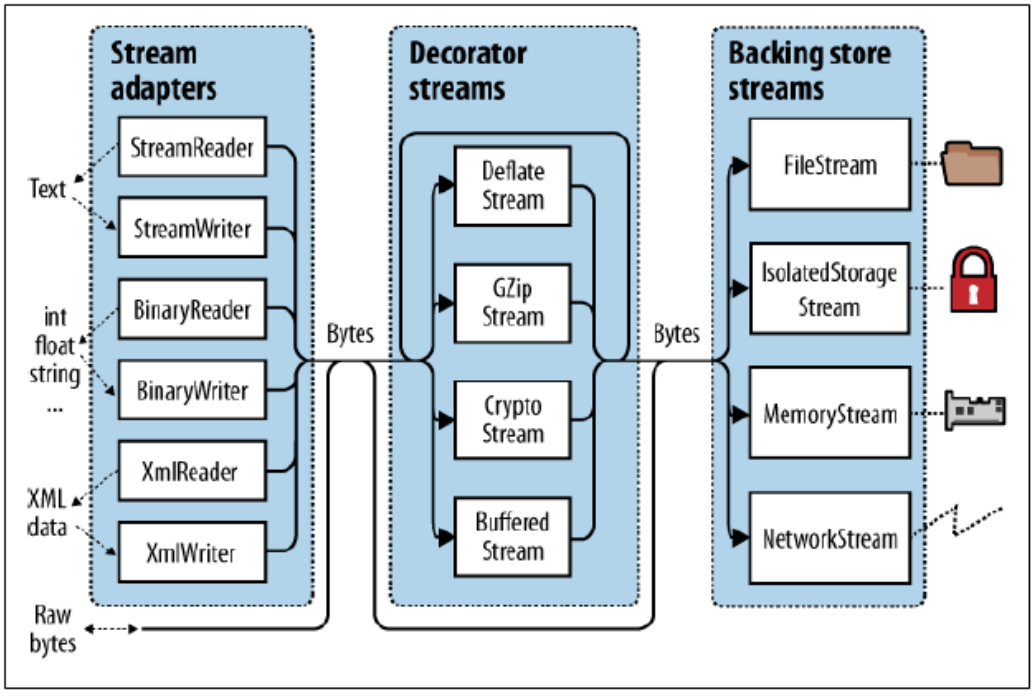
Τύπος	Σημασία
Interlocked	Αυτός ο τύπος παρέχει ατομικές λειτουργίες για μεταβλητές που μοιράζονται μεταξύ πολλαπλών threads.
Monitor	Παρέχει τον συγχρονισμό των threaded αντικειμένων μέσω της χρήσης lock και wait signals.
Mutex	Χρησιμοποιείται για το συγχρονισμό μεταξύ διαφορετικών εφαρμογών
ParameterizedThreadStart	Αυτός ο τύπος delegate επιτρέπει σε ένα thread να αποκτήσει κάποιο αριθμό παραμέτρων
Semaphore	Αυτός ο τύπος επιτρέπει τον περιορισμό του αριθμού των thread που μπορούν να έχουν πρόσβαση

	ταυτόχρονα σε ένα συγκεκριμένο πόρο ή σε ένα είδος πόρων
Thread	Αναπαριστά ένα νήμα το οποίο εκτελείται μέσα στο CLR. Με αυτό τον τύπο, μπορούν να δημιουργηθούν πληθώρα νέων threads πηγάζοντας από το αρχικό application domain
ThreadPool	Αυτός ο τύπος επιτρέπει την αλληλεπίδραση του συγκεκριμένου πλήθους νημάτων που διαχειρίζεται από το CLR, με μια συγκεκριμένη επεξεργασία
ThreadStart	Αυτός ο delegate τύπος χρησιμοποιείται για τον καθορισμό μεθόδου που θα κληθεί για ένα συγκεκριμένο νήμα. Σε αντίθεση με το ParameterizedThreadStart delagete, οι στόχοι του ThreadStart πρέπει πάντα να έχουν το ίδιο πρωτότυπο
Timer	Αυτός ο τύπος παρέχει έναν μηχανισμό εκτέλεσης της μεθόδου για συγκεκριμένα χρονικά στιγμιότυπα
TimerCallBack	Αυτός ο delegate τύπος χρησιμοποιείται σε συνδυασμό με τους τύπους Timer.

3.2 Είσοδος έξοδος αρχείων

Κατά την δημιουργία desktop εφαρμογών, η ικανότητα αποθήκευσης πληροφοριών μεταξύ των διαφορετικών sessions των χρηστών είναι σημαντική. Η διαχείριση αρχείων/φακέλων και η ανάγνωση/αποθήκευσή τους γίνεται μέσω του System.IO namespace.

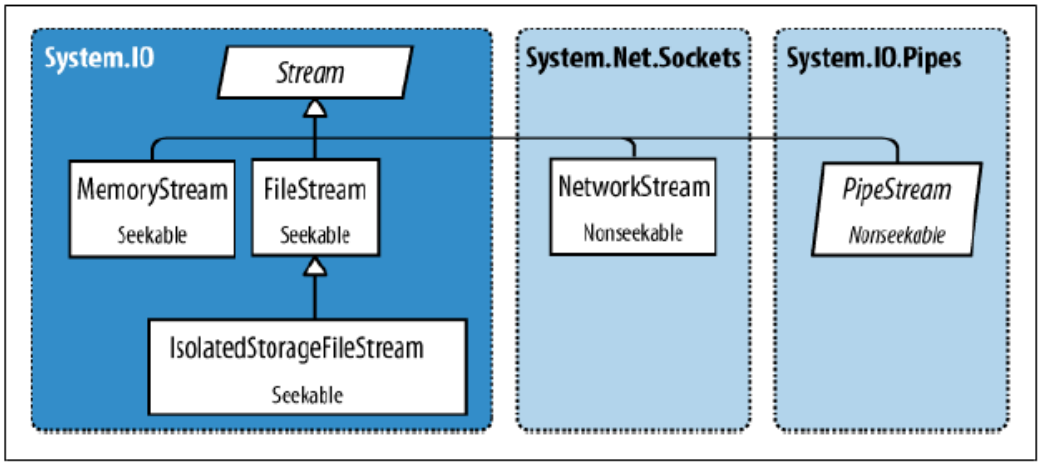
Στο .NET Framework, το namespace αυτό είναι αφιερωμένο στις βασικές βιβλιοθήκες κλάσεων που αφορούν υπηρεσίες εισόδου-εξόδου αρχείων, string buffers, όπως και διευθύνσεις μνήμης.



Εικόνα 24. Η αρχιτεκτονική των Stream.

Πίνακας 5. I/O κλάσεις

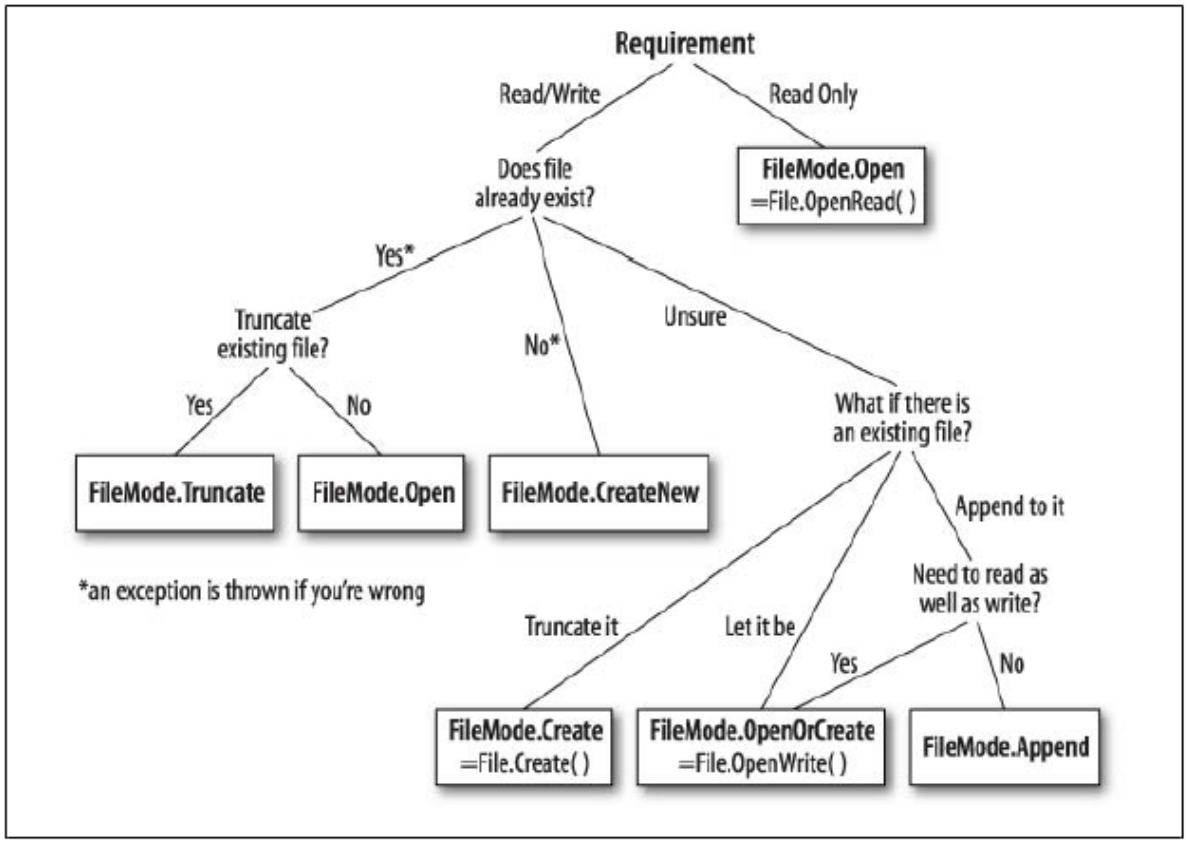
Μη αφαιρετική I/O κλάση	Σημασία
BinaryReader **BinaryWriter**	Επιτρέπει την αποθήκευση και επαναφορά απλών τύπων δεδομένων (ακέραιοι, Booleans, string κτλ) σε δυαδική μορφή
BufferedStream	Αυτή η κλάση επιτρέπει την προσωρινή αποθήκευση των byte ενός stream, που μπορούν να επαναχρησιμοποιηθούν αργότερα
Directory **DirectoryInfo**	Χρησιμοποιούνται για τη διαμόρφωση της δομής φακέλων. Ο τύπος Directory, εκθέτει τη λειτουργικότητά του χρησιμοποιώντας static members, ενώ το DirectoryInfo παρέχει ίδια λειτουργικότητα με την περίπτωση χρήσης object reference.
DriveInfo	Αυτή η κλάση παρέχει λεπτομερείς πληροφορίες για τα drive που χρησιμοποιεί το συγκεκριμένο σύστημα
File **FileInfo**	Αυτές οι κλάσεις μπορούν να χρησιμοποιηθούν για την διαχείριση αρχείων σε ένα μηχάνημα. Ο τύπος File, εκθέτει τη λειτουργικότητά του χρησιμοποιώντας static members, ενώ το FileInfo παρέχει ίδια λειτουργικότητα με την περίπτωση χρήσης object reference.
FileStream	Αυτή η κλάση παρέχει τυχαία πρόσβαση στα αρχεία (π.χ. δυνατότητες αναζήτησης) με τα δεδομένα να αναπαρίστανται ως ακολουθία bytes
FileSystemWatcher	Αυτή η κλάση παρέχει τη δυνατότητα παρακολούθησης εξωτερικών αρχείων σε κάποιο συγκεκριμένο φάκελο
MemoryStream	Αυτή η κλάση παρέχει τυχαία πρόσβαση σε streamed δεδομένα που είναι αποθηκευμένα στη μνήμη και όχι σε ένα φυσικό αρχείο
Path	Αυτή η κλάση εκτελεί λειτουργίες για τους System.String τύπους που περιέχουν διαδρομή αρχείων ή φακέλων σε μορφή ανεξάρτητη από την πλατφόρμα
StreamWriter **StreamReader**	Αυτές οι κλάσεις χρησιμοποιούνται για την αποθήκευση και επαναφορά κειμένων από και προς ένα αρχείο. Αυτός ο τύπος δεν υποστηρίζει τυχαία προσπέλαση στα αρχεία
StringWriter **StringReader**	Όπως και οι κλάσεις StreamWriter/StreamReader, αυτές οι κλάσεις λειτουργούν με κείμενα text. Παρ' όλα αυτά, το επίπεδο αποθήκευσης αφορά string buffer και όχι φυσικό αρχείο



Εικόνα 25. Τα Backing store Stream.

3.3 DirectoryInfo και FileInfo τύποι

Το System.IO παρέχει τέσσερεις κλάσεις που παρέχουν τη δυνατότητα διαχείρισης ανεξάρτητων αρχείων καθώς και αλληλεπίδραση με τη δομή φακέλων σε έναν Η/Υ. Οι πρώτοι δυο τύποι, Directory και File, παρέχουν δυνατότητα δημιουργίας, διαγραφής, αντιγραφής και μετακίνησης χρησιμοποιώντας διάφορα στατικά μέλη. Οι παρόμοιοι τύποι DirectoryInfo και FileInfo παρέχουν παρόμοια λειτουργικότητα αλλά με χρήση instance-level μεθόδων.



Εικόνα 26. Επιλέγοντας FileMode.

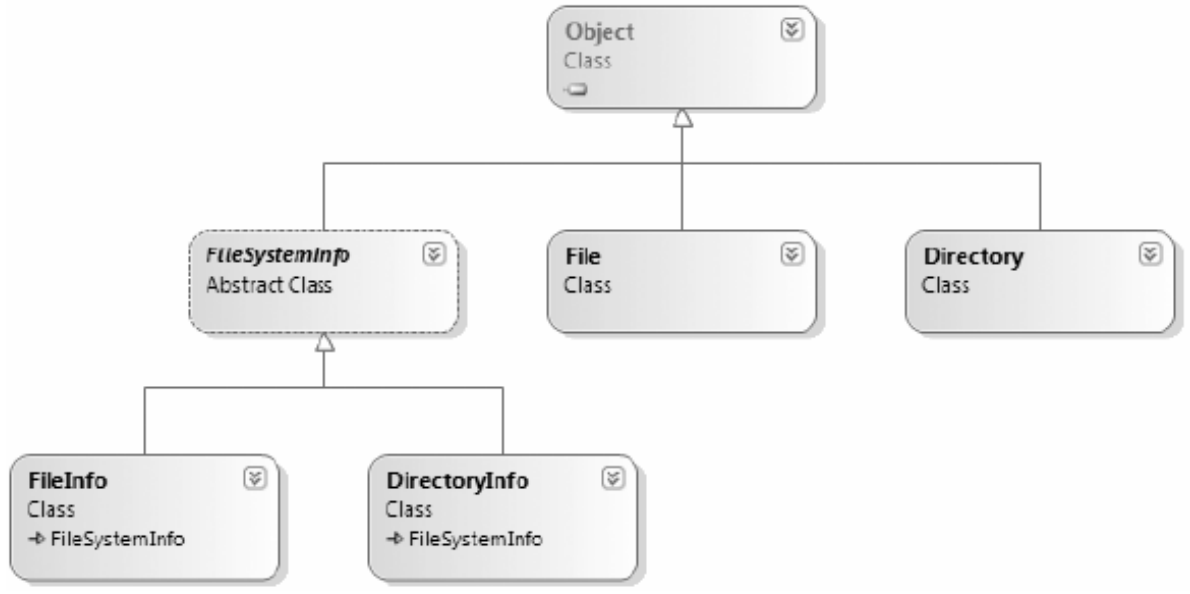
Τα FileInfo και DirectoryInfo, τυπικά εξυπηρετούν ως καλύτερες επιλογές για την απόκτηση πλήρων λεπτομερειών για ένα αρχείο ή φάκελο (π.χ. ημερομηνία δημιουργίας, δυνατότητες εγγραφής/ανάγνωσης) επειδή τα μέλη τους τείνουν να επιστρέφουν αντικείμενα.

3.4 Ιδιότητες του FileSystemInfo

Πίνακας 6. Ιδιότητες του FileSystemInfo

Ιδιότητα	Σημασία
Atributes	Επιστρέφει ή αλλάζει τα γνωρίσματα που αφορούν το συγκεκριμένο αρχείο που παρουσιάζεται από το FileAttributes enumeration (π.χ. εάν είναι το αρχείο ή ο φάκελος μόνο για ανάγνωση, κρυπτογραφημένο, συμπιεσμένο ή κρυφό)
CreationTime	Επιστρέφει ή αλλάζει το χρόνο δημιουργίας του συγκεκριμένου αρχείου ή φακέλου.
Exists	Χρησιμοποιείται για να διαπιστώνεται αν υπάρχει ή όχι το συγκεκριμένο αρχείο/φάκελος, επιστρέφει Boolean μεταβλητή.
Extension	Επιστρέφει την προέκταση ενός αρχείου.
FullName	Επιστρέφει την πλήρη διαδρομή του φακέλου ή του αρχείου
LastAccessTime	Επιστρέφει ή αλλάζει το χρόνο που το συγκεκριμένο αρχείο ή φάκελος είχε τελευταία πρόσβαση.
LastWriteTime	Επιστρέφει ή αλλάζει το χρόνο που στο συγκεκριμένο αρχείο ή φάκελος είχαν γραφτεί δεδομένα
Name	Επιστρέφει το όνομα του συγκεκριμένου αρχείου ή φακέλου.

Επίσης το FileSystemInfo καθορίζει την ύπαρξη της Delete() και Refresh() μεθόδου.



Εικόνα 27. Κλάσεις File- και Directory-.

Ιδιότητες του DirectoryInfo

Πίνακας 7. Ιδιότητες του DirectoryInfo

Μέλος	Σημασία
Create()	Δημιουργία ενός φακέλου (ή ενός σετ από φακέλους) στη

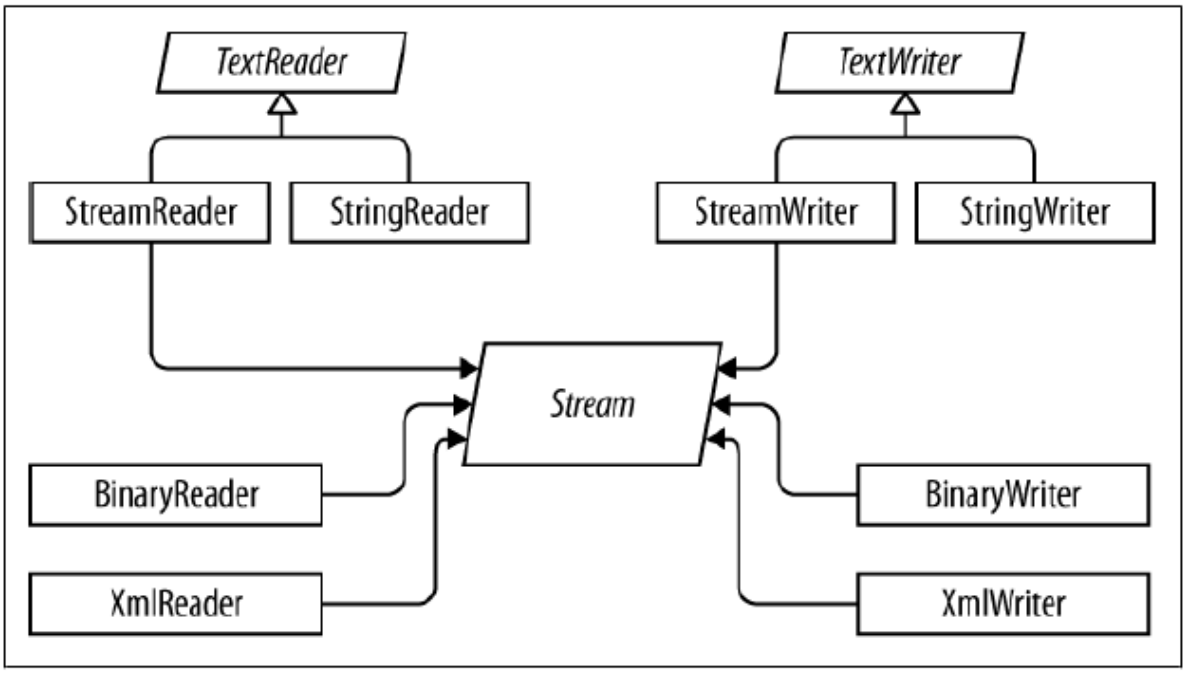
CreateSubdirectory()	δοθείσα διαδρομή
Delete()	Διαγράφει ένα φάκελο και όλα τα περιεχόμενα του
GetDirectories()	Επιστρέφει ένα πίνακα από DirectoryInfo αντικείμενα που αναπαριστούν όλους τους υποφακέλους στο συγκεκριμένο directory.
GetFiles()	Επιστρέφει ένα πίνακα από FileInfo αντικείμενα που αναπαριστούν όλα τα αρχεία στο δοθέν directory.
MoveTo()	Μετακινεί ένα φάκελο και τα περιεχόμενα του σε νέο path.
Parent	Επιστρέφει το γονικό directory από το συγκεκριμένο φάκελο.
Root	Επιστρέφει το root τμήμα μιας διαδρομής

3.5 BinaryWriters και BinaryReaders

Και τα δύο σετ προέρχονται από το System.Object. Αυτοί οι τύποι επιτρέπουν την ανάγνωση και εγγραφή διακεκριμένων τύπων δεδομένων σε ένα stream με δυαδική μορφή. Η BinaryWriter κλάση ορίζει τη Write() μέθοδο, η οποία εγγράφει δεδομένα στο υποσκάπτων stream.

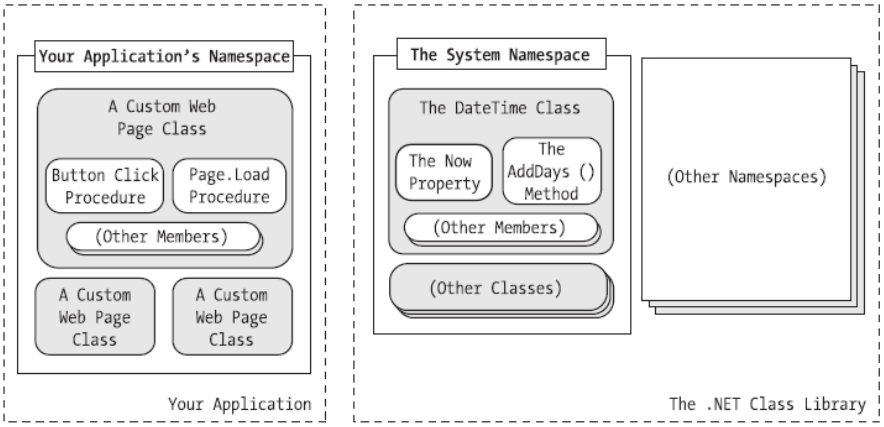
Πίνακας 8. Binary Writers/Readers

Μέλος	Σημασία
BaseStream	Αυτή η ιδιότητα μόνο ανάγνωσης παρέχει πρόσβαση στο υποσκάπτων stream που χρησιμοποιείται από το BinaryWriter αντικείμενο.
Close()	Αυτή η μέθοδος κλείνει το Binary Stream.
Flush()	Αυτή η μέθοδος εκτελεί τη λειτουργία Flush στο συγκεκριμένο Binary Stream.
Seek()	Αυτή η μέθοδος θέτει τη θέση του index στο συγκεκριμένο stream.
Write()	Αυτή η μέθοδος γράφει δεδομένα στο συγκεκριμένο stream.



Εικόνα 28. Readers και Writers.

NameSpaces



Εικόνα 29. Προβολή δυο namespaces.

Πίνακας 9. .NET Namespaces

.NET Namespace	Χρήση
System	Μέσα στο System Namespace, υπάρχουν πολλαπλοί τύποι που βοηθούν στη διαχείριση εσωτερικών δεδομένων, μαθηματικές πράξεις, τυχαίους αριθμούς, μεταβλητές στο περιβάλλον εκτέλεσης, garbage collection.
System.Collections **System.Collections.Generic**	Περιγράφουν ένα αριθμό βασικών τύπων όπως διεπαφές που επιτρέπουν στη δημιουργία προσαρμοσμένων συλλογών.
System.Data **System.Data.Common** **System.Data.EntityClient** **System.Data.SqlClient**	Αυτά τα namespaces χρησιμεύουν στην επικοινωνία με σχεσιακές βάσεις δεδομένων
System.IO **System.IO.Compression** **System.IO.Ports**	Καθορίζουν πολλαπλούς τύπους επικοινωνίας με είσοδο/έξοδο αρχείων, συμπίεση δεδομένων και χειρισμό ports.
System.Reflection **System.Reflection.Emit**	Αυτά τα namespaces καθορίζουν ανακάλυψη τύπων δεδομένων κατά το runtime και δυναμική δημιουργία τύπων δεδομένων
System.Runtime.InteropServices	Παρέχει την κατάλληλη υποδομή για την αλληλεπίδραση με unmanaged κώδικα (π.χ. διάφορα dll)
System.Drawing **System.Windows.Forms**	Καθορίζουν τύπους που χρησιμοποιούνται για τη δημιουργία κλασσικών desktop εφαρμογών χρησιμοποιώντας το αρχικό UI (Windows Forms) της .NET
System.Windows **System.Windows.Controls** **System.Windows.Shapes**	Αυτά τα namespaces είναι τα αρχικά για μια πληθώρα άλλων namespaces που χρησιμοποιούνται στο WPF (Windows Presentation Foundation) UI
System.Linq **System.Xml.Linq**	Παρέχουν τύπους δεδομένων για τον προγραμματισμό με χρήση του LINQ api

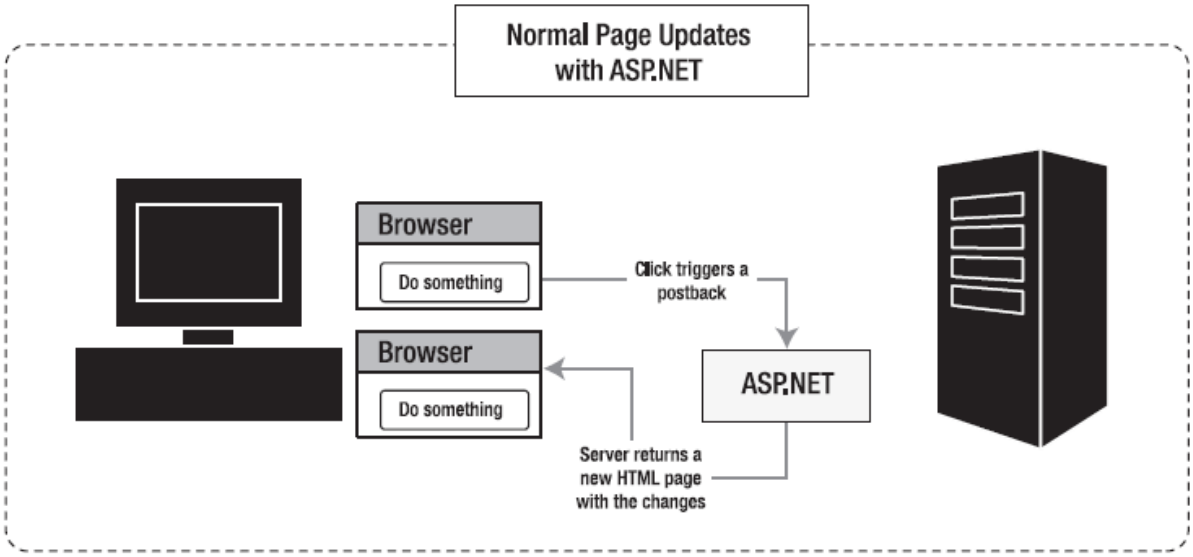
System.Data.DataSetExtensions	
System.Web	Ένα από τα πολλά namespaces που επιτρέπουν τη δημιουργία ASP.NET web εφαρμογών
System.ServiceModel	Χρησιμοποιείται για τη δημιουργία κατανεμημένων εφαρμογών χρησιμοποιώντας το WCF (Windows Communication Foundation) api.
System.Workflow.Runtime **System.Workflow.Activities**	Αυτά είναι δύο από τα πολλά namespaces που καθορίζουν τύπους για τη χρησιμοποίηση τους σε "workflow-enabled" εφαρμογές χρησιμοποιώντας το WWF (Windows Workflow Foundation) api.
System.Threading **System.Threading.Tasks**	Αυτό το namespace καθορίζει διάφορους τύπους για τη δημιουργία πολυνηματικών εφαρμογών που μπορούν να κατανέμουν workloads σε πληθώρα επεξεργαστών.
System.Security	Η ασφάλεια είναι ένα ενσωματωμένο χαρακτηριστικό στον κόσμο της .NET. Σε αυτό το namespace εμπεριέχονται διάφοροι τύποι που αφορούν ασφάλεια όπως permissions, κρυπτογραφία, κτλ
System.Xml	Περιγράφει τύπους που αφορούν την αλληλεπίδραση με τον XML τύπο δεδομένων.

ΚΕΦΑΛΑΙΟ 4ο

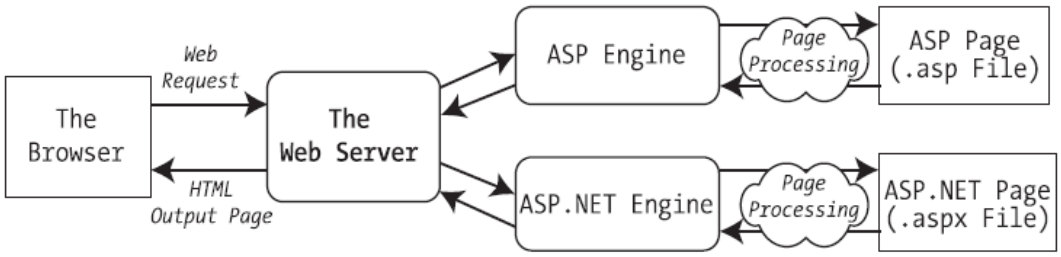
ASP.NET

4.1 ASP.NET Ιστοσελίδες

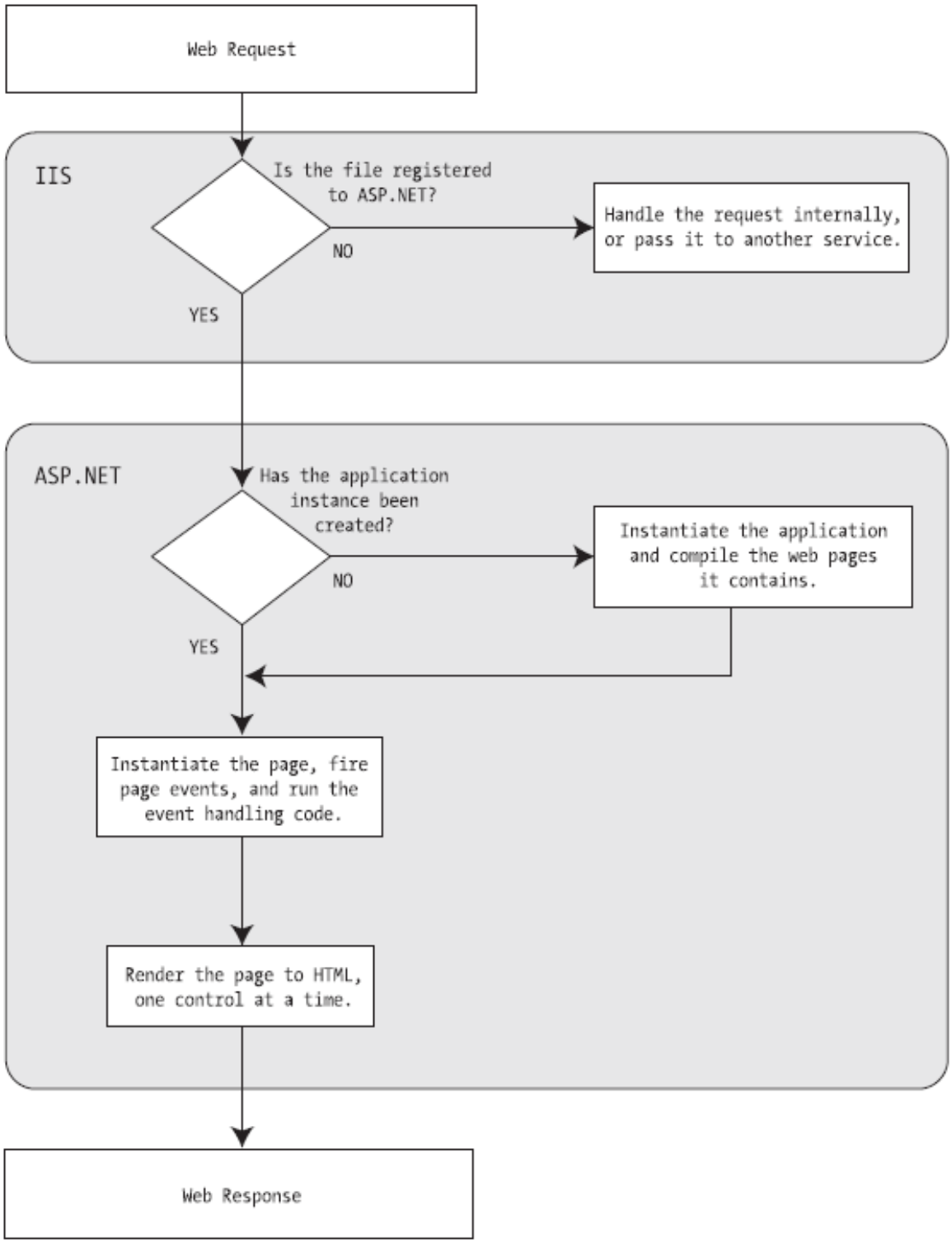
Η προεπιλεγμένη λογική που χρησιμοποιείται στο Visual Studio και στις ιστοσελίδες είναι η χρήση μιας τεχνικής γνωστής ως code-behind. Αυτή η τεχνική επιτρέπει στο διαχωρισμό του κώδικα εμφάνισης/HTML από τον κώδικα προγραμματιστικής λογικής σε ξεχωριστά αρχεία.



Εικόνα 30. Page updates με Asp.NET.



Εικόνα 31. Τρόπος διαχείρισης ASP file request απο τον IIS.

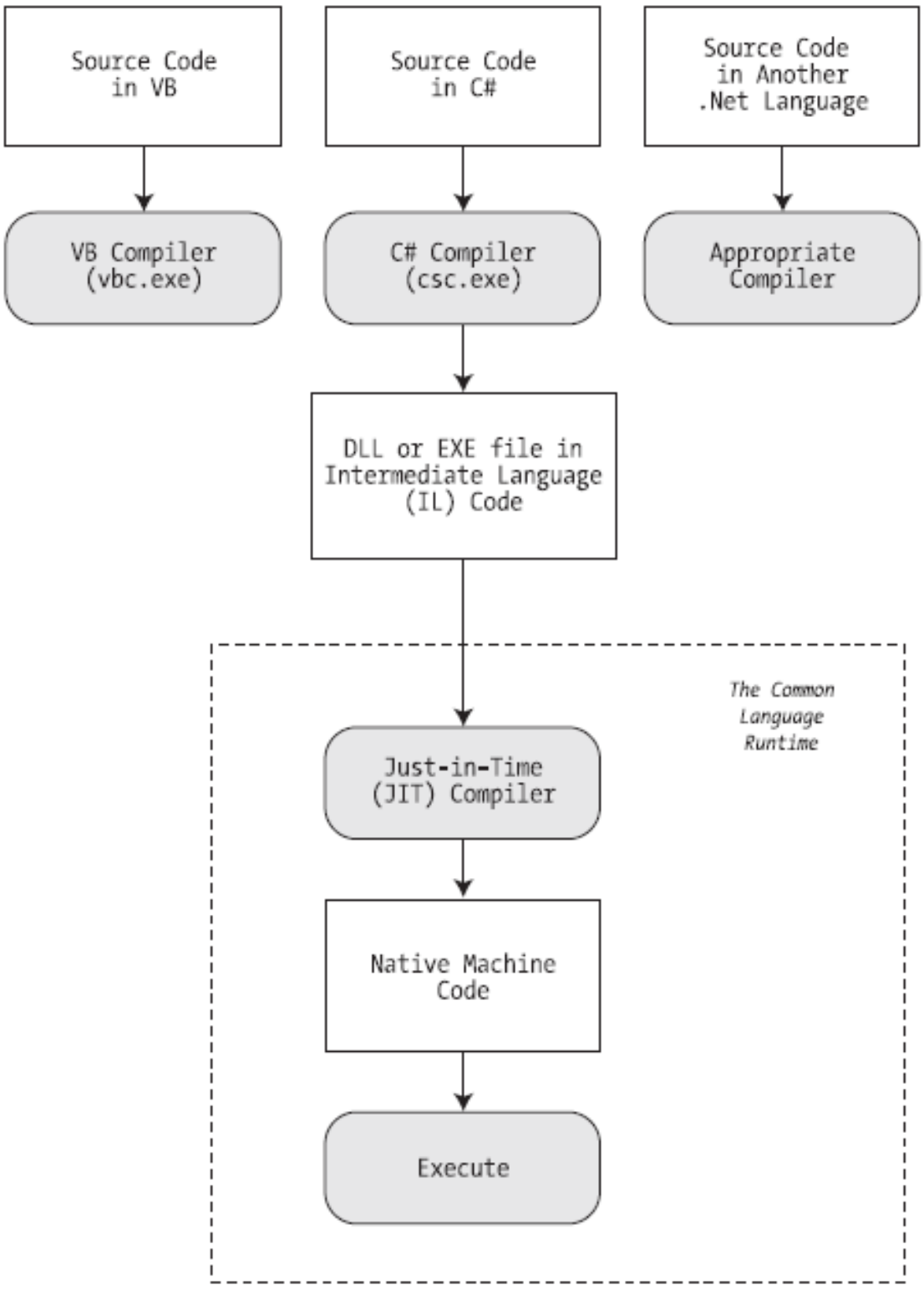


Εικόνα 32. Τα στάδια σε ένα request στην Asp.NET.

Επίσης επιτρέπει:

- Οι designers να δουλεύουν στο HTML Markup και οι programmers να δουλεύουν σε C# κώδικα ταυτόχρονα, λόγω του σαφή διαχωρισμού στον κώδικα.

- Ο κώδικας δεν είναι εκτεθειμένος στους designers, απλοποιώντας έτσι την επεξεργασία του κώδικα εμφάνισης.
- Τα αρχεία κώδικα μπορούν να γραφτούν σε πολλά ξεχωριστά *.aspx αρχεία.

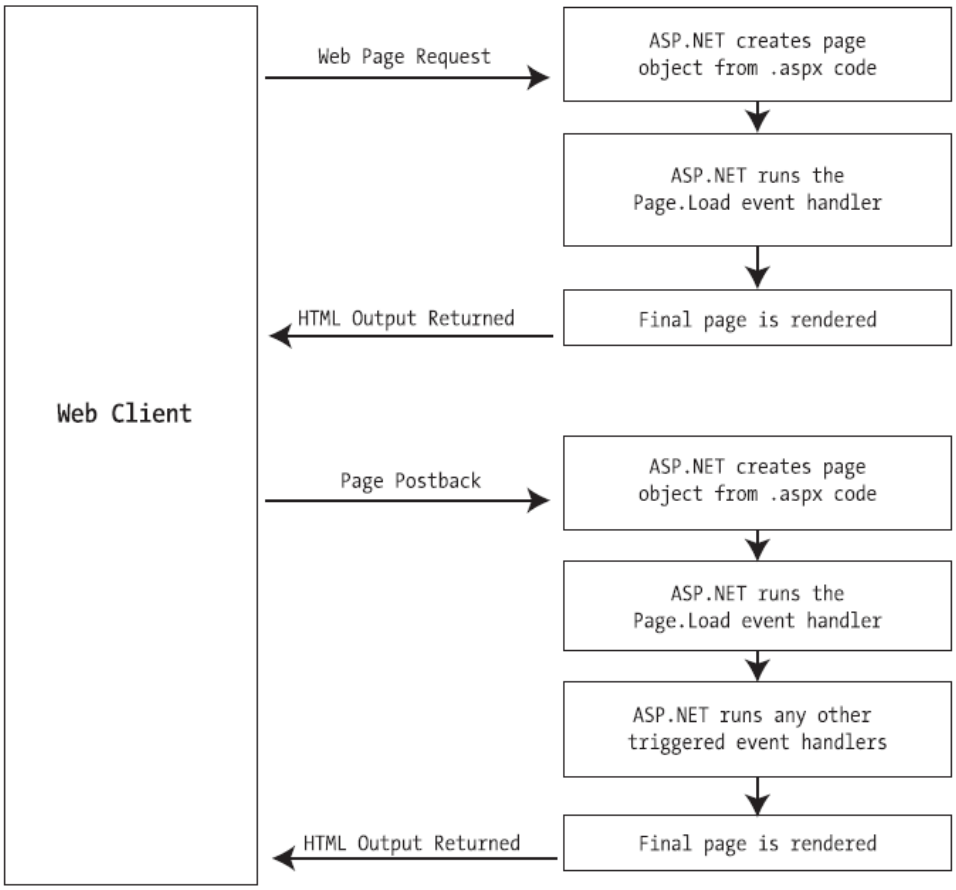


Εικόνα 33. Το compilation στην Asp.NET.

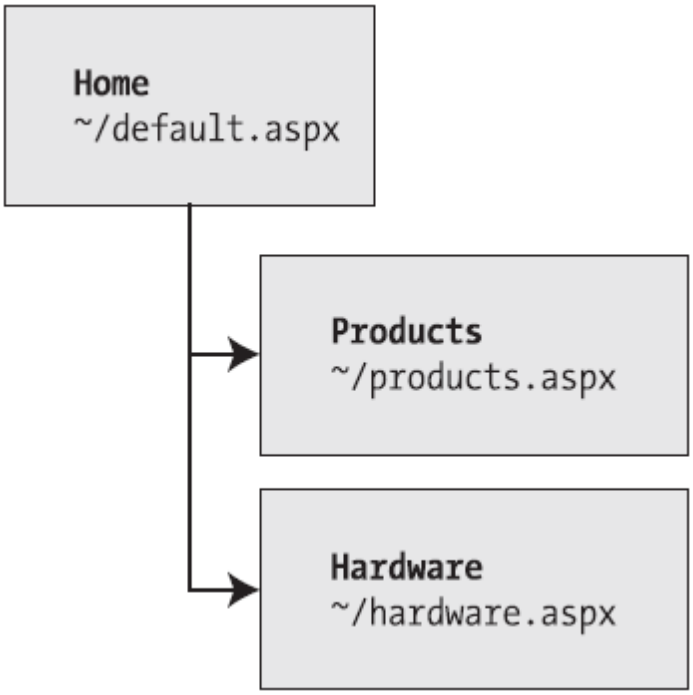
4.2 Ο ρόλος του Web.Config αρχείου

Πίνακας 10. Ο ρόλος του Web.Config αρχείου

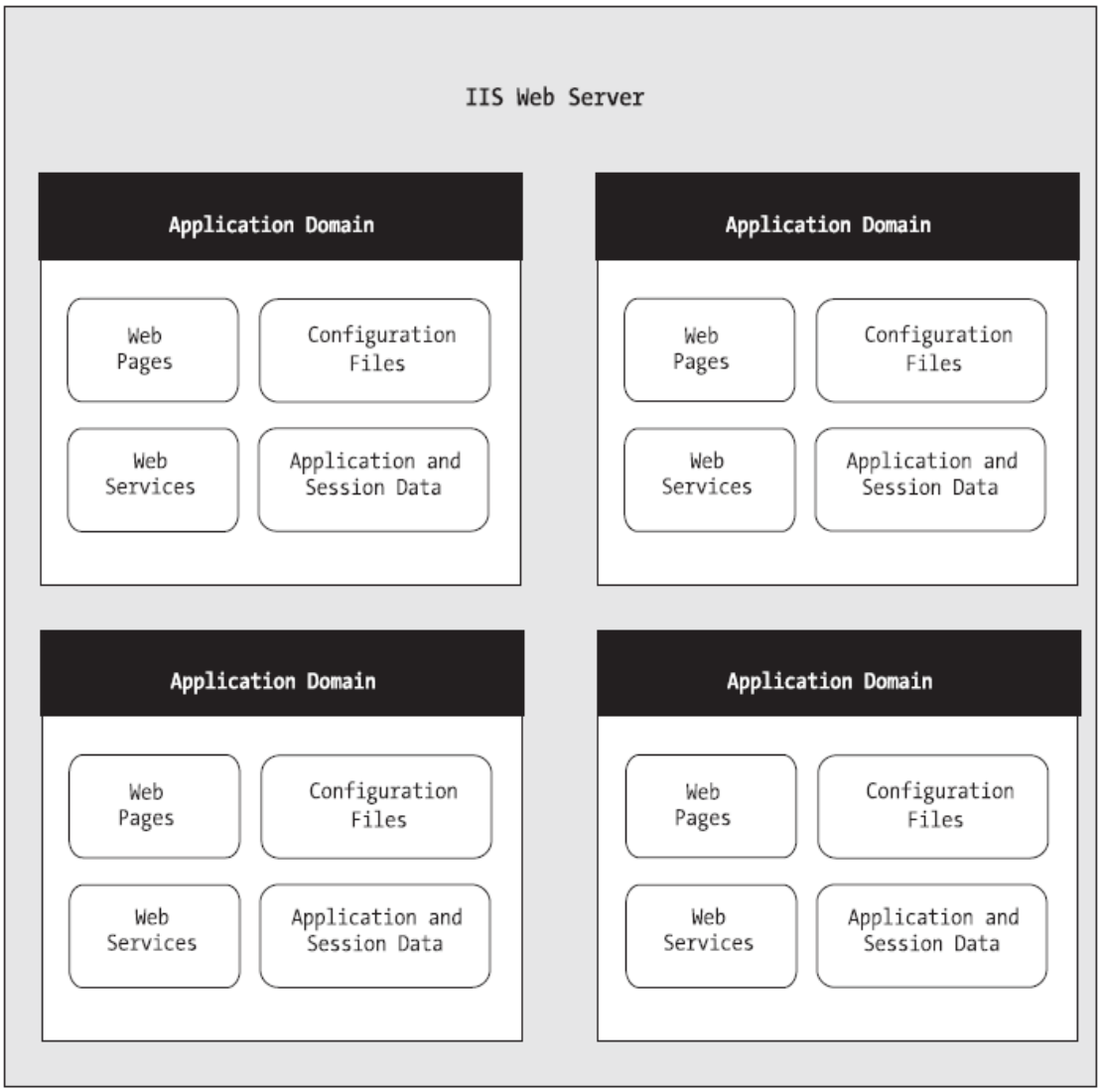
Στοιχείο	Σημασία
<appSettings>	Αυτό το στοιχείο χρησιμοποιείται για την εγκαθίδρυση προσαρμοσμένων ονομάτων ή values που μπορούν να διαβαστούν στη μνήμη για χρήση στην ιστοσελίδα χρησιμοποιώντας το ConfigurationManager τύπο.
<authentication>	Αυτό το στοιχείο ασφαλείας χρησιμοποιείται για να καθορίζει το επίπεδο ασφαλείας και σύνδεσης για αυτή την web εφαρμογή.
<authorization>	Άλλο ένα ασφαλειο-κεντρικό στοιχείο που χρησιμοποιείται για να καθορίσει ποιοί χρήστες έχουν πρόσβαση σε ποιούς πόρους στον Web Server.
<connectionStrings>	Αυτό το στοιχείο χρησιμοποιείται για να κρατάει τα connection strings που χρησιμοποιούνται μέσα σε αυτή την ιστοσελίδα
<customErrors>	Αυτό το στοιχείο χρησιμοποιείται για να αναφέρει στο runtime ακριβώς πώς να εμφανίζει τα σφάλματα που εμφανίζονται κατά την διάρκεια της λειτουργίας της web εφαρμογής.
<globalization>	Αυτό το στοιχείο χρησιμοποιείται για τη ρύθμιση των globalization settings για αυτή την web εφαρμογή
<namespaces>	Το στοιχείο αυτό καταγράφει όλα τα namespaces που περιλαμβάνονται στη web εφαρμογή και έχουν γίνει pre-compile.
<sessionState>	Αυτό το στοιχείο ρυθμίζει το πώς και πού θα αποθηκεύονται τα session data από το .NET Runtime
<trace>	Αυτό το στοιχείο χρησιμοποιείται για να ενεργοποιεί ή να απενεργοποιεί την υποστήριξη για tracing από την web εφαρμογή



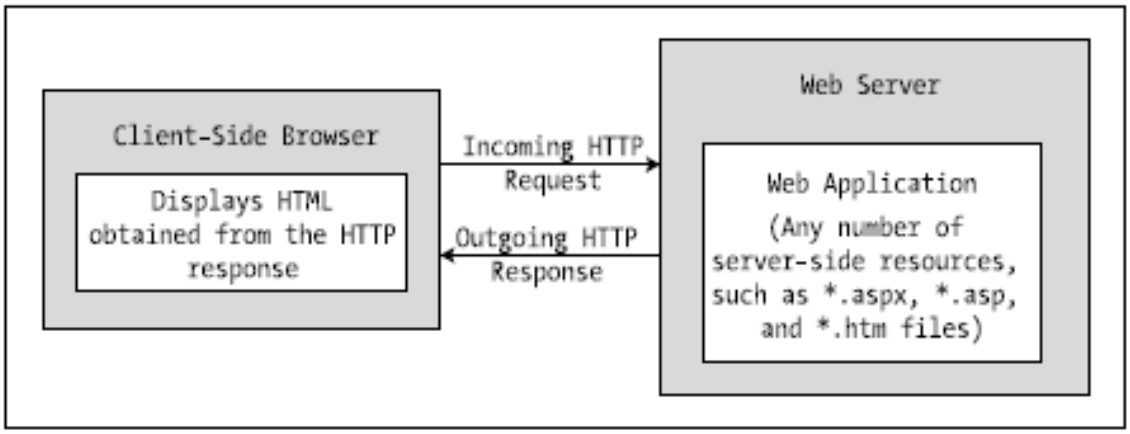
Εικόνα 34. Η ακολουθία επεξεργασίας σελίδας.



Εικόνα 35. Tree nodes σε ένα Asp.NET sitemap.



Εικόνα 36. Εφαρμογές Asp.NET.



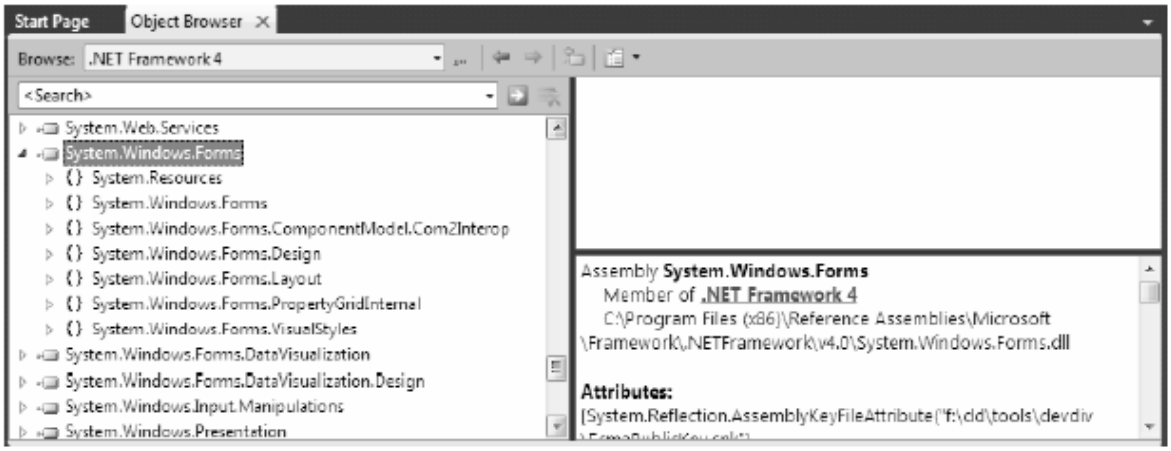
Εικόνα 37. Ο κύκλος HTTP αίτησης/απάντησης.

ΚΕΦΑΛΑΙΟ 5ο

WINDOWS FORMS, STREAMS ΚΑΙ XML

5.1 Προγραμματίζοντας με Windows Forms

Από την πρώτη ημέρα εμφάνισης της .NET πλατφόρμας, οι base class libraries συμπεριέλαβαν ένα συγκεκριμένο api, με την ονομασία Windows Forms. Το Windows Forms Toolkit παρέχει τους κατάλληλους τύπους για τη δημιουργία desktop Graphical User Interfaces (GUIs), τη δημιουργία προσαρμοσμένων στοιχείων ελέγχου, διαχείριση πόρων (π.χ. πίνακες από συμβολοσειρές και εικονίδια) και την ικανότητα εκτέλεσης διεργασιών συσχετιζόμενων με το desktop.



Εικόνα 38. Τα namespaces του System.WIndows.Forms.dll

5.2 Τα Windows Forms Namespaces

Το Windows Forms api, παρέχει τη χρήση εκατοντάδων τύπων, οι περισσότεροι από τους οποίους είναι οργανωμένοι στο assembly System.Windows.Forms.dll

Γενικές κατηγορίες του System.Windows.Forms namespace:

- Core Infrastructure. Τύποι που αναπαριστούν βασικές εφαρμογές ενός Windows Forms προγράμματος και διάφορους τύπους για τη διευκόλυνση της διαλειτουργικότητας με ActiveX controls.
- Controls. Αυτοί οι τύποι χρησιμοποιούνται για τη δημιουργία γραφικών UI (π.χ. κουμπιά, γραμμή μενού, μπάρα προόδου, data grid views), όλα αυτά πηγάζουν από τη βασική κλάση Control. Τα controls είναι διαμορφώσιμα κατά τη σχεδίαση και ορατά κατά το runtime
- Components. Είναι όλοι οι τύποι που δεν αντλούνται από την κλάση Controls, αλλά μπορούν ακόμα να παρέχουν οπτικά χαρακτηριστικά σε ένα Windows Forms πρόγραμμα (πχ. Tooltip, ErrorProvider). Άλλα στοιχεία όπως Timer και BackgroundWorker δεν είναι εμφανίσιμα κατά το runtime αλλά μπορούν να ρυθμιστούν κατά το runtime.
- Common dialog boxes. Παρέχουν αρκετές τυπικές μορφές dialog boxes για κοινές λειτουργίες (όπως OpenFileDialog, PrintDialog και ColorDialog).

Υπάρχει φυσικά η δυνατότητα για τη δημιουργία προσαρμοσμένων παραθύρων ανάλογα με τις ανάγκες τις εφαρμογής.

5.3 .NET και Streams

Η αρχιτεκτονική των .NET streams βασίζεται σε 3 βασικές έννοιες: backing stores, decorators και adapters .Το Backing store είναι το endpoint το οποίο μετατρέπει την είσοδο και την έξοδο σε κάτι χρήσιμο όπως ένα αρχείο ή μια σύνδεση σε ένα δίκτυο.

Για την ακρίβεια είναι είτε ένα από τα δύο ή και τα δύο:

- Πηγή από την οποία τα byte μπορούν να διαβαστούν σειριακά
- Προορισμός από τον οποίο τα byte μπορούν να γραφτούν σειριακά.

Αντίθετα σε σχέση με έναν πίνακα, όλα τα δεδομένα βρίσκονται στη μνήμη μονομιάς, σε μια σειρά και διαχειρίζονται είτε ως ένα byte είτε ως ένα διαχειρίσιμο block μνήμης. Ως εκ τούτου, ένα stream μπορεί να χρησιμοποιήσει λίγη μνήμη, ανεξάρτητα από το μέγεθος του backing store.

Τα stream εμπίπτουν σε δυο κατηγορίες:

- Backing store streams, αυστηρά καθορισμένοι τύποι αποθήκευσης όπως τα FileStream ή NetworkStream.
- Decorator streams, τα οποία «τρέφονται» από άλλα stream, μετατρέποντας τα δεδομένα με κάποιο τρόπο όπως DeflateStream ή CryptoStream

Τα Decorator streams έχουν τα εξής αρχιτεκτονικά προτερήματα:

- Απελευθερώνουν τα backing store streams από την ανάγκη προσθήκης χαρακτηριστικών όπως συμπίεση ή κρυπτογραφία.
- Τα streams δεν εμπίπτουν σε αλλαγές στο interface μετά από χρήση decoration
- Οι decorators μπορούν να συνδεθούν στο runtime.
- Μπορούν να χρησιμοποιηθούν οι decorators μαζί στη σειρά. (π.χ. συμπιεστής και έπειτα κρυπτογράφηση)

Τα backing stores καθώς και τα decorator streams, αποκλειστικά διαχειρίζονται σε μορφή bytes. Αν και αυτό προκαλεί ευελιξία και αποδοτικότητα, υπάρχουν περιπτώσεις που οι εφαρμογές δουλεύουν με υψηλότερου επιπέδου δεδομένα όπως κείμενα ή XML.

Οι adaptors αναλαμβάνουν να γεφυρώσουν το χάσμα μεταξύ ενός απλού byte stream και των δεδομένων υψηλότερου επιπέδου με τον τρόπο του wrapping των streams σε μια κλάση με ειδικευμένες μεθόδους για το συγκεκριμένο format δεδομένων.

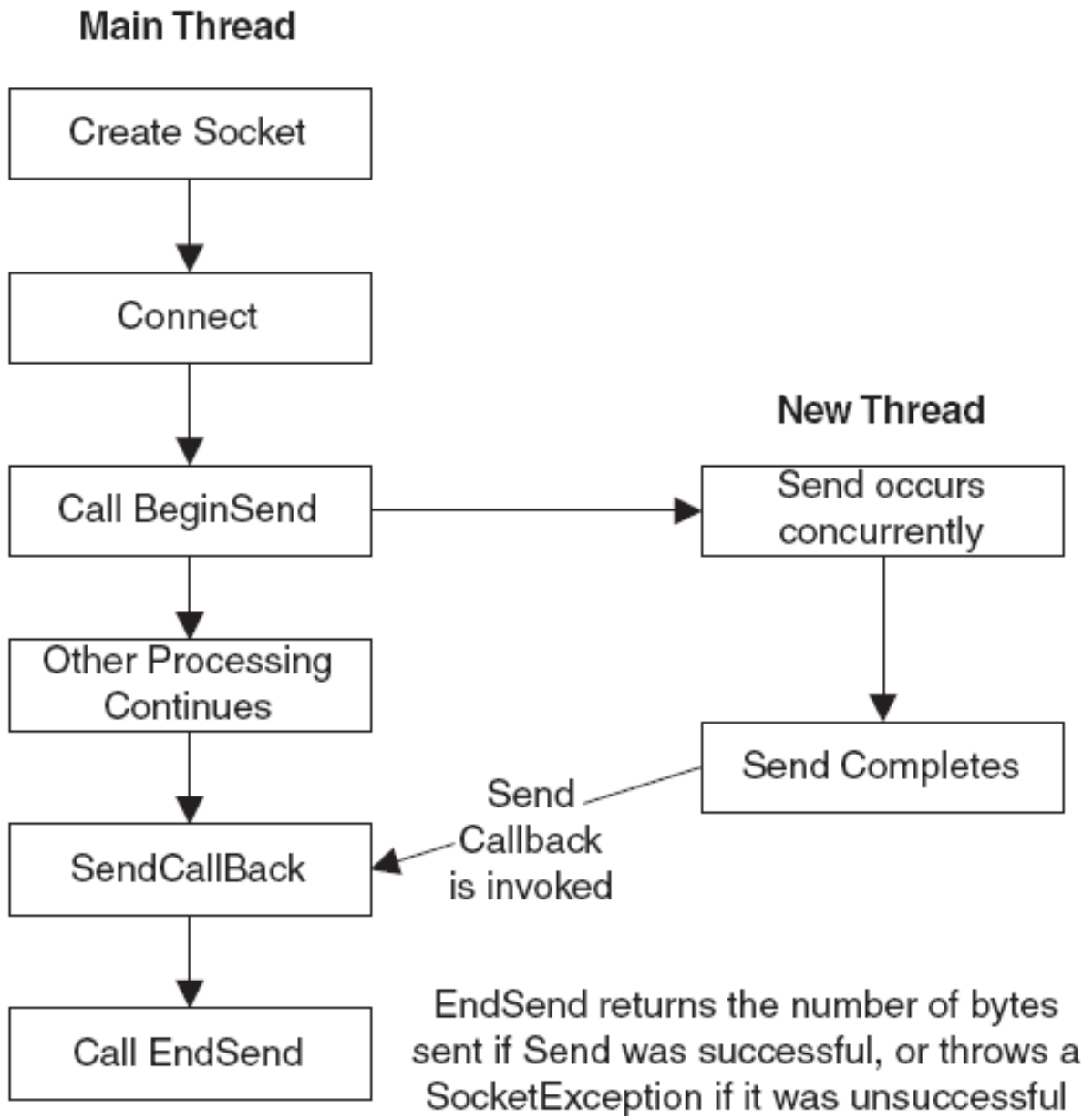
Πίνακας 11. Τυπικά internet protocols

Ακρωνύμιο	Επέκταση	Σημασία
DNS	Domain Name Service	Μετατρέπει τα domain names (πχ ebay.com) σε διευθύνσεις IP (πχ 2.84.13.204)

FTP	File Transfer Protocol	Πρωτόκολλο αποστολής και λήψης αρχείων βασισμένο στο Internet
HTTP	Hypertext Transfer Protocol	Ανακτά ιστοσελίδες και εκτελεί υπηρεσίες διαδικτύου
IIS	Internet Information Services	Web server λογισμικό από τη Microsoft
IP	Internet Protocol	Το επίπεδο Network κάτω από τα TCP και UDP
LAN	Local Area Network	Τοπικό δίκτυο
POP	Post Office Protocol	Ανακτά email μέσω Internet
SMTP	Simple Mail Transfer Protocol	Αποστέλλει email μέσω Internet
TCP	Transmission and Control Protocol	Πρωτόκολλο επιπέδου μεταφοράς πάνω στο οποίο χτίζονται οι περισσότερες υψηλού-επιπέδου υπηρεσίες
UDP	Universal Datagram Protocol	Πρωτόκολλο επιπέδου μεταφοράς για υπηρεσίες με χαμηλό overhead όπως Voice Over IP
UNC	Universal Naming Convention	\\Computer\Sharename\Filename
URI	Uniform Resource Identifier	Σύστημα ονομασίας πόρων στο Internet (π.χ. http://www.amazon.com)
URL	Uniform Resource Locator	Τεχνική σημασία: υποσέτ του URI, κυριολεκτική σημασία: συνώνυμο του URI.

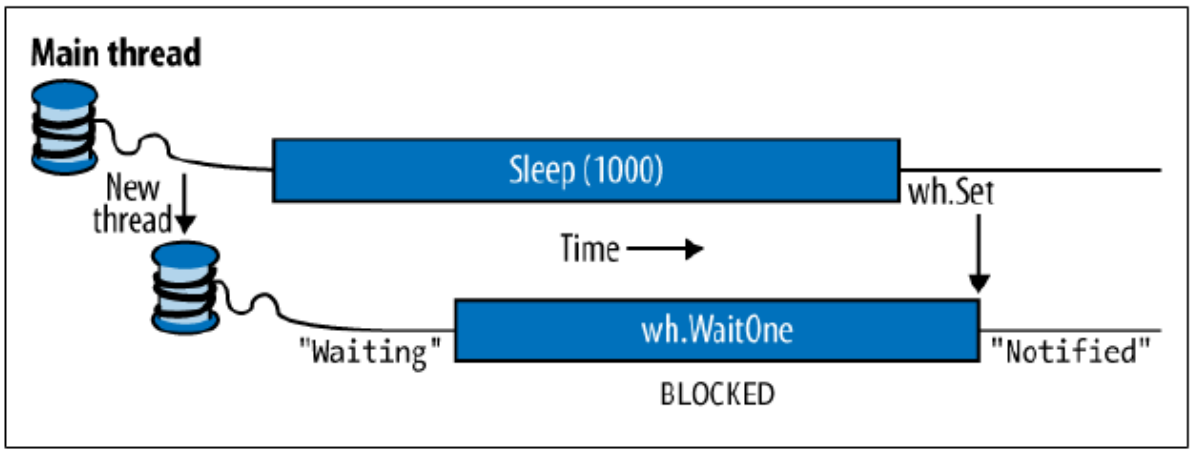
5.4 Γιατί υπάρχουν οι ασύγχρονες μέθοδοι

Υπάρχουν περιπτώσεις που χρειάζεται να οριστούν έξτρα νήματα για την παράλληλη εκτέλεση κάποιων διεργασιών. Στην περίπτωση όμως ενός Tcp ή web server, όπου υπάρχει πιθανότητα να χρειαστεί η εκτέλεση 1000+ ταυτόχρονων client requests, κάτι τέτοιο θα ήταν ασύμφορο. Εάν αφιερωθεί 1 thread σε κάθε εισερχόμενο request, τότε κατά προεπιλογή θα υπήρχε «σπατάλη» ενός gigabyte μνήμης, μόνο για τα thread overheads. Σε αυτή την περίπτωση οι ασύγχρονες μέθοδοι επιλύουν αυτό το πρόβλημα. Μέσα από ένα συγκεκριμένο μοτίβο, πολλές ταυτόχρονες διαδικασίες διαχειρίζονται από λίγα threads. Αυτό επιτρέπει στη συγγραφή thread-efficient προγραμμάτων.



Εικόνα 39.Ένα παράδειγμα asynchronous Send().

Μια ασύγχρονη μέθοδος στοχεύει στο να μην «μπλοκάρει» ποτέ κανένα thread, χρησιμοποιώντας ένα μοτίβο επιστροφής μέσω των callbacks (μπλοκάρισμα εννοείται η εισαγωγή σε μία WaitSleepJoin κατάσταση ή το να προκαλέσει ένα άλλο thread να κάνει το ίδιο, «σπαταλώντας» τον πόρο αυτό). Για να επιτευχθεί κάτι τέτοιο, μια ασύγχρονη μέθοδος πρέπει να απέχει από την κλήση μεθόδου που να προκαλεί το μπλοκάρισμά της.



Εικόνα 40. Το EventWaitHandle.

Υπάρχει όμως και μια εξαίρεση στον κανόνα του μη-μπλοκαρίσματος. Είναι γενικά αποδεκτό το να μπλοκάρει μια μέθοδος ενώ κάνει κλήση προς ένα database server – αν και άλλα threads «συναγωνίζονται» για τον ίδιο server. Αυτό γίνεται επειδή σε ένα υψηλά «ταυτόχρονο» σύστημα, η βάση δεδομένων πρέπει να σχεδιαστεί με τέτοιο τρόπο ώστε η πληθώρα των ερωτημάτων να εκτελείται εξαιρετικά γρήγορα. Εάν όμως φτάσει στο σημείο να γίνονται πολλές χιλιάδες requests ταυτόχρονα από πολλά threads, τότε ο database server πιθανώς να μην μπορεί να τα εξυπηρετήσει. Σε τέτοιες περιπτώσεις το thread economy δεν είναι κανόνας. Παρόμοια, η εγγραφή ή η ανάγνωση σε ένα τοπικό σκληρό δίσκο συνήθως λειτουργεί σωστά με threads που λειτουργούν σειριακά. Σε περίπτωση όμως πολλαπλών ασύγχρονων νημάτων, υπάρχει πιθανότητα το τοπικό file system να μπλοκάρει.

5.5 Τεχνολογία XML

Το System.Xml namespace περιλαμβάνει τα εξής namespaces και κλάσεις:

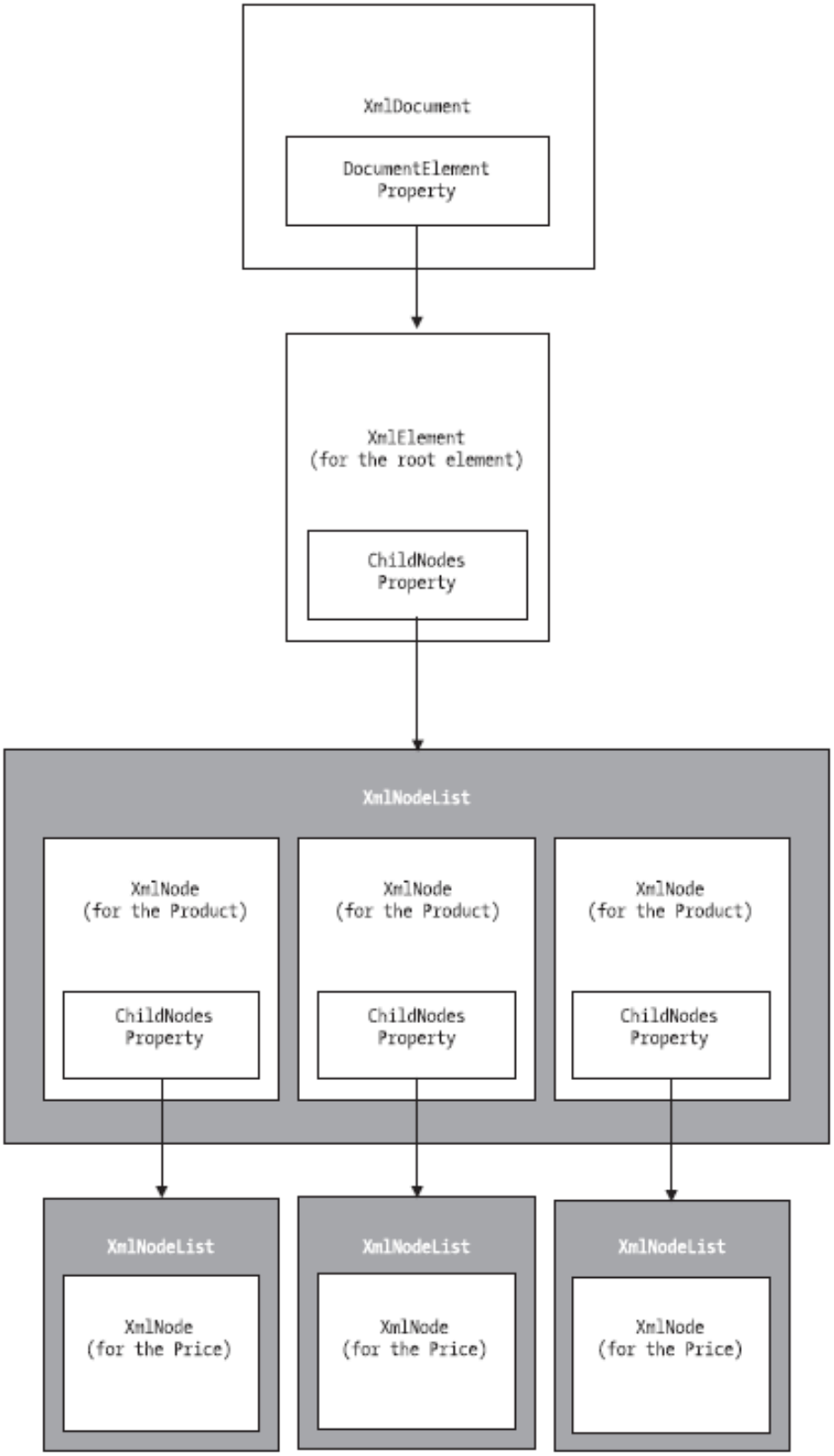
Πίνακας 12. System.Xml Namespace

Στοιχείο	Περιγραφή
System.Xml.* **XmlReader** **XmlWriter** **XmlDocument**	Το XmlReader και XmlWriter περιγράφουν υψηλού επιπέδου streams, υψηλών επιδόσεων, για ανάγνωση ή εγγραφή XML streams. Το XmlDocument αναπαριστά το αρχείο σε μορφή W3C-Style Dom
System.Xml.XPath	Υποδομή και Api για το XPath, μια string-based querying XML γλώσσα.
System.Xml.XmlSchema	Υποδομή και Api για XSD schemas
System.Xml.Xsl	Υποδομή και Api για XSLT μετατροπές από XML
System.Xml.Serialiation	Υποστηρίζει το serialization κλάσεων από και προς XML
System.Xml.XLinq	Μοντέρνα, απλοποιημένη, LINQ-κεντρική έκδοση του XmlDocument

5.6 Διαβάζοντας XML Attributes

Ο XmlReader παρέχει έναν indexer για να αποδίδει άμεση (και τυχαία) προσπέλαση στα γνωρίσματα ενός στοιχείου – είτε βάση ονόματος είτε βάση θέσης.

Χρησιμοποιώντας τον indexer έχουμε το ίδιο αποτέλεσμα που θα είχαμε αν καλούσαμε το GetAttribute.



Εικόνα 41. Ένα XML Document στην μνήμη.

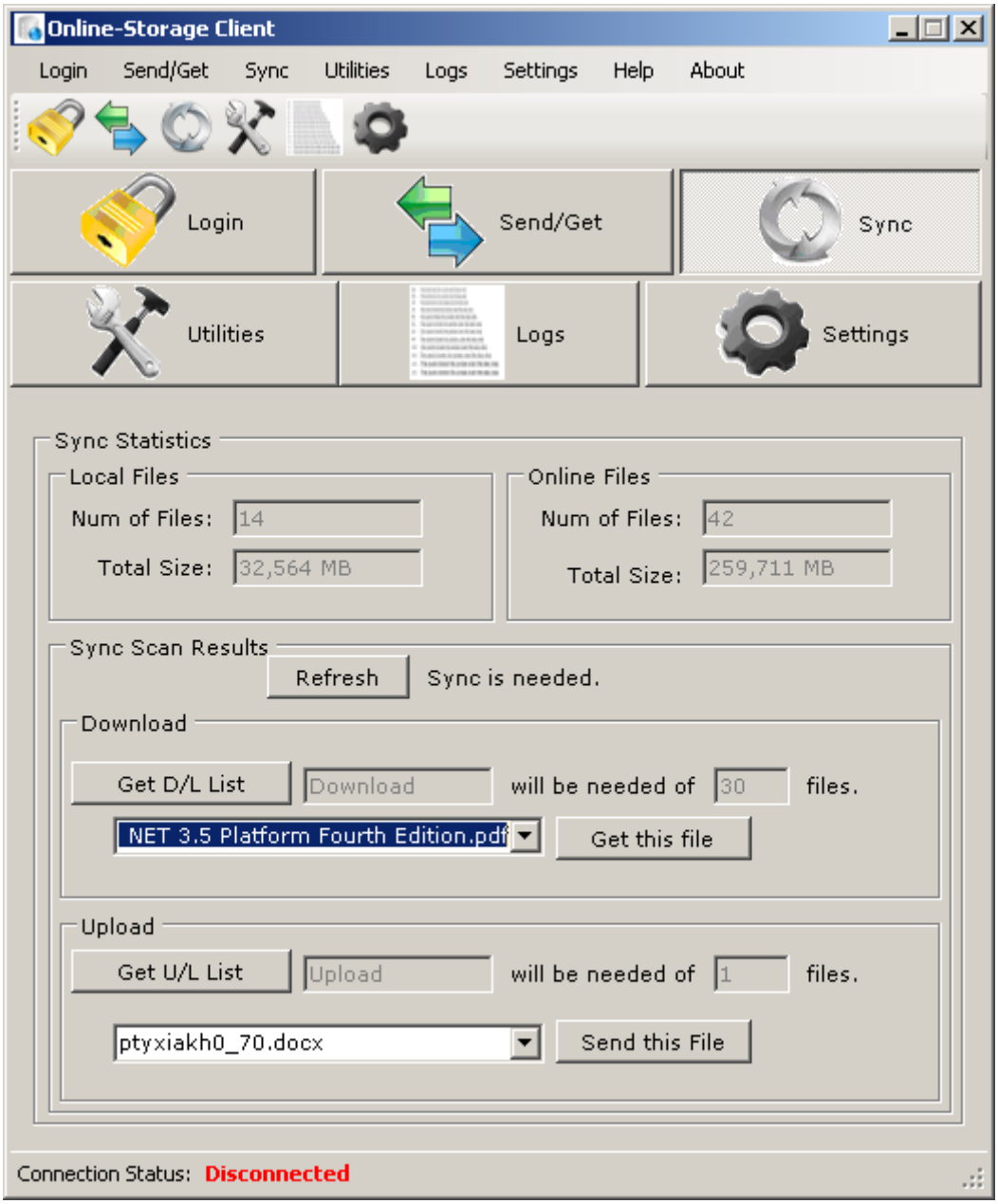
Το W3C είναι μια συντομογραφία του World Wide Web Consortium, όπου μέσω αυτής της κοινοπραξίας καθορίζονται τα standard της XML.

ΚΕΦΑΛΑΙΟ 6ο

ΓΕΝΙΚΗ ΠΕΡΙΓΡΑΦΗ ΠΤΥΧΙΑΚΗΣ

6.1 Γενικά χαρακτηριστικά - δυνατότητες Client εφαρμογής

- Επιλογή online mode (το οποίο λειτουργεί κάνοντας μεταφορά/λήψη messages/αρχείων απο και προς τον server, απλά παίρνοντας την ip απο το domain "online-storage.dyndns.org" στο παρασκήνιο)
- Επιλογή custom mode (για να δουλεύει σε τοπικό δίκτυο αλλά και για περιπτώσεις που ο server έχει μια διαφορετική internet ip απο αυτή του domain)

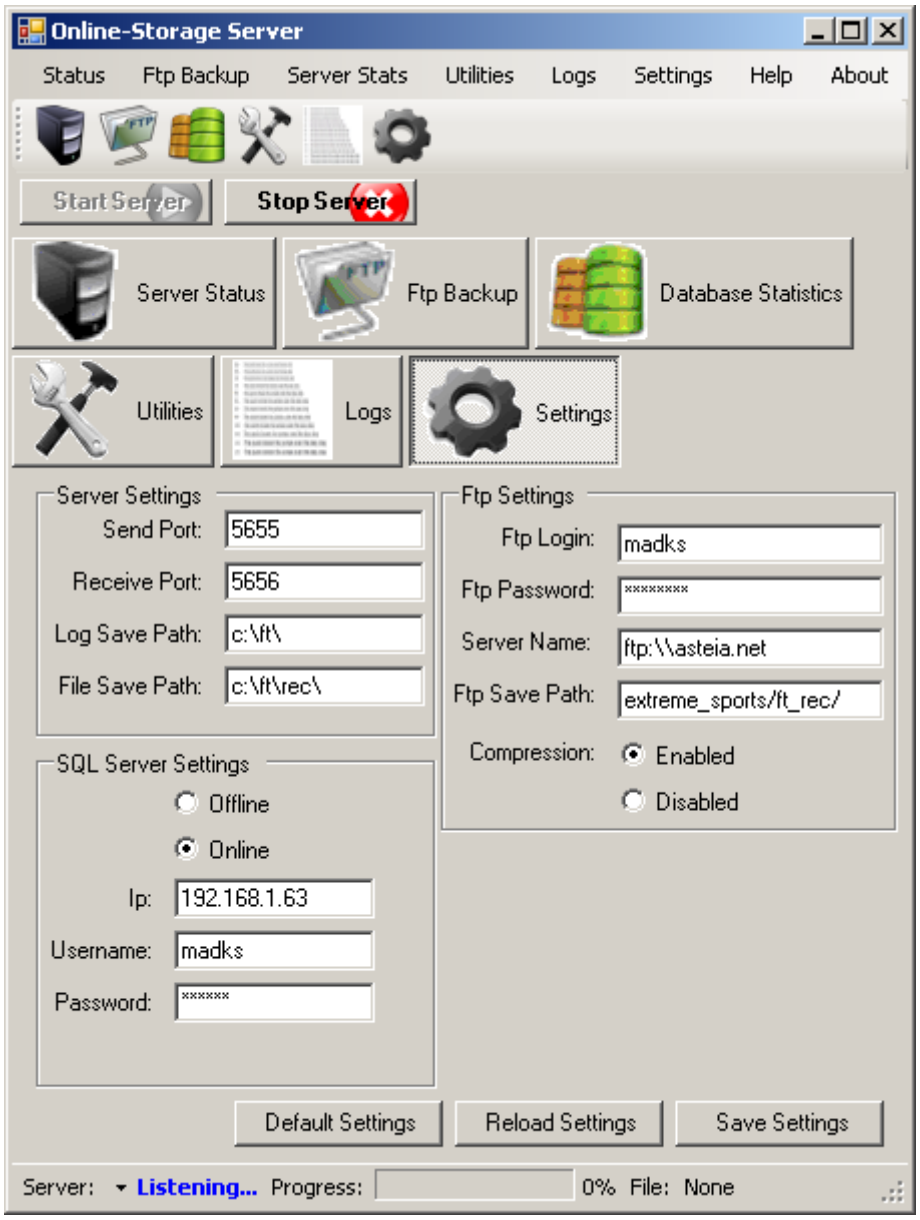


- Η καρτέλα login κάνει σύνδεση με τον server. Το αποτέλεσμα του login εμφανίζεται στην καρτέλα logs (εκεί κυρίως εμφανίζονται τα αποτελέσματα των ενεργειών)
- Η καρτέλα send/get έχει δυο λειτουργίες:
  - ✓ Να στέλνει αρχεία προς τον server.
  - ✓ Να βλέπει τα αρχεία που βρίσκονται στον server. Επίσης μπορεί να γίνει επιλογή για διαγραφή του επιλεγμένου αρχείου, ή λήψη του τοπικά.
- Η καρτέλα Sync συγκρίνει τα τοπικά αρχεία με τα αρχεία που βρίσκονται στον server (δηλαδή τον αριθμό τους και το συνολικό μέγεθος) πατώντας το πλήκτρο Refresh. Αν διαπιστωθεί οτι υπάρχει διαφορά στο πλήθος αλλά και στο συνολικό μέγεθος μεταξύ του local και remote directory, τότε ενεργοποιούνται τα πλήκτρα “Get D/L List” και “Get U/L List”. Εμφανίζονται στα αντίστοιχα combo boxes τα αρχεία που διαφέρουν και πατώντας το πλήκτρο “Get this file” ή “Send this file” λαμβάνονται τοπικά ή αποστέλλονται στον server.
- Η καρτέλα Utilities, παρέχει ip information και ένα προγραμματάκι που χωρίζει σε κομμάτια/ξαναενώνει αρχεία.
- Η καρτέλα Logs εμφανίζει μηνύματα κατάστασης, μεταφοράς και σφαλμάτων γενικά. Έχει την επιλογή καθαρισμού και αποθήκευσης του log.
- Η καρτέλα settings έχει τις global ρυθμίσεις του client προγράμματος. Οι ρυθμίσεις είναι ενεργές σε public μορφή κατά το runtime και αποθηκεύονται/φορτώνονται απο ένα xml αρχείο, το clientSettings.xml Το πλήκτρο “get online ip”, εμφανίζει την τρέχουσα ip του online server.

6.2 Γενικά χαρακτηριστικά - δυνατότητες Server εφαρμογής

- Τα πλήκτρα Start / Stop server.
- Καρτέλα Server Status εμφανίζει κάποια στοιχεία του server, μερικά τα παίρνει απο την βάση δεδομένων, άλλα τα υπολογίζει με διάφορες μεθόδους.
- Καρτέλα FTP Backup:
  - ✓ Τα πλήκτρα compress/decompress συμπιέζουν/αποσυμπιέζουν όλα τα αρχεία στον φάκελο αποθήκευσης του server σε ένα zip αρχείο, το "rec.zip".
  - ✓ Το πλήκτρο Latest Uploaded, εμφανίζει στο text box απο κάτω, το μέγεθος του τελευταίου backup καθώς και το πότε έγινε upload.
  - ✓ Το πλήκτρο D/l ftp backup, κάνει download το ανωτέρω αρχείο αυτόματα, κάνοντας σύνδεση στον ftp server και αποθηκεύοντας το, στον φάκελο αποθήκευσης (όπου μετά με ένα Decompress ολοκληρώνεται το restore)
  - ✓ Το πλήκτρο U/L ftp backup, αποστέλλει αυτόματα το "rec.zip" που υπάρχει ήδη στον φάκελο του server και κάνει connect, upload και disconnect με τον ftp server.
- Καρτέλα Database Utilities εμφανίζει κάποια custom views απο τη βάση δεδομένων.
- Utilities, έχει παρόμοια Utilities με την ίδια καρτέλα στον client.
- Logs όμοια με client

- Lines περιέχει τις ρυθμίσεις για τον server, ftp connection και α server connection για τις global μεταβλητές, ρυθμίσεις οι οποίες αποθηκεύονται/φορτώνονται απο xml αρχείο.



6.3 Το Asp.NET site

- Το web page design έγινε μέσω html/css.
- Η σελίδα εγγραφής περιέχει κώδικα σε c#, η οποία αναλαμβάνει να κάνει Register χρήστες στην κοινή βάση δεδομένων που έχει με την server εφαρμογή.
- Η σελίδα Λήψη έχει τα τελευταία source και project files.


nline
Storage
Αρχική
Εγγραφή
Λήψη
Info
Full Name:
asdf
Username:
asdf
Password:
Re Password:
Passwords don't match!
Address:
Age:
Gender:
Select
Save

ΚΕΦΑΛΑΙΟ 7ο

SERVER SOURCE CODE

7.1 Load server form

Αυτή η μέθοδος εκκινεί με το φόρτωμα της server form. Αυτό που κάνει πρώτο είναι να σιγουρέψει ότι υπάρχουν οι φάκελοι c:\ft\ (ο οποίος είναι ο main φάκελος της εφαρμογής), c:\ft\rec (όπου αποθηκεύονται τα αρχεία που λαμβάνει και στέλνει ο server).

Έπειτα σιγουρεύει ότι υπάρχουν τα αρχεία serverSettings.xml και defaultServerSettings.xml. Σε περίπτωση που δεν υπάρχουν, τα δημιουργεί, σε αντίθετη περίπτωση φορτώνει τις ρυθμίσεις της εφαρμογής από αυτά τα αρχεία.

```
private void Form1_Load(object sender, EventArgs e)
{
    if (!Directory.Exists(@"c:/ft/"))
        Directory.CreateDirectory(@"c:/ft/");
    if (!Directory.Exists(@"c:/ft/rec/"))
        Directory.CreateDirectory(@"c:/ft/rec/");
    if (!File.Exists(@"c:/ft/serverSettings.xml"))
        saveXmlSettings("serverSettings.xml");
    if (!File.Exists(@"c:/ft/defaultServerSettings.xml"))
        saveXmlSettings("defaultServerSettings.xml");

    loadXmlSettings("serverSettings.xml");
}
```

7.2 Refresh server stats

Η μέθοδος αυτή αναλαμβάνει να ανανεώσει τα στατιστικά του Server. Πατώντας το κουμπί “Refresh”, αναλαμβάνει να:

- Μετράει τον αριθμό όλων τον αρχείων στο φάκελο c:\ft\rec
- Ελέγχει εάν υπάρχει σύνδεση με βάση δεδομένων. Αν υπάρχει τότε εκτελεί την μέθοδο numOfUsers.
- Εμφανίζει αν ο server είναι σε Listening ή Offline status.
- Εμφανίζει το συνολικό χρόνο που βρίσκεται σε Listening mode ο server (μέσω της static int μεταβλητής totalUptime, η οποία αυξάνεται κάθε 1 δευτερόλεπτο μέσω ενός timer)

```
public static int totalUptime = 0;
private void button14_Click(object sender, EventArgs e)
{
    int fileCount = Directory.GetFiles(fileSavePath, "*.*",
SearchOption.TopDirectoryOnly).Length; // Will Retrieve count of all files in directry
but not sub directries
    if (useDb == true)
        textBox4.Text = numOfUsers();
    textBox1.Text = toolStripStatusServerStatus.Text;
    textBox3.Text = fileCount.ToString();
    textBox5.Text = (calcFolderFileSize(fileSavePath)/(1024*1024)).ToString("0.00") +
" MB";

    if (totalUptime < 60)
```

```
        textBox2.Text = totalUptime.ToString() + " secs";
    else
        textBox2.Text = (totalUptime / 60).ToString() + " mins " + (totalUptime %
60).ToString() + " secs";
}
```

Η calcFolderFileSize, χρησιμοποιεί τα FileInfo και Directory types για να μετρήσει το μέγεθος τον αρχείων που εμπεριέχονται στον φάκελο c:\ft\rec.

```
private float calcFolderFileSize(string folder)
{
    float folderFileSize = 0;

    try
    {
        //Checks if the path is valid or not
        if (!Directory.Exists(folder))
            return folderFileSize;
        else
        {
            try
            {
                foreach (string file in Directory.GetFiles(folder))
                {
                    if (File.Exists(file))
                    {
                        FileInfo finfo = new FileInfo(file);
                        folderFileSize += finfo.Length;
                    }
                }

                foreach (string dir in Directory.GetDirectories(folder))
                    folderFileSize += calcFolderFileSize(dir);
            }
            catch (NotSupportedException e)
            {
                updateLogText("--Unable to calculate folder size:" + e.Message);
            }
        }
    }
    catch (UnauthorizedAccessException e)
    {
        updateLogText("--Unable to calculate folder size:" + e.Message);
    }

    return folderFileSize;
}
```

Η μέθοδος numOfUsers, η οποία εκτελεί query στη βάση δεδομένων και επιστρέφει τον αριθμό των registered χρηστών που βρίσκονται στον πίνακα ContactInfo.

```
private string numOfUsers()
{
    SqlConnection myConnection = new SqlConnection("user=" + dbUserName + ";" +
"password=" + dbPassword + ";server=" + dbIp + ";" + "Trusted_Connection=no;" +
"database=test_db; " + "connection timeout=3");
    string num_ = "0";
    try
    {
        SqlDataReader myReader = null;
```

```
        SqlCommand myCommand = new SqlCommand("SELECT COUNT(*) AS NumOfUsers FROM
ContactInfo", myConnection);
        myCommand.Connection.Open();
        myReader = myCommand.ExecuteReader();


        while (myReader.Read())
        {
            num_ = myReader["NumOfUsers"].ToString();
        }
    }
    catch (Exception ex)
    {
        updateLogText("--Error. " + ex.Message + "\r\n");
    }

    myConnection.Close();
    return num_;
}
```

7.3 Start server

Το κουμπί cmdStartServer εκτός από τον timer1 (ο οποιος κάνει count το uptime κάθε 1 δευτερόλεπτο) ξεκινάει με ένα βρόχο επανάληψης do-while, ο οποίος επαναλαμβάνεται επ' άπειρον (μέχρι τουλάχιστον η Boolean μεταβλητή flag_end να πάρει την τιμή true, που σημαίνει δηλαδή ότι έχει πατηθεί το πλήκτρο Stop server).

```
private void cmdStartServer_Click(object sender, EventArgs e)
{
       timer1.Start();
       do
       {
              flag_end = false;
              listen();

       }while (flag_end == true);
}
```

Ξεκινάει την μέθοδο listen(), η οποία είναι και η κύρια μέθοδος με την οποία ο server μπαίνει σε "Listening" mode και αφουγκράζεται τα εισερχόμενα μηνύματα, files και requests.

```
private void timer1_Tick(object sender, EventArgs e)
{
       totalUptime++;
}
```

Η μέθοδος listen(), αρχικώς ελέγχει και σβήνει αν υπάρχουν τα προσωρινά αρχεία cache.bak, dir.msg και message.msg, για αποφυγή conflict με μεταγενέστερες διεργασίες που πιθανόν να τα χρησιμοποιήσουν.

Έπειτα ενεργοποιεί το manual reset event allDone.Set(), το οποίο θέτει το thread σε κατάσταση ετοιμότητας, έτοιμο να ξεκινήσει. Δημιουργείται ένα IPEndpoint το οποίο δέχεται οποιαδήποτε ip στην port 5656 το οποίο γίνεται bind πάνω στο socket sock, τύπου tcp.

Με την εκκίνηση της τεχνικής try-catch, το socket εισέρχεται σε listening mode και αναμένει εισερχόμενα connections. Το manual reset event allDone.Reset() φέρνει το thread σε κατάσταση “block” αναμένοντας εισερχόμενο connection.

Σε περίπτωση που ξεκινήσει η διαδικασία για το connection με το socket, καλείται η μέθοδος AcceptCallback.

```
private void listen()
{
    // deleting cache
    if (File.Exists(@"c:\FT\rec\cache.bak"))
        File.Delete(@"c:\FT\rec\cache.bak");
    if (File.Exists(@"c:\FT\rec\dir.msg"))
        File.Delete(@"c:\FT\rec\dir.msg");
    // deleting message
    if (File.Exists(@"c:\FT\rec\message.msg"))
        File.Delete(@"c:\FT\rec\message.msg");

    allDone.Set();
    cmdStartServer.Enabled = false;
    cmdStopServer.Enabled = true;
    toolStripStatusServerStatus.Text = "Listening...";
    toolStripStatusServerStatus.ForeColor = System.Drawing.Color.Blue;

    string tempPort = txtPort.Text;
    txtServerLog.Text += "--Initializing Server...\r\n";
    IPEndPoint ipEnd = new IPEndPoint(IPAddress.Any, Int32.Parse(tempPort));
    sock = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);

    try
    {
        sock.Bind(ipEnd);
        //start Listening...

        sock.Listen(100);
        allDone.Reset();

        //lblStatus.Text = "Listening..."; //original
        toolStripStatusServerStatus.ForeColor = System.Drawing.Color.Blue;
        toolStripStatusServerStatus.Text = "Listening...";
        txtServerLog.Text += "--Listening...\r\n";
        cmdStartServer.Enabled = false;

        //cmdStartServer.Enabled = false;

        //sock.BeginAccept(new AsyncCallback(SocketListener.acceptCallback),sock);
        sock.BeginAccept(new AsyncCallback(AcceptCallback), sock);

        //allDone.WaitOne();

    }
    catch (Exception ex)
    {
        txtServerLog.Text += "--File Receiving fail." + ex.Message + "\r\n";
    }

}
```

Η μέθοδος AcceptCallback ξεκινάει να λαμβάνει δεδομένα από το client socket, αφού δώσει σήμα στο main thread να συνεχίσει. Εκκινεί ένα instance της κλάσης StateObject και έπειτα καλεί την μέθοδο BeginReceive. Η μέθοδος αυτή διαβάζει τα δεδομένα από το client socket ασύγχρονα, χωρίς να μπλοκάρει το thread της εφαρμογής.

```
public void AcceptCallback(IAsyncResult ar)
{
    // Signal the main thread to continue.
    updateLogText("------AcceptCallback------");
    updateLogText("--Inter-thread communication Set(), main thread continues");

    allDone.Reset();

    updateLogText("--Getting the socket that handles the client request.");

    Socket listener = (Socket)ar.AsyncState;
    Socket handler = listener.EndAccept(ar);

    updateLogText("--Microsoft .EndAccept builtin function just received the data in a
new socket.");

    updateLogText("--Creating the state object.");
    StateObject state = new StateObject();

    state.workSocket = handler;

    updateLogText("--Socket starts receiving data through the state object.");
    handler.BeginReceive(state.buffer, 0, StateObject.BufferSize, 0, new
AsyncCallback(ReadCallback), state);
}
```

Η μέθοδος ReadCallBack ανακτά το state object και το handler socket από το ασύγχρονο αντικείμενο “ar”. Αποθηκεύει το πλήθος των byte που έλαβε από το client socket στην integer μεταβλητή bytesRead. Με μια συνθήκη if, γίνεται έλεγχος για το αν τα δεδομένα που έχουν ληφθεί είναι μηδενικά. Αν δεν είναι, τότε ξεκινάει η διαδικασία αποθήκευσής τους, στην συγκεκριμένη περίπτωση στο αρχείο cache.bak μέσω του filestream fs. Έπειτα, καλείται αναδρομικά ξανά η μέθοδος BeginReceive, ώστε να λάβει και τα υπόλοιπα δεδομένα από το client socket.

Αν το bytesRead έχει την τιμή μηδέν, τότε σημαίνει ότι έχουν ληφθεί όλα τα δεδομένα από το client socket. Το αρχείο cache.bak κλείνει, δημιουργείται ένα byte array μεγέθους όσο το αρχείο cache.bak. Αποθηκεύεται το όνομα αρχείου μαζί με το extension στην string μεταβλητή filename (το οποίο βρίσκεται στα πρώτα 4 byte του byte array).

Μια δεύτερη συνθήκη if, ελέγχει για το όνομα του αρχείου. Εάν είναι τύπου “message.msg”, σημαίνει ότι το δεν πρόκειται για αρχείο αλλά το ότι ο client θέλησε να στείλει ένα μήνυμα στον server.

Στη περίπτωση που όντως το αρχείο είναι τύπου message, τότε το αρχείο γίνεται load σε ένα string και έπειτα διασπάται σε 3 τμήματα, το action, το usr και το password. Η μεταβλητή action ελέγχεται με άλλο ένα βρόχο if. Στην περίπτωση που:

- **Ισούται με "login":** σημαίνει ότι ο client αιτείται σύνδεσης με τον server. Αν ο server είναι συνδεμένος με την βάση δεδομένων (δηλαδή το useDb έχει την τιμή true), τότε καλείται η μέθοδος authenticateUser. Εάν το αποτέλεσμα της μεθόδου αυτής είναι αληθές, τότε ο client θεωρείται logged in.
- **Ισούται με "dir":** σημαίνει ότι ο client αιτείται της εντολής dir, δηλαδή θέλει να ενημερωθεί για τα περιεχόμενα του κύριου φάκελου αποθήκευσης στον server. Καλείται η μέθοδος send_dir() η οποία εκτελεί την αντίστοιχη διεργασία, δηλαδή στέλνει μήνυμα πίσω στον client με τα περιεχόμενα του φακέλου.
- **Ισούται με "get":** σημαίνει ότι ο client έχει επιλέξει ένα συγκεκριμένο αρχείο από τον φάκελο του server και επιθυμεί να το λάβει. Ο server το στέλνει πίσω μέσω της μεθόδου send_file(psw), όπου στην θέση του string psw είναι αποθηκευμένο το filepath του αρχείου τοπικά.
- **Ισούται με "del":** σημαίνει ότι ο client έχει επιλέξει ένα συγκεκριμένο αρχείο από τον φάκελο του server και επιθυμεί να το σβήσει. Μέσω της File.Delete(psw), σβήνεται το αρχείο που βρίσκεται στον τοπικό φάκελο του string psw.
- **Ισούται με "size":** σημαίνει ότι ο client επιθυμεί να μάθει το μέγεθος και το πλήθος όλων των αρχείων στον κύριο φάκελο του server. Ο server επιστρέφει την απάντηση σε KB μέσω μηνύματος πίσω στον client κάνοντας χρήση της μεθόδου send_size(filecount, size).

Εάν το αρχείο δεν είναι τύπου message, σημαίνει ότι είναι κανονικό αρχείο. Ένας binary writer, αποθηκεύει το αρχείο που βρίσκεται στον buffer byteArray, με index = 4 + το μήκος της ονομασίας του αρχείου και αποθηκεύει τα length - 4 - fileNameLen bytes απο τον buffer (δηλαδή τα byte που περιέχουν καθαρά το αρχείο και όχι τις πληροφορίες που εστάλησαν απο το network stream)

Αφού αποθηκεύτηκε το αρχείο, αν διαπιστωθεί ότι ο server είναι συνδεδεμένος με την βάση δεδομένων, τότε κάνει log διάφορες πληροφορίες για τον νεοαποκτηθέν αρχείο στη db, μέσω της μεθόδου logFiles. Το byteArray αδειάζει και τότε καλείται ο garbage collector. Η διεργασία λήψης του αρχείου, τίθεται σε κατάσταση wait, με την χρήση του manual reset event. Έπειτα κλείνει το socket και του client και του server (μέσω της handler.Shutdown(SocketShutdown.Both))), κλείνουν τα sock, handler sockets και γίνεται χειροκίνητη απελευθέρωση των πόρων που είχε χρησιμοποιήσει το state object.

Η serverRestart() επειδή βρίσκεται σε άλλο thread, γίνεται η κλήση της μέσω της delegate μεθόδου delegate void serverRestartDelegate(); ή οποία μέσω του cmdStartServer.Invoke((serverRestartDelegate)clickStart); επιτρέπει την επανεκκίνηση του server.

```
public void ReadCallback(IAsyncResult ar)
{

    // Retrieve the state object and the handler socket
    // from the asynchronous state object.
    StateObject state = (StateObject)ar.AsyncState;//original
    Socket handler = state.workSocket;
```

```
    // Read data from the client socket.
    int bytesRead = handler.EndReceive(ar);

    if (bytesRead > 0)
    {
        // There might be more data, so store the data received so far.
        //state.state.sb.Write(state.buffer, 0, bytesRead);//original
        //state.sb.Write(state.buffer, 0, bytesRead);
        FileStream fs = new FileStream(@"c:\FT\rec\cache.bak", FileMode.Append);
        Int32 length = Int32.Parse(fs.Length.ToString());
        fs.Close();

        /*
        FileStream fileStream = new FileStream(@"c:\FT\rec\cache.bak",
FileMode.Append);//test
        fileStream.Write(state.buffer, 0, bytesRead);//test
        fileStream.Close();                     */

        BinaryWriter bWrite2 = new BinaryWriter(File.Open(@"c:\FT\rec\cache.bak",
FileMode.Append)); ;
        bWrite2.Write(state.buffer, 0, bytesRead);//test
        bWrite2.Close();

        if (length > 1000000)
        {
            //MessageBox.Show("Big: Biger than 1MB, length = " + length);//test
            //updateLogText("--File bigger than 1Mb, using hard disk cache");
            /*
            FileStream fileStream = new FileStream(@"c:\FT\rec\cache.bak",
FileMode.Append);
            fileStream.Write(state.buffer, 0, bytesRead);
            fileStream.Close();
                */
        }

        // Get the rest of the data.
        handler.BeginReceive(state.buffer, 0, StateObject.BufferSize, 0, new
AsyncCallback(ReadCallback), state);

        toolStripProgressBar1.Value = 0;

    }
    else
    {
        // All the data has arrived; put it in response.
        FileStream fs = new FileStream(@"c:\FT\rec\cache.bak", FileMode.Open);
        Int32 length = Int32.Parse(fs.Length.ToString());
        fs.Close();

        if (length > 1)
        {
            if (length > 100000000)
                MessageBox.Show("Huge: Biger than 100MB, length = " + length);//test

            // Set the position to the beginning of the stream.

            fs = new FileStream(@"c:\FT\rec\cache.bak", FileMode.Open);
            Int32 cacheLen = Int32.Parse(fs.Length.ToString());

            byte[] byteArray = new byte[cacheLen];
            fs.Read(byteArray, 0, cacheLen);
            fs.Close();
```

```
            int fileNameLen = BitConverter.ToInt32(byteArray, 0); // file name length
with extension

            string fileName = Encoding.ASCII.GetString(byteArray, 4, fileNameLen); //
the file name with extension
            updateLogText("--filename= " + fileName);

            if (fileName == "message.msg")
            {
                // deleting cache
                if (File.Exists(@"c:\FT\rec\cache.bak"))
                    File.Delete(@"c:\FT\rec\cache.bak");

                updateLogText("--Message not file " + fileName + "--");
                string message = Encoding.ASCII.GetString(byteArray, 4 + fileNameLen,
length - 4 - fileNameLen);
                string[] credentials = message.Split('@');
                string action = credentials[0];
                string usr = credentials[1];
                string psw = credentials[2];

                // deleting message
                if (File.Exists(@"c:\FT\rec\message.msg"))
                    File.Delete(@"c:\FT\rec\message.msg");

                updateLogText("--Message from client: " + message.ToString());
                updateLogText("--Username: '" + usr + "' password: '" + psw + "' .
Client wants to... " + action);

                if (action == "login")                                  {
                    // try to login
                    // if it logs
                    // global bool loged = true
                    // it can file transfer

                    /* test authenticate start*/
                    if (useDb == true)
                        authenticateUser(usr, psw);//test
                    //if (usr == "madks" && psw == "121284")//original

                    if (flag == true)
                    {
                        loged = true;
                        updateLogText("--Login ok");
                    }
                    else
                    {
                        updateLogText("--Failed to login");
                    }
                    /* test authenticate end*/
                }
                else if (action == "dir")
                {
                    //MessageBox.Show("Opening send socket, and sending dir msg");

                    //MessageBox.Show("psw = '" + psw + "'");

                    send_dir();
                    // deleting message
                    if (File.Exists(@"c:\FT\rec\message.msg"))
                        File.Delete(@"c:\FT\rec\message.msg");
```

```
                    if (File.Exists(@"c:\FT\rec\dir.msg"))
                        File.Delete(@"c:\FT\rec\dir.msg");
                }
                else if (action == "get")
                {
                    //MessageBox.Show("Opening send socket, and sending file");

                    //MessageBox.Show("psw = '" + psw + "'");

                    send_file(psw);
                    // deleting message
                    if (File.Exists(@"c:\FT\rec\message.msg"))
                        File.Delete(@"c:\FT\rec\message.msg");
                    if (File.Exists(@"c:\FT\rec\dir.msg"))
                        File.Delete(@"c:\FT\rec\dir.msg");

                }
                else if (action == "del")
                {
                    //MessageBox.Show("Opening send socket, and sending dir msg");

                    //MessageBox.Show("Deleting path '" + psw + "'");
                    // deleting message

                    if (File.Exists(psw))
                        File.Delete(psw );


                    // deleting message
                    if (File.Exists(@"c:\FT\rec\message.msg"))
                        File.Delete(@"c:\FT\rec\message.msg");
                    if (File.Exists(@"c:\FT\rec\dir.msg"))
                        File.Delete(@"c:\FT\rec\dir.msg");
                }
                else if (action == "size")
                {

                    int fileCount = Directory.GetFiles(fileSavePath, "*.*",
SearchOption.TopDirectoryOnly).Length; // Will Retrieve count of all files in directry
but not sub directries

                    //textBox3.Text = fileCount.ToString();
                    //textBox5.Text = (calcFolderFileSize(fileSavePath) / (1024 *
1024)).ToString("0.00") + " MB";

                    send_size(fileCount, (calcFolderFileSize(fileSavePath) / (1024 *
1024)));

                }
            }
            else
            {

                string receivePath = "c:/FT/rec/"; // the directory to save the file

                //updateLogText("--Final Capacity='" + capacity + "'");

                updateLogText("--Final Length='" + length + "'");
                updateLogText("--Final Position='" + state.sb.Position.ToString() +
"'");
                updateLogText("Filename: " + fileName + " ...");
```

```
                BinaryWriter bWrite = new BinaryWriter(File.Open(receivePath +
fileName, FileMode.Append)); ;
                bWrite.Write(byteArray, 4 + fileNameLen, length - 4 - fileNameLen);
                bWrite.Close();
                updateLogText("--File saved at " + receivePath + fileName);

                //logFiles(Boolean virusChecked, string title, DateTime modifiedDate,
string fileName, string fileExtension, int size, string description)

                FileInfo fInfo = new FileInfo(@receivePath+fileName);
                long megethosArxeiou = fInfo.Length;// gia meta
                if (useDb == true)
                    logFiles(false, "Titlos Arxeiou", DateTime.Now,
Path.GetFileNameWithoutExtension(fileName), Path.GetExtension(fileName),
megethosArxeiou, "Perigrafh Arxeiou", receivePath);
                // logging apothikeushs arxeioy End

                if (File.Exists(@"c:\FT\rec\cache.bak"))
                    File.Delete(@"c:\FT\rec\cache.bak");

                byteArray = new byte[0];
                byteArray = null;
                GC.Collect();
            }

        }

        //Free byte[] array test
        // Signal that all bytes have been received.
        state.buffer = null;
        allDone.Reset();
        updateLogText("--All bytes received.");

        // logging apothikeushs arxeioy


        updateLogText("--Closing the sockets");
        handler.Shutdown(SocketShutdown.Both);
        handler.Close();
        state.sb.Flush();
        state.sb.Close();
        state.sb.Dispose();
        state.free();
        updateLogText("--Closing the handler");
        updateLogText("--Server Closed");
        //toolStripStatusServerStatus.Text = "Offline";//original
        //toolStripStatusServerStatus.ForeColor = System.Drawing.Color.Red;
        sock.Close();

        updateLogText("--Trying to revive the server...1");

        // deleting message
        if (File.Exists(@"c:\FT\rec\message.msg"))
            File.Delete(@"c:\FT\rec\message.msg");

        serverRestart();              //test option 1
        //listen();                     //test option 2
        //cmdStartServer.PerformClick();  //test option 3

    }
    enableCmdStartServer();
    //clickCmdStartServer();
```

}

7.4 Compress files

Με το πάτημα του κουμπιού “Compress Files”, εκκινείται η εξής διαδικασία:

- Γίνεται έλεγχος αν υπάρχει στον κύριο φάκελο αποθήκευσης το αρχείο “rec.zip”. Αν υπάρχει τότε διαγράφεται διότι προέρχεται από παλαιότερο backup.
- Αποθηκεύονται στο array filenames τα ονόματα όλων των αρχείων που υπάρχουν στον φάκελο C:\ft\rec
- Γίνεται ορισμός του ZipOutputStream s και δημιουργείται το αρχείο rec.zip στον φάκελο του server (το ZipOutputStream είναι στοιχείο της library ICSharpCode.SharpZipLib.dll)
- Ορίζεται το level of compression (9 είναι το μέγιστο)
- Για κάθε string (δηλαδή για κάθε αρχείο που υπάρχει στον φάκελο αποθήκευσης) αποθηκεύεται σε μια μεταβλητή integer το μήκος του αρχείου. Έπειτα γίνεται εγγραφή/πρόσθεση στο zip stream s όλα τα bytes του τρέχοντος αρχείου (που είναι αποθηκευμένα στον buffer “buffer”, με index 0 και μήκος bytes ίσο με sourceBytes)
- Όταν το sourceBytes έχει την τιμή μηδέν, τότε σημαίνει ότι έχουν προστεθεί όλα τα bytes του συγκεκριμένου αρχείου στο αρχείο zip.rec. Το loop συνεχίζεται και για το επόμενο αρχείο που υπάρχει στο array “filenames”.
- Όταν προστεθούν όλα τα αρχεία στο rec.zip, καλούνται οι μέθοδοι .finish() και .close() για το ZipOutputStream s.

-

```
private void button6_Click(object sender, EventArgs e)
{
    try
    {
        //lblUpdate.Visible = true;
        //lblUpdate.Refresh();
        // Set up a string to hold the path to the temp folder
        string sTargetFolderPath = ("c:\\FT\\rec");
        string sZipFileName = "rec";

        if (File.Exists(@"c:\FT\rec\rec.zip"))
            File.Delete(@"c:\FT\rec\rec.zip");
        /*
        // Zip up the files - From SharpZipLib Demo Code
        if (File.Exists("c:\\FT\\rec" + "\\" + sZipFileName + ".zip"))
            File.Delete("c:\\FT\\rec" + "\\" + sZipFileName + ".zip");
        */
        string[] filenames = Directory.GetFiles(sTargetFolderPath);

        using (ZipOutputStream s = new ZipOutputStream(File.Create("c:\\FT\\rec" +
"\\" + sZipFileName + ".zip")))
        {
            s.SetLevel(9); // 0-9, 9 being the highest level of compression

            byte[] buffer = new byte[4096];

            int i = 0;
            foreach (string file in filenames)
```

```
            {
                toolStripStatusLabel4.Visible = true;
                toolStripStatusLabel3.Visible = true;
                toolStripStatusLabel5.Visible = true;
                //toolStripProgressBar1.Value += 10;
                toolStripStatusLabel4.Text = toolStripProgressBar1.Value.ToString() +
"%";
                updateLogText("--Compressing file" + filenames[i].ToString() +
"\r\n");
                i++;

                ZipEntry entry = new ZipEntry(Path.GetFileName(file));

                entry.DateTime = DateTime.Now;
                s.PutNextEntry(entry);

                using (FileStream fs = File.OpenRead(file))
                {
                    int sourceBytes;
                    do
                    {
                        sourceBytes = fs.Read(buffer, 0, buffer.Length);
                        s.Write(buffer, 0, sourceBytes);

                    } while (sourceBytes > 0);
                }
            }

            //toolStripProgressBar1.Value = 100;
            toolStripStatusLabel4.Text = "100%";
            s.Finish();
            s.Close();
        }

        // Notify user
        updateLogText("--Zip file " + "c:\\FT\\rec" + " created.\r\n");

    }
    catch (Exception ex)
    {
        updateLogText("Zip Operation Error. "+ ex.Message.ToString());
    }

}
```

7.5 Decompress

Με το πάτημα του κουμπιού “Decompress Files”, εκκινείται η εξής διαδικασία:

- Ορίζεται το string password ίσο με null και έπειτα ένα νέο ZipInputStream με ονομασία s.
- Το s.Password παίρνει την τιμή του string password, δηλαδή null.
- Εκκινεί ο βρόχος επανάληψης While για κάθε entry που υπάρχει στο ZipInputStream s (που μέσω της μεθόδου .GetNextEntry() αποθηκεύεται σε μια προσωρινή μεταβλητή theEntry).
- Αν η theEntry έχει τιμή διάφορη του null, τότε ελέγχεται αν το directoryName έχει διαφορετική τιμή του “” (δηλαδή του κενού). Σε καταφατική περίπτωση, δημιουργείται ο αντίστοιχος φάκελος.

- Ελέγχεται το string “filename”. Αν δεν είναι empty, ελέγχεται πάλι για το αν υπάρχει το directory του filename και έπειτα μέσω ενός fileStream εκκινείται η διαδικασία αποσυμπίεσης του αρχείου.
- Για την αποσυμπίεση ορίζεται ο buffer “data”. Για όσο το size>0, ο filestream writer αποθηκεύει τα δεδομένα που βρίσκονται στον buffer, με index 0 και πλήθος size bytes. Μετά ο stream writer κλείνει το αρχείο.
- Όταν τελειώσει η αποσυμπίεση όλων των αρχείων, κλείνει το στοιχείο ZipInputStream.

```
private void button7_Click(object sender, EventArgs e)
//public static void UnZipFiles(string zipPathAndFile, string outputFolder, string
password, bool deleteZipFile)
{
    string zipPathAndFile = "c:\\FT\\rec\\rec.zip";
    string outputFolder = "c:\\FT\\rec\\";
    bool deleteZipFile = false;
    string password = null;
    ZipInputStream s = new ZipInputStream(File.OpenRead(zipPathAndFile));

    if (password != null && password != String.Empty)
        s.Password = password;
    ZipEntry theEntry;
    string tmpEntry = String.Empty;
    while ((theEntry = s.GetNextEntry()) != null)
    {
        string directoryName = outputFolder;
        string fileName = Path.GetFileName(theEntry.Name);
        // create directory
        if (directoryName != "")
        {
            Directory.CreateDirectory(directoryName);
        }
        if (fileName != String.Empty)
        {
            if (theEntry.Name.IndexOf(".ini") < 0)
            {
                string fullPath = directoryName + "\\" + theEntry.Name;
                fullPath = fullPath.Replace("\\ ", "\\");
                string fullDirPath = Path.GetDirectoryName(fullPath);
                if (!Directory.Exists(fullDirPath))
Directory.CreateDirectory(fullDirPath);
                FileStream streamWriter = File.Create(fullPath);
                int size = 2048;
                byte[] data = new byte[2048];
                while (true)
                {
                    size = s.Read(data, 0, data.Length);
                    if (size > 0)
                    {
                        streamWriter.Write(data, 0, size);
                    }
                    else
                    {
                        break;
                    }
                }
                streamWriter.Close();
            }
        }
```

```
    }
    s.Close();
    if (deleteZipFile)
        File.Delete(zipPathAndFile);
}
```

7.6 Latest uploaded ftp

Με το πάτημα του κουμπιού “Latest Uploaded” δημιουργείται ένα νέο object τύπου FTP, το ftplib (το οποίο εμπεριέχεται στο source code του αρχείου ftblib.cs).

- Γίνεται απόπειρα σύνδεσης με τον ftp server μέσω της μεθόδου connect(), η οποία παίρνει arguments τα: server name, user name και password.
- Γίνεται μετακίνηση στο directory το οποίο βρίσκεται αποθηκευμένο στην string μεταβλητή ftpSavePath μέσω της μεθόδου ChangeDir(ftpSavePath).
- Με το βρόχο επανάληψης foreach, εκτυπώνονται τα περιεχόμενα του τρέχοντος directory μέσω του string f.

```
private void button8_Click(object sender, EventArgs e)
{
    FTP ftplib = new FTP();

    try
    {
        // there are server, user and password properties
        // that can be set within the ftplib object as well
        // those properties are actually set when
        // you call the Connect(server, user, pass) function

        ftplib.Connect(ftpServerName, ftpLogin, ftpPassword);
        ftplib.ChangeDir(ftpSavePath);
    }
    catch (Exception ex)
    {
        updateLogText("--Sfalma ftp server apo ftplib" + ex.Message + "\r\n");
    }

    try
    {
        foreach (string f in ftplib.ListFiles())
            textBox6.Text = f;
    }
    catch (Exception ex)
    {
        updateLogText("Sfalma sto ftp file listing. " + ex.Message + "\r\n");
    }
}
```

7.7 Download Ftp backup

Με το πάτημα του κουμπιού “Download Ftp backup” δημιουργείται ένα νέο object τύπου FTP, το ftplib (το οποίο εμπεριέχεται στο source code του αρχείου ftblib.cs).

- Γίνεται απόπειρα σύνδεσης με τον ftp server μέσω της μεθόδου connect(), η οποία παίρνει arguments τα: server name, user name και password.
- Γίνεται μετακίνηση στο directory το οποίο βρίσκεται αποθηκευμένο στην string μεταβλητή ftpSavePath μέσω της μεθόδου ChangeDir(ftpSavePath).

- Γίνεται έλεγχος για το αν υπάρχει ήδη zip backup file στον φάκελο αποθήκευσης του server. Αν υπάρχει τότε μέσω της File.Delete() το αρχείο σβήνεται.
- Καλείται η μέθοδος ftplib.OpenDownload(), η οποία παίρνει 3 ορίσματα: όνομα αρχείου που θα γίνει download, directory path που θα αποθηκευτεί τοπικά και για τον αν θα είναι resume enabled.
- Για όσο η DoDownload έχει τιμή μεγαλύτερη του 0, αυτό σημαίνει ότι το αρχείο είναι ακόμα υπό download (χρησιμοποιείται για την εκτύπωση του download status του αρχείου).
- Μετά την ολοκλήρωση της λήψης του αρχείου, κλείνει το connection με τον ftp server μέσω της ftplib.Disconnect().

```
private void button4_Click(object sender, EventArgs e)
{
    FTP ftplib = new FTP();

    try
    {
        // there are server, user and password properties
        // that can be set within the ftplib object as well
        // those properties are actually set when
        // you call the Connect(server, user, pass) function
        ftplib.Connect(ftpServerName, ftpLogin, ftpPassword);
        ftplib.ChangeDir(ftpSavePath);
    }
    catch (Exception ex)
    {
        updateLogText("====================================\r\n");
        updateLogText("--Ftp server error from ftplib. " + ex.Message + " \r\n");
        updateLogText("====================================\r\n");
    }

    try
    {
        int perc = 0;
        // open the file with resume support if it already exists, the last

        // peram should be false for no resume
        if (File.Exists("c:\\FT\\rec" + "\\" + "rec" + ".zip"))
            File.Delete("c:\\FT\\rec" + "\\" + "rec" + ".zip");

        ftplib.OpenDownload("rec.zip", "c:/FT/rec/rec.zip", true);
        while (ftplib.DoDownload() > 0)
        {
            perc = (int)((ftplib.BytesTotal * 100) / ftplib.FileSize);
            updateLogText("--Downloading. Total bytes: " + ftplib.BytesTotal + "
filesize: " + ftplib.FileSize + " percent:  " + perc + "% \r\n");
        }


        ftplib.Disconnect();
    }
    catch (Exception ex)
    {
        updateLogText("====================================\r\n");
        updateLogText("--Download error from ftplib. " + ex.Message + " \r\n");
        updateLogText("====================================\r\n");
    }
}
```

ΚΕΦΑΛΑΙΟ 8ο

CLIENT SOURCE CODE

8.1 Show Files

Το κουμπί “Show Files” αναλαμβάνει να εμφανίσει σε tree node τα περιεχόμενα του κύριου φακέλου αποθήκευσης του server. Αυτό γίνεται ως εξής:

- Καλείται η dir_server() method, η οποία πρώτα καθαρίζει το treeView1 από όλα τα nodes.
- Γίνεται αποστολή του message τύπου “dir” στον server. Ο server εκτελεί την εντολή, δηλαδή αποθηκεύει προσωρινά τα directory contents του κύριου φάκελου αποθήκευσης του server και αποστέλλει τα αποτελέσματα πίσω στον client, μέσω message.
- Ο client, μέσω της μεθόδου get_dir(), δημιουργεί ένα καινούργιο socket με χρήση endpoint με τον server για την λήψη του message που περιέχει τα directory contents.
- Μετά την λήψη και αποθήκευση του μηνύματος, ο client καλεί την read_dir(). Αν υπάρχει το αρχείο dir.msg, τότε μέσω ενός buffer αποθηκεύεται προσωρινά σε ένα string. Γίνεται διαχωρισμός του string, ώστε να αποθηκευτούν όλα τα αρχεία σε ένα ξεχωριστό string array.
- Αν το action ισούται με dir, σημαίνει ότι όντως το messages περιέχει directory contents. Έτσι ξεκινάει η δημιουργία του treeView1, το οποίο προσθέτει ένα node για κάθε αρχείο που βρίσκει στον πίνακα filenames.

```
private void button11_Click(object sender, EventArgs e)
{
    dir_server();
}

public void dir_server()
{
    treeView1.Nodes.Clear();
    send("dir");
    get_dir();
    read_Dir();
}

public void get_dir()
{
    try
    {

        if (File.Exists(@"c:\FT\rec\cache.bak"))
            File.Delete(@"c:\FT\rec\cache.bak");
        if (File.Exists(@"c:\FT\rec\dir.msg"))
            File.Delete(@"c:\FT\rec\dir.msg");
```

```
        IPAddress[] ipAddress = Dns.GetHostAddresses(textBox2.Text);// original2
        IPEndPoint ipEnd = new IPEndPoint(ipAddress[0], Int32.Parse(receivePort));

        if (onlineMode == true)
        {
            IPHostEntry hostInfo = Dns.GetHostEntry("online-
storage.dyndns.org");//lolz
                ipEnd = new IPEndPoint(hostInfo.AddressList[0],
Int32.Parse(receivePort));

            //IPEndPoint ipEnd = new IPEndPoint(ipAddress[0], 5655);//original
        }


        Socket clientSock = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.IP);
        clientSock.Connect(ipEnd);


        byte[] clientData = new byte[1024 * 5000];
        string receivedPath = "C:/FT/rec/";

        int receivedBytesLen = clientSock.Receive(clientData);

        int fileNameLen = BitConverter.ToInt32(clientData, 0);
        string fileName = Encoding.ASCII.GetString(clientData, 4, fileNameLen);

        //MessageBox.Show("Client:" + clientSock.RemoteEndPoint + " connected & File "
+ fileName + " started received.");

        BinaryWriter bWrite = new BinaryWriter(File.Open(receivedPath + fileName,
FileMode.Append)); ;
        bWrite.Write(clientData, 4 + fileNameLen, receivedBytesLen - 4 - fileNameLen);

        //MessageBox.Show("File: received & saved at path: " + fileName +
receivedPath);

        bWrite.Close();
        clientSock.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show("File Sending fail." + ex.Message);
    }
}

public void read_Dir()
{
    try
    {
        // Set the position to the beginning of the stream.
        FileStream fs = null;
        Int32 cacheLen = 0;
        byte[] byteArray = null;

        if (File.Exists(@"c:\FT\rec\dir.msg"))
        {
            fs = new FileStream(@"c:\FT\rec\dir.msg", FileMode.Open);
            cacheLen = Int32.Parse(fs.Length.ToString());
            byteArray = new byte[cacheLen];
            fs.Read(byteArray, 0, cacheLen);
            fs.Close();
```

```
        }
        else
            throw new System.InvalidOperationException("Error reading message");

        //int fileNameLen = BitConverter.ToInt32(byteArray, 0); // file name length
with extension //original
        //string fileName = Encoding.ASCII.GetString(byteArray, 4, fileNameLen); //
the file name with extension //original
        //updateLogText("--filename= " + fileName); //original
        //string message = Encoding.ASCII.GetString(byteArray, 4 + fileNameLen, length
- 4 - fileNameLen);//original

        string message = Encoding.ASCII.GetString(byteArray);//original
        string[] credentials = message.Split('|');
        string action = credentials[0];

        // dir files msg read start
        string files = credentials[1];
        string[] filenames = files.Split(':');
        // dir files msg read end

        txtClientLog.Text += "--Message from client: " + message.ToString();

        if (action == "dir")
        {
            //MessageBox.Show("dir");
            // try to login
            // if it logs
            // global bool loged = true
            // it can file transfer

            //
            // kyrios fakelos start
            TreeNode main_folder = new TreeNode();
            main_folder.Name = "mainfolder";
            main_folder.Text = "Main Folder";
            main_folder.Tag = "Main Folder";
            treeView1.Nodes.Add(main_folder);
            // kyrios fakelos end

            TreeNode[] file = new TreeNode[filenames.Length];
            for (int i = 0; i < filenames.Length; i++)
            {
                treeView1.SelectedNode = main_folder;
                file[i] = new TreeNode();
                file[i].Name = filenames[i].ToString();
                file[i].Text = filenames[i].ToString();
                file[i].Tag = filenames[i].ToString();
                treeView1.SelectedNode.Nodes.Add(file[i]);
                treeView1.SelectedNode.ExpandAll();
                treeView1.Sort();
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error. " + ex.Message);
    }
}
```

8.2 Refresh sync status

Αναλαμβάνει να ελέγξει για το αν τα τοπικά αρχεία του client είναι synchronized με τα αρχεία του server:

- Γίνεται πρώτα έλεγχος και σβήσιμο των προσωρινών αρχείων στον τοπικό φάκελο αποθήκευσης (cache και messages αρχεία).
- Υπολογίζεται το πλήθος των αρχείων στον φάκελο του client καθώς και το συνολικό μέγεθός τους. Το αποτέλεσμα εμφανίζεται στα αντίστοιχα text boxes.
- Καλείται η μέθοδος askSize_server(), αναλαμβάνει να στείλει μήνυμα στον server για το πλήθος των αρχείων και το συνολικό μέγεθός τους σε MB. Το αποτέλεσμα από το μήνυμα που λαμβάνει ο client το εμφανίζει στα αντίστοιχα text boxes.

```
private void button9_Click_1(object sender, EventArgs e)
{
    if (File.Exists(@"c:\FT\rec\cache.bak"))
        File.Delete(@"c:\FT\rec\cache.bak");
    if (File.Exists(@"c:\FT\rec\dir.msg"))//wtf
        File.Delete(@"c:\FT\rec\dir.msg");

    textBox5.Text = (calcFolderFileSize(fileSavePath) / (1024 *
1024)).ToString("0.000") + " MB";
    int fileCount = Directory.GetFiles(fileSavePath, "*.*",
SearchOption.TopDirectoryOnly).Length; // Will Retrieve count of all files in directry
but not sub directries
    textBox6.Text = fileCount.ToString();
    askSize_server();
}
```

8.3 Get download list

Το κουμπί “Get Download list” αναλαμβάνει να συγκρίνει τα τοπικά αρχεία με τα αρχεία στον server. Αν διαπιστώσει ότι ο τοπικός φάκελος δεν έχει όλα τα αρχεία που έχει ο server, τότε δημιουργείται ένα drop down list box το οποίο περιέχει όλα τα αρχεία που δεν υπάρχουν. Επιλέγοντας το αντίστοιχο αρχείο και πατώντας το κουμπί “Get this file” ξεκινάει αυτόματα η διαδικασία λήψης του συγκεκριμένου αρχείου στον τοπικό φάκελο.

```
private void button5_Click(object sender, EventArgs e)
{
    // count local files, count remote files
    // apothikeush twn local file name kai twn remote se 2 pinakes
    // remote pinakas arxeiwn
    send("dir");
    get_dir();
    read_dirSync();

    if (Int32.Parse(textBox9.Text) > 0)
    {
        button10.Enabled = true;
        comboBox1.Enabled = true;
    }
    else
    {
        button10.Enabled = false;
        comboBox1.Enabled = false;
    }
```

}

8.4 Read dirsync

Η μέθοδος αυτή ανοίγει το αρχείο dir.msg.

- Αν υπάρχει το αρχείο dir.msg, τότε μέσω ενός buffer αποθηκεύεται προσωρινά σε ένα string. Γίνεται διαχωρισμός του string, ώστε να αποθηκευτούν όλα τα αρχεία σε ένα ξεχωριστό string array.
- Αν το action ισούται με dir, σημαίνει ότι όντως το message περιέχει directory contents. Κάθε αρχείο που υπάρχει (και χωρίζεται με ":") αποθηκεύεται στον πίνακα filenames.
- Ο πίνακας fileNamesLocal περιέχει τις ονομασίες όλων των τοπικών αρχείων.
- Γίνεται σύγκριση των remote files (δηλαδή τα περιεχόμενα του string array filenames) με τον πίνακα fileNamesLocal. Κάθε αρχείο που εντοπίζεται ότι δεν υπάρχει στον τοπικό φάκελο, αποθηκεύεται στο string array filesToGet και αυξάνεται ο counter κατά ένα.
- Αν ο counter είναι ίσος με το μηδέν, αυτό σημαίνει ότι δεν χρειάζεται να γίνει λήψη αρχείου από τον server στον client. Διαφορετικά το combolist box αρχικά γίνεται clear και έπειτα προστίθεται σε αυτό κάθε αρχείο που υπάρχει στον πίνακα filesToGet.

```
public void read_dirSync()
{
    try
    {
        // Set the position to the beginning of the stream.
        FileStream fs = null;
        Int32 cacheLen = 0;
        byte[] byteArray = null;

        if (File.Exists(@"c:\FT\rec\dir.msg"))
        {
            fs = new FileStream(@"c:\FT\rec\dir.msg", FileMode.Open);
            cacheLen = Int32.Parse(fs.Length.ToString());
            byteArray = new byte[cacheLen];
            fs.Read(byteArray, 0, cacheLen);
            fs.Close();
        }
        else
            throw new System.InvalidOperationException("Error reading message");

        //int fileNameLen = BitConverter.ToInt32(byteArray, 0); // file name length
with extension //original
        //string fileName = Encoding.ASCII.GetString(byteArray, 4, fileNameLen); //
the file name with extension //original
        //updateLogText("--filename= " + fileName); //original
        //string message = Encoding.ASCII.GetString(byteArray, 4 + fileNameLen, length
- 4 - fileNameLen);//original

        string message = Encoding.ASCII.GetString(byteArray);//original
        string[] credentials = message.Split('|');
        string action = credentials[0];
        //string usr = credentials[1];//original
        //string psw = credentials[2];//original
```

```
        // dir files msg read start
        string files = credentials[1];
        string[] filenames = files.Split(':');
        // dir files msg read end

        txtClientLog.Text += "--Message from client: " + message.ToString();
        //txtClientLog.Text += "--Username: '" + usr + "' password: '" + psw + "' .
Client wants to... " + action;
        //txtClientLog.Text += "--Files: '" + filenames[0] + filenames[1] +
filenames[2];

        //foreach (string line in filenames)
        //MessageBox.Show("File: '"+ line +"'" );

        if (action == "dir")
        {
            // server pinakas arxeiwn
            //string[] fileNamesServer = new
string[Int32.Parse(textBox1.Text.ToString())];// xreizetai? :o
            txtClientLog.Text += "\r\n--Server Files\r\n-- Num of files = '" +
filenames.Length + "'\r\n";
            for (int i = 0; i < filenames.Length; i++)
            {
                //filenames[i] = filenames[i].Substring(10);
                txtClientLog.Text += "-- File " + filenames[i] + "\r\n";
            }

            //topikos pinakas arxeiwn
            string[] fileNamesLocal = Directory.GetFiles(@"c:\FT\rec", "*.*",
SearchOption.TopDirectoryOnly);
            //string[] directories = Directory.GetDirectories(@"c:\FT\rec", "*.*",
SearchOption.AllDirectories); //original

            txtClientLog.Text += "--Client Files\r\n-- Num of files = '" +
fileNamesLocal.Length + "'\r\n";
            for (int i = 0; i < fileNamesLocal.Length; i++)
            {
                fileNamesLocal[i] = fileNamesLocal[i].Substring(10);
                txtClientLog.Text += "-- File " + fileNamesLocal[i] + "\r\n";
            }

            // sygkrish pinaka local me ton global, apothikeush twn diff se 3o pinaka
            int counter = 0;
            bool found;
            string[] filesToGet = new string[filenames.Length];

            for (int j = 0; j < filenames.Length; j++)
            {
                found = false;

                for (int i = 0; i < fileNamesLocal.Length; i++)
                {
                    if (fileNamesLocal[i] == filenames[j])
                    {
                        found = true;
                    }
                }

                if (found == false && filenames[j] != "dir.msg")
                {
                    filesToGet[counter] = filenames[j];
                    counter++;
```

```
                }

            }

            if (counter > 0)
            {
                txtClientLog.Text += "--Client needs to get " + counter + "
files\r\n";
                textBox9.Text = counter.ToString();
                textBox8.Text = "Download";

            }
            else
            {
                txtClientLog.Text += "--Client is synced\r\n";
                textBox9.Text = "0";
                textBox8.Text = "Nothing";
            }

            for (int i = 0; i < counter; i++)
                txtClientLog.Text += "-- File " + filesToGet[i] + "\r\n";

            // request ta files pou einai ston pinaka diff ena ena

            comboBox1.Items.Clear();
            for (int i = 0; i < counter; i++)
            {
                //txtClientLog.Text += "-- Requesting file " + filesToGet[i] + "from
server\r\n";
                // na doume an tha kollaei to sympan edw :P
                comboBox1.Items.Add(filesToGet[i]);
                comboBox1.SelectedIndex = 0;
            }

        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error. " + ex.Message);
    }
}
```

ΚΕΦΑΛΑΙΟ 9ο

ASP.NET SITE SOURCE CODE

9.1 Register.aspx

Ο πηγαίος κώδικας της σελίδας Register.aspx αναλαμβάνει να δημιουργήσει connection με την SQL Server 2008 βάση δεδομένων και να εισάγει τα δεδομένα της φόρμας που έγιναν submit από τον χρήστη στον κατάλληλο πίνακα. Αυτό γίνεται ως εξής:

- Δημιουργείται SqlConnection με ονομασία conn. Το connection string της σύνδεσης με τον Sql server ορίζεται από το web config file που υπάρχει αποθηκευμένο στο ίδιο directory.
- Η μέθοδος ExecuteInsert παίρνει τα ορίσματα από την form (συγκεκριμένα τα name, username, password, gender, age και address) που είχαν εισαχθεί και τα τοποθετεί στο insert query με ονομασία sql.
- Ανοίγει το connection με τον server μέσω της try-catch-finally τεχνικής.
- Κάθε μια παράμετρος προστίθεται στον sql command και έπειτα ορίζεται το command type ίσο με Text.
- Εκτελείται το query και μέσω της finally κλείνει η σύνδεση με τον sql server.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.SqlClient;
using System.Web.Configuration;

public partial class _Register : System.Web.UI.Page
{
    public string GetConnectionString()
    {
        //sets the connection string from your web config file "ConnString" is the
name of your Connection String
        return
System.Configuration.ConfigurationManager.ConnectionStrings["MyConsString"].Connection
String;
    }
    private void ExecuteInsert(string name, string username, string password, string
gender, string age, string address)
    {
        SqlConnection conn = new SqlConnection(GetConnectionString());
        string sql = "INSERT INTO tblRegistration (Name, UserName, Password, Gender,
Age, Address) VALUES " + " (@Name,@UserName,@Password,@Gender,@Age,@Address)";
        try
        {
            conn.Open();
            SqlCommand cmd = new SqlCommand(sql, conn);
            SqlParameter[] param = new SqlParameter[6];

            //param[0] = new SqlParameter("@id", SqlDbType.Int, 20);
```

```
            param[0] = new SqlParameter("@Name", SqlDbType.VarChar, 50);
            param[1] = new SqlParameter("@UserName", SqlDbType.VarChar, 50);
            param[2] = new SqlParameter("@Password", SqlDbType.VarChar, 50);
            param[3] = new SqlParameter("@Gender", SqlDbType.Char, 10);
            param[4] = new SqlParameter("@Age", SqlDbType.Int, 100);
            param[5] = new SqlParameter("@Address", SqlDbType.VarChar, 50);
            param[0].Value = name;
            param[1].Value = username;
            param[2].Value = password;
            param[3].Value = gender;
            param[4].Value = age;
            param[5].Value = address;

            for (int i = 0; i < param.Length; i++)
            {
                cmd.Parameters.Add(param[i]);
            }

            cmd.CommandType = CommandType.Text;
            cmd.ExecuteNonQuery();
        }

        catch (System.Data.SqlClient.SqlException ex)
        {
            string msg = "Insert Error:";
            msg += ex.Message;
            throw new Exception(msg);
        }
        finally
        {
            conn.Close();
        }

    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        if (TxtPassword.Text == TxtRePassword.Text)
        {
            //call the method to execute insert to the database
            ExecuteInsert(TxtName.Text, TxtUserName.Text, TxtPassword.Text,
DropDownList1.SelectedItem.Text, TxtAge.Text, TxtAddress.Text);
            Response.Write("Record was successfully added!");
            ClearControls(Page);
        }
        else
        {
            Response.Write("Password did not match");
            TxtPassword.Focus();
        }

    }
    public static void ClearControls(Control Parent)
    {
        if (Parent is TextBox)
        { (Parent as TextBox).Text = string.Empty; }
        else
        {
            foreach (Control c in Parent.Controls)
                ClearControls(c);
        }
    }
  }
```

ΚΕΦΑΛΑΙΟ 10ο

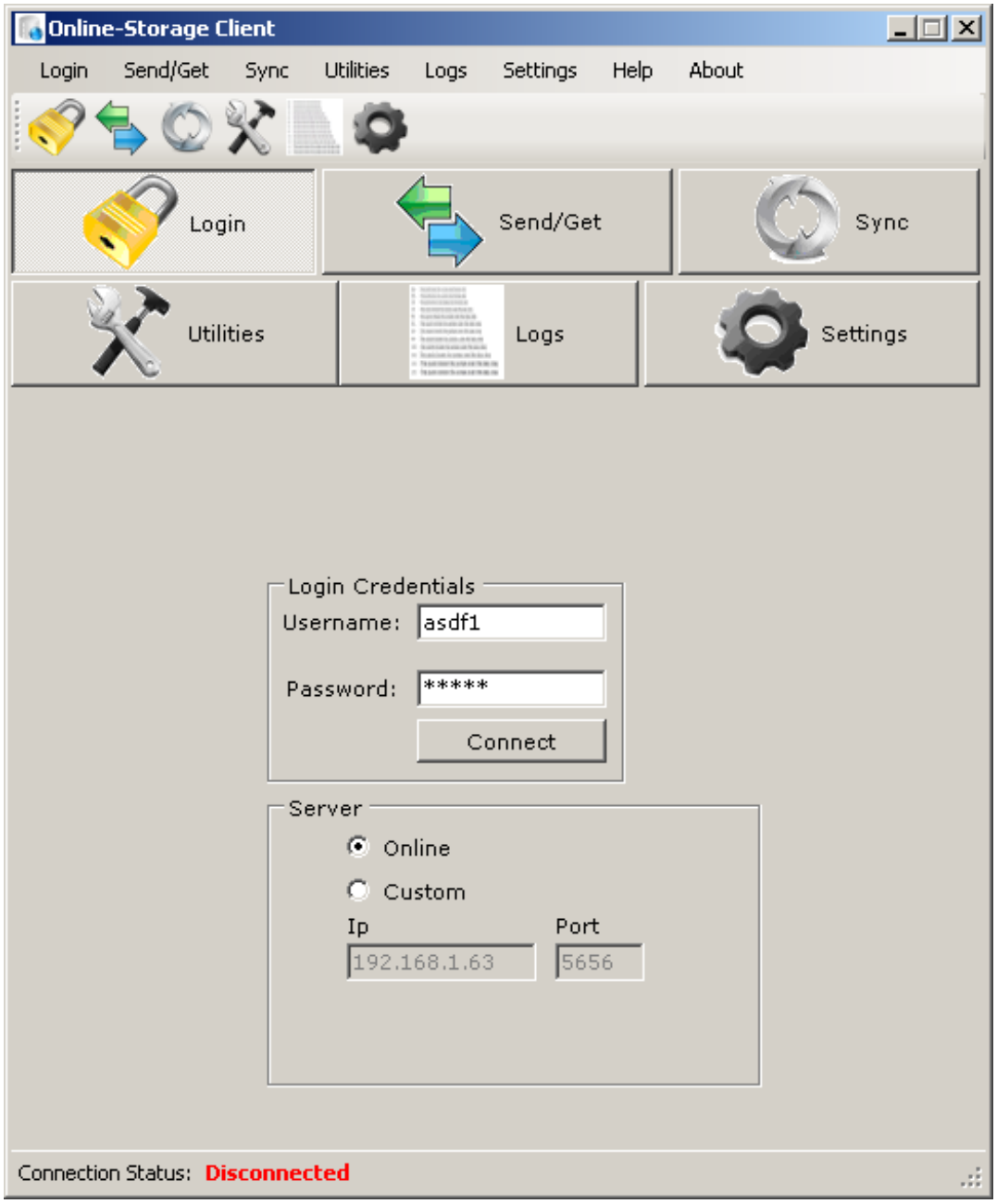
ΣΕΝΑΡΙΑ ΛΕΙΤΟΥΡΓΙΑΣ ΚΑΙ ΧΡΗΣΗΣ ΤΗΣ ΠΛΑΤΦΟΡΜΑΣ

10.1 Σενάρια χρήσης Client Εφαρμογής

10.1.1 Λειτουργία εμφάνισης των online αρχείων

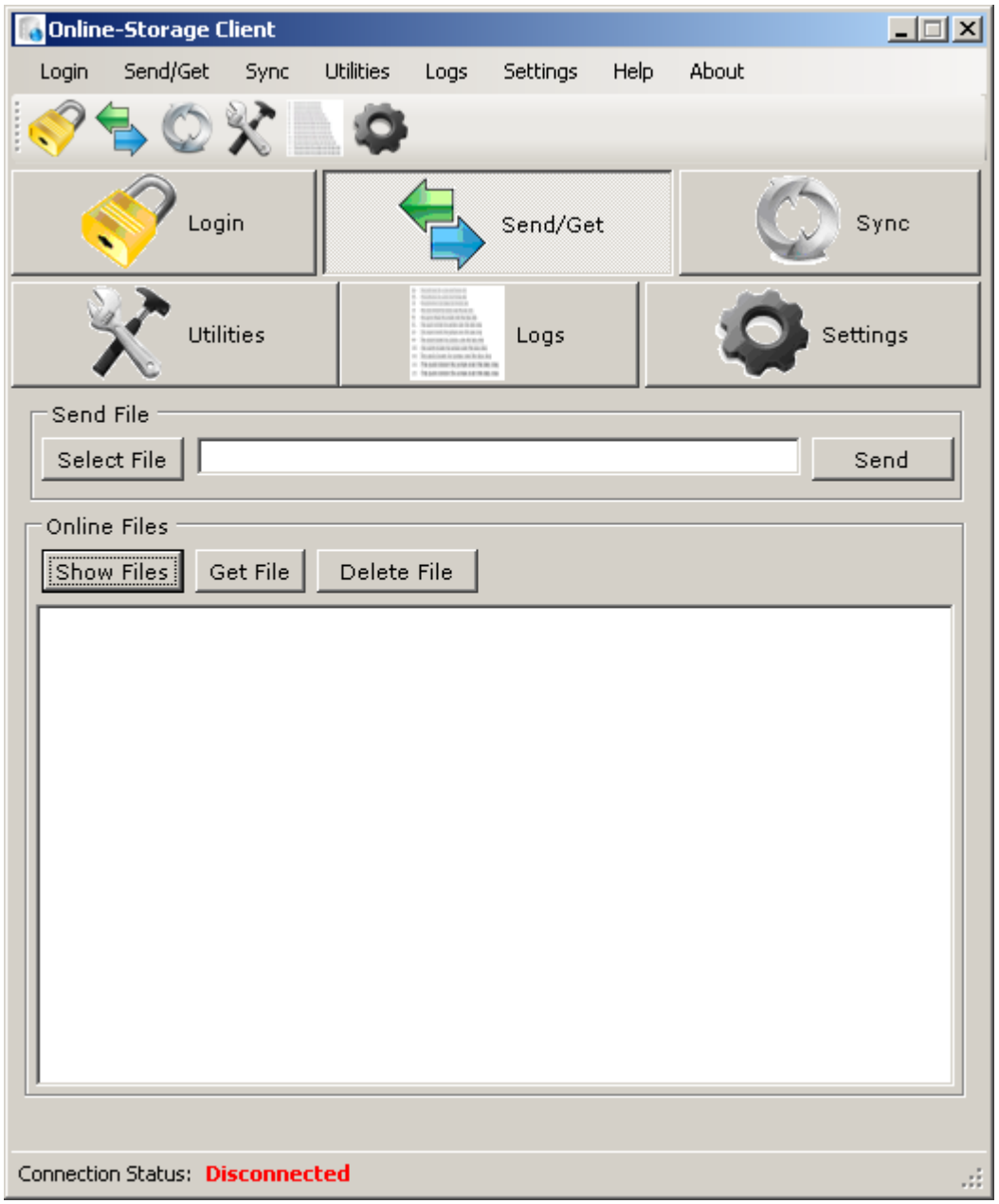
Η λειτουργία "Show Files" αναλαμβάνει να εμφανίσει τα αποθηκευμένα αρχεία στον server. Αυτό γίνεται ως εξής:

- Αφού έχει γίνει το login, ο χρήστης επιλέγει την καρτέλα "Send/Get".



Εικόνα 42. Καρτέλα Login

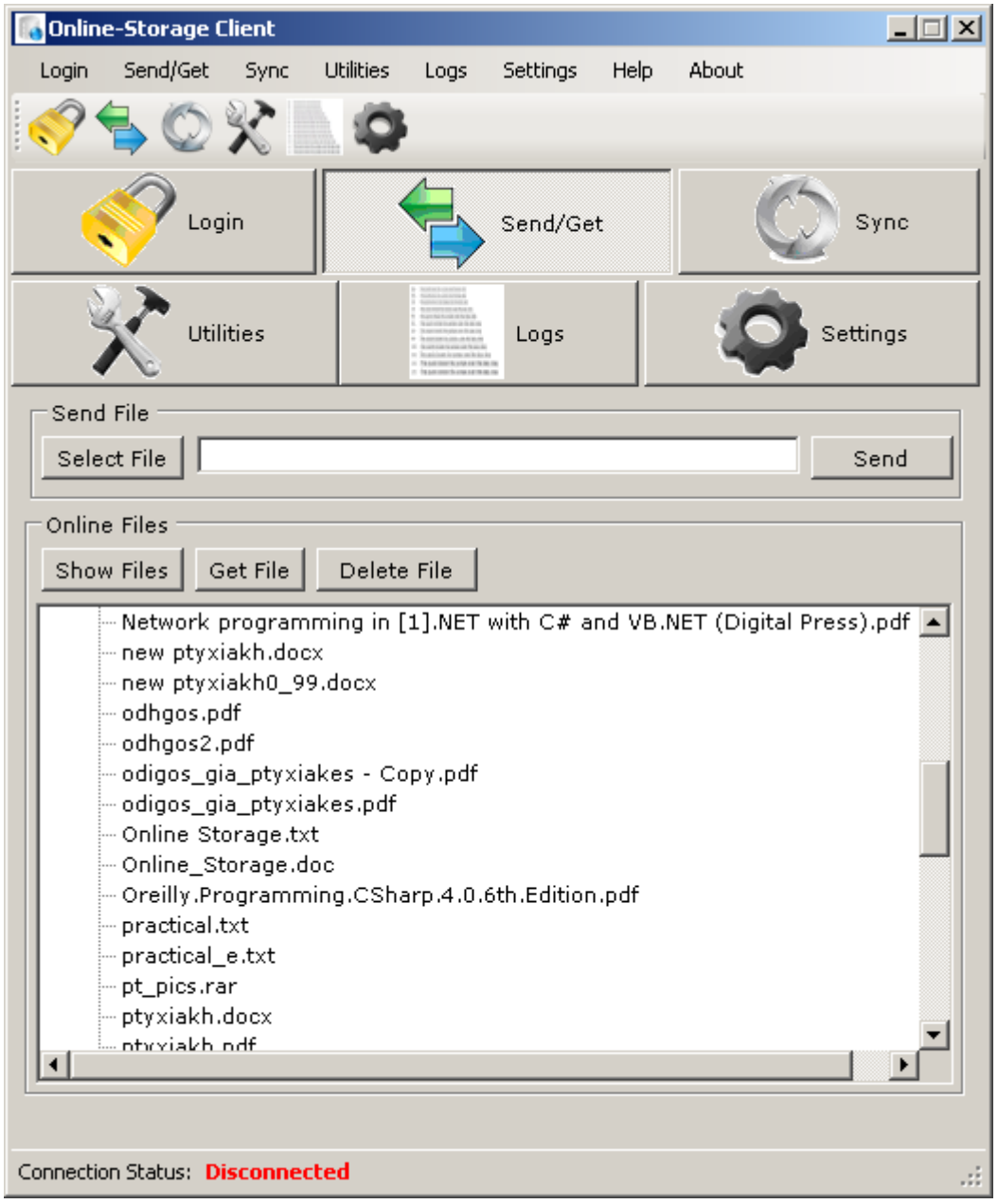
- Στην καρτέλα αυτή, θα χρειαστεί να πατήσει το κουμπί "Show Files".



Εικόνα 43. Καρτέλα Send/Get

- Το client πρόγραμμα ανανεώνει το τρέχον file view. Αποστέλλει αίτηση στην server εφαρμογή για προβολή όλων των τύπων αρχείων απο τον server.

- Αφού έρθει η απάντηση απο τον server, η client εφαρμογή "αποκωδικοποιεί" το response και εμφανίζει το ολικό δέντρο αρχείων στο πεδίο εμφάνισης του tree view, ακριβώς κάτω απο το κουμπί "Show Files".

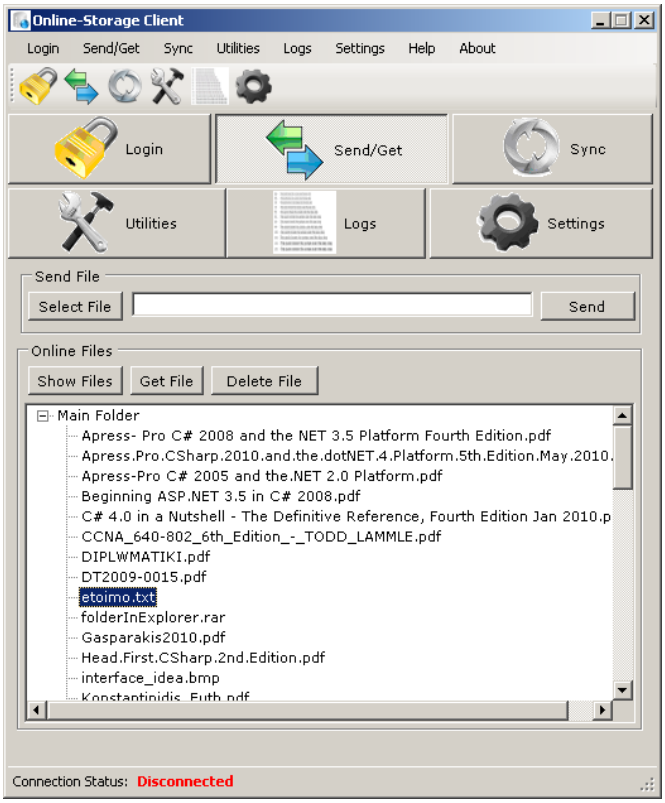


Εικόνα 44. Show Files.

10.1.2 Λειτουργία διαγραφής επιλεγμένου online αρχείου

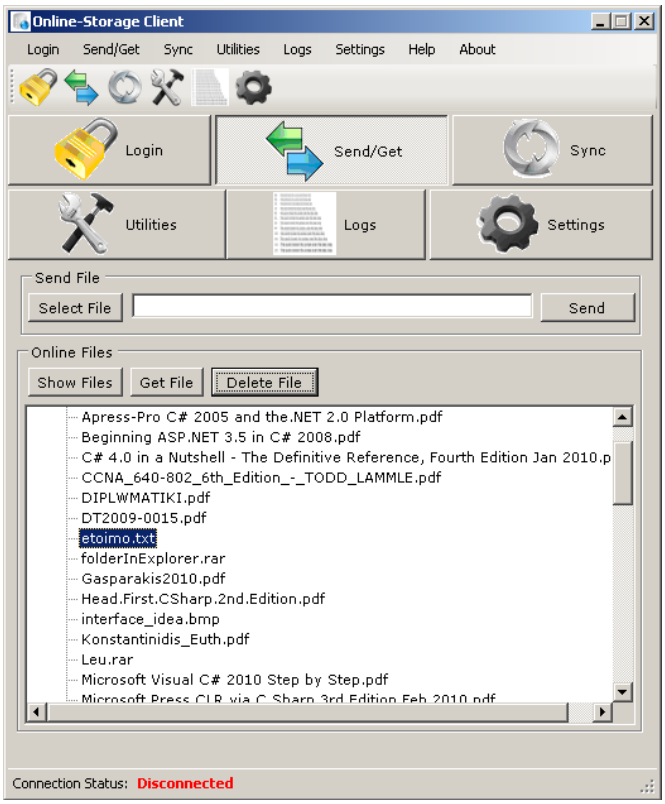
Το κουμπί "Delete File" αναλαμβάνει να σβήσει κάποιο συγκεκριμένο αρχείο απο τον server:

- Αφού έχει γίνει προβολή/ανανέωση των αρχείων στον Server, ο χρήστης θα χρειαστεί να επιλέξει κάποιο αρχείο απο το tree view που εμφανίζεται ακριβώς απο κάτω.



Εικόνα 45. Επιλογή αρχείου

- Έπειτα πατώντας το πλήκτρο "Delete File", αποστέλλεται μήνυμα-αίτηση προς τον server για την διαγραφή του επιλεγμένου αρχείου.
- Η server εφαρμογή διαβάζει την αίτηση του client και διαγράφει το συγκεκριμένο αρχείο.



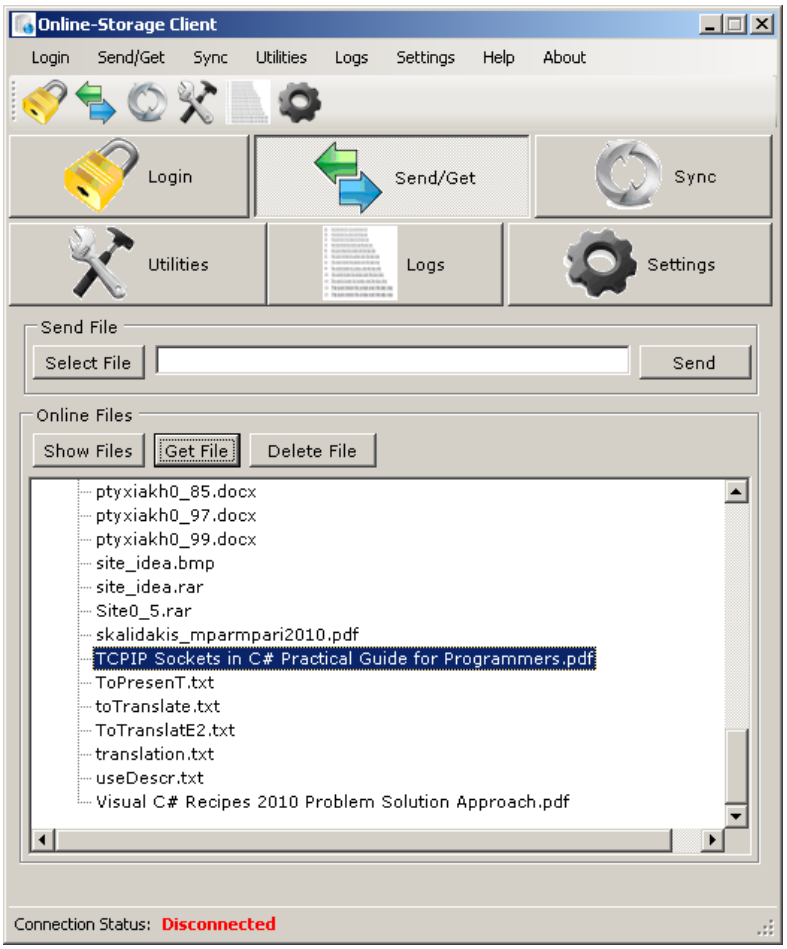
Εικόνα 46. Delete File

- Πατώντας ξανά το πλήκτρο "Show Files", για να γίνει ανανέωση των διαθέσιμων αρχείων στον server, διαπιστώνουμε οτι όντως το αρχείο που επιλέξαμε δεν υπάρχει πλέον στο δέντρο των αρχείων.

10.1.3 Σενάριο λήψης online αρχείου

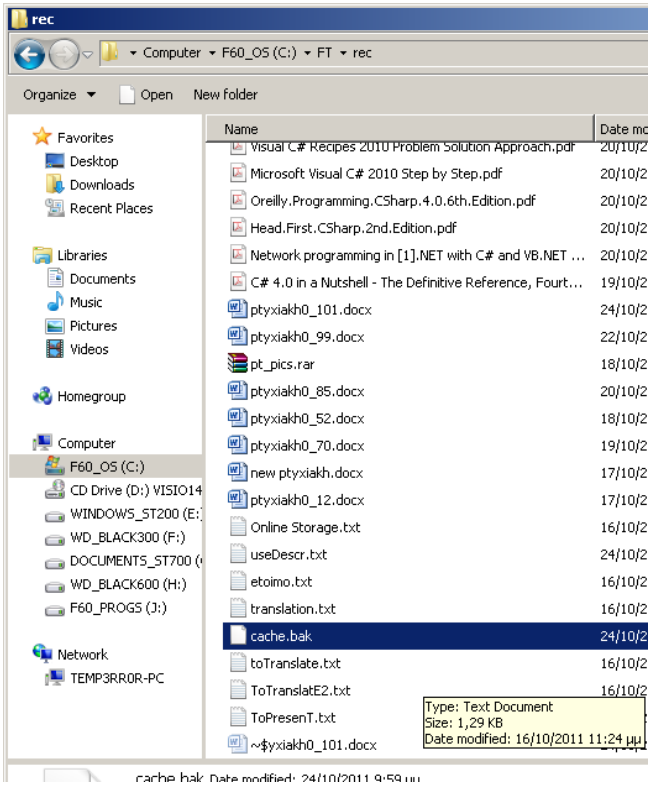
Το κουμπί "Get File" έχει παρόμοιο τρόπο λειτουργίας με το "Delete FIle" απο την πλευρά του χρήστη. Δηλαδή:

- Πατάμε του πλήκτρο "Show Files", γίνεται προβολή των διαθέσιμων αρχείων που υπάρχουν στον server.
- Επιλέγουμε το απομακρυσμένο αρχείο που θέλουμε να αποκτήσουμε και πατάμε το κουμπί "Get file".



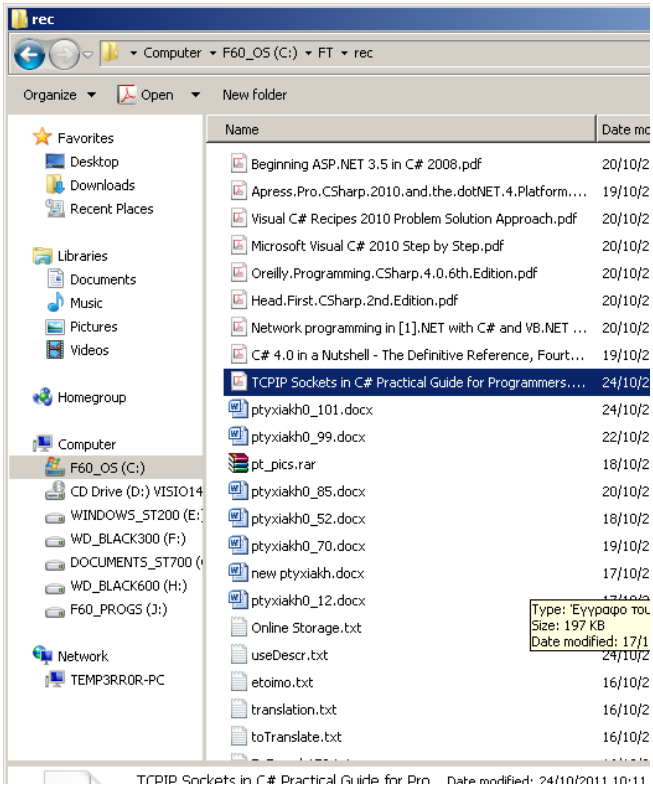
Εικόνα 47. Get File

- Η client εφαρμογή απέστειλε αίτηση για λήψη αρχείου. Ο server πραγματοποιεί σύνδεση με τον client στην port 5655 και εκκινείται η διαδικασία αποστολής του αρχείου μέσω του socket.
- Ο client λαμβάνει τα δεδομένα απο τον server και τα αποθηκεύει προσωρινά μέσω του buffer cache.bak.



Εικόνα 48. Buffer cache.bak

- Όταν ολοκληρωθεί η λήψη όλων των bytes, το αρχείο cache.bak μετονομάζεται. Πλέον το αρχείο έχει ληφθεί κανονικά και βρίσκεται στον φάκελο c:\ft\rec\.

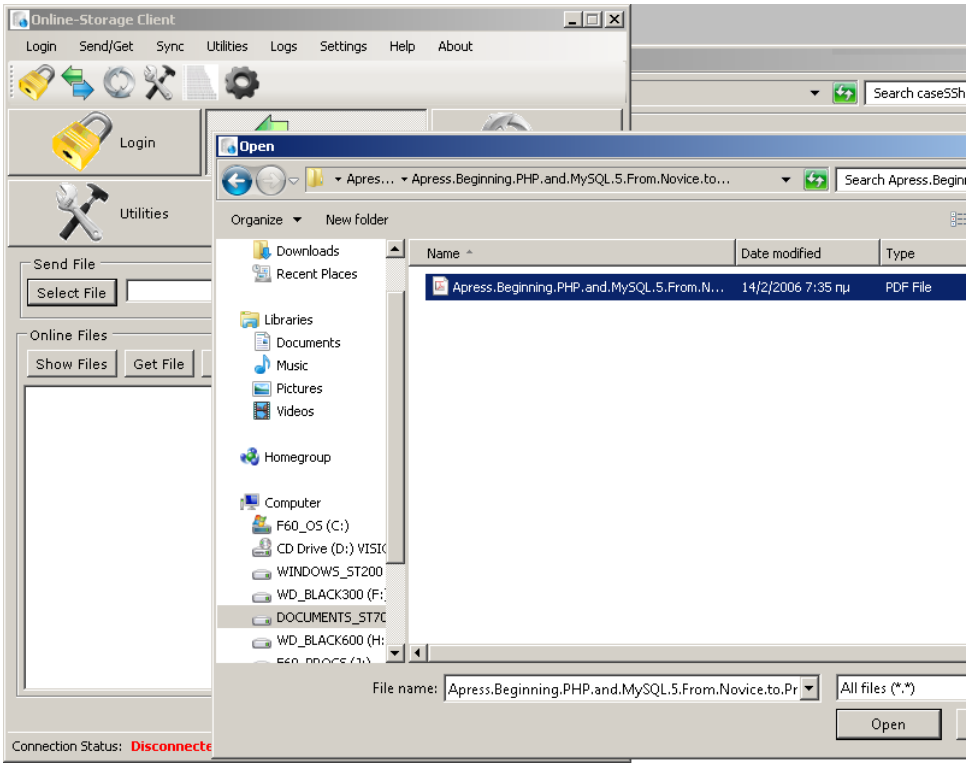


Εικόνα 49. Ολοκλήρωση λήψης αρχείου

10.1.4 Σενάριο αποστολής τοπικού αρχείου στον online αποθηκευτικό χώρο

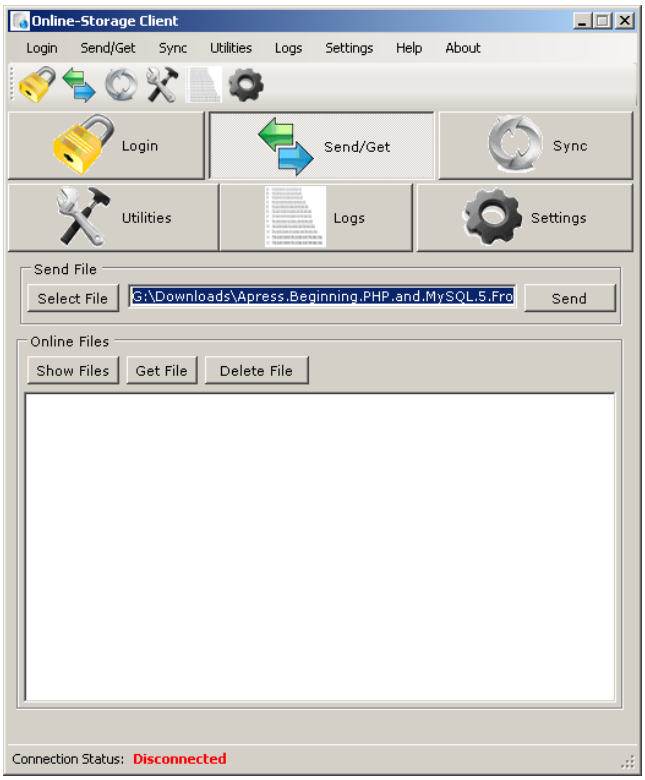
Για την αποστολή αρχείου απο τον client προς τον server θα χρειαστεί η συγκεκριμένη διαδικασία:

- Επιλέγουμε την καρτέλα "Send/Get".
- Πατάμε το πλήκτρο "Select File". Ανοίγει ένα File Open Dialog με αρχικό φάκελο τον "c:\".



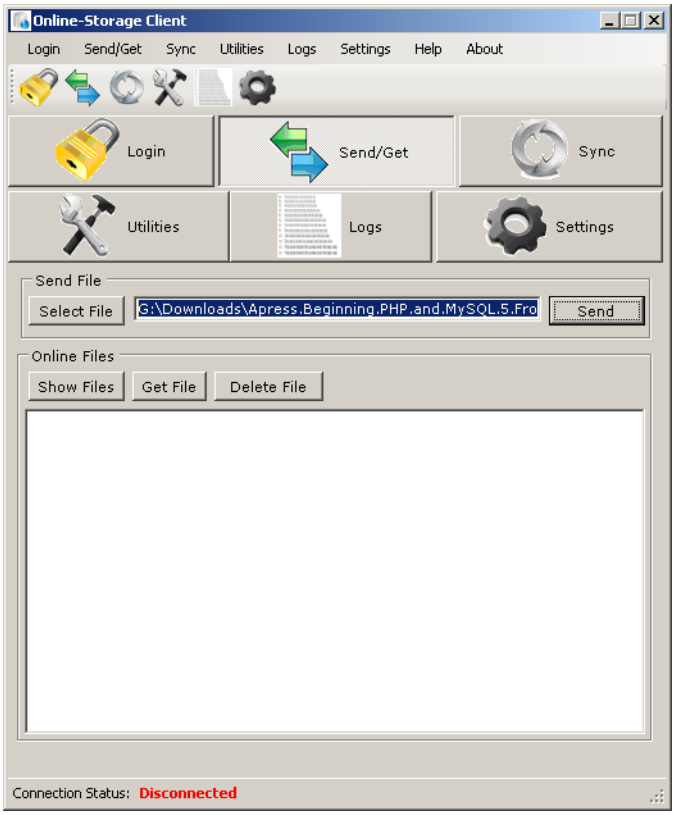
Εικόνα 50. Select File Dialog

- Ο χρήστης θα χρειαστεί να πλοηγηθεί στον φάκελο που επιθυμεί, να μαρκάρει το αρχείο και έπειτα να πατήσει το πλήκτρο "Open".
- Δεξιά απο το πλήκτρο "Select File", στο text box, εμφανίζεται το επιλεγμένο αρχείο.



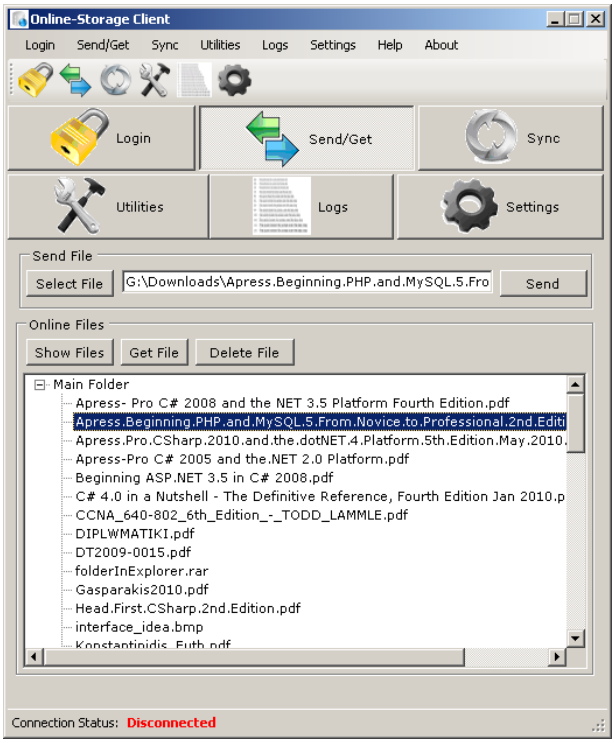
Εικόνα 51. Επιλεγμένο αρχείο για Send

- Πατώντας το πλήκτρο "Send" εκκινείται η διαδικασία αποστολής του αρχείου στον server. (δηλαδή πραγματοποιείται σύνδεση με την server εφαρμογή και γίνεται αποστολή των bytes του αρχείου).



Εικόνα 52. Αποστολή επιλεγμένου αρχείου

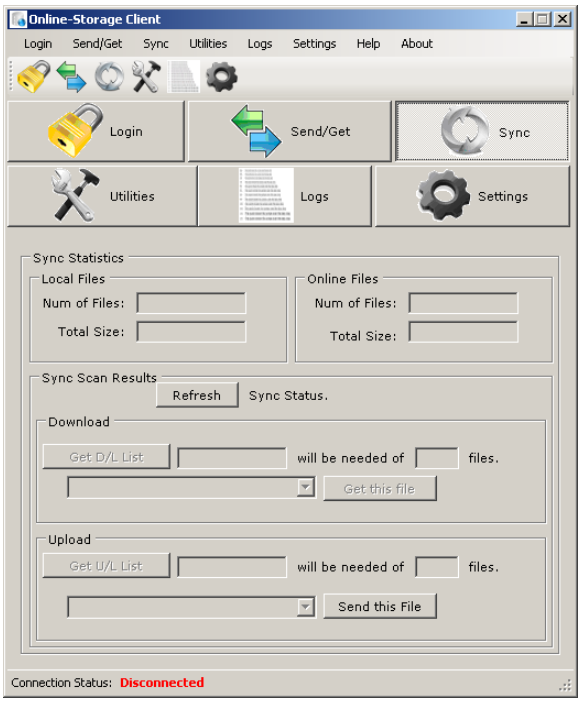
- Όταν ολοκληρωθεί η αποστολή του αρχείου, πατώντας το πλήκτρο "Show Files", θα εμφανιστεί στην λίστα με τα υπάρχοντα αρχεία στον server, επιβεβαιώνοντας έτσι το επιτυχές upload του αρχείου.



Εικόνα 53. Επιβεβαίωση του upload

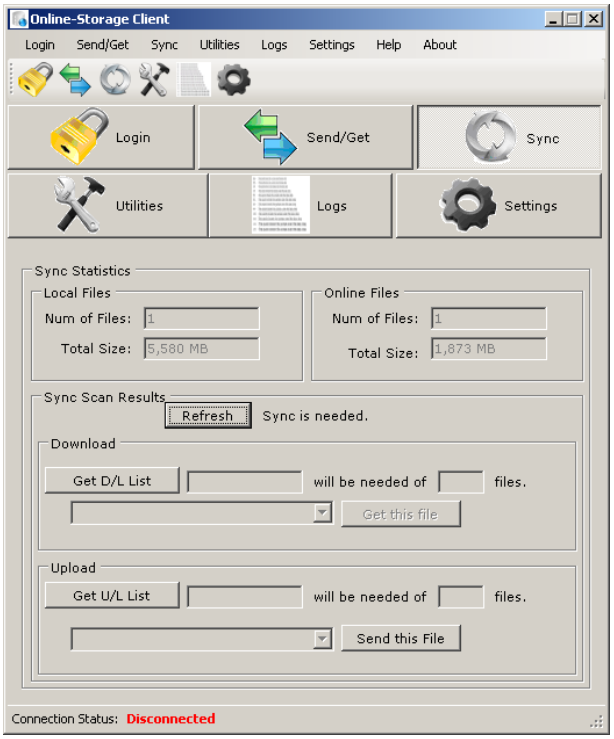
10.1.5 Σενάριο συγχρονισμού τοπικών αρχείων με τα online αρχεία

Η διαδικασία συγχρονισμού των αρχείων μεταξύ client και server γίνεται μέσω της καρτέλας "Sync" στην client εφαρμογή:



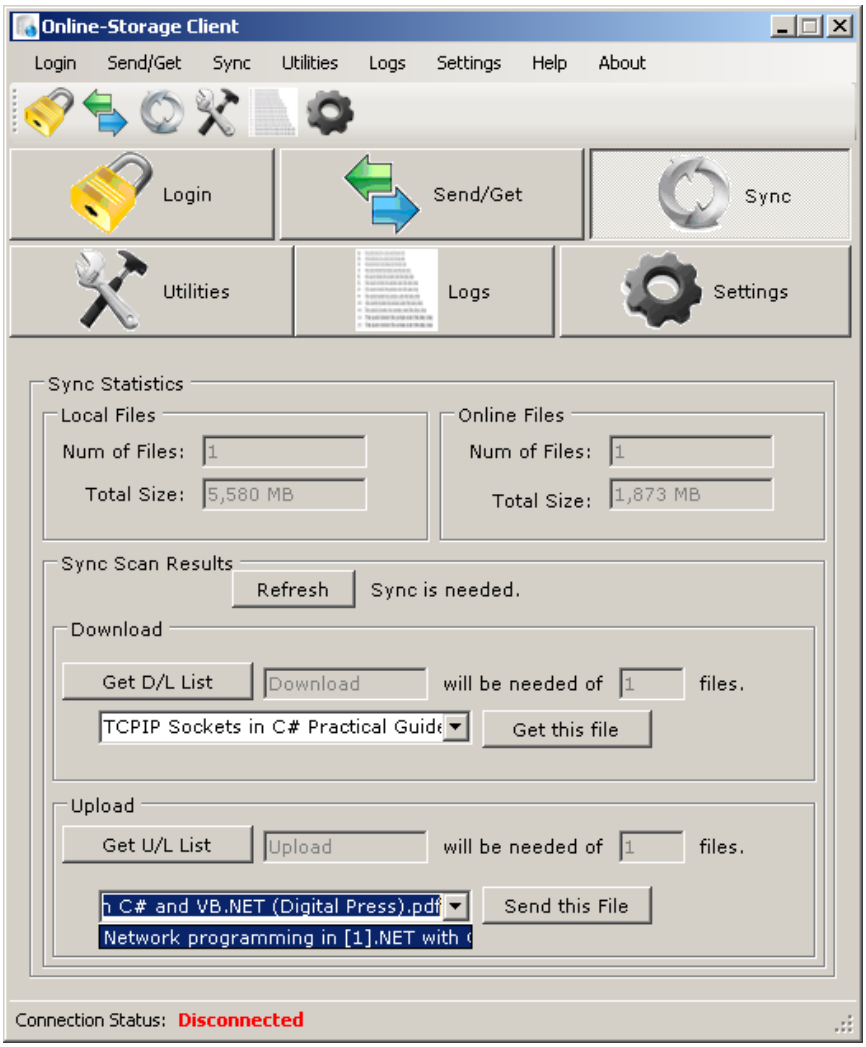
Εικόνα 54. Καρτέλα Sync

- Αρχικώς θα χρειαστεί να πατηθεί το κουμπί "Refresh". Ξεκινάει η διαδικασία καταμέτρησης αρχείων καθώς και του χώρου που καταλαμβάνουν τα αρχεία. Αυτή η διαδικασία γίνεται και στον τοπικό φάκελο του client και στον φάκελο του server.



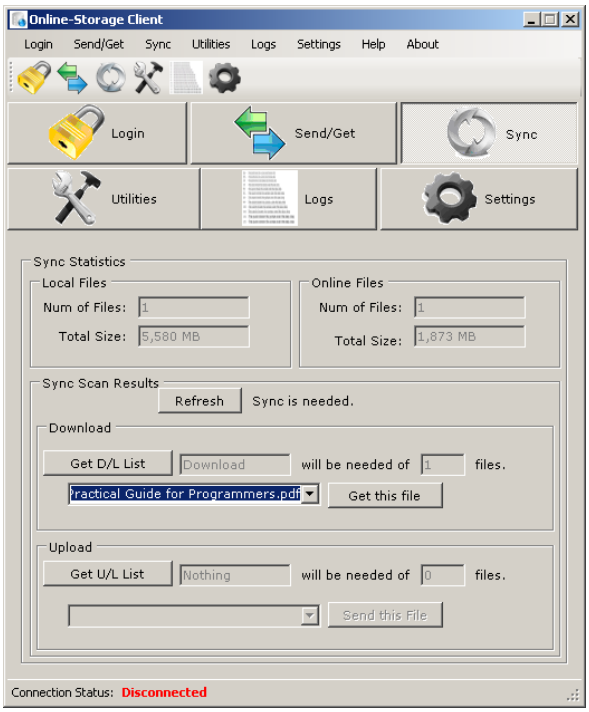
Εικόνα 55. Sync Refresh

- Οι τιμές στα πεδία Num of files και Total Size ανανεώνονται. Αν διαπιστωθεί διαφορά στο πλήθος των αρχείων και στο μέγεθός τους μεταξύ client και server, τότε ενεργοποιούνται τα κουμπιά "Get D/L List" και "Get U/L List".
- Πατώντας τα κουμπιά "Get D/L List" και "Get U/L List" γίνεται σύγκριση των αρχείων που υπάρχουν στον client και στον server.



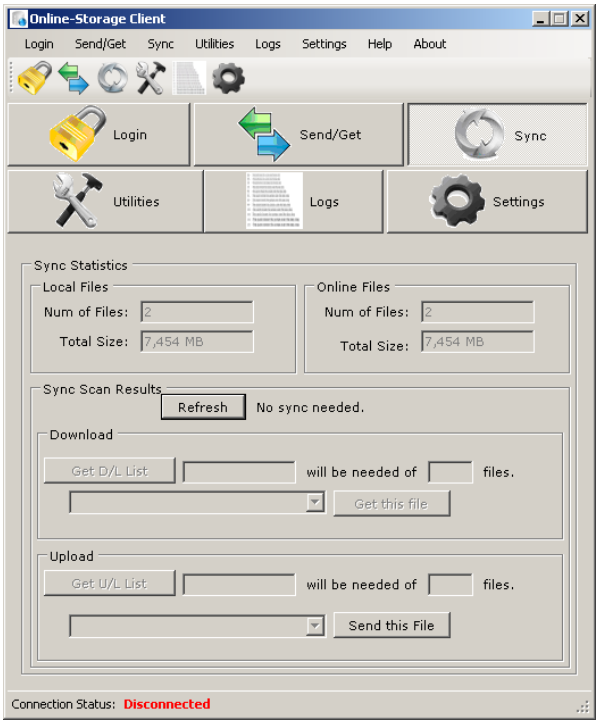
Εικόνα 56. SyncUpload List

- Στην περίπτωση που πατήσουμε το πλήκτρο "Get D/L List", αν διαπιστωθεί ότι τα αρχεία στον client είναι λιγότερα, τότε εμφανίζονται όλα τα αρχεία που λείπουν στο combo box που βρίσκεται στο Download πεδίο.
- Παρόμοια διαδικασία γίνεται στην περίπτωση "Get U/L List", μόνο που εξετάζεται η περίπτωση των αρχείων που πρόκειται να γίνουν Upload στον server.
- Αφού επιλέξουμε απο το combo box το αρχείο που θέλουμε να λάβουμε ή να στείλουμε, πατάμε το πλήκτρο "Get this file" ή "Send this file" αντίστοιχα.



Εικόνα 57. Sync Download List

- Γίνεται connection με τον server και εκκινείται η διαδικασία λήψης ή αποστολής του αρχείου που επιλέξαμε μέσω της port 5655 ή 5656 αντίστοιχα.
- Το αρχείο που λήφθηκε ή απεστάλη αφαιρείται απο το combo box.
- Επιβεβαίωση της ολοκλήρωσης αποστολής/λήψης όλων αρχείων γίνεται είτε μέσω του πλήκτρου Refresh είτε χειροκίνητα μέσω του πλήκτρου "Show Files" στην καρτέλα "Send/Get".



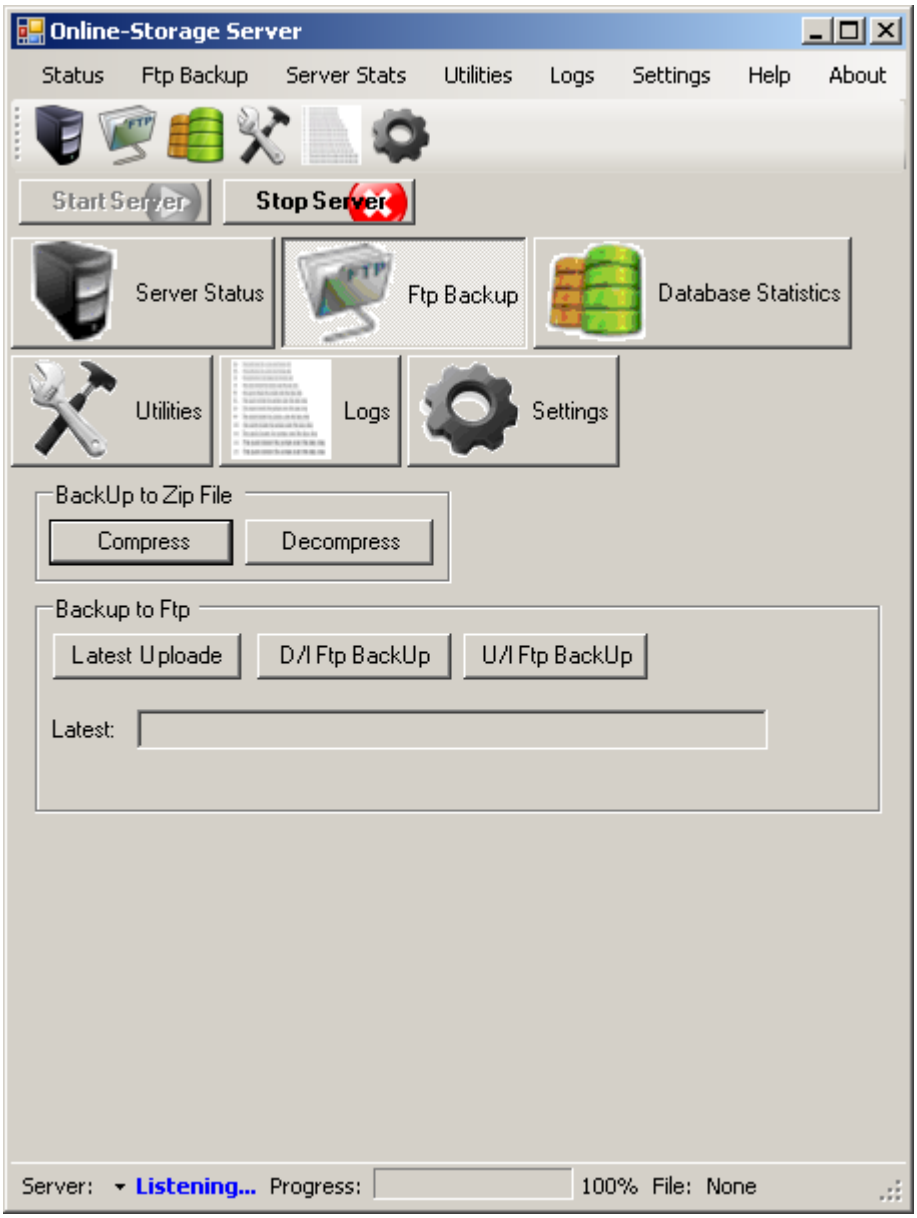
Εικόνα 58. Ολοκλήρωση Sync

10.2 Σενάρια χρήσης Server Εφαρμογής

10.2.1 Σενάριο συμπίεσης/αποσυμπίεσης backup αρχείου

Συμπίεση:

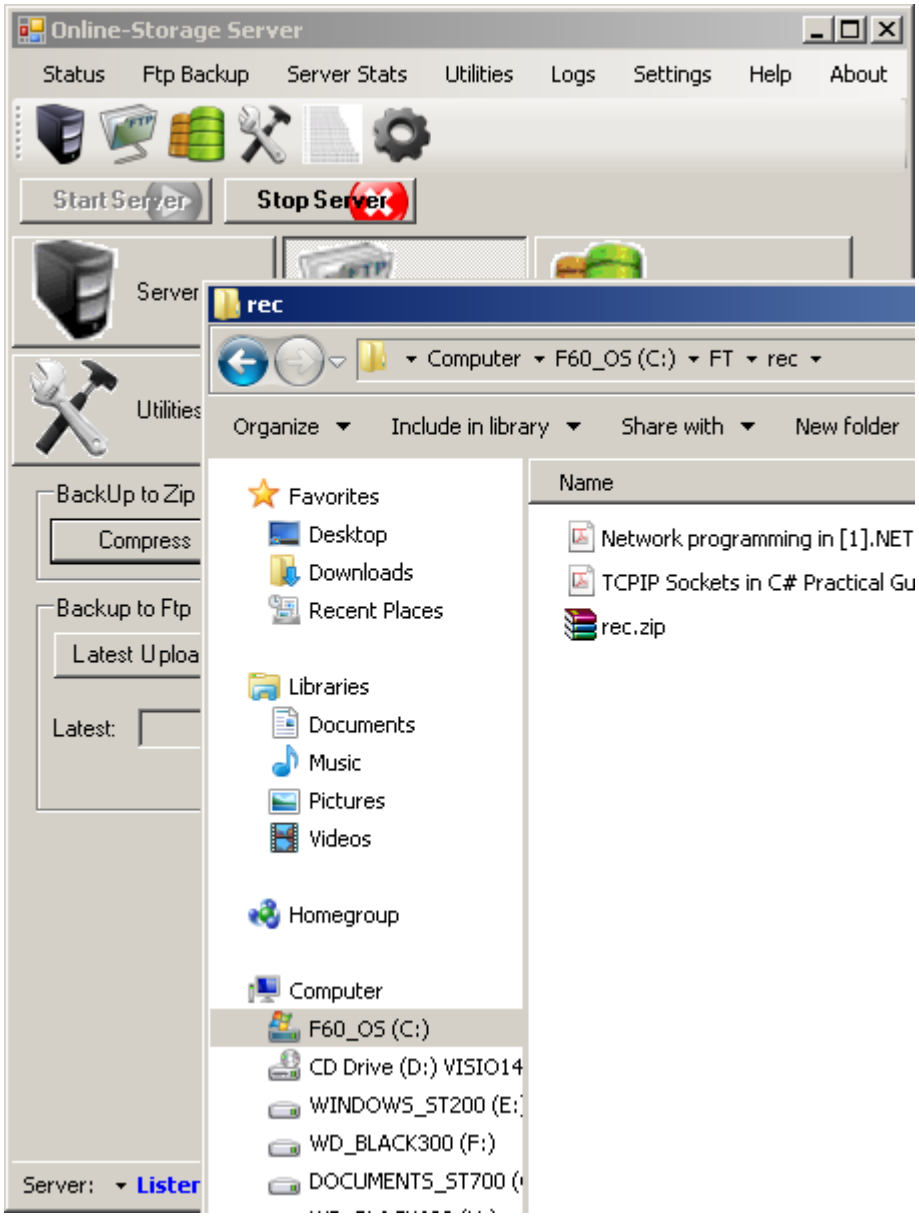
Στην καρτέλα Ftp Backup, όταν πατηθεί το κουμπί “Compress”, εκκινείται η εξής διαδικασία:



Εικόνα 59. Compress File

- Γίνεται έλεγχος αν υπάρχει στον κύριο φάκελο αποθήκευσης το αρχείο “rec.zip”. Αν υπάρχει τότε διαγράφεται διότι προέρχεται από παλαιότερο backup.

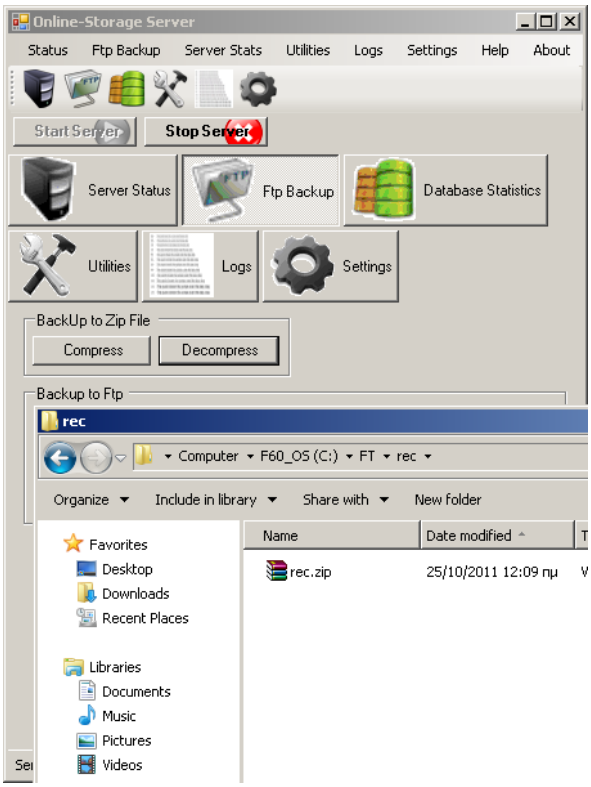
- Δημιουργείται το αρχείο rec.zip στον φάκελο του server που περιέχει όλα τα αρχεία που προϋπήρχαν.



Εικόνα 60. Το backup αρχείο "rec.zip"

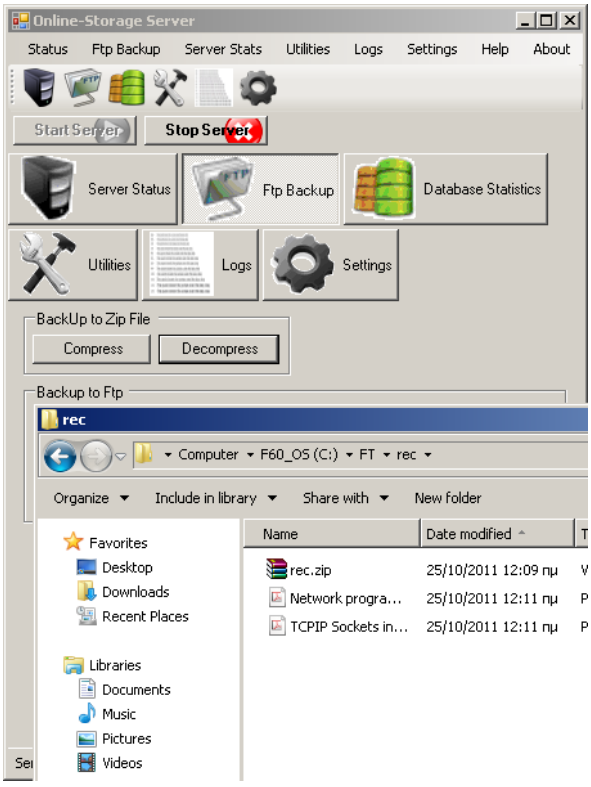
Αποσυμπίεση:

Στην καρτέλα Ftp Backup, όταν πατηθεί το κουμπί “Decompress”, εκκινείται η εξής διαδικασία:



Εικόνα 61. Πριν το Decompress

- Γίνεται έλεγχος αν υπάρχει στον κύριο φάκελο αποθήκευσης το αρχείο “rec.zip”.
- Αν υπάρχει ξεκινάει η διαδικασία αποσυμπίεσης όλων των αρχείων που υπάρχουν στο "rec.zip" στον root folder.

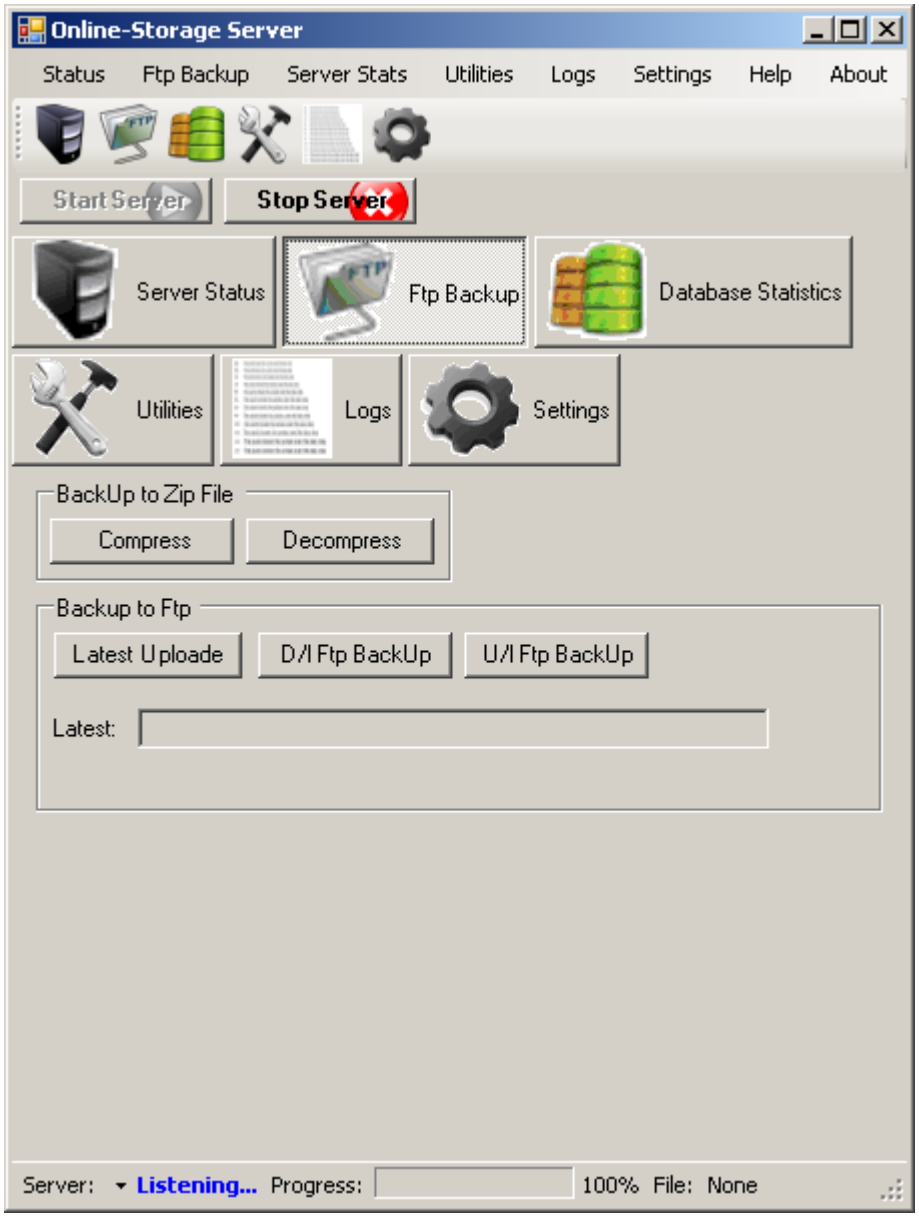


Εικόνα 62. Μετά το Decompress

10.2.2 Σενάριο χρήσης Ftp Server για απομακρυσμένη αποθήκευση των backup αρχείων

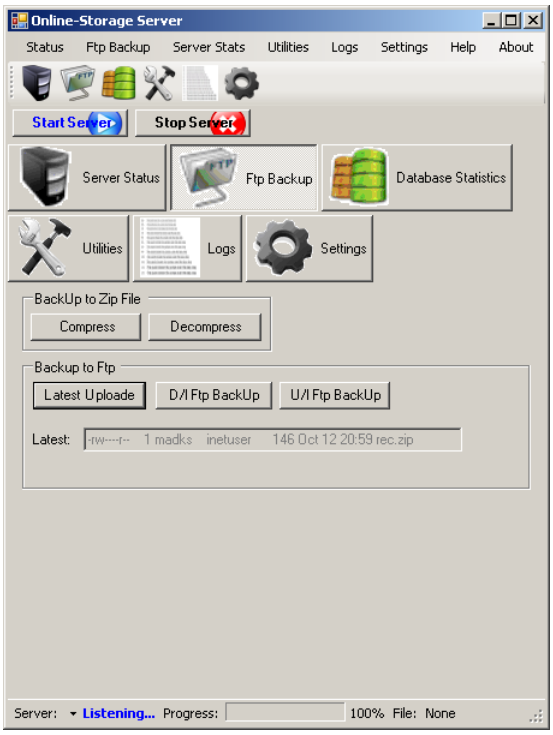
Πληροφορίες τελευταίου uploaded Backup σε FTP Server:

Με το πάτημα του κουμπιού “Latest Uploaded” γίνεται προβολή του τελευταίου backup αρχείου στον Ftp server.



Εικόνα 63. Καρτέλα Ftp Backup

- Γίνεται σύνδεση με τον ftp server με βάση το username, password, ftp server name που έχει οριστεί στα settings.
- Εμφανίζεται το όνομα, μέγεθος και ημερομηνία που έγινε upload το τελευταίο backup.

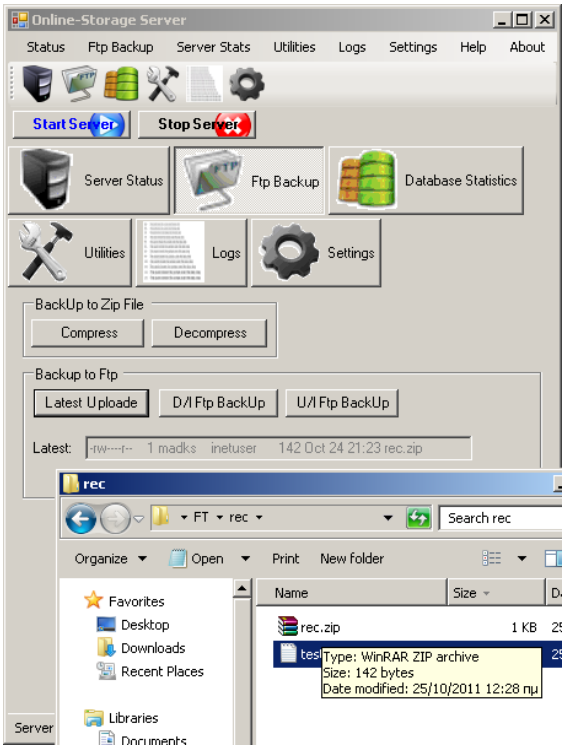


Εικόνα 64. Latest Uploaded Ftp backup

Λήψη backup αρχείου από τον FTP Server:

Με το πάτημα του κουμπιού “Download Ftp backup” γίνεται λήψη του τελευταίου backup αρχείου απο τον Ftp server.

- Γίνεται σύνδεση με τον ftp server με βάση το username, password, ftp server name που έχει οριστεί στα settings.

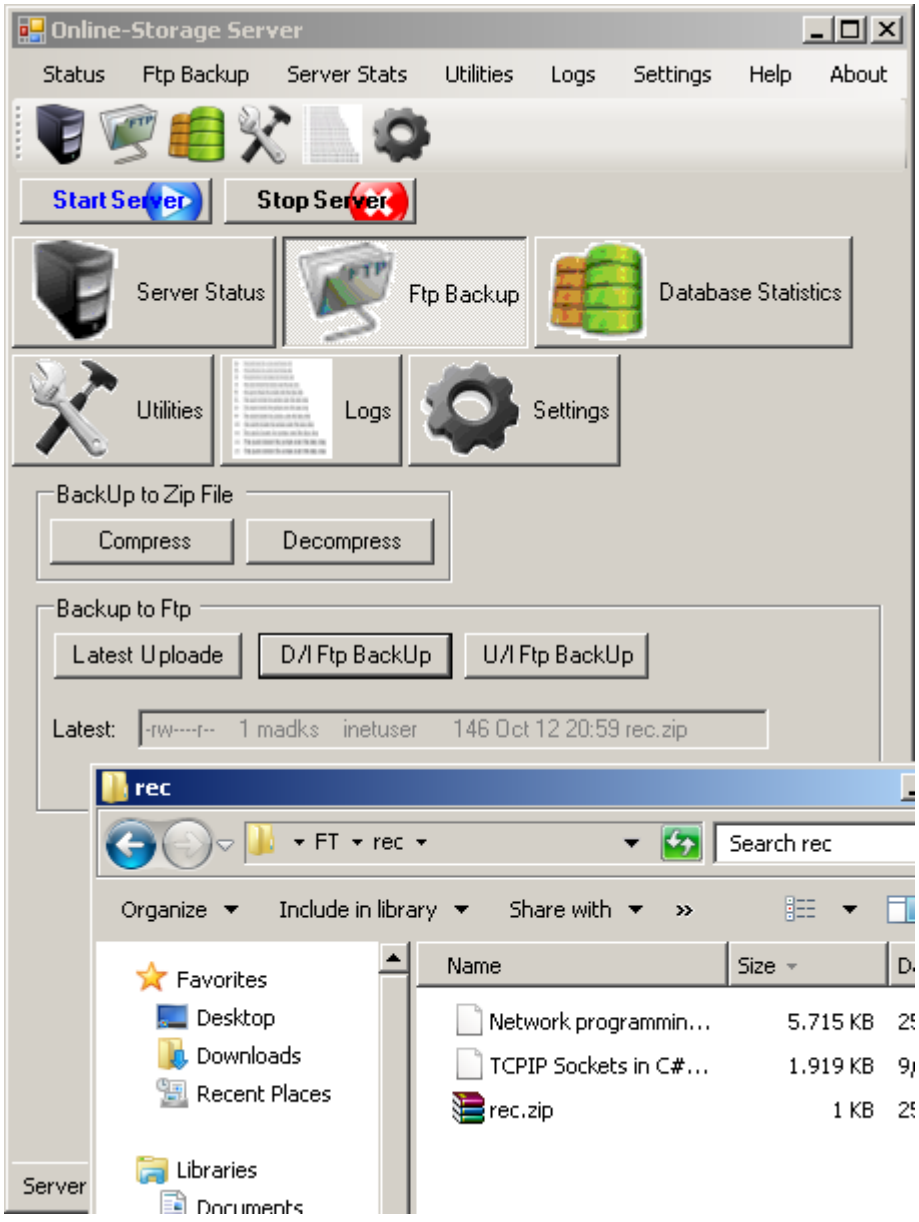


Εικόνα 65. Download backup file απο Ftp

- Γίνεται έλεγχος για το αν υπάρχει ήδη zip backup file στον φάκελο αποθήκευσης του server. Αν υπάρχει, το παλιό backup αρχείο σβήνεται.
- Ξεκινάει η λήψη του τελευταίου "rec.zip" απο τον Ftp server στον τοπικό φάκελο αποθήκευσης.

Upload backup αρχείου στον FTP Server:

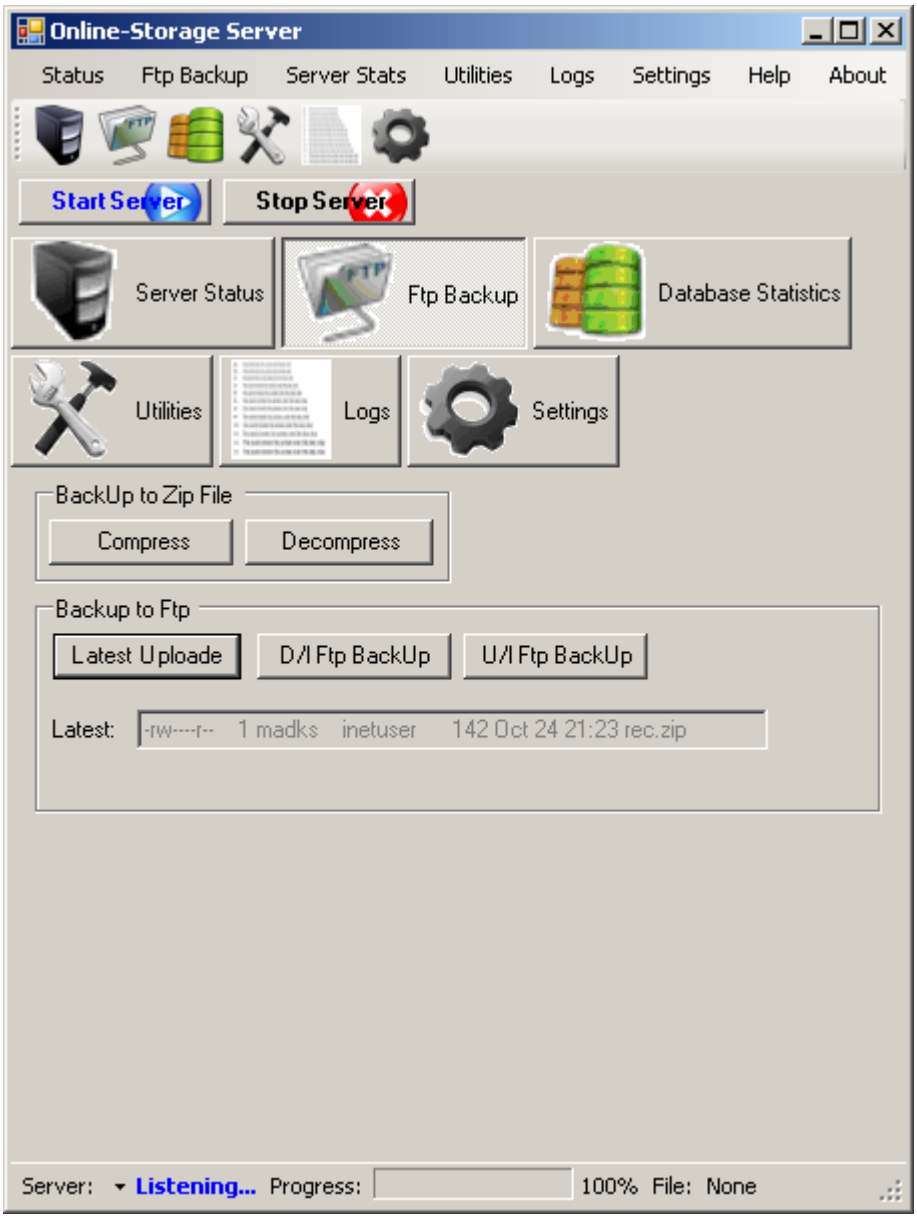
- Με το πάτημα του κουμπιού "Upload Ftp backup" γίνεται αποθήκευση του τελευταίου backup αρχείου στον Ftp server.



Εικόνα 66. Upload backup file στον Ftp

- Γίνεται σύνδεση με τον ftp server με βάση το username, password, ftp server name που έχει οριστεί στα settings.

- Γίνεται έλεγχος για το αν υπάρχει ήδη zip backup file στον φάκελο αποθήκευσης του server. Αν δεν υπάρχει, η διαδικασία σταματάει.
- Ξεκινάει η αποστολή του τελευταίου "rec.zip" απο τον τοπικό φάκελο αποθήκευσης, στον Ftp server.

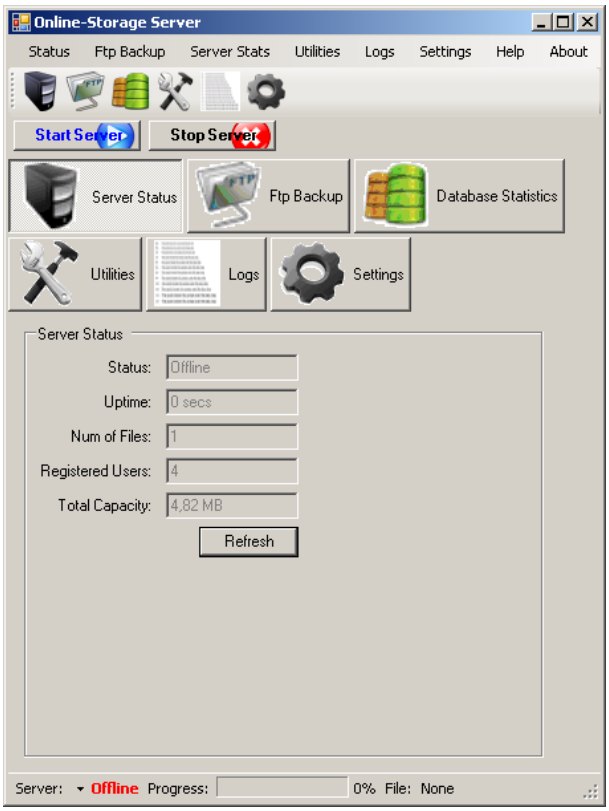


Εικόνα 67. Ανανέωση του Latest Uploaded Ftp backup

10.2.3 Λειτουργία εμφάνισης κατάστασης Server:

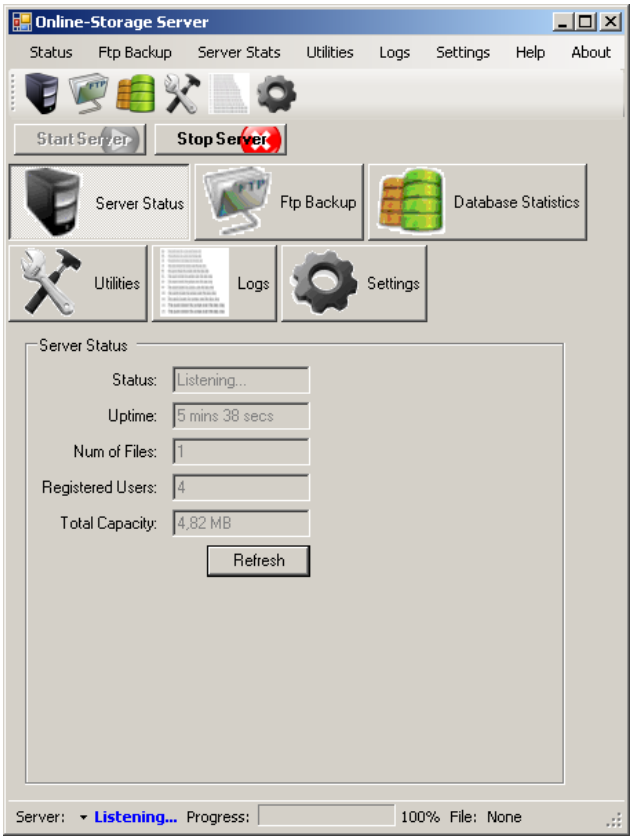
Το κουμπί “Refresh” στην καρτέλα Server status, αναλαμβάνει να ανανεώσει τα στατιστικά του Server.

- Μετράει τον αριθμό όλων τον αρχείων στο φάκελο c:\ft\rec
- Ελέγχει εάν υπάρχει σύνδεση με βάση δεδομένων και αν ναι, εμφανίζει τον αριθμό των registered users.
- Εμφανίζει αν ο server είναι σε Listening ή Offline status.



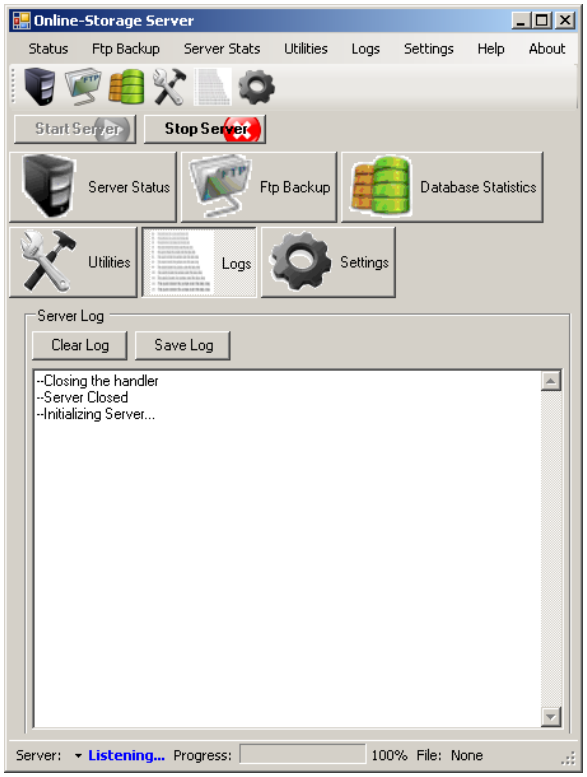
Εικόνα 68. Server status offline

- Εμφανίζει το συνολικό χρόνο που βρίσκεται σε Listening mode ο server.



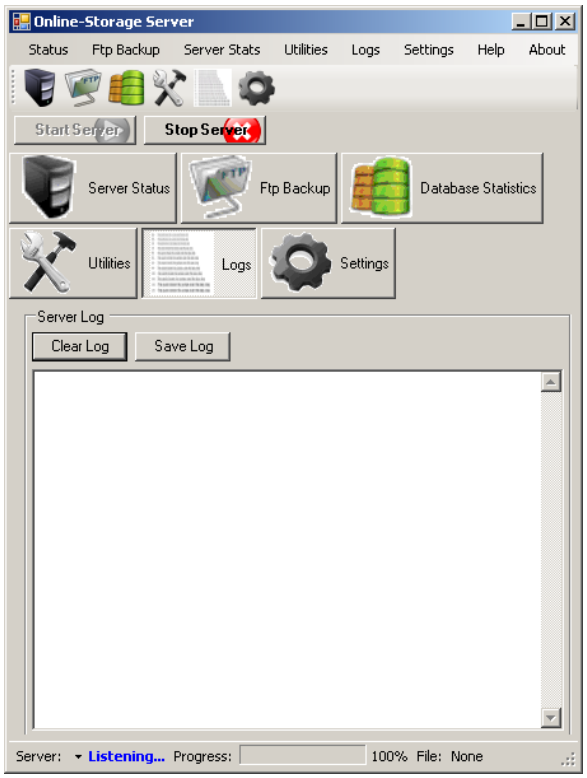
Εικόνα 69. Server status listening

10.2.4 Αρχεία καταγραφής – Logs



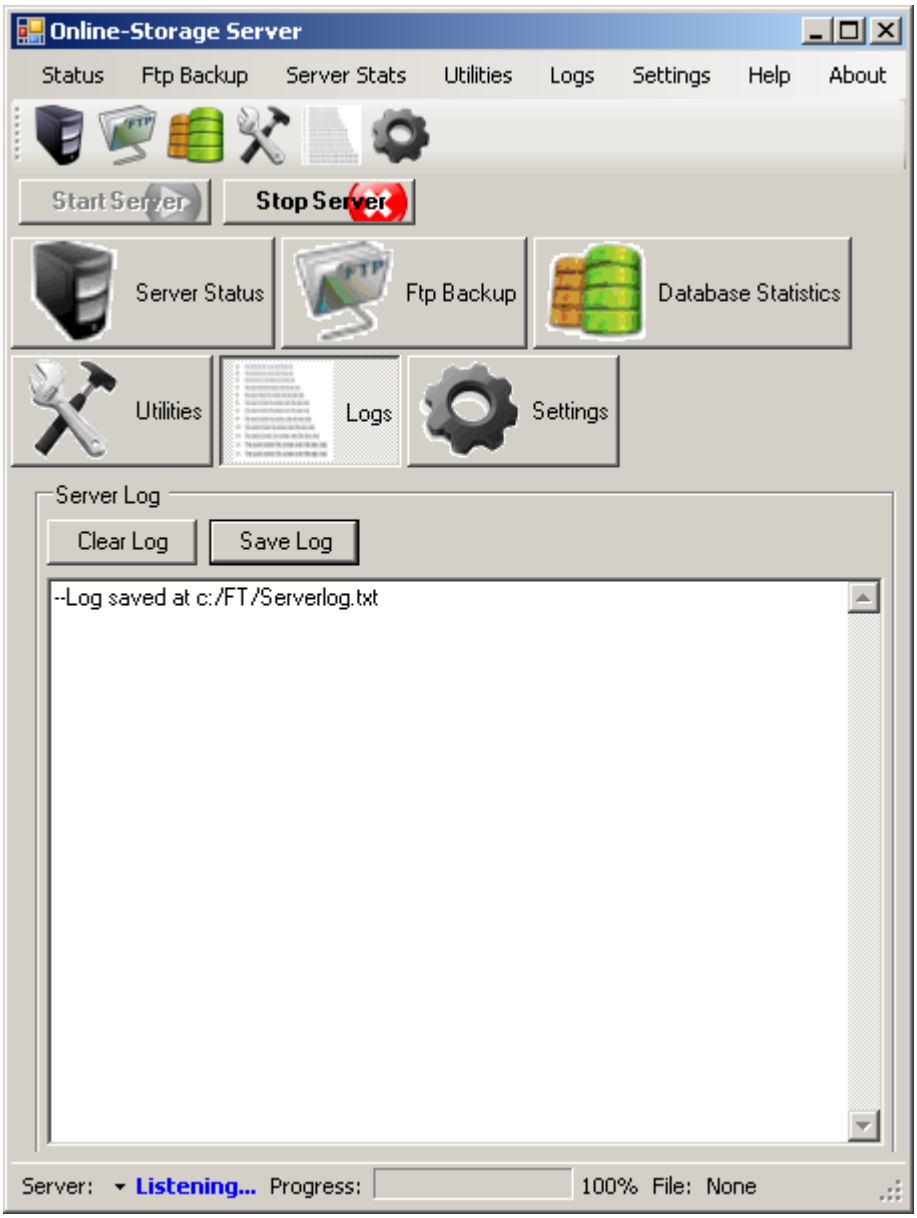
Εικόνα 70. Καρτέλα Logs

Το κουμπί "Clear log" αναλαμβάνει να καθαρίσει την προβολή του συγκεκριμένου log απο το αντίστοιχο text area.



Εικόνα 71. Clear log

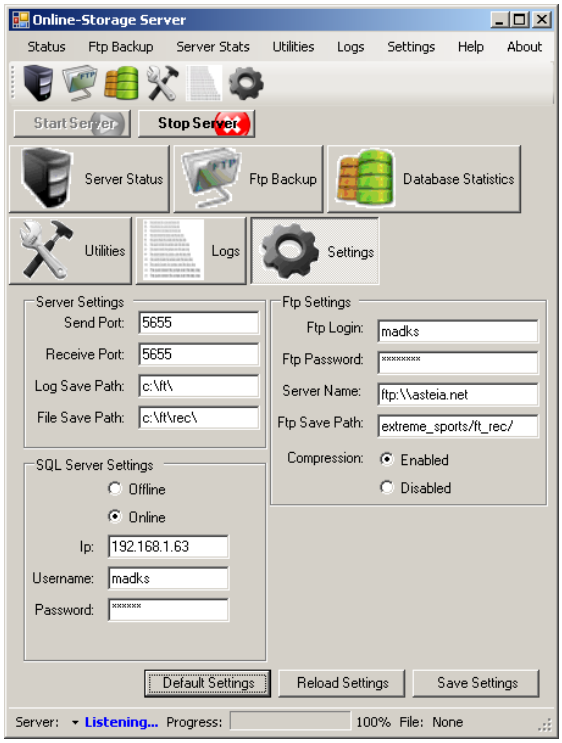
Το κουμπί "Save log" αποθηκεύει ότι text υπάρχει στο text area, στο αρχείο Clientlog.txt που βρίσκεται στο path "c:\FT\".



Εικόνα 72. Save log

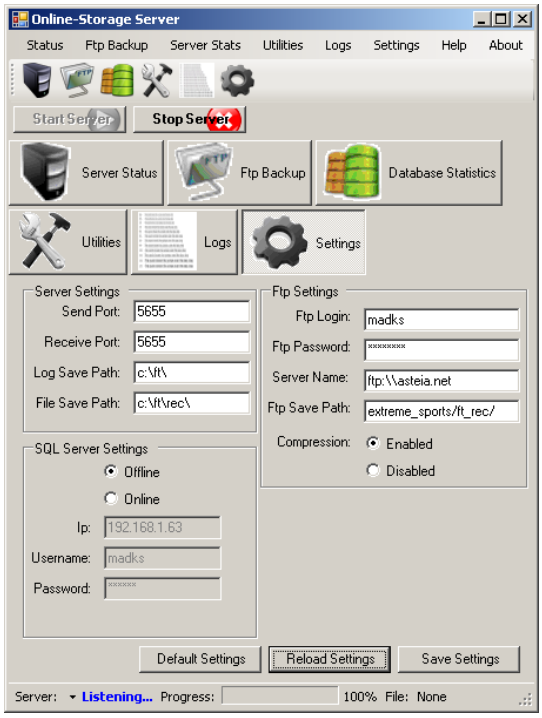
10.2.5 Λειτουργία αποθήκευσης – επαναφοράς ρυθμίσεων εφαρμογής

Το κουμπί "Default Settings" κάνει ανάγνωση τον default ρυθμίσεων που βρίσκονται στο αρχείο defaultServerSettings.xml και τις εμφανίζει στα αντίστοιχα πεδία.



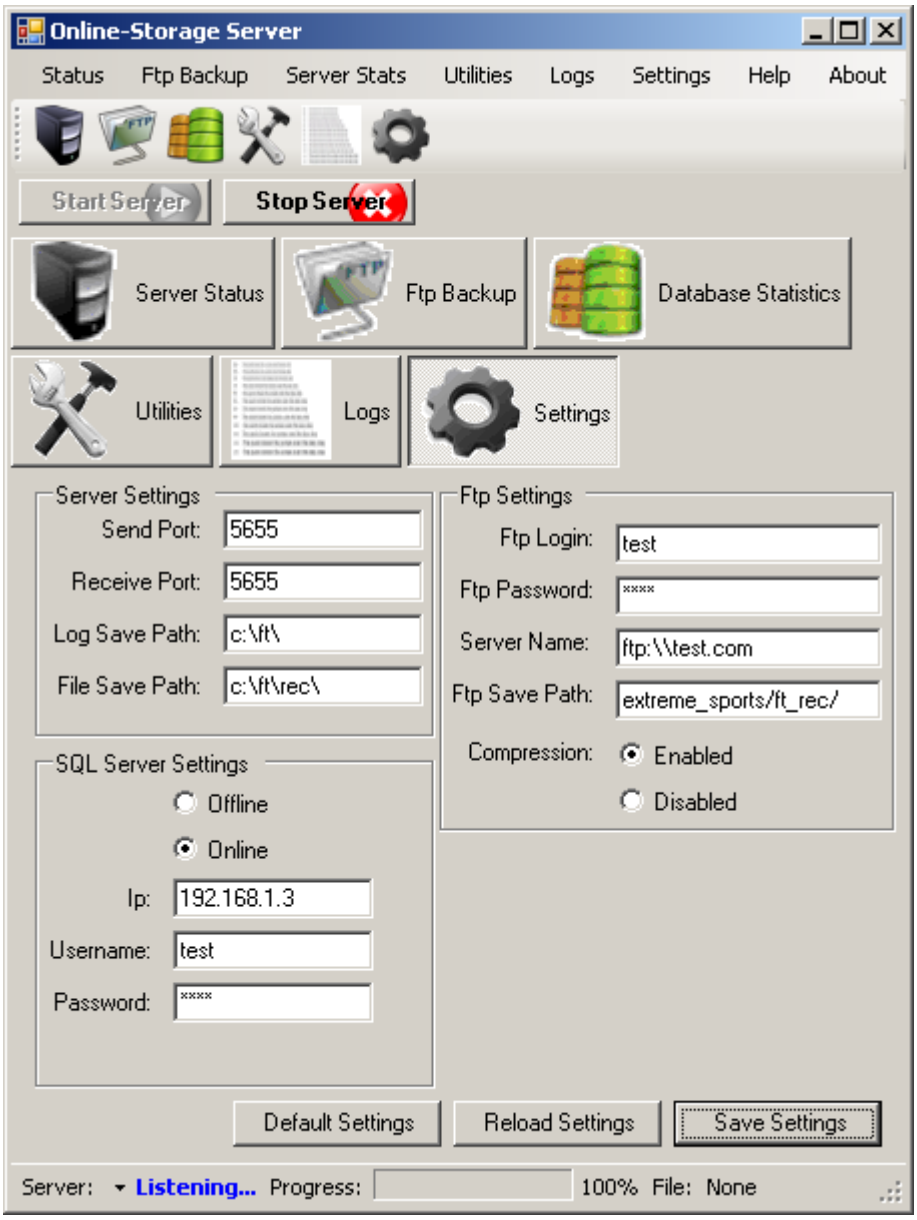
Εικόνα 73. Default Settings button

Το κουμπί "Reload Settings" κάνει ανάγνωση τον τελευταίων ρυθμίσεων που βρίσκονται στο αρχείο serverSettings.xml και τις εμφανίζει στα πεδία textbox.



Εικόνα 74. Reload Settings button

Το κουμπί "Save Settings" κάνει ανάγνωση όλων των τρεχόντων ρυθμίσεων που εμφανίζονται στα πεδία textbox και αποθηκεύει όλες αυτές τις ρυθμίσεις στο αρχείο serverSettings.xml.



Εικόνα 75. Save Settings button

10.3 Σενάρια λειτουργίας ASP.NET site

10.3.1 Εγγραφή χρήστη

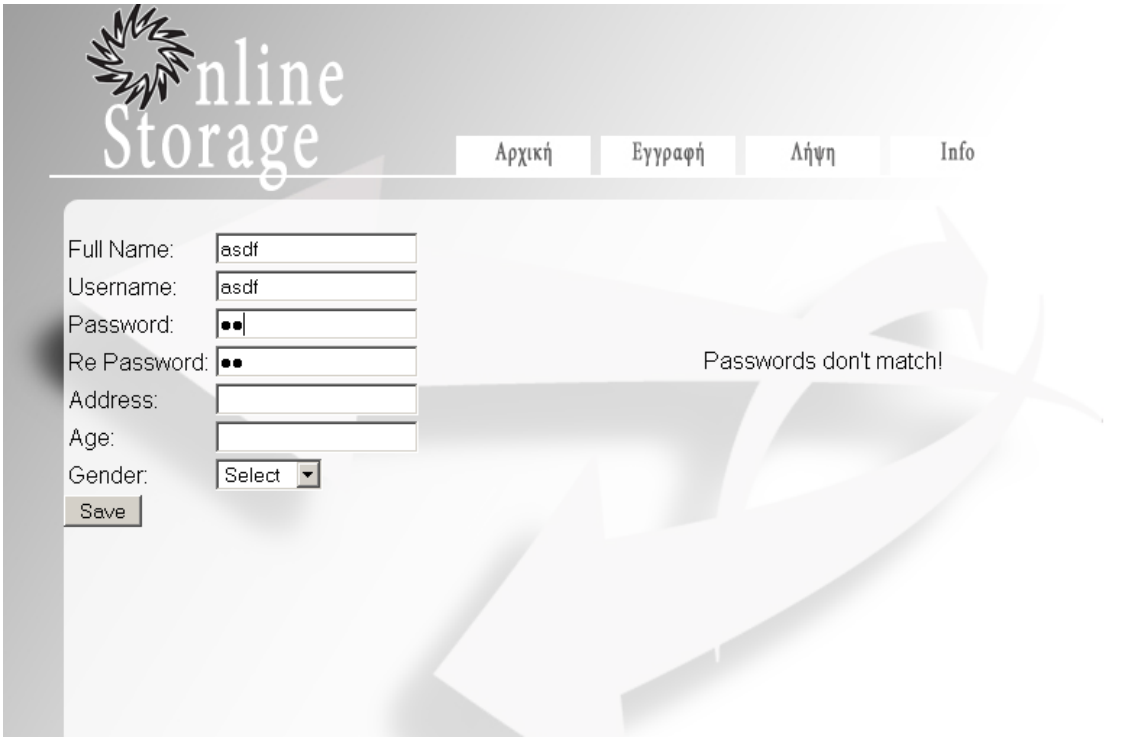
Για την εγγραφή νέου χρήστη για την δυνατότητα χρήσης της Online Storage πλατφόρμας:

Εικόνα 76. Register.aspx

- Ο χρήστης θα χρειαστεί να πλοηγηθεί στην ιστοσελίδα http://online-storage.dyndns.org.

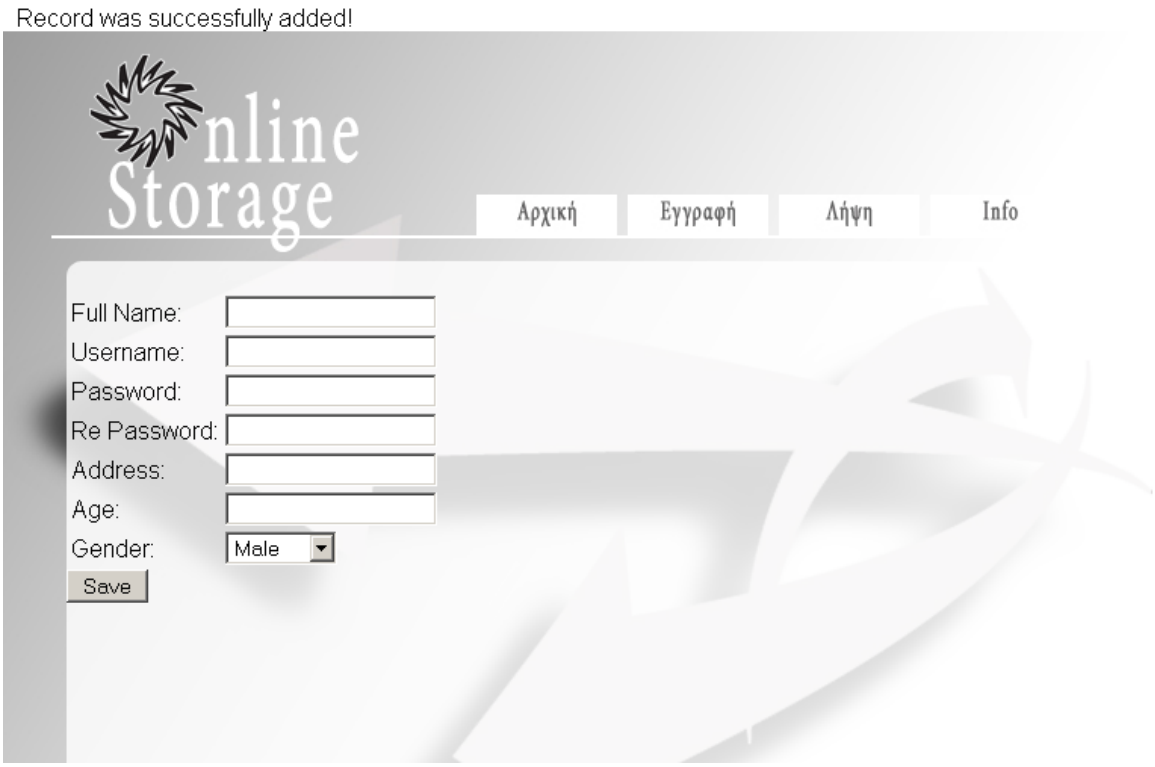
Εικόνα 77. Ελλιπή δεδομένα

- Πατώντας στον μενού «Λήψη» της ιστοσελίδας (ή αλλιώς στο link http://online-storage.dyndns.org/Register.aspx), ο χρήστης μεταφέρεται στην σελίδα εγγραφής νέου χρήστη.
- Συμπληρώνει ο χρήστης τα πεδία και έπειτα πατάει το κουμπί “Save”.
- Σε περίπτωση μη αποδεκτής εισαγωγής σε κάποιο από τα πεδία, εμφανίζεται μήνυμα λάθους στα δεξιά του αντίστοιχου πεδίου.



Εικόνα 78. Λανθασμένα δεδομένα

- Με την επιτυχή εισαγωγή την εγγραφής του χρήστη, εμφανίζεται μήνυμα επιβεβαίωσης στο άνω τμήμα της σελίδας.

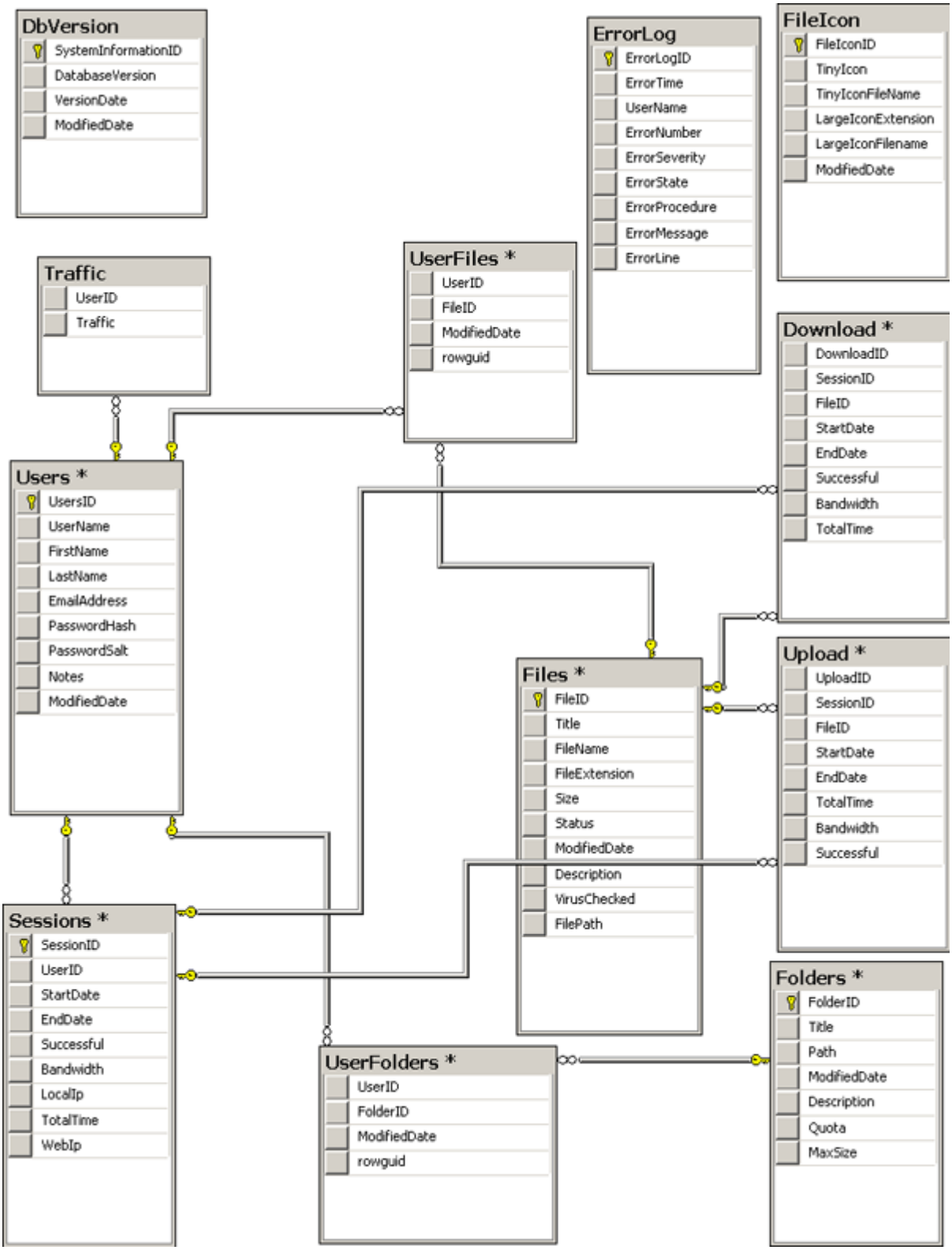


Εικόνα 79. Επιτυχές Register

ΚΕΦΑΛΑΙΟ 11ο

ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ ΠΛΑΤΦΟΡΜΑΣ

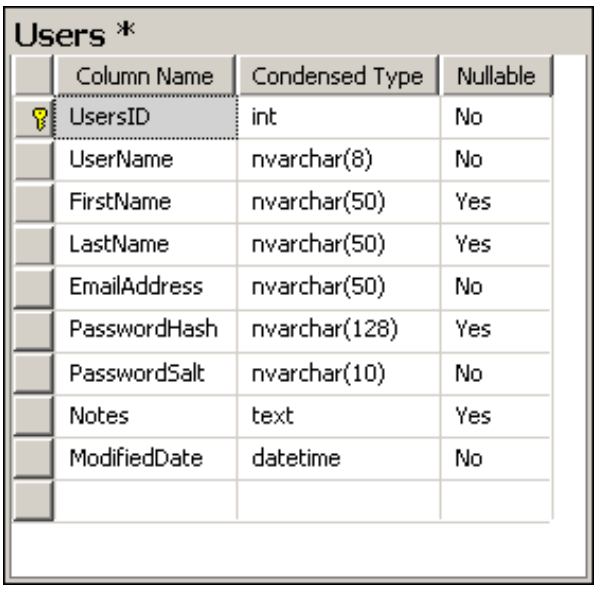
11.1 Διάγραμμα οντοτήτων - συσχετίσεων της κοινής Βάσης Δεδομένων



Εικόνα 80. Βάση Δεδομένων Πλατφόρμας

11.2 Οντότητες της κοινής Βάσης Δεδομένων

11.2.1 Οντότητα dbo.Users



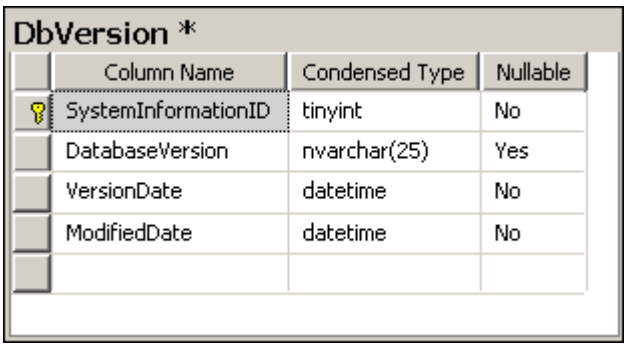
Εικόνα 81. Οντότητα dbo.Users

Στην οντότητα αυτή αποθηκεύονται όλοι οι χρήστες που θα έχουν πρόσβαση στην πλατφόρμα Online-Storage. Χρησιμοποιείται απο το Asp.NET site για την εγγραφή νέου χρήστη.

Πίνακας 13. Attributes dbo.Users

Γνωρίσματα (attributes)	Περιγραφή
UsersID	Ο μοναδικός αριθμός αναγνώρισης του χρήστη, αποτελεί πρωτεύον κλειδί της οντότητας, έχει την ιδιότητα auto increment.
UserName	Προσδιορίζει το user name με το οποίο θα συνδέεται ο χρήστης.
FirstName	Το όνομα του χρήστη.
LastName	Το επίθετο του χρήστη.
EmailAddress	Η email διεύθυνση του χρήστη.
PasswordHash	Ο κωδικός του χρήστη, χρησιμοποιείται για το authentication.
PasswordSalt	Χρησιμοποιείται σε συνδυασμό με το PasswordHash.
Motes	Σημειώσεις που πρόσθεσε ο χρήστης κατά την εγγραφή.
ModifiedDate	Ημερομηνία και ώρα εγγραφής του χρήστη.

11.2.2 Οντότητα dbo.DbVersion



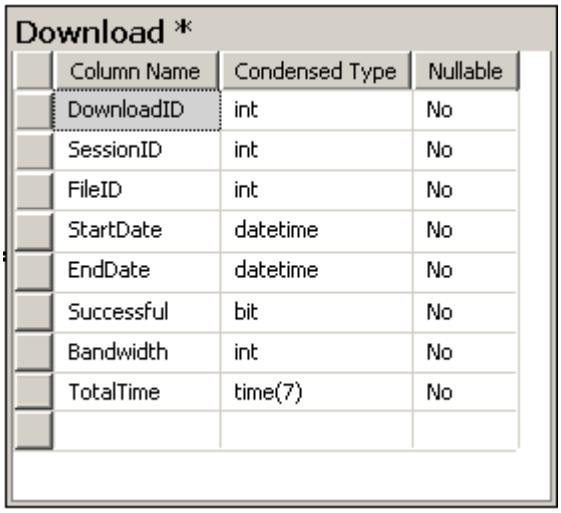
Εικόνα 82. Οντότητα dbo.DbVersion

Η οντότητα αυτή περιέχει μια εγγραφή για κάθε έκδοση. Χρησιμοποιείται για την ταυτοποίηση της τρέχουσας έκδοση της βάσης δεδομένων. Η εγγραφή με την μεγαλύτερη τιμή του γνωρίσματος SystemInformationID περιγράφει και την τελευταία-τρέχουσα έκδοση της βάσης δεδομένων.

Πίνακας 14. Attributes dbo.DbVersion

Γνωρίσματα(attributes)	Περιγραφή
SystemInformationID	Αύξων αριθμός της συγκεκριμένης έκδοσης.
DatabaseVersion	Η τρέχουσα έκδοση της βάσης δεδομένων.
VersionDate	Ημερομηνία δημιουργίας της έκδοσης αυτής.
ModifiedDate	Τελευταία ημερομηνία τροποποίησης της έκδοσης.

11.2.3 Οντότητα dbo.Download



Εικόνα 83. Οντότητα dbo.Download

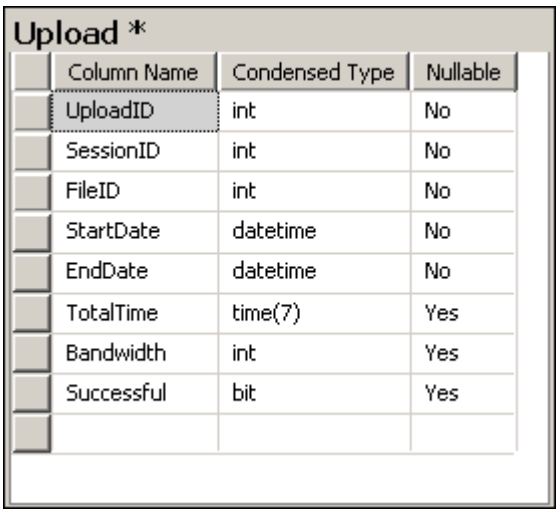
Η οντότητα Download περιγράφει κάθε διαδικασία download που ξεκίνησε απο ένα συγκεκριμένο session.

Πίνακας 15. Attributes dbo.Download

Γνωρίσματα(attributes)	Περιγραφή
DownloadID	Αριθμός αναγνώρισης του συγκεκριμένου download.
SessionID	Αριθμός αναγνώρισης του session που δημιουργήθηκε για το συγκεκριμένο download.
FileID	Αριθμός αναγνώρισης του αρχείου που έγινε download.
StartDate	Ημερομηνία - ώρα έναρξης του download.
EndDate	Ημερομηνία - ώρα λήξης του download.
Successful	Boolean οντότητα που προσδιορίζει εάν ήταν επιτυχές το download ή όχι.
Bandwidth	Bandwidth λήψης του αρχείου.
TotalTime	Συνολικός χρόνος που χρειάστηκε το download

.

11.2.4 Οντότητα dbo.Upload



Εικόνα 84. Οντότητα dbo.Upload

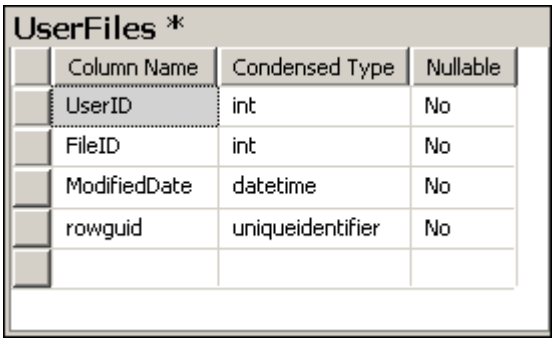
Η οντότητα Upload περιγράφει κάθε διαδικασία upload που ξεκίνησε απο ένα συγκεκριμένο session.

Πίνακας 16. Attributes dbo.Upload

Γνωρίσματα(attributes)	Περιγραφή

UploadID	Αριθμός αναγνώρισης του συγκεκριμένου upload.
SessionID	Αριθμός αναγνώρισης του session που δημιουργήθηκε για το συγκεκριμένο upload.
FileID	Αριθμός αναγνώρισης του αρχείου που έγινε upload
StartDate	Ημερομηνία - ώρα έναρξης του upload.
EndDate	Ημερομηνία - ώρα λήξης του upload.
Successful	Boolean οντότητα που προσδιορίζει εάν ήταν επιτυχές το upload ή όχι.
Bandwidth	Bandwidth ανεβάσματος του αρχείου.
TotalTime	Συνολικός χρόνος που χρειάστηκε το upload.

11.2.5 Οντότητα dbo.UserFiles



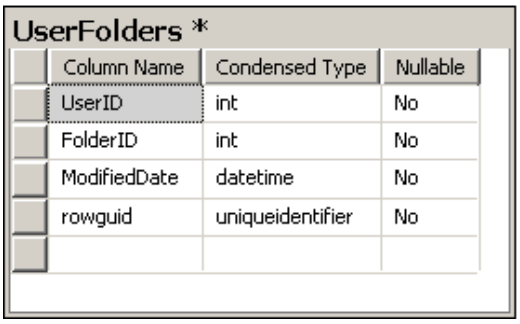
Εικόνα 85. Οντότητα dbo.UserFiles

Αυτή η οντότητα αναλαμβάνει να αντιστοιχήσει ένα αρχείο με έναν συγκεκριμένο χρήστη.

Πίνακας 17. Attributes dbo.UserFiles

Γνωρίσματα(attributes)	Περιγραφή
UserID	Αριθμός αναγνώρισης του χρήστη.
FileID	Αριθμός αναγνώρισης του αρχείου.
ModifiedDate	Ημερομηνία τροποποίησης του αρχείου.
rowguid	Χρησιμοποιείται για να ορίσει την μοναδικότητα της εγγραφής

11.2.6 Οντότητα dbo.UserFolders



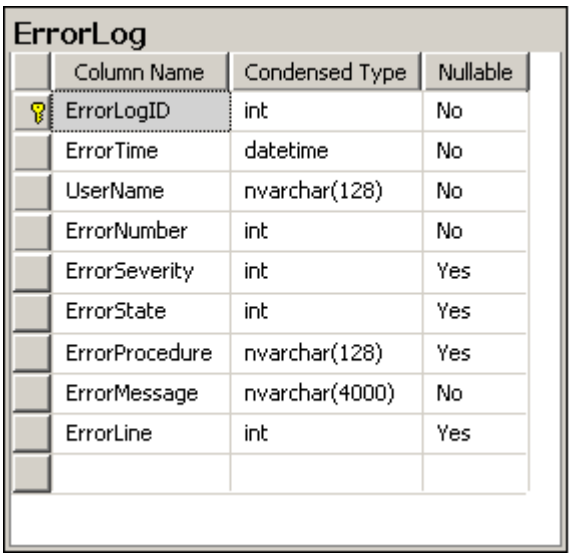
Εικόνα 86. Οντότητα dbo.UserFolders

Αυτή η οντότητα αναλαμβάνει να αντιστοιχήσει ένα φάκελο με έναν συγκεκριμένο χρήστη.

Πίνακας 18. Attributes dbo.UserFolders

Γνωρίσματα(attributes)	Περιγραφή
UserID	Αριθμός αναγνώρισης του χρήστη.
FolderID	Αριθμός αναγνώρισης του φακέλου.
ModifiedDate	Ημερομηνία τροποποίησης του φακέλου.
rowguid	Χρησιμοποιείται για να ορίσει την μοναδικότητα της εγγραφής.

11.2.7 Οντότητα dbo.ErrorLog



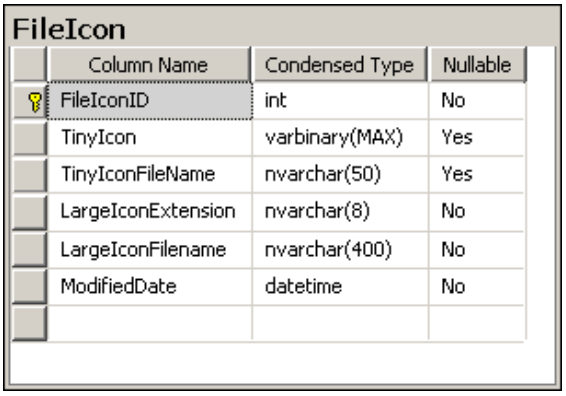
Εικόνα 87. Οντότητα dbo.ErrorLog

Η οντότητα Error log, καταγράφει με λεπτομέρεια τα σφάλματα που δημιουργήθηκαν κατά την εκτέλεση της πλατφόρμας για μελλοντικό debugging.

Πίνακας 19. Attributes dbo.ErrorLog

Γνωρίσματα(attributes)	Περιγραφή
ErrorLogID	Μοναδικός αριθμός αναγνώρισης του σφάλματος.
ErrorTime	Ημερομηνία - ώρα που συνέβη το σφάλμα.
UserName	Το όνομα του χρήστη στον οποίον συνέβη το σφάλμα.
ErrorNumber	Ο αριθμός σφάλματος.
ErrorSeverity	Η σοβαρότητα του σφάλματος.
ErrorState	Η κατάσταση που προκάλεσε το σφάλμα.
ErrorProcedure	Η διαδικασία η οποία προκάλεσε το συγκεκριμένο σφάλμα.
ErrorMessage	Το μήνυμα του σφάλματος.
ErrorLine	Η γραμμή στην οποία συνέβη το σφάλμα.

11.2.8 Οντότητα dbo.FileIcon



Εικόνα 88. Οντότητα dbo.FileIcon

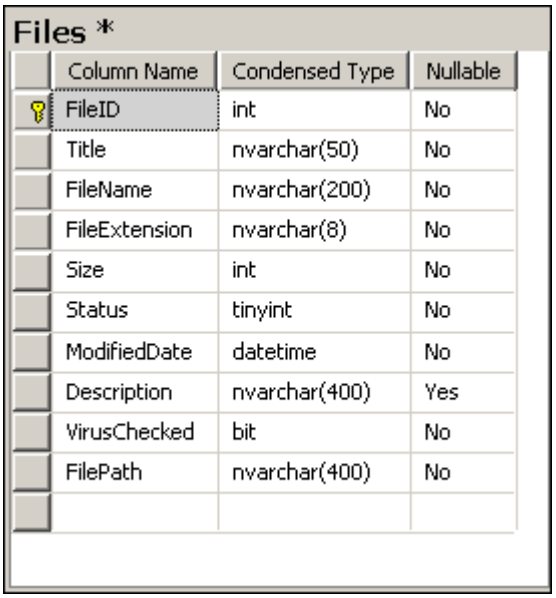
Η οντότητα FileIcon είτε αναλαμβάνει απλά την καταγραφή του file path (σε περίπτωση large icon), είτε αποθηκεύει icon στην βάση δεδομένων (στην περίπτωση tiny icon).

Πίνακας 20. Attributes dbo.FileIcon

Γνωρίσματα(attributes)	Περιγραφή
FileIconID	Ο μοναδικός αριθμός αναγνώρισης το icon.
TinyIcon	Στην περίπτωση χρήσης μικρού icon, αποθηκεύεται στην βάση δεμένων υπό τον τύπο Varbinary(MAX).
TinyIconFileName	Το file name του μικρού icon.

LargeIconExtension	Το extension του icon σε περίπτωση χρήσης μεγάλου icon.
LargeIconFileName	Το όνομα του αρχείου/πλήρες path, σε περίπτωση χρήσης μεγάλου icon.
ModifiedDate	Η ημερομηνία τροποποίησης του icon.

11.2.9 Οντότητα dbo.Files



Εικόνα 89. Οντότητα dbo.Files

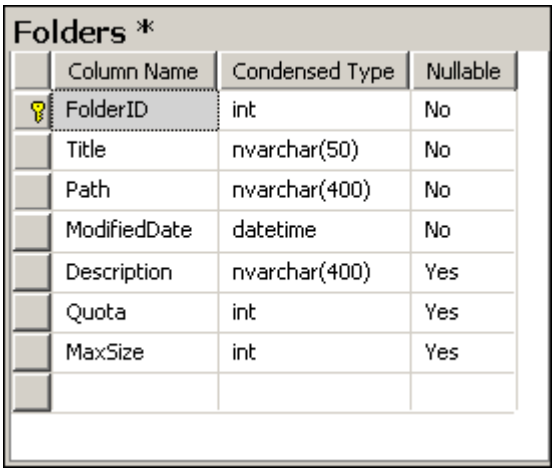
Καταγράφει το κάθε αρχείο που υπάρχει αποθηκευμένο στον server, καθώς και λεπτομέρειες (όπως την ημερομηνία τροποποίησης, περιγραφή, extension κτλ.).

Πίνακας 21. Attributes dbo.Files

Γνωρίσματα(attributes)	Περιγραφή
FileID	Μοναδικός αριθμός.
Title	Τίτλος του αρχείου.
FileName	Όνομα του αρχείου χωρίς την προέκταση.
FileExtension	Προέκταση του αρχείου.
Size	Μέγεθος του αρχείου.
Status	Κατάσταση του αρχείου.
ModifiedDate	Ημερομηνία τελευταίας τροποποίησης του αρχείου.
Description	Περιγραφή του αρχείου.

VirusChecked	Περιγράφει για το αν έχει γίνει έλεγχος με antivirus ή όχι για το συγκεκριμένο αρχείο.
FilePath	Το path αποθήκευσης του αρχείου.

11.2.10 Οντότητα dbo.Folders



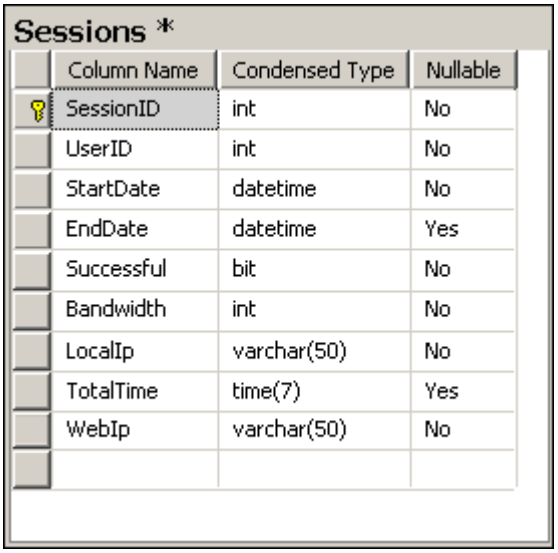
Εικόνα 90. Οντότητα dbo.Folders

Καταγράφει τον κάθε φάκελο που υπάρχει αποθηκευμένος στον server, καθώς και λεπτομέρειες (όπως την ημερομηνία τροποποίησης, περιγραφή, μέγεθος κτλ.).

Πίνακας 22. Attributes dbo.Folders

Γνωρίσματα(attributes)	Περιγραφή
FolderID	Μοναδικός αριθμός του φακέλου.
Title	Τίτλος του φακέλου.
Path	Διαδρομή αποθήκευσης του φακέλου.
ModifiedDate	Ημερομηνία τελευταίας τροποποίησης του φακέλου.
Description	Περιγραφή του φακέλου.
Quota	Όριο μέγεθος του φακέλου.
MaxSize	Μέγιστο μέγεθος του φακέλου.

11.2.11 Οντότητα dbo.Sessions



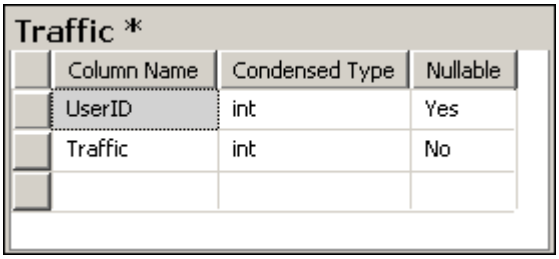
Εικόνα 91. Οντότητα dbo.Sessions

Καταγράφει το κάθε session μεταξύ user και server application.

Πίνακας 23. Attributes dbo.Sessions

Γνωρίσματα(attributes)	Περιγραφή
SessionID	Αριθμός αναγνώρισης του session που δημιουργήθηκε για το συγκεκριμένο download.
UserID	Αριθμός αναγνώρισης του χρήστη για το συγκεκριμένο session.
StartDate	Ημερομηνία - ώρα έναρξης του session.
EndDate	Ημερομηνία - ώρα λήξης του session.
Successful	Boolean οντότητα που προσδιορίζει εάν ήταν επιτυχές το session ή όχι.
Bandwidth	Bandwidth του session.
LocalIp	Τοπική ip (σε περίπτωση χρήσης του custom mode).
TotalTime	Συνολικός χρόνος που χρειάστηκε το download.
WebIp	Web ip (σε περίπτωση χρήσης του online mode)

11.2.12 Οντότητα dbo.Traffic



Εικόνα 92. Οντότητα dbo.Traffic

Καταγράφει την τρέχουσα κίνηση (download + upload) ενός συγκεκριμένου χρήστη.

Πίνακας 24. Attributes dbo.Traffic

Γνωρίσματα(attributes)	Περιγραφή
UserID	Αριθμός αναγνώρισης του χρήστη.
Traffic	Συνολική κίνηση (upload + download bandwidth)

ΒΙΒΛΙΟΓΡΑΦΙΑ

Βιβλία

1. Fiach Reid – **"Network Programming in .NET (with C# and Visual Basic.NET)"** – *Elsevier Digital Press*
2. Andrew Troelsen – **"Pro C# 2010 and the .NET 4 Platform (Fifth Edition)"** – *Apress*
3. Matthew MacDonald – **"Beginning ASP.NET 3.5 in C# 2008 (From Novice to Professional – Second Edition)"** – *Apress*
4. John Sharp – **"Microsoft Visual C# 2010 (Step by Step)"** – *Microsoft Press*
5. Ian Griffiths, Matthew Adams, Jesse Liberty – **"Programming C# 4.0"** – *O'Reilly*
6. Joseph Albahari, Ben Albahari – **"C# 4.0 in a Nutshell (Fourth Edition)"** – *O'Reilly*
7. David B. Makofske, Michael J. Donahoo, Kenneth L. Calvert – **"TCP/IP Sockets in C# (Practical Guide for Programmers)"** – *Elsevier Digital Press*
8. Andrew Stellman, Jennifer Greene – **"Head First C# (A Brain-Friendly Guide – Second Edition)"** – *O'Reilly*
9. Charlie Russel, Craig Zacker – **"Introducing Windows Server 2008 R2"** – *Microsoft Press*
10. Todd Lammle – **"CCNA: Cisco Certified Network Associate (Study Guide – Sixth Edition)"** – *Wiley Publishing, Inc.*

Ιστοσελίδες

1. FileStream.Write Method - http://msdn.microsoft.com/ens/library/system.io.filestream.write.aspx
2. Array Copy Method – http://msdn.microsoft.com/en-us/library/z50k9bft.aspx
3. C# Byte Arrays – http://www.dotnetperls.com/byte-array
4. File Methods - http://msdn.microsoft.com/en-us/library/3z2ck8eh.aspx
5. BinaryReader.ReadBytes Method – http://msdn.microsoft.com/en-us/library/system.io.binaryreader.readbytes.aspx
6. C# FileStream Explained – http://www.aspfree.com/c/a/C-Sharp/C-Sharp-FileStream-Explained/
7. FileStream Open File – http://www.csharp-examples.net/filestream-open-file/
8. FileStream Class – http://msdn.microsoft.com/en-us/library/system.io.filestream.aspx
9. Terminating non-essential threads in C# - http://stackoverflow.com/questions/355019/automatically-terminating-non-essential-threads-in-c
10. Threading in C# - http://www.albahari.com/threading/part3.aspx

11. File transfer with Network streams - http://www.eggheadcafe.com/software/aspnet/29777478/how-to-tranfer-file-using-networkstream-through-sockets-in-c.aspx
12. The C# Asynchronous Programming Model – http://www.c-sharpcorner.com/UploadFile/logisimo/107252009003250AM/1.aspx
13. Client/Server: Simple File Sharing – http://www.csharphelp.com/2006/10/client-server-simple-file-sharing-and-file-transfer/
14. Asynchronous File Transfer Issues – http://social.msdn.microsoft.com/forums/en-US/netfxnetcom/thread/29b1854d-998a-4c1c-8c36-89499c65e1f0
15. Asynchronous Stream Processing – http://msdn.microsoft.com/en-us/magazine/cc337900.aspx
16. Asynchronous File I/O – http://msdn.microsoft.com/en-us/library/kztecsys%28VS.71%29.aspx
17. File Transfer using C# .NET 2.0 - http://www.codeproject.com/KB/cs/SocketApplication.aspx
18. Sending files over network – http://www.gamedev.net/topic/485998-sending-an-exe-file-over-network-using-asynchronous-c-sockets/
19. C# DataGridView Tips – http://www.dotnetperls.com/datagridview-tips
20. C# DataGridView Tutorial – http://www.dotnetperls.com/datagridview-tutorial
21. FTP Client Library – http://www.csharphelp.com/2005/11/ftp-client-library-in-c/
22. Simple FTP Demo Application in C# - http://www.codeguru.com/csharp/csharp/cs_internet/desktopapplications/article.php/c13163
23. Ftp Class Library in C# - http://www.dreamincode.net/forums/topic/35902-create-an-ftp-class-library-in-c%23/
24. Ftp C# Examples – http://www.example-code.com/csharp/ftp.asp
25. C# FTP Upload – http://www.vcskicks.com/csharp_ftp_upload.php
26. C# FTP Client Library- http://www.codeproject.com/script/Articles/ViewDownloads.aspx?aid=10968
27. FtpWebRequest Class – http://msdn.microsoft.com/en-us/library/system.net.ftpwebrequest.aspx
28. Connection Strings for SQL Server 2008 – http://www.connectionstrings.com/sql-server-2008
29. System.Data.SqlClient Namespace – http://msdn.microsoft.com/en-us/library/system.data.sqlclient.aspx