
Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφορική»


**Μεταπτυχιακή Διατριβή**

| | |
|---|---|
| Τίτλος Διατριβής | **Εφαρμογή αυτόματης δημιουργίας αποδόσεων**<br><br>**Application for automatic betting odds creation** |
| Ονοματεπώνυμο Φοιτητή | **Στυλιανός Μούρτζιος** |
| Πατρώνυμο | **Θεόδωρος** |
| Αριθμός Μητρώου | **ΜΠΠΛ/ 13053** |
| Επιβλέπων | **Παναγιωτόπουλος Θέμης** |

Ημερομηνία Παράδοσης **05/2017**

**Τριμελής Εξεταστική Επιτροπή**

| | | |
|---|---|---|
| (υπογραφή) | (υπογραφή) | (υπογραφή) |
| Όνομα Επώνυμο<br>Βαθμίδα | Όνομα Επώνυμο<br>Βαθμίδα | Όνομα Επώνυμο<br>Βαθμίδα |

## ΠΕΡΙΛΗΨΗ

Αντικείμενο της διατριβής αυτής είναι η ανάπτυξη μιας εφαρμογής, η οποία θα έχει τη δυνατότητα να λαμβάνει από το διαδίκτυο δεδομένα που αφορούν στοιχηματικές αποδόσεις. Τα δεδομένα αυτά θα λαμβάνονται μέσα από συγκεκριμένα sites που έχουν σαν αντικείμενο τη διεξαγωγή αντίστοιχων στοιχηματικών παιχνιδιών που αφορούν το ποδόσφαιρο. Επιπρόσθετα, η εφαρμογή που θα αναπτυχθεί θα διαθέτει μηχανισμούς τροποποίησης των παραπάνω αποδόσεων ώστε να τις μεταβάλλει σύμφωνα με συγκεκριμένα κριτήρια και να τις εμφανίζει στο χρήστη. Τέλος, θα υπάρχει λειτουργικότητα για την αποθήκευση ιστορικού τελικών αποτελεσμάτων από ποδοσφαιρικά πρωταθλήματα σε μία βάση δεδομένων. Ζητούμενο είναι η υλοποίηση που θα ακολουθηθεί να καταστήσει την εφαρμογή αυτή portable, ώστε να είναι εύκολη η χρήση της σε οποιοδήποτε μηχάνημα, χωρίς πρόσθετες ή εξειδικευμένες τεχνικές απαιτήσεις.

## ABSTRACT

The object of the current dissertation is the development of an application which will be able to retrieve betting odd data from the internet. These data will be received from specific internet sites that manage relevant betting procedures about soccer. Additionally, the application will implement modification functions on the above odd data so as to recalculate them according to specific criteria and present them to the user. Finally, another mechanism will take place for storing soccer match results in a database. What is mainly requested here is that our application is structured that allows it to be portable in order to be easily usable in any computer device, without the need for additional or special technical requirements.

# Περιεχόμενα

# Εισαγωγή

Η παρούσα εργασία έχει σαν αντικείμενό της τη διερεύνηση των μεθοδολογιών και εργαλείων με τα οποία μπορούμε μέσα από τη γλώσσα προγραμματισμού C# να λάβουμε, να εμφανίσουμε και να τροποποιήσουμε δεδομένα που προέρχονται από σελίδες του διαδικτύου. Στόχος είναι να παρουσιαστούν διαδικασίες οι οποίες μας επιτρέπουν τις λειτουργίες αυτές. Πιο συγκεκριμένα, θα υλοποιηθεί μια εφαρμογή που θα διαχειρίζεται τη λήψη πληροφοριών αλλά και στοιχηματικών δεδομένων για 8 γνωστές ευρωπαϊκές ποδοσφαιρικές διοργανώσεις. Τα δεδομένα των στοιχηματικών αποδόσεων (για τα 1, X και 2) θα λαμβάνονται από 2 sites, στη συνέχεια θα υπολογίζονται οι μέσοι όροι τους και θα εμφανίζονται σε κατάλληλο πίνακα. Τέλος, θα γίνεται επαναυπολογισμός των τιμών των αποδόσεων αυτών χρησιμοποιώντας έναν καθορισμένο συντελεστή.

Πέρα από τη λειτουργία αυτή, η εφαρμογή θα περιέχει μια καρτέλα για τη λήψη αποδόσεων σχετικά με τον πρώτο παίκτη που θα σκοράρει σε μια διοργάνωση που θα επιλέγεται από το χρήστη μέσω κατάλληλα ενημερωμένης λίστας για τους αγώνες που βρίσκονται σε εξέλιξη την τρέχουσα περίοδο. Ο χρήστης θα βλέπει το σύνολο των παικτών που συμμετέχουν στον επιλεγμένο αγώνα μαζί με τις αποδόσεις που έχει ο καθένας για την περίπτωση «Πρώτος σκόρερ» και θα έχει τη δυνατότητα να επιλέξει την εντεκάδα παικτών που επιθυμεί. Κατόπιν, θα εισάγει 2 τιμές που ο ίδιος επιθυμεί και θα υπολογίζει τη συνολική απόδοση που προκύπτει για αυτούς τους 13 παίκτες. Θα μπορεί μετά να τροποποιεί οποιεσδήποτε από αυτές τις 13 τιμές όσο θέλει και να επαναυπολογίζει την αθροιστική απόδοση, μέχρι να φτάσει στην επιθυμητή για αυτόν τιμή.

Τέλος, η εφαρμογή θα ενσωματώνει τη δυνατότητα να αποθηκεύει τα δεδομένα που λαμβάνει για τα τελικά αποτελέσματα βαθμολογιών των πρωταθλημάτων σε μια online βάση δεδομένων. Για τη λειτουργικότητα αυτή θα υπάρχει ξεχωριστή καρτέλα μέσα στην οθόνη της εφαρμογής η οποία θα επιτρέπει στο χρήστη να αναζητά αποθηκευμένα δεδομένα για το πρωτάθλημα της επιλογής του, με δυνατότητα αναζήτησης για συγκεκριμένη εβδομάδα. Τα δεδομένα αυτά θα εμφανίζονται σε κατάλληλο πίνακα.

Η εφαρμογή θα υλοποιηθεί με τη φιλοσοφία να είναι portable. Ο χρήστης θα έχει τη δυνατότητα να τρέχει την εφαρμογή σε οποιοδήποτε μηχάνημα, χωρίς να απαιτείται εγκατάσταση κάποιου περιβάλλοντος ανάπτυξης κώδικα ή δημιουργίας τοπικής βάσης δεδομένων. Η αλληλεπίδραση της εφαρμογής με όλα τα δεδομένα που εμφανίζει ή αποθηκεύει θα γίνεται μέσω του internet, ώστε να είναι πλήρως φορητή και με real-time ενημέρωση.

# Εργαλεία για την ανάπτυξη της εφαρμογής

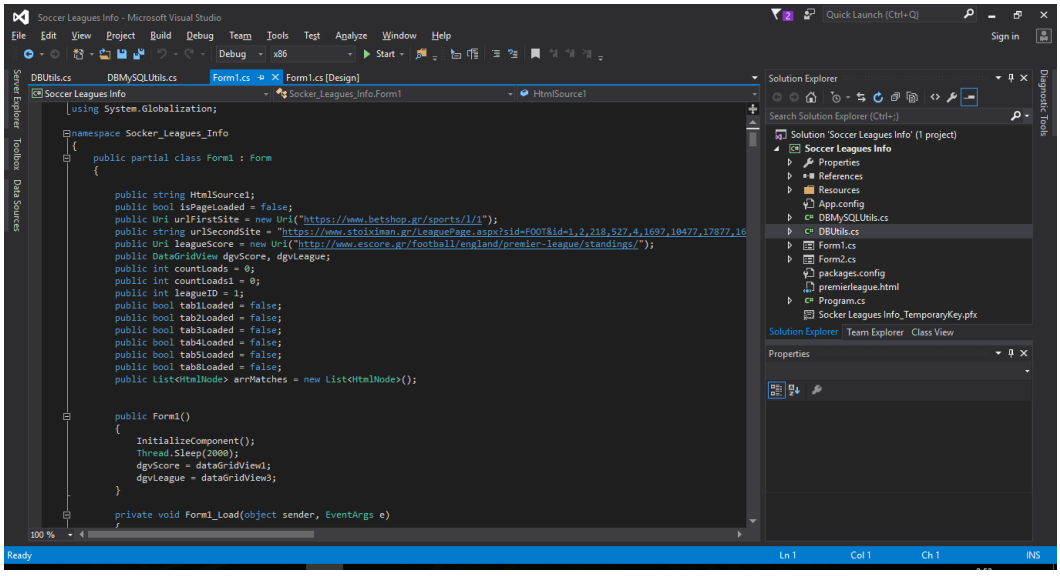
## MICROSOFT VISUAL STUDIO

Το περιβάλλον που χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής που παρουσιάζεται σε αυτή τη διατριβή είναι το Microsoft Visual Studio. Το IDE αυτό χρησιμοποιείται κατεξοχήν για την ανάπτυξη προγραμμάτων για Microsoft Windows OS, καθώς και για web sites & web applications. Χρησιμοποιεί πλατφόρμες ανάπτυξης λογισμικού όπως τα Windows API, Windows Forms και Windows Presentation Foundation.

Το Visual Studio περιλαμβάνει έναν code editor ο οποίος υποστηρίζει intelliSense και code refactoring. Διαθέτει επίσης ενσωματωμένα εργαλεία όπως σχεδιαστή φορμών για GUI applications, web designer & class designer. Υποστηρίζει μια πληθώρα γλωσσών προγραμματισμού με ενδεικτικότερες τις C, C++, C# και VB.NET. Για την εφαρμογή της παρούσας διατριβής χρησιμοποιήθηκε η γλώσσα C#, για την οποία γίνεται εκτενέστερη αναφορά στην επόμενη ενότητα.

Υπάρχουν οι εξής διαθέσιμες εκδόσεις για το Visual Studio:

- Professional, η οποία αποτελεί την entry level εμπορική έκδοση
- Enterprise, η οποία περιλαμβάνει κάποια επιπρόσθετα εργαλεία σε σχέση με την Professional edition που αφορούν software development, database development, collaboration, metrics, architecture, testing & reporting
- Test Professional, που έχει λειτουργία εξειδικευμένη για dedicated testing
- Community, η οποία ξεκίνησε το Νοέμβριο του 2014 και ουσιαστικά αποτελεί μια free version του Visual Studio, παρόμοια σε λειτουργικότητα με την Professional. Σαν θετικό της συγκεκριμένης έκδοσης περιλαμβάνεται η παροχή υποστήριξης για extensions. Η έκδοση αυτή απευθύνεται κατά κύριο λόγο σε μεμονωμένους developers ή μικρές ομάδες ατόμων, γι' αυτό και επιλέχθηκε στα πλαίσια της διατριβής αυτής ανάμεσα στις υπάρχουσες εκδόσεις του Visual Studio.
- Express, η οποία είναι μια απλουστευμένη έκδοση του Visual Studio απευθυνόμενη κυρίως σε μαθητές ή χομπίστες developers




Οθόνη εργασίας του Visual Studio Community Edition 2015

## C# (ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ)

Η C# αποτελεί μια multi-paradigm γλώσσα προγραμματισμού γενικού σκοπού, η οποία εσωκλείει functional, generic, object-oriented & component-oriented προγραμματιστικές αρχές. Ο όρος paradigm αναφέρεται σε κατηγοριοποιήσεις των γλωσσών προγραμματισμού, με βάση το στυλ προγραμματισμού που ακολουθείται. Αναπτύχθηκε από τη Microsoft με αφετηρία την έκδοση 1.0 τον Ιανουάριο του 2002. Η τελευταία διαθέσιμη έκδοση είναι η 6.0 (Ιούνιος 2015).

Για πολλούς, η C# θεωρείται μια γλώσσα πανομοιότυπη με τη Java. Θεωρούνται και οι δύο ως επαναστατικές γλώσσες, οι οποίες άλλαξαν ριζικά τον τρόπο προγραμματισμού και δανείζονται πολλά στοιχεία η μία από την άλλη. Η σύνταξή της επίσης είναι όμοια με αυτή της Java, αλλά και της C και C++. Κάποια ενδεικτικά στοιχεία της σύνταξής της είναι:

- Η χρήστη του συμβόλου ; στο τέλος μιας πρότασης
- Η χρήση των { και } για την ομαδοποίηση statements
- Η ανάθεση τιμών σε μεταβλητές γίνεται με το σύμβολο = και η σύγκριση δύο στοιχείων με τα σύμβολα ==
- Τα σύμβολα [ και ] χρησιμοποιούνται για τον ορισμό ενός πίνακα, καθώς και για την ανάθεση τιμής σε κάποιο στοιχείο του

Κάποια στοιχεία που τη διαφοροποιούν χαρακτηριστικά από τις παραπάνω γλώσσες είναι τα εξής:

- **Φορητότητα:** Λόγω του σχεδιασμού της, η C# αποτελεί αντικατοπτρίζει περισσότερο από κάθε άλλη γλώσσα το **Common Language Infrastructure** (**CLI**) framework, το οποίο περιγράφει πηγαίο κώδικα και runtime περιβάλλον τέτοιο ώστε να επιτρέπει σε πολλαπλές γλώσσες προγραμματισμού υψηλού επιπέδου να μπορούν να χρησιμοποιηθούν από διάφορες πλατφόρμες, χωρίς να υπάρχει ανάγκη για τροποποίηση του κώδικα με βάση συγκεκριμένες αρχιτεκτονικές.
- Δημιουργία **implicit μεταβλητών και πινάκων** με τη χρήση της ειδικής λέξης var.
- Υποστήριξη αυστηρά καθορισμένου **Boolean τύπου δεδομένων,** οι οποίος ορίζεται με την ειδική λέξη bool.
- Όλες οι μέθοδοι πρέπει να ορίζονται μέσα σε **κλάσεις**. Δεν επιτρέπεται η χρήση καθολικών μεταβλητών ή συναρτήσεων, στη θέση τους χρησιμοποιούνται static members των public κλάσεων.
- Οι local μεταβλητές δεν επισκιάζουν τις μεταβλητές ενός block

Program.cs

```
1  using System;
2
3  namespace DotnetBot {
4
5    public static class Program {
6
7      public static void Main(string[] args) {
8
9        string message = "";
10         if (args.Length < 1) {
11           message = "Welcome to .NET Core!";
12         }
13         else {
14           foreach (string item in args) {
15             message += item;
16           }
17         }
```

Παράδειγμα κώδικα σε C#

# MYSQL

Η MySQL αποτελεί ένα open-source σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων (RDBMS). Είναι γραμμένη σε C και C++ και ο parser που χρησιμοποιεί είναι γραμμένος σε yacc αλλά παράλληλα χρησιμοποιεί και έναν custom λεκτικό αναλυτή. Η MySQL δημιουργήθηκε αρχικά από τη σουηδική εταιρεία MySQL AB το 1994 και η πρώτη της έκδοση παρουσιάστηκε το 1995. Το 2008 η εταιρεία εξαγοράστηκε από την Sun Microsystems, η οποία με τη σειρά της εξαγοράστηκε από την Oracle το 2010, οπότε και η MySQL περιήλθε στην ιδιοκτησία της. Η χρήση της είναι ιδιαίτερα δημοφιλής σε ιστοσελίδες και διαδικτυακά προγράμματα. Ενδεικτικά παραδείγματα διάσημων ιστοσελίδων που τη χρησιμοποιούν είναι το Flickr, η Google, το Facebook, το Twitter, το YouTube και η Wikipedia.

Στα πλαίσια της παρούσας διατριβής, έγινε εγκατάσταση μιας MySQL βάσης σε έναν online hosting server, ώστε να ικανοποιηθεί η δυνατότητα για portability της εφαρμογής και να υπάρχει real-time online πρόσβαση στα δεδομένα της βάσης. Η υλοποίηση αυτή επίσης ουσιαστικά αποδεσμεύει τον χρήστη από την ανάγκη για local εγκατάσταση και παραμετροποίηση μιας βάσης δεδομένων κάθε φορά που επιθυμεί να τρέξει την εφαρμογή ακόμα και σε διαφορετικό μηχάνημα.

Τα τεχνικά χαρακτηριστικά της database που χρησιμοποιείται είναι τα εξής:

**host** = "198.46.81.30";
**port** = 3306;
**database** = "csshou5_soccer";
**username** = "csshou5_soccer";
**password** = "QweRTy!2#4%$";

**Database:**
MySQL version: 5.5
Name: csshou5_soccer
Type: InnoDB
Collation: utf8_general_ci

**Tables:**

sli_leagues:
structure:
**id** Primary Key smallint(6) Auto Increment
**name** varchar(255)

sli_score:
structure:
**id** int(11) AUTO_INCREMENT
**lid** smallint(6)
**week** tinyint(3) UNSIGNED
**position** smallint(6)
**name** varchar(255)
**points** smallint(6)

Τα τεχνικά χαρακτηριστικά του database server που χρησιμοποιείται είναι τα εξής:

**CPU** : Xeon E3-1220L V2 Dual Core
**CPU speed** : 2.3 GHz
**RAM** : 128 GB DDR3
**OS** : CentOS 6.7

# Οθόνες της εφαρμογής

Η εφαρμογή αποτελείται από δύο βασικές φόρμες – οθόνες:
1. Την εισαγωγική οθόνη
2. Τη βασική οθόνη

## ΕΙΣΑΓΩΓΙΚΗ ΟΘΟΝΗ

Η εισαγωγική οθόνη της εφαρμογής

Η παραπάνω οθόνη εμφανίζεται όταν ξεκινά η εφαρμογή και παραμένει active για 2 δευτερόλεπτα. Για το design της οθόνης αυτής χρησιμοποιείται

- 1 Windows Form (Splashscreen)
- 1 picturebox όπου εμφανίζουμε την επιλεγμένη εικόνα, η οποία βρίσκεται στο Resources folder του project

Ο κώδικας της σελίδας παρατίθεται ακολούθως:

```
namespace Soccer_Leagues_Info
{
    public partial class Splashscreen : Form
    {
        public Splashscreen()
        {
            InitializeComponent();
        }
```

Αναλυτικότερα, δημιουργείται η public κλάση Splashscreen, η οποία κληρονομεί τις ιδιότητες της base class ***Form*** (`System.Windows.Forms.Form`). Με τη δήλωση ***partial*** δίνεται η δυνατότητα οριστεί η ίδια κλάση μέσα σε δύο διαφορετικά source files που ανήκουν στο ίδιο namespace.

```
public partial class Splashscreen : Form
```

Στη συνέχεια υπάρχει ο constructor της κλάσης, ο οποίος περιέχει τη μέθοδο InitializeComponent. Η μέθοδος αυτή δημιουργείται αυτόματα από τον Windows Forms Designer και διαχειρίζεται όλα τα στοιχεία που φαίνονται μέσα στη φόρμα.

```
public Splashscreen()
        {
            InitializeComponent();
        }
```

Η εισαγωγική οθόνη εμφανίζεται στο χρήστη για 2 δευτερόλεπτα. Ο καθορισμός του χρόνου αυτού καθορίζεται μέσα στον κώδικα της βασικής φόρμας (Form1). Μέσα στον constructor της φόρμας αυτής βρίσκεται η εντολή

```
Thread.Sleep(2000);
```

η οποία σταματάει το εκτελούμενο thread για 2 δευτερόλεπτα, οπότε και εμφανίζεται η Splashscreen. Η διαχείριση της σωστής εμφάνισης των 2 οθονών γίνεται στον κώδικα που υπάρχει στο αρχείο *Program.cs*, όπου καθορίζουμε τον τρόπο με τον οποίο εκτελείται το Thread. Εκεί ορίζουμε μια public static μεταβλητή τύπου Splashscreen

```
public static Soccer_Leagues_Info.Splashscreen splashForm = null;
```

Στη συνέχεια πρέπει να δηλώσουμε στον compiler ότι βρισκόμαστε σε Single Thread Apartment model. Η δήλωση αυτή είναι απαραίτητη όταν υλοποιούμε μια Windows Forms εφαρμογή, καθώς σε περίπτωση που λείπει η εφαρμογή αυτομάτως θα χρησιμοποιήσει το multithreaded apartment model, το οποίο δεν υποστηρίζεται για Windows Forms.

```
[STAThread]
```

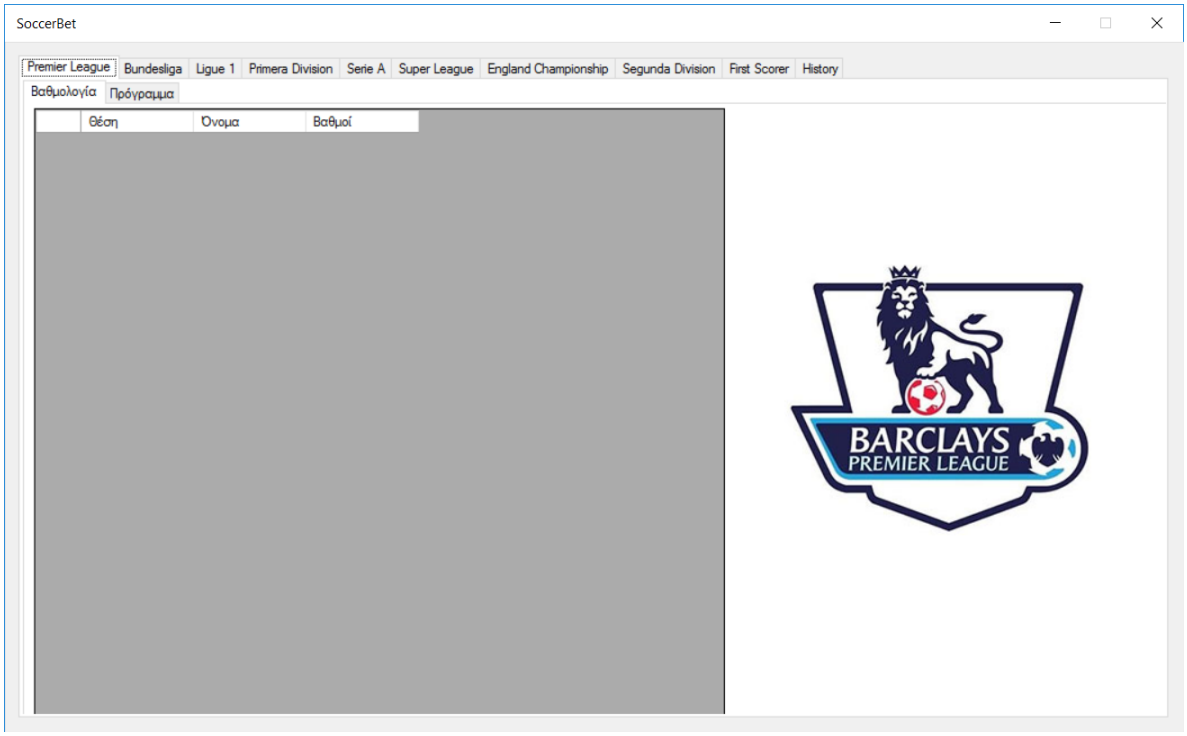
Στη συνέχεια δημιουργούμε το Thread που ρυθμίζει τον τρόπο εμφάνισης των οθονών:

```
            Thread splashThread = new Thread(new ThreadStart(
                        delegate
                        {
                            splashForm = new Soccer_Leagues_Info.Splashscreen();
                            Application.Run(splashForm);
                        }
                        ));

            splashThread.SetApartmentState(ApartmentState.STA);
            splashThread.Start();

            Form1 mainForm = new Form1();
            mainForm.Load += new EventHandler(mainForm_Load);
            Application.Run(mainForm);
```

Ακολούθως παρουσιάζεται η μέθοδος `mainForm_Load`, η οποία κλείνει την Splashscreen ώστε να γίνει το φόρτωμα της βασικής οθόνης (Form1).

```
static void mainForm_Load(object sender, EventArgs e)
        {
            //close splash
            if (splashForm == null)
            {
                return;
            }

            splashForm.Invoke(new Action(splashForm.Close));
            splashForm.Dispose();
            splashForm = null;
        }
```

## ΒΑΣΙΚΗ ΟΘΟΝΗ




Η βασική οθόνη της εφαρμογής

Στη βασική οθόνη υλοποιούνται όλες οι λειτουργικότητες της εφαρμογής. Τα πρώτα 8 tabs αφορούν τις διοργανώσεις για τις οποίες λαμβάνονται δεδομένα. Συγκεκριμένα, οι διοργανώσεις αυτές είναι οι εξής:

1. Premier League (Αγγλία)
2. Bundesliga (Γερμανία)
3. Ligue 1 (Γαλλία)
4. Primera Division (Ισπανία)
5. Serie A (Ιταλία)
6. Super League (Ελλάδα)
7. England Championship (Αγγλία)
8. Segunda Division (LaLiga2 Ισπανίας)

Για κάθε επιλεγμένη διοργάνωση, υπάρχουν 2 tabs που εμφανίζουν

1. τον πίνακα με τις βαθμολογίες των ομάδων
2. πίνακα με το πρόγραμμα αγώνων, τις αποδόσεις που λαμβάνονται από 2 στοιχηματικά sites για κάθε αγώνα, τους υπολογισμούς των μέσων όρων από τα sites και τον επαναυπολογισμό των μέσων όρων με συντελεστή 116%.

Το τελευταίο tab αφορά την υλοποίηση για τον 1ο παίκτη που θα σκοράρει. Στην επόμενη ενότητα γίνεται αναλυτική περιγραφή για την υλοποίηση των λειτουργιών της βασικής οθόνης.

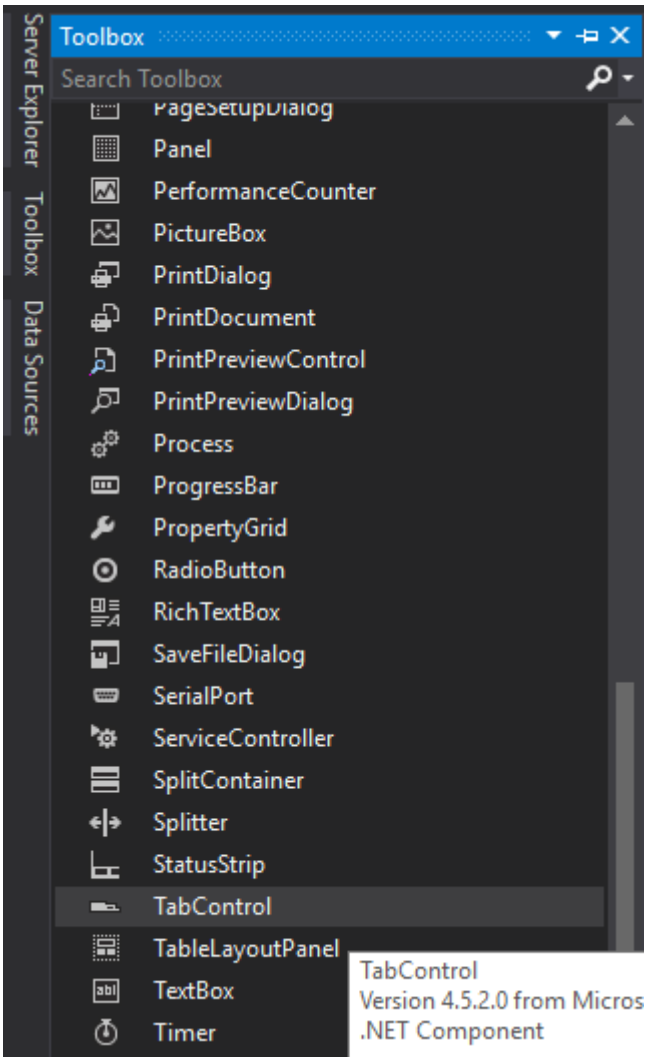
# Υλοποίηση λειτουργιών της βασικής οθόνης

## ΣΧΕΔΙΑΣΗ ΒΑΣΙΚΗΣ ΟΘΟΝΗΣ

Για τη σχεδίαση της βασικής οθόνης υλοποιήθηκε η Form1.cs η οποία περιλαμβάνει τα εξής σχεδιαστικά components:

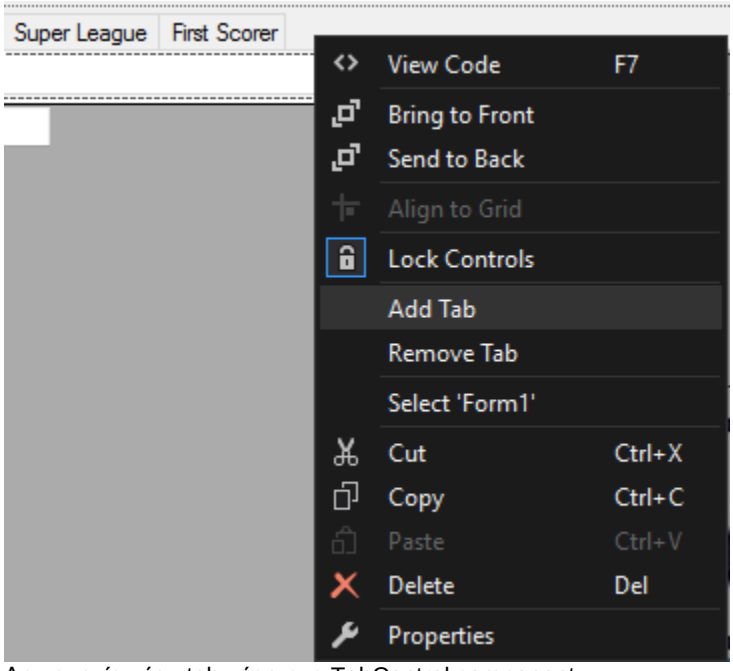
- Την ***Form1***, που είναι ουσιαστικά μία Windows Form και αποτελεί το βασικό στοιχείο πάνω στο οποίο δομούνται όλα τα υπόλοιπα στοιχεία εμφάνισης
- Το ***TabControl1***, το οποίο είναι ένα ***TabControl*** και χρησιμοποιείται για τη διαχείριση των tabs 1ου επιπέδου. Στο ***TabControl1*** βρίσκονται:
    - Τα πρωταθλήματα για τα οποία εμφανίζονται τα στοιχηματικά δεδομένα
    - Η καρτέλα για τη λειτουργικότητα «Πρώτος παίκτης που θα σκοράρει»

Για να εισάγουμε ένα ***TabControl*** στη φόρμα μας, πηγαίνουμε στο Toolbox που βρίσκεται στο αριστερό μενού του Visual Studio και το εντοπίζουμε μέσα στην ενότητα “All Windows Forms”. Κατόπιν το τοποθετούμε στη φόρμα με drag n drop.



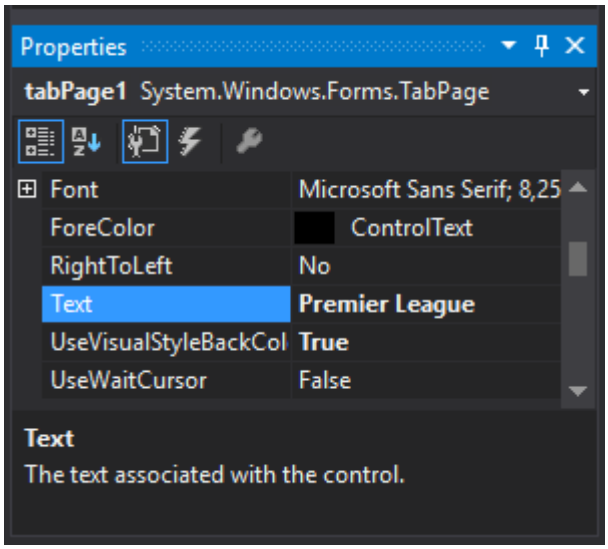

Η θέση του TabControl μέσα στο Toolbox του Visual Studio

Κατόπιν θα χρειαστεί να προσθέσουμε καρτέλες (Tabs) για όλα τα πρωταθλήματα, αλλά και για τη σελίδα «Πρώτος παίκτης». Για να το κάνουμε αυτό πάμε στο ***TabControl1***, κάνουμε δεξί κλικ και επιλέγουμε *Add Tab*.



Δημιουργία νέου tab μέσα στο TabControl component

Κάθε νέο tab που προστίθεται δημιουργεί αυτόματα από κάτω του το χώρο της σελίδας που του αντιστοιχεί, το οποίο ονομάζεται ***tabPage***. Για να μπορέσουμε να τοποθετήσουμε τους τίτλους που επιθυμούμε σε κάθε tab που φτιάχνουμε λοιπόν, θα πρέπει να κάνουμε δεξί κλικ στο tabPage (σε οποιοδήποτε σημείο του) που μας ενδιαφέρει και να επιλέξουμε Properties. Ακολούθως θα δούμε στο κάτω μέρος της δεξιάς στήλης του project την ενότητα Properties. Ο τίτλος του tab ορίζεται μέσα στο πεδίο ***Text***. Εκεί εισάγουμε το επιθυμητό λεκτικό.



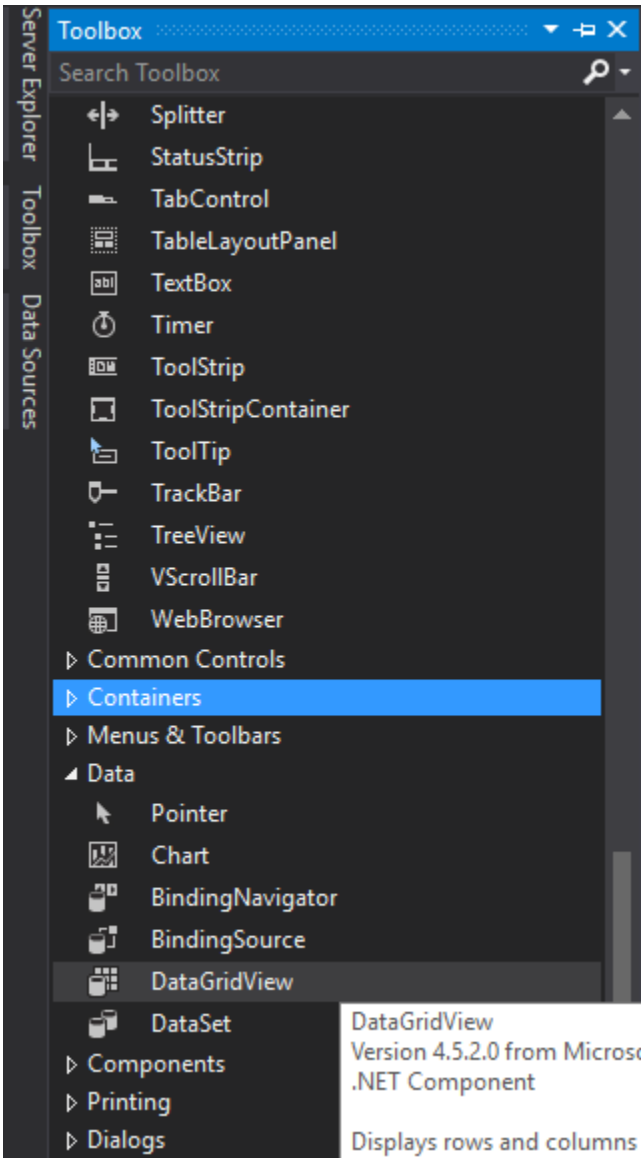
Εισαγωγή τίτλου για ένα Tab μέσα στο Properties

Κάτω από κάθε πρωτάθλημα στο ***TabControl1***, βρίσκονται τα εξής στοιχεία:

- Τα ***TabControl2*** έως ***TabControl7***, τα οποία χρησιμοποιούνται για τη διαχείριση της εμφάνισης των κατάλληλων καρτελών για τις λειτουργικότητες «Βαθμολογία» και «Πρόγραμμα»

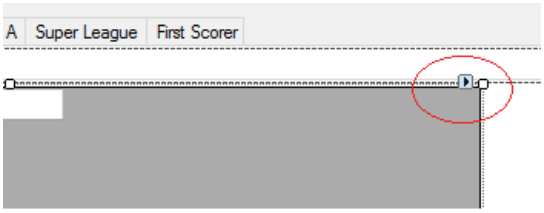
- **DataGridView** όπου γίνεται η εμφάνιση όλων των δεδομένων για βαθμολογίες και αποδόσεις

Για να εισάγουμε ένα ***DataGridView*** στη φόρμα μας, πηγαίνουμε στο Toolbox που βρίσκεται στο αριστερό μενού του Visual Studio και το εντοπίζουμε μέσα στην ενότητα “Data”. Κατόπιν το τοποθετούμε στη φόρμα με drag n drop.

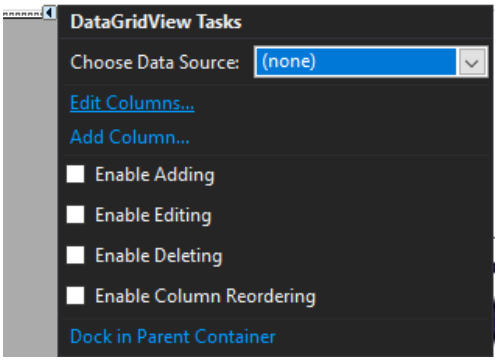


Η θέση του DataGridView μέσα στο Toolbox του Visual Studio

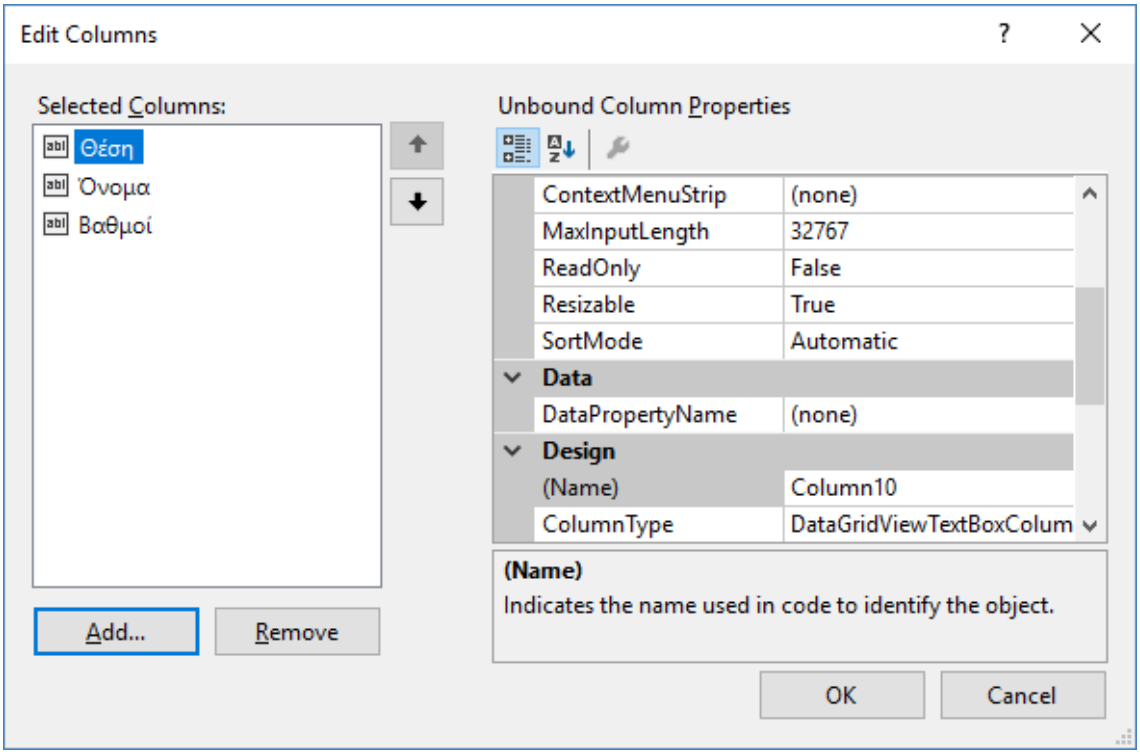
Ένα ***DataGridView*** αποτελεί ουσιαστικά έναν πίνακα δεδομένων. Για να ορίσουμε κατάλληλους τίτλους στις στήλες ενός ***DataGridView*** θα πρέπει εφόσον το έχουμε επιλέξει με το ποντίκι να πατήσουμε το εικονίδιο που εμφανίζεται πάνω δεξιά στην περιοχή του ***DataGridView***




Στο menu που εμφανίζεται επιλέγουμε ***Edit Columns***

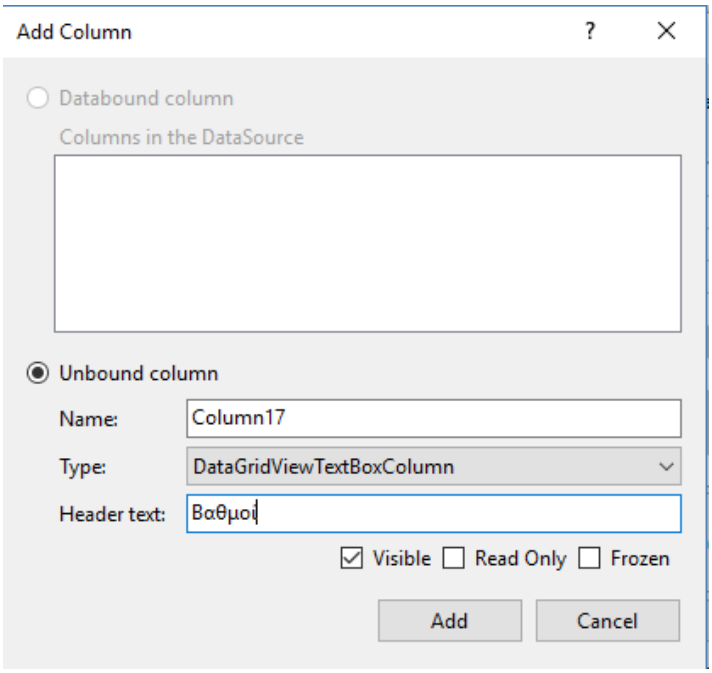



Στη συνέχεια εμφανίζεται η οθόνη επεξεργασίας για τις στήλες του DataGridView




Οθόνη επεξεργασίας στηλών του DataGridView

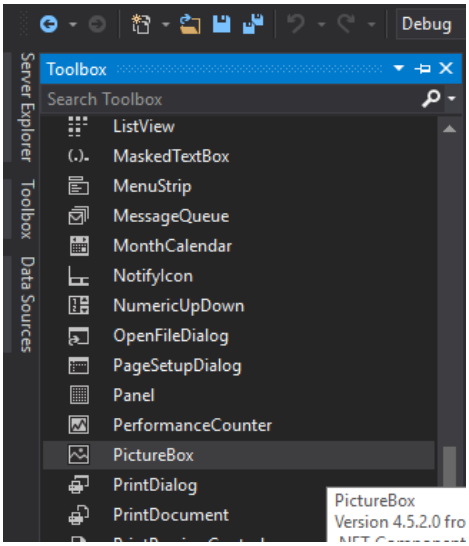
Για να προσθέσουμε μια στήλη, επιλέγουμε το κουμπί Add. Στην οθόνη παραμετροποίησης του νέου Column εισάγουμε τον επιθυμητό τίτλο στο πεδίο Header Text και πατάμε Add.




Οθόνη παραμετροποίησης του νέου Column

- Ένα ***PictureBox*** όπου εισάγουμε την εικόνα με το logo που αντιστοιχεί σε κάθε πρωτάθλημα

Για να εισάγουμε ένα ***PictureBox*** στη φόρμα μας, πηγαίνουμε στο Toolbox που βρίσκεται στο αριστερό μενού του Visual Studio και το εντοπίζουμε μέσα στην ενότητα “All Windows Forms”. Κατόπιν το τοποθετούμε στη φόρμα με drag n drop.

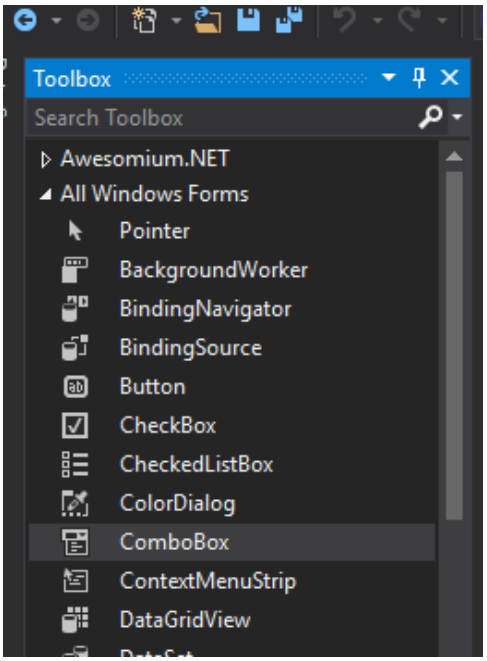



Η θέση του PictureBox μέσα στο Toolbox του Visual Studio

Στην καρτέλα «Πρώτος παίκτης που θα σκοράρει» χρησιμοποιούνται τα παρακάτω επιπλέον στοιχεία σχεδιασμού:

1. Δύο ***ComboBox***, όπου θα περιέχονται λίστες με τιμές για να επιλέξει ο χρήστης.

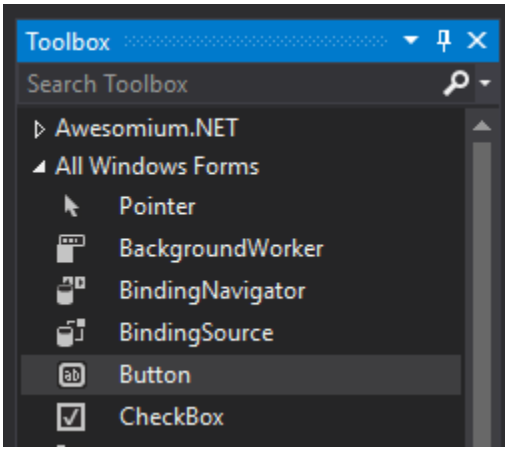
   Για να εισάγουμε ένα ***ComboBox*** στη φόρμα μας, πηγαίνουμε στο Toolbox που βρίσκεται στο αριστερό μενού του Visual Studio και το εντοπίζουμε μέσα στην ενότητα “All Windows Forms”. Κατόπιν το τοποθετούμε στη φόρμα με drag n drop.



Η θέση του ComboBox μέσα στο Toolbox του Visual Studio

2. Ένα ***button*** το οποίο θα χρησιμοποιείται για μεταφορά δεδομένων από το ένα DataGridView στο άλλο.

   Για να εισάγουμε ένα ***button*** στη φόρμα μας, πηγαίνουμε στο Toolbox που βρίσκεται στο αριστερό μενού του Visual Studio και το εντοπίζουμε μέσα στην ενότητα “All Windows Forms”. Κατόπιν το τοποθετούμε στη φόρμα με drag n drop.



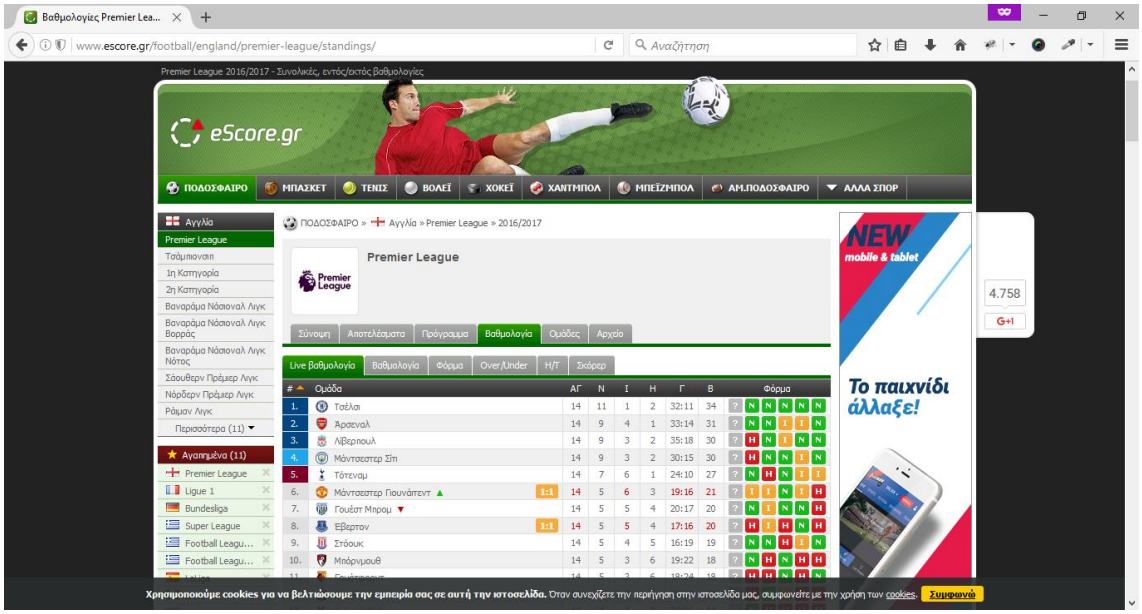
Η θέση του Button μέσα στο Toolbox του Visual Studio

## ΒΑΘΜΟΛΟΓΙΑ ΑΓΩΝΙΣΤΙΚΗΣ

Η βαθμολογία μιας αγωνιστικής εμφανίζεται στο πρώτο tab που περιέχεται σε κάθε κατηγορία πρωταθλήματος. Για την εμφάνιση των δεδομένων, χρησιμοποιείται ένα DaraGridView. Αρχικά, διαμορφώνουμε τις στήλες του DataGridView ώστε να δώσουμε κατάλληλους τίτλους στις στήλες του. Τα στοιχεία που θέλουμε να εμφανίσουμε εδώ είναι:

- Ομάδα
- Αγωνιστική (ΑΓ)
- Βαθμοί (Β)

Το site που θα χρησιμοποιήσουμε για να πάρουμε τα δεδομένα των βαθμολογιών για όλα τα πρωταθλήματα είναι το www.escore.gr .



Οθόνη του site για τη βαθμολογία της Premier League

Εφόσον η βαθμολογία της Premier League είναι το πρώτο στοιχείο της βασικής οθόνης που εμφανίζεται κατά την εκτέλεση της εφαρμογής, το παραπάνω url είναι ορισμένο μέσα στον κώδικα ως το πρώτο site που θα γίνει parse, ώστε να γίνει κατευθείαν η φόρτωση των δεδομένων στο φόρτωμα της βασικής οθόνης. Για όλα τα άλλα πρωταθλήματα, τα δεδομένα φορτώνονται εφόσον επιλεχθεί το αντίστοιχο tab.

```
public Uri leagueScore = new Uri("http://www.escore.gr/football/england/premier-
league/standings/");
```

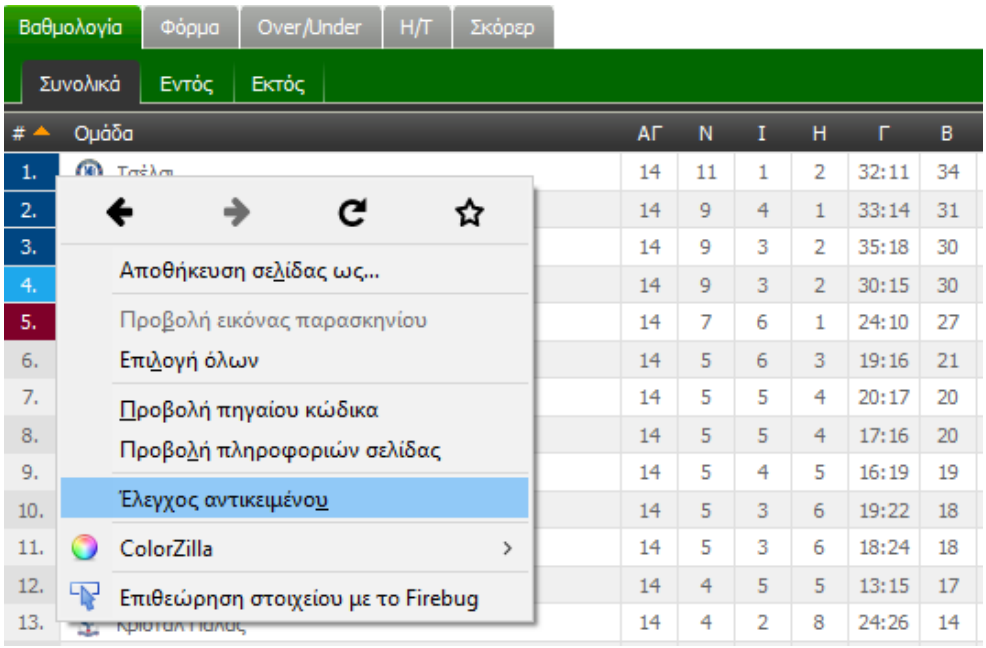
Το ***DataGridView1*** θα περιέχει τα δεδομένα της βαθμολογίας για την Premier League. Μέσα στον κώδικα γίνεται αρχικά ορισμός μιας public μεταβλητής τύπου DataGridView με όνομα *dgvScore*, η οποία θα διαχειρίζεται την εμφάνιση της βαθμολογίας για κάθε πρωτάθλημα.

```
public DataGridView dgvScore
```

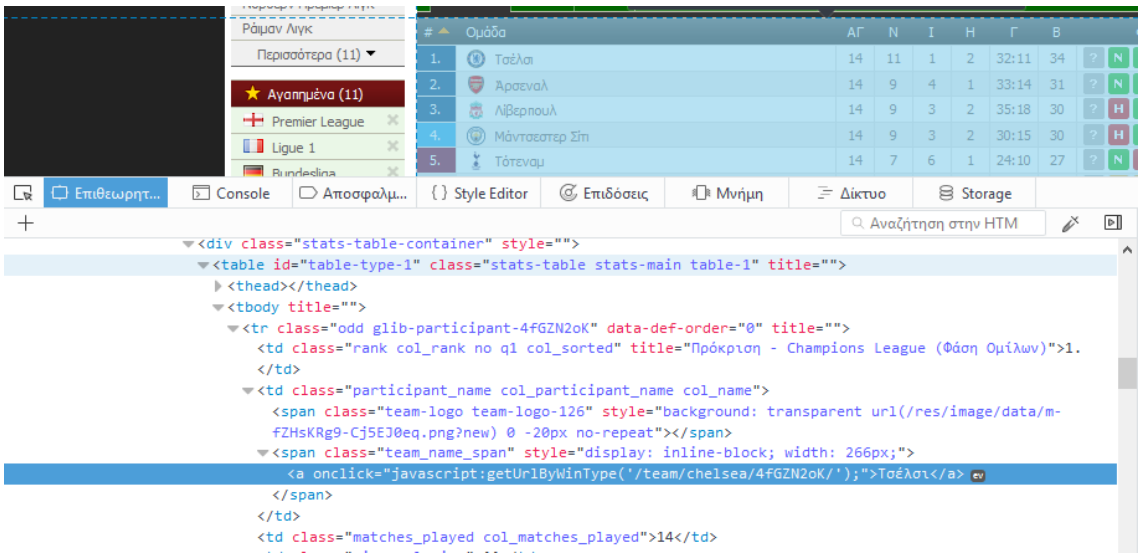
Στη συνέχεια, εφόσον τα στοιχεία της Premier League πρέπει να εμφανιστούν πρώτα κατά την εκτέλεση, αντιστοιχίζουμε τη μεταβλητή αυτή με το ***DataGridView1*** μέσα στον constructor της φόρμας της βασικής οθόνης.

```
dgvScore = dataGridView1;
```

Τα δεδομένα που μας ενδιαφέρουν και χρειάζεται να εισαχθούν στο DataGridView της αντίστοιχης σελίδας αφορούν τις στήλες 2 (Ομάδα), 3 (Αγωνιστική) και 8 (Βαθμολογία) του πίνακα που απεικονίζεται παραπάνω. Για να μπορέσουμε να τα πάρουμε από το site, θα πρέπει να διερευνήσουμε τη δομή της σελίδας αυτής και κατ' επέκταση του πίνακα. Για να γίνει αυτό, πηγαίνουμε με το ποντίκι πάνω από το δεδομένο που μας ενδιαφέρει και αφού κάνουμε δεξί κλικ επιλέγουμε «Έλεγχος αντικειμένου». Για παράδειγμα, ας υποθέσουμε ότι κάνουμε έλεγχο για το στοιχείο με το όνομα της ομάδας στη 2η στήλη.

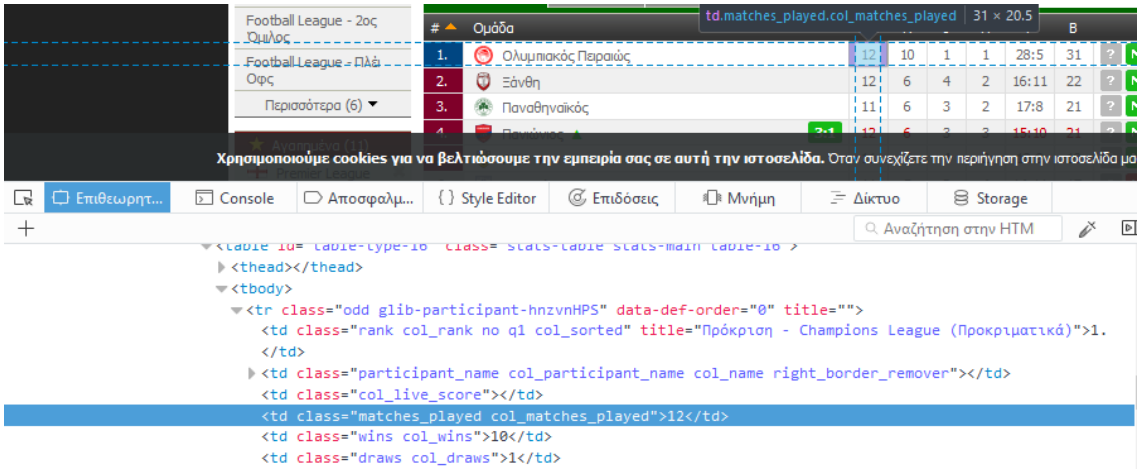



Στη συνέχεια ο browser μας εμφανίζει τη θέση του στοιχείου μέσα στη σελίδα. Στην περίπτωσή μας έχουμε έναν HTML πίνακα ο οποίος περιέχει την κλάση stats-main. Τα δεδομένα του πίνακα βρίσκονται μέσα στο κυρίως σώμα του (tbody), το οποίο περιέχει τις σειρές του (tr).




Η δομή του πίνακα όπως εμφανίζεται κατά τον έλεγχο αντικειμένου μέσω του browser. Εδώ φαίνεται η θέση που βρίσκεται το όνομα της ομάδας.

Το όνομα της ομάδας βρίσκεται μέσα στο κελί του πίνακα (td) το οποίο έχει σαν class name το **participant_name**. Η τιμή για το όνομα συγκεκριμένα φαίνεται αν πάμε στο span που περιέχεται στο παραπάνω td, το οποίο έχει class name **team_name_span**. Αντίστοιχα, η τιμή για την αγωνιστική βρίσκεται στο td με class name **matches_played**.




Η βαθμολογία βρίσκεται στο td με class name **goals**. Εδώ παρατηρούμε κάτι που θέλει ιδιαίτερη προσοχή. Η κλάση **goals** περιέχει 2 στοιχεία και η τιμή που μας ενδιαφέρει αφορά το 2ο στοιχείο της, οπότε θα πρέπει να προβλέψουμε ώστε να πάρουμε την τιμή του στοιχείο αυτού.


Για να εμφανίσουμε τα παραπάνω δεδομένα στην εφαρμογή μας, θα πρέπει να πραγματοποιήσουμε screen scraping. Η ορολογία αυτή αναφέρεται στη διαδικασία όπου κάνουμε download μία ιστοσελίδα και στη συνέχεια κάνουμε parsing στο html της σελίδας ώστε να τραβήξουμε πληροφορίες από αυτό. Στην C# έχει δημιουργηθεί μια ειδική βιβλιοθήκη για την εκτέλεση τέτοιων λειτουργιών, η οποία ονομάζεται HtmlAgilityPack. Η βιβλιοθήκη αυτή ενσωματώνει ειδικές κλάσεις για το κατέβασμα και τη διερεύνηση html σελίδων.

Στον κώδικα της εφαρμογής μας η μέθοδος που έχουμε υλοποιήση για το parsing των δεδομένων της βαθμολογίας είναι η getLeagueScore, η οποία παίρνει σαν παράμετρο το url του site που περιέχει τα δεδομένα αυτά.

```
private void getLeagueScore(string uriHtml)
```

Η βασική κλάση που περιέχεται στην HtmlAgilityPack είναι η *HtmlDocument*. Η κλάση αυτή αποθηκεύει ένα πλήρες Html έγγραφο, οπότε θα δημιουργήσουμε ένα νέο αντικείμενο της κλάσης αυτής το οποίο θα χρησιμοποιούμε κατά τη διαδικασία του parsing.

```
HtmlAgilityPack.HtmlDocument doc = new HtmlAgilityPack.HtmlDocument();
```

Στη συνέχεια είναι απαραίτητο να χρησιμοποιήσουμε την παρακάτω μέθοδο ώστε να φορτώσουμε στο αντικείμενο που δημιουργήσαμε τον HTML κώδικα από το site που θέλουμε. Το url του site είναι η παράμετρος κλήσης της getLeagueScore.

```
doc.LoadHtml(uriHtml);
```

Για τη διαδικασία του parsing των δεδομένων θα χρησιμοποιήσουμε nodes μέσα στο HtmlDocument που ορίσαμε. Κάθε node ουσιαστικά αποτελεί ένα row του html table και θα χρησιμοποιείται για την προσπέλασή του, ώστε να λαμβάνουμε τα επιθυμητά δεδομένα. Πρώτο βήμα είναι να καθορίσουμε την αρχική θέση που βρίσκονται οι κόμβοι, η οποία είναι το η πρώτη σειρά μέσα στο body του table. Ορίζουμε λοιπόν την παρακάτω μεταβλητή για το σκοπό αυτό

```
var nodes = doc.DocumentNode.SelectNodes("//table[contains(@class,'stats-
main')]/tbody/tr");
```

Στη μεταβλητή αυτή πλέον έχουμε αποθηκεύσει όλο το body του πίνακα με τις σειρές του (tr). Για κάθε μεμονωμένο node μέσα στο nodes τώρα θα πρέπει να αποθηκεύουμε τις τιμές για το όνομα και τη βαθμολογία της ομάδας, τα οποία όπως περιγράφηκε παραπάνω βρίσκονται στις θέσεις που μας υπέδειξε ο έλεγχός τους μέσω του browser. Ξεκινάμε λοιπόν τη διαδικασία ελέγχοντας αρχικά αν ο html πίνακας περιέχει δεδομένα (δεν είναι null). Ύστερα, ορίζουμε έναν counter i ο οποίος θα χρησιμοποιηθεί για την αρίθμηση στην πρώτη στήλη του DataGridView (θέση) και θα αυξάνεται κατά 1 κάθε φορά που πάμε σε νέο node (νέα γραμμή). Επιπρόσθετα, μέσα στο nodes ψάχνουμε μέσω ενός νέου node ανά γραμμή ώστε να κάνουμε parse τις τιμές που θέλουμε και να τις αποθηκεύσουμε σε μεταβλητές.

Άλλη μια μεταβλητή που χρησιμοποιείται είναι το roundIndex, για να καθορίσει σε ποια αγωνιστική είμαστε. Επειδή, μπορεί κάποιες ομάδες της εκάστοτε διοργάνωσης να έχουν συμμετάσχει σε λιγότερες αγωνιστικές κατά την τρέχουσα εβδομάδα, το roundIndex συγκρίνεται με την αγωνιστική στην οποία βρίσκεται κάθε ομάδα, και αν ο αριθμός currRound είναι μεγαλύτερος από τον roundIndex, γίνεται αυτός ο νέος roundIndex. Κατα αυτόν τον τρόπο στο τέλος το roundIndex θα περιέχει το μεγαλύτερο αριθμό αγωνιστικής για την τρέχουσα διοργάνωση.

```
if (nodes != null)
{
	int i = 0;
	int roundIndex = 0;
	foreach (var node in nodes)
	{
		var teamName =
node.SelectSingleNode(".//td[contains(@class,'participant_name')]/span[@class='te
am_name_span']");
		var teamPoints =
node.SelectSingleNode(".//td[contains(@class,'goals')][2]");
```

Η θέση που ορίσαμε για την τιμή της teamPoints περιέχει το προσδιοριστικό στοιχείο [2]. Αυτό δικαιολογείται καθώς όπως προαναφέρθηκε θέλουμε να πάρουμε το 2ο στοιχείο που ανήκει στην κλάση goals.

Τέλος, για να εισάγουμε τις τιμές αυτές στο DataGridView της σελίδας, χρησιμοποιούμε τη μέθοδο `dgvScore.Rows.Add` όπως φαίνεται ακολούθως.

```
if (this.dgvScore.InvokeRequired)
{
	Invoke((MethodInvoker)(() => dgvScore.Rows.Add(i+1,teamName.InnerText,
teamPoints.InnerText) ));
}
else
{
	dgvScore.Rows.Add(i+1, teamName.InnerText, teamPoints.InnerText);
```

Υπάρχει περίπτωση την ώρα που γίνεται προσθήκη τιμών στο dgvScore με τη μέθοδο Add, να κληθεί από κάποιο άλλο thread το dvgScore και όχι από το εκείνο που το είχε καλέσει αρχικά. Αυτό μπορεί να οδηγήσει το DataGridView σε μία ασταθή κατάσταση, κάτι που το Visual Studio προβλέπει και δημιουργεί το error “InvalidOperationException”. Αυτό βέβαια είναι εμφανές μόνο κατά το debugging ενώ σε runtime μπορεί απλά να “κολλήσει” την εφαρμογή. Για να αποτραπεί λοιπόν αυτό το error κανουμε ένα thread-safe call που εξασφαλίζει ότι δε θα προκληθεί αστάθεια. Αυτό είναι εφικτό με τη χρήση του Invoke. Έτσι εάν βρισκόμαστε σε διαφορετικό thread από το αρχικό, αρα το InvokeRequired = true, κάνουμε add το control στο thread και μετά τρέχουμε το Add.

## ΠΡΟΓΡΑΜΜΑ ΑΓΩΝΙΣΤΙΚΗΣ

Στην ενότητα αυτή παρουσιάζεται ένας πίνακας ο οποίος περιέχει το πρόγραμμα της αγωνιστικής, τις στοιχηματικές αποδόσεις (1 X 2) που περιέχονται σε δύο στοιχηματικά sites, τους μέσους όρους των αποδόσεων από τα sites αυτά και τέλος τις τροποποιημένες αποδόσεις που δημιουργούνται αυτόματα σύμφωνα με ένα κριτήριο που έχει οριστεί στον κώδικα. Οι τροποποιημένες αποδόσεις προκύπτουν με την εξής διαδικασία:

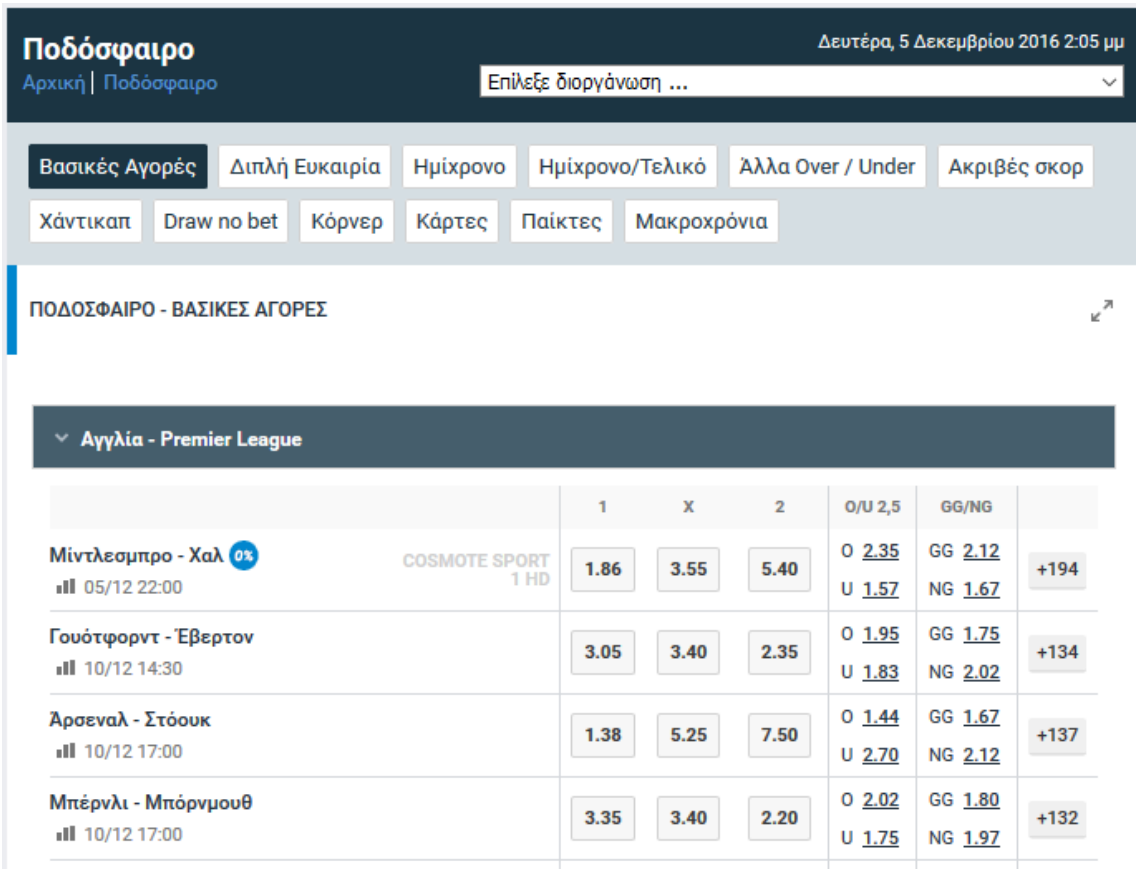
> Το άθροισμα των μέσων όρων των αποδόσεων που προκύπτει από τα 2 sites όταν διαιρεθεί με το 100 μας δίνει μία τιμή που αποτελεί έναν συντελεστή ποσοστού αποδόσεων, το οποίο κυμαίνεται από 80% έως 100%
>
> Για κάθε στοιχηματική απόδοση θέλουμε να επαναυπολογίσουμε την τιμή του μέσου όρου της με συντελεστή ποσοστού το 116%

Αρχικά, στην κατάλληλη καρτέλα θα πρέπει να δημιουργήσουμε ένα DataGridView όπου θα εισάγονται τα δεδομένα που θα εμφανίζει η εφαρμογή. Το τροποποιούμε κατάλληλα (σύμφωνα με τη διαδικασία που αναλύθηκε στην προηγούμενη ενότητα) ώστε να διαμορφώσουμε τις εξής στήλες προς εμφάνιση:

- **Ημερομηνία-Ώρα**
- **Γηπεδούχοι**
- **Φιλοξενούμενοι**
- **1** (απόδοση από betshop.gr)
- **X** (απόδοση από betshop.gr)
- **2** (απόδοση από betshop.gr)
- **1** (απόδοση από stoiximan.gr)
- **X** (απόδοση από stoiximan.gr)
- **2** (απόδοση από stoiximan.gr)
- **Μ.Ο. : 1** (μέσος όρος αποδόσεων των δύο sites για το 1)
- **Μ.Ο. : X** (μέσος όρος αποδόσεων των δύο sites για το X)
- **Μ.Ο. : 2** (μέσος όρος αποδόσεων των δύο sites για το 2)
- **Μ.Ο.1 With Factor** (επαναυπολογισμός 1 με συντελεστή 116%)
- **Μ.Ο.X With Factor** (επαναυπολογισμός X με συντελεστή 116%)
- **Μ.Ο.2 With Factor** (επαναυπολογισμός 2 με συντελεστή 116%)

Για τα δεδομένα των αποδόσεων 1 X και 2 χρησιμοποιούμε τα sites https://www.betshop.gr/ και https://www.stoiximan.gr/ . Το betshop χρησιμοποιείται επίσης για τη λήψη των τιμών στις 3 πρώτες στήλες που αναφέρονται παραπάνω. Όπως και στην προηγούμενη ενότητα, εφόσον η Premier League είναι το πρώτο πρωτάθλημα που εμφανίζεται κατά το τρέξιμο της εφαρμογής, τα url που θα χρησιμοποιηθούν αρχικά για τα παραπάνω δεδομένα ορίζονται στην αρχή του προγράμματος ως εξής:

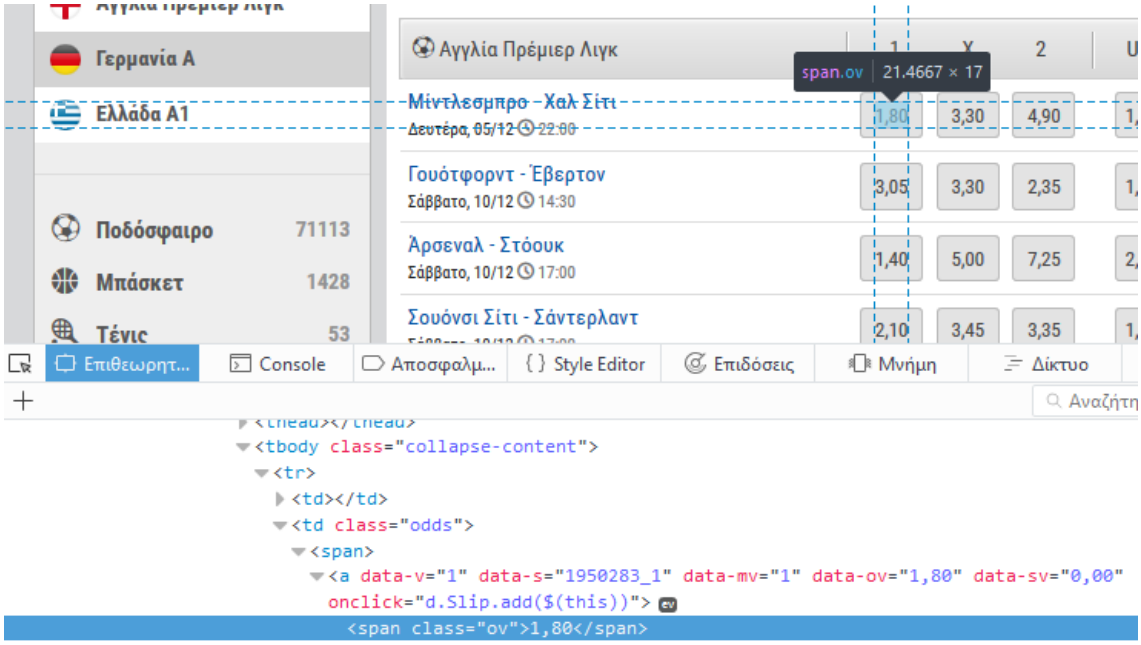
```
public Uri urlFirstSite = new Uri("https://www.betshop.gr/sports/l/1");

public string urlSecondSite =
"https://www.stoiximan.gr/LeaguePage.aspx?sid=FOOT&id=1,2,218,527,4,1697,10477,17
877,1698,17891,18092,18443";
```




Σελίδα του stoiximan.gr για τις αποδόσεις της Premier League


Σελίδα του betshop.gr για τις αποδόσεις της Premier League

Πρώτο βήμα είναι να εντοπίσουμε τη θέση των δεδομένων που μας ενδιαφέρουν μέσα στο html της σελίδας. Αυτό θα γίνει με την ίδια ακριβώς διαδικασία που περιγράφηκε στην προηγούμενη ενότητα, ελέγχοντας δηλαδή το κάθε αντικείμενο μέσω του browser.




Έλεγχος αντικειμένου για την τιμή που επιθυμούμε ώστε να βρεθεί η θέση της μέσα στο html της σελίδας

Στην εφαρμογή μας έχει υλοποιηθεί η μέθοδος ***checkDatagridHtml*** για τη διαχείριση του parsing των δεδομένων αυτών, καθώς και της προσθήκης τους στο DataGridView της εκάστοτε σελίδας. Επίσης, έχει οριστεί μια public μεταβλητή τύπου DataGridView με όνομα *dgvLeague*, η οποία θα διαχειρίζεται την εμφάνιση της βαθμολογίας για κάθε πρωτάθλημα.

```
public DataGridView dgvLeague;
```

Το πρώτο βήμα είναι να κάνουμε parse τα δεδομένα από το stoiximan.gr . Τα δεδομένα αυτά δεν θα εισαχθούν απευθείας στο DataGridView, αλλά θα χρησιμοποιήσουμε έναν ενδιάμεσο πίνακα για την προσωρινή αποθήκευσή τους. Αρχικά μέσα στην κλάση Form1 ορίζονται οι μεταβλητές που περιέχουν τα url των 2 παραπάνω στοιχηματικών site που θα χρησιμοποιήσουμε για το parsing των data.

```
public Uri urlFirstSite = new Uri("https://www.betshop.gr/sports/l/1");

public string urlSecondSite =
"https://www.stoiximan.gr/LeaguePage.aspx?sid=FOOT&id=1,2,218,527,4,1697,10477,17
877,1698,17891,18092,18443";
```

Κατόπιν θα χρησιμοποιήσουμε μια local μεταβλητή μέσα στην ***checkDatagridHtml*** όπου θα αντιστοιχίζεται το url του εκάστοτε πρωταθλήματος για τα δεδομένα που το αφορούν μέσα στο stoiximan.

```
string urlBet =  urlSecondSite;
```

Για το χειρισμό των html data της παραπάνω σελίδας θα χρησιμοποιήσουμε πάλι την HtmlAgilityPack βιβλιοθήκη, με τη διαφοροποίηση ότι θα χρησιμοποιήσουμε και ένα αντικείμενο τύπου HtmlWeb. Το HtmlWeb είναι μία κλάση η οποία παίρνει το html μιας σελίδας μέσω HTTP. Χρησιμοποιώντας τη μέθοδο Load της κλάσης, μπορούμε να αναλύσουμε (parse) το DOM Μιας σελίδας και να το φορτώσουμε σε ένα HtmlDocument του HtmlAgilityPack.

```
HtmlWeb HtmlWEB1 = new HtmlWeb();
HtmlAgilityPack.HtmlDocument doc1 = HtmlWEB1.Load(urlBet);
doc1 = HtmlWEB1.Load(urlBet);
```

Ορίζουμε ένα νέο αντικείμενο τύπου DataTable, το οποίο θα αποτελεί τον ενδιάμεσο πίνακα για την προσωρινή αποθήκευση των δεδομένων που θα ληφθούν από το Stoiximan. Ο πίνακας αυτός θα αποθηκεύσει τα rows του πίνακα που εμφανίζεται στο stoiximan ακριβώς με τη δομή που έχουν μέσα στη σελίδα, οπότε τα πεδία που δημιουργούμε για τον **betTable1** βρίσκονται σε πλήρη αντιστοιχία με αυτά του site.

```
DataTable betTable1 = new DataTable();

betTable1.Columns.Add("Ομάδες", typeof(String));
betTable1.Columns.Add("1", typeof(String));
betTable1.Columns.Add("X", typeof(String));
betTable1.Columns.Add("2", typeof(String));
betTable1.Columns.Add("O/U 2,5", typeof(String));
betTable1.Columns.Add("GG/NG", typeof(String));
betTable1.Columns.Add("Κάτι", typeof(String));
```

Αγγλία - Premier League

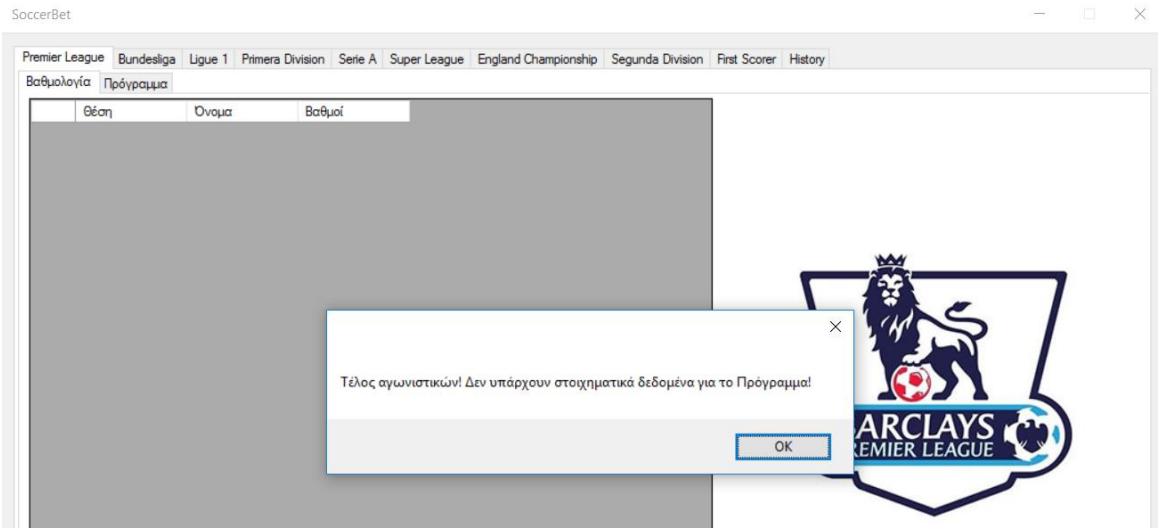
| | 1 | X | 2 | O/U 2,5 | GG/NG | |
|---|---|---|---|---|---|---|
| Γουότφορντ - Έβερτον<br>10/12 14:30 | 3.00 | 3.40 | 2.38 | O 1.95<br>U 1.83 | GG 1.75<br>NG 2.02 | +134 |
| Άρσεναλ - Στόουκ<br>10/12 17:00 | 1.38 | 5.25 | 7.50 | O 1.44<br>U 2.70 | GG 1.67<br>NG 2.12 | +137 |
| Μπέρνλι - Μπόρνμουθ<br>10/12 17:00 | 3.30 | 3.40 | 2.20 | O 2.00<br>U 1.80 | GG 1.80<br>NG 1.97 | +132 |
| Χαλ - Κρίσταλ Πάλας<br>10/12 17:00 | 3.10 | 3.35 | 2.35 | O 2.05<br>U 1.75 | GG 1.80<br>NG 1.95 | +132 |

Δομή του πίνακα στο stoiximan.gr , σε αντιστοιχία με τα πεδία που ορίσαμε παραπάνω

Στη συνέχεια, ξανά με τη λογική των nodes που αναφέρθηκε στην προηγούμενη ενότητα, προσπελαύνουμε το κάθε row του html table, κάνουμε parse τα δεδομένα και τα προσθέτουμε στα rows του betTable1.

```
HtmlNode sBet1 = doc1.DocumentNode.SelectSingleNode("//table");
            if (sBet1 != null)
            {

                var nodes1 = doc1.DocumentNode.SelectNodes("//table/tbody/tr");
                try
                {
                    var rows1 = nodes1.Select(tr => tr
                         .Elements("td")
                         .Select(td => StringReplaceChar(td.InnerText.Trim()))
                         .ToArray());
                    foreach (var row1 in rows1)
                    {
                        betTable1.Rows.Add(row1);
                    }
                }
                catch
                {
                    MessageBox.Show("Τέλος αγωνιστικών! Δεν υπάρχουν στοιχηματικά
δεδομένα για το Πρόγραμμα!");

                }
            }
```

Στον παραπάνω κώδικα παρατηρούμε ότι υπάρχει ένα try-catch block για το χειρισμό πιθανού exception. Αυτό είναι απολύτως αναγκαίο και χρήσιμο, καθώς σε περίπτωση που έχουν τελειώσει οι αγωνιστικές ενός πρωταθλήματος, τα site που περιέχουν τα data για τις στοιχηματικές αποδόσεις δεν περιέχουν καθόλου δεδομένα. Έτσι, αν ο κώδικας επιχειρήσει να προσπελάσει μια σελίδα χωρίς data θα αποτύχει και αυτό θα οδηγήσει σε αποτυχία εκτέλεσης της εφαρμογής. Εφόσον λοιπόν ισχύει κάτι τέτοιο, ο κώδικας θα οδηγηθεί στο catch block, όπου και έχουμε ορίσει την εμφάνιση ενός MessageBox με κατάλληλο ενημερωτικό μήνυμα προς τον χρήστη, χωρίς να επηρεάζεται η υπόλοιπη εφαρμογή (βαθμολογίες, ιστορικό κτλ.)




Προχωράμε με το parsing των αντίστοιχων δεδομένων για το πρόγραμμα αγωνιστικής από το betshop.gr. Εδώ πρέπει να σημειωθεί ότι τα 2 αυτά sites επιλέχθηκαν καθώς παρουσιάζουν τα δεδομένα για το πρόγραμμα κάθε αγωνιστικής σε πλήρη αντιστοιχία μεταξύ τους, οπότε αυτό καθιστά πολύ εύκολη τη διαχείριση μέσω της εφαρμογής.

Για τα δεδομένα λοιπόν του betshop χρησιμοποιούμε την HtmlAgilityPack βιβλιοθήκη, στην οποία θα φορτώσουμε το url που ορίζεται μέσα στη μεταβλητή *HtmlSource1* στην αρχή της κλάσης

```
public string HtmlSource1;
```

Η αντιστοίχιση του url γίνεται αρχικά μέσω του *WebControl2,* η λειτουργία του οποίου περιγράφεται στην επόμενη ενότητα.

```
webControl2.Source = urlFirstSite;
```

Το φόρτωμα της σελίδας γίνεται μέσα από την μέθοδο ***OnLoadingFrameComplete1***, για την οποία γίνεται ανάλυση στην επόμενη ενότητα. Εδώ γίνεται και η κλήση της ***checkDatagridHtml.***

```
HtmlSource1 = webControl2.ExecuteJavascriptWithResult("document.body.innerHTML");
checkDatagridHtml();
```

Θα προσπελάσουμε τα στοιχεία του html table χρησιμοποιώντας και πάλι τη λογική των nodes.

```
var nodes = doc.DocumentNode.SelectNodes("//table[contains(@class,'sports-
table')][1]/tbody/tr");
```

Ορίζουμε στη συνέχεια κάποιες απαραίτητες μεταβλητές, οι οποίες επεξηγούνται ακολούθως:

`int i = 0;` *Έχει το ρόλο μετρητή για την προσπέλαση των στοιχείων του betTable1 που δημιουργήθηκε προηγουμένως*

`float odd11` *Απόδοση για το στοιχείο 1 από το πρώτο site*

`float odd1X` *Απόδοση για το στοιχείο X από το πρώτο site*

`float odd12` *Απόδοση για το στοιχείο 2 από το πρώτο site*

`float odd21` *Απόδοση για το στοιχείο 1 από το δεύτερο site*

`float odd2X` *Απόδοση για το στοιχείο X από το δεύτερο site*

`float odd22` *Απόδοση για το στοιχείο 2 από το δεύτερο site*

`float mo1` *Μέσος όρος αποδόσεων των 2 sites για το στοιχείο 1*

`float moX` *Μέσος όρος αποδόσεων των 2 sites για το στοιχείο X*

`float mo2` *Μέσος όρος αποδόσεων των 2 sites για το στοιχείο 2*

`float moSumPercent` *Υπολογισμός ποσοστού απόδοσης από τα 2 sites*

`float mo1New` *Επαναυπολογισμένος μέσος όρος για το στοιχείο 1*

`float mo2New` *Επαναυπολογισμένος μέσος όρος για το στοιχείο X*

`float moXNew` *Επαναυπολογισμένος μέσος όρος για το στοιχείο 2*

`float moFactor = 0.116f;` *Ο συντελεστής που θα χρησιμοποιήσουμε για τον επαναυπολογισμό των μέσων όρων (δηλαδή η τιμή 116%)*

`var getDateTime = " ";` *Αποθήκευση τιμής για ημερομηνία και ώρα διεξαγωγής αγώνα*

`string strHomeTeam = "";` *Αποθήκευση τιμής για γηπεδούχο*

`string strawayTeam = "";` *Αποθήκευση τιμής για φιλοξενούμενη ομάδα*

Πραγματοποιούμε το parsing των δεδομένων με τη χρήση nodes όπως αναφέρθηκε και παραπάνω. Για κάθε τιμή που λαμβάνουμε κάνουμε αρχικά έλεγχο ώστε να διαπιστωθεί αν υπάρχει όντως τιμή ή το συγκεκριμένο στοιχείο είναι null, οπότε και θα καταχωρήσουμε στη μεταβλητή μια προκαθορισμένη ενδεικτική τιμή. Πρώτα λαμβάνουμε την τιμή για την ημερομηνία και ώρα διεξαγωγής του αγώνα.

```
var matchDate = node.SelectSingleNode(".//td/div[@class='datetime']/text()");
if (matchDate != null)
{
	var getDate = matchDate.InnerText.Trim() + "/" +
DateTime.Now.Year.ToString();

	var getTime =
node.SelectSingleNode(".//td/div[@class='datetime']/span[@class='time']/text()");

 getDateTime = getDate + " " + getTime.InnerText.Trim();
}
else
{
	getDateTime = "?? d?a??s?µ?";
}
```

Χρησιμοποιούμε δύο local μεταβλητές όπου αποθηκεύονται ξεχωριστά οι τιμές για την ημερομηνία και την ώρα και κατόπιν ενοποιούνται στην *getDateTime*. Εδώ γίνεται χρήση της μεθόδου **Trim()** που υπάρχει στη C#, η οποία αφαιρεί τους κενούς χαρακτήρες που μπορεί να υπάρχουν σε ένα string. Παρατηρούμε επίσης ότι στο path που ορίζουμε για το node χρησιμοποιούμε στην αρχή τους χαρακτήρες ***.//*** . Η χρήση της τελείας είναι απαραίτητη ώστε να πάρουμε τα td στοιχεία που ανήκουν μόνο στο table που έχουμε καθορίσει ότι βρίσκεται το αρχικό node, δηλαδή στο

```
//table[contains(@class,'sports-table')][1]/tbody/tr
```

Σε περίπτωση που παραλειπόταν η τελεία και κάθε node χρησιμοποιούσε για το parsing των τιμών ένα path της μορφής

```
//td/div[@class='datetime']/span[@class='time']/text()
```

τότε θα έπαιρνε όλα τα td στοιχεία από όλα τα tables που υπάρχουν στο html της σελίδας.

Με την ίδια λογική ακολουθεί το parsing για τις τιμές «Γηπεδούχος» και «Φιλοξενούμενοι»

```
var homeTeam = node.SelectSingleNode(".//td/div[@class='matches']/a/span[1]");
```

```
                    if (homeTeam != null)
                    {
                        strHomeTeam = homeTeam.InnerText;
                    }else
                    {
                        strHomeTeam = "hometeam";
                    }

                    var awayTeam =
node.SelectSingleNode(".//td/div[@class='matches']/a/span[2]");
                    if (awayTeam != null)
                    {
                        strawayTeam = awayTeam.InnerText;
                    }else
                    {
                        strawayTeam = "visitor";
                    }
```

Για τις στοιχηματικές τιμές του betshop επίσης ακολουθείται η παραπάνω διαδικασία. Εδώ χρησιμοποιούμε local μεταβλητές για την προσωρινή αποθήκευση της parsed τιμής και στη συνέχεια χρησιμοποιούμε την ενδιάμεση μεταβλητή *currText* για τη μετατροπή των τιμών από string σε αριθμό μέσω της μεθόδου ***Convert.ToSingle()***

```
var odd1 = node.SelectSingleNode(".//td[@class='odds']/span[1]");
                    if (odd1 != null)
                    {
                        var currText = StringReplaceChar(odd1.InnerText);
                        odd11 = Convert.ToSingle(currText);
                    }

var oddX = node.SelectSingleNode(".//td[@class='odds']/span[2]");
                    if (oddX != null)
                    {
                        var currText = StringReplaceChar(oddX.InnerText);
                        odd1X = Convert.ToSingle(currText);
                    }
var odd2 = node.SelectSingleNode(".//td[@class='odds']/span[3]");
                    if (odd2 != null)
                    {
                        var currText = StringReplaceChar(odd2.InnerText);
                        odd12 = Convert.ToSingle(currText);
                    }
```

Εδώ γίνεται χρήση της μεθόδου ***StringReplaceChar()***, την οποία έχουμε υλοποιήσει ώστε στην τιμή που λαμβάνουμε να γίνεται αντικατάσταση του χαρακτήρα **.** με **,** για να έχουμε τον δεκαδικό αριθμό της απόδοσης στη σωστή μορφή.

```
private string StringReplaceChar(string x)
        {
                if (x.Contains("."))
                x = x.Replace(".", ",");
        return x;
```

Ακολούθως κάνουμε την ίδια μετατροπή και για τις τιμές που έχουμε λάβει από το stoiximan, οι οποίες είναι αποθηκευμένες στον πίνακα *betTable1* όπως περιγράφηκε παραπάνω

```
if (betTable1.Rows[i].Field<string>("1") != null)
                    {
                        odd21 =
Convert.ToSingle(betTable1.Rows[i].Field<string>("1"));
                    }
if (betTable1.Rows[i].Field<string>("X") != null)
                    {
                        odd2X =
Convert.ToSingle(betTable1.Rows[i].Field<string>("X"));
                    }
if (betTable1.Rows[i].Field<string>("2") != null)
                    {
                        odd22 =
Convert.ToSingle(betTable1.Rows[i].Field<string>("2"));
                    }
```

Εφόσον έχουμε λάβει όλες τις τιμές που χρειαζόμαστε και από τα 2 sites, θα υπολογίσουμε τους μέσους όρους των 2 τιμών για τα στοιχεία 1 X και 2 ως εξής

```
mo1 = (odd11 + odd21) / 2;
moX = (odd1X + odd2X) / 2;
mo2 = (odd12 + odd22) / 2;
```

Κατόπιν υπολογίζουμε τον τρέχοντα συντελεστή απόδοσης που προκύπτει από το άθροισμα των μέσων όρων των στοιχείων 1 X και 2 που υπολογίστηκαν προηγουμένως.

```
moSumPercent = (mo1 + mo2 + moX) / 100;
```

Για να γίνει ο υπολογισμός των νέων αποδόσεων με το συντελεστή 116% που ορίσαμε παραπάνω, κάνουμε τους ακόλουθους υπολογισμούς, χρησιμοποιώντας ουσιαστικά την απλή μέθοδο των τριών.

```
mo1New = moFactor * mo1 / moSumPercent;
moXNew = moFactor * moX / moSumPercent;
mo2New = moFactor * mo2 / moSumPercent;
```

Στο σημείο αυτό έχει γίνει ο υπολογισμός για όλα τα στοιχεία ενός row που θα εισαχθεί στο DataGridView, οπότε κάνουμε add ένα νέο row με τον τρόπο που ακολουθεί (επεξήγηση για τη χρήση του Invoke γίνεται στην προηγούμενη ενότητα). Επίσης αυξάνουμε τον row counter που έχουμε ορίσει κατά 1, ώστε στο επόμενο στάδιο του foreach να μεταβεί στην επόμενη γραμμή.

```
if (this.dgvLeague.InvokeRequired)
{
        Invoke((MethodInvoker)() => dgvLeague.Rows.Add(getDateTime, strHomeTeam,
strawayTeam, odd1.InnerText, oddX.InnerText, odd2.InnerText, odd21.ToString(),
odd2X.ToString(), odd22.ToString(), mo1, moX, mo2, mo1New, moXNew, mo2New)));
}
else
```

```
{
        dgvLeague.Rows.Add(getDateTime, strHomeTeam, strawayTeam, odd1.InnerText,
oddX.InnerText, odd2.InnerText, odd21.ToString(), odd2X.ToString(),
odd22.ToString(), mo1, moX, mo2, mo1New, moXNew, mo2New);
}

i++;
```

Στο τέλος της ***checkDatagridHtml***, εφόσον προστέθηκαν όλα τα row, καθαρίζουμε τον ενδιάμεσο πίνακα *betTable1* από όλα τα δεδομένα που είχαμε αποθηκεύσει σε αυτόν.

```
betTable1.Clear();
```


| | Ημερομηνία - Ώρα | Γηπεδούχοι | Φιλοξενούμενοι | 1 | X | 2 | 1 | X | 2 | M.O: 1 | M.O: X | M.O: 2 | M.O.1 With Factor | M.O.X With Factor | M.O.2 With Factor |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ▶ | Σάββατο, 15/04... | Τότεναμ | Μπόρνμουθ | 1,23 | 6,50 | 9,75 | 1,26 | 6,20 | 11,00 | 0,615 | 3,25 | 4,875 | 0,8162472 | 4,313501 | 6,470252 |
| | Σάββατο, 15/04... | Στόουκ | Χαλ Σίτι | 1,97 | 3,40 | 4,00 | 1,83 | 3,60 | 4,45 | 0,985 | 1,7 | 2 | 2,438847 | 4,209178 | 4,951974 |
| | Σάββατο, 15/04... | Έβερτον | Μπέρνλι | 1,42 | 4,40 | 7,25 | 1,44 | 4,45 | 7,50 | 1,43 | 4,425 | 7,375 | 1,253817 | 3,879818 | 6,466364 |
| | Σάββατο, 15/04... | Κρίσταλ Πάλας | Λέστερ | 1,85 | 3,50 | 4,30 | 1,95 | 3,50 | 4,00 | 1,84 | 3,55 | 4,375 | 2,185766 | 4,217102 | 5,197133 |
| | Σάββατο, 15/04... | Γουότφορντ | Σουόνσι Σίτι | 2,45 | 3,25 | 2,95 | 3,10 | 3,55 | 2,25 | 2,14 | 3,425 | 3,7 | 2,679331 | 4,288181 | 4,632488 |
| | Σάββατο, 15/04... | Σάντερλαντ | Γουέστ Χαμ | 3,20 | 3,40 | 2,25 | 2,45 | 3,35 | 2,90 | 3,15 | 3,475 | 2,25 | 4,117183 | 4,541972 | 2,940845 |
| | Σάββατο, 15/04... | Σαουθάμπτον | Μάντσεστερ Σίτι | 4,25 | 3,85 | 1,75 | 4,25 | 4,00 | 1,75 | 4,25 | 3,925 | 1,75 | 4,967255 | 4,587406 | 2,04534 |
| | Κυριακή, 16/04/... | Γουέστ Μπρομ. | Λίβερπουλ | 4,20 | 3,55 | 1,85 | 4,15 | 3,65 | 1,88 | 4,225 | 3,775 | 1,8 | 5,00102 | 4,468368 | 2,130612 |
| | Κυριακή, 16/04/... | Μάντσεστερ Γ. | Τσέλσι | 2,65 | 3,10 | 2,75 | 2,65 | 3,25 | 2,75 | 3,45 | 3,55 | 2,25 | 4,326487 | 4,451891 | 2,821621 |

Οθόνη της καρτέλας «Πρόγραμμα» σε λειτουργία, με δεδομένα από το πρωτάθλημα της Premier League

## ΦΟΡΤΩΣΗ ΤΩΝ ΔΕΔΟΜΕΝΩΝ ΣΤΙΣ ΚΑΡΤΕΛΕΣ ΠΡΩΤΑΘΛΗΜΑΤΩΝ

Κατά το φόρτωμα της φόρμας για τη βασική οθόνη ενεργοποιείται η μέθοδος Form1_Load.

```
private void Form1_Load(object sender, EventArgs e)
        {
            webControl1.Source = leagueScore;
            webControl1.LoadingFrameComplete += OnLoadingFrameComplete;
            webControl2.Source = urlFirstSite;
            webControl2.LoadingFrameComplete += OnLoadingFrameComplete1;

            tabControl1.SelectedIndexChanged += Tabs_SelectedIndexChanged;
        }
```

Η μέθοδος αυτή καθορίζει ποια θα είναι τα πρώτα url που θα χρησιμοποιηθούν για λήψη δεδομένων, τα οποία όπως έχει προαναφερθεί ορίζονται στην αρχή του κώδικα και αφορούν την Premier League, μιας και είναι το πρώτο πρωτάθλημα που εμφανίζεται κατά την εκκίνηση της εφαρμογής. Παρατηρούμε εδώ τη χρήση 2 webControl στοιχείων.Το webcontrol είναι ένα εργαλείο που λειτουργεί σαν browser. Όταν δέχεται μία τιμή στη μέθοδο Source, φορτώνει την αντίστοιχη σελίδα και σε ένα object κρατάει όλα τα δεδομένα του. Από τα δύο webcontrols που έχουμε λοιπόν, το πρώτο φορτώνει τη βαθμολογία (leagueScore) και το δεύτερο τα προγράμματα από τα δύο επιλεγμένα websites( urlFirstSite, urlSecondSite).

Για να εξασφαλίσουμε ότι θα έχει ολοκληρωθεί το loading της σελίδας πρωτού ανασύρουμε δεδομένα από αυτήν, είναι απαραίτητη αυτή η γραμμή κώδικα:

```
webControl1.LoadingFrameComplete += OnLoadingFrameComplete;
```

Χωρίς αυτή τη γραμμή παρουσιάζονται σφάλματα όπως “doc is not ready” ή “doc is still loading” κάτι που μας αποτρέπει να πάρουμε όλα τα απαραίτητα δεδομένα. Ακολούθως φαίνεται η κατασκευή της ***OnLoadingFrameComplete***

```
private void OnLoadingFrameComplete(Object sender, FrameEventArgs e)
        {
            if (e.IsMainFrame)
            {
                countLoads++;
                if (countLoads < 2)
                {
                    string source1 =
webControl1.ExecuteJavascriptWithResult("document.body.innerHTML");
                    getLeagueScore(source1);
                }
            }
        }
```

Μέσα στη μέθοδο αυτή χρησιμοποιούμε τη μεταβλητή *countLoads*, η οποία ορίζεται και αρχικοποιείται στην αρχή του κώδικα στην κλαση Form1

```
public int countLoads = 0;
```

Με τη μεταβλητή αυτή ουσιαστικά υπολογίζουμε αν είναι η πρώτη φορά που φορτώνεται η σελίδα ή αν έχει ξαναφορτωθεί σε πρότερο χρόνο. Για κάθε φορά που η σελίδα φορτώνεται επιτυχώς ο μετρητής αυξάνεται κατά 1. Για την πρώτη φορά που θα φορτωθεί, θα πάρει την τιμή 1.

```
if (e.IsMainFrame)
{
       countLoads++;
```

Στη συνέχεια γίνεται έλεγχος αν είναι η πρώτη φορά που φορτώνει η σελίδα αυτή. Αν αυτό δεν ισχύει, δίνουμε εντολή στο webcontrol1 να αποθηκεύσει στη μεταβλητή *source1* το body από τη σελίδα που του έχουμε αντιστοιχίσει, ΕΦΟΣΟΝ εκτός από το dom (όλα τα html στοιχεία δηλαδή) έχει φορτωθεί και τρέξει και οτιδήποτε έχει να κάνει με Javascript στη σελίδα, μέσω της μεθόδου *ExecuteJavascriptWithResult*. Έτσι, εξασφαλίζουμε ότι θα έχουμε σαν δεδομένα ότι βλέπει ακριβώς ο επισκέπτης της κανονικής σελίδας και όχι μονο το αρχικό html.

```
if (countLoads < 2)
                {
                    string source1 =
webControl1.ExecuteJavascriptWithResult("document.body.innerHTML");
                    getLeagueScore(source1);
                }
```

Τα παραπάνω ισχύουν επακριβώς και για την ***OnLoadingFrameComplete1***, η οποία διαχειρίζεται το φόρτωμα της σελίδας που αντιστοιχεί στο *webControl2*. Συνεχίζουμε με την επεξήγηση της παρακάτω γραμμής η οποία επίσης βρίσκεται όπως είδαμε στην Form1_Load

```
tabControl1.SelectedIndexChanged += Tabs_SelectedIndexChanged;
```

Η μέθοδος ***Tabs_SelectedIndexChanged*** τρέχει κάθε φορά που ο χρήστης αλλάζει Tab και καθορίζει ποια url θα χρησιμοποιηθούν στην εκάστοτε περίπτωση για λήψη και εμφάνιση δεδομένων για το αντίστοιχο επιλεγμένο πρωτάθλημα. Στη δήλωση της φόρμας Form1 χρησιμοποιούμε τις εξής Boolean μεταβλητές:

```
public bool tab1Loaded = false;
public bool tab2Loaded = false;
public bool tab3Loaded = false;
public bool tab4Loaded = false;
public bool tab5Loaded = false;
public bool tab6Loaded = false;
public bool tab7Loaded = false;
public bool tab8Loaded = false;
public bool tab9Loaded = false;
```

Καθεμία από τις μεταβλητές αυτές αποτελεί στην ουσία ένα flag που σηματοδοτεί αν ο χρήστης φορτώνει το συγκεκριμένο tab για πρώτη φορά.

Όταν η μέθοδος ***Tabs_SelectedIndexChanged*** τρέχει ελέγχει σε ποιο tab είμαστε, και αποδίδει τα αντίστοιχα url στα urlFirstSite, urlSecondSite, leagueScore και επίσης σετάρει ποια θα είναι τα νέα DataGridViews για βαθμολογία και πρόγραμμα, δίνοντας τιμές στα *dvgScore* και *dvgLeague* αντίστοιχα. Έπειτα ελέγχει τη μεταβλητή *tab<tabnumber>Loaded* για να εξετάσει αν ο χρήστης βρίσκεται πρώτη φορά σε αυτό το tab ή όχι. Εάν βρίσκεται για πρώτη φορά (*tabLoaded = False*) τότε τρέχει την ***newPage()***. Η ***newPage()*** λαμβάνοντας υπόψιν της τα url που καθορίστηκαν προηγουμένως, τα φορτώνει στα webcontrols που αναφέρθηκαν παραπάνω.

Ένα τμήμα της υλοποίησης της ***Tabs_SelectedIndexChanged*** φαίνεται ακολούθως. Στο παράδειγμα αυτό, φαίνεται η αντιστοίχιση που γίνεται στην περίπτωση που επιλεχθεί το Tab με όνομα ‘Bundesliga’

```
string selectedTab = tabControl1.SelectedTab.Text;

            switch (selectedTab)
            {
                case "Bundesliga":
                    urlFirstSite = new Uri("https://www.betshop.gr/sports/l/3");
                    urlSecondSite = "https://www.stoiximan.gr/league/Soccer-
FOOT/Bundesliga-Germany-216";
                    leagueScore = new
Uri("http://www.escore.gr/football/germany/bundesliga/standings/");
                    dgvScore = dataGridView2;
                    dgvLeague = dataGridView4;
                    if (tab1Loaded == false)
                    {
                        leagueID = 2;
                        newPage();
                        tab1Loaded = true;
                        countLoads = 0;
                        countLoads1 = 0;
                    }else
```

```
                {
                    countLoads = 1;
                    countLoads1 = 1;
                }

                break;
```

Με ανάλογο τρόπο υλοποιείται η λειτουργικότητα για όλα τα Tabs, αντιστοιχίζοντας απλά στις μεταβλητές *urlFirstSite, urlSecondSite* και *leagueScore* τα κατάλληλα url. Ακολουθεί και η παρουσίαση του κώδικα της ***newPage()*** .

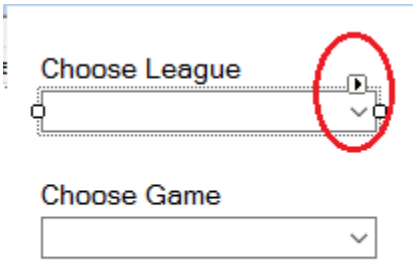
```
public void newPage(bool runScores=true)
        {

            Console.WriteLine("New Page");
            webControl1.Source = leagueScore;
            webControl1.LoadingFrameComplete += OnLoadingFrameComplete;
            if (runScores == true)
            {
                webControl2.Source = urlFirstSite;
                webControl2.LoadingFrameComplete += OnLoadingFrameComplete1;
            }
        }
```

## ΠΡΩΤΟΣ ΠΑΙΚΤΗΣ ΠΟΥ ΘΑ ΣΚΟΡΑΡΕΙ

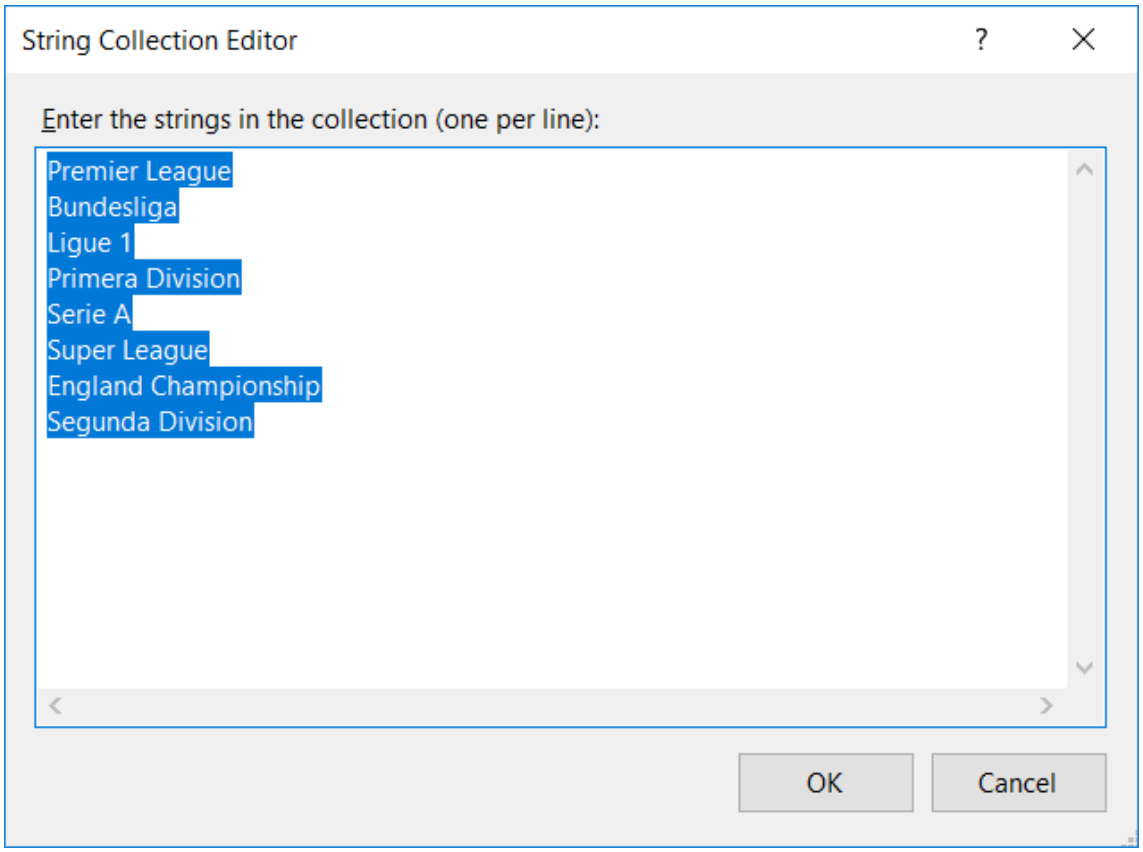
Για την σελίδα της καρτέλας ‘First Scorer’ χρησιμοποιούνται 2 ComboBox στοιχεία, τα οποία θα περιέχουν τις εξής τιμές:

1. Πρωτάθλημα
2. Αγώνες που βρίσκονται σε εξέλιξη για το επιλεγμένο πρωτάθλημα στην τρέχουσα αγωνιστική περίοδο

Αρχικά, στο 1ο comboBox εισάγουμε manually τα λεκτικά για τα πρωταθλήματα, κάνοντάς το edit πατώντας το εικονίδιο που φαίνεται στην εικόνα



Στη συνέχεια επιλέγουμε ‘Edit Items’ και εισάγουμε τους τίτλους των πρωταθλημάτων όπως απεικονίζεται ακολούθως

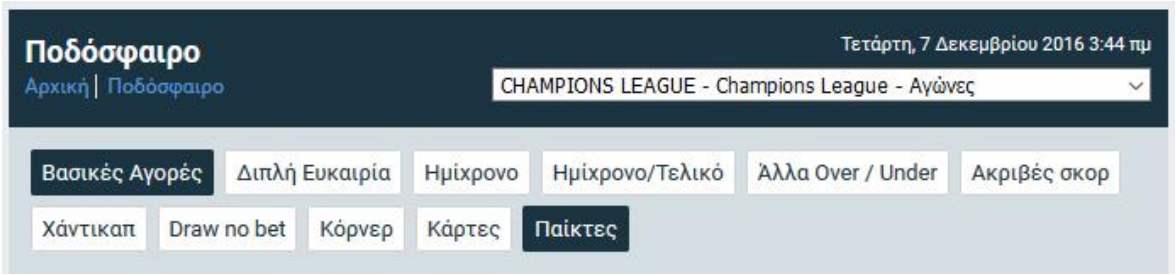



Οι τιμές του comboBox πρέπει να αντιστοιχηθούν με το url που θα φορτώνεται για την καθεμία, ώστε να κάνουμε parse τις αντίστοιχες τιμές. Τα δεδομένα για τον Πρώτο Παίκτη θα τα λαμβάνουμε από το www.stoiximan.gr . Η αντιστοίχιση κάθε τιμής με το σωστό url γίνεται μέσα στη μέθοδο ***comboBox1SelectedIndexChanged*** που παρατίθεται ακολούθως***.*** Αρχικά ορίζουμε τη string μεταβλητή selectedItem, όπου θα αποθηκεύεται η επιλεγμένη από το χρήστη τιμή στο comboBox. Ορίζουμε επίσης τη string μεταβλητή *urlLeague* όπου θα ορίζουμε το url που θα χρησιμοποιείται ανά περίπτωση. Το url αυτό θα χρησιμοποιείται σαν παράμετρος για την κλήση της ***loadLeagueScorers()*** η οποία περιγράφεται παρακάτω.

```
string selectedItem = comboBox1.Text;
string urlLeague = "";

switch (selectedItem)
{
case "Premier League":
     urlLeague = "https://www.stoiximan.gr/league/Soccer-FOOT/Premier-League-
England-1";
     loadLeagueScorers(urlLeague);
     break;
case "Bundesliga":
```

```
        urlLeague = "https://www.stoiximan.gr/league/Soccer-FOOT/Bundesliga-Germany-
216";
        loadLeagueScorers(urlLeague);
        break;
case "Ligue 1":
        urlLeague = "https://www.stoiximan.gr/league/Soccer-FOOT/Ligue-1-France-
215";
        loadLeagueScorers(urlLeague);
        break;
case "Primera Division":
        urlLeague = "https://www.stoiximan.gr/league/Soccer-FOOT/Primera-Division-
Spain-5";
        loadLeagueScorers(urlLeague);
        break;
case "Serie A":
        urlLeague = "https://www.stoiximan.gr/league/Soccer-FOOT/Serie-A-Italy-
1635";
        loadLeagueScorers(urlLeague);
        break;
case "Super League":
        urlLeague = "https://www.stoiximan.gr/league/Soccer-FOOT/Super-League-
Greece-1636";
        loadLeagueScorers(urlLeague);
        break;
case "England Championship":
        urlLeague = "https://www.stoiximan.gr/league/Soccer-FOOT/Championship-
England-2r";
        loadLeagueScorers(urlLeague);
        break;
case "Segunda Division":
        urlLeague = "https://www.stoiximan.gr/league/Soccer-FOOT/Europa-League-
Matches-Europa-League-182761";
        loadLeagueScorers(urlLeague);
        break;
}
```

Τα στοιχεία για τον ‘Πρώτο Παίκτη’ θα ληφθούν όπως προαναφέρθηκε από το stoiximan.gr. Η μορφή της σελίδας που περιέχει τα στοιχεία αυτά μέσα στο site (στην οποία οδηγούμαστε από τα παραπάνω url) φαίνεται στην επόμενη εικόνα.

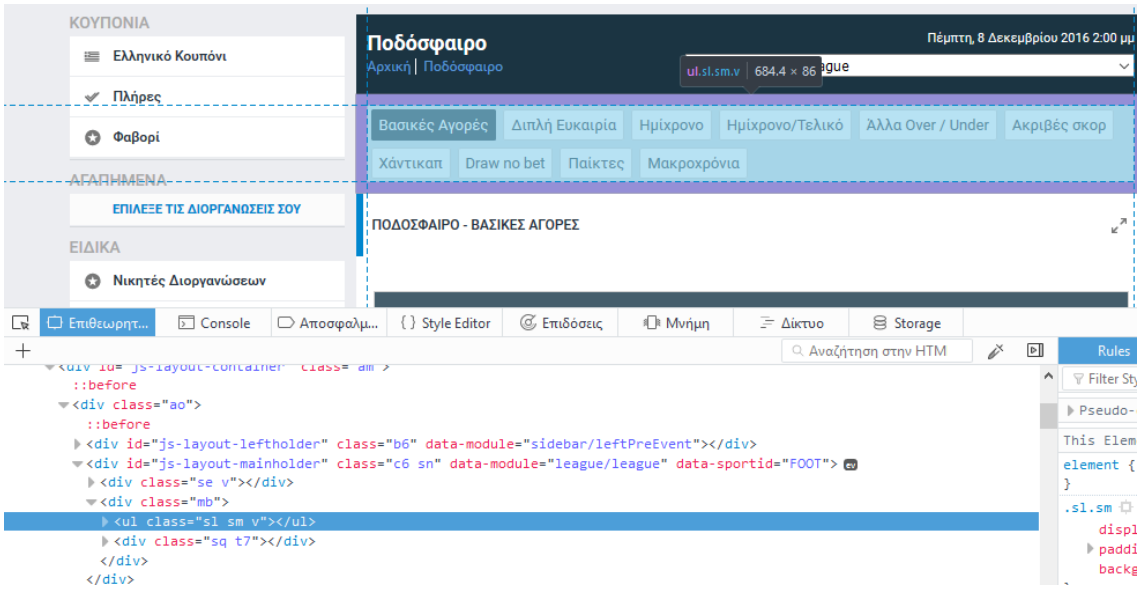



Όπως βλέπουμε, υπάρχει ένα μενού με πολλές διαθέσιμες επιλογές. Η ενότητα περιέχει τα δεδομένα του ενδιαφέροντός μας είναι η καρτέλα «Παίκτες». Η υλοποίηση της μεθόδου ***loadLeagueScorers()*** θα προσπαθήσει ακριβώς να μιμηθεί τη διαδικασία που ακολουθεί ο χρήστης μέσα στο παραπάνω μενού μέχρι να πατήσει στην ενότητα «Παίκτες» ώστε να δει τα αντίστοιχα δεδομένα. Το 1ο βήμα είναι να φορτωθεί η σελίδα που προέρχεται από το url που περνάει σαν παράμετρος κλήσης της μεθόδου αυτής. Η σελίδα θα φορτωθεί και θα αναλυθεί

μέσω ενός HtmlWeb αντικειμένου, όπως περιγράφηκε και σε προηγούμενη φάση στην εργασία αυτή.

```
HtmlWeb HtmlWEB1 = new HtmlWeb();
HtmlAgilityPack.HtmlDocument docLeague = HtmlWEB1.Load(urlLeague);
docLeague = HtmlWEB1.Load(urlLeague);
```

Το 2ο βήμα είναι να αποθηκεύσουμε σε μια μεταβλητή την 'τοποθεσία' του παραπάνω μενού μέσα στη σελίδα. Η μεταβλητή αυτή θα ονομάζεται *sBet1*. Για τον εντοπισμό της θέσης του μενού χρησιμοποιούμε και πάλι τον 'Έλεγχο αντικειμένου' μέσω του browser.

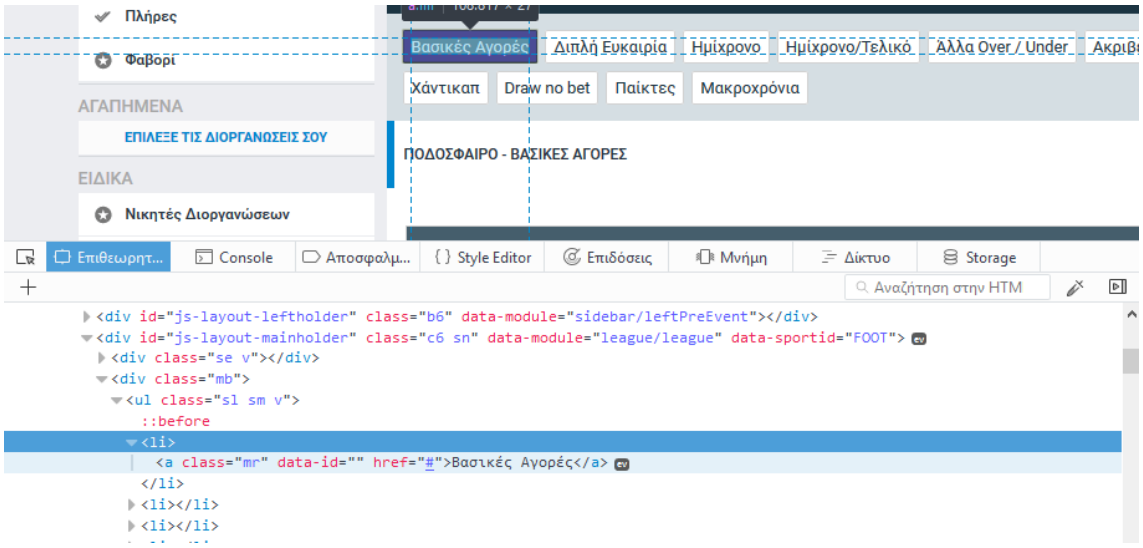


Η θέση του μενού μέσα στη σελίδα

Αποθηκεύουμε λοιπόν στην *sBet1* το παραπάνω node.

```
HtmlNode sBet1 = docLeague.DocumentNode.SelectSingleNode("//div[@id='js-layout-
mainholder']/div[@class='mb']/ul[contains(@class, 'sl')]");
```

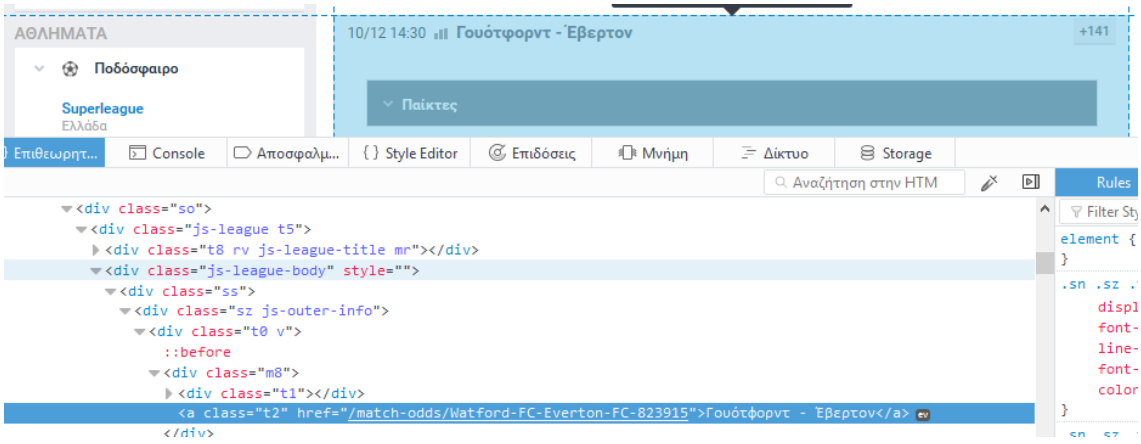
Το 3ο βήμα είναι να 'καθοδηγήσουμε' την εφαρμογή να πατήσει την επιλογή «Παίκτες». Εδώ λοιπόν αρχικά αποθηκεύουμε στη μεταβλητή *nodes* όλο το block των επιλογών αυτών. Όπως φαίνεται και στην επόμενη εικόνα, κάνοντας έλεγχο αντικειμένου σε μία από τις επιλογές θα δούμε ότι ανήκουν όλες σε μια html λίστα, άρα είναι list items που καθορίζονται από το header <li> και το λεκτικό τους βρίσκεται μέσα στο tag <a>




Πρέπει λοιπόν η εφαρμογή να εντοπίσει αν το λεκτικό του τρέχοντος list item που παρσάρει είναι το ‘Παίκτες΄ και στην περίπτωση που αυτό είναι αληθές παίρνει το href για την τοποθεσία αυτή και το ενώνει με το hostname https://www.stoiximan.gr μέσα στο string *urlPlayers* ώστε να σχηματίσει το πλήρες url για την επιλογή ‘Παίκτες’ του μενού.

```
if (sBet1 != null)
            {

                var nodes = sBet1.SelectNodes(".//li/a");

                foreach (var row in nodes)
                {

                    if (row.InnerHtml == "Παίκτες")
                    {
                        string PlayersHref = row.Attributes["href"].Value;
                        urlPlayers = ("https://www.stoiximan.gr" +
PlayersHref).Replace("&amp;", "&");
                    }

                }
```

Η τελευταία διαδικασία που πρέπει να γίνει είναι αφενός ο browser της εφαρμογής να φορτώσει μέσω ενός HtmlWeb το url που δημιουργήθηκε προηγουμένως και αφετέρου να γεμίσουμε το ***comboBox2*** με τους αγώνες που εμφανίζονται στη σελίδα ‘Παίκτες΄. Στην επόμενη εικόνα φαίνεται η θέση ενός αγώνα μέσα στη σελίδα.



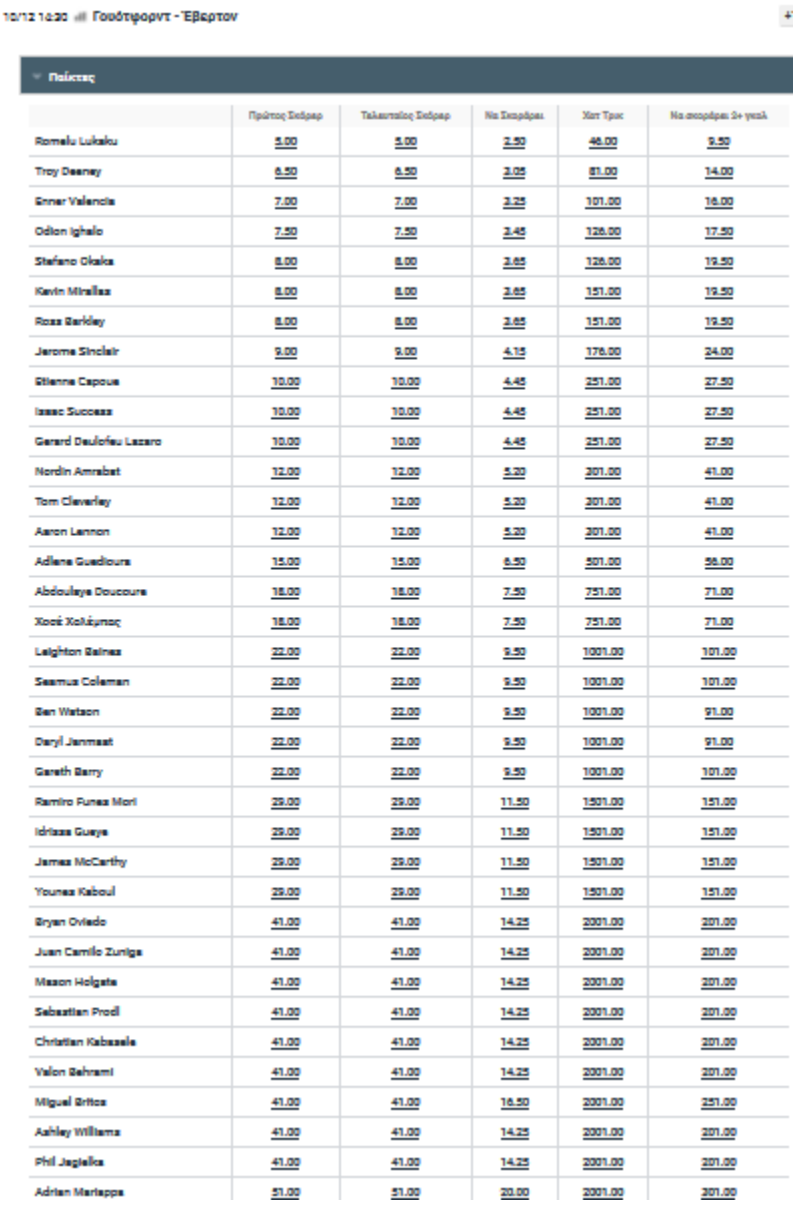

Διαπιστώνουμε λοιπόν ότι η βασική κλάση που περιέχει τον πίνακα αυτό είναι η js-league-body, οπότε ορίζουμε το αρχικό node σε αυτή τη θέση και το αποθηκεύουμε στη μεταβλητή *sBet2*. Στη συνέχεια δημιουργούμε ένα νέο node με όνομα *nodes1* που θα 'κοιτάει' στην υποκλάση ss της js-league-body και τέλος κάνουμε parse τον αγώνα δημιουργώντας τη μεταβλητή *node1* η οποία θα προσπελάσει όλα τα στοιχεία μέσα στο *nodes1*. Το λεκτικό κάθε αγώνα προστίθεται στα items του ***comboBox2*** με τη μέθοδο ***comboBox2.Items.Add()***

```
if (urlPlayers != "")
                {
                    HtmlWeb HtmlWEB2 = new HtmlWeb();
                    HtmlAgilityPack.HtmlDocument docPlayers =
HtmlWEB2.Load(urlPlayers);
                    docPlayers = HtmlWEB2.Load(urlPlayers);

                    HtmlNode sBet2 =
docPlayers.DocumentNode.SelectSingleNode("//div[@class='js-league-body']");
                    if (sBet2 != null)
                    {
                        var nodes1 = sBet2.SelectNodes(".//div[@class='ss']");
                        comboBox2.Items.Clear();
                        foreach (var node1 in nodes1)
                        {
                            arrMatches.Add(node1);
                            var matchName =
node1.SelectSingleNode(".//div[@class='m8']/a[@class='t2']/text()");
                            comboBox2.Items.Add(matchName.InnerHtml);

                        }
                    }
                }
```

Παρατηρούμε παραπάνω επίσης τη χρήση της μεταβλητής *arrMatches*. Η μεταβλητή αυτή έχει οριστεί στην αρχή της κλάσης ως εξής:

```
public List<HtmlNode> arrMatches = new List<HtmlNode>();
```

Η μεταβλητή αυτή είναι μία λίστα που γεμίζει με στοιχεία τύπου ***HtmlNode*** και θα χρησιμοποιηθεί στην εφαρμογή για αποθήκευση blocks που περιλαμβάνουν τα στοιχεία για τον «Πρώτο παίκτη» κάθε αγώνα, ακριβώς με τη μορφή που υπάρχουν μέσα στη σελίδα που έχει γίνει parse μέσω του *node1*. Η επόμενη εικόνα δείχνει ακριβώς πώς θα είναι ένα τέτοιο block που θα αποθηκεύεται σαν στοιχείο στην *arrMatches*



10/12 14:30 Γουότφορντ - Έβερτον

Παίκτης

| | Πρώτος Σκόρερ | Τελευταίος Σκόρερ | Να Σκοράρει | Χατ Τρικ | Να σκοράρει 2+ γκολ |
|---|---|---|---|---|---|
| Romelu Lukaku | 5.00 | 5.00 | 2.50 | 46.00 | 9.50 |
| Troy Deeney | 6.50 | 6.50 | 3.05 | 81.00 | 14.00 |
| Enner Valencia | 7.00 | 7.00 | 3.25 | 101.00 | 16.00 |
| Odion Ighalo | 7.50 | 7.50 | 3.45 | 126.00 | 17.50 |
| Stefano Okaka | 8.00 | 8.00 | 3.65 | 126.00 | 19.50 |
| Kevin Mirallas | 8.00 | 8.00 | 3.65 | 151.00 | 19.50 |
| Ross Barkley | 8.00 | 8.00 | 3.65 | 151.00 | 19.50 |
| Jerome Sinclair | 9.00 | 9.00 | 4.15 | 176.00 | 24.00 |
| Etienne Capoue | 10.00 | 10.00 | 4.45 | 251.00 | 27.50 |
| Isaac Success | 10.00 | 10.00 | 4.45 | 251.00 | 27.50 |
| Gerard Deulofeu Lazaro | 10.00 | 10.00 | 4.45 | 251.00 | 27.50 |
| Nordin Amrabat | 12.00 | 12.00 | 5.20 | 301.00 | 41.00 |
| Tom Cleverley | 12.00 | 12.00 | 5.20 | 301.00 | 41.00 |
| Aaron Lennon | 12.00 | 12.00 | 5.20 | 301.00 | 41.00 |
| Adlene Guedioura | 15.00 | 15.00 | 6.50 | 501.00 | 56.00 |
| Abdoulaye Doucoure | 18.00 | 18.00 | 7.50 | 751.00 | 71.00 |
| Χοσέ Χολέμος | 18.00 | 18.00 | 7.50 | 751.00 | 71.00 |
| Leighton Baines | 22.00 | 22.00 | 9.50 | 1001.00 | 101.00 |
| Seamus Coleman | 22.00 | 22.00 | 9.50 | 1001.00 | 101.00 |
| Ben Watson | 22.00 | 22.00 | 9.50 | 1001.00 | 91.00 |
| Daryl Janmaat | 22.00 | 22.00 | 9.50 | 1001.00 | 91.00 |
| Gareth Barry | 22.00 | 22.00 | 9.50 | 1001.00 | 101.00 |
| Ramiro Funes Mori | 29.00 | 29.00 | 11.50 | 1501.00 | 151.00 |
| Idrissa Gueye | 29.00 | 29.00 | 11.50 | 1501.00 | 151.00 |
| James McCarthy | 29.00 | 29.00 | 11.50 | 1501.00 | 151.00 |
| Younes Kaboul | 29.00 | 29.00 | 11.50 | 1501.00 | 151.00 |
| Bryan Oviedo | 41.00 | 41.00 | 14.25 | 2001.00 | 201.00 |
| Juan Camilo Zuniga | 41.00 | 41.00 | 14.25 | 2001.00 | 201.00 |
| Mason Holgate | 41.00 | 41.00 | 14.25 | 2001.00 | 201.00 |
| Sebastian Prodl | 41.00 | 41.00 | 14.25 | 2001.00 | 201.00 |
| Christian Kabasele | 41.00 | 41.00 | 14.25 | 2001.00 | 201.00 |
| Valon Behrami | 41.00 | 41.00 | 14.25 | 2001.00 | 201.00 |
| Miguel Britos | 41.00 | 41.00 | 16.50 | 2001.00 | 251.00 |
| Ashley Williams | 41.00 | 41.00 | 14.25 | 2001.00 | 201.00 |
| Phil Jagielka | 41.00 | 41.00 | 14.25 | 2001.00 | 201.00 |
| Adrian Mariappa | 51.00 | 51.00 | 20.00 | 2001.00 | 201.00 |

Το γέμισμα της *arrMatches* γίνεται με την εντολή

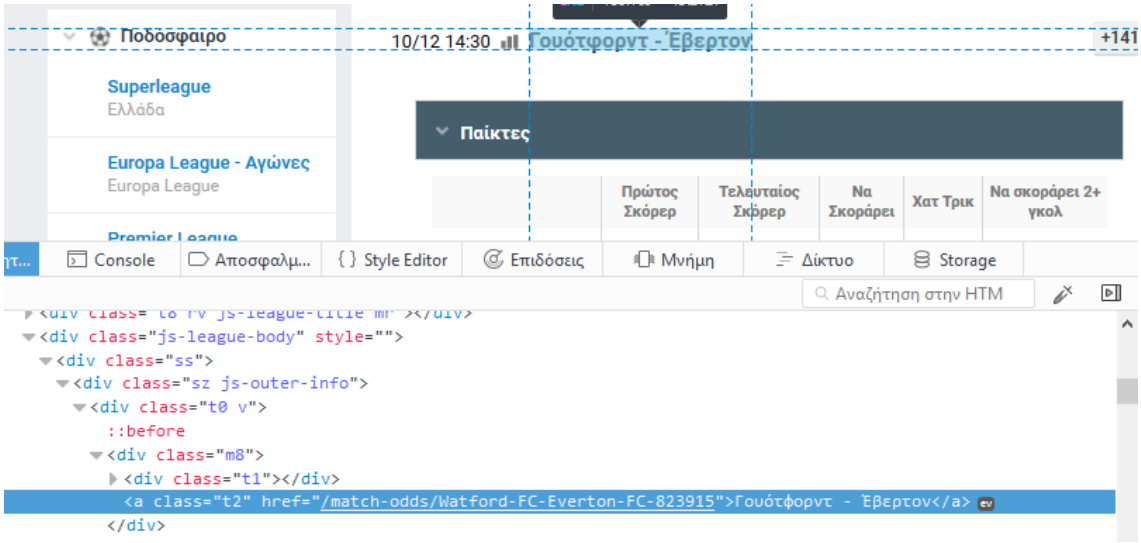
```
arrMatches.Add(node1);
```

η οποία όπως είδαμε υπάρχει μέσα στην ***loadLeagueScorers()***

Τα στοιχεία που αποθηκεύτηκαν στην παραπάνω λίστα θα αντιστοιχηθούν τώρα με την επιλογή αγώνα μέσα στο comboBox2, ώστε τα δεδομένα για Παίκτη και Απόδοση από τα κατάλληλα blocks να εισαχθούν στο ***dataGridView13.*** Η διαδικασία αυτή γίνεται μέσα στη μέθοδο ***comboBox2_SelectedIndexChanged()***. Το parsing των δεδομένων γίνεται με nodes όπως έχουμε δει και στις προηγούμενες παρόμοιες διαδικασίες, με τη διαφορά ότι τα δεδομένα

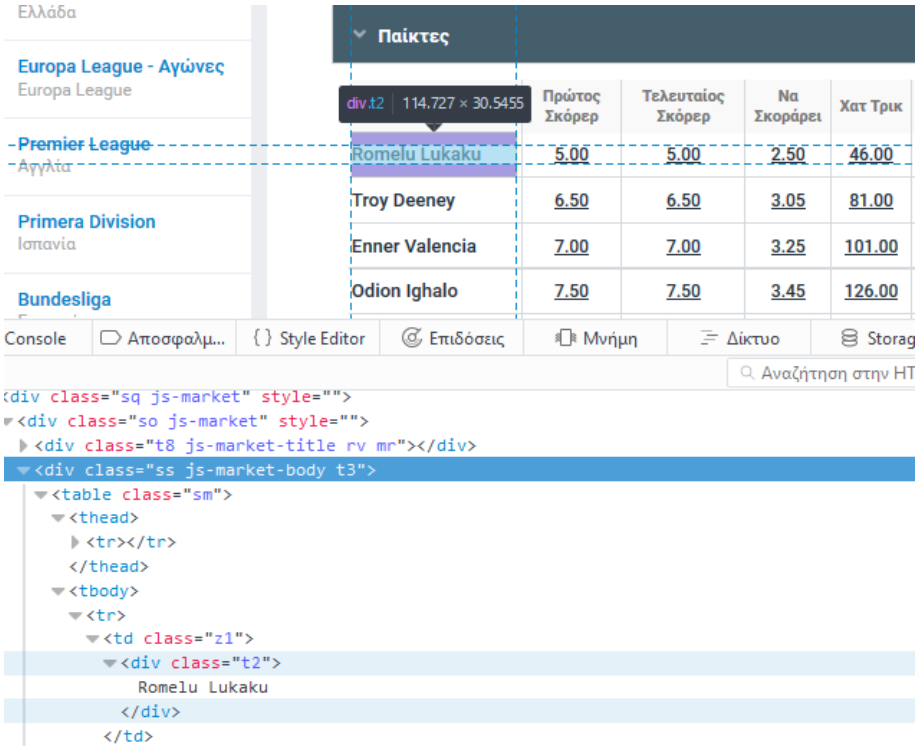
τώρα βρίσκονται μέσα στη λίστα *arrMatches* και όχι στο site. Για να καθοδηγηθούμε όμως ώστε να ξέρουμε που βρίσκονται τα δεδομένα που χρειαζόμαστε, θα χρησιμοποιήσουμε κανονικά μέσα στο site τον έλεγχο αντικειμένου μέσα στον browser.

Η πρώτη διαδικασία είναι να κάνουμε parse τον τίτλο κάθε αγώνα και να ελέγξουμε αν το επιλεγμένο item στο comboBox2 (το οποίο ορίζουμε στη μεταβλητή *selectedItem*) είναι το ίδιο. Ο τίτλος αγώνα που γίνεται parse θα αποθηκεύεται στη μεταβλητή *matchName*. Η θέση του τίτλου αγώνα μέσα στη σελίδα φαίνεται στην επόμενη εικόνα

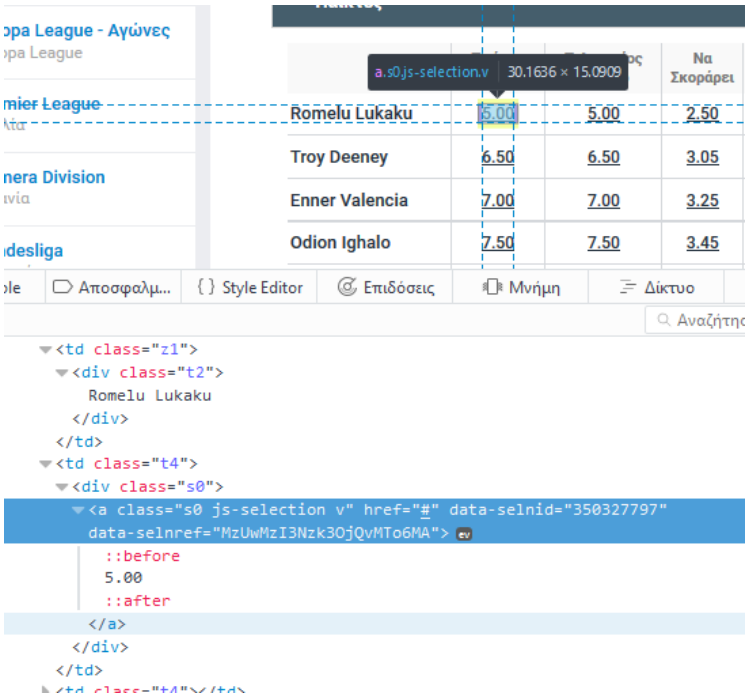



```
string selectedItem = comboBox2.Text;
            foreach (var currDoc in arrMatches)
            {
                var matchName =
currDoc.SelectSingleNode(".//div[@class='m8']/a[@class='t2']/text()");

                if (matchName.InnerHtml == selectedItem)
                {
                    var tableScorer =
currDoc.SelectSingleNode(".//div[contains(@class,'js-market-
body')]/table[@class='sm']/tbody");
                    dataGridView13.Rows.Clear();
                    dgvPlayersOdss.Rows.Clear();
```

Εφόσον αντιστοιχηθεί το επιλεγμένο comboBox item με τον parsed τίτλο αγώνα, δημιουργούμε ένα νέο node με τίτλο *tableScorer* το οποίο θα 'κοιτάει' το table με τα δεδομένα, το οποίο φαίνεται στην επόμενη εικόνα. Επίσης, κάνουμε Clear τα δεδομένα από τα 2 dataGridView που υπάρχουν στη σελίδα, για την περίπτωση που πριν είχαν γεμίσει από προηγούμενες επιλογές αγώνων του χρήστη.




Τα δεδομένα που μας ενδιαφέρουν βρίσκονται στην 1η και 2η στήλη του παραπάνω table. Επομένως πρέπει να δημιουργήσουμε ένα node που θα κάνει προσπέλαση τον πίνακα ανά σειρά και για κάθε σειρά θα αποθηκεύει τις 2 αυτές τιμές. Για το λόγο αυτό δημιουργούμε τις 2 local μεταβλητές *playerName* και *playerOdds,* καθώς και το node *tableScorerRows*, που θα πηγαίνει σε κάθε σειρά του πίνακα. Η θέση της τιμής στην 1η στήλη φαίνεται στην προηγούμενη εικόνα, ενώ για τη 2η στήλη φαίνεται ακολούθως



Εφόσον λοιπόν βρεθούν οι τιμές αυτές δεν είναι null, προστίθενται σαν row μέσα στο ***dataGridView13***. Ακολουθεί ο κώδικας που υλοποιεί όλη την παραπάνω διαδικασία.

```
if (tableScorer != null)
{
	var tableScorerRows = tableScorer.SelectNodes(".//tr");
	foreach (var singleRow in tableScorerRows)
	{
		var playerName =
singleRow.SelectSingleNode(".//td[1]/div[@class='t2']");
		var playerOdds =
singleRow.SelectSingleNode(".//td[2]/div[@class='s0']/a/text()");
		if ((playerName != null) && (playerOdds != null))
		{

dataGridView13.Rows.Add(playerName.InnerHtml.Trim(),
playerOdds.InnerHtml.Trim());
		}
	 }
}
```

Αφού ολοκληρωθεί η καταχώρηση όλων των παικτών για τον επιλεγμένο αγώνα στο ***dataGridView13***, ο χρήστης θα μπορεί να μεταφέρει 11 παίκτες της επιλογής του στο δεύτερο dataGridView (***dgvPlayersOdds***). Για το λόγο αυτό έχει δημιουργηθεί η μέθοδος ***dataGridView13_CellDoubleClick()***, με την οποία ο χρήστης κάνοντας διπλό κλικ σε έναν παίκτη (δηλαδή row) του ***dataGridView13*** το μεταφέρει στο ***dgvPlayersOdds***. Ο κώδικας της μεθόδου είναι ο εξής:

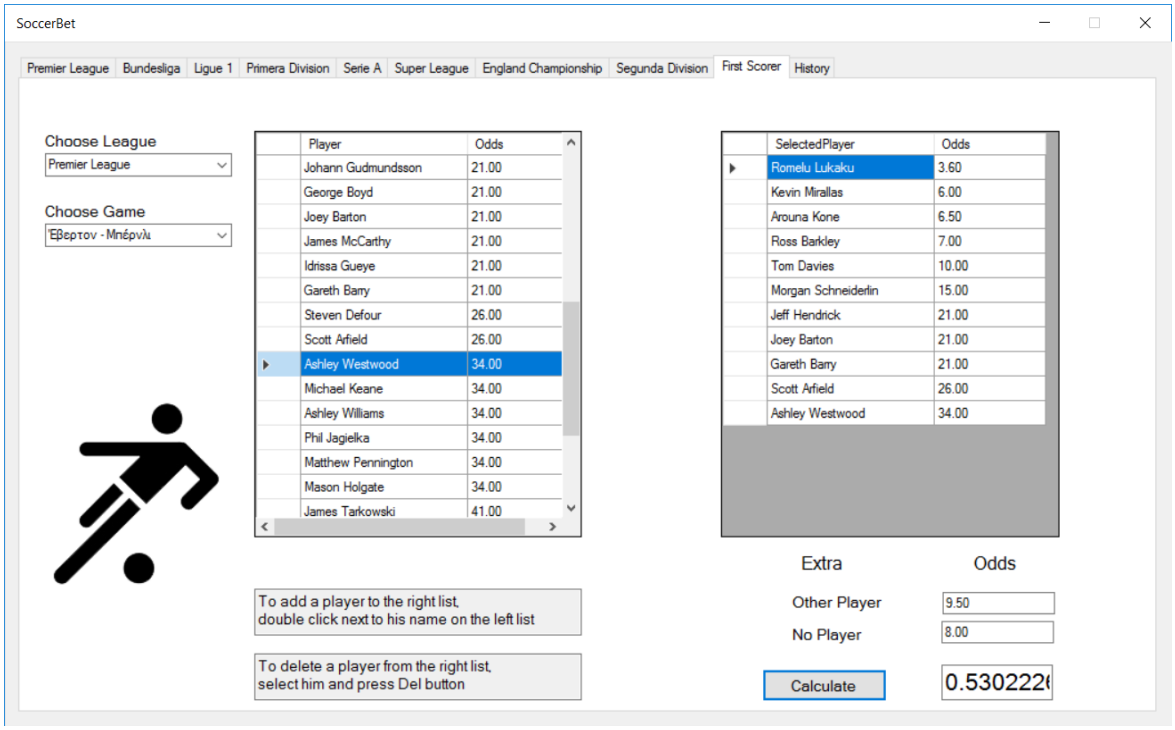
```
if (dataGridView13.SelectedRows.Count > 0)
{
	var selectedRow = dataGridView13.SelectedRows[0];
	var selectedRowName = selectedRow.Cells[0].Value;
	var selectedRowOdds = selectedRow.Cells[1].Value;

	if (dgvPlayersOdss.RowCount < 11)
	{
		dgvPlayersOdss.Rows.Add(selectedRowName, selectedRowOdds);
	}
}
```

Αρχικά ελέγχεται αν ο χρήστης έχει επιλέξει σειρά στο ***dataGridView13*** με τη μέθοδο ***Count().*** Εφόσον αυτό ισχύει, αποθηκεύουμε την επιλεγμένη σειρά στην local μεταβλητή *selectedRow* και στη συνέχεια ξεχωρίζουμε τις 2 τιμές που περιέχει αποθηκεύοντάς τις στις 2 local μεταβλητές ***selectedRowName*** & ***seletctedRowOdds***. Στη συνέχεια γίνεται έλεγχος στο ***dgvPlayersOdss*** με τη μέθοδο ***RowCount()*** ώστε να διαπιστωθεί αν υπάρχει διαθέσιμη σειρά για νέα καταχώρηση (δηλαδή δεν έχει συμπληρωθεί η 11αδα παικτών) και αν όντως υπάρχει ελεύθερη θέση στο DataGridView γίνεται Add η καταχώρηση αυτή. Υπάρχει επίσης δυνατότητα να αφαιρεθεί μια εγγραφή, απλά επιλέγοντάς τη και πατώντας Delete. Το πεδίο που βρίσκεται η τιμή για την απόδοση κάθε παίκτη στο ***dgvPlayersOdss*** έχει οριστεί ως editable, ώστε ο χρήστης να μπορεί να τροποποιεί όποια τιμή επιθυμεί και να προχωρά σε επαναυπολογισμό του συνολικού συντελεστή ποσοστού των αποδόσεων.

Στη συνέχεια ο χρήστης εισάγει 2 τιμές που ο ίδιος επιθυμεί στα πεδία '*Other Player*' & '*No Player*' και μπορεί να προχωρήσει στον υπολογισμό του συνολικού ποσοστού απόδοσης για τις 13 αυτές τιμές. Για την εκτέλεση του υπολογισμού χρησιμοποιείται το button '*Calculate*' και η μέθοδος που διαχειρίζεται τον υπολογισμό είναι η ***btnCalculateMoPlayers_Click()***

```
CultureInfo customCulture =
(System.Globalization.CultureInfo)System.Threading.Thread.CurrentThread.CurrentCu
lture.Clone();
customCulture.NumberFormat.NumberDecimalSeparator = ".";

Thread.CurrentThread.CurrentCulture = customCulture;
decimal extraPlayer1, extraPlayer2;
extraPlayer1 = extraPlayer2 = 0.00M;

extraPlayer1 = Convert.ToDecimal(extraPlayer1Odds.Text);
extraPlayer2 = Convert.ToDecimal(extraPlayer2Odds.Text);

decimal moSum = 0.00M;
int i = 2;
foreach (DataGridViewRow row in dgvPlayersOdss.Rows)
{
       var currOdds = Convert.ToDecimal(row.Cells[1].Value);
       moSum = moSum + currOdds;
       i = i + 1;
}

moSum = (moSum + extraPlayer1 + extraPlayer2) / 13 * 10;
txtMoPlayers.Text = Convert.ToInt16(moSum).ToString();
```

Εδώ ορίζουμε αρχικά τη μεταβλητή *customCulture* για να καθορίσουμε στο σύστημα τη χρήση της τελείας για τους δεκαδικούς αριθμούς, καθώς στο ελληνικό μετρικό σύστημα χρησιμοποιείται το κόμμα για το σκοπό αυτό. Αντιστοιχούμε τη μεταβλητή αυτή στο Thread που χρησιμοποιεί η εφαρμογή και κατόπιν ορίζουμε τις μεταβλητές δεκαδικού τύπου *extraPlayer1* και *extraPlayer2* στις οποίες κάνουμε parse τις 2 τιμές που εισήγαγε ο χρήστης στα input fields *extraPlayer1Odds* & *extraPlayer2Odds*, αφού πρώτα τις μετατρέψουμε από string σε δεκαδικούς με τη μέθοδο ***Convert.ToDecimal().*** Στη συνέχεια ορίζουμε τη μεταβλητή δεκαδικού τύπου *moSum* όπου με τον foreach βρόγχο υπολογίζουμε το άθροισμα των αποδόσεων για όλες τις σειρές που περιέχει το DataGridView ***dgvPlayersOdss***. Τέλος, προσθέτουμε στη *moSum* τις 2 τιμές των input fields, διαιρούμε τη *moSum* με το 13 χρησιμοποιώντας τη μέθοδο ***Divide()*** και εμφανίζουμε την τελική τιμή που προκύπτει στο field txtMoPlayers, αφού τη μετατρέψουμε σε string μέσω της ***Convert.ToInt16(moSum).ToString();***




Οθόνη της καρτέλας «Πρώτος Παίκτης» σε λειτουργία, με δεδομένα από το πρωτάθλημα της Premier League

## ΑΠΟΘΗΚΕΥΣΗ ΚΑΙ ΕΜΦΑΝΙΣΗ ΙΣΤΟΡΙΚΟΥ

Για την αποθήκευση του ιστορικού χρησιμοποιείται η μέθοδος insertDataInScoreTable(). Αυτή δέχεται δύο μεταβλητές: την *currArray* και την *roundNumber*. Το *currArray* είναι κάθε φορά το *pointsTabletable* από την *getLeagueScore()*, και εμπερικλείει στην ουσία τα αποτελέσματα του parsing της εκάστοτε σελίδας που περιέχει βαθμολογία διοργανώσεως. Η μεταβλητή *roundNumber* έρχεται επίσης από την προαναφερθείσα function και περιέχει τον αριθμό αγωνιστικής της τρέχουσας εβδομάδος.

Για την αποθήκευση του ιστορικού χρειάζεται να δημιουργήσουμε μία καινούρια σύνδεση στη βάση και κατόπιν να την ανοίξουμε. Τα στοιχεία της σύνδεσης περιέχονται στο αρχείο *DBUtils.cs*

```
MySqlConnection connection = DBUtils.GetDBConnection();
connection.Open();
```

Έπειτα σε ένα string με όνομα *sql* αποθηκεύουμε το ερώτημα προς τη βάση. Οι τιμές είναι παραμετρικές, π.χ. *@lid*, *@round*, για μεγαλύτερη ασφάλεια και αποφυγή Sql injections. Κατόπιν δημιουργούμε μια καινούρια MySql εντολή (cmd) στην οποία θέτουμε τη σύνδεση από πριν, καθώς και σαν CommandText το *sql*.

```
string sql = "Insert into sli_score(lid, round, week, year, position, name,
points, leaguegroup) values ( @lid, @round, @week, @year, @position, @name,
@points, @group)";

MySqlCommand cmd = new MySqlCommand();
                cmd.Connection = connection;
                cmd.CommandText = sql;
                cmd.Prepare();
```

Το επόμενο βήμα είναι να δημιουργήσουμε τις παραμέτρους χρησιμοποιώντας την εντολή

*MysqlParameter <Variablename> = new MySqlParameter('<parametername>, <parameter Type>;*

και αμέσως μετά να τις προσθέσουμε στην εντολή. Επομένως, πλέον για να ορίσουμε τιμή π.χ. για την @week παράμετρο του ερωτήματος sql της εντολής cmd, θα τη θέσουμε στη μεταβλητή *weekParam*.

```
MySqlParameter weekParam = new MySqlParameter("@week", MySqlDbType.Int16);
```

Οι μεταβλητές *Position*, *name*, *points* και *group* μας έρχονται απο το currArray. Η μεταβλητή της διοργάνωσης *leagueID* έχει οριστεί κατά την αλλαγή Tab, ενώ η *round* αποτελεί τη μία από τις δύο μεταβλητές που ορίζουμε στη function. Τέλος, τα *week* και *year* τα υπολογίζουμε με βάση την τρέχουσα εβδομάδα κα το τρέχων έτος αντίστοιχα.

```
DateTime thisDay = DateTime.Today;
CultureInfo ciCurr = CultureInfo.CurrentCulture;
int yearNum = ciCurr.Calendar.GetYear(thisDay);
int weekNum = ciCurr.Calendar.GetWeekOfYear(thisDay,
CalendarWeekRule.FirstFourDayWeek, DayOfWeek.Monday);
```

Εφόσον οριστούν όλες οι εντολές, εκτελούμε την MySql εντολή cmd ως εξής:

```
cmd.ExecuteNonQuery();
```

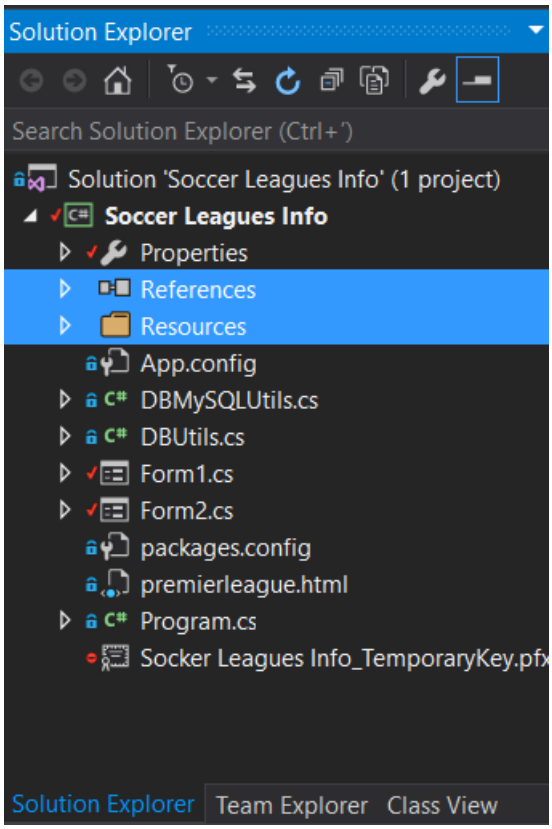
Φροντίζουμε πάντα στο τέλος, να κλείνουμε τη σύνδεση και να την καταστρέφουμε για εξοικονόμηση πόρων και bandwidth.

```
connection.Close();
connection.Dispose();
connection = null;
```

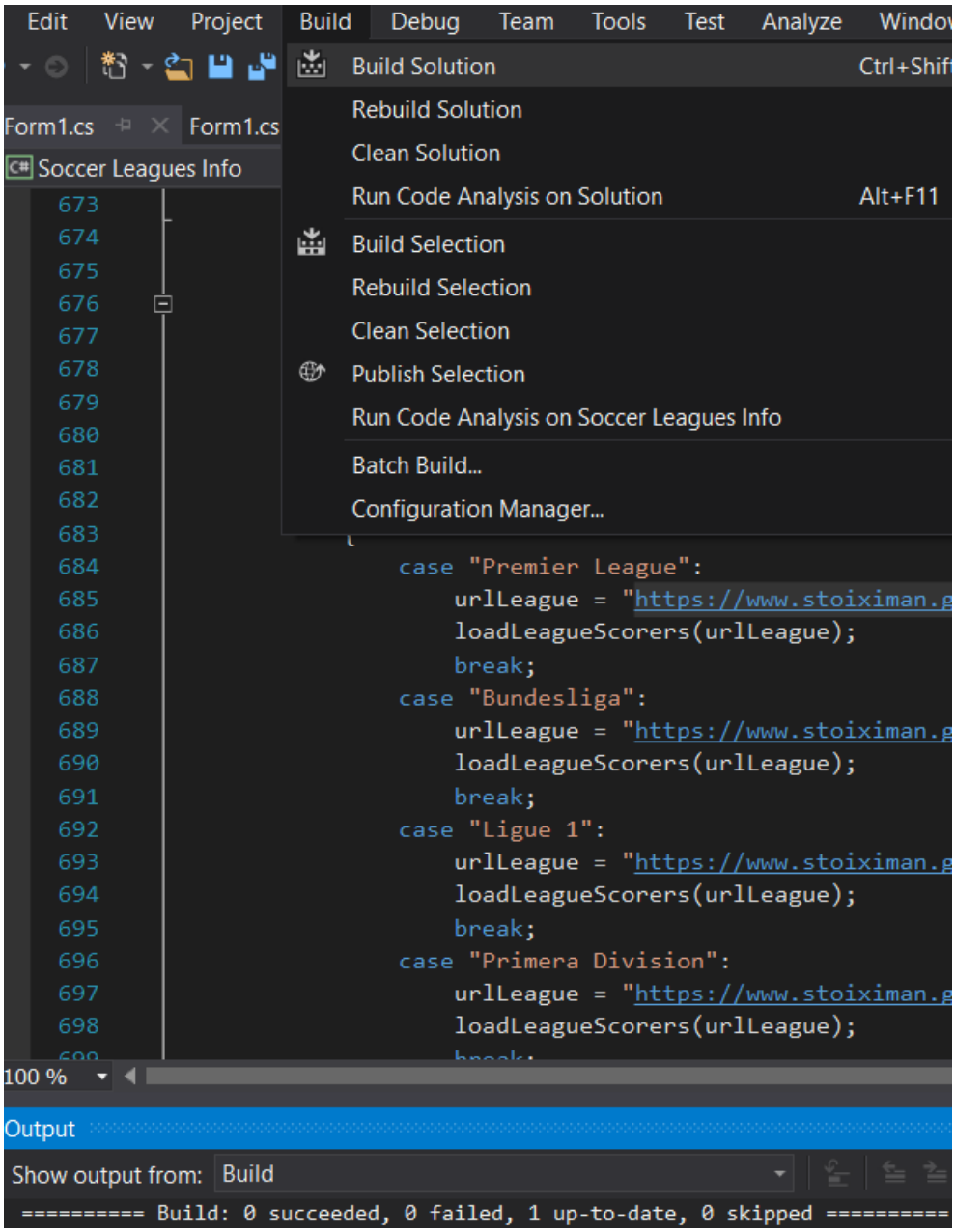
## ΔΗΜΙΟΥΡΓΙΑ ΕΚΤΕΛΕΣΙΜΟΥ ΑΡΧΕΙΟΥ

Εφόσον είναι ζητούμενο η εφαρμογή να είναι portable ώστε να μπορεί να εκτελείται σε οποιοδήποτε μηχάνημα χωρίς την ανάγκη εγκατάστασης development περιβάλλοντος (Visual Studio), θα πρέπει να δημιουργήσουμε ένα εκτελέσιμο αρχείο για την εφαρμογή.

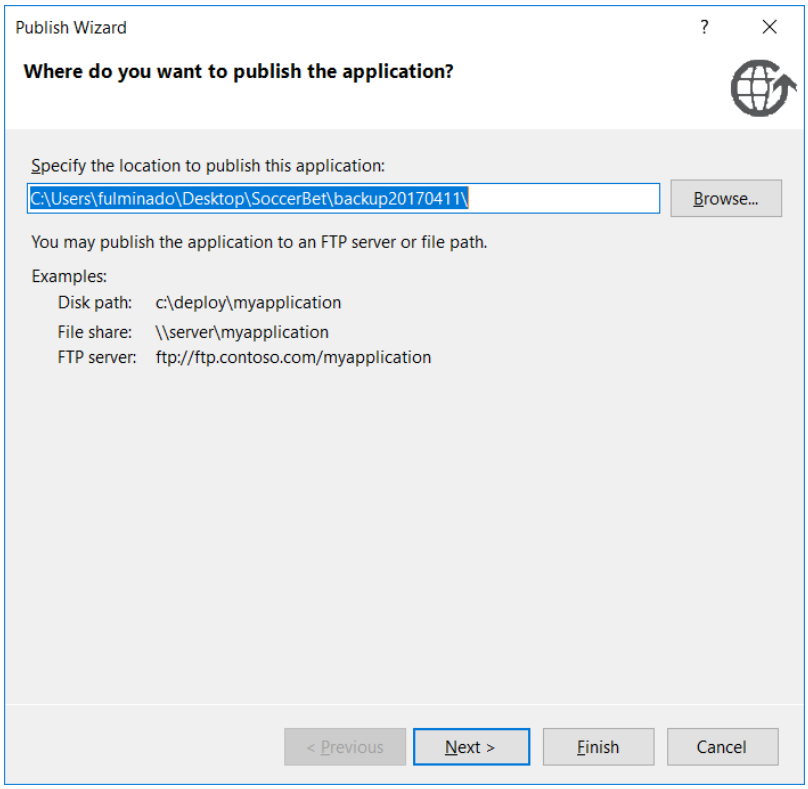
Φροντίζουμε αρχικά να εμπεριέχονται όλα τα απαραίτητα αρχεία στην εφαργμογή, οπως εικόνες και βιβλιοθήκες που δεν είναι native στο περιβάλλον του Visual Studio. Στο Solution Explorer, στα References και τα Resources, φαίνονται τα έξτρα αρχεία που θα περιληφθούν.



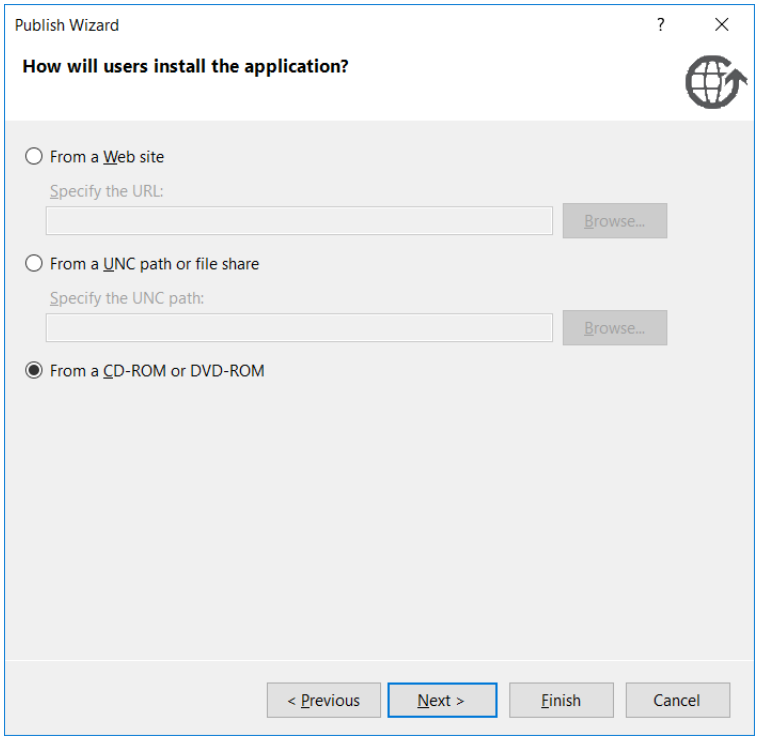

Αφού φροντίσουμε για την ενέργεια αυτή, από το μενού του Visual Studio επιλέγουμε *Build* και ακολούθως πατάμε στο *Build Solution*. Σε αυτό το σημείο θα φανούν αν υπάρχουν errors ή αρχεία που λείπουν.



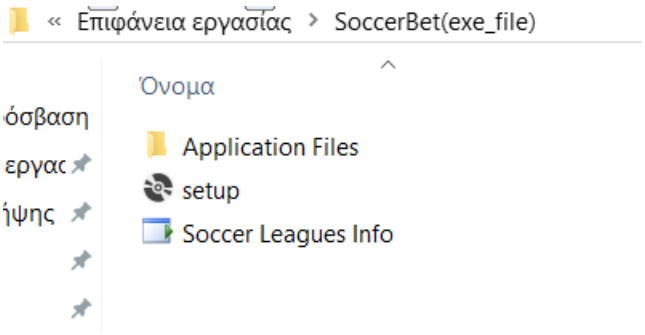

Έπειτα, από το ίδιο μενού επιλέγουμε *Publish Selection*. Ανοίγει ένα παράθυρο στο οποίο μας δίδεται επιλογή ονόματος για την εφαρμογή. Ουσιαστικά εδώ επιλέγουμε το folder όπου θα δημιουργηθούν τα αρχεία της εφαρμογής. Εισάγουμε το όνομα που επιθυμούμε και πατάμε Next.




Στην επόμενη οθόνη που έχει τίτλο “*How will users install the application?*”, επιλέγουμε το “*From a CD-ROM or DVD-ROM*” και πατάμε “*Next*”.



Στην επόμενη οθόνη πρέπει να είναι επιλεγμένο το “*The application will not check for updates*”. Έτσι, φτάνουμε στην τελευταία οθόνη όπου πατάμε το κουμπί *Finish*. Μόλις ολοκληρωθεί η διαδικασία αυτομάτως θα ανοίξει ο φάκελος που δημιουργήσαμε και θα περιέχει μέσα τα αρχεία της εφαρμογής μας.




Για να χρησιμοποιηθεί λοιπόν η εφαρμογή σε ένα οποιοδήποτε pc, θα πρέπει ο χρήστης να εκτελέσει το αρχείο setup που φαίνεται στην προηγούμενη εικόνα.

# Συμπεράσματα – Περίληψη

Έχοντας ολοκληρώσει την εκπόνηση της παρούσας εργασίας, συμπεραίνουμε ότι η εξέλιξη των γλωσσών προγραμματισμού τις καθιστά ικανές να λαμβάνουν και να διαχειρίζονται real-time δεδομένα από πολλές πηγές στο διαδίκτυο. Συγκεκριμένα, στην C# η οποία και χρησιμοποιήθηκε για την υλοποίηση αυτή, διαθέτει επαρκείς μηχανισμούς για parsing και επεξεργασία δεδομένων μέσα από ιστοσελίδες, πράγμα το οποίο δίνει σε έναν developer τη δυνατότητα να δημιουργήσει δικές του εφαρμογές που θα εξυπηρετούν διάφορες λειτουργίες.

Οι μηχανισμοί αυτοί, όπως έγινε και στην εφαρμογή που αναπτύχθηκε στα πλαίσια της παρούσας διατριβής, μπορούν να χρησιμοποιηθούν για τη λήψη συσχετιζόμενων δεδομένων από 2 ή και περισσότερες ιστοσελίδες, τα οποία στη συνέχεια μπορούν να υποστούν διαδικασίες απλής προβολής, επεξεργασίας ή και σύγκρισης. Τέτοιου τύπου εφαρμογές αποδεικνύονται πολύ χρήσιμες σε περιπτώσεις όπου θέλουμε να έχουμε μια πλήρη και σφαιρική εικόνα για δεδομένα που προέρχονται από διάφορα sites και μπορεί να περιέχουν πληροφορίες που απαιτούν ακρίβεια και αξιοπιστία (για παράδειγμα πληροφορίες καιρού, οικονομικά δεδομένα, στατιστικά δεδομένα κ.ά.)

Υπάρχουν όμως και κάποια σημαντικά μειονεκτήματα που εντοπίστηκαν στη λειτουργία της εφαρμογής μας και κατ' επέκταση στη λειτουργία τέτοιου τύπου εφαρμογών γενικότερα. Το πρώτο ζήτημα αφορά την άμεση εξάρτηση των εφαρμογών από τις ιστοσελίδες τις οποίες χρησιμοποιούν για να λάβουν τα δεδομένα. Πρακτικά αυτό σημαίνει ότι αν η ιστοσελίδα που χρησιμοποιούμε αλλάξει δομή (στον html κώδικά της) ή είναι μη διαθέσιμη για κάποιο λόγο (π.χ. server maintenance ή error) αυτομάτως η εφαρμογή καθίσταται μη λειτουργική.

Το δεύτερο ζήτημα αφορά την άμεση εξάρτηση της εφαρμογής με την ανάγκη ύπαρξης ενεργής σύνδεσης στο διαδίκτυο. Αν ένας χρήστης τρέξει την εφαρμογή μας σε μηχάνημα που δεν είναι συνδεδεμένο με το internet δεν θα μπορέσει να δει δεδομένα. Θα ήταν μάλιστα λάθος να χρησιμοποιηθεί μια λογική αποθήκευσης «ιστορικού» ή προβολής των τελευταίων ληφθέντων δεδομένων, καθώς η φιλοσοφία της εφαρμογής είναι να λαμβάνουμε real-time δεδομένα και θα ήταν λάθος να δείχνουμε παρελθοντικές τιμές από τη στιγμή ειδικά που μπορεί να αφορούν ευμετάβλητα δεδομένα που υπόκεινται σε συχνές μεταβολές.

# Βιβλιογραφία

https://en.wikipedia.org/wiki/Microsoft_Visual_Studio

https://en.wikipedia.org/wiki/MySQL

https://en.wikipedia.org/wiki/C_Sharp_(programming_language)

https://www.codeproject.com/articles/1041115/webscraping-with-csharp

https://www.dotnetperls.com/

http://articles.runtings.co.uk/2009/09/introduction-to-htmlagilitypack-library.html

http://htmlagilitypack.codeplex.com/

https://dev.mysql.com/doc/connector-net/en/connector-net-programming-connecting-open.html

https://www.codeproject.com/articles/43438/connect-c-to-mysql

http://www.introprogramming.info/wp-content/uploads/2013/07/Books/CSharpEn/Fundamentals-of-Computer-Programming-with-CSharp-Nakov-eBook-v2013.pdf

http://www.w3schools.com/tags/tag_tbody.asp

https://en.wikibooks.org/wiki/C_Sharp_Programming/Advanced

http://stackoverflow.com/questions/1361033/what-does-stathread-do

https://web.archive.org/web/20090611124052/http://www.sellsbrothers.com/askthewonk/Secure/WhatdoestheSTAThreadattri.htm

http://stackoverflow.com/questions/846994/how-to-use-html-agility-pack

http://blog.olussier.net/2010/03/30/easily-parse-html-documents-in-csharp/

# Παράρτημα – Κώδικας της εφαρμογής

## ΑΡΧΕΙΟ PROGRAM.CS

```
using System;
using System.Threading;
using System.Windows.Forms;
using MySql.Data.MySqlClient;
using Soccer_Leagues_Info.SqlConn;

namespace Socker_Leagues_Info
{
   static class Program
   {
      public static Soccer_Leagues_Info.Splashscreen splashForm = null;
      /// <summary>
      /// The main entry point for the application.
      /// </summary>
      [STAThread]
      static void Main()
      {
         Console.WriteLine("Getting Connection ...");
         MySqlConnection conn = DBUtils.GetDBConnection();

         try
         {
            Console.WriteLine("Openning Connection ...");

            conn.Open();

            Console.WriteLine("Connection successful!");
         }
         catch (Exception e)
         {
            Console.WriteLine("Error: " + e.Message);
         }

         Console.Read();
         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         Thread splashThread = new Thread(new ThreadStart(
                  delegate
                  {
                     splashForm = new Soccer_Leagues_Info.Splashscreen();
                     Application.Run(splashForm);
                  }
                  ));

         splashThread.SetApartmentState(ApartmentState.STA);
         splashThread.Start();

         Form1 mainForm = new Form1();
```

```
            mainForm.Load += new EventHandler(mainForm_Load);
            Application.Run(mainForm);

        }

        static void mainForm_Load(object sender, EventArgs e)
        {
            //close splash
            if (splashForm == null)
            {
                return;
            }

            splashForm.Invoke(new Action(splashForm.Close));
            splashForm.Dispose();
            splashForm = null;
        }
    }
}
```

## ΑΡΧΕΙΟ DBUTILS.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using MySql.Data.MySqlClient;

namespace Soccer_Leagues_Info.SqlConn
{
    class DBUtils
    {
        public static MySqlConnection GetDBConnection()
        {
            string host = "198.46.81.30";
            int port = 3306;
            string database = "csshou5_soccer";
            string username = "csshou5_soccer";
            string password = "QweRTy!2#4%$";

            return DBMySQLUtils.GetDBConnection(host, port, database, username, password);
        }
    }
}
```

## ΑΡΧΕΙΟ DBMYSQLUTILS.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using MySql.Data.MySqlClient;

namespace Soccer_Leagues_Info.SqlConn
{
   class DBMySQLUtils
   {
      public static MySqlConnection
       GetDBConnection(string host, int port, string database, string username, string password)
      {
         // Connection String.
         String connString = "Server=" + host + ";Database=" + database
            + ";port=" + port + ";User Id=" + username + ";password=" + password + ";charset =
utf8;";

         MySqlConnection conn = new MySqlConnection(connString);

         return conn;
      }
   }
}
```

## ΑΡΧΕΙΟ FORM2.CS

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Soccer_Leagues_Info
{
    public partial class Splashscreen : Form
    {
        public Splashscreen()
        {
            InitializeComponent();
        }
    }
}
```

## ΑΡΧΕΙΟ FORM1.CS

```
using System;
using System.Windows.Forms;
using HtmlAgilityPack;
using System.Data;
using System.Linq;
using System.Threading;
using Awesomium.Core;
using MySql.Data.MySqlClient;
using Soccer_Leagues_Info.SqlConn;
using System.Collections.Generic;
using System.Globalization;
using System.Text.RegularExpressions;

namespace Socker_Leagues_Info
{
   public partial class Form1 : Form
   {

      public string HtmlSource1;
      public bool isPageLoaded = false;
      public Uri urlFirstSite = new Uri("https://www.betshop.gr/sports/l/1");
      public string urlSecondSite = "https://www.stoiximan.gr/league/Soccer-FOOT/Premier-
League-England-1";
      public Uri leagueScore = new Uri("http://www.escore.gr/football/england/premier-
league/standings/");
      public DataGridView dgvScore, dgvLeague;
      public int countLoads = 0;
      public int countLoads1 = 0;
      public int leagueID = 1;
      public bool tab1Loaded = false;
      public bool tab2Loaded = false;
      public bool tab3Loaded = false;
      public bool tab4Loaded = false;
      public bool tab5Loaded = false;
      public bool tab6Loaded = false;
      public bool tab7Loaded = false;
      public bool tab8Loaded = false;
      public bool tab9Loaded = false;
      public List<HtmlNode> arrMatches = new List<HtmlNode>();


      public Form1()
      {
         InitializeComponent();
         Thread.Sleep(2000); //
         dgvScore = dataGridView1;
         dgvLeague = dataGridView3;
      }

      private void Form1_Load(object sender, EventArgs e)
      {
         webControl1.Source = leagueScore;
```

```
            webControl1.LoadingFrameComplete += OnLoadingFrameComplete;
            webControl2.Source = urlFirstSite;
            webControl2.LoadingFrameComplete += OnLoadingFrameComplete1;

            tabControl1.SelectedIndexChanged += Tabs_SelectedIndexChanged;

        }


        private void OnLoadingFrameComplete(Object sender, FrameEventArgs e)
        {
            if (e.IsMainFrame)
            {
                countLoads++;
                if (countLoads < 2)
                {
                    string source1 =
webControl1.ExecuteJavascriptWithResult("document.body.innerHTML");
                    getLeagueScore(source1);
                }
            }
        }

        private void OnLoadingFrameComplete1(Object sender, FrameEventArgs e)
        {
            if (e.IsMainFrame)
            {
                countLoads1++;
                if (countLoads1 < 2)
                {
                    HtmlSource1 =
webControl2.ExecuteJavascriptWithResult("document.body.innerHTML");
                    checkDatagridHtml();
                }

            }
        }


        void Tabs_SelectedIndexChanged(object sender, EventArgs e)
        {
            string selectedTab = tabControl1.SelectedTab.Text;

            switch (selectedTab)
            {
                case "Bundesliga":
                    urlFirstSite = new Uri("https://www.betshop.gr/sports/l/3");
                    urlSecondSite = "https://www.stoiximan.gr/league/Soccer-FOOT/Bundesliga-
Germany-216";
                    //leagueScore = new
Uri("http://www.escore.gr/football/germany/bundesliga/standings/");
                    leagueScore = new
Uri("http://www.escore.gr/football/germany/bundesliga/standings/?t=xQV0Y9j3&ts=ljIBgFCg");
                    dgvScore = dataGridView2;
                    dgvLeague = dataGridView4;
                    if (tab1Loaded == false)
```

```
                {
                    leagueID = 2;
                    newPage();
                    tab1Loaded = true;
                    countLoads = 0;
                    countLoads1 = 0;
                }else
                {
                    countLoads = 1;
                    countLoads1 = 1;
                }

                break;
            case "Ligue 1":
                urlFirstSite = new Uri("https://www.betshop.gr/sports/l/46");
                urlSecondSite = "https://www.stoiximan.gr/league/Soccer-FOOT/Ligue-1-France-
215";
                //leagueScore = new Uri("http://www.escore.gr/football/france/ligue-1/standings/");
                leagueScore = new Uri("http://www.escore.gr/football/france/ligue-
1/standings/?t=OO2KUIR8&ts=pSvKVQK1");
                dgvScore = dataGridView5;
                dgvLeague = dataGridView6;
                if (tab2Loaded == false)
                {
                    leagueID = 3;
                    newPage();
                    tab2Loaded = true;
                    countLoads = 0;
                    countLoads1 = 0;
                }
                else
                {
                    countLoads = 1;
                    countLoads1 = 1;
                }
                break;
            case "Primera Division":
                urlFirstSite = new Uri("https://www.betshop.gr/sports/l/5");
                urlSecondSite = "https://www.stoiximan.gr/league/Soccer-FOOT/Primera-Division-
Spain-5";
                leagueScore = new Uri("http://www.escore.gr/football/spain/primera-
division/standings/");
                dgvScore = dataGridView7;
                dgvLeague = dataGridView8;
                if (tab3Loaded == false)
                {
                    leagueID = 4;
                    newPage();
                    tab3Loaded = true;
                    countLoads = 0;
                    countLoads1 = 0;
                }
                else
                {
                    countLoads = 1;
                    countLoads1 = 1;
```

```
                }
                break;
            case "Serie A":
                urlFirstSite = new Uri("https://www.betshop.gr/sports/l/38");
                urlSecondSite = "https://www.stoiximan.gr/league/Soccer-FOOT/Serie-A-Italy-
1635";
                leagueScore = new Uri("http://www.escore.gr/football/italy/serie-a/standings/");
                dgvScore = dataGridView9;
                dgvLeague = dataGridView10;
                if (tab4Loaded == false)
                {
                    leagueID = 5;
                    newPage();
                    tab4Loaded = true;
                    countLoads = 0;
                    countLoads1 = 0;
                }
                else
                {
                    countLoads = 1;
                    countLoads1 = 1;
                }
                break;
            case "Super League":
                urlFirstSite = new Uri("https://www.betshop.gr/sports/l/330");
                urlSecondSite = "https://www.stoiximan.gr/league/Soccer-FOOT/Super-League-
Greece-1636";
                leagueScore = new Uri("http://www.escore.gr/football/greece/super-
league/standings/");
                dgvScore = dataGridView11;
                dgvLeague = dataGridView12;
                if (tab5Loaded == false)
                {
                    leagueID = 6;
                    newPage();
                    tab5Loaded = true;
                    countLoads = 0;
                    countLoads1 = 0;
                }
                else
                {
                    countLoads = 1;
                    countLoads1 = 1;
                }
                break;
            case "England Championship":
                urlFirstSite = new Uri("https://www.betshop.gr/sports/l/59");
                urlSecondSite = "https://www.stoiximan.gr/league/Soccer-FOOT/Championship-
England-2r";
                //leagueScore = new
Uri("http://www.escore.gr/football/england/championship/standings/");
                leagueScore = new
Uri("http://www.escore.gr/football/england/championship/standings/?t=4rGIVFO5&ts=OYqFNvN
n");
                dgvScore = dataGridView14;
                dgvLeague = dataGridView15;
```

```
                if (tab6Loaded == false)
                {
                    leagueID = 7;
                    newPage();
                    tab6Loaded = true;
                    countLoads = 0;
                    countLoads1 = 0;
                }
                else
                {
                    countLoads = 1;
                    countLoads1 = 1;
                }
                break;
            case "Segunda Division":
                urlFirstSite = new Uri("https://www.betshop.gr/sports/l/336");
                urlSecondSite = "https://www.stoiximan.gr/league/Soccer-FOOT/Segunda-Division-
Spain-10000";
                leagueScore = new Uri("http://www.escore.gr/football/spain/laliga2/standings/");
                dgvScore = dataGridView16;
                dgvLeague = dataGridView17;
                if (tab7Loaded == false)
                {
                    leagueID = 8;
                    newPage();
                    tab7Loaded = true;
                    countLoads = 0;
                    countLoads1 = 0;
                }
                else
                {
                    countLoads = 1;
                    countLoads1 = 1;
                }
                break;
            case "History":

                if (tab9Loaded == false)
                {
                    leagueID = 9;
                    tab9Loaded = true;
                    countLoads = 0;
                    countLoads1 = 0;
                }
                else
                {
                    countLoads = 1;
                    countLoads1 = 1;
                }
                break;
            default:
                break;
        }
    }
```

```
public void newPage(bool runScores=true)
{
   webControl1.Source = leagueScore;
   webControl1.LoadingFrameComplete += OnLoadingFrameComplete;
   if (runScores == true)
   {
      webControl2.Source = urlFirstSite;
      webControl2.LoadingFrameComplete += OnLoadingFrameComplete1;
   }
}


private void checkDatagridHtml()
{
   // get league stoiximan -- start
   string urlBet = urlSecondSite;
   HtmlWeb HtmlWEB1 = new HtmlWeb();
   HtmlAgilityPack.HtmlDocument doc1 = HtmlWEB1.Load(urlBet);
   doc1 = HtmlWEB1.Load(urlBet);

   DataTable betTable1 = new DataTable();

   betTable1.Columns.Add("Ομάδες", typeof(String));
   betTable1.Columns.Add("1", typeof(String));
   betTable1.Columns.Add("X", typeof(String));
   betTable1.Columns.Add("2", typeof(String));
   betTable1.Columns.Add("O/U 2,5", typeof(String));
   betTable1.Columns.Add("GG/NG", typeof(String));
   betTable1.Columns.Add("Κάτι", typeof(String));

   HtmlNode sBet1 = doc1.DocumentNode.SelectSingleNode("//table");
   if (sBet1 != null)
   {

      var nodes1 = doc1.DocumentNode.SelectNodes("//table/tbody/tr");

      try
      {
         var rows1 = nodes1.Select(tr => tr
            .Elements("td")
            .Select(td => StringReplaceChar(td.InnerText.Trim()))
            .ToArray());
         foreach (var row1 in rows1)
         {
            betTable1.Rows.Add(row1);
         }
      }
      catch
      {
         MessageBox.Show("Τέλος αγωνιστικών! Δεν υπάρχουν στοιχηματικά δεδομένα για
το Πρόγραμμα!");

      }

   }
```

```
        // get league stoiximan -- end

        // get league betshop
        DataTable betTable2 = new DataTable();
        betTable2.Columns.Add("Column1", typeof(DateTime));
        betTable2.Columns.Add("Column2", typeof(String));
        betTable2.Columns.Add("Column3", typeof(String));
        betTable2.Columns.Add("Column4", typeof(Decimal));
        betTable2.Columns.Add("Column5", typeof(Decimal));
        betTable2.Columns.Add("Column6", typeof(Decimal));
        betTable2.Columns.Add("Column7", typeof(Decimal));
        betTable2.Columns.Add("Column8", typeof(Decimal));
        betTable2.Columns.Add("Column9", typeof(Decimal));
        betTable2.Columns.Add("Column10", typeof(Decimal));
        betTable2.Columns.Add("Column11", typeof(Decimal));
        betTable2.Columns.Add("Column12", typeof(Decimal));
        betTable2.Columns.Add("Column13", typeof(Decimal));
        betTable2.Columns.Add("Column14", typeof(Decimal));
        betTable2.Columns.Add("Column15", typeof(Decimal));
        DataGridViewRow row = new DataGridViewRow();
        HtmlAgilityPack.HtmlDocument doc = new HtmlAgilityPack.HtmlDocument();


        doc.LoadHtml(HtmlSource1);
        var nodes = doc.DocumentNode.SelectNodes("//table[contains(@class,'sports-
table')][1]/tbody/tr");

        if (nodes != null)
        {
            int i = 0;
            float odd11, odd1X, odd12, odd21, odd2X, odd22, mo1, moX, mo2, moSumPercent,
mo1New, mo2New, moXNew;
            odd11 = odd1X = odd12 = odd21 = odd2X = odd22 = mo1 = moX = mo2 = mo1New =
moXNew = mo2New = 0.00f;
            float moFactor = 0.116f;
            var getDateTime = "";
            foreach (var node in nodes)
            {
                DataRow dr = betTable2.NewRow();
                var matchDate = node.SelectSingleNode(".//td/div[@class='datetime']/text()");
                int addtoDataGrid = GetNextWeekday(matchDate);
                if (addtoDataGrid == 99)
                {
                    if (matchDate != null)
                    {
                        var getDate = matchDate.InnerText.Trim() + "/" +
DateTime.Now.Year.ToString();
                        var getTime =
node.SelectSingleNode(".//td/div[@class='datetime']/span[@class='time']/text()");
                        getDateTime = getDate + " " + getTime.InnerText.Trim();
                        dr[0] = Convert.ToDateTime(getDateTime);
                    }
                    else
                    {
                        dr[0] = null;
                    }
```

```
                var homeTeam =
node.SelectSingleNode(".//td/div[@class='matches']/a/span[1]");
                if (homeTeam != null)
                {
                    dr[1] = homeTeam.InnerText;
                }
                else
                {
                    dr[1] = "hometeam";
                }
                var awayTeam =
node.SelectSingleNode(".//td/div[@class='matches']/a/span[2]");
                if (awayTeam != null)
                {
                    dr[2] = awayTeam.InnerText;
                }
                else
                {
                    dr[2] = "visitor";
                }
                var odd1 = node.SelectSingleNode(".//td[@class='odds']/span[1]");
                if (odd1 != null)
                {
                    var currText = StringReplaceChar(odd1.InnerText);
                    odd11 = Convert.ToSingle(currText);
                }
                var oddX = node.SelectSingleNode(".//td[@class='odds']/span[2]");
                if (oddX != null)
                {
                    var currText = StringReplaceChar(oddX.InnerText);
                    odd1X = Convert.ToSingle(currText);
                }
                var odd2 = node.SelectSingleNode(".//td[@class='odds']/span[3]");
                if (odd2 != null)
                {
                    var currText = StringReplaceChar(odd2.InnerText);
                    odd12 = Convert.ToSingle(currText);
                }

                String currTeams = dr[1] + " - " + dr[2];
                int currBetTable1Row = getRespectiveRow(betTable1, currTeams);
                if (currBetTable1Row != 998)
                {
                    if (betTable1.Rows[currBetTable1Row].Field<string>("1") != null)
                    {
                        odd21 =
Convert.ToSingle(betTable1.Rows[currBetTable1Row].Field<string>("1"));
                    }
                    if (betTable1.Rows[currBetTable1Row].Field<string>("X") != null)
                    {
                        odd2X =
Convert.ToSingle(betTable1.Rows[currBetTable1Row].Field<string>("X"));
                    }
                    if (betTable1.Rows[currBetTable1Row].Field<string>("2") != null)
                    {
```

```
                    odd22 =
Convert.ToSingle(betTable1.Rows[currBetTable1Row].Field<string>("2"));
                }
            }

            dr[3] = odd11;
            dr[4] = odd1X;
            dr[5] = odd12;
            dr[6] = odd21;
            dr[7] = odd2X;
            dr[8] = odd22;


            mo1 = (odd11 + odd21) / 2;
            moX = (odd1X + odd2X) / 2;
            mo2 = (odd12 + odd22) / 2;
            moSumPercent = (mo1 + mo2 + moX) / 100;
            mo1New = moFactor * mo1 / moSumPercent;
            moXNew = moFactor * moX / moSumPercent;
            mo2New = moFactor * mo2 / moSumPercent;


            dr[9] = mo1;
            dr[10] = moX;
            dr[11] = mo2;
            dr[12] = mo1New;
            dr[13] = moXNew;
            dr[14] = mo2New;

            Console.WriteLine(dr[0] + "," + dr[1] + "," + dr[2] + "," + dr[3] + "," + dr[4] + "," +
dr[5] + "," + dr[6] + "," + dr[7] + "," + dr[8] + "," + dr[9] + "," + dr[10] + "," + dr[11] + "," + dr[12] +
"," + dr[13] + "," + dr[14]);

            if (this.dgvLeague.InvokeRequired)
            {
                Invoke((MethodInvoker)(() => dgvLeague.Rows.Add(getDateTime,
homeTeam.InnerText, awayTeam.InnerText, odd1.InnerText, oddX.InnerText, odd2.InnerText,
betTable1.Rows[i].Field<string>("1"), betTable1.Rows[i].Field<string>("X"),
betTable1.Rows[i].Field<string>("2"), mo1, moX, mo2, mo1New, moXNew, mo2New)));
            }
            else
            {
                dgvLeague.Rows.Add(getDateTime, homeTeam.InnerText,
awayTeam.InnerText, odd1.InnerText, oddX.InnerText, odd2.InnerText,
betTable1.Rows[i].Field<string>("1"), betTable1.Rows[i].Field<string>("X"),
betTable1.Rows[i].Field<string>("2"), mo1, moX, mo2, mo1New, moXNew, mo2New);
            }

            i++;
            betTable2.Rows.Add(dr);
        }

    }
}
betTable1.Clear();
betTable2.Clear();
```

```
        isPageLoaded = true;

    }

    private string StringReplaceChar(string x)
    {
        if (x.Contains("."))
        x = x.Replace(".", ",");
    return x;

    }

    private void getLeagueScore(string uriHtml)
    {

        HtmlAgilityPack.HtmlDocument doc = new HtmlAgilityPack.HtmlDocument();

        doc.LoadHtml(uriHtml);
        DataTable pointsTabletable = new DataTable();

        pointsTabletable.Columns.Add("Column1", typeof(String));
        pointsTabletable.Columns.Add("Column2", typeof(String));
        pointsTabletable.Columns.Add("Column3", typeof(String));
        pointsTabletable.Columns.Add("Column4", typeof(String));
        var nodes = doc.DocumentNode.SelectNodes("//table[contains(@class,'stats-
main')]/tbody/tr");

        if (nodes != null)
        {
            int i = 0;
            int roundIndex = 0;
            foreach (var node in nodes)
            {
                DataRow dr = pointsTabletable.NewRow();
                var teamName =
node.SelectSingleNode(".//td[contains(@class,'participant_name')]/span[@class='team_name_s
pan']");
                if (teamName != null)
                {
                    dr[0] = i+1;
                    dr[1] = teamName.InnerText;
                }
                var teamPoints = node.SelectSingleNode(".//td[contains(@class,'goals')][2]");
                if (teamPoints != null)
                {
                    dr[2] = teamPoints.InnerText;
                }

                var currRoundText =
node.SelectSingleNode(".//td[contains(@class,'matches_played')]"); ;
                Console.WriteLine(currRoundText.InnerText);
                if (currRoundText != null)
                {
                    int currRound = int.Parse(currRoundText.InnerText);
                    if (currRound > roundIndex)
                    {
```

```
                    roundIndex = currRound;
                }
            }

            var currGroup =
node.SelectSingleNode("../../..//thead/tr/th[contains(@class,'participant_name')]/a/span[@class=
'txt']");
            dr[3] = "N/A";
            try
            {
                if (currGroup.InnerText != "Ομάδα")
                {
                    var theadNode = node.ParentNode.PreviousSibling;
                    var groupName =
theadNode.SelectSingleNode(".//tr/th[contains(@class,'participant_name')]/a/span[@class='txt']"
);
                }
            }
            catch (Exception e)
                {
                  Console.WriteLine("Error: " + e);
                  Console.WriteLine(e.StackTrace);
                }
            if (this.dgvScore.InvokeRequired)
            {
                Invoke((MethodInvoker)(() => dgvScore.Rows.Add(i+1,teamName.InnerText,
teamPoints.InnerText) ));
            }
            else
            {
                dgvScore.Rows.Add(i+1, teamName.InnerText, teamPoints.InnerText);

                Console.WriteLine("in else");
            }
            i++;
            pointsTabletable.Rows.Add(dr);

        }
        if (checkRoundValuesExist(leagueID, roundIndex) == true)
        {

        }

    }
}


private void insertDataInScoreTable(DataTable currArray, int roundNumber )
{
    // Get Connection
    MySqlConnection connection = DBUtils.GetDBConnection();
    connection.Open();
    try
    {
        // Insert statement.
```

```
            string sql = "Insert into sli_score(lid, round, week, year, position, name, points,
leaguegroup) values ( @lid, @round, @week, @year, @position, @name, @points, @group)";

            MySqlCommand cmd = new MySqlCommand();
            cmd.Connection = connection;
            cmd.CommandText = sql;
            cmd.Prepare();
            // Create Parameter.
            MySqlParameter lidParam = new MySqlParameter("@lid", MySqlDbType.Int16);
            MySqlParameter roundParam = new MySqlParameter("@round",
MySqlDbType.Int16);
            MySqlParameter weekParam = new MySqlParameter("@week", MySqlDbType.Int16);
            MySqlParameter yearParam = new MySqlParameter("@year", MySqlDbType.Int16);
            MySqlParameter positionParam = new MySqlParameter("@position",
MySqlDbType.Int16);
            MySqlParameter nameParam = new MySqlParameter("@name",
MySqlDbType.VarChar, 255);
            MySqlParameter pointsParam = new MySqlParameter("@points",
MySqlDbType.Int16);
            MySqlParameter groupParam = new MySqlParameter("@group",
MySqlDbType.VarChar, 255);

            cmd.Parameters.Add(lidParam);
            cmd.Parameters.Add(roundParam);
            cmd.Parameters.Add(weekParam);
            cmd.Parameters.Add(yearParam);
            cmd.Parameters.Add(positionParam);
            cmd.Parameters.Add(nameParam);
            cmd.Parameters.Add(pointsParam);
            cmd.Parameters.Add(groupParam);


            DateTime thisDay = DateTime.Today;
            CultureInfo ciCurr = CultureInfo.CurrentCulture;

            int yearNum = ciCurr.Calendar.GetYear(thisDay);
            int weekNum = ciCurr.Calendar.GetWeekOfYear(thisDay,
CalendarWeekRule.FirstFourDayWeek, DayOfWeek.Monday);

            foreach (DataRow dr in currArray.Rows)
            {

               cmd.Parameters["@lid"].Value = Convert.ToInt16(leagueID);
               cmd.Parameters["@round"].Value = Convert.ToInt16(roundNumber);
               cmd.Parameters["@week"].Value = Convert.ToInt16(weekNum);
               cmd.Parameters["@year"].Value = Convert.ToInt16(yearNum);
               cmd.Parameters["@position"].Value = Convert.ToInt16(dr[0]);
               cmd.Parameters["@name"].Value = Convert.ToString(dr[1]).ToString();
               cmd.Parameters["@points"].Value = Convert.ToInt32(dr[2]);
               cmd.Parameters["@group"].Value = Convert.ToString(dr[3]).ToString();
               cmd.ExecuteNonQuery();
            }

         }
         catch (Exception e)
         {
```

```
            Console.WriteLine("Error: " + e);
            Console.WriteLine(e.StackTrace);
        }
        finally
        {
            // Close connection
            connection.Close();
            // Freeing Resources
            connection.Dispose();
            connection = null;
        }
    }


    private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
    {
        string selectedItem = comboBox1.Text;
        string urlLeague = "";

        switch (selectedItem)
        {
            case "Premier League":
                urlLeague = "https://www.stoiximan.gr/league/Soccer-FOOT/Premier-League-
England-1";
                loadLeagueScorers(urlLeague);
                break;
            case "Bundesliga":
                urlLeague = "https://www.stoiximan.gr/league/Soccer-FOOT/Bundesliga-Germany-
216";
                loadLeagueScorers(urlLeague);
                break;
            case "Ligue 1":
                urlLeague = "https://www.stoiximan.gr/league/Soccer-FOOT/Ligue-1-France-215";
                loadLeagueScorers(urlLeague);
                break;
            case "Primera Division":
                urlLeague = "https://www.stoiximan.gr/league/Soccer-FOOT/Primera-Division-
Spain-5";
                loadLeagueScorers(urlLeague);
                break;
            case "Serie A":
                urlLeague = "https://www.stoiximan.gr/league/Soccer-FOOT/Serie-A-Italy-1635";
                loadLeagueScorers(urlLeague);
                break;
            case "Super League":
                urlLeague = "https://www.stoiximan.gr/league/Soccer-FOOT/Super-League-
Greece-1636";
                loadLeagueScorers(urlLeague);
                break;
            case "England Championship":
                urlLeague = "https://www.stoiximan.gr/league/Soccer-FOOT/Championship-
England-2r";
                loadLeagueScorers(urlLeague);
                break;
            case "Segunda Division":
```

```
                urlLeague = "https://www.stoiximan.gr/league/Soccer-FOOT/Europa-League-
Matches-Europa-League-182761";
                loadLeagueScorers(urlLeague);
                break;
            }
        }

        private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
        {
            string selectedItem = comboBox2.Text;
            foreach (var currDoc in arrMatches)
            {
                var matchName =
currDoc.SelectSingleNode(".//div[@class='m9']/a[@class='ud']/text()");

                if (matchName.InnerHtml == selectedItem)
                {
                    var tableScorer = currDoc.SelectSingleNode(".//div[contains(@class,'js-market-
body')]/table[@class='tx']/tbody");
                    dataGridView13.Rows.Clear();
                    dgvPlayersOdss.Rows.Clear();
                    if (tableScorer != null)
                    {
                        var tableScorerRows = tableScorer.SelectNodes(".//tr");
                        foreach (var singleRow in tableScorerRows)
                        {
                            var playerName = singleRow.SelectSingleNode(".//td[1]/div[@class='ud']");
                            var playerOdds =
singleRow.SelectSingleNode(".//td[2]/div[@class='tb']/a/text()");
                            if ((playerName != null) && (playerOdds != null))
                            {
                                dataGridView13.Rows.Add(playerName.InnerHtml.Trim(),
playerOdds.InnerHtml.Trim());
                            }
                        }
                    }
                    break;
                }
            }
        }

        private void loadLeagueScorers(string urlLeague)
        {
            string urlPlayers = "";
            HtmlWeb HtmlWEB1 = new HtmlWeb();
            HtmlAgilityPack.HtmlDocument docLeague = HtmlWEB1.Load(urlLeague);
            docLeague = HtmlWEB1.Load(urlLeague);

            HtmlNode sBet1 = docLeague.DocumentNode.SelectSingleNode("//div[@id='js-layout-
mainholder']/div[@class='mc']/ul[contains(@class, 'tw')]");
            if (sBet1 != null)
            {

                var nodes = sBet1.SelectNodes(".//li/a");

                foreach (var row in nodes)
```

```
            {

                if (row.InnerHtml == "Παίκτες")
                {
                    string PlayersHref = row.Attributes["href"].Value;
                    urlPlayers = ("https://www.stoiximan.gr" + PlayersHref).Replace("&amp;", "&");
                }

            }
            if (urlPlayers != "")
            {

                HtmlWeb HtmlWEB2 = new HtmlWeb();
                HtmlAgilityPack.HtmlDocument docPlayers = HtmlWEB2.Load(urlPlayers);
                docPlayers = HtmlWEB2.Load(urlPlayers);

                HtmlNode sBet2 = docPlayers.DocumentNode.SelectSingleNode("//div[@class='js-
league-body']");
                if (sBet2 != null)
                {
                    var nodes1 = sBet2.SelectNodes(".//div[@class='u3']");
                    comboBox2.Items.Clear();
                    if (nodes1 != null)
                    {
                        foreach (var node1 in nodes1)
                        {
                            arrMatches.Add(node1);
                            var matchName =
node1.SelectSingleNode(".//div[@class='m9']/a[@class='ud']/text()");
                            comboBox2.Items.Add(matchName.InnerHtml);

                        }
                    }

                }
            }

        }
    }

    private void btnCalculateMoPlayers_Click(object sender, EventArgs e)
    {
        CultureInfo customCulture =
(System.Globalization.CultureInfo)System.Threading.Thread.CurrentThread.CurrentCulture.Clo
ne();
        customCulture.NumberFormat.NumberDecimalSeparator = ".";

        Thread.CurrentThread.CurrentCulture = customCulture;
        decimal extraPlayer1, extraPlayer2;
        extraPlayer1 = extraPlayer2 = 0.00M;
        extraPlayer1 = Convert.ToDecimal(extraPlayer1Odds.Text);
        extraPlayer2 = Convert.ToDecimal(extraPlayer2Odds.Text);

        decimal moSum = 0.00M;
        int i = 2;
        foreach (DataGridViewRow row in dgvPlayersOdss.Rows)
```

```
        {
            var currOdds = Convert.ToDecimal(row.Cells[1].Value);
            moSum = moSum + currOdds;
            i = i + 1;
        }
        moSum = moSum + extraPlayer1 + extraPlayer2;
        moSum = decimal.Divide(100,  moSum) ;
        txtMoPlayers.Text = moSum.ToString();

    }

    private bool checkRoundValuesExist(int league, int round)
    {
        // Get Connection
        MySqlConnection connection = DBUtils.GetDBConnection();
        connection.Open();
        try
        {
            // Insert statement.
            string sql = "SELECT * FROM sli_score WHERE lid=@lid AND round=@round AND
year=@year";

            MySqlCommand cmd = new MySqlCommand();
            cmd.Connection = connection;
            cmd.CommandText = sql;
            cmd.Prepare();
            // Create Parameter.
            MySqlParameter lidParam = new MySqlParameter("@lid", MySqlDbType.Int16);
            MySqlParameter roundParam = new MySqlParameter("@round",
MySqlDbType.Int16);
            MySqlParameter yearParam = new MySqlParameter("@year", MySqlDbType.Int16);


            cmd.Parameters.Add(lidParam);
            cmd.Parameters.Add(roundParam);
            cmd.Parameters.Add(yearParam);

            DateTime thisDay = DateTime.Today;
            CultureInfo ciCurr = CultureInfo.CurrentCulture;
            int yearNum = ciCurr.Calendar.GetYear(thisDay);


            cmd.Parameters["@lid"].Value = Convert.ToInt16(league);
            cmd.Parameters["@round"].Value = Convert.ToInt16(round);
            cmd.Parameters["@year"].Value = Convert.ToInt16(yearNum);
            // string rowCount;
            var firstColumn = cmd.ExecuteScalar();

            if (firstColumn != null)
            {
                return false;
            }
            else
            {
                return true;
            }
```

```
        }
        catch (Exception e)
        {
            MessageBox.Show(e.Message);
            return false;
        }
        finally
        {
            // Close connection
            connection.Close();
            // Freeing Resources
            connection.Dispose();
            connection = null;
        }
    }

    private void loadHistory()
    {

        cmbRounds.Items.Clear();
        cmbRounds.SelectedIndex = -1;

        if (cmbleague.SelectedItem != null)
            {
                MySqlConnection connection = DBUtils.GetDBConnection();
                try
                {
                        connection.Open();
                        DateTime thisDay = DateTime.Today;
                        CultureInfo ciCurr = CultureInfo.CurrentCulture;
                        int yearNum = ciCurr.Calendar.GetYear(thisDay);
                        string leagueValue = null;
                        Int16 leagueNumber = 0;
                        leagueValue = cmbleague.SelectedItem.ToString();
                        leagueNumber = nameToId(leagueValue);

                        string sql = "Select DISTINCT round, week, year from sli_score WHERE
lid=@lid ORDER BY week";
                        MySqlCommand cmd = new MySqlCommand();
                        cmd.Connection = connection;
                        cmd.CommandText = sql;
                        cmd.Prepare();
                        MySqlParameter lidParam = new MySqlParameter("@lid",
MySqlDbType.Int16);
                        cmd.Parameters.Add(lidParam);
                        cmd.Parameters["@lid"].Value = Convert.ToInt16(leagueNumber);

                        MySqlDataReader reader = cmd.ExecuteReader();

                        while (reader.Read())
                        {
                            var currRound = reader.GetInt32("round");
                            var currYear = reader.GetInt32("year");
                            var currWeek = reader.GetInt32("week");
```

```
                    string sRound = currRound + "η Αγωνιστική";
                    ComboboxItem item = new ComboboxItem();
                    item.Text = sRound;
                    item.Value = currWeek + "-" + currYear + '-' + currRound;

                    cmbRounds.Items.Add(item);
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
            finally
            {
                // Close connection
                connection.Close();
                // Freeing Resources
                connection.Dispose();
                connection = null;
            }
        }

    }
    public class ComboboxItem
    {
        public string Text { get; set; }
        public object Value { get; set; }

        public override string ToString()
        {
            return Text;
        }
    }

    private string idToname(Int32 leagueId)
    {
        string leagueName = null;
        switch (leagueId)
        {
            case 1:
                leagueName = "Premiere League";
                break;
            case 2:
                leagueName = "Bundesliga";
                break;
            case 3:
                leagueName = "Ligue 1";
                break;
            case 4:
                leagueName = "Primera Division";
                break;
            case 5:
                leagueName = "Serie A";
                break;
            case 6:
                leagueName = "Super League";
```

```
                break;
            case 7:
                leagueName = "England Championship";
                break;
            case 8:
                leagueName = "Segunda Division";
                break;
        }

        return leagueName;
    }

    private void btnSearch_Click(object sender, EventArgs e)
    {
        Int16 weekValue = 0;
        Int16 yearValue = 0;
        Int16 roundValue = 0;
        string leagueValue = null;
        Int16 leagueNumber = 0;

        if (cmbRounds.SelectedItem != null)
        {
            string dateValue = (cmbRounds.SelectedItem as ComboboxItem).Value.ToString();
            string[] arrayDate = dateValue.Split('-');
            weekValue = Int16.Parse(arrayDate[0]);
            yearValue = Int16.Parse(arrayDate[1]);
            roundValue = Int16.Parse(arrayDate[2]);

        }

        if (cmbleague.SelectedItem != null)
        {
            leagueValue = cmbleague.SelectedItem.ToString();
            leagueNumber = nameToId(leagueValue);
        }

        MySqlConnection connection = DBUtils.GetDBConnection();

        try
        {
            connection.Open();
            string sql = null;
            int searchCase = 0;
            if ((leagueNumber == 0) && (weekValue == 0))
            {
                sql = "Select lid,position,name,points,leaguegroup from sli_score WHERE
week=@week AND year=@year ORDER BY lid";
            }
            else if ((leagueNumber == 0) && (weekValue > 0))
            {
                sql = "Select lid,position,name,points,leaguegroup from sli_score WHERE
week=@week AND year=@year ORDER BY lid";
                searchCase = 1;
            }
            else if ((leagueNumber > 0) && (weekValue > 0))
            {
```

```
            sql = "Select lid,position,name,points,leaguegroup from sli_score WHERE lid=@lid
AND week=@week AND year=@year ORDER BY lid";
            searchCase = 2;
        }
        else
        {
            sql = "Select lid,position,name,points,leaguegroup from sli_score WHERE
week=@week AND year=@year ORDER BY lid";
        }

        MySqlCommand cmd = new MySqlCommand();
        cmd.Connection = connection;
        cmd.CommandText = sql;
        cmd.Prepare();

        // Create Parameter. Week and year parameters are used in all cases
        MySqlParameter weekParam = new MySqlParameter("@week", MySqlDbType.Int16);
        cmd.Parameters.Add(weekParam);
        MySqlParameter yearParam = new MySqlParameter("@year", MySqlDbType.Int16);
        cmd.Parameters.Add(yearParam);

        DateTime thisDay = DateTime.Today;
        CultureInfo ciCurr = CultureInfo.CurrentCulture;
        int yearNum = ciCurr.Calendar.GetYear(thisDay);
        int weekNum = ciCurr.Calendar.GetWeekOfYear(thisDay,
CalendarWeekRule.FirstFourDayWeek, DayOfWeek.Monday);

        switch (searchCase)
        {
            case 0:
                cmd.Parameters["@week"].Value = Convert.ToInt16(weekNum); //get as week
current week
                cmd.Parameters["@year"].Value = Convert.ToInt16(yearNum); //get as year
current year
                break;
            case 1:
                cmd.Parameters["@week"].Value = Convert.ToInt16(weekValue);
                cmd.Parameters["@year"].Value = Convert.ToInt16(yearValue);
                break;
            case 2:
                cmd.Parameters["@week"].Value = Convert.ToInt16(weekValue);
                cmd.Parameters["@year"].Value = Convert.ToInt16(yearValue);
                MySqlParameter leagueParam = new MySqlParameter("@lid",
MySqlDbType.Int16);
                cmd.Parameters.Add(leagueParam);
                cmd.Parameters["@lid"].Value = Convert.ToInt16(leagueNumber);
                break;
        }


        MySqlDataReader reader = cmd.ExecuteReader();
        dataGridView18.Rows.Clear();
        while (reader.Read())
        {
            string league = idToname(reader.GetInt32("lid"));
            string position = reader.GetInt32("position").ToString();
```

```
            string name = reader.GetString("name");
            string points = reader.GetInt16("points").ToString();
            string group = reader.GetString("leaguegroup");
            dataGridView18.Rows.Add(league, position, name, points,group);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
    finally
    {
        // Close connection
        connection.Close();
        // Freeing Resources
        connection.Dispose();
        connection = null;
    }
}

private Int16 nameToId(String leagueName)
{
    Int16 leagueNum = 0;
    switch (leagueName)
    {
        case "Premiere League":
            leagueNum = 1;
            break;
        case "Bundesliga":
            leagueNum = 2;
            break;
        case "Ligue 1":
            leagueNum = 3;
            break;
        case "Primera Division":
            leagueNum = 4;
            break;
        case "Serie A":
            leagueNum = 5;
            break;
        case "Super League":
            leagueNum = 6;
            break;
        case "England Championship":
            leagueNum = 7;
            break;
        case "Segunda Division":
            leagueNum = 8;
            break;
        case "All":
            leagueNum = 0;
            break;

    }

    return leagueNum;
```

```
    }

    public static int GetNextWeekday(HtmlNode checkDate )
    {
        DateTime dt = DateTime.Today;
        DateTime currHtmlDate = DateTime.Today;
        int daysToAdd = 0;
        int isValidDate = 0;
        if (checkDate != null)
        {
            Regex rgx = new Regex("[^0-9/]");
            var getDate = rgx.Replace(checkDate.InnerText.Trim(), "") + "/" +
DateTime.Now.Year.ToString();

            // The (... + 7) % 7 ensures we end up with a value in the range [0, 6]
            daysToAdd = ((int)DayOfWeek.Friday - (int)dt.DayOfWeek + 7) % 7;
            currHtmlDate = Convert.ToDateTime(getDate);
        }else
        {
            return isValidDate;
        }
        DateTime currFriday = dt.AddDays(daysToAdd);
        DateTime currSunday = currFriday.AddDays(2);


        if ((currHtmlDate >= currFriday ) && (currHtmlDate <= currSunday))
        {
            isValidDate = 99;
        }
        return isValidDate;
    }

    private void dataGridView13_CellDoubleClick(object sender, DataGridViewCellEventArgs
e)
    {
        if (dataGridView13.SelectedRows.Count > 0)
        {
            var selectedRow = dataGridView13.SelectedRows[0];
            var selectedRowName = selectedRow.Cells[0].Value;
            var selectedRowOdds = selectedRow.Cells[1].Value;

            if (dgvPlayersOdss.RowCount < 11)
            {
                dgvPlayersOdss.Rows.Add(selectedRowName, selectedRowOdds);
            }

        }
    }

    private void cmbleague_SelectedIndexChanged(object sender, EventArgs e)
    {
        loadHistory();
    }

    public int getRespectiveRow(DataTable db1, String Teams)
    {
```

```
            String gameText = "";
            int currLine = 998;
            int counter = 0;
            if (Teams != null)
            {

                foreach (DataRow row in db1.Rows)
                {
                    gameText = row["Ομάδες"].ToString();
                    bool b = gameText.Contains(Teams);
                    if (b == true)
                    {
                        currLine = counter;
                    }
                    counter += 1;
                }
            }
                return currLine;
        }
    }
}
```