ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ **Ε**ΠΙΚΟΙΝΩΝΙΩΝ, **Η**ΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ **Σ**ΥΣΤΗΜΑΤΩΝ **Π**ΛΗΡΟΦΟΡΙΚΗΣ

Διεπαφή χρήστη μέσω XML σε περιβάλλον .NET

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ελευθέριος-Νικόλαος Σ. Τρίκκας

Επιβλέπων : Γεώργιος Ι. Στασινόπουλος
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούνιος 2009

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ **Ε**ΠΙΚΟΙΝΩΝΙΩΝ, **Η**ΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ **Σ**ΥΣΤΗΜΑΤΩΝ **Π**ΛΗΡΟΦΟΡΙΚΗΣ

Διεπαφή χρήστη μέσω XML σε περιβάλλον .NET

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ελευθέριος-Νικόλαος Σ. Τρίκκας

Επιβλέπων : Γεώργιος Ι. Στασινόπουλος
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την ……

.....	.....	.....
Γ. Στασινόπουλος	Μ. Θεολόγου	Β. Λούμος
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούνιος 2009

.....
Ελευθέριος-Νικόλαος Σ. Τρίκκας

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

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΠΕΡΙΛΗΨΗ 11

ABSTRACT 13

ΚΕΦΑΛΑΙΟ 1. ΕΙΣΑΓΩΓΗ 17

1.1 Η ιδέα της δημιουργίας του διαδικτύου 17

1.2 Το διαδίκτυο σήμερα 18

1.3 Συνοπτική περίληψη της διπλωματικής εργασίας 22

ΚΕΦΑΛΑΙΟ 2. Η ΓΛΩΣΣΑ XML 25

2.1 Η ιστορία της XML 25

2.2 Δομή και σύνταξη της XML 27
2.2.1 – Εισαγωγή στις βασικότερες έννοιες της XML 27
2.2.2 – Μεταβλητές ιδιοτήτων (attributes) 30
2.2.3 – Η σημασία και η χρήση των namespaces 30
2.2.4 – Άλλα περιεχόμενα στοιχείου 32

ΚΕΦΑΛΑΙΟ 3. ΤΟ ΠΕΡΙΒΑΛΛΟΝ .NET FRAMEWORK ΚΑΙ ΤΑ ΕΡΓΑΛΕΙΑ ΑΝΑΠΤΥΞΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ 35

3.1 Το περιβάλλον .NET 35

3.2 Διεπαφή χρήστη 36

3.3 Εργαλεία συστήματος 38
3.3.1 - Η γλώσσα προγραμματισμού C# 38
3.3.2 - Το πρόγραμμα Visual Studio Professional Edition 39
3.3.3 - ASP.NET 41

ΚΕΦΑΛΑΙΟ 4. Η ΕΦΑΡΜΟΓΗ XML EDITOR 45

4.1 Σχεδιασμός της εφαρμογής XML Editor 45

4.2 Υλοποίηση του XML Editor 46
4.2.1 - Γραφική υλοποίηση του προγράμματος 46
4.2.2 - Υλοποίηση βασικών συναρτήσεων 50

4.3 Εκτέλεση της εφαρμογής 61

4.4 Μελλοντικές βελτιοποιήσεις της εφαρμογής 75

ΚΕΦΑΛΑΙΟ 5. ΠΗΓΑΙΟΣ ΚΩΔΙΚΑΣ ΠΡΟΓΡΑΜΜΑΤΟΣ (SOURCE CODE) 78

5.1 Η κλάση TreeViewToXml 78

5.2 Η κλάση Default 80

ΒΙΒΛΙΟΓΡΑΦΙΑ ΚΑΙ ΣΥΝΔΕΣΜΟΙ ΑΝΑΦΟΡΑΣ 95

ΠΙΝΑΚΑΣ ΠΙΝΑΚΩΝ ΚΑΙ ΣΧΗΜΑΤΩΝ

Σχήμα 1.1 – Dial-up σύνδεση

Σχήμα 1.2 – DSL σύνδεση

Πίνακας 1.3 – Χαρακτηριστικά τύπων xDSL

Σχήμα 2.1 – Σχέση της XML με άλλες μορφές αποθήκευσης και παρουσίασης

Σχήμα 3.1 – Το μοντέλο πελάτη-εξυπηρετητή

Σχήμα 3.2 – Η αρχική σελίδα του Visual Studio 2005 Professional Edition

Σχήμα 3.3 – Αναπαράσταση του TreeView Control

Σχήμα 4.1 – Γραφική υλοποίηση (design view) του XMl Editor

Σχήμα 4.2 – Η εργαλειοθήκη Toolbox του Visual Studio 2005 Professional Edition

Σχήμα 4.3 – Η αρχική σελίδα του XML Editor

Σχήμα 4.4 – Το επιλεγμένο αρχείο δεν είναι xml και πρέπει να επιλεγεί άλλο από τη λίστα

Σχήμα 4.5 – Δεντρική μορφή αρχείου xml

Σχήμα 4.6 – Επιλογή κόμβου με element→Delicious, attribute→Parent και attribute value→“Apples”

Σχήμα 4.7 – Οι πέντε βασικές λειτουργίες της εφαρμογής όπως εμφανίζονται για το επιλεγμένο element Macintosh

Σχήμα 4.8 – Η αρχική σελίδα της εφαρμογής XML Editor.

Σχήμα 4.9 – Απεικόνιση στη λίστα όλων των αρχείων του φακέλου.

Σχήμα 4.10 – Εμφάνιση μηνύματος σφάλματος για μη επιλογή xml αρχείου προς επεξεργασία.

Σχήμα 4.11 – Το πρόγραμμα XML Editor και οι δυνατότητες του

Σχήμα 4.12 – Επιλογή του element Fruit και απεικόνιση των attributes και values στον editor

Σχήμα 4.13 – Μετονομασία element με μη προκαθορισμένο όνομα

Σχήμα 4.14 – Οι attributes του element Fruit όπως απεικονίζονται στην DropDown List

Σχήμα 4.15 – Διαδικασία διαμόρφωσης της attribute Second και της τιμής της “23s”

Σχήμα 4.16– Δεντρική μορφή του αρχείου NEWXML.xml πριν τη διαμόρφωση

Σχήμα 4.17 – Δεντρική μορφή του αρχείου NEWXML.xml μετά τη διαμόρφωση

Σχήμα 4.18 – Επιλογή του element Delicious για προσθήκη νέου element

Σχήμα 4.19 – Προσθήκη element εκτός λίστας προϋπάρχοντων ονομάτων

Σχήμα 4.20 – Απεικόνιση αρχείου μετά την προσθήκη του element Oranges

Σχήμα 4.21 – Προσθήκη attribute με όνομα pineapples και τιμή 18

Σχήμα 4.22 – Εμφάνιση μηνύματος λάθους σε απόπειρα διαγραφής του root element

Σχήμα 4.23 – Επιλογή του element Macintosh προς διαγραφή

Σχήμα 4.24 – Η εικόνα του αρχείου μετά τη διαγραφή του element Macintosh

Σχήμα 4.25 –Το πεδίο διαγραφής των attributes γεμίζει με τα ονόματα του επιλεγμένου κόμβου Fruit

Σχήμα 4.26 – Η εικόνα του αρχείου αμέσως μετά τη διαγραφή της attribute με όνομα Parent
Σχήμα 4.27 – Το τροποποιημένο αρχείο NEWXML.xml
Σχήμα 4.28 – Δεύτερη τροποποίηση του ήδη τροποποιημένου αρχείου NEWXML.xml
Σχήμα 4.29 – Εμφάνιση αρχικής σελίδας του XML Editor με ένα δις τροποποιημένο και αποθηκευμένο αρχείο

Περίληψη

Σκοπός της εργασίας αυτής είναι η υλοποίηση μιας δυναμικής ιστοσελίδας με ένα Graphical User Interface (GUI) που κινεί την εφαρμογή στην οποία ο χρήστης μπορεί να διαγράψει ή να τροποποιήσει ή να εισάγει νέα elements και attributes σε ένα προϋπάρχον αρχείο XML και στη συνέχεια να αποθηκεύεται το παραπάνω αρχείο για περαιτέρω χρήση. Τα ονόματα (και οι τιμές αντίστοιχα για τα attributes) κατά την εισαγωγή και την τροποποίηση θα επιλέγονται είτε από κάποια προϋπάρχοντα ονόματα που βρίσκονται σε DropDown List είτε ο χρήστης θα μπορεί να γράψει ένα δικό του όνομα με τη χρήση του Textbox. Η DropDown List και το Textbox εξηγούνται επαρκώς στο κεφάλαιο 4 της εργασίας. Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκε το πρόγραμμα Microsoft Visual Studio 2005 Professional Edition, ενώ η εφαρμογή είναι γραμμένη σε aspx (ASP.NET) με χρήση της γλώσσας προγραμματισμού C#.

Η συγκεκριμένη εφαρμογή μπορεί να χρησιμοποιηθεί σε οποιονδήποτε από τους σημερινούς φυλλομετρητές (browsers). Για τις ανάγκες της εργασίας η εκτέλεση της εφαρμογής του XML Editor έγινε στον Mozilla Firefox 3.0 . Γίνεται μια γνωριμία του αναγνώστη με τα κυριότερα στοιχεία της μετα-γλώσσας XML, δηλαδή τα στοιχεία (elements), τις ιδιότητες (attributes) και τις τιμές αυτών (values) και πως όλα αυτά συντάσσονται και δημιουργούν ένα έγκυρο (valid) xml αρχείο. Ο χρήστης μεταβάλλει τα elements και τις attributes ενός xml αρχείου με κύρια βοήθεια τη χρήση της δεντρικής μορφής του αρχείου. Έτσι, όταν επιλέγει ένα κόμβο του δέντρου ο κόμβος αυτός χρωματίζεται διαφορετικά από τους υπολοίπους, τα στοιχεία του κόμβου εμφανίζονται σε TextBoxes και μπορεί να διαγράψει ή να διαμορφώσει ή να προσθέσει ένα καινούριο κόμβο σε αυτόν.

Για την καλύτερη κατανόηση του XML Editor και του τρόπου που αυτός εφαρμόζεται παρατίθεται αναλυτικά σχόλια, καθώς και αρκετές εικόνες που διευκολύνουν την κατανόηση της λειτουργίας του.

Λέξεις κλειδιά

XML, elements, attributes, ASP.NET, C#, Visual Studio 2005, δυναμική ιστοσελίδα, επεξεργασία XML αρχείου, δεντρική μορφή XML αρχείου, προσθήκη στοιχείου ή ιδιότητας και τιμής, διαγραφή στοιχείου ή ιδιότητας και τιμής, διαμόρφωση στοιχείου ή ιδιότητας και τιμής

Abstract

The aim of this project is the implementation of a web-based page with a GUI that runs an application. With such an application the user is able to delete, edit or add new elements as well as attributes in a preexistent XML file. The above stated file could be later saved for farther usage. The names (and the values for the attributes respectively) as far as the introduction and the editing are concerned will be chosen either from preexistent names already stated in a DropDown List, or the user will type a name of his/her own using the Textbox. The DropDown List as well as the Textbox are fully explained in Chapter 4 of this dissertation. In order to implement the particular application the programme used was Microsoft Visual Studio 2005 Professional Edition, while ASP.NET was used to write it also using programming language C#.

The particular application can be used in any of the nowadays browsers. For this project's needs, the execution of this XML Editor application took place on Mozilla Firefox 3.0. The reader will be acknowledged with the most significant components of XML that is the elements, the attributes and their values. Moreover the way in which all these types compose in order to develop an XML file will also be clarified. The user can alter the elements and the attributes additionally, using the tree-view of the file. Hence, when a tree node is selected, the node is highlighted in a different, from the rest, color. Furthermore, the node components appear in TextBoxes and the user can actually delete, modulate or add a new node to that one.

For a better XML Editor comprehension and the way this is applied, analytic commentary as well as a variety of images that can help the reader's understanding are cited in this paper.

KeyWords

XML, elements, attributes, ASP.NET, C#, Visual Studio 2005, web-based page, XML file editing, tree view of an XML file, element or attribute or value addition / removal/ editing

Ευχαριστίες

Αισθάνομαι την ανάγκη να ευχαριστήσω θερμά τον Καθηγητή και επιβλέποντα της παρούσας διπλωματικής εργασίας κ. Γ. Στασινόπουλο για την ευκαιρία που μου έδωσε να εργαστώ σε ένα σύγχρονο αντικείμενο και για τη συνεχή βοήθεια και καθοδήγηση του κατά την διάρκεια της εργασίας.

Επίσης, ευχαριστώ την οικογένειά μου καθώς και τους φίλους μου για την αγάπη και την υποστήριξή τους σε όλη τη διάρκεια της φοιτητικής μου διαδρομής,.

1

Εισαγωγή

1.1 Η ιδέα της δημιουργίας του διαδικτύου

Το διαδίκτυο, στην πιο πρωτόγονη μορφή του, γεννήθηκε τη δεκαετία του 1970 στις ΗΠΑ, ως ένα σύστημα ενδοπανεπιστημιακής συνεννόησης. Είχε στηριχθεί σε μαθηματικές αρχές κι εφαρμογές των αρχών του 20ου αιώνα, πολλές από τις οποίες αναπτύχθηκαν για στρατιωτικούς σκοπούς. Το όνομά του το οφείλει στο πρώτο πρόγραμμα επικοινωνίας μεταξύ ετερογενών δικτύων που ανέπτυξε η ARPA και ονόμασε Internetting. Ο πυρήνας του Διαδικτύου ξεκίνησε το 1969 με την ονομασία ARPANET στην Υπηρεσία Προηγμένων Αμυντικών Ερευνών (Defense Advanced Research Projects Agency, *DARPA*) του υπουργείου Άμυνας των ΗΠΑ. Η αρχική έρευνα που συνέβαλε στο ARPANET περιελάμβανε εργασίες στα αποκεντρωμένα δίκτυα, το queueing theory και την ανταλλαγή πακέτων packet switching. Στις 11 Ιανουαρίου 1983 το ARPANET άλλαξε το βασικό του δικτυακό πρωτόκολλο επικοινωνίας από το NCP στο TCP/IP, ξεκινώντας έτσι το Διαδίκτυο όπως το γνωρίζουμε σήμερα. Σημαντικό βήμα στην ανάπτυξη του Διαδικτύου έκανε το Εθνικό Ίδρυμα Επιστημών (National Science Foundation, NSF) των ΗΠΑ, το οποίο έχτισε την πρώτη Διαδικτυακή πανεπιστημιακή ραχοκοκκαλιά, το NSFNet, το 1986. Ακολούθησε η ενσωμάτωση άλλων σημαντικών δικτύων, όπως το Usenet, το Fidonet και το Bitnet.

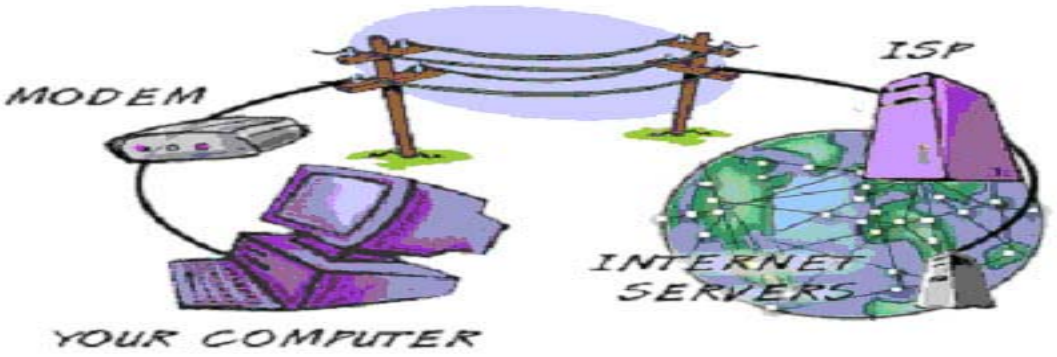
Το Διαδίκτυο γνώρισε τρομακτική ανάπτυξη κατά τη δεκαετία του 1990, απορροφώντας επιτυχώς την πλειοψηφία των παλιότερων δικτύων υπολογιστών. Αυτή η ανάπτυξη συχνά αποδίδεται στην έλλειψη κεντρικού ελέγχου για το Διαδίκτυο, η οποία επιτρέπει την οργανική ανάπτυξη του, όπως και στο μη ιδιοκτησιακό καθεστώς των πρωτοκόλλων του, τα οποία απέτρεψαν την άσκηση ελέγχου από μία και μόνο εταιρεία, καθώς επίσης και στο γεγονός ότι το ακριβό αυτό πανεπιστημιακό σύστημα γίνεται προσιτό στο κοινό, περνώντας από την ευρύτερη ακαδημαϊκή κοινότητα και στους απλούς πολίτες. Μόλις δέκα χρόνια αργότερα, περισσότεροι από 350 εκατομμύρια άνθρωποι σε όλο τον κόσμο είχαν πρόσβαση στο διαδίκτυο, κι ο αριθμός παρουσιάζει ραγδαία αυξητικές τάσεις.

1.2 Το διαδίκτυο σήμερα

Το ίντερνετ ασκεί τρομερή επιρροή στην γνώση αλλά και τη διαμόρφωση απόψεων. Μέσα από την αναζήτηση *λέξεων-κλειδιών* (key words) μέσω της χρήσης μηχανών αναζήτησης όπως το Google,το Yahoo κλπ, εκατομμύρια άνθρωποι έχουν εύκολη και άμεση πρόσβαση σε ένα τεράστιο, παγκόσμιο και ποικίλο όγκο πληροφοριών. Συγκρινόμενο με τις έντυπες εγκυκλοπαίδειες και τις παραδοσιακές βιβλιοθήκες, το Ίντερνετ αντιπροσωπεύει μία ξαφνική και απότομη αποκέντρωση των πληροφοριών και των δεδομένων. Παράλληλα, η μεγάλη διάδοση του διαδικτύου σε μια σειρά εκφάνσεις του ανθρώπινου βίου, και κυρίως στο εμπόριο, οφείλεται κατά κύριο λόγο στην ανάπτυξη των φυλλομετρητών (browsers) όπως ο Internet Explorer της Microsoft, ο ανοιχτού κώδικα Firefox, ο Opera και τελευταία ο Safari που αναπτύχθηκε από την Applekαι καθώς επίσης και στη δημιουργία του παγκόσμιου ιστού, του world wide web (www), της ευρωπαϊκής προσφοράς στην ανάπτυξη των νέων τεχνολογιών, που γεννήθηκε στο CERN και το οποίο πολλοί ταυτίζουν με το ίντερνετ. Επίσης η ανάπτυξη μιας σειράς προγραμμάτων που προσφέρουν τη δυνατότητα της άμεσης και σχετικά χαμηλού κόστους πληροφορίας όπως τα προγράμματα άμεσης αλληλογραφίας, άμεσης συζήτησης, μεταφοράς εικόνας και ήχου, συζήτησης με εικόνα και ήχο σε πραγματικό χρόνο και πολλά ακόμη, παρέχουν τη δυνατότητα διεύρυνσης των ανθρωπίνων δραστηριοτήτων με νέους τρόπους, που συνήθως φέρουν το γράμμα "e" ως διακριτικό του μέσου μέσω του οποίου συντελούνται. Στην αλληλογραφία (mail) προστέθηκε η ηλεκτρονική αλληλογραφία (e-mail), στο εμπόριο (commerce) το ηλεκτρονικό εμπόριο (e-commerce), στις ομάδες συζήτησης οι ηλεκτρονικές ομάδες συζήτησης (e-forums) κοκ..

Η γλώσσα που χρησιμοποιείται περισσότερο για την επικοινωνία στο Διαδίκτυο είναι η Αγγλική. Αυτό συμβαίνει κυρίως λόγω της Αμερικανικής καταγωγής του Ίντερνετ, της χρήσης της Αγγλικής στον προγραμματισμό λογισμικού και στην αδυναμία των πρώτων γενιών υπολογιστών να χρησιμοποιήσουν άλλους χαρακτήρες πέραν του λατινικού αλφάβητου. Έχοντας μεγαλώσει αρκετά τα τελευταία χρόνια, το Διαδίκτυο περιλαμβάνει πλέον επαρκές περιεχόμενο και στις υπόλοιπες γλώσσες των περισσότερο ανεπτυγμένων χωρών

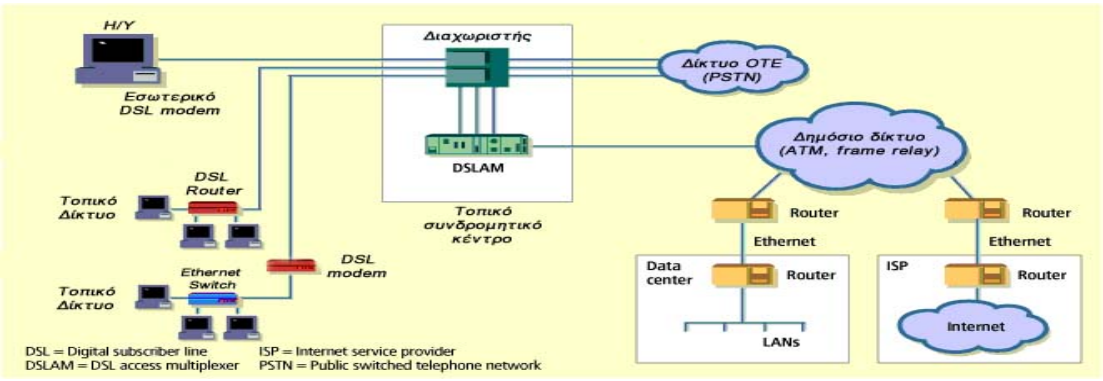
Κοινές μέθοδοι πρόσβασης στο Διαδίκτυο είναι η επιλογική(dial-up) και η ευρυζωνική (cable ή DSL). Ως επιλογική μέθοδος πρόσβασης ορίζεται η σύνδεση με το Internet για την οποία απαιτείται και χρησιμοποιείται μια σταθερή τηλεφωνική γραμμή και ένα modem. Η σύνδεση με το Internet επιτυγχάνεται μέσω τηλεφωνικής κλήσης σε κάποιον ISP. Κύρια διαφορά της dial-up από την ευρυζωνική πρόσβαση είναι η ύπαρξη χρονοχρέωσης και οι πολύ μικρότερες ταχύτητες ρυθμού δεδομένων. Ακολούθως παρατίθεται το σχήμα μιας dial-up σύνδεσης:



Σχήμα 1.1 – Dial-up σύνδεση

Κατά την ευρυζωνική πρόσβαση, ο χρήστης μπορεί να έχει α)αξιόπιστη και διαρκή σύνδεση στο διαδίκτυο, χωρίς την ύπαρξη χρονοχρέωσης, β)υψηλές ταχύτητες μεταβίβασης δεδομένων και γ) προηγμένες υπηρεσίες επικοινωνιών και βελτίωση της σχέσης μεταξύ κράτους–πολίτη-επιχειρήσεων: eHealth, eGovernment, eLearning, eBusiness. Το DSL προέρχεται από τα αρχικά των λέξεων Digital Subscriber Line και στην ουσία αποτελεί μια τεχνολογία που μετατρέπει το απλό τηλεφωνικό καλώδιο σε ένα δίαυλο ψηφιακής επικοινωνίας μεγάλου εύρους ζώνης με τη χρήση ειδικών modems, τα οποία τοποθετούνται στις δυο άκρες της γραμμής. Ο δίαυλος αυτός μεταφέρει τόσο τις χαμηλές όσο και τις

υψηλές συχνότητες ταυτόχρονα, τις χαμηλές για τη μεταφορά του σήματος της φωνής και τις υψηλές για τα δεδομένα. Ανάλογα με το είδος του modem που θα συνδέσουμε, πετυχαίνουμε και διαφορετικές επιδόσεις. Με το DSL επιτυγχάνονται υψηλότερες ταχύτητες μεταφοράς δεδομένων (μέχρι και 52,8 Mbps από το Διαδίκτυο ή άλλο απομακρυσμένο Τηλεπικοινωνιακό Δίκτυο προς το χρήστη -downstream- και 2,3 Mbps από το χρήστη προς το Διαδίκτυο -upstream-)ενώ ταυτόχρονα μεταφέρονται και τα αναλογικά σήματα της φωνής. Ακολουθεί η σχηματική αναπαράσταση μιας ευρυζωνικής πρόσβασης :



Σχήμα 1.2 – DSL σύνδεση

Οι τεχνολογίες DSL αναφέρονται γενικά ως xDSL και οι κυριότερες από αυτές είναι: ADSL, HDSL, SDSL και VDSL . Αναλυτικότερα:

•**ADSL**: Η τεχνολογία ADSL εξασφαλίζει πρόσβαση υψηλών ταχυτήτων στο Διαδίκτυο και σε άλλα Τηλεπικοινωνιακά Δίκτυα, δίνοντας τη δυνατότητα για ταυτόχρονη μετάδοση φωνής και δεδομένων (δεδομένα, κινούμενη εικόνα, γραφικά) μέσω της απλής τηλεφωνικής γραμμής. Κύριο χαρακτηριστικό της τεχνολογίας είναι ότι η μεταφορά δεδομένων γίνεται με ασύμμετρο τρόπο, δηλαδή προσφέρει διαφορετικό ρυθμό για τη λήψη και διαφορετικό για την αποστολή δεδομένων. Το σημαντικότερο είναι ότι το εύρος ζώνης δεν το μοιραζόμαστε, αλλά είναι εξ' ολοκλήρου στη διάθεσή μας. Ωστόσο θα πρέπει να τονιστεί το γεγονός ότι η απόδοση του ADSL εξαρτάται σημαντικά από την απόσταση του χρήστη από τον τηλεπικοινωνιακό παροχέα

•**HDSL**: Το ακρωνύμιο HDSL προέρχεται από τα αρχικά των λέξεων High-bit-rate Digital Subscriber Line και σε αντίθεση με το ADSL είναι συμμετρικό και προσφέρει τον ίδιο ρυθμό μεταφοράς δεδομένων (μέχρι 2 Mbps) τόσο για τη αποστολή όσο και για τη λήψη. Ωστόσο, η μέγιστη απόσταση μεταξύ των δύο άκρων δεν μπορεί να υπερβαίνει τα 3,5 km. Μια άλλη βασική διαφορά από το ADSL είναι ότι απαιτείται η εγκατάσταση 2 τηλεφωνικών γραμμών (2 συνεστραμμένα καλώδια).

•**SDSL**: Το SDSL**,** Single-line Digital Subscriber Line, είναι μια τεχνολογία παρόμοια με το HDSL όσον αφορά στο ρυθμό μεταφοράς δεδομένων (μέχρι 2 Mbps), που απαιτεί όμως μόνο ένα συνεστραμμένο ζεύγος χαλκού. Για το λόγο αυτό, η μέγιστη απόσταση μεταξύ των δύο άκρων δεν μπορεί να ξεπερνά τα 3 km.

.•**VDSL**: Το VDSL**,** Very-high-data-rate Digital Subscriber Line, βρίσκεται ακόμη σε φάση ανάπτυξης και υπόσχεται να δώσει εντυπωσιακά μεγαλύτερες ταχύτητες που μπορεί να φτάνουν τα 52 Mbps, με περιορισμό όμως τη μέγιστη απόσταση μεταξύ των δύο άκρων του χάλκινου αγωγού. Ανάλογα με την υλοποίηση, το VDSL δε μπορεί να ξεπερνά το 1,5 km και οι ρυθμοί μετάδοσης κυμαίνονται για τη λήψη από 13 έως 52 Mbps και για την αποστολή από 1,5 έως 2,3 Mbps.

Στον παρακάτω πίνακα παρατίθενται μια συγκεντρωτική περιγραφή των τύπων DSL :

xDSL	Περιγραφή	Ρυθμός Δεδομένων		Απόσταση
		upstream	downstream	
ADSL	Asymmetric Digital Subscriber Line	16 – 1024 Kbps	1.5 - 8 Mbps	1.544 Mbps - 5.5 km 2.048 Mbps - 4.8 km 6.312 Mbps - 3.6 km 8.448 Mbps - 2.7 km
ADSL lite	Splitterless DSL	512Kbps	1.5 Mbps	5.5 km
ADSL2	Asymmetric Digital Subscriber Line 2	1 - 3.5 Mbps	12 Mbps	< 6 km
ADSL2+	Asymmetric Digital Subscriber Line 2 +	1 - 3.5 Mbps	24 Mbps	< 6 km
HDSL	High bit-rate Digital Subscriber Line	1.544 Mbps duplex 2.048 Mbps	1.544 Mbps 2.048 Mbps	1.544 Mbps - 5.5 km 2.048 Mbps - 3.6 km
SDSL	Symmetric DSL	4.6 Mbps duplex (2UTP) 2.3 Mbps duplex (UTP)	4.6 Mbps 2.3 Mbps	3 km (UTP) 6km (2UTP)
VDSL	Very high rate Digital Subscriber Line	2.3 Mbps	52.8 Mbps asymmetric 26 Mbps symmetric	12.96 Mbps - 1.4 km 25.82 Mbps - 0.9 km 51.84 Mbps - 0.3 km

Πίνακας 1.3 – Χαρακτηριστικά τύπων xDSL

Μεταξύ όσων βρίσκουν σύμφωνους τους ερευνητές σήμερα και σχετικά με το μέλλον και την περαιτέρω εξέλιξη του internet είναι το γεγονός ότι στο άμεσο μέλλον θα γνωρίσουν ανάπτυξη τα ασύρματα δίκτυα κι οι ασύρματες εφαρμογές όπως κι η συνδυασμένη τεχνολογία δορυφορικών συστημάτων, κινητής τηλεφωνίας και ίντερνετ. Παράλληλα, οι ταχύτητες πρόσβασης θα αυξάνουν συνεχώς και θα κατευθυνθούμε προς το ένα terabit ή ακόμη υψηλότερες ταχύτητες.

1.3 Συνοπτική περίληψη της διπλωματικής εργασίας

Αφού αναφερθήκαμε συνοπτικά στην πορεία του διαδικτύου από τη δημιουργία του μέχρι και σήμερα καθώς και στις προοπτικές εξέλιξης του μπορούμε να κάνουμε μια σύντομη περιγραφή της διπλωματικής εργασίας. Σκοπός της εργασίας αυτής είναι η υλοποίηση μιας διαδικτυακής εφαρμογής κατά την οποία ο χρήστης μπορεί μέσω δυναμικής σελίδας να διαγράψει ή να τροποποιήσει ή να εισάγει elements και attributes σε ένα προϋπάρχον αρχείο XML και στη συνέχεια να αποθηκεύεται το παραπάνω αρχείο για περαιτέρω χρήση. Τα ονόματα(και οι τιμές αντίστοιχα για τα attributes) κατά την εισαγωγή και την τροποποίηση θα επιλέγονται είτε από κάποια προϋπάρχοντα ονόματα που βρίσκονται σε ένα Combobox είτε ο χρήστης θα μπορεί να γράψει ένα δικό του όνομα με τη χρήση του Textbox(τα Combobox και Textbox εξηγούνται επαρκώς στο κεφάλαιο 4 της παρούσας διπλωματικής εργασίας). Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκε το πρόγραμμα Microsoft Visual Studio 2005 Professional Edition, μια σύντομη περιγραφή του οποίου θα παρακολουθήσουμε στο 4^{ο} κεφάλαιο της εργασίας αυτής..

Κατά το *2^{ο} κεφάλαιο* ο αναγνώστης εισάγεται στις βασικότερες έννοιες της XML, κάτι άκρως απαραίτητο προκειμένου να κατανοήσει έννοιες όπως element,attribute,tags,namespaces.Εν συνεχεία, στο *3^{ο} κεφάλαι*ο γίνεται αναφορά για την ανάπτυξη διεπαφής χρήστη (UI, User Interface) η οποία υλοποιεί την αμφίδρομη επικοινωνία συστήματος χρήστη καθώς και τα κυριότερα χαρακτηριστικά της. Παράλληλα γίνεται μια σύντομη παρουσίαση του περιβάλλοντος .NET 2.0 Framework και των εργαλείων που χρησιμοποιήθηκαν για τη διεκπεραίωση της παρούσας διπλωματικής όπως τα Visual Studio 2005 Professional Edition και ASP.NET 2.0. Τέλος, γίνεται μια αναφορά στη γλώσσα προγραμματισμού που χρησιμοποιήθηκε και η οποία είναι η Visual C# 2005.

Στο *4ο* λαμβάνει χώρα ο σχεδιασμός της υλοποίησης της εφαρμογής μας και η επίδειξη της εκτέλεσης του προγράμματος αντίστοιχα με απεικόνιση screenshots κατά τη διάρκεια της ανάλυσης και ανάπτυξης του κώδικα του προγράμματος. Εν κατακλείδι, στο *5ο κεφάλαιο* ο αναγνώστης μπορεί να βρει τον πηγαίο κώδικα της εφαρμογής (source code) ενώ ακολουθεί ο απολογισμός και ο επίλογος της εργασίας αυτής.

2

Η γλώσσα XML

2.1 Η ιστορία της XML

Η XML (**Extensible Markup Language)** δεν είναι μία σημειακή γλώσσα όπως η HTML, αλλά μία γλώσσα που χρησιμοποιείται για την περιγραφή μίας σημειακής γλώσσας. Ο τεχνικός όρος μιας τέτοιας γλώσσας είναι μετα-γλώσσα. Χρησιμοποιώντας την XML ένας προγραμματιστής μπορεί να προσδιορίσει τις σημειακές γλώσσες που περιγράφουν ηλεκτρονικά κυκλώματα, πληροφορίες για ανταλλαγή ηλεκτρονικών δεδομένων, τα αρχεία που παράγονται από τους διακομιστές Web, μηχανικά μέρη αεροσκαφών και ούτω καθεξής.

Οι ρίζες της XML μπορούν ν' αναζητηθούν στην εκρηκτική ανάπτυξη του Παγκόσμιου Ιστού στα μέσα της δεκαετίας του 1990 και στους πολέμους των browser που έλαβαν χώρα μεταξύ της Microsoft Corporation και της Netscape Corporation, όπου καθεμιά από αυτές πάλευε για την απόλυτη κυριαρχία. Καθώς το Web γινόταν όλο και πιο μεγάλο, και όλο και περισσότεροι χρήστες το χρησιμοποιούσαν, άρχισαν ν' ανακαλύπτονται από τους προγραμματιστές, που χρησιμοποιούσαν HTML, διάφορα προβλήματα, όπως για παράδειγμα το γεγονός ότι ο ίδιος πόρος HTML εμφανιζόταν με διάφορες μορφές, ανάλογα με τον browser που χρησιμοποιούνταν. Αυτό σήμαινε ότι, οι σχεδιαστές των ιστοσελίδων έπρεπε το λιγότερο να διπλασιάσουν τις προσπάθειές τους.

Ορισμένοι κατασκευαστές browser ανέπτυξαν εργαλεία HTML που δεν ήταν αναγνωρίσιμα από άλλους browser. Ήταν σχεδόν αδύνατον να διακριθεί οποιαδήποτε σημαντική μέσα σε μια ιστοσελίδα: πουθενά αλλού αυτό δεν ήταν πιο προφανές από ότι στην χρήση των μηχανών αναζήτησης. Ακόμη και στα μέσα της δεκαετίας του 1990 πολλοί χρήστες εξέφραζαν την απογοήτευσή τους όσον αφορούσε αυτά τα προγράμματα εξαιτίας του όγκου των ανακτημένων αρχείων που επέστρεφαν οι μηχανές, τα οποία, στην καλύτερη περίπτωση, σχετίζονταν οριακά με την αναζήτηση. Η αιτία αυτής της "φτωχής" απόδοσης δεν ήταν η τεχνολογία των μηχανών αναζήτησης - στην πραγματικότητα επρόκειτο για εξεζητημένα προγράμματα που αποτελούν αποδεικτικό της εξυπνάδας των προγραμματιστών - αλλά το ότι τα δεδομένα που επεξεργάζονταν, η σελίδες HTML, δεν έδιναν πολλά στοιχεία για το περιεχόμενό τους.

Εξαιτίας αυτών των προβλημάτων η Κοινοπραξία Παγκόσμιου Ιστού, η ομάδα που ελέγχει την διαδικασία τυποποίησης του Ιστού, αποφάσισε το 1996 ν' αναπτύξει μία σημειακή γλώσσα που μελλοντικά θα υποσκέλιζε την HTML. Τα στόχοι αυτής της γλώσσας ήταν:

•Να χρησιμοποιείται εύκολα στο Internet.

•Να μπορεί να υποστηρίζει πολλές εφαρμογές οι οποίες θα κυμαίνονται από browser μέχρι βάσεις δεδομένων μηχανών αναζήτησης.

•Να είναι συμβατή με την SGML, την γλώσσα επεξεργασίας κειμένου που αποτέλεσε την έμπνευση για την HTML.

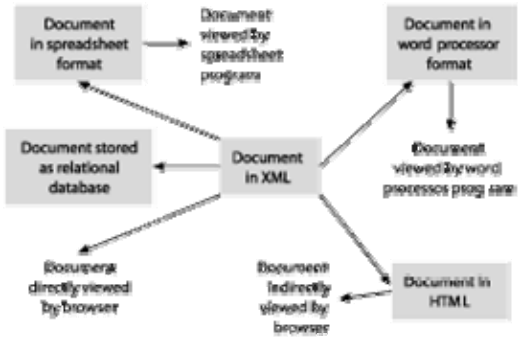
•Να μην αποτελεί πολύπλοκη διαδικασία η ανάπτυξη επεξεργαστών κειμένων γραμμένων σε γλώσσες που θα βασίζονταν σε XML, για παράδειγμα θα έπρεπε να είναι εύκολη η εγγραφή ενός προγράμματος για τον έλεγχο της σαφήνειας ενός κειμένου πόρου.

•Ο αριθμός των προαιρετικών εργαλείων της γλώσσας να είναι χαμηλός.

•Να είναι εύκολη η ανάγνωση και κατανόηση των αρχείων XML.

•Να είναι εύκολο ν' αναπτυχθούν, με την χρήση απλών συντακτών, αρχεία γραμμένων σε γλώσσα βασιζόμενη σε XML.

Το 1998 η γλώσσα, XML, παρουσιάστηκε παγκόσμια ως μία τελευταία υπόδειξη της Κοινοπραξίας Παγκόσμιου Ιστού. Σαν αποτέλεσμα, έγινε μία σταθερά για το Internet. Βασιζόταν στην γλώσσα επεξεργασίας κειμένων SGML, η οποία αποτέλεσε την έμπνευση για την HTML. Ο ρόλος της XML συνοψίζεται στο Σχήμα 2.1, το οποίο παρουσιάζει την σχέση της με άλλες μορφές αποθήκευσης και παρουσίασης.



Σχήμα 2.1 – Σχέση της XML με άλλες μορφές αποθήκευσης και παρουσίασης

2.2 Δομή και σύνταξη της XML

2.2.1 – Εισαγωγή στις βασικότερες έννοιες της XML

Όπως και στην HTML (Hyper Text Mark-up Language) έτσι και η XML είναι μια γλώσσα πού επισημαίνει επιπλέον πληροφορία αναφερόμενη στο βασικό κείμενο. Στην XML προσδιορίζομε την πληροφορία αυτή με την βοήθεια ***ετικετών*** (tags), δηλ. με ένα αναγνωριστικό περικλειόμενο από <>. Με τις ετικέτες (tags) επιτυγχάνουμε το "markup".Tags υπάρχουν και στην HTML, π.χ. το ζεύγος <B> </B> ή το μοναδικό
. Αυτά όμως είναι προκαθορισμένα και σχετίζονται μόνο με την εμφάνιση του κειμένου. Τα tags της XML είναι προσδιορίσιμα ελεύθερα από τον χρήστη, ο οποίος μπορεί να τους δίδει και όποια σημασία αυτός ορίσει. Για τούτο πρέπει και πάντα να ζευγαρώνονται, δηλ. <text1>my car is here</text1>. Το αναγνωριστικό <text1> </text1> δεν προσδιορίζει αναγκαστικά το πώς θα εμφανισθεί το περικλειόμενο κείμενο (όπως θα έκανε το <B> </B> της HTML), αλλά μπορεί π.χ. να συνδυάζεται με μία ενέργεια, όπως θα καθορίζει δικός μας κώδικας που θα επεξεργασθεί το αρχείο XML. Το ζεύγος <text1> και </text1> ορίζει ένα ***στοιχείο*** (element), του οποίου το όνομα είναι το text1 και το περιεχόμενο οτιδήποτε περικλείεται μεταξύ του <text1> (opening tag) και του <text1> (closing tag).

Το επόμενο βήμα είναι η ενθυλάκωση (nesting) που δίδει την δύναμη να κατασκευασθούν στοιχεία με πολύπλοκη εσωτερική δενδρική δομή, π.χ.

```
<message>
  <to>you@yourAddress.com</to>
  <from>me@myAddress.com</from>
  <subject>XML Is Really Cool</subject>
  <text>
    How many ways is XML cool? Let me count the ways...
  </text>
 <another_text/>
</message>
```

Τα πυκνοτυπωμένα είναι tags (αναγνωριστικά), πάντα ζευγαρωμένα. Τόσο τα tags, όσο και η περικλειόμενη πληροφορία είναι σχεδόν πάντα κείμενο (text). Η XML είναι εύκαμπτη, ευκόλως εννοούμενη, άλλα βεβαίως και σπάταλη σε κωδικοποίηση (text αντί bits). Στο παραπάνω παράδειγμα έχουμε στοιχεία με το όνομα message, to, from, subject, text, another_text. Το στοιχείο another_text είναι κενό (empty element) και αντί ζεύγους <another_text> (ανοίγοντος tag) και </another_text> (κλείνοντος tag), έχομε την συντομογραφία <another_text/>, πού ανοίγει και κλείνει συγχρόνως. Συνήθως, όπως παρατηρούμε και στο παραπάνω παράδειγμα, το περιεχόμενο του στοιχείου είναι άλλο (άλλα) ενθυλακωμένα στοιχεία, ή / και κείμενο (text). Ένα στοιχείο με περιεχόμενο μόνο κείμενο λέγεται text element. Επίσης για το περικλειόμενο κείμενο χρησιμοποιείται και ο ορισμός PCDATA (Parsed Character Data), καθόσον το κείμενο αυτό λαμβάνεται υπ΄όψιν σαν ενιαία και ξεχωριστή οντότητα από τους parsers.

Το πρώτο παράδειγμα είναι ένας κατάλογος διευθύνσεων (address book), το οποίο αποτελεί ένα πλήρες κείμενο XML (XML document).

```
<?xml version="1.0"?>
<addressbook>
 <!-- This is my good friend Bob. -->
 <contact>
  <name>Bob Rizzo</name>
  <city>New York</city>
  <address>1212 W 304th Street</address>
  <state>New York</state>
  <zip>10011</zip>
```

```
  <phone>
    <voice>212-555-1212</voice>
    <fax/>
  </phone>
  <email>frizzo@fruity.com</email>
  <web>http://www.fruity.com/rizzo</web>
  <company>Bob&amp;apos;s Ratchet Service</company>
 </contact>

 <!-- This is my old college roommate Peter. -->
 <contact>
  <name>Peter Rosenberg</name>
  <address>1162 E 412th Street</address>
  <city>New York</city>
  <state>New York</state>
  <zip>10011</zip>
  <phone>
    <voice>212-555-1818</voice>
    <fax>212-555-1819</fax>
  </phone>
  <email>srosenberg@fruity.com</email>
  <web>http://www.fruity.com/rosenberg</web>
  <company>Rosenberg&amp;apos;s Shoes &amp;amp; Glasses</company>
 </contact>
</addressbook>
```

Παράδειγμα 2.1 - Κείμενο addressbook.xml

Η δενδρική μορφή είναι προφανής και μπορούμε στον browser να ανοιγοκλείνουμε τα στοιχεία (φύλλα κάποιου βάθους). Παρατηρούμε επίσης πώς διαχωρίζονται τα σχόλια, πώς κλείνει ένα στοιχείο (element) που δεν έχει πληροφορία, (υπάρχει παραπάνω τέτοιο παράδειγμα), πώς γράφονται τα δεσμευμένα σύμβολα &, ‘. (το “ γίνεται " και τα <,> < , > αντίστοιχα). Ο browser απλώς αντιλαμβάνεται και αποκωδικοποιεί το συντακτικό της δομής και την παρουσιάζει σαν δένδρο. Παραβιάζοντας την σύνταξη βλέπουμε σχετικές παρατηρήσεις πού εκδίδει ο browser.

2.2.2 – Μεταβλητές ιδιοτήτων (attributes)

Κάθε ***στοιχείο*** (element) δυνατόν να έχει και ***ιδιότητες*** (attributes), οι μεταβλητές των οποίων πρέπει να περικλείονται σε απλά η διπλά quotes και μπορούν να εμπεριέχουν κενά και άλλα σημεία στίξης, και οι οποίες παρέχουν κάποιες περαιτέρω πληροφορίες για ένα στοιχείο. Σημειώνεται επίσης πως δεν υπάρχει κάποιος περιορισμός στον αριθμό των ιδιοτήτων ενός στοιχείου(element). Οι ιδιότητες μαζί με τις τιμές τους συντάσσονται σύμφωνα με το ακόλουθο παράδειγμα:

<movie type=”comedy” rating=”for adults” year=”1998”> …. </movie>

Το tag (movie) συνοδεύεται από ότι ο χρήστης θέλει να περιλάβει σαν ιδιότητες (comedy, rating, year). Κάθε μεταβλητή ιδιότητας φέρει σαν τιμή ένα text string. Η σημασία και χρήση των attributes είναι θέμα του χρήστη ή της εφαρμογής που επεξεργάζεται το XML. Κάθε μεταβλητή ιδιοτήτων θα μπορούσε να αποτελέσει υποστοιχείο (sub element), δηλ. το παραπάνω παράδειγμα θα μπορούσε να γραφεί σαν:

```
<movie>
        <type>comedy</type>
        <rating>for adults</rating>
        <year>1998</year>
</movie>
```

Το τι θα προτιμηθεί έγκειται συνήθως στην εμπειρία του προγραμματιστή, π.χ. ένα μακρύ κείμενο δεν θα ήταν κομψό ή πρακτικό να περιλαμβάνεται στις ιδιότητες. Γι’ αυτό συνήθως στην πράξη σε ένα έγγραφο XML ορίζουμε τις πιο σημαντικές πληροφορίες ως στοιχεία(elements) και τα λιγότερο σημαντικά ως ιδιότητες(attributes).

2.2.3 – Η σημασία και η χρήση των namespaces

Με τα namespaces εξασφαλίζεται ελευθερία και ανεξαρτησία στην επιλογή του κειμένου των tags, δηλαδή στην ονοματοδότηση των επί μέρους στοιχείων (elements) ανάμεσα σε διαφορετικά κείμενα. Ας υποθέσουμε ότι ένα στοιχείο υπάρχει σε διαφορετικά κείμενα με

το ίδιο όνομα. Δηλαδή έχομε <any_tag>text1</any_tag> στο doc1.xml και <any_tag>text2</any_tag> στο doc2.xml. Αν τώρα φέρουμε μαζί τα δύο αυτά στοιχεία, για παράδειγμα σε ένα τρίτο doc3.xml, δεν θα μπορούν να διαφοροποιούνται με βάση το όνομά τους. Η λύση έρχεται με ένα πρόθεμα και μία σύμβαση πού υποδηλοί το πού έχει ορισθεί το κάθε στοιχείο. Έτσι γράφουμε στο doc3.xml το tag του πρώτου σαν nsp1:any_tag και το tag του δεύτερου σαν nsp2:any_tag. Όλα τα στοιχεία πού ορίζονται τοπικά μέσα στο doc3.xml δεν φέρουν κανένα πρόθεμα, δηλαδή έχουν το default namespace. Επομένως δυνατόν να συνυπάρχουν και τα τρία tags: nsp1:any_tag, nsp2: any_tag και any_tag. Πώς όμως εξασφαλίζεται η μοναδικότητα των προθεμάτων πού επιλέγονται ανεξάρτητα σε διάφορα σημεία του κόσμου; Τα ίδια τα προθέματα nsp1, nsp2 ορίζονται αποκλειστικά στο doc3.xml και διασυνδέονται με κάποια URIs, των οποίων η μοναδικότητα είναι έτσι ή αλλιώς εξασφαλισμένη. Έτσι μπορούμε να ορίσουμε στην αρχή ενός XML document ονόματι vehicles.xml (μέσα στο opening tag αυτού) τα nsp1, nsp2

```
<vehicles xmlns:it= "http://www.italian_trolleys.com"
        xmlns:ru= "http://www.russian_trolleys.com">
```

και μετά να αναφερόμαστε στο κείμενο μας σε it:trolleys, ru:trolleys, τα οποία είναι διαφορετικά του δικού μας trolleys. Άρα τα namespaces δίδουν την δυνατότητα συσχέτισης στοιχείων ενός κειμένου XML με συγκεκριμένα URIs. Προσέξτε επίσης ότι αρκεί κάποιος ορισμός URIs στην αρχή του στοιχείου (συνήθως του root) όπου θα γίνει χρήση κάποιου namespace. Η χρήση κάποιου tag σαν it:trolleys, ru:trolleys μέσα στο κείμενο δεν απαιτεί σύνδεση με το αντίστοιχο URI, ούτε καν να είναι αυτό το URI υπαρκτό.

Αν κάποιος εμβαθύνει στην XML, μπορεί εύκολα να διαπιστώσει, πχ κατά τη δημιουργία διαδραστικών σελίδων XML, πως τα namespaces παρέχουν πολλές ευκολίες και είναι εξόχως σημαντικά. Μερικές από τις χρήσεις των namespaces είναι και οι εξής δύο:

•***Επαναχρησιμοποίηση***: Τα namespaces μπορούν να επιτρέψουν σε οποιοδήποτε αριθμό XML documents να αναφερθούν σε αυτά. Αυτό, επιτρέπει στα namespaces να επαναχρησιμοποιούνται όσο και όταν χρειάζεται και αποτρέπουν τους developers από το να κατασκευάζουν νέα namespaces για κάθε έγγραφο που δημιουργούν.

•***Πολλαπλότητα***: Με τον ίδιο τρόπο που πολλά XML documents μπορούν να αναφέρονται στο ίδιο namespace, έτσι και ένα έγγραφο (document) μπορεί να αναφέρεται σε περισσότερα του ενός namespaces. Αυτό είναι δευτερεύον αποτέλεσμα του διαχωρισμού elements σε λογικές, ταξινομημένες ομάδες.

2.2.4 – Άλλα περιεχόμενα στοιχείου

Πέραν του κειμένου, μπορούμε να περιλάβουμε οποιαδήποτε κωδικοποιημένη ή μη ψηφιακή πληροφορία (εικόνα, video), η οποία όμως περικλείεται από <![CDATA[..]]>. Ο όρος CDATA (Character Data) είναι σε αντίθεση με τον PCDATA (Parsed CDATA), δηλαδή η CDATA δεν υπόκειται σε parsing. Ο parser αγνοεί το περικλειόμενο, το οποίο διοχετεύεται όπου προβλέπει η εφαρμογή.

Ένα πλήρες κείμενο XML (XML document), είναι ένα αρχείο text (δημιουργημένο με Notepad, Word, κάποια εφαρμογή όπως το Visual Studio κλπ) το οποίο έχει την κατάληξη .xml. Περιέχει μία και μόνη δενδρική δομή, με ένα και μόνο στοιχείο ρίζας. Το όνομα του στοιχείου ρίζας (root element συμβολιζόμενο γενικά και με /) πρέπει να συμπίπτει με το όνομα του αρχείου. Έτσι κάθε κείμενο XML παρομοιάζεται με ένα σύστημα αρχειοθέτησης με root directory το .xml αρχείο, folders, subfolders τα elements και files το περιεχόμενο των elements. Αυτή η αναλογία βοηθά μνημοτεχνικά την κατανόηση της δομής του XML, χωρίς όμως περαιτέρω προεκτάσεις.

Ένα παράδειγμα εφαρμογής όσων αναλύσαμε παραπάνω είναι το ακόλουθο:

```
<?xml version="1.0"?>

<!--This is the way to write a comment-->
<vehicles>The root element (must be exactly one) is opened; this the first opening tag
   <cars type = "passenger car">
     <sedan no_doors="4"/><!--This was both an openening and a closing tag-->
     <jeep price="high">Another jeep defined below</jeep>
     <formula1 price= "very high">
            <![CDATA[<tag_to_be_ignored>Whatever placed here remains unparsed,
              </tag_to_be_ignored>, etc, qwerty, <, !, ""=,  ASDD, etc,...]]>
     </formula1>
```

```
    </cars>

    <trolleys/><!--only a placeholder - to be expanded later-->

    <ambulances type ="public use" importance="1">
            <?targetdatabase SELECT * FROM Hospitals?>
    </ambulances>

    <jeeps type="military car" importance="2">
       <jeep price="3000" colour="brown" user="army"/>
       <jeep price="5200" user="navy" colour="gray"/>
       <jeep price="2800" colour="blue" user="air force"/>
    </jeeps>

    <lories>
        Some text belonging to 'lories'
       <lory1 no_of_axes="2" tons="three">lory one text</lory1>
       <lory2 no_of_axes="6" tons="14">
            <traction motor="400PS" no_of_axes="3"/>
            <tender lenght="15.53m" no_of_axes="3"/>
       </lory2>
       More text belonging to 'lories', placed between its subelements
       <military_lory/>
    </lories>

</vehicles>
<!--the root element has just been closed; this is necessarily the last closing tag-->
```

Παράδειγμα 2.2 – Ένα πλήρες κείμενο XML

3

Το περιβάλλον .NET Framework και τα εργαλεία ανάπτυξης της εφαρμογής

3.1 Το περιβάλλον .NET

Το Microsoft .NET Framework είναι ένα software component που παρέχεται από τη Microsoft και περιέχει ένα μεγάλο αριθμό έτοιμων κλάσεων, μεθόδων και συναρτήσεων για κοινές προγραμματικές απαιτήσεις. Σχεδιάστηκε για να χρησιμοποιείται στις περισσότερες εφαρμογές που δημιουργούνται για να εκτελούνται σε περιβάλλον windows. Οι "έτοιμου κώδικα" λύσεις του καλύπτουν ένα πολύ μεγάλο φάσμα προγραμματιστικών αναγκών σε τομείς όπως η διεπαφή χρήστη (UI), η πρόσβαση σε δεδομένα, η σύνδεση με βάσεις δεδομένων, η κρυπτογραφία, η ανάπτυξη διαδικτυακών εφαρμογών και τα δίκτυα επικοινωνιών. Η βιβλιοθήκη των κλάσεων του .NET χρησιμοποιείται από τους προγραμματιστές που σε συνδυασμό με τα δικά τους τμήματα κώδικα μπορούν πολύ απλούστερα σε σχέση με το παρελθόν να παράγουν πολύπλοκες εφαρμογές .Τα προγράμματα που γράφονται για το .NET εκτελούνται σε ένα προγραμματιστικό περιβάλλον που διαχειρίζεται της απαιτήσεις του χρόνου εκτέλεσης των προγραμμάτων. Αυτό το περιβάλλον είναι μέρος του .NET Framework και είναι γνωστό ως CLR (Common Language

Runtime) και εξασφαλίζει την εμφάνιση μιας εικονικής μηχανής εφαρμογής, έτσι ώστε οι προγραμματιστές να μη χρειάζεται να σκέφτονται τις δυνατότητες που μπορεί να έχει ο υπολογιστής του κάθε χρήστη που θα εκτελέσει το πρόγραμμα. Για τις ανάγκες της διπλωματικής εγκαταστήσαμε την έκδοση 2.0 προκειμένου να υποστηρίζεται στο εργαλείο Visual Studio 2005 Professional Edition το οποίο και χρησιμοποιήσαμε για να φέρουμε εις πέρας την ολοκλήρωση της.

3.2 Διεπαφή χρήστη

Η διεπαφή χρήστη (UI, User Interface) υλοποιεί την αμφίδρομη επικοινωνία συστήματος-χρήστη (υπολογιστή-ανθρώπου). Παραδοσιακά, σε εφαρμογές επεξεργασίας δεδομένων ο κύκλος λειτουργίας του προγράμματος ήταν: είσοδος-επεξεργασία-έξοδος και το μεγαλύτερο τμήμα του κώδικα αφιερωνόταν στην επεξεργασία. Η είσοδος γινόταν αποκλειστικά από το πληκτρολόγιο. Ουσιαστικά, ο προγραμματιστής επινοούσε μια "γλώσσα" εισόδου που έπρεπε να κατέχει ο χρήστης και την οποία αποδεχόταν το σύστημα, ώστε να την επεξεργάζεται και να παράγει το αποτέλεσμα στη γλώσσα εξόδου που ήταν συνήθως γραμμογραφημένες εκτυπώσεις. Η διεπαφή χρήστη σε τέτοια συστήματα ήταν ρυθμού χαρακτήρων.

Σήμερα, στα πιο μοντέρνα προγράμματα για λόγους ευχρηστίας η διεπαφή βασίζεται σε γραφικά (GUI, Graphical User Interfaces). Η υλοποίηση GUI απαιτεί μεν περισσότερο κώδικα και η εκτέλεση τους καταναλώνει περισσότερους πόρους συστήματος, αλλά από την άλλη οι ενέργειες του εκάστοτε χρήστη καταδεικνύονται και δεν πληκτρολογούνται. Τα δεδομένα πληκτρολογούνται μόνο όταν αυτό είναι πιο πρακτικό από το να επιλέγονται. Γενικότερα, η χρήση GUI είναι η καλύτερη επιλογή για όχι και τόσο έμπειρους χρήστες. Παρ' όλα αυτά, για έμπειρους χρήστες κυρίως οι εφαρμογές ρυθμού χαρακτήρων όπως γράψαμε στην προηγούμενη παράγραφο εξακολουθούν να χρησιμοποιούνται ακόμα και σήμερα, όπως στα συστήματα κρατήσεων αεροπορικών εισιτηρίων. Τα κυριότερα χαρακτηριστικά που καθορίζουν τη χρηστικότητα μιας διεπαφής είναι:

- Συνέπεια
- Απλότητα
- Χρήση μεταφορών από την πραγματικότητα που μοντελοποιείται από το πρόγραμμα
- Ελαχιστοποίηση ενεργειών του χρήστη

• Άμεση ανάδραση

• Βοήθεια: Συμβουλές, μηνύματα, online βοήθεια ευαίσθητη στα συμφραζόμενα

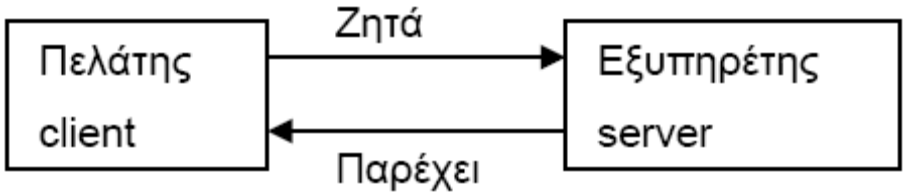
• Ελαχιστοποίηση απομνημόνευσης

• Κατανοητά μηνύματα λαθών: εξειδικευμένα, καθοδηγητικά, με θετικό τόνο

• Εναρμόνιση με προσλαμβάνουσες παραστάσεις χρήστη

• Ευκαμψία και προσαρμοστικότητα

Η διεπαφή χρήστη δεν πρέπει να σχετίζεται με την υλοποίηση της λειτουργικότητας. Τα δύο αυτά υποσυστήματα πρέπει να επικοινωνούν μέσω της δημόσιας επαφής (public interface) του δεύτερου. Έτσι είναι εύκολο να υλοποιηθούν διαφορετικές διεπαφές που να επιτρέπουν στους χρήστες να προσπελαύνουν τις λειτουργίες με διαφορετικό τρόπο. Μια από αυτές είναι το μοντέλο πελάτη-εξυπηρετητή (client-server). Ένα άρθρωμα υλοποιεί συγκεκριμένη λειτουργικότητα και την προσφέρει ως υπηρεσία σε καταναλωτές (έναν ή περισσότερους). Χρησιμοποιείται ιδιαίτερα όταν ο πελάτης και ο εξυπηρετητής δεν βρίσκονται στην ίδια μηχανή, όπως στην περίπτωση κατανομής, για παράδειγμα σε ένα σύστημα βάσεων δεδομένων και για την περίπτωση της διπλωματικής αυτής .



Σχήμα 3.1 – Το μοντέλο πελάτη-εξυπηρετητή

Η ανάπτυξη ξεκινά με τη σχεδίαση φορμών (Windows Forms για την περίπτωση μας), την τοποθέτηση των χειριστηρίων (controls) και τη ρύθμιση των ιδιοτήτων τους. Το αρχικό αυτό πρωτότυπο αποτελείται από dummy screens και δεν υλοποιεί καμία λειτουργικότητα παρά μόνον τη διεπαφή χρήστη. Το πρωτότυπο αυτό εξελίσσεται ως προς την αισθητική και την ευχρηστία του και έπειτα προστίθεται κώδικας πίσω από τις οθόνες που προσδίδει συμπεριφορά στα διάφορα χειριστήρια που είναι τοποθετημένα πάνω τους.

Τα χειριστήρια (controls) είναι επαναχρησιμοποιούμενα στοιχεία της διεπαφής χρήστη με τυποποιημένη, αλλά προσαρμοσμένη συμπεριφορά. Η συμπεριφορά τους αλλάζει

μέσω των ιδιοτήτων (properties) τους που μπορούν να τεθούν στις αρχικές τους τιμές πάνω στον πίνακα ιδιοτήτων, ενώ αρκετές μπορούν να αλλάζουν και κατά τη διάρκεια εκτέλεσης του προγράμματος. Το κάθε χειριστήριο έχει συνδεθεί με ένα σύνολο συμβάντων, για παράδειγμα με το πάτημα του πλήκτρου ενός ποντικιού ή με την εστίαση (focus). Όταν λάβει χώρα το συμβάν εκτελείται και ο κώδικας που έχει γραφτεί γι' αυτό.

3.3 Εργαλεία συστήματος

3.3.1 - Η γλώσσα προγραμματισμού C#

Η C# είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού που αναπτύχθηκε από τη Microsoft ως μέρος του .NET περιβάλλοντος ανάπτυξης εφαρμογών. Η C# συνδυάζει χαρακτηριστικά από διάφορες καταξιωμένες γλώσσες προγραμματισμού, όπως η Delphi και η Java, με έμφαση στην αποδοτικότητα και την απλότητα. Πρόκειται για μια σύγχρονη, γενικού σκοπού, απλή, αλλά και ισχυρή γλώσσα προγραμματισμού, ενώ είναι ιδανική για την ανάπτυξη εφαρμογών σε κατανεμημένα περιβάλλοντα, όπως το Internet. Πολλοί άνθρωποι πιστεύουν πως ο ρόλος που έπαιξε η C# στην αρχιτεκτονική του περιβάλλοντος .NET είναι πλήρως αντίστοιχος με αυτόν της C στην εφαρμογή του UNIX. Αν κάποιος γνωρίζει ήδη μια γλώσσα προγραμματισμού όπως C, C++ ή Java θα δει πως η σύνταξη της C# του είναι ήδη οικεία γιατί χρησιμοποιεί τα ίδια σύμβολα "{" και "}" που υποδηλώνουν τα τμήματα του κώδικα. Η σύνταξη της δε έχει πάρει αρκετά στοιχεία από την C++.

Ο δημιουργός της C# είναι ο Anders Hejslberg[10].Η προηγούμενη εμπειρία του σε γλώσσες προγραμματισμού και πλαίσια εργασίας (Visual J++, Borland Delphi, Turbo Pascal) μπορεί να διακριθεί ολοφάνερα στον τρόπο σύνταξης της γλώσσας ως επίσης και καθ' ολοκληρίαν του πυρήνα CLR[11]. Σε συνεντεύξεις και δημοσιεύσεις του έχει δηλώσει πως τα ψεγάδια στις άλλες γλώσσες τον οδήγησαν στην θεμελίωση του CLR και κατ' επέκταση στη δημιουργία και το σχεδιασμό της γλώσσας αυτής .

Για τη δημιουργία της γλώσσας έπρεπε να εκπληρώνονται κάποιοι στόχοι, οι οποίοι ήταν:

- Έπρεπε σκόπιμα να είναι μια απλή, μοντέρνα, γενικού σκοπού και αντικειμενοστραφής γλώσσα προγραμματισμού.
- Λόγω της στιβαρότητας, αντοχής των προγραμμάτων και λαμβάνοντας υπόψιν την παραγωγικότητα των προγραμματιστών ως πολύ σημαντικά θέματα, η γλώσσα που θα κατασκευαζόταν θα έπρεπε να περιέχει δυνατό έλεγχο τύπων, έλεγχο στα όρια των πινάκων, να ελέγχει συνεχώς για κάθε προσπάθεια τοποθέτησης των μεταβλητών, φορητότητα πηγαίου κώδικα και αυτόματο garbage collection.
- Η γλώσσα επίσης θα ήταν σκόπιμο να χρησιμοποιηθεί στο πεδίο της ανάπτυξης εφαρμογών ώστε να εκμεταλλεύεται τα κατανεμημένα περιβάλλοντα.
- Οι προγραμματιστές που ήδη γνώριζαν μια γλώσσα παρεμφερή όπως η C και η C++ να μη χρειαζόντουσαν παραπάνω χρόνο ώστε να μάθουν μια νέα γλώσσα από την αρχή.
- Υποστήριξη διεθνοποίησης της (internationalization).
- Αν και υπήρχε σκοπός οι εφαρμογές σε γλώσσα C# να είναι πολύ πιο οικονομικές σε πόρους και σε απαιτήσεις συστήματος για την εκτέλεση των, η γλώσσα δεν ήταν σκόπιμο να συγκριθεί άμεσα με την αποδοτικότητα και το μέγεθος των γλωσσών C και assembly.

Παράλληλα στο .NET Framework που βρίσκονται οι βιβλιοθήκες υπάρχει μια πληθώρα κλάσεων για κάθε δουλειά που είναι πολύ καλά οργανωμένες σε χώρους ονομάτων (namespaces), έτσι ώστε με λίγη διαίσθηση να βρίσκει κανείς την κλάση που του χρειάζεται.

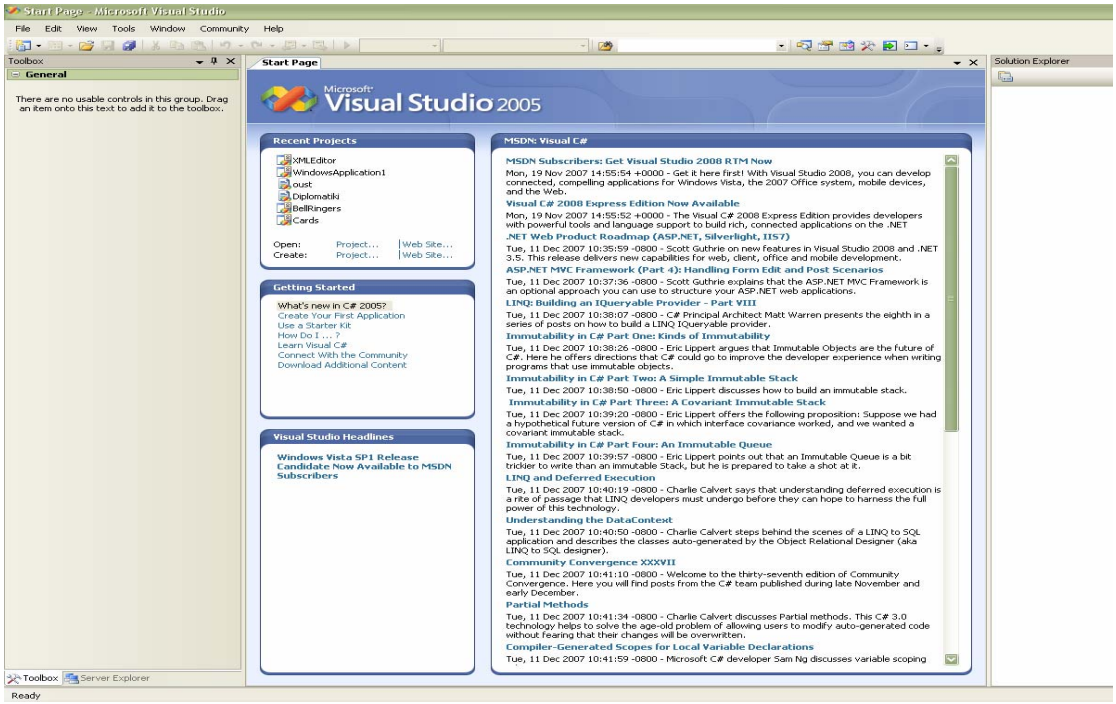
Για περισσότερες πληροφορίες σχετικά με τη σύνταξη και τη λογική της C# μπορεί κάποιος να ανατρέξει στη βιβλιογραφία που παρατίθεται στο τέλος της εργασίας .

3.3.2 - Το πρόγραμμα Visual Studio Professional Edition

Το Visual Studio 2005 Professional Edition σχεδιάστηκε ειδικά για επαγγελματίες προγραμματιστές που εργάζονται μόνοι ή σε μικρές ομάδες. Προσφέρει πλήρη πρόσβαση στα Microsoft.NET Framework 2.0, ένα ισχυρό, πλήρως λειτουργικό περιβάλλον ανάπτυξης

υποστήριξης για εργαλεία προγραμματισμού που επεκτείνουν το ολοκληρωμένο περιβάλλον ανάπτυξης του Visual Studio και εργαλεία δημιουργίας εφαρμογών Windows και Web με multi-tier αρχιτεκτονική. Επίσης ο χρήστης μπορεί να βρει και λειτουργίες όπως η βελτιωμένη λειτουργία Edit and Continue, μια λειτουργία εξοικονόμησης χρόνου, που του επιτρέπει να εντοπίζει και να διορθώνει γρήγορα τα σφάλματα χωρίς να χρειάζεται να κάνει επανεκκίνηση των εκάστοτε εφαρμογών του.Το πολύ χρήσιμο αυτό εργαλείο μας επιτρέπει να:

- ✓ Εργαζόμαστε σε ένα περιβάλλον υψηλής παραγωγικότητας με εργαλεία editing και debugging καθώς και με έξυπνους visual designers που μας επιτρέπουν να δημιουργούμε γρήγορα εφαρμογές για Windows, Web και φορητές συσκευές.
- ✓ Αξιοποιήσουμε την υποστήριξη γλωσσών προγραμματισμού της Microsoft και άλλων κατασκευαστών για να δημιουργήσουμε εφαρμογές με υψηλές επιδόσεις.
- ✓ Εκμεταλλευτούμε την αποδοτικότητα του .NET Framework 2.0, μιας ισχυρής επιχειρηματικής πλατφόρμας για να αναπτύξουμε εφαρμογές Windows και Web με υψηλές επιδόσεις. Το .NET Framework μειώνει σημαντικά τον όγκο του κώδικα που πρέπει να γράψουμε, συχνά έως και 50 τοις εκατό για τις κοινές εργασίες προγραμματισμού.
- ✓ Μετατρέπουμε αυτόματα πλήρεις εφαρμογές των Windows στο Internet, σε τοπικά δίκτυα ή σε CD με ένα απλό κλικ του ποντικιού χάρη στην ενσωματωμένη υποστήριξη ClickOnce.
- ✓ Χρησιμοποιούμε ένα περιβάλλον σχεδίασης δεδομένων με drag and drop**,** για να σχεδιάσουμε οπτικά τις τοπικές και απομακρυσμένες βάσεις δεδομένων και τα queries χωρίς να χρειάζεται να συντάξουμε κώδικα.
- ✓ Απλοποιήσουμε την κατασκευή εφαρμογών για φορητές συσκευές Windows CE συμπεριλαμβανομένων ορισμένων από τα πιο δημοφιλή PDA και Smartphone, με πλήρη υποστήριξη για native ή managed κώδικα (.NET Compact Framework 2.0).



Σχήμα 3.2 – Η αρχική σελίδα του Visual Studio 2005 Professional Edition

Όπως αναφέραμε και προηγουμένως, ένα από τα κύρια πλεονεκτήματα που η Microsoft πρόσφερε στους developers μέσω του IDE είναι το περιβάλλον σχεδίασης δεδομένων με drag and drop. Στην πραγματικότητα, σε παλιότερες εκδόσεις της γλώσσας προγραμματισμού Visual Basic οι developers μπορούσαν να προσθέτουν μη-γραφικά στοιχεία όπως για παράδειγμα οι timers.Αυτά τα πλήρως γραφικά στοιχεία είναι όλα διαθέσιμα σε ένα ειδικό παράθυρό του Visual Studio που λέγεται Toolbox.

3.3.3 - ASP.NET

Η ASP.NET είναι ένα framework για διαδικτυακές εφαρμογές και δημιουργήθηκε από τη Microsoft έτσι ώστε ο σχεδιασμός "δυναμικών" ιστοσελίδων και εφαρμογών καθώς και των XML Web Services να γίνει το δυνατό απλούστερος. Αποτελεί αναπόσπαστο μέρος της πλατφόρμας .NET και είναι ο διάδοχος της τεχνολογίας ASP(Active Server Pages). Είναι δομημένη κατά το πρότυπο CLR(Common Language Runtime) και επιτρέπει στους προγραμματιστές να γράφουν ASP.NET κώδικα σε οποιαδήποτε γλώσσα που είναι συμβατή με από το πλαίσιο εργασίας .NET .

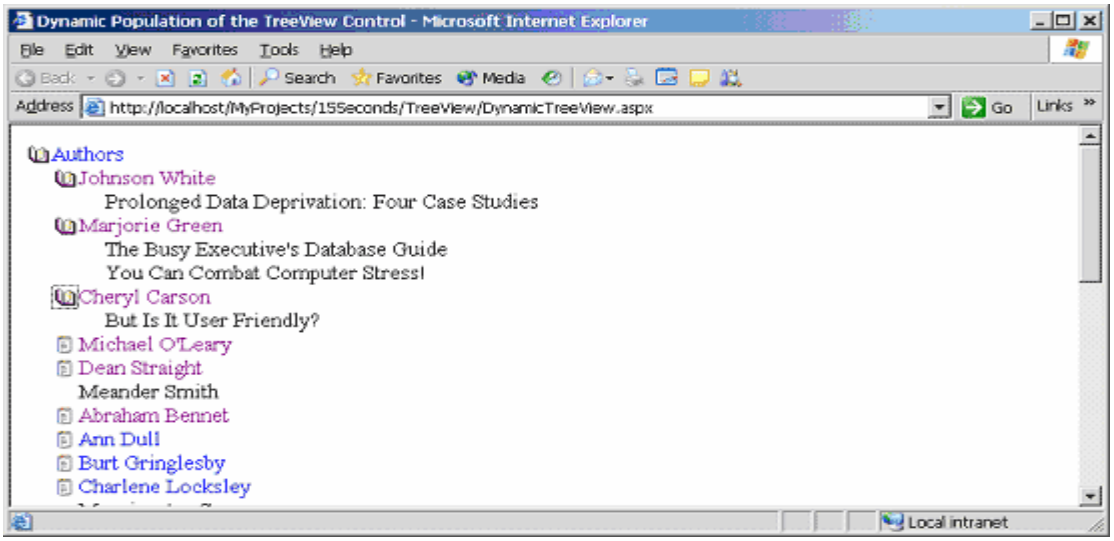
Στην προηγούμενη παράγραφο είδαμε πως η ASP.NET αποτελεί το διάδοχο της ASP. Κι αυτό γιατί η Microsoft με την έλευση του .NET framework έπρεπε να αναβαθμίσει το ASP framework κι έτσι γεννήθηκε η ASP.NET. Τα βασικά χαρακτηριστικά της συμπεριλαμβάνουν τα παρακάτω:

- Ένα μοντέλο λογικοποιημένου προγράμματος που χρησιμοποιεί Web Forms όπου περιέχει λογική παρουσίαση και αρχεία κώδικα που είναι ανεξαρτητοποιημένα από τη λεγόμενη business logic. Μπορεί κανείς να γράψει κώδικα σε οποιαδήποτε γλώσσα προγραμματισμού που υποστηρίζει το .NET συμπεριλαμβανομένου φυσικά και της C#.
- Δυναμικά στοιχεία ελέγχου(data controls) για απεικόνιση, μετατροπή και υποστήριξη δεδομένων από μια βάση δεδομένων.
- Επιλογές για caching client state χρησιμοποιώντας cookies στον υπολογιστή του πελάτη(client) σε μια ειδική υπηρεσία (ASP.NET State Service) πάνω στον εκάστοτε browser ή σε μια SQL βάση δεδομένων.

Με την κυκλοφορία του Visual Studio 2005 που βασίζεται στο .NET 2.0 ,η κατασκευάστρια εταιρεία Microsoft εμπλούτισε ακόμα περισσότερο την ASP.NET. Έχει λάβει χώρα ένας μεγάλος αριθμός βελτιώσεων έτσι ώστε να βελτιστοποιείται η παραγωγή και η συντηρησιμότητα μιας ιστοσελίδας. Συγκεκριμένα, οι τρεις κυριότερες βελτιώσεις είναι:

- Εμπλουτισμένος σχεδιασμός των ιστοσελίδων με χρήση Master Pages, Themes και Web Parts. Ο χρήστης μπορεί, χρησιμοποιώντας το Master Pages, αρκετά γρήγορα να προωθήσει ένα κοινό layout για όλες τις Web pages της εφαρμογής. Τα Themes βοηθούν στην υλοποίηση μιας συνεπούς όψης της κατασκευαζόμενης ιστοσελίδας καθώς σιγουρεύουν πως όλα τα στοιχεία ελέγχου εμφανίζονται κατά τον ίδιο τρόπο όποτε χρειάζεται. Τέλος τα Web Parts δίνουν την ικανότητα να δημιουργηθούν αρθρωτές ιστοσελίδες που οι χρήστες μπορούν να προσαρμόσουν στις δικές τους ανάγκες.
- Νέα data source controls που επιτρέπουν να κατασκευάζονται εφαρμογές που μπορούν να εμφανίζονται και να επεξεργάζονται γρήγορα και εύκολα τα δεδομένα. Τα data source controls μπορούν να λειτουργήσουν με μια πληθώρα από πηγές δεδομένων, όπως Microsoft Access, SQL Server, XML αρχεία και Web Services.
- Νέα και αναβαθμισμένα στοιχεία ελέγχου(controls) για απεικόνιση και επεξεργασία δεδομένων όπως τα GridView, DetailsView και FormView. Κάθε χρήστης μπορεί

επίσης να χρησιμοποιήσει το TreeView control για να εμφανίσει στην οθόνη του τα δεδομένα σε δενδρική μορφή.



Σχήμα 3.3 – Αναπαράσταση του TreeView control

4

Η εφαρμογή XML Editor

4.1 Σχεδιασμός της εφαρμογής XML Editor

Στόχος μας ήταν η δημιουργία μιας δυναμικής ιστοσελίδας .aspx γραμμένης σε γλώσσα προγραμματισμού C# με ένα GUI που «κινεί» την εφαρμογή. Τα δεδομένα εισάγονται, επεξεργάζονται και τέλος αποθηκεύονται από το χρήστη ενώ θα υπάρχει και η δυνατότητα αναίρεσης επιλογών του κατά την εκτέλεση της εφαρμογής. Ο XML Editor θα πρέπει να έχει κάποια σημαντικά χαρακτηριστικά όπως:

- Να ελέγχει το είδος των αρχείων προς επεξεργασία. Να γίνονται δεκτά μόνο τα .xml αρχεία και όχι οποιαδήποτε άλλη μορφή αρχείου.
- Να διαβάζει το αρχείο xml από κάποιο φάκελο αρχείων (folder) και στη συνέχεια να αποτυπώνει το αρχείο σε δεντρική μορφή, προκειμένου να είναι εύκολος ο διαχωρισμός των elements, attributes καθώς και των τιμών τους.
- Να μπορεί ο χρήστης με το πάτημα του κέρσορα του ποντικιού πάνω σε ένα στοιχείο του δέντρου να επιλέξει τον προς διαμόρφωση ή διαγραφή κόμβο ή να μπορεί να προσθέσει κάποιον άλλο κόμβο στον επιλεγμένο.

- Να έχει δυνατότητα αναίρεσης ανεπιθύμητων ή λάθος αλλαγών από το χρήστη και να επιστρέφει το προς επεξεργασία xml αρχείο στην αρχική του μορφή. Προφανώς για μικροαλλαγές ενός-δύο βημάτων θα μπορεί ο χρήστης να διορθώνει το λάθος του μέσω του πλήκτρου «Πίσω» (Back) του browser του.
- Να έχει δυνατότητα αποθήκευσης του διαμορφωθέντος αρχείου xml.

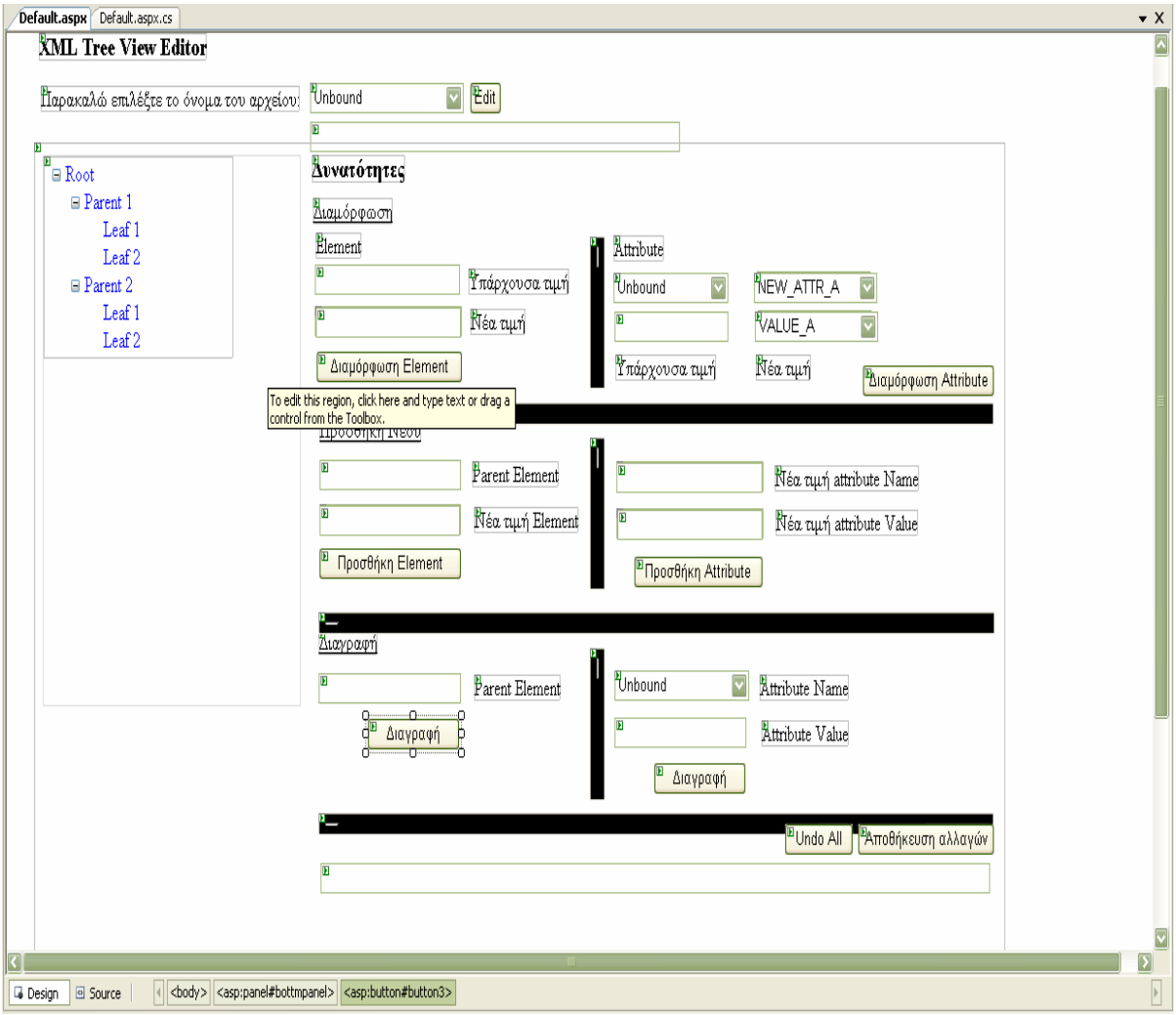
Στο παρόν κεφάλαιο μπορούμε να δούμε πως υλοποιούνται οι παραπάνω προϋποθέσεις, παραθέτοντας και τον απαραίτητο κώδικα για κάθε περίπτωση.

4.2 Υλοποίηση του XML Editor

4.2.1 - Γραφική υλοποίηση του προγράμματος

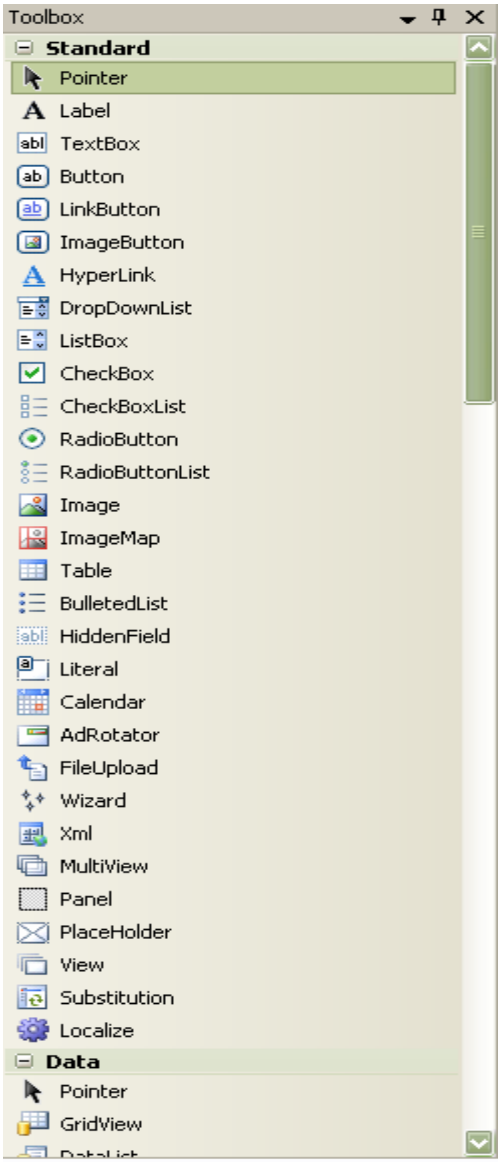
Στη συνέχεια παρατίθεται η υλοποίηση της εφαρμογής του XML Editor. Με βάση όσα αναφέραμε προηγουμένως, το πρόγραμμα μας θα αποτελείται από κάποια controls τα οποία έχουν συγκεκριμένο ρόλο στη σωστή εκτέλεση της εφαρμογής. Το καθένα απ' αυτά είναι υπεύθυνο για κάποια συγκεκριμένη λειτουργία, η οποία αποτυπώνεται στο πρόγραμμα μας με ένα κομμάτι κώδικα. Το σύνολο των λειτουργιών όλων των controls σε συνδυασμό με τη συγγραφή κώδικα για κάθε χαρακτηριστικό του XML Editor, περιορισμό ή δυνατότητα δηλαδή, μας δίνει τον τελικό κώδικα του προγράμματος.

Η γραφική υλοποίηση του XML Editor απεικονίζεται παρακάτω:



Σχήμα 4.1 – Γραφική υλοποίηση (design view) του XML Editor

Κοιτώντας αναλυτικότερα το σχήμα 4.1 μπορούμε να διακρίνουμε όλα τα controls από τα οποία αποτελείται η εφαρμογή. Τα controls μπορεί κανείς να τα επιλέξει από την καρτέλα του Visual Studio 2005 που λέγεται Toolbox (σχήμα 4.2).



Σχήμα 4.2 – Η εργαλειοθήκη Toolbox του Visual Studio 2005 Professional Edition

Συγκεκριμένα, στην τελική της μορφή, η εφαρμογή αποτελείται από τα εξής στοιχεία ελέγχου:

- ✓ Το TreeView Web Server Control που χρησιμοποιείται για να απεικονίζει κάποια δεδομένα κατανεμημένα ιεραρχικά, όπως ένα αρχείο xml στην περίπτωση μας, χάρη στο αυτόματο data binding που διαθέτει το οποίο επιτρέπει στους κόμβους να ταξινομούνται σωστά με βάση την ιεραρχία τους. Όταν ο χρήστης επιλέγει κάποιο κόμβο προς τροποποίηση του, ο κόμβος εκείνος θα χρωματίζεται με διαφορετικό χρώμα σε σχέση με τους υπόλοιπους κόμβους.

- ✓ Το Panel Web Server Control που λειτουργεί ως container εντός της σελίδας για άλλα controls. Στην περίπτωση μας το ένα από τα δύο Panel Web Server Control εμπεριέχει και συγχρόνως εμφανίζει την ανάπτυξη όλης της εφαρμογής ενώ το άλλο περιέχει το TreeView Web Server Control, το οποίο και αποτυπώνει κάθε φορά που εκτείνουμε ή αποκρύπτουμε τους κόμβους του xml δέντρου.

- ✓ Το DropDownList Web Server Control που επιτρέπει στους χρήστες να επιλέγουν τιμές από κάποια προκαθορισμένη λίστα. Μέχρι ο χρήστης να πατήσει το βελάκι δίπλα στο control αυτό, τα στοιχεία της λίστας παραμένουν κρυφά και αυτό ακριβώς είναι το πλεονέκτημα του σε σχέση με το ListBox Web Server Control.

- ✓ Το TextBox Web Server Control εξασφαλίζει ένα τρόπο για τους χρήστες να τυπώσουν πληροφορίες σε μια Web Form Page όπως κείμενο και αριθμούς. Στην περίπτωση μας θα χρησιμεύσει για την εμφάνιση των ονομάτων των κόμβων και την επεξεργασία τους.

- ✓ Το Label Web Server Control αποτελεί ένα εργαλείο που δίνει τη δυνατότητα στον προγραμματιστή να απεικονίζει στην οθόνη κείμενο που δε μεταβάλλεται και μπορεί να αλλαχθεί προγραμματιστικά όπως και όταν αυτός επιθυμεί.

- ✓ Το Image Web Server Control πραγματοποιεί την απεικόνιση εικόνων σε μια ASP.NET ιστοσελίδα που μπορούν να επεξεργαστούν από τον προγραμματιστή με τον δικό του τρόπο.

- ✓ Τέλος, το σημαντικότατο Button Web Server Control που δίνει τη δυνατότητα στον developer να στέλνει εντολές στο πρόγραμμα. Τα Buttons υποβάλλουν την ιστοσελίδα στον server και προωθούν κάποιες διαδικασίες αναμένοντας την ενεργοποίηση κάποιων event όπως το Click event. Στην περίπτωση αυτή

όταν ο χρήστης «κλικάρει» το Button τότε ενεργοποιούνται οι διαδικασίες που περιγράφονται στη συνάρτηση.

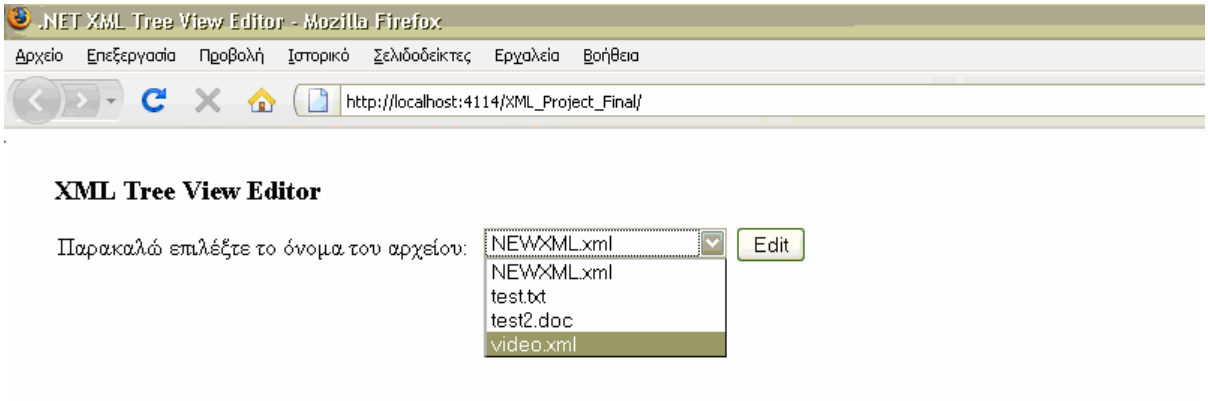
4.2.2 - Υλοποίηση βασικών συναρτήσεων

Μετά τη γνωριμία με τα controls που θα χρησιμοποιηθούν για την τελική πραγματοποίηση της εφαρμογής μας, μπορούν να επιδειχθούν οι βασικότερες συναρτήσεις που εκτελούνται κατά την εκτέλεση του προγράμματος. Αυτές είναι οι εξής:

α) Συναρτήσεις που ενεργοποιούνται μέχρι τη στιγμή της μετατροπής των αρχείων

Η πρώτη συνάρτηση που ενεργοποιείται στο πρόγραμμα μας είναι η *Page_Load*. Με την κλήση της συνάρτησης μπορούμε να δούμε στην αρχική σελίδα της εφαρμογής μας πως αυτή «γεμίζει» το στοιχείο ελέγχου dropdown list (Σχήμα 4.3).

```
protected void Page_Load(object sender, EventArgs e)
    {
        //Perorm this only once
        if (xmlFileNames.Items.Count == 0)
        {
            //get whatever is on that directory - FULL paths
            string[] filePaths =
Directory.GetFiles(@"/XML_Project_Final/XMLData");

            // get just path names
            // and populate dropdown with filenames
            string[] fileNames = new string[filePaths.Length];
            for (int i = 0; i<filePaths.Length; i++ )
            {
                fileNames[i] = Path.GetFileName(filePaths[i]);
                xmlFileNames.Items.Add(fileNames[i]);
            }
        }
    }
```

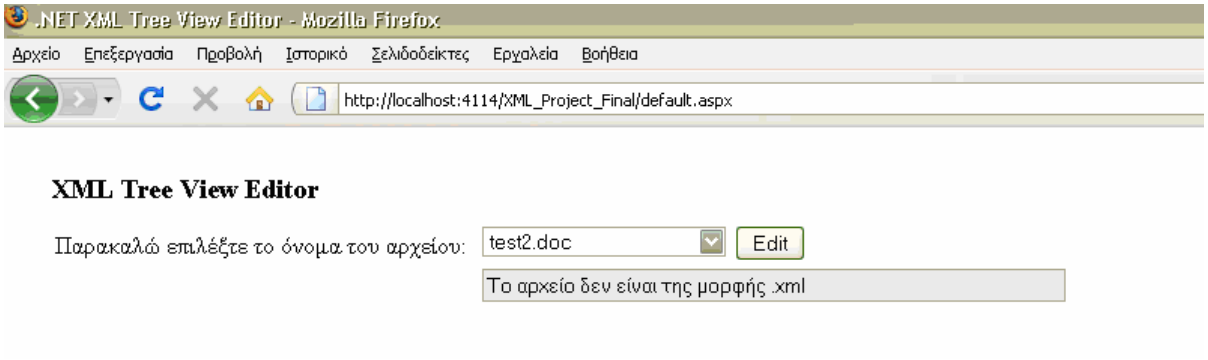


Σχήμα 4.3 – Η αρχική σελίδα του XML Editor

Στη συνέχεια «κλικάροντας» το κουμπί Edit διασφαλίζεται ότι το προς μετατροπή αρχείο από το φάκελο των αρχείων μας θα είναι μορφής xml, αλλιώς εμφανίζεται μήνυμα που γράφει στο χρήστη πως το αρχείο δεν είναι μορφής xml και κατά συνέπεια πρέπει να επιλέξει άλλο αρχείο από τη λίστα (Σχήμα 4.4). Αυτό επιτυγχάνεται με τη χρήση της συνάρτησης *Button1_Click* . Η συνάρτηση αυτή, αφού πρώτα ελέγχει την εγκυρότητα του αρχείου, δημιουργεί ένα DOM document και αφού πληρούνται πλέον όλες οι προϋποθέσεις μπορεί να συνεχιστεί η εκτέλεση του προγράμματος.

```
protected void Button1_Click(object sender, EventArgs e)
    {
        errorMsg.Visible = false;
        bottmPanel.Visible = false;

        //check if the selected is a .xml file
        if (!xmlFileNames.SelectedItem.Text.EndsWith("xml"))
        {
            errorMsg.Text = "Το αρχείο δεν είναι της μορφής .xml";
            errorMsg.Visible = true;
        }
        else
        {
            try
            {
                // SECTION 1. Create a DOM Document and load the XML data
into it.
                XmlDocument dom = new XmlDocument();
                dom.Load(@"/XML_Project_Final/XMLData/" +
xmlFileNames.SelectedItem.Text);

                // SECTION 2. Initialize the TreeView control.
                treeXML.Nodes.Clear();
                treeXML.Nodes.Add(new TreeNode(dom.DocumentElement.Name));
                TreeNode tNode = new TreeNode();
                tNode = treeXML.Nodes[0];
```

```
                // SECTION 3. Populate the TreeView with the DOM nodes.
                AddNode(dom.DocumentElement, tNode);
                treeXML.ExpandAll();
            }
            catch (XmlException xmlEx)
            {
                errorMsg.Text = xmlEx.Message;
                errorMsg.Visible = true;

            }
            catch (Exception ex)
            {
                errorMsg.Text = ex.Message;
                errorMsg.Visible = true;

            }
            //If you pass all tests, show panel with editor
            bottmPanel.Visible = true;
        }
    }
```

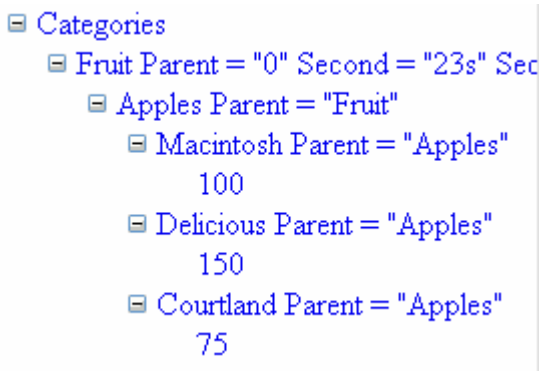


Σχήμα 4.4 – Το επιλεγμένο αρχείο δεν είναι xml και πρέπει να επιλεγεί άλλο από τη λίστα

Στο σημείο αυτό ενεργοποιείται μια άλλη συνάρτηση, η *AddNode* η οποία εμφανίζει στον χρήστη το xml αρχείο σε δεντρική μορφή, δείχνοντας του ξεκάθαρα όλα τα elements, τα attributes καθώς και τις τιμές των attributes όπου αυτές υπάρχουν (Σχήμα 4.5).

```
private void AddNode(XmlNode inXmlNode, TreeNode inTreeNode)
    {
        XmlNode xNode;
        TreeNode tNode;
        XmlNodeList nodeList;
        int i;

        // Loop through the XML nodes until the leaf is reached.
        // Add the nodes to the TreeView during the looping process.
```

```
        if (inXmlNode.HasChildNodes)
        {
            nodeList = inXmlNode.ChildNodes;
            for (i = 0; i <= nodeList.Count - 1; i++)
            {
                xNode = inXmlNode.ChildNodes[i];

                string NodeString = xNode.Name;

                //Check for attributes if any
                if (xNode.Attributes != null)
                {
                    int attrNum = xNode.Attributes.Count;
                    for (int count = 0; count < attrNum; count++)
                    {
                        string atName = xNode.Attributes[count].Name;
                        string atValue = xNode.Attributes[count].Value;

                        //and apend to NodeString
                        NodeString = NodeString + " " + atName + " = \"" +
atValue + "\"";
                    }
                }

                //Now add to tree
                inTreeNode.ChildNodes.Add(new TreeNode(NodeString));

                tNode = inTreeNode.ChildNodes[i];
                AddNode(xNode, tNode);
            }
        }
```

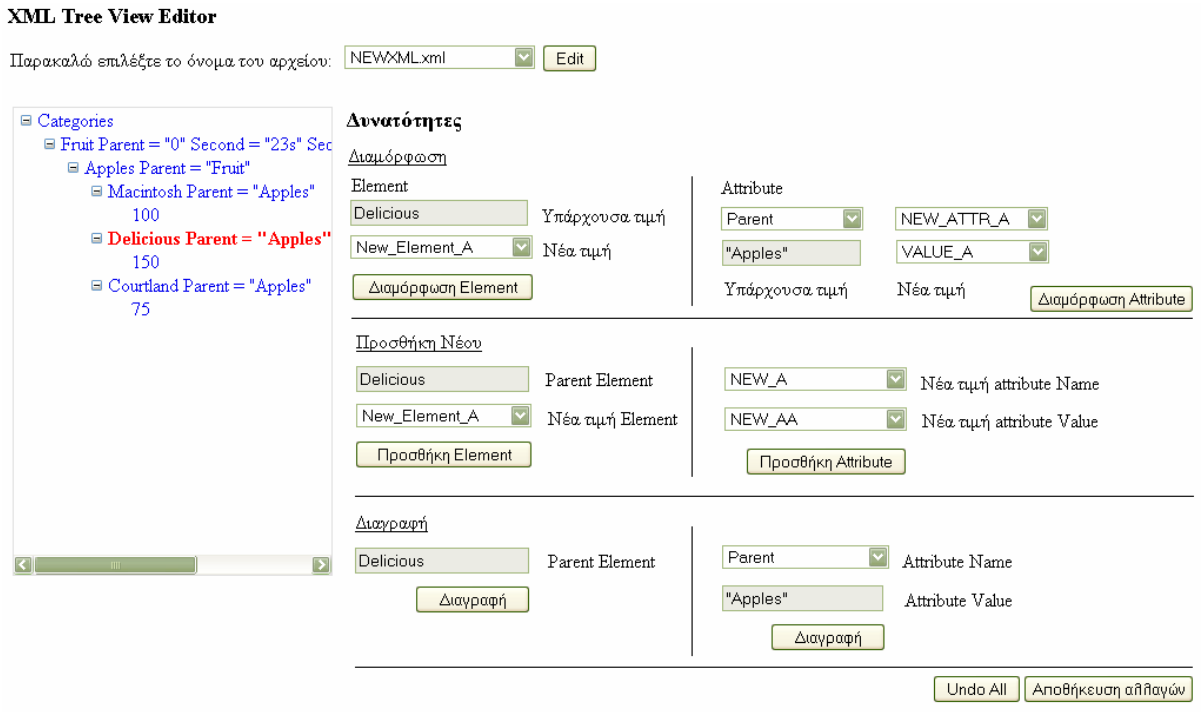


Σχήμα 4.5 – Δεντρική μορφή αρχείου xml

Πλέον ο χρήστης είναι έτοιμος να επιλέξει κάποιο κόμβο του δέντρου και να προχωρήσει στη διαδικασία της μετατροπής του αρχείου. Οπότε πρέπει κατά την επιλογή ενός κόμβου του δέντρου να «τρέχει» μια ακόμα η συνάρτηση, η *sthchanged* ο κώδικας της οποίας ακολουθεί παρακάτω:

```
protected void sthchanged(object sender, EventArgs e)
    {
        //0. Clear all boxes and dropdowns + visibility
        edit_Attr_existing_name_dd.Items.Clear();
        delete_attr_name_drop.Items.Clear();
        editableBox1.Visible = false;
        edit_element_new_dropdown.Visible = true;
        edit_element_new_dropdown.SelectedIndex = 0;
        infoBox.Visible = false;
        edit_new_attr_name_dropdown.Visible = true;
        edit_new_attr_value_dropdown.Visible = true;
        edit_new_attr_name_dropdown.SelectedIndex = 0;
        edit_new_attr_value_dropdown.SelectedIndex = 0;

        edit_Attr_new_name_text.Visible = false;
        edit_Attr_new_value_text.Visible = false;
        edit_Attr_existing_value_text.Text = "";
        add_New_Element_Text.Visible = false;
        add_New_Element_Text.Text = "";
        add_element_new_dropdown.Visible = true;
        add_element_new_dropdown.SelectedIndex = 0;
        add_attr_name_text.Visible = false;
        add_attr_value_text.Visible = false;
        add_element_new_attributeName_dropdown.Visible = true;
        add_element_new_attributeName_dropdown.SelectedIndex = 0;
        add_element_new_attributeValue_dropdown.Visible = true;
        add_element_new_attributeValue_dropdown.SelectedIndex = 0;
        delete_attr_value_txt.Text = "";


        //1. Get the  element of selected line
        string selectedElement = getElement(treeXML.SelectedValue);

        //2. And populate all controls that deal with element
        edit_element_existing_name.Text = selectedElement;
        add_parent_element_text.Text = selectedElement;
        delete_Element_text.Text = selectedElement;

        //3. Get all attributes - if any and populate a [] string
        string[] sa = getAttributes(treeXML.SelectedValue);

        //4. populate all dropdowns with attribute names
        populateDropDownWithAtributes(delete_attr_name_drop, sa);
        populateDropDownWithAtributes(edit_Attr_existing_name_dd, sa);

        //5. populate all textboxes with attribute values of selected
attribute
        findAttributeValueForSelectedName(edit_Attr_existing_name_dd,
edit_Attr_existing_value_text, sa);
        findAttributeValueForSelectedName(delete_attr_name_drop,
delete_attr_value_txt, sa);
    }
```

Όπως μπορούμε να δούμε, με την κλήση της συνάρτησης ο επιλεγμένος κόμβος του δέντρου χρωματίζεται διαφορετικά από τους υπολοίπους ενώ ταυτόχρονα «γεμίζουν» όλα τα στοιχεία ελέγχου (controls) με τα ονόματα των σχετιζόμενων elements και attributes (Σχήμα 4.6).



Σχήμα 4.6 – Επιλογή κόμβου με element→Delicious, attribute→Parent και attribute value→“Apples”

β) Συναρτήσεις επεξεργασίας xml αρχείων

Εκτός από τις προαναφερόμενες βασικές συναρτήσεις που εκτελούνται κάθε φορά που «τρέχει» το πρόγραμμα, υπάρχουν και συναρτήσεις που ενεργοποιούνται με το πάτημα κάποιου αντίστοιχου κουμπιού του προγράμματος. Οι πέντε βασικότερες λειτουργίες είναι οι εξής:

1) ***Διαμόρφωση*** element, attribute και της τιμής της value. Το καινούριο τους όνομα μπορεί να το επιλέξει ο χρήστης του προγράμματος από κάποια προϋπάρχοντα ονόματα (DropDown List) ή να γράψει το δικό του όνομα σε TextBox.

Διαμόρφωση element

```
protected void Button2_Click(object sender, EventArgs e)
    {
        // Only do this if a node is selected
        if ((treeXML.SelectedNode != null))
        {
            if (!editableBox1.Visible )
            {
                ReplaceElement(treeXML.SelectedNode.Text,
edit_element_new_dropdown.SelectedValue);
                edit_element_existing_name.Text =
edit_element_new_dropdown.SelectedValue;
            }
            else
            {
                if (editableBox1.Text.Length > 0)
                {
                    ReplaceElement(treeXML.SelectedNode.Text,
editableBox1.Text.Replace(" ", "_"));
                    edit_element_existing_name.Text = editableBox1.Text;
                }
                else
                {
                    infoBox.Text = "Παρακαλώ εισάγετε τιμή για το element
.";
                    infoBox.Visible = true;
                }
            }
            editableBox1.Visible = false;
            edit_element_new_dropdown.Visible = true;
            edit_element_new_dropdown.SelectedIndex = 0;
        }
        else{
            infoBox.Text = "Παρακαλώ επιλέξτε κάποιο βρόγχο από το tree
view";
            infoBox.Visible = true;
        }
    }
```

Διαμόρφωση attribute

```
protected void Button7_Click(object sender, EventArgs e)
    {
        //Check to see if we have any attributes
        if (edit_Attr_existing_name_dd.SelectedIndex != -1)
        {
            // See if there is a new attribute or one of the dropdown
            string newName,newValue = "";
            if (edit_Attr_new_name_text.Visible) { newName =
edit_Attr_new_name_text.Text.Replace(" ", "_"); }
            else {newName = edit_new_attr_name_dropdown.SelectedValue; }
            if (edit_Attr_new_value_text.Visible) { newValue =
edit_Attr_new_value_text.Text.Replace(" ", "_"); }
            else {newValue = edit_new_attr_value_dropdown.SelectedValue; }

            //Populate TreeViiew
            if (newName != "" && newValue != "" )
            {
```

```
                ReplaceAttribute(treeXML.SelectedNode.Text,
edit_Attr_existing_name_dd.Text, edit_Attr_existing_value_text.Text,
newName, "\""+ newValue + "\"");
            }
            else
            {
                infoBox.Text = "Παρακαλώ εισάγετε μια τιμή για το
attribute.";
                infoBox.Visible = true;
            }
        }
        else
        {
            infoBox.Text = "Ο επιλεγμένος βρόγχος δεν έχει attributes.";
            infoBox.Visible = true;
        }
    }
```

2) ***Προσθήκη*** element, attribute ή τιμής value κάποιας attribute. Για το όνομα του element ή της attribute και της τιμής της ισχύει ότι και στη λειτουργία της διαμόρφωσης, δηλαδή είτε μέσω DropDown List είτε μέσω TextBox.

Προσθήκη element

```
protected void Button5_Click(object sender, EventArgs e)
    {
        //only do this if a node is selected
        if (treeXML.SelectedNode != null)
        {
            if (add_New_Element_Text.Visible)
            {
                if (add_New_Element_Text.Text.Length > 0)
                {
                    TreeNode tn = new
TreeNode(add_New_Element_Text.Text.Replace(" ", "_"));
                    treeXML.SelectedNode.ChildNodes.Add(tn);
                }
                else
                {
                    infoBox.Text = "Παρακαλώ εισάγετε κάποιο όνομα για το
element.";
                    infoBox.Visible = true;
                }
            }
            else
            {
                TreeNode tn = new
TreeNode(add_element_new_dropdown.SelectedValue);
                treeXML.SelectedNode.ChildNodes.Add(tn);
            }
        }
        else
        {
            infoBox.Text = "Παρακαλώ επιλέξτε κάποιο βρόγχο από το tree
view.";
            infoBox.Visible = true;
        }
```

```
    }
```

Προσθήκη attribute

```
protected void Button8_Click(object sender, EventArgs e)
    {
         //only do this if a node is selected
        if (treeXML.SelectedNode != null)
        {
            // See if there is a new attribute or one of the dropdown
            string newName, newValue = "";
            if (add_attr_name_text.Visible) { newName =
add_attr_name_text.Text.Replace(" ", "_"); }
            else { newName =
add_element_new_attributeName_dropdown.SelectedValue; }
            if (add_attr_value_text.Visible) { newValue =
add_attr_value_text.Text.Replace(" ", "_"); }
            else { newValue =
add_element_new_attributeValue_dropdown.SelectedValue; }

            //Populate TreeViiew
            if (newName != "" && newValue != "")
            {
                treeXML.SelectedNode.Text = treeXML.SelectedNode.Text + "
" + newName + " = \"" + newValue + "\"";
            }
            else
            {
                infoBox.Text = "παρακαλώ εισάγετε μια τιμή για τα
attributes.";
                infoBox.Visible = true;
            }

        }
        else
        {
            infoBox.Text = "Παρακαλώ επιλέξτε κάποιο βρόγχο από το tree
view.";
            infoBox.Visible = true;
        }

      }
```

3) ***Διαγραφή*** element και attribute. Τα elements μπορούν να διαγραφούν απευθείας. Αντίθετα για την περίπτωση των attributes έχει σχεδιαστεί μια DropDown List που

περιέχει όλες τις attributes του επιλεγμένου element σε περίπτωση που είναι από δύο και πάνω.

Διαγραφή element

```
protected void Button3_Click(object sender, EventArgs e)
    {
        if (treeXML.SelectedNode != null)
        {
            //ensure you cant delete root
            if (treeXML.SelectedNode.Parent != null)
            {
            treeXML.SelectedNode.Parent.ChildNodes.RemoveAt(0);
            }
            else
            {
                infoBox.Text = "Δεν μπορείτε να διαγράψετε το root
element.";
                infoBox.Visible = true;
           }
        }
        else
        {
            infoBox.Text = "Παρακαλώ επιλέξτε κάποιο βρόγχο από το tree
view.";
            infoBox.Visible = true;
        }
    }
```

Διαγραφή attribute

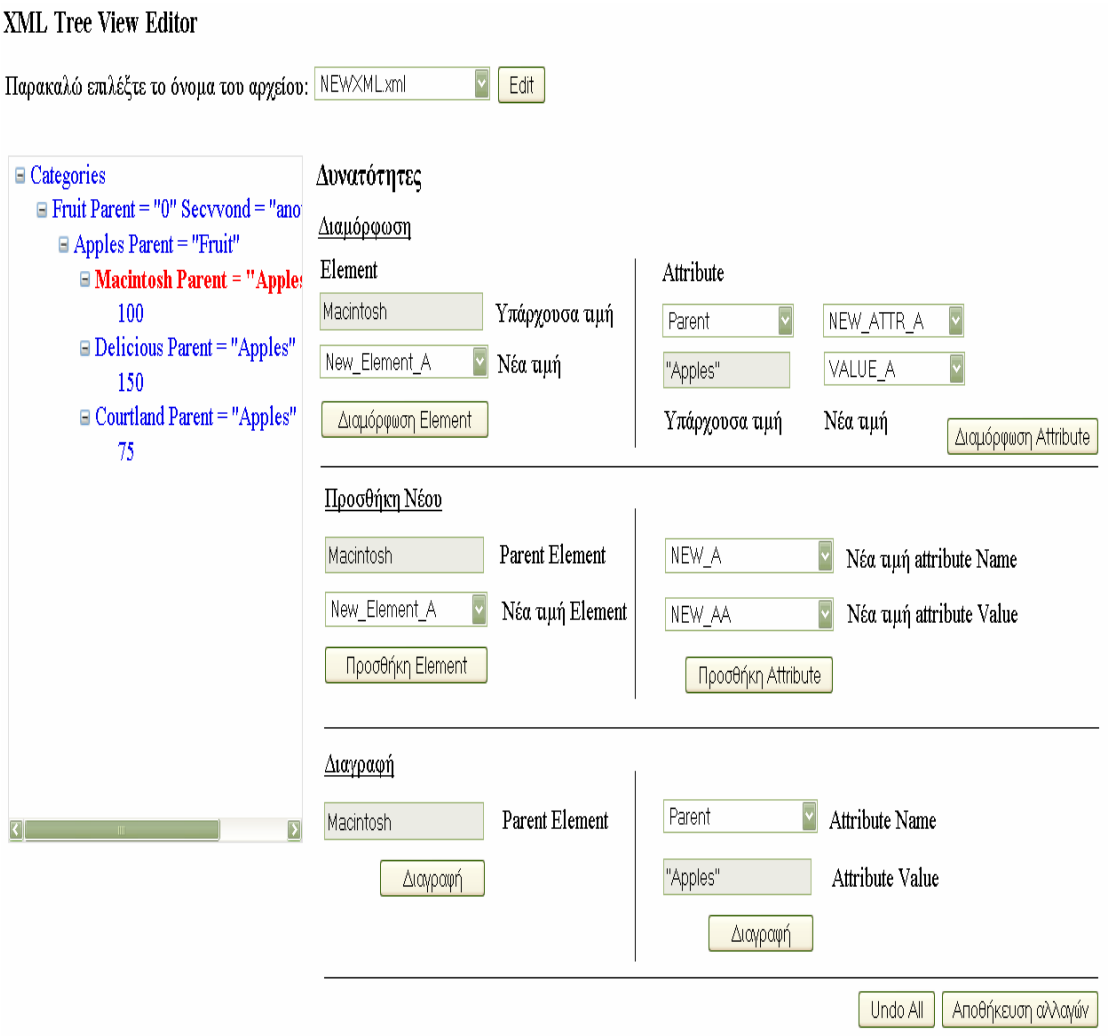
```
protected void Button6_Click(object sender, EventArgs e)
    {
        //only do this if a node is selected and there are some attributes
        if ((treeXML.SelectedNode != null) &&
(delete_attr_name_drop.Items.Count >0))
        {
            RemoveAttribute(treeXML.SelectedNode.Value,
delete_attr_name_drop.SelectedValue, delete_attr_value_txt.Text);

            //Clear Controls
            delete_attr_name_drop.Items.Clear();
            delete_attr_value_txt.Text = "";

        }
        else
        {
            infoBox.Text = "Παρακαλώ επιλέξτε κάποιο βρόγχο με
attributes";
            infoBox.Visible = true;
        }
    }
```

4) ***Αναίρεση*** αλλαγών. Χάρη στη λειτουργία της αναίρεσης (Undo) ο χρήστης μπορεί να αναιρέσει οποιεσδήποτε αλλαγές έχει κάνει σε ένα αρχείο xml και να το επαναφέρει την αρχική του μορφή.

5) ***Αποθήκευση*** αλλαγών. Με τη λειτουργία αυτή (Save) ο χρήστης μπορεί να αποθηκεύσει τις αλλαγές που έχει κάνει στο xml αρχείο. Το αρχείο αποθηκεύεται εκεί που βρίσκονται και τα υπόλοιπα προϋπάρχοντα αρχεία.

```
protected void Button4_Click(object sender, EventArgs e)
    {
        try
        {
            TreeViewToXml.exportToXml(this.treeXML,
@"/XML_Project_Final/XMLData/" +
xmlFileNames.SelectedItem.Text.Replace(".xml", "") + "_" +
System.DateTime.Now.ToLongDateString().Replace(" ", "_") + ".xml");

            infoBox.Text = "Το αρχείο αποθηκεύτηκε με επιτυχία.";
            infoBox.Visible = true;

            xmlFileNames.Items.Clear();
            //Reload contents of dropdown
            //get whatever is on that directory - FULL paths
            string[] filePaths =
Directory.GetFiles(@"/XML_Project_Final/XMLData");

            // get just path names
            // and populate dropdown with filenames
            string[] fileNames = new string[filePaths.Length];
            for (int i = 0; i < filePaths.Length; i++)
            {
                fileNames[i] = Path.GetFileName(filePaths[i]);
                xmlFileNames.Items.Add(fileNames[i]);
            }
        }
        catch (XmlException xmlEx)
        {
            infoBox.Text = xmlEx.Message;
            infoBox.Visible = true;

        }


    }
```

Γενικότερα, οι παραπάνω λειτουργίες αποτυπώνονται καλύτερα στο Σχήμα 4.7.



Σχήμα 4.7 – Οι πέντε βασικές λειτουργίες της εφαρμογής όπως εμφανίζονται για το επιλεγμένο element Macintosh.

4.3 Εκτέλεση της εφαρμογής

Αφού αναλύσαμε τον σχεδιασμό και την υλοποίηση της εφαρμογής, είμαστε πλέον σε θέση να επιδείξουμε τον τρόπο λειτουργίας της. Η επίδειξη της λειτουργίας του θα γίνει στον browser Mozilla Firefox 3.0 .Καταρχάς, υποθέτουμε πως ένα χρήστης θέλει να τροποποιήσει

κάποιο αρχείο xml που προϋπάρχει στον υπολογιστή του. Όταν ο χρήστης τρέχει το πρόγραμμα τότε η αρχική σελίδα που εμφανίζεται μπροστά του είναι το σχήμα 4.8:

Σχήμα 4.8 – Η αρχική σελίδα της εφαρμογής XML Editor.

Μπορούμε να δούμε πως αν πιέσουμε το βελάκι του στοιχείου ελέγχου DropDown List θα εμφανιστούν όλα τα αρχεία που έχουμε αποθηκεύσει στο φάκελο αρχείων μας, όπως τον έχουμε καθορίσει στον κώδικα του προγράμματος. Στην περίπτωση μας ο φάκελος δεν περιέχει μόνο αρχεία xml αλλά και αρχεία άλλου είδους (με διαφορετική κατάληξη).

Σχήμα 4.9 – Απεικόνιση στη λίστα όλων των αρχείων του φακέλου.

Ας υποθέσουμε πως ο χρήστης θέλει να τροποποιήσει το αρχείο test.txt. Επομένως, επιλέγει το αρχείο από τη λίστα και πατάει Edit. Αυτό όμως δεν θα επιτραπεί από τον XML Editor και έτσι θα εμφανιστεί ένα μήνυμα στην οθόνη του υπολογιστή του που θα γράφει ότι το αρχείο δεν είναι μορφής xml, οπότε και δεν μπορεί να το επεξεργαστεί. Το ίδιο αποτέλεσμα θα είχε οποιαδήποτε προσπάθεια με αρχεία διαφορετικής κατάληξης (doc, xls, pdf κλπ).

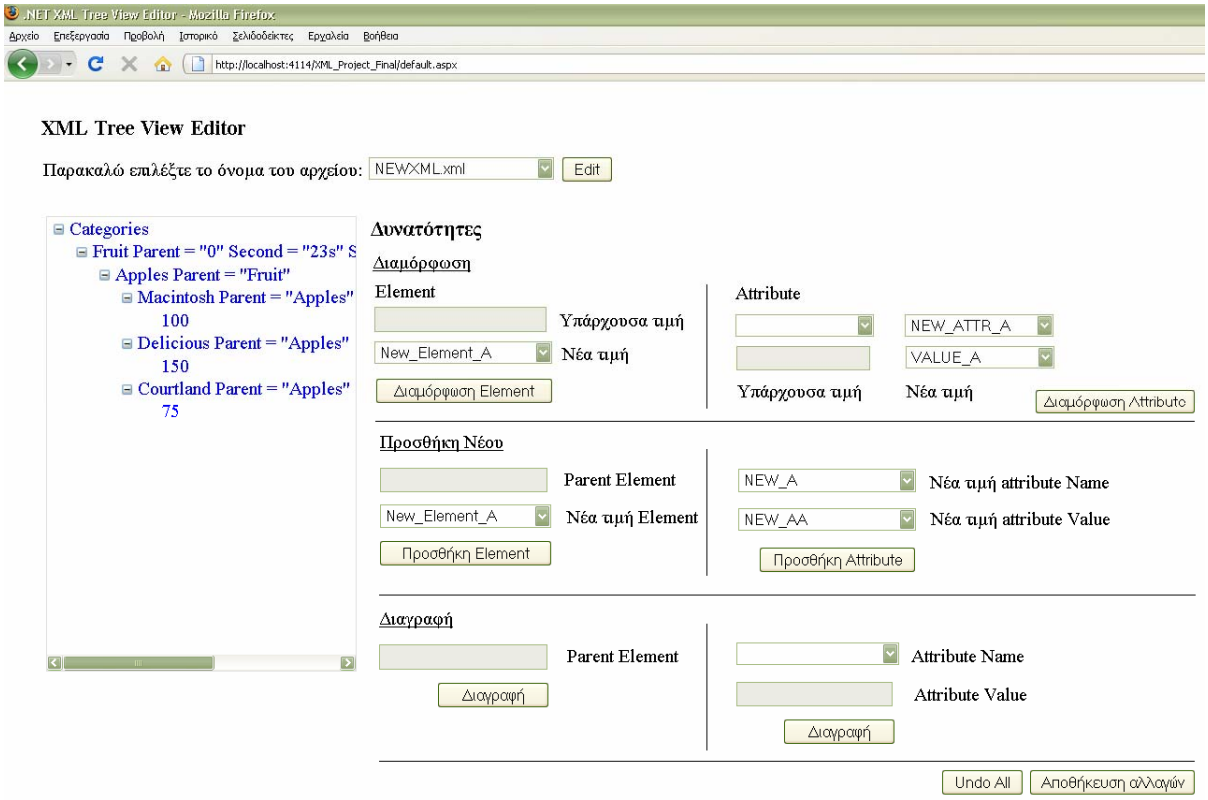
Σχήμα 4.10 – Εμφάνιση μηνύματος σφάλματος για μη επιλογή xml αρχείου προς επεξεργασία.

Στη συνέχεια ο χρήστης επιλέγει ένα αρχείο xml, ας υποθέσουμε το NEWXML.xml. Έχουμε δημιουργήσει ένα αρχείο που περιέχει elements, attributes και τιμές (values) για τα attributes με σκοπό να γίνει καλύτερη κατανόηση των δυνατοτήτων της εφαρμογής από τον αναγνώστη. Η μορφή του αρχείου xml είναι η παρακάτω:

```
<?xml version="1.0"?>
<Categories>
  <Fruit Parent="0" Second="23s" Secvvond= "another_value">
    <Apples Parent="Fruit">
      <Macintosh Parent="Apples">
        100
      </Macintosh>
      <Delicious Parent="Apples">
        150
      </Delicious>
      <Courtland Parent="Apples">
```

```
      75
    </Courtland>
  </Apples>
  <Pears Parent="Fruit"></Pears>
 </Fruit>
</Categories>
```

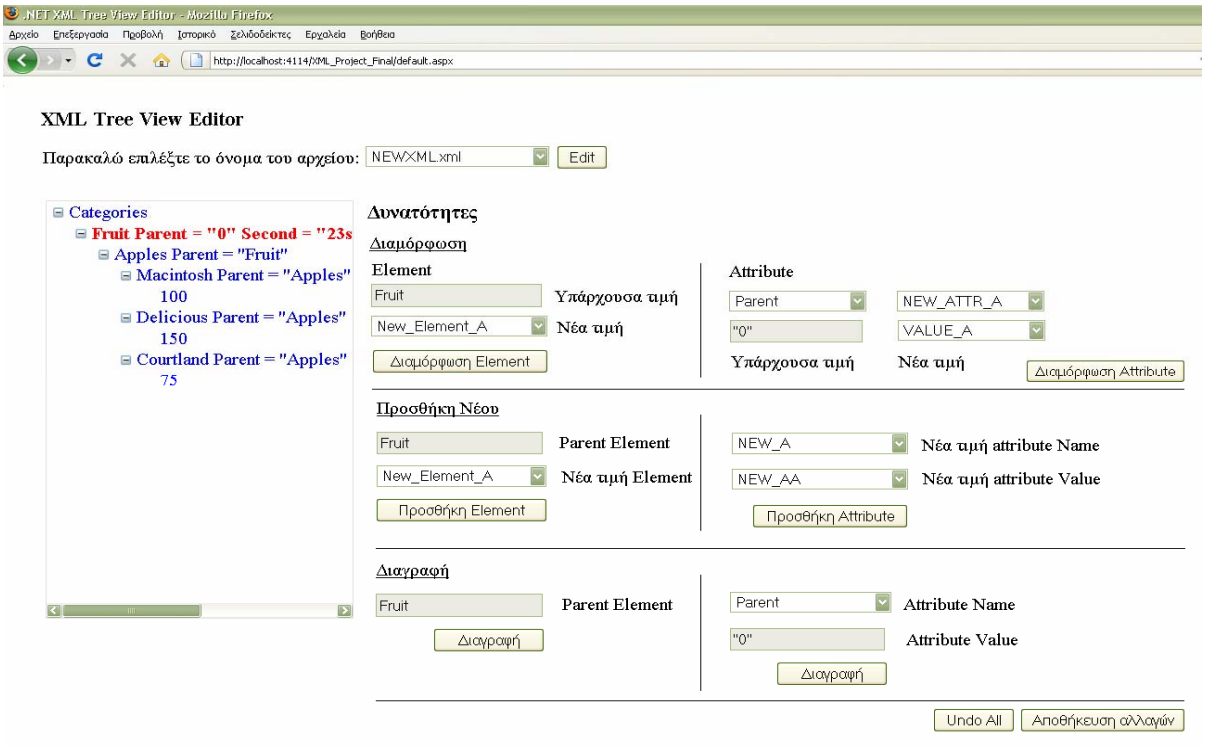
Όταν ο χρήστης επιλέξει το κουμπί Edit δίπλα στο βελάκι της λίστας θα εμφανιστεί στην οθόνη του η εξής εικόνα:



Σχήμα 4.11 – Το πρόγραμμα XML Editor και οι δυνατότητες του

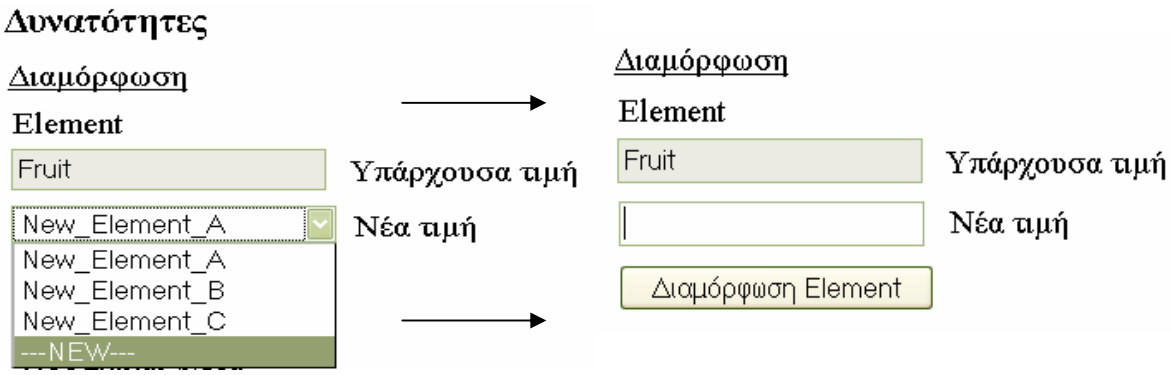
Όπως μπορούμε να παρατηρήσουμε στο σχήμα 4.12, επιλέγοντας ένα element από το TreeView αυτό χρωματίζεται διαφορετικά από τα υπόλοιπα. Παράλληλα, με την επιλογή του κόμβου «γεμίζουν» τα TextBoxes με το προσωρινό όνομα του element και της attribute και της τιμής της (value) αν υπάρχει.

Έχοντας ως βάση την εικόνα αυτή θα επισημάνουμε βήμα προς βήμα τις δυνατότητες του προγράμματος:



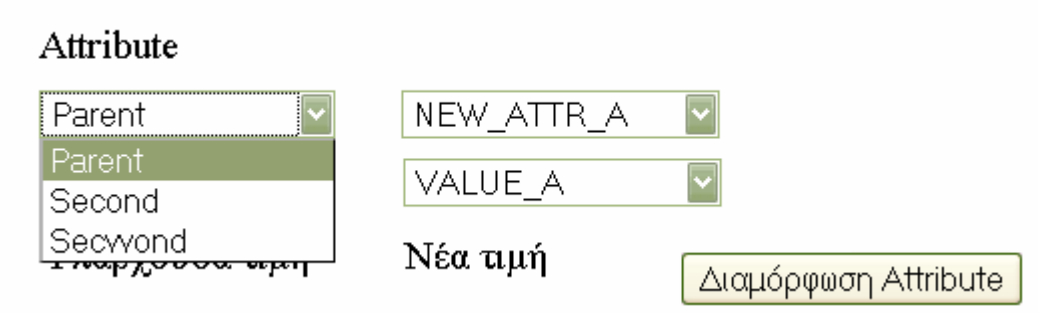
Σχήμα 4.12 – Επιλογή του element Fruit και απεικόνιση των attributes και values στον editor

- Διαμόρφωση element: Ο χρήστης μπορεί να επιλέξει ένα νέο όνομα για το επιλεγμένο element από τη λίστα που στα δεξιά της αναγράφει «Νέα τιμή». Αν δεν θέλει να το μετονομάσει βάσει των προκαθορισμένων ονομάτων της λίστας μπορεί επιλέγοντας το στοιχείο της λίστας NEW να γράψει ότι όνομα θέλει στο TextBox που εμφανίζεται (βλέπε σχήμα 4.13). Για να τελειώσει τη μετατροπή «κλικάρει» το κουμπί *Διαμόρφωση Element*.



Σχήμα 4.13 – Μετονομασία element με μη προκαθορισμένο όνομα

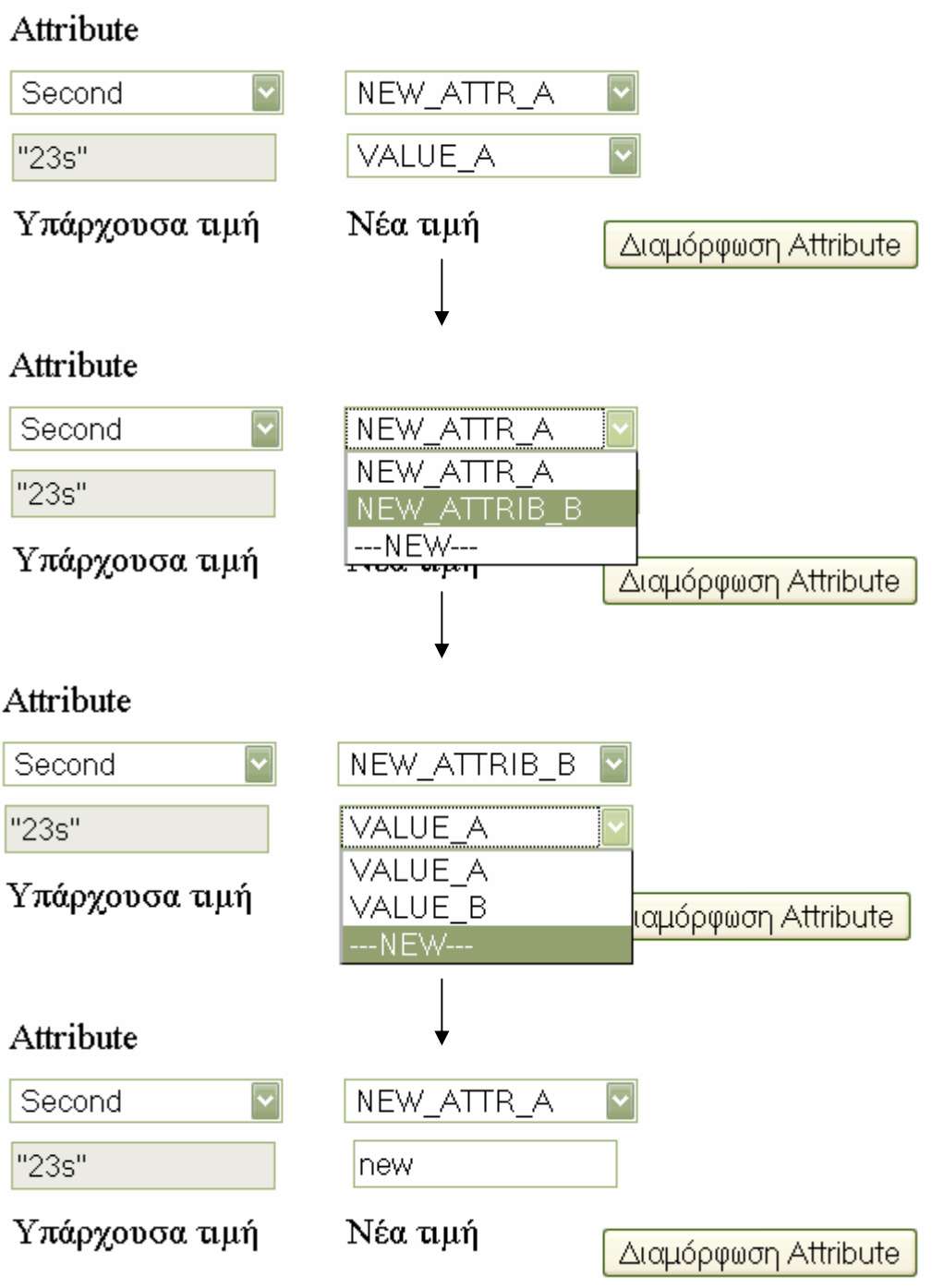
- Διαμόρφωση attribute και τιμής: Όπως και στην περίπτωση των elements έτσι και σε ενδεχόμενη επιθυμία του χρήστη να μεταβάλλει την τιμή κάποιας attribute ή της τιμής της ή ακόμα και των δύο ακολουθείται η ίδια διαδικασία. Δηλαδή, αφού έχουμε επιλέξει τον κόμβο που θέλουμε να τροποποιήσουμε (εικόνα 4.12) εμφανίζεται μία λίστα με όλες τις attributes που περιέχει ο κόμβος αυτός. Στην περίπτωση μας, ο επιλεχθείς κόμβος ήταν ο κόμβος με το element Fruit. Όπως βλέπουμε στο TreeView στοιχείο της εφαρμογής, περιέχει και τρεις attributes (Parent, Second, Secvvond) με αντίστοιχες τιμές "0", "23s" και "another_value". Όταν επιλεγεί το element Fruit απεικονίζονται στην οθόνη μας και οι τρεις attributes σε μια λίστα (Σχήμα 4.14).



Σχήμα 4.14 – Οι attributes του element Fruit όπως απεικονίζονται στην DropDown List

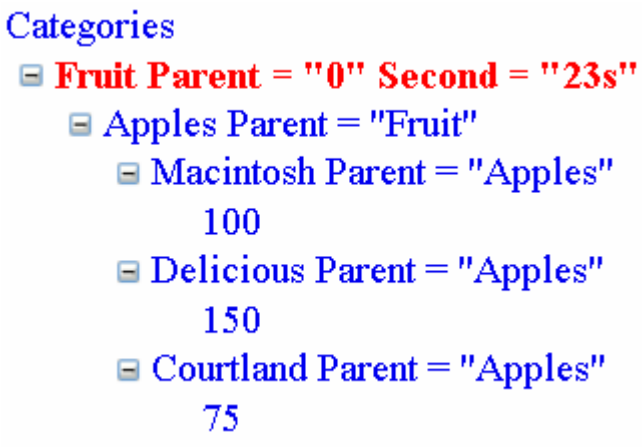
Έστω ότι θέλουμε να μεταβάλλουμε την attribute με το όνομα Second. Ακριβώς κάτω από το όνομα Second μπορούμε να παρατηρήσουμε το TextBox που αναγράφει την τρέχουσα τιμή της πριν τη μετατροπή. Στη λίστα δεξιά από την attribute Second υπάρχει μια λίστα που περιέχει κάποιες προϋπάρχουσες τιμές από τις οποίες μπορούμε να επιλέξουμε το νέο όνομα της. Εκεί ακριβώς περιέχεται και η επιλογή NEW, που δίνει τη δυνατότητα στο χρήστη να γράψει οποιαδήποτε τιμή επιθυμεί. Με όμοιο τρόπο μπορούμε να αλλάξουμε και την τιμή των values από τη λίστα που εμφανίζεται εκ δεξιών του TextBox που αναγράφει την υπάρχουσα τιμή της attribute Second "23s". Επομένως, αν ο χρήστης αποφασίσει πως θέλει να αλλάξει το όνομα της Second στο προϋπάρχον όνομα της λίστας NEW_ATTRIB_B

και της τιμής από "23s" σε μια νέα τιμή εκτός λίστας, για παράδειγμα "new", η σειρά των ενεργειών θα είναι όπως στο σχήμα 4.15 . «Κλικάροντας» το κουμπί *Διαμόρφωση Attribute* η διαδικασία λαμβάνει τέλος.



Σχήμα 4.15 – Διαδικασία διαμόρφωσης της attribute Second και της τιμής της "23s"

Επομένως, η εικόνα που θα εμφανίζει η δεντρική μορφή του αρχείου πριν και μετά την παραπάνω διαμόρφωση αποτυπώνεται στις παρακάτω εικόνες.

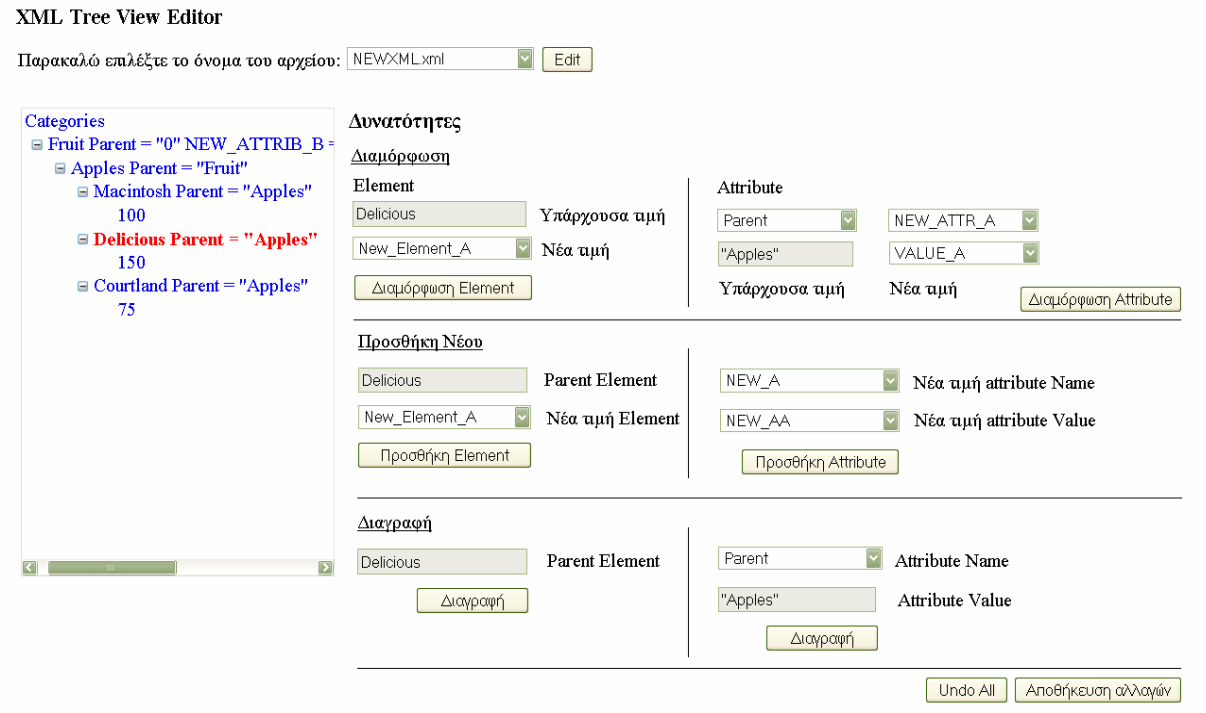


Σχήμα 4.16– Δεντρική μορφή του αρχείου NEWXML.xml πριν τη διαμόρφωση

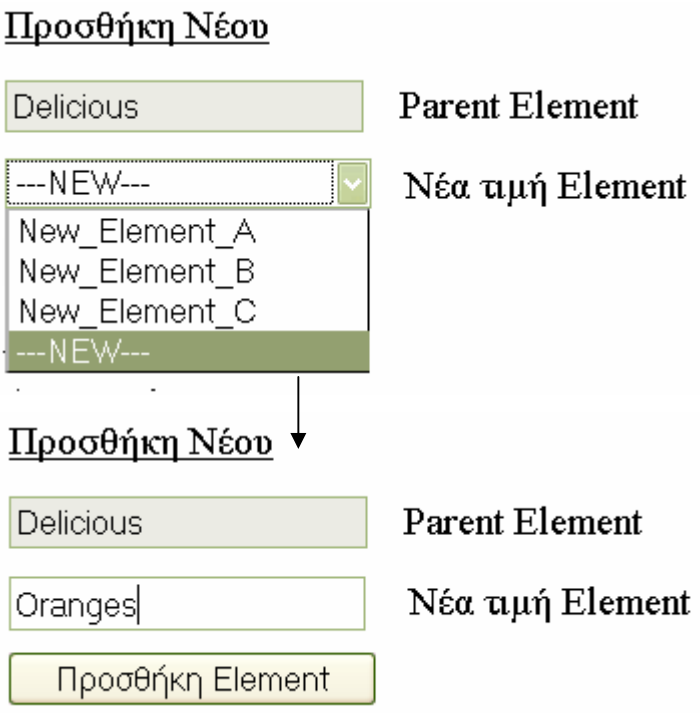
Σχήμα 4.17 – Δεντρική μορφή του αρχείου NEWXML.xml μετά τη διαμόρφωση

- Προσθήκη element: Στην περίπτωση αυτή ο χρήστης μπορεί να προσθέσει ένα element σε οποιοδήποτε σημείο του δέντρου επιθυμεί. Για παράδειγμα έστω ότι θέλει να προσθέσει ένα element με όνομα “Oranges” κάτω από το element “Delicious”. Για να πραγματοποιηθεί η προσθήκη αρκεί να επιλέξει με τον κέρσορα του ποντικού το element Delicious κάτω από το οποίο θα προστεθεί το element “Oranges”(Σχήμα 4.18), μετά να επιλέξει το NEW από τη λίστα με τα προϋπάρχοντα ονόματα (Σχήμα 4.19) και στο TextBox που εμφανίζεται να γράψει Oranges (μιας και το όνομα Oranges δεν εμφανίζεται ως προϋπάρχον) ενώ τέλος «κλικάρει» το κουμπί *Προσθήκη Element* για να ολοκληρωθεί η ενέργεια (Σχήμα 4.20). Προφανώς αν θέλουμε να

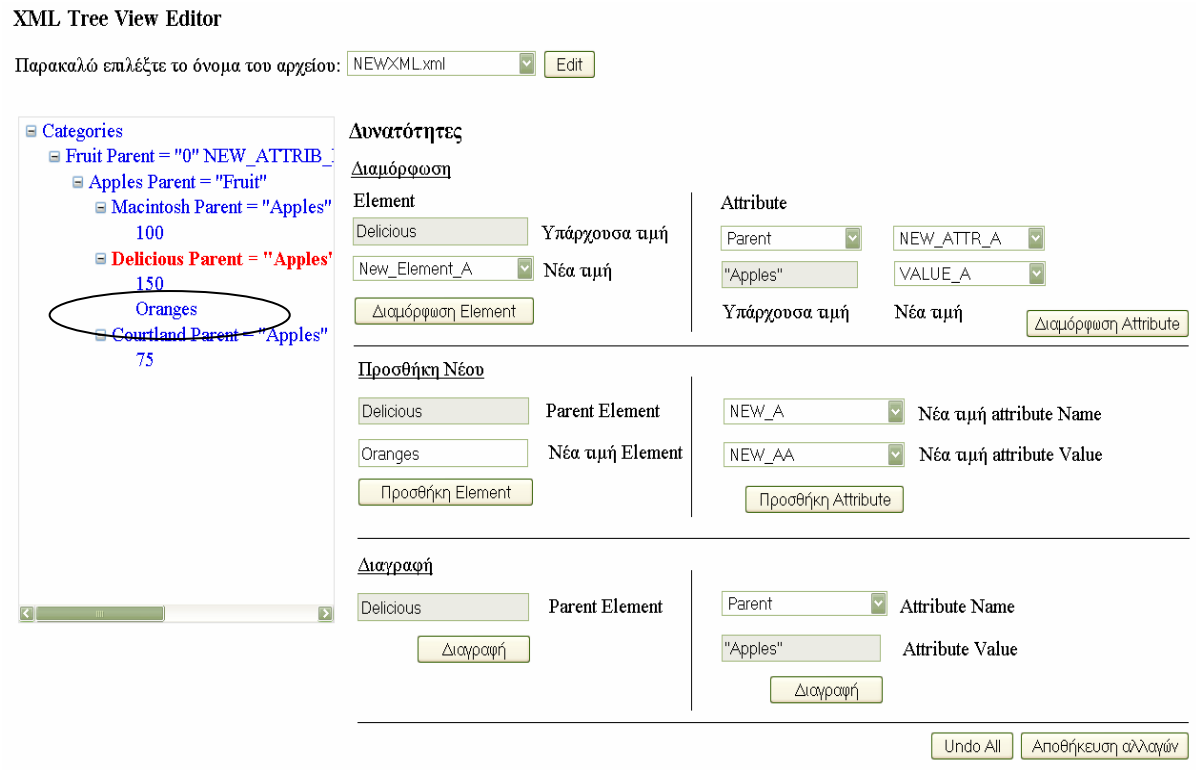
προσθέσουμε κάποιο element με όνομα που αναγράφεται στη λίστα αρκεί να το επιλέξουμε από τη DropDown List.



Σχήμα 4.18 – Επιλογή του element Delicious για προσθήκη νέου element

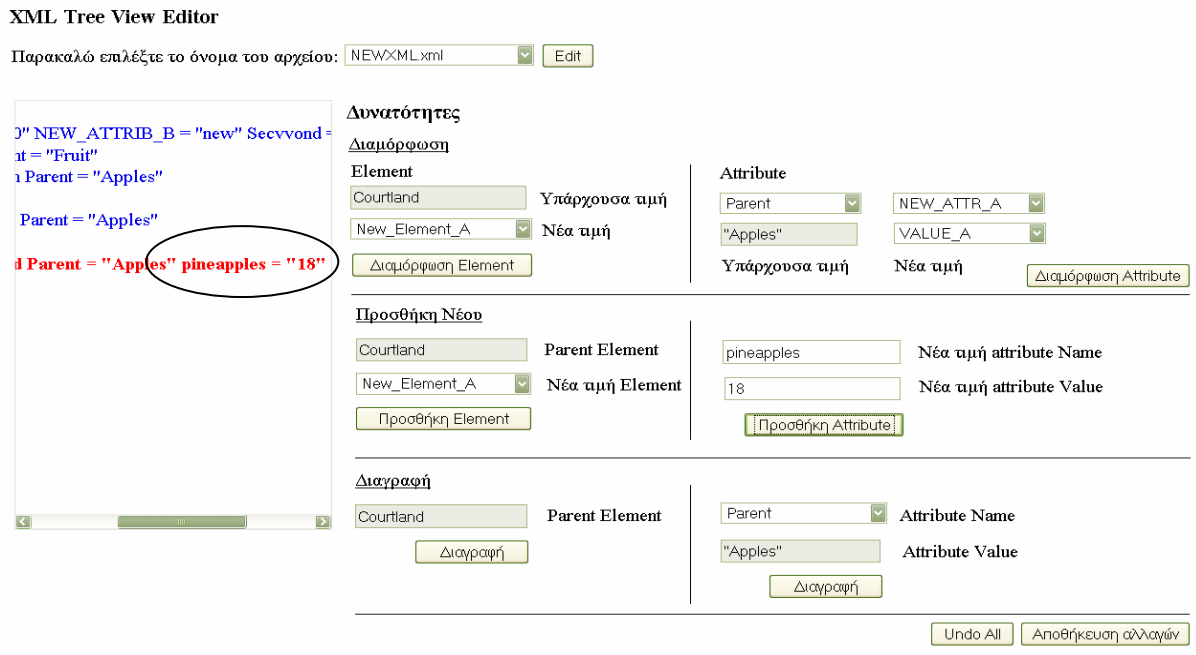


Σχήμα 4.19 – Προσθήκη element εκτός λίστας προϋπάρχοντων ονομάτων



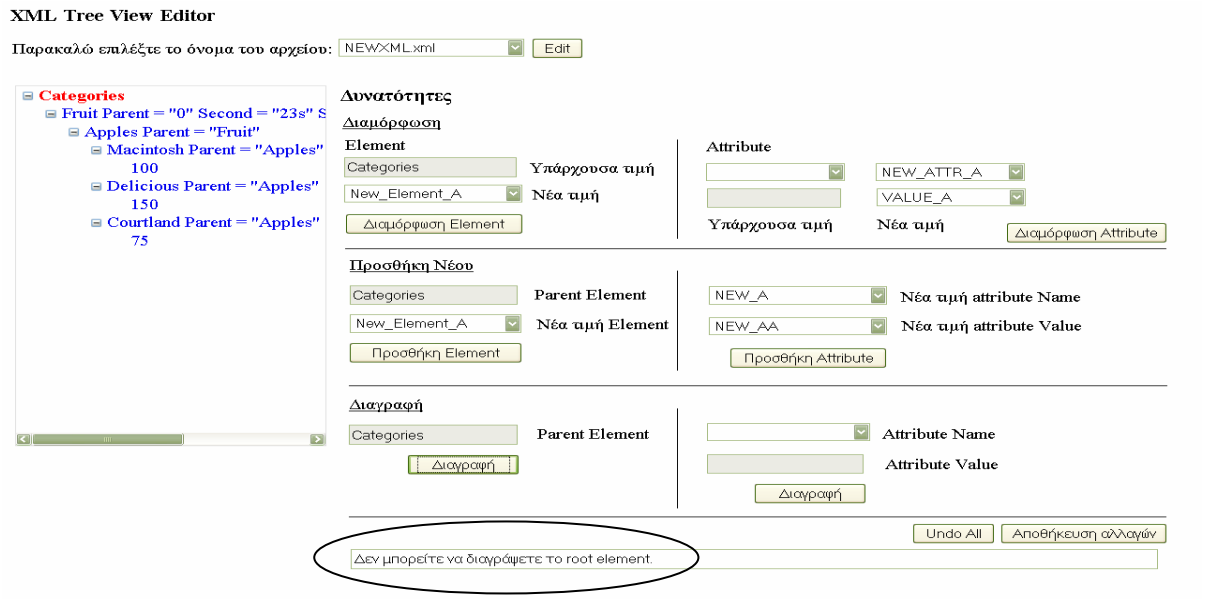
Σχήμα 4.20 – Απεικόνιση αρχείου μετά την προσθήκη του element Oranges

- Προσθήκη attribute και τιμής: Η διαδικασία που ακολουθείται είναι όμοια με πριν με τη μόνη διαφορά πως εκτός της attribute προστίθεται και μια τιμή γι' αυτή.



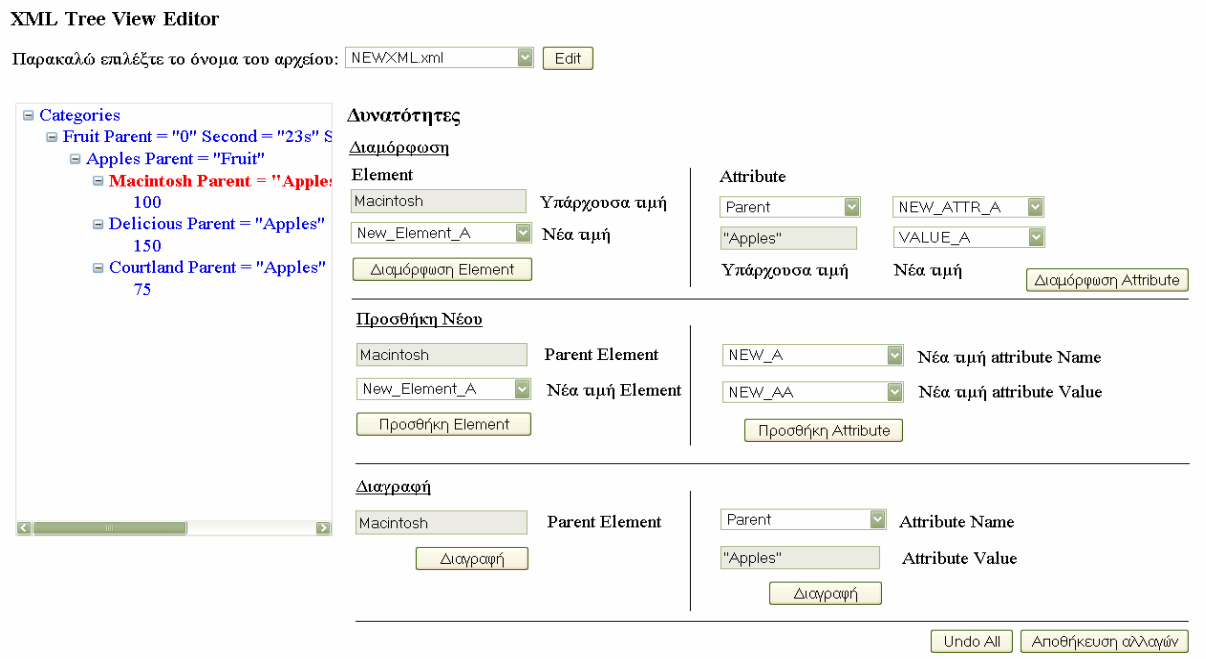
Σχήμα 4.21 – Προσθήκη attribute με όνομα pineapples και τιμή 18

- Διαγραφή element: Πριν περιγράψουμε τη διαδικασία διαγραφής element πρέπει να τονίσουμε ιδιαίτερα πως το πρόγραμμα δεν δίνει τη δυνατότητα στο χρήστη να διαγράψει το root element καθώς δίνει σωστά αποτελέσματα μόνο για valid αρχεία και όπως γνωρίζουμε ένα αρχείο xml χωρίς root element δεν είναι valid αρχείο. Κατά συνέπεια εάν θέλουμε να διαγράψουμε το root element Categories του παραδείγματος μας θα εμφανιστεί ένα μήνυμα σφάλματος που γράφει "*Δεν μπορείτε να διαγράψετε το root element*" (Σχήμα 4.22).

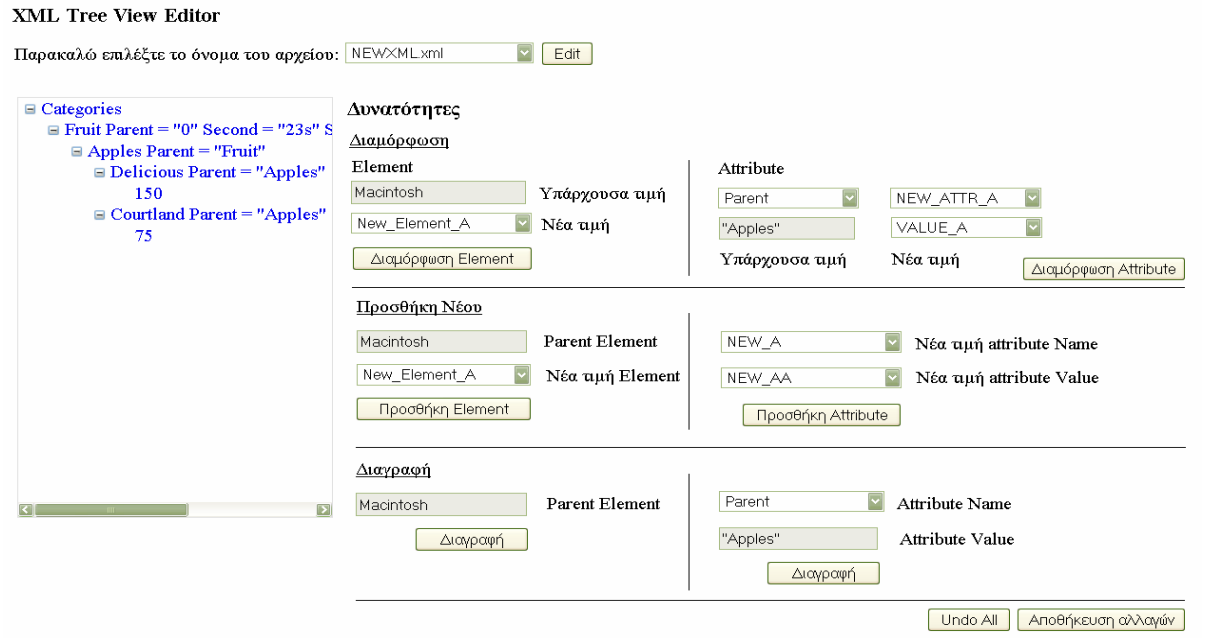


Σχήμα 4.22 – Εμφάνιση μηνύματος λάθους σε απόπειρα διαγραφής του root element

Φυσικά οποιοδήποτε άλλο element του αρχείου μπορεί να διαγραφεί εύκολα αρκεί να το επιλέξουμε και να «κλικάρουμε» πάνω στο κουμπί *Διαγραφή* .Στις παρακάτω εικόνες μπορούμε να δούμε το αποτέλεσμα της διαγραφής του element Macintosh και προφανώς των απογόνων του (element 100).



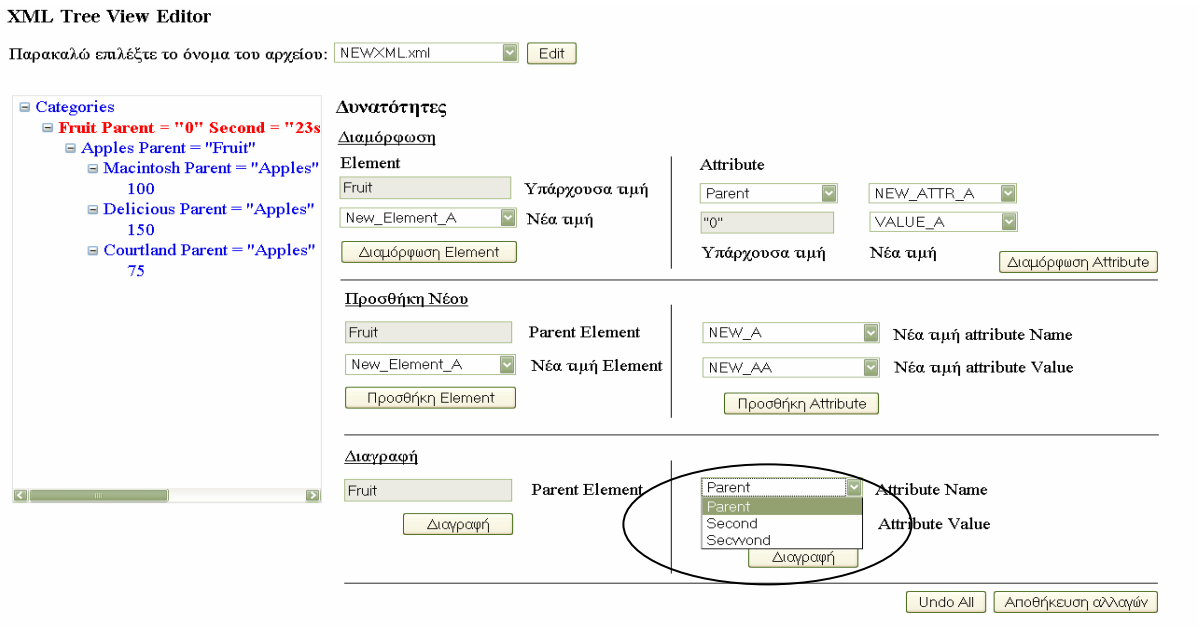
Σχήμα 4.23 – Επιλογή του element Macintosh προς διαγραφή



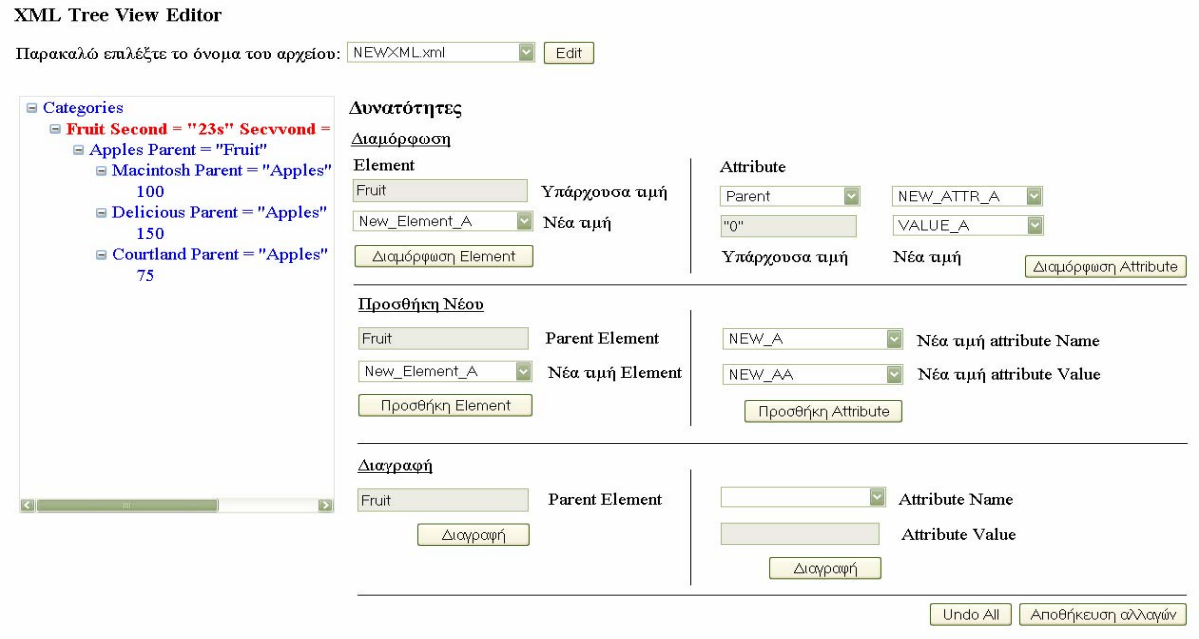
Σχήμα 4.24 – Η εικόνα του αρχείου μετά τη διαγραφή του element Macintosh

- Διαγραφή attribute: Όπως και πριν έτσι και στην περίπτωση διαγραφής μιας attribute αρκεί να επιλέξουμε τον κόμβο του δέντρου που περιέχει την προς διαγραφή attribute. Επειδή όμως υπάρχουν και κόμβοι με περισσότερες από μια, στην περίπτωση αυτή επιλέγουμε την attribute που θέλουμε να διαγράψουμε μέσα από μια

DropDown List. Για παράδειγμα, έστω ότι θέλουμε να διαγράψουμε την attribute Parent του element Fruit. Στη λίστα του πεδίου διαγραφής θα εμφανιστούν τα ονόματα και των τριών attributes που περιέχονται στον κόμβο, δηλαδή Parent, Second, Secvvond (Σχήμα 4.25). Επιλέγοντας την attribute Parent για διαγραφή, το αρχείο πλέον θα έχει τη μορφή του σχήματος 4.26 .



Σχήμα 4.25 –Το πεδίο διαγραφής των attributes γεμίζει με τα ονόματα του επιλεγμένου κόμβου Fruit

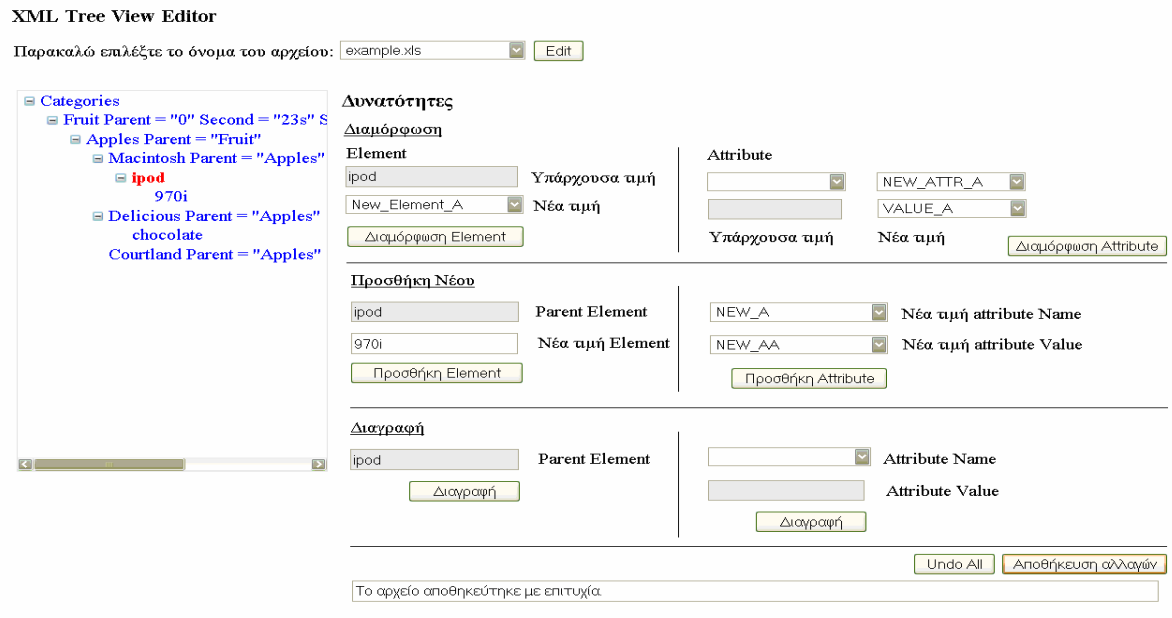


Σχήμα 4.26 – Η εικόνα του αρχείου αμέσως μετά τη διαγραφή της attribute με όνομα Parent

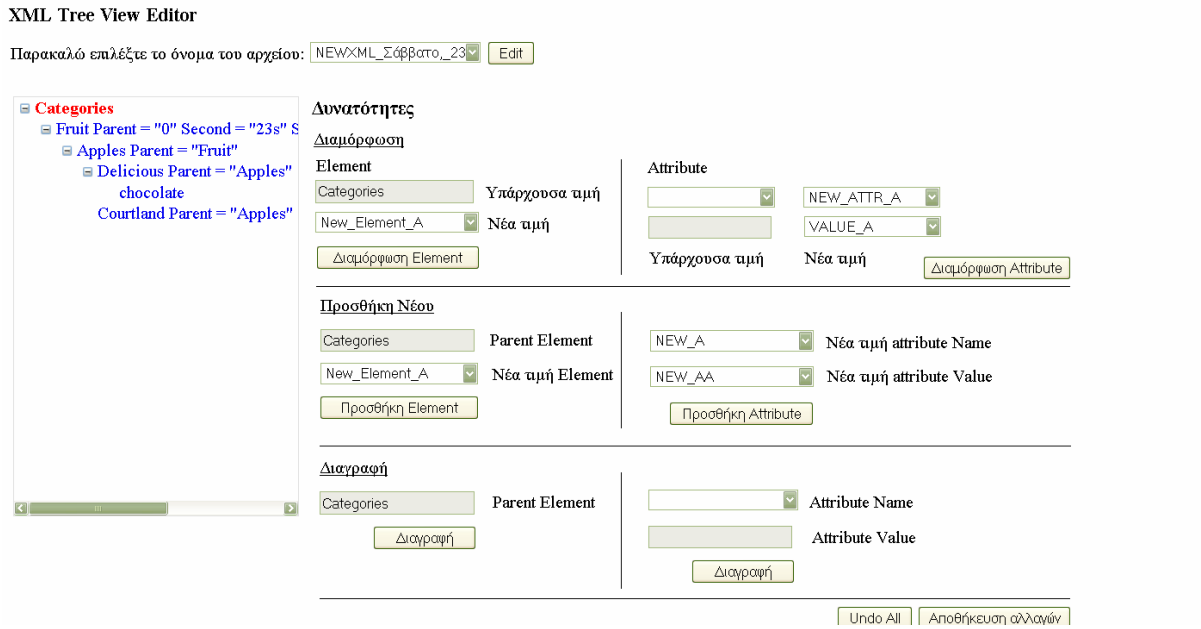
- Undo All: Η λειτουργία αυτή είναι πολύ χρήσιμη γιατί «κλικάροντας» το κουμπί αυτό αναιρούνται αυτόματα όλες οι αλλαγές που έχουν γίνει μέχρι τη στιγμή εκείνη στο αρχείο xml.

- Αποθήκευση αλλαγών: Με τη χρήση της λειτουργίας αυτής μπορούμε να αποθηκεύσουμε το τροποποιημένο xml αρχείο στον ίδιο φάκελο XML Data που βρίσκονται ήδη και τα υπόλοιπα αρχεία που εμφανίζονται στη λίστα του σχήματος 4.9.

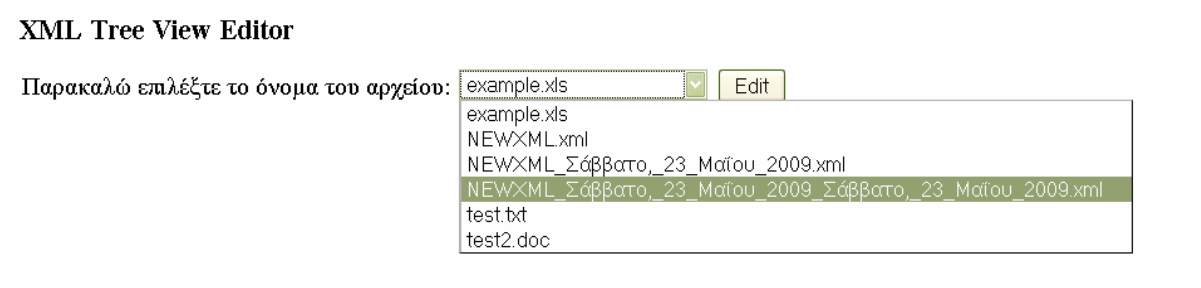
Υποθέτουμε και πάλι για παράδειγμα το αρχείο NEWXML.xml. Κάνουμε τις αλλαγές που επιθυμούμε και τέλος θέλουμε να αποθηκεύσουμε το τροποποιημένο αυτό αρχείο (Σχήμα 4.27). «Κλικάροντας» το κουμπί *Αποθήκευση αλλαγών* το αρχείο σώζεται σαν αρχείο xml φυσικά, ενώ πρέπει να τονίσουμε πως το όνομα με το οποίο αποθηκεύεται προκύπτει ως εξής: Το πρώτο μέρος του ονόματος είναι το παλαιό όνομα του αρχείου ακολουθούμενο από την ημερομηνία της αλλαγής (ημέρα, μήνας και έτος) που το πρόγραμμα διαβάζει αυτόματα από τον υπολογιστή του χρήστη. Δεύτερη αλλαγή στο ήδη αλλαγμένο αρχείο (όπως στο σχήμα 4.28) σημαίνει την εμφάνιση της ημερομηνίας δύο φορές, κάτι που επαναλαμβάνεται συνεχώς (Σχήμα 4.29). Μετά την αποθήκευση του αρχείου με το προσωρινό αυτό όνομα, ο χρήστης μπορεί να πάει στην τοποθεσία που αποθηκεύονται τα αρχεία και να μετονομάσει όπως αυτός θέλει το κάθε αρχείο.



Σχήμα 4.27 – Το τροποποιημένο αρχείο NEWXML_Σάββατο,_23_Μαϊου_2009.xml



Σχήμα 4.28 – Το τροποποιημένο αρχείο NEWXML_Σάββατο,_23_Μαΐου_2009_Σάββατο,_23_Μαΐου_2009.xml



Σχήμα 4.29 – Εμφάνιση αρχικής σελίδας του XML Editor με ένα δις τροποποιημένο και αποθηκευμένο αρχείο

4.4 Μελλοντικές βελτιοποιήσεις της εφαρμογής

Στο σημείο αυτό και μετά την εκτέλεση της εφαρμογής μπορούμε να κάνουμε μερικές παρατηρήσεις που μελλοντικά θα βοηθούσαν έτσι ώστε το πρόγραμμα XML Editor να αποδώσει ακόμα καλύτερα.

Ο πιο σημαντικός περιορισμός είναι ότι το πρόγραμμα δίνει σωστά αποτελέσματα μόνο για valid xml αρχεία. Αν το προς επεξεργασία αρχείο δεν είναι valid τότε το πρόγραμμα θα μας δώσει λάθος αποτελέσματα. Επομένως, μια αρκετά χρήσιμη μελλοντική βελτίωση θα ήταν αν η εφαρμογή μπορούσε να ελέγξει η ίδια από την αρχή εάν το αρχείο xml είναι valid. Εάν είναι πράγματι, τότε και μόνο τότε να συνεχίζει τη διαδικασία μετατροπής του αρχείου από το χρήστη. Αλλιώς θα εμφανίζεται ένα μήνυμα σφάλματος που θα ενημερώνει τον χρήστη ότι το αρχείο xml δεν είναι έγκυρο. Στην παρούσα φάση, ο χρήστης μπορεί να ελέγχει την εγκυρότητα των αρχείων του με άλλους τρόπους, όπως για παράδειγμα μέσω ιστοσελίδων στο διαδίκτυο, όπως η ιστοσελίδα www.validome.org/xml.

Παράλληλα, ένα ακόμα σημείο που μπορεί να ερευνηθεί για δυνατή βελτίωση της εφαρμογής μας είναι το γεγονός ότι προς το παρόν το πρόγραμμα δουλεύει σωστά εάν τα κλαδιά των βρόγχων καταλήγουν σε pure text element χωρίς κενά. Για παράδειγμα τα xml αρχεία:

```
<?xml version="1.0"?>
<Categories>
  <Fruit Parent="0" Second="23s" Secvvond= "another_value">
  </Fruit>
</Categories>
```

και

```
<?xml version="1.0"?>
<Categories>
</Categories>
```

παρόλο που θεωρητικά είναι valid xml αρχεία, δηλαδή έγκυρα, δεν θα διαβαστούν σωστά από το πρόγραμμα γιατί δεν καταλήγουν σε pure text element χωρίς κενά. Θα πρέπει δηλαδή να είναι της μορφής:

```
<?xml version="1.0"?>
<Categories>
  <Fruit Parent="0" Second="23s" Secvvond= "another_value">SOMETHING
  </Fruit>
</Categories>
```

και

```
<?xml version="1.0"?>
<Categories>
</Categories>.
```

Προφανώς ίδια περίπτωση λάθους είναι και η ακόλουθη και καλό θα ήταν να την αποφύγει κάποιος προκειμένου να πάρει σωστό αποτέλεσμα:

```
<?xml version="1.0"?>
<Categories>I LIKE XML
</Categories>.
```

Σωστή θα είναι η παρακάτω μορφή, δηλαδή με underscores αντί κενού, γιατί όπως είπαμε και πριν πρέπει τα κλαδιά των βρόγχων να καταλήγουν σε pure text element χωρίς κενά.

```
<?xml version="1.0"?>
<Categories>I _LIKE _XML
</Categories>
```

5

Πηγαίος κώδικας προγράμματος (source code)

5.1 Η κλάση TreeViewToXml

Παρακάτω παρουσιάζεται η κλάση TreeeViewToXml, την οποία πρόκειται να συναντήσουμε στον πηγαίο κώδικα της εφαρμογής μας. Η κλάση αυτή ορίζεται γιατί με τη χρήση της όπως θα δούμε στο κυρίως πρόγραμμα εξάγεται το δεδομένο TreeView σε μορφή XML και επιστρέφει μια συμβολοσειρά (string) που εμπεριέχει όλα τα tags:

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.IO;
using System.Collections;

/// <summary>
/// Summary description for TreeViewToXml
/// </summary>
```

```
public class TreeViewToXml
{

    private static string xmlstring = "";
    private static StreamWriter sr;

      public TreeViewToXml()
      {

      }

    /// <summary>
    /// Exports the given TreeView to Xml and returns a string containing
all tags
    /// </summary>
    /// <param name="tv">TreeView - This TreeView will be parsed</param>
    ///
    public static void exportToXml(TreeView tv, string filename)
    {
        sr = new StreamWriter(filename, false, System.Text.Encoding.UTF8);
        sr.WriteLine("<?xml version=\"1.0\" encoding=\"utf-8\" ?>");

        IEnumerator ie = tv.Nodes.GetEnumerator();
        if (ie.MoveNext())
        {
            TreeNode tn = (TreeNode)ie.Current;
            sr.WriteLine("<" + tn.Text + ">");
            parseNode(tn);
        }

        sr.Close();
    }

    private static void parseNode(TreeNode tn)
    {
        IEnumerator ie = tn.ChildNodes.GetEnumerator();

        string parentnode = "";

        parentnode = tn.Text;

        while (ie.MoveNext())
        {
            TreeNode ctn = (TreeNode)ie.Current;

            if (ctn.ChildNodes.Count == 0)
            {
                sr.Write(ctn.Text);
            }
            else
            {
                sr.Write("<" + ctn.Text + ">");
            }
            if (ctn.ChildNodes.Count > 0)
            {
                parseNode(ctn);
            }
        }

        sr.Write("</" + parentnode.Split(' ')[0] + ">");
        sr.WriteLine("");
```

```
    }
}
```

5.2 Η κλάση Default

Στις ακόλουθες γραμμές παρουσιάζεται η κλάση Default, που αποτελεί ουσιαστικά το κυρίως πρόγραμμα. Η υλοποίηση και η δομή της εξηγήθηκε επαρκώς στο 4ο κεφάλαιο, κατά συνέπεια αρκεί να παραθέσουμε ολόκληρο τον πηγαίο κώδικα της εφαρμογής.

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Xml;
using System.IO;

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

    /// <summary>
    /// Load the files of a given directory on the dropdown
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    protected void Page_Load(object sender, EventArgs e)
    {
        //Perorm this only once
        if (xmlFileNames.Items.Count == 0)
        {
            //get whatever is on that directory - FULL paths
            string[] filePaths =
Directory.GetFiles(@"/XML_Project_Final/XMLData");

            // get just path names
            // and populate dropdown with filenames
            string[] fileNames = new string[filePaths.Length];
            for (int i = 0; i<filePaths.Length; i++ )
            {
                fileNames[i] = Path.GetFileName(filePaths[i]);
                xmlFileNames.Items.Add(fileNames[i]);
            }
        }
    }
```

```
    //*******************************************************
    // HELPERS
    // *******************************************************

    /// <summary>
    /// Recursive Function to add a node on the tree.
    /// </summary>
    /// <param name="inXmlNode"></param>
    /// <param name="inTreeNode"></param>
    private void AddNode(XmlNode inXmlNode, TreeNode inTreeNode)
    {
        XmlNode xNode;
        TreeNode tNode;
        XmlNodeList nodeList;
        int i;

        // Loop through the XML nodes until the leaf is reached.
        // Add the nodes to the TreeView during the looping process.
        if (inXmlNode.HasChildNodes)
        {
            nodeList = inXmlNode.ChildNodes;
            for (i = 0; i <= nodeList.Count - 1; i++)
            {
                xNode = inXmlNode.ChildNodes[i];

                string NodeString = xNode.Name;

                //Check for attributes if any
                if (xNode.Attributes != null)
                {
                    int attrNum = xNode.Attributes.Count;
                    for (int count = 0; count < attrNum; count++)
                    {
                        string atName = xNode.Attributes[count].Name;
                        string atValue = xNode.Attributes[count].Value;

                        //and apend to NodeString
                        NodeString = NodeString + " " + atName + " = \"" +
atValue + "\"";
                    }
                }

                //Now add to tree
                inTreeNode.ChildNodes.Add(new TreeNode(NodeString));

                tNode = inTreeNode.ChildNodes[i];
                AddNode(xNode, tNode);
            }
        }
        else
        {
            // Here you need to pull the data from the XmlNode based on
the
            // type of node, whether attribute values are required, and so
forth.
            inTreeNode.Text = (inXmlNode.OuterXml).Trim();
        }
    }

    /// <summary>
    /// Get element node from selected line
```

```
    /// </summary>
    /// <param name="inNode"></param>
    /// <returns></returns>
    private string getElement(string inNode)
    {
        string[] elementValue;
        elementValue = inNode.Split(' ');
        return elementValue[0];
    }

    /// <summary>
    /// get attribute pairs from selected line
    /// </summary>
    /// <param name="inNode"></param>
    /// <returns></returns>
    private string[] getAttributes(string inNode)
    {
        string[] returnedPairs = new string[(inNode.Split(' ').Length) /
3];

        //0. Get all info in string array
        string[] attributePairs = inNode.Split(' ');

        //1. Determine if there are any attributes
        if (attributePairs.Length > 1)
        {
            //1. How many attributes?
            int numOfAttr = (attributePairs.Length) / 3;

            //2. Fetch attr name & values
            for (int i = 0; i < numOfAttr; i++)
            {
                returnedPairs[i] = attributePairs[1 + (i * 3)] + " " +
attributePairs[3 + (i * 3)];
            }
            return returnedPairs;
        }
        else
        {
            return returnedPairs;
        }
    }

    /// <summary>
    /// load attribute value
    /// </summary>
    /// <param name="ddl"></param>
    /// <param name="tb"></param>
    /// <param name="values"></param>
    private void findAttributeValueForSelectedName(DropDownList ddl,
TextBox tb, string[] values)
    {
        if (ddl.Items.Count > 0)
        {
            tb.Text = values[ddl.SelectedIndex].Split(' ')[1];
        }
    }

    /// <summary>
    /// Populate dropdown with attribute names
    /// </summary>
```

```
    /// <param name="ddl"></param>
    /// <param name="values"></param>
    private void populateDropDownWithAtributes(DropDownList ddl, string[]
values)
    {
        for (int i = 0; i < values.Length; i++)
        {
            ddl.Items.Add(values[i].Split(' ')[0]);
        }

    }

    /// <summary>
    /// Edit element of selected node
    /// </summary>
    /// <param name="selectedNode"></param>
    /// <param name="newElement"></param>
    private void ReplaceElement(string selectedNode, string newElement)
    {
        //Replace Element
        string[] elementValue;
        elementValue = selectedNode.Split(' ');
        elementValue[0] = newElement;

        //Formulate a new string
        string newSelectedNode = "";
        if (elementValue.Length == 1)
        {
            newSelectedNode = elementValue[0];
        }
        else
        {
            for (int i = 0; i < elementValue.Length; i++)
            {
                newSelectedNode = newSelectedNode + " " + elementValue[i];
            }
        }

        //Put on treeview
        treeXML.SelectedNode.Text = newSelectedNode;
    }

    /// <summary>
    /// Edit selected attribute
    /// </summary>
    /// <param name="selectedNode"></param>
    /// <param name="newElement"></param>
    private void ReplaceAttribute(string selectedNode, string
oldAttributeName, string oldAttributeValue, string newAttributeName,
string newAttributeValue )
    {
        //only do so if there is an attribute
        if (edit_Attr_existing_name_dd.SelectedIndex != -1 &&
edit_Attr_existing_value_text.Text.Length > 0)
        {

            //Replace Element
            string[] elementValue;
            elementValue = selectedNode.Split(' ');

            //find index of name & value
```

```
            int indexName = 0;
            int indexValue = 0;
            for (int i = 0; i < elementValue.Length; i++)
            {
                if (elementValue[i] == oldAttributeName)
                {
                    indexName = i;
                }
                if (elementValue[i] == oldAttributeValue)
                {
                    indexValue = i;
                }

            }



            elementValue[indexName] = newAttributeName;
            elementValue[indexValue] = newAttributeValue;


            //Formulate a new string
            string newSelectedNode = "";
            if (elementValue.Length == 1)
            {
                newSelectedNode = elementValue[0];
            }
            else
            {
                for (int i = 0; i < elementValue.Length; i++)
                {
                    newSelectedNode = newSelectedNode + " " +
elementValue[i];
                }
            }

            //Put on treeview
            treeXML.SelectedNode.Text = newSelectedNode.Trim();
        }
        else
        {
            infoBox.Text = "Ο επιλεγμένος βρόγχος δεν έχει attributes.";
            infoBox.Visible = true;

        }
    }

    /// <summary>
    /// Remove selected attribute
    /// </summary>
    /// <param name="selectedNode"></param>
    /// <param name="newElement"></param>
    private void RemoveAttribute(string selectedNode, string
AttributeName, string AttributeValue)
    {
        //Replace Element
        string[] elementValue;
        elementValue = selectedNode.Split(' ');

        //find index of name & value
        int indexName = 0;
```

```
        int indexValue = 0;
        for (int i = 0; i < elementValue.Length; i++)
        {
            if (elementValue[i] == AttributeName)
            {
                indexName = i;
            }
            if (elementValue[i] == AttributeValue)
            {
                indexValue = i;
            }

        }
        elementValue[indexName] = "__REMOVED__";
        elementValue[indexName+1] = "__REMOVED__";
        elementValue[indexValue] = "__REMOVED__";

        //Formulate a new string
        string newSelectedNode = "";

        for (int i = 0; i < elementValue.Length; i++)
        {
            if (elementValue[i] != "__REMOVED__")
            {
            newSelectedNode = newSelectedNode + " " + elementValue[i];
            }
        }

        //Put on treeview
        treeXML.SelectedNode.Text = newSelectedNode.Trim();
    }



    //*******************************************************
    // EVENTS
    // *******************************************************

    /// <summary>
    /// Event that is fired on each treeview click
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    protected void sthchanged(object sender, EventArgs e)
    {
        //0. Clear all boxes and dropdowns + visibility
        edit_Attr_existing_name_dd.Items.Clear();
        delete_attr_name_drop.Items.Clear();
        editableBox1.Visible = false;
        edit_element_new_dropdown.Visible = true;
        edit_element_new_dropdown.SelectedIndex = 0;
        infoBox.Visible = false;
        edit_new_attr_name_dropdown.Visible = true;
        edit_new_attr_value_dropdown.Visible = true;
        edit_new_attr_name_dropdown.SelectedIndex = 0;
        edit_new_attr_value_dropdown.SelectedIndex = 0;

        edit_Attr_new_name_text.Visible = false;
        edit_Attr_new_value_text.Visible = false;
        edit_Attr_existing_value_text.Text = "";
        add_New_Element_Text.Visible = false;
```

```
        add_New_Element_Text.Text = "";
        add_element_new_dropdown.Visible = true;
        add_element_new_dropdown.SelectedIndex = 0;
        add_attr_name_text.Visible = false;
        add_attr_value_text.Visible = false;
        add_element_new_attributeName_dropdown.Visible = true;
        add_element_new_attributeName_dropdown.SelectedIndex = 0;
        add_element_new_attributeValue_dropdown.Visible = true;
        add_element_new_attributeValue_dropdown.SelectedIndex = 0;
        delete_attr_value_txt.Text = "";


        //1. Get the  element of selected line
        string selectedElement = getElement(treeXML.SelectedValue);

        //2. And populate all controls that deal with element
        edit_element_existing_name.Text = selectedElement;
        add_parent_element_text.Text = selectedElement;
        delete_Element_text.Text = selectedElement;

        //3. Get all attributes - if any and populate a [] string
        string[] sa = getAttributes(treeXML.SelectedValue);

        //4. populate all dropdowns with attribute names
        populateDropDownWithAtributes(delete_attr_name_drop, sa);
        populateDropDownWithAtributes(edit_Attr_existing_name_dd, sa);

        //5. populate all textboxes with attribute values of selected
attribute
        findAttributeValueForSelectedName(edit_Attr_existing_name_dd,
edit_Attr_existing_value_text, sa);
        findAttributeValueForSelectedName(delete_attr_name_drop,
delete_attr_value_txt, sa);
    }

    /// <summary>
    /// Fired when attribute value of dropdown changes
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    protected void edit_Attr_existing_name_dd_SelectedIndexChanged(object
sender, EventArgs e)
    {
        findAttributeValueForSelectedName(edit_Attr_existing_name_dd,
edit_Attr_existing_value_text, getAttributes(treeXML.SelectedValue));
    }

    /// <summary>
    /// Fired when attribute value of dropdown2 changes
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    protected void delete_attr_name_SelectedIndexChanged(object sender,
EventArgs e)
    {
        findAttributeValueForSelectedName(delete_attr_name_drop,
delete_attr_value_txt, getAttributes(treeXML.SelectedValue));
    }

    /// <summary>
    /// Fired on change valu of dropdown - to get the editable textbox up
```

```
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    protected void edit_element_new_dropdown_SelectedIndexChanged(object
sender, EventArgs e)
    {
        if (edit_element_new_dropdown.SelectedValue == "---NEW---")
        {
            editableBox1.Text = "";

            edit_element_new_dropdown.Visible = false;
            editableBox1.Visible = true;
        }
    }

    /// <summary>
    /// Event fired on changed attribute NAME - edit
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    protected void DropDownList1_SelectedIndexChanged(object sender,
EventArgs e)
    {
        if (edit_new_attr_name_dropdown.SelectedValue == "---NEW---")
        {
            edit_Attr_new_name_text.Text = "";

            edit_new_attr_name_dropdown.Visible = false;
            edit_Attr_new_name_text.Visible = true;
        }


    }
    /// <summary>
    /// Event fired on changed on attribute dropdown VALUE - edit
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    protected void DropDownList2_SelectedIndexChanged(object sender,
EventArgs e)
    {
        if (edit_new_attr_value_dropdown.SelectedValue == "---NEW---")
        {
            edit_Attr_new_value_text.Text = "";

            edit_new_attr_value_dropdown.Visible = false;
            edit_Attr_new_value_text.Visible = true;
        }

    }

    /// <summary>
    /// Add new Element
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    protected void add_element_new_dropdown_SelectedIndexChanged(object
sender, EventArgs e)
    {
        if (add_element_new_dropdown.SelectedValue == "---NEW---")
        {
```

```
            add_New_Element_Text.Text = "";

            add_element_new_dropdown.Visible = false;
            add_New_Element_Text.Visible = true;
        }
    }

    /// <summary>
    /// Fired at Add Attribute Value dropdown
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    protected void
add_element_new_attributeValue_dropdown_SelectedIndexChanged(object
sender, EventArgs e)
    {
        if (add_element_new_attributeValue_dropdown.SelectedValue == "---
NEW---")
        {
            add_attr_value_text.Text = "";

            add_element_new_attributeValue_dropdown.Visible = false;
            add_attr_value_text.Visible = true;
        }
    }

    /// <summary>
    /// Fired at Add Attribute Name dropdow
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    protected void
add_element_new_attributeName_dropdown_SelectedIndexChanged(object sender,
EventArgs e)
    {
        if (add_element_new_attributeName_dropdown.SelectedValue == "---
NEW---")
        {
            add_attr_name_text.Text = "";

            add_element_new_attributeName_dropdown.Visible = false;
            add_attr_name_text.Visible = true;
        }

    }

    //*********************************************************
    // BUTTONS
    // *********************************************************


    /// <summary>
    /// Click on Edit
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    protected void Button1_Click(object sender, EventArgs e)
    {
        errorMsg.Visible = false;
        bottmPanel.Visible = false;
```

```
        //check if the selected is a .xml file
        if (!xmlFileNames.SelectedItem.Text.EndsWith("xml"))
        {
            errorMsg.Text = "Το αρχείο δεν είναι της μορφής .xml";
            errorMsg.Visible = true;
        }
        else
        {
            try
            {
                // SECTION 1. Create a DOM Document and load the XML data
into it.
                XmlDocument dom = new XmlDocument();
                dom.Load(@"/XML_Project_Final/XMLData/" +
xmlFileNames.SelectedItem.Text);

                // SECTION 2. Initialize the TreeView control.
                treeXML.Nodes.Clear();
                treeXML.Nodes.Add(new TreeNode(dom.DocumentElement.Name));
                TreeNode tNode = new TreeNode();
                tNode = treeXML.Nodes[0];

                // SECTION 3. Populate the TreeView with the DOM nodes.
                AddNode(dom.DocumentElement, tNode);
                treeXML.ExpandAll();
            }
            catch (XmlException xmlEx)
            {
                errorMsg.Text = xmlEx.Message;
                errorMsg.Visible = true;

            }
            catch (Exception ex)
            {
                errorMsg.Text = ex.Message;
                errorMsg.Visible = true;

            }
            //If you pass all tests, show panel with editor
            bottmPanel.Visible = true;
        }
    }

    /// <summary>
    /// Edit Element Button
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    protected void Button2_Click(object sender, EventArgs e)
    {
        // Only do this if a node is selected
        if ((treeXML.SelectedNode != null))
        {
            if (!editableBox1.Visible )
            {
                ReplaceElement(treeXML.SelectedNode.Text,
edit_element_new_dropdown.SelectedValue);
                edit_element_existing_name.Text =
edit_element_new_dropdown.SelectedValue;
            }
            else
```

```
            {
                if (editableBox1.Text.Length > 0)
                {
                    ReplaceElement(treeXML.SelectedNode.Text,
editableBox1.Text.Replace(" ", "_"));
                    edit_element_existing_name.Text = editableBox1.Text;
                }
                else
                {
                    infoBox.Text = "Παρακαλώ εισάγετε τιμή για το element
.";
                    infoBox.Visible = true;
                }
            }
            editableBox1.Visible = false;
            edit_element_new_dropdown.Visible = true;
            edit_element_new_dropdown.SelectedIndex = 0;
        }
        else{
            infoBox.Text = "Παρακαλώ επιλέξτε κάποιο βρόγχο από το tree
view";
            infoBox.Visible = true;
        }


    }
    /// <summary>
    /// Save XML File
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    protected void Button4_Click(object sender, EventArgs e)
    {
        try
        {
            TreeViewToXml.exportToXml(this.treeXML,
@"/XML_Project_Final/XMLData/" +
xmlFileNames.SelectedItem.Text.Replace(".xml", "") + "_" +
System.DateTime.Now.ToLongDateString().Replace(" ", "_") + ".xml");

            infoBox.Text = "Το αρχείο αποθηκεύτηκε με επιτυχία.";
            infoBox.Visible = true;

            xmlFileNames.Items.Clear();
            //Reload contents of dropdown
            //get whatever is on that directory - FULL paths
            string[] filePaths =
Directory.GetFiles(@"/XML_Project_Final/XMLData");

            // get just path names
            // and populate dropdown with filenames
            string[] fileNames = new string[filePaths.Length];
            for (int i = 0; i < filePaths.Length; i++)
            {
                fileNames[i] = Path.GetFileName(filePaths[i]);
                xmlFileNames.Items.Add(fileNames[i]);
            }
        }
        catch (XmlException xmlEx)
        {
            infoBox.Text = xmlEx.Message;
```

```
            infoBox.Visible = true;

        }


    }

    /// <summary>
    /// Add New Element button
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    protected void Button5_Click(object sender, EventArgs e)
    {
        //only do this if a node is selected
        if (treeXML.SelectedNode != null)
        {
            if (add_New_Element_Text.Visible)
            {
                if (add_New_Element_Text.Text.Length > 0)
                {
                    TreeNode tn = new
TreeNode(add_New_Element_Text.Text.Replace(" ", "_"));
                    treeXML.SelectedNode.ChildNodes.Add(tn);
                }
                else
                {
                    infoBox.Text = "Παρακαλώ εισάγετε κάποιο όνομα για το
element.";
                    infoBox.Visible = true;
                }
            }
            else
            {
                TreeNode tn = new
TreeNode(add_element_new_dropdown.SelectedValue);
                treeXML.SelectedNode.ChildNodes.Add(tn);
            }
        }
        else
        {
            infoBox.Text = "Παρακαλώ επιλέξτε κάποιο βρόγχο από το tree
view.";
            infoBox.Visible = true;
        }
    }



    /// <summary>
    /// EDIT Button Attributes - Commit changes in attributes
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    protected void Button7_Click(object sender, EventArgs e)
    {
        //Check to see if we have any attributes
        if (edit_Attr_existing_name_dd.SelectedIndex != -1)
        {
            // See if there is a new attribute or one of the dropdown
```

```
            string newName,newValue = "";
            if (edit_Attr_new_name_text.Visible) { newName =
edit_Attr_new_name_text.Text.Replace(" ", "_"); }
            else {newName = edit_new_attr_name_dropdown.SelectedValue; }
            if (edit_Attr_new_value_text.Visible) { newValue =
edit_Attr_new_value_text.Text.Replace(" ", "_"); }
            else {newValue = edit_new_attr_value_dropdown.SelectedValue; }

            //Populate TreeViiew
            if (newName != "" && newValue != "" )
            {
                ReplaceAttribute(treeXML.SelectedNode.Text,
edit_Attr_existing_name_dd.Text, edit_Attr_existing_value_text.Text,
newName, "\""+ newValue + "\"");
            }
            else
            {
                infoBox.Text = "Παρακαλώ εισάγετε μια τιμή για το
attribute.";
                infoBox.Visible = true;
            }
        }
        else
        {
            infoBox.Text = "Ο επιλεγμένος βρόγχος δεν έχει attributes.";
            infoBox.Visible = true;
        }
    }



    /// <summary>
    /// Add New atributes Button
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    protected void Button8_Click(object sender, EventArgs e)
    {
         //only do this if a node is selected
        if (treeXML.SelectedNode != null)
        {
            // See if there is a new attribute or one of the dropdown
            string newName, newValue = "";
            if (add_attr_name_text.Visible) { newName =
add_attr_name_text.Text.Replace(" ", "_"); }
            else { newName =
add_element_new_attributeName_dropdown.SelectedValue; }
            if (add_attr_value_text.Visible) { newValue =
add_attr_value_text.Text.Replace(" ", "_"); }
            else { newValue =
add_element_new_attributeValue_dropdown.SelectedValue; }

            //Populate TreeViiew
            if (newName != "" && newValue != "")
            {
                treeXML.SelectedNode.Text = treeXML.SelectedNode.Text + "
" + newName + " = \"" + newValue + "\"";
            }
            else
            {
```

```
                infoBox.Text = "παρακαλώ εισάγετε μια τιμή για τα
attributes.";
                infoBox.Visible = true;
            }

        }
        else
        {
            infoBox.Text = "Παρακαλώ επιλέξτε κάποιο βρόγχο από το tree
view.";
            infoBox.Visible = true;
        }

    }



    /// <summary>
    /// Delete Node Button
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    protected void Button3_Click(object sender, EventArgs e)
    {
        if (treeXML.SelectedNode != null)
        {
            //ensure you cant delete root
            if (treeXML.SelectedNode.Parent != null)
            {
            treeXML.SelectedNode.Parent.ChildNodes.RemoveAt(0);
            }
            else
            {
                infoBox.Text = "Δεν μπορείτε να διαγράψετε το root
element.";
                infoBox.Visible = true;
           }
        }
        else
        {
            infoBox.Text = "Παρακαλώ επιλέξτε κάποιο βρόγχο από το tree
view.";
            infoBox.Visible = true;
        }
    }

    /// <summary>
    /// Delete Attribute Button
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    protected void Button6_Click(object sender, EventArgs e)
    {
        //only do this if a node is selected and there are some attributes
        if ((treeXML.SelectedNode != null) &&
(delete_attr_name_drop.Items.Count >0))
        {
            RemoveAttribute(treeXML.SelectedNode.Value,
delete_attr_name_drop.SelectedValue, delete_attr_value_txt.Text);

            //Clear Controls
```

```
            delete_attr_name_drop.Items.Clear();
            delete_attr_value_txt.Text = "";

        }
        else
        {
            infoBox.Text = "Παρακαλώ επιλέξτε κάποιο βρόγχο με
attributes";
            infoBox.Visible = true;
        }
    }
}
```

Βιβλιογραφία και σύνδεσμοι αναφοράς

[1] J. Sharp, *Microsoft Visual C# 2005 Step by Step*, Microsoft Press, 2005.

[2] D. J. Reilly, *Programming Web Forms*, Microsoft Press, 2005.

[3] T. Thangarathinam, *Professional ASP.NET 2.0 XML*, John Wiley and Sons, 2006.

[4] Γ. Στασινόπουλος,, *Διαδίκτυο και εφαρμογές*, Εθνικό Μετσόβιο Πολυτεχνείο, 2007.

[5] Ι. Στ. Βενιέρης, *Δίκτυα Ευρείας Ζώνης 2η έκδοση*, Εκδόσεις Τζιόλα, 2007.

[6] D. Megginson, *Imperfect XML*, Pearson Education, 2006.

[7] E. T. Ray, *Learning XML*, O'Reilly, 2003.

[8] D. Esposito, *Programming ASP.NET*, Microsoft Press, 2003.

[9] J. Sanford, *Professional ASP.NET 2.0 Design*, John Wiley and Sons, 2007.

[10] A. Rusell,Jones, *Mastering ASP.NET with Visual C#*, John Wiley and Sons, 2002.

[11] XML Validator: http://www.validome.org/xml

[12] XML Tutorial: http://www.w3schools.com/xml/

[13] Microsoft Developer Network: http://msdn.microsoft.com

[14] Wikipedia: http://en.wikipedia.org/wiki/ASP.NET

[15] Wikipedia: http://en.wikipedia.org/wiki/C_Sharp_(programming_language)

[16] Developer Fusion Community: http://www.developerfusion.com/