**ΑΝΩΤΑΤΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΜΕΣΟΛΟΓΓΙΟΥ**
**ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΟΙΚΟΝΟΜΙΑΣ**
**ΤΜΗΜΑ ΕΦΑΡΜΟΓΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΣΤΗΝ ΔΙΟΙΚΗΣΗ ΚΑΙ ΣΤΗΝ ΟΙΚΟΝΟΜΙΑ**

# ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

*ΓΙΑ ΤΗΝ ΑΠΟΚΤΗΣΗ ΤΟΥ*

**ΠΤΥΧΙΟΥ ΕΦΑΡΜΟΣΤΗ ΠΛΗΡΟΦΟΡΙΚΗΣ ΣΤΗΝ ΔΙΟΙΚΗΣΗ ΚΑΙ ΣΤΗΝ ΟΙΚΟΝΟΜΙΑ**

## ΑΝΑΛΥΣΗ, ΣΧΕΔΙΑΣΗ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΠΛΗΡΟΦΟΡΙΑΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ΗΛΕΚΤΡΟΝΙΚΟΥ ΚΑΤΑΣΤΗΜΑΤΟΣ ΣΤΟ ΔΙΑΔΙΚΤΥΟ

***ΣΠΟΥΔΑΣΤΗΣ :***
**ΠΑΥΛΑΤΟΣ ΣΑΡΑΝΤΗΣ *Α.Μ.* : 9556**

***ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ :***
**Φείδας Χρήστος**



**ΜΕΣΟΛΟΓΓΙ 2007**

# Πρόλογος

Η παρούσα διπλωματική εργασία εκπονήθηκε για την απόκτηση του πτυχίου Εφαρμοστή Πληροφορικής Στην Διοίκηση Και Στην Οίκονομία, του τμήματος Εφαρμογών Πληροφορικής στην Διοίκηση και στην Οικονομία της Σχολής Διοίκησης και Οικονομίας, που βρίσκεται στο Ανώτατο Τεχνολογικό Ίδρυμα Μεσολογγίου.

# Περιεχόμενα

# Σχήματα, Εικόνες, Διαγράμματα

# 1 ΕΙΣΑΓΩΓΗ

## 1.1 Ορισμός Πληροφοριακού Συστήματος

Το Πληροφοριακό Σύστημα ( Π.Σ ) ορίζεται ως ένα σύστημα, που αποτελείται απο **υποσυστημάτα,** δέχεται σαν είσοδο **δεδομένα,** τα οποία τα αποθηκεύει και τα επεξεργάζεται, έτσι ώστε να παράγει **πληροφορίες**, σύμφωνα με τις συγκεκριμένες απαιτήσεις του πληροφοριακού σύστηματος, που έχουν καθοριστεί από τους **χρήστες** του. Ένα Π.Σ έχει κοινωνικό χαρακτήρα και μπορεί να δημιουργηθεί με την χρήση ηλεκτρονικών υπολογιστών ή χωρίς την χρήση τους.

Για την καλύτερη κατανόηση του παραπάνω ορισμού, επεξηγούνται διάφορες εννοιές του :

- **Υποσυστήματα** : είναι μέρη του πληροφοριακού συστήματος που αλληλεπιδρούν μεταξύ τους, έτσι ώστε το πληροφοριακό σύστημα να επιτύχει τον στόχο του.
- **Δεδομένα** : είναι η είσοδος του πληροφοριακού συστήματος, μπορεί να είναι κάθε είδους ψηφιοποιημένης μορφής όπως κείμενο, φωτογραφίες κ.ά. Τα δεδομένα εισάγονται απο τους χρήστες του Π.Σ ή σε ορισμένες περιπτώσεις απο το ίδιο το Π.Σ.
- **Πληροφορίες** : είναι η έξοδος του Π.Σ και μπορεί να είναι κάθε είδους ψηφιοποιημένης. Το Π.Σ παράγει πληροφορίες, ύστερα απο επεξεργασία των δεδομένων που δέχεται, με σκοπό το Π.Σ να καλύψει τις ανάγκες των χρηστώντου και να επιτύχει τον στόχο του.
- **Χρήστες** : είναι οι άνθρωποι που χρησιμοποιούν το Π.Σ. Οι χρήστες του Π.Σ καθορίζουν τις απαιτήσεις και τον σκοπό του Π.Σ, σύμφωνα με τις αναγκές τους. Χρήστες ενός Π.Σ μπορεί να είναι οι πελάτε ή οι προμηθευτές μιάς επιχείρησης, ο διαχειριστής του Π.Σ, κ.ά.

# 1.2 Φάσεις Ανάπτυξης Πληροφοριακού Συστήματος

Κατά την ανάπτυξη ενός πληροφοριακού συστήματος ακολουθούνται διάφορες φάσεις, οι οποίες είναι σημεία ελέγχου για την ορθή ανάπτυξη του πληροφοριακού συστήματος. Το τέλος μιας φάσης ανάπτυξης σηματοδοτεί την αρχή μιας άλλης φάσης ανάπτυξης, κατά αυτόν τον τρόπο η μια φάση διαδέχεται την άλλη, μέχρι την ολοκλήρωση του πληροφοριακού συστήματος.

Οι φάσεις ανάπτυξης πληροφοριακού συστήματος στην βιβλιογραφία μπορεί να αναφέρονται ως "συμβατική ανάλυση συστημάτων", είτε ως "παραδοσιακή ανάλυση συστημάτων", είτε ως "κύκλος ανάπτυξης και ζωής πληροφοριακών συστημάτων" και πιο συχνά ως "μοντέλο καταρράκτη".

Αν και υπάρχουν διαφορετικές γνώμες για τις φάσεις ανάπτυξης πληροφοριακού συστήματος, η πιο διαδεδομένη και βασική δομή τους είναι η ακόλουθη :

- **Μελέτη σκοπιμότητας**
- **Έρευνα συστημάτων**
- **Ανάλυση συστημάτων**
- **Σχεδιασμός συστημάτων**
- **Υλοποίηση συστημάτων**
- **Αναθεώρηση και συντήρηση**

## 1.2.1 Μελέτη Σκοπιμότητας

Κατά την φάση της μελέτης σκοπιμότητας εξετάζονται το παρόν σύστημα(εάν υπάρχει), οι απαιτήσεις που πρέπει να ικανοποίησει, τα προβλήματα που προκύπτουν από τις απαιτήσεις που πρέπει να ικανοποιηθούν και έρευνα τις διαφορές εναλλακτικές λύσεις που μπορεί να προκύψουν για τη λύση του προβλήματος.

Η φάση της μελέτης σκοπιμότητας είναι η βασικότερη φάση, αφού είναι η πρώτη φάση της ανάπτυξης του πληροφοριακού συστήματος και πρέπει να εγγυηθεί, ότι το προτεινόμενο σύστημα είναι εφικτό από άποψη νομική, οργανωτική, κοινωνική, τεχνική και οικονομική. Την φάση της μελέτης σκοπιμότητας, την συναντάμε συνήθως στην

ανάπτυξη μεγάλων και πολύπλοκων πληροφοριακών συστημάτων, που προορίζονται π.χ. για δημόσιους οργανισμούς ή για ιδιωτικές εταιρείες. Για τις ανάγκες της παρούσας διπλωματικής και του συστήματος, το οποίο θα αναπτύξουμε η φάση της μελέτης σκοπιμότητας είναι περιττή και θα παραληφθεί.

### 1.2.2 Έρευνα Συστημάτων

Μετά την φάση της μελέτης σκοπιμότητας ακολουθεί η φάση της έρευνας συστημάτων, η οποία είναι μια λεπτομερής έρευνα για την εφαρμογή του πληροφοριακού συστήματος. Κατά την φάση της έρευνας συστημάτων συλλέγονται πιο λεπτομερείς πληροφορίες, απ' αυτές που συλλέχθηκαν κατά την φάση της μελέτης σκοπιμότητας.

Εξετάζονται ζητήματα όπως οι λειτουργικές απαιτήσεις του υπάρχοντος ή του νέου συστήματος και αν αυτές ικανοποιούνται ή μπορούν να ικανοποιηθούν, διάφοροι περιορισμοί που μπορεί να προκύψουν, ο όγκος των δεδομένων που θα επεξεργαστεί το σύστημα, διάφορες περιπτώσεις εξαιρέσης και τα προβλήματα που εμφανίζουν οι παρούσες μέθοδοι εργασίας.

Επίσης την φάση της έρευνας συστημάτων την συναντάμε συνήθως στην ανάπτυξη μεγάλων και πολύπλοκων πληροφοριακών συστημάτων, όπως και την φάση της μελέτης σκοπιμότητας. Οπότε για τις ανάγκες της παρούσας διπλωματικής η φάση της έρευνας συστημάτων είναι περιττή και θα παραληφθεί και αυτή.

### 1.2.3 Ανάλυση Συστημάτων

Η φάση της αναλύσης συστημάτων είναι βασική, διότι κατά την διάρκεια αυτής της φάσης γίνεται προσπάθεια να γίνουν κατανοητές όλες οι πτυχές του πληροφοριακού συστήματος, με τη βοήθεια των πληροφοριών που συλλέχθηκαν σε αυτήν και στις προηγούμενες δύο φάσεις.

Ο σκοπός αυτής της φάσης είναι να παρουσιάσει ξεκάθαρα γιατί αναπτύχθηκε το σύστημα με αυτόν τον τρόπο, με τον οποίο αναπτύχθηκε εάν υπάρχει παρόν σύστημα ή τον τρόπο που πρέπει να αναπτυχθεί ένα νέο σύστημα απ' την αρχή ή για την αντικατάσταση του παλαιότερου. Σε αυτή την φάση θα τεθούν οι στόχοι για τον σχεδιασμό του συστήματος και θα εξακριβωθούν ποιες είναι ακριβώς οι απαιτήσεις του.

### 1.2.4 Σχεδιασμός Συστημάτων

Κατά την φάση του σχεδιασμού συστημάτων πραγματοποιείται ο σχεδιασμός ολόκληρου του συστήματος, αφορά τόσο τα μέρη του υπολογιστή όσο και τα χειρωνακτικά μέρη, εάν υπάρχουν, του συστήματος.

Σε αυτή τη φάση τα βασικότερα ζητήματα, τα οποία μας απασχολούν είναι τα εξής:

- Τα δεδομένα εισόδου του συστήματος και ο τρόπος που θα γίνει η εισαγωγή τους στο σύστημα.
- Τα δεδομένα εξόδου του συστήματος και ο τρόπος που θα γίνει η εξαγωγή τους από σύστημα.
- Οι διαδικασίες του συστήματος, οι οποίες μετατρέπουν τα δεδομένα εισόδου σε δεδομένα εξόδου.
- Η δομή των ηλεκτρονικών και των μη ηλεκτρονικών αρχείων, εάν υπάρχουν, του συστήματος.
- Η ασφάλεια του συστήματος.
- Και τέλος οι διάφοροι έλεγχοι, που πρέπει να γίνουν για τη σωστή λειτουργία του συστήματος και τα σχέδια, που θα μας οδηγήσουν στην επόμενη φάση της υλοποίησης του.

### 1.2.5 Υλοποίηση Συστημάτων

Στην φάση της υλοποίησης συστημάτων συγκεντρώνονται όλες οι απαραίτητες πληροφορίες, που συλλέχθηκαν κατά τη διάρκεια των προηγούμενων φάσεων και με διάφορες διαδικασίες οδηγούμαστε στην υλοποίηση του συστήματος.

Σε αυτή τη φάση ανάπτυξης του συστήματος πρέπει να ελεγχθούν τα διαθέσιμα συστήματα υλικού και λογισμικού, εάν υπάρχουν ή να αγοραστούν και να εγκατασταθούν καινούργια, σύμφωνα με τις απαιτήσεις του συστήματος. Εάν υπάρχουν προγράμματα υπολογιστών, ο σχεδιασμός τους, η κωδικοποίηση τους καθώς και ο έλεγχός τους πρέπει να γίνει από συγκεκριμένους ανθρώπους, που ονομάζονται προγραμματιστές υπολογιστών. Εναλλακτικά μπορούν να αγοραστούν και να χρησιμοποιηθούν, ως μέρη του συστήματος κάποια έτοιμα πακέτα εφαρμογών, που υπάρχουν στην αγορά. Σε αυτή την περίπτωση δεν χρειάζεται η συμβολή των προγραμματιστών, αλλά κάποιων άλλων ανθρώπων που θα αποφασίσουν ποιο πακέτο εφαρμογών, ανταποκρίνεται καλύτερα στις απαιτήσεις του συστήματος.

Άλλα βασικά ζητήματα, που μας απασχολούν κατά τη διάρκεια αυτής της φάσης είναι τα εξής:

- **Ο έλεγχος ποιότητας**, κατά τον οποίο εξετάζεται το υλικό και το λογισμικό, καθώς και οι χειρωνακτικές διαδικασίες, εάν υπάρχουν, στο ποσό εξυπηρετούν και ανταποκρίνονται στις ανάγκες των χρηστών.
- **Η εκπαίδευση και η κατάρτιση των χρηστών**, που θα χρησιμοποιήσουν το σύστημα.
- **Η τεκμηρίωση του συστήματος**.
- **Οι διαδικασίες ασφαλείας του συστήματος**, έτσι ώστε να εμποδιστεί η μη εξουσιοδοτημένη πρόσβαση και η αποκατάσταση του συστήματος να είναι δυνατή, εάν υπάρξει πρόβλημα.
- Και τέλος **η μετάβαση από το παλαιό στο νέο σύστημα**, εάν υπάρχει φυσικά παλαιό.

### 1.2.6 Αναθεώρηση Και Συντήρηση

Σε αυτή την φάση ανάπτυξης του συστήματος, το σύστημα έχει τεθεί πλέον σε λειτουργία. Κατά τον πρώτο καιρό λειτουργίας συστήματος, επικρατεί ένα κλίμα αλλαγής και αναθεώρησης ολόκληρου του συστήματος. Με σκοπό την εξασφάλιση της συνεχούς και αποδοτικής λειτουργίας του συστήματος, καθώς και να εξασφαλίσει ότι έχει ανταποκριθεί πλήρως στις λειτουργικές απαιτήσεις, οι οποίες είχαν καθοριστεί σε προηγούμενες φάσεις. Ακόμα πρέπει να καθορισθούν οι διαδικασίες, οι οποίες θα είναι ασφαλιστική δικλίδα για την σωστή και ασφαλή συντήρηση του συστήματος.

Για τις ανάγκες της παρούσας διπλωματικής, αυτή η φάση είναι περιττή και θα παραληφθεί, οπότε δεν αναλύεται περαιτέρω.

## 1.3 Αντικείμενο Της Διπλωματικής

Το αντικείμενο αυτής της διπλωματικής, είναι να παρουσιάσει τον τρόπο με τον οποίο γίνεται η ανάπτυξη ενός πληροφοριακού συστήματος. Συγκεκριμένα θα αναπτυχθεί ένα πληροφοριακό σύστημα για ένα ηλεκτρονικό κατάστημα, που οι χρήστες- πελάτες του θα έχουν πρόσβαση σε αυτό, μέσω του διαδικτύου(Internet).

Η ανάπτυξη του συστήματος θα γίνει σύμφωνα με τις φάσεις ανάπτυξης ενός πληροφοριακού συστήματος, που αναλύθηκαν παραπάνω. Για τις ανάγκες αυτού του συστήματος, θα χρησιμοποιηθούν οι φάσεις της ανάλυσης, του σχεδιασμού και της υλοποίησης συστημάτων, ενώ οι φάσεις της μελέτης σκοπιμότητας, της έρευνας συστημάτων και της αναθεώρησης και συντήρησης θα παραληφθούν, διότι εκτός ότι είναι μακρόχρονιες φάσεις στην ανάπτυξη πληροφοριακών συστημάτων και τις συναντάμε σε μακροχρόνια και μεγάλα πληροφοριακά συστήματα, που αφορούν μεγάλους οργανισμούς και εταιρείες, στην περίπτωση του συστήματος που θα γίνει ανάπτυξη δεν έχουν και τόσο μεγάλη σημασία.

# 1.4 Διάρθρωση Του Τόμου

Το πρώτο κεφάλαιο είναι εισαγωγικό. Δίνεται ένας **ορισμός** για τα πληροφοριακά συστήματα και αναλύονται οι **φάσεις ανάπτυξης** ενός πληροφοριακού συστήματος. Ακόμα περιγράφεται **το αντικείμενο**, με το οποίο θα ασχοληθεί η παρούσα διπλωματική.

Στο δεύτερο κεφάλαιο, το οποίο είναι η ανάλυση του συστήματος που θα αναπτύξουμε, αναλύονται **το διαδίκτυο (Internet)**, η γλώσσα προγραμματισμού **HTML**, η τεχνολογία **ASP**, η γλώσσα διαχείριση βάσεων δεδομένων **SQL** και συστήματα και τεχνολογίες για την **ασφάλεια** των πληροφοριακών συστημάτων.

Στο τρίτο κεφάλαιο που αποτελεί τον σχεδιασμό του συστήματος, περιγράφονται η γλώσσα μοντελοποίησης **UML**, τα **διαγράμματα ροής δεδομένων** και το μοντέλο **οντοτήτων - τη συσχετίσεων**.

Στο τελευταίο κεφάλαιο, που είναι η υλοποίηση του συστήματος αναφέρονται στα **εργαλεία** υλοποίησης που χρησιμοποιήθηκαν, οι τεχνολογίες **.NET** καθώς και παρουσιάζεται η υλοποίηση, παραθέτωντας τον κώδικα που χρησιμοποιήθηκε.

# 2 Ανάλυση Του Συστήματος

## 2.1 Το Διαδίκτυο τους (Internet) Και Σχετική Ορολογία

Το διαδίκτυο (Internet) είναι ένα παγκόσμιο δίκτυο ηλεκτρονικών υπολογιστών, οι οποίοι ανταλλάζουν δεδομένα μεταξύ τους, σύμφωνα με ένα κοινό πρωτόκολλο επικοινωνίας το **TCP/IP (Transmission Control Protocol/Internet Protocol)**. Οι χρήστες του διαδικτύου έχουν την δυνατότητα, να αντλήσουν χρήσιμες πληροφορίες εύκολα και γρήγορα από μια τεράστια βάση πληροφοριών, να αποστείλουν ή να λάβουν ηλεκτρονική αλληλογραφία, να ανταλλάξουν ηλεκτρονικά αρχεία και γενικά να χρησιμοποιήσουν μια πληθώρα υπηρεσιών που προσφέρει το διαδίκτυο.

### 2.1.1 Τα Πρωτόκολλα Μεταφοράς

Η μεταφορά της πληροφορίας μέσω του διαδικτύου, σε οποιαδήποτε μορφή είναι η πληροφορία (π.χ. ηλεκτρονικό αρχείο, ηλεκτρονικό μήνυμα κ.α), επιτυγχάνεται με τη χρήση ενός κοινού για κατάλληλου πρωτόκολλου μεταφοράς. Το πρωτόκολλο μεταφοράς, εγγυάται και δεσμεύει τους δύο υπολογιστές που ανταλλάζουν δεδομένα, πώς θα τα αποστείλουν και πώς θα τα λάβουν. Τα πιο γνωστά πρωτόκολλα μεταφοράς, που χρησιμοποιούνται περισσότερο είναι τα εξής:

- **HTTP (Hyper Text Transfer Protocol)**, το οποίο είναι για τη μεταφορά της πληροφορίας, μέσω της υπηρεσίας της περιήγησης (σερφάρισμα) στον παγκόσμιο ιστό.

- **SMTP (Simple Mail Transfer Protocol)**, το οποίο είναι για τη μεταφορά της πληροφορίας, μέσω της υπηρεσίας του ηλεκτρονικού ταχυδρομείου (e-mail).
- **FTP (File Transfer Protocol)**, το οποίο είναι για τη μεταφορά της πληροφορίας, μέσω της υπηρεσίας της μεταφοράς των ηλεκτρονικών αρχείων.
- **NNTP (Network News Transfer Protocol)**, το οποίο είναι για τη μεταφορά της πληροφορίας, με τη χρήση του Usenet και των ομάδων συζητήσεων (Newsgroups).

### 2.1.2 Παγκόσμιος Ιστός (WWW – World Wide Web)

Ο παγκόσμιος ιστός είναι το δημοφιλέστερο και το πιο αναπτυσσόμενο μέρος του διαδικτύου. Πολλοί πιστεύουν ότι ο όρος διαδίκτυο και ο όρος παγκόσμιος ιστός είναι το ίδιο, φυσικά αυτή είναι λανθασμένη άποψη και σαφώς ο παγκόσμιος ιστός είναι ένα μέρος του διαδικτύου, το οποίο είναι το μέσο για την εύκολη και γρήγορη ανάκτηση του μεγάλου όγκου πληροφοριών, που διατίθενται μέσω του διαδίκτυου. Το πρωτόκολλο μεταφοράς που χρησιμοποιεί ο παγκόσμιος ιστός, για τη μεταφορά της πληροφορίας είναι το HTTP (Hyper Text Transfer Protocol).

### 2.1.3 Ιστιοσελίδα (Web Page)

Η πληροφορία που διατίθεται, μέσω του παγκόσμιου ιστού, εμφανίζεται μορφοποιημένη με τη γλώσσα HTML (Hyper Text Markup Language) σε μορφή ιστιοσελίδας (Web Page) και έχει την κατάληξη .htm ή .html. Η ιστιοσελίδα εκτός από στατικό κείμενο, μπορεί να περιέχει ήχο, εικόνες, πολύμεσα κ.α. Εκτός από την μορφοποίηση .html, υπάρχουν και άλλες μορφοποίησεις ιστοσελίδων, όπως .php, .asp κ.α.

### 2.1.4 Διακομιστής Ιστού (Web Server)

Ο διακομιστής ιστού είναι ένας ηλεκτρονικός υπολογιστής, που περιέχει ιστιοσελίδες σε μορφή ηλεκτρονικού αρχείου. Αυτός ο ηλεκτρονικός υπολογιστής, μέσω του κατάλληλου λογισμικού και των κατάλληλων δικτυακών συνδέσεων, διαθέτει τις ιστοσελίδες στο παγκόσμιο ιστό. Ο διακομιστής ιστού δέχεται αιτήματα από τους χρήστες του διαδικτύου, που θέλουν να δουν τις ιστιοσελίδες του και αυτος με την σειρά του τα ικανοποιεί, προβάλλοντας τις ζητούμενες ιστιοσελίδες.

### 2.1.5 Φυλλομετρητής(Web Browser)

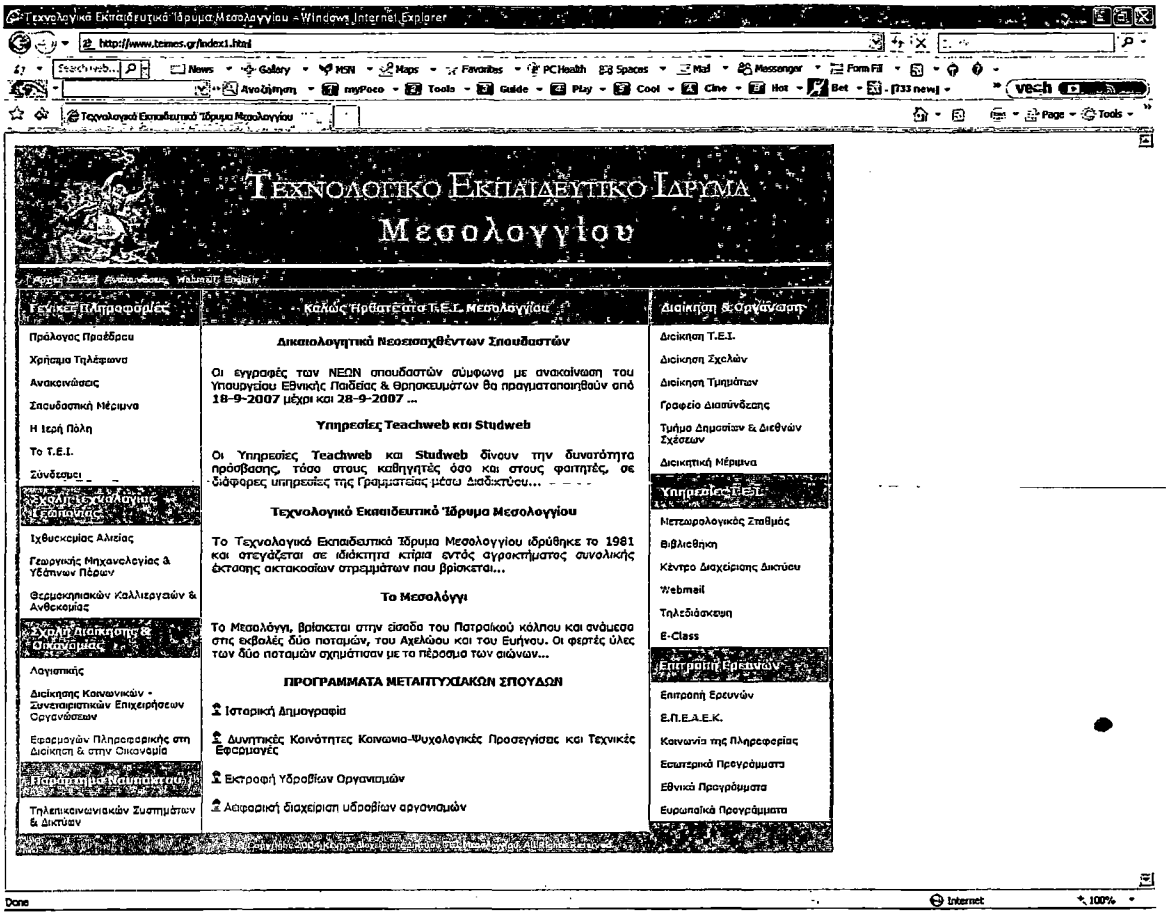
Ο φυλλομετρητής συμπεριλαμβάνεται στο λογισμικό ενός ηλεκτρονικού υπολογιστή. Η χρησιμότητα του είναι ότι προβάλλει στον χρήστη που τον χρησιμοποιεί, τις ζητούμενες από αυτόν ιστιοσελίδες. Ο χρήστης εισάγει στον φυλλομετρητή τη ζητούμενη ιστοσελίδα και ο φυλλομετρητής ανακτά τη συγκεκριμένη ιστοσελίδα από το διακομιστή ιστού που την περιέχει, την λαμβάνει και την προβάλλει στο παράθυρο του χρήστη. Υπάρχουν διάφοροι φυλλομετρητές που έχουν κατασκευαστεί από διαφορετικές εταιρείες, οι πιο γνωστοί απ' αυτούς είναι:

- **Microsoft Internet Explorer**
- **Netscape Navigator**
- **Firefox Mozilla**

### 2.1.5.1 Microsoft Internet Explorer

Ο Internet Explorer είναι ο δημοφιλέστερος φυλλομετρητής, αφού διατίθεται μαζί με το λειτουργικό σύστημα της Microsoft τα Windows. Η πιο πρόσφατη έκδοση του, είναι ο Internet Explorer 7 και διατίθεται δωρεάν από την εταιρία Microsoft στην παρακάτω ηλεκτρονική διεύθυνση:

**http://www.microsoft.com/windows/downloads/ie/getitnow.mspx**




**2.1: Το βασικό παράθυρο του Microsoft Internet Explorer**

## 2.1.5.2 Netscape Navigator

Ο Netscape Navigator είναι ένα από τα δημοφιλέστερους και πιο εύχρηστους φυλλομετρητές. Η πιο πρόσφατη έκδοση του, είναι ο Netscape Navigator 9 και διατίθεται δωρεάν από την εταιρία Netscape στην παρακάτω ηλεκτρονική διεύθυνση:

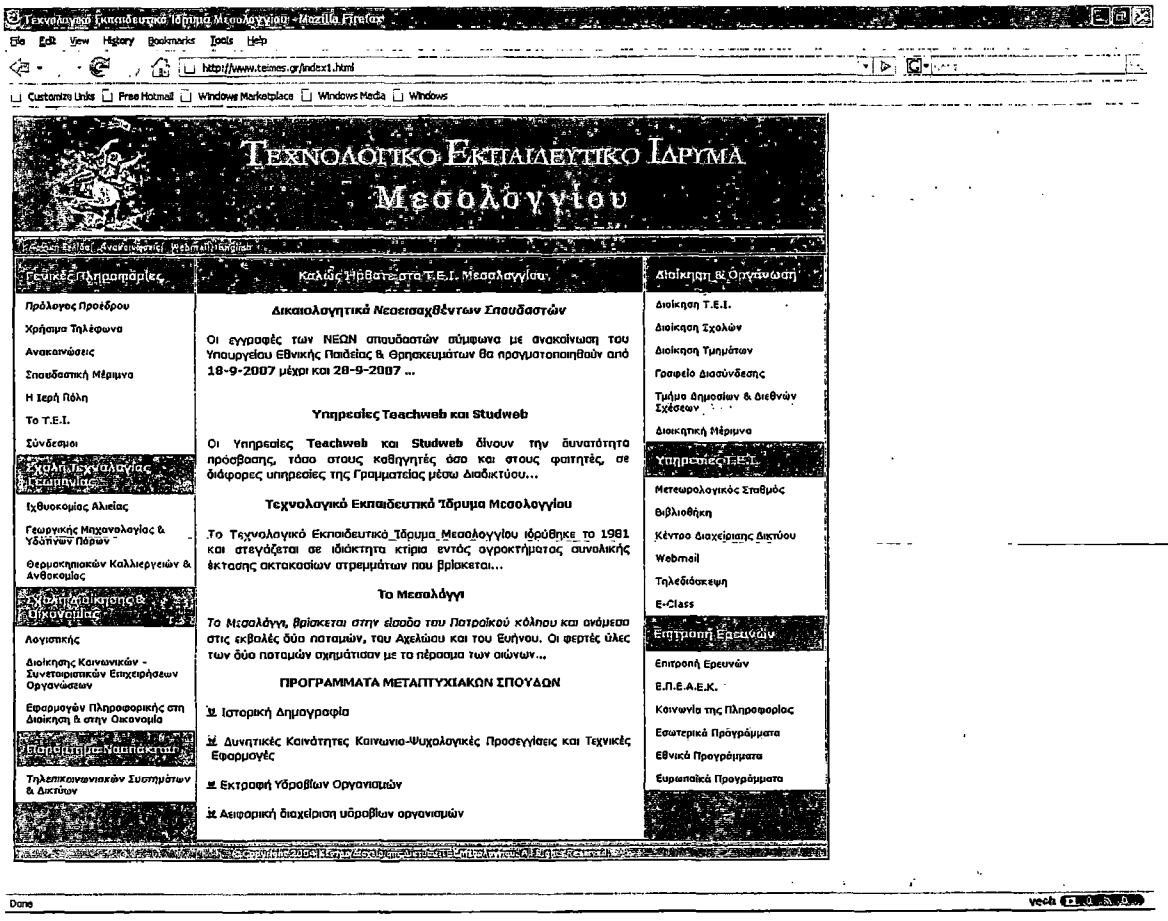
http://browser.netscape.com/


**2.2: Το βασικό παράθυρο του Netscape Navigator**

## 2.1.5.3 Firefox Mozilla

Ο Firefox Mozilla είναι σχετικά καινούργιος φυλλομετρητής, λόγω της ευχρησίας του τον χρησιμοποιούν πολλοί χρήστες του διαδικτύου και έχει γίνει ανταγωνιστικός. Η τρέχουσα έκδοση του, είναι ο Firefox Mozilla 2 και διατίθεται δωρεάν από την εταιρία Firefox στην παρακάτω ηλεκτρονική διεύθυνση:

**http://www.mozilla.com/en-US/firefox/**




**2.3: Το βασικό παράθυρο του Firefox Mozilla**

## 2.1.6 Ηλεκτρονική Διευθύνση Ιστού (URL)

Η ηλεκτρονική διεύθυνση ιστού ή αλλιώς **URL (Uniform Resource Locator)** αντιπροσωπεύει με μοναδικό τρόπο μια ιστιοσελίδα. Το URL είναι αρκετό για να ανακτηθεί μια ιστιοσελίδα από τον διακομιστή ιστού στον οποίο φιλοξενείται. Η μορφή μιας ηλεκτρονικής διεύθυνσης είναι η ακόλουθη:

http://www.ypes.gr/Services/eea.htm

και αποτελείται από τα εξής μέρη:

- **http://** - υποδηλώνει το πρωτόκολλο μεταφοράς HTTP, με το οποίο γίνεται η μεταφορά της πληροφορίας.

- **www.ypes** - υποδηλώνει το όνομά του διακομιστή ιστού, που περιέχει την ιστιοσελίδα. Το όνομα μπορεί να είναι οποιοδήποτε, αλλά το 90% των διακομιστών ιστού, χρησιμοποιούν το www μπροστά.
- **.gr** - υποδηλώνει το περιεχόμενο της ιστιοσελίδας π.χ.
  - ✓ .com - εμπορικό περιεχόμενο,
  - ✓ .edu - εκπαιδευτικό περιεχόμενο,
  - ✓ .gov - κυβερνητικό περιεχόμενο,
  - ✓ .org - μη κερδοσκοπικό περιεχόμενο,

  ή υποδηλώνει τη χώρα από την οποία προέρχεται η ιστιοσελίδα π.χ.
  - ✓ .gr - για την Ελλάδα,
  - ✓ .au - για την Αυστραλία κ.ά.
- **/Services/** - υποδηλώνει το όνομά του φακέλου, , ο οποίος περιέχει την ιστιοσελίδα σε μορφή αρχείου.
- **eea.htm** - υποδηλώνει το όνομά του αρχείου της ιστοσελίδας.

### 2.1.7 Τοποθεσία Ιστού (Web Site)

Η τοποθεσία ιστού είναι μια συλλογή ιστιοσελίδων που φιλοξενούνται σε έναν διακομιστή ιστού και αφορούν μια εταιρεία, έναν οργανισμό, έναν ιδιώτη ή άλλες ομάδες. Μια τοποθεσία ιστού μπορεί να φιλοξενηθεί σε διάφορες εταιρείες, που προσφέρουν αυτήν την υπηρεσία (**Web hosting**) ή αν πρόκειται για κάποια μεγάλη εταιρία ή κάποιο κλειστό σύστημα, να αναπτυχθεί ένας διακομιστής ιστού που θα τα φιλοξενήσει.

### 2.1.8 Υπερσύνδεσμος (Hyperlink)

Το βασικότερο χαρακτηριστικό που διευκολύνει την περιήγηση στον παγκόσμιο ιστό είναι η χρήση της δομής του υπερκειμένου (**HyperText**). Η αναζήτηση της πληροφορίας και η εναλλαγή μέσα στο υπερκείμενο γίνεται μέσω των υπερσυνδέσμων, οι οποίοι βρίσκονται σε εμφάνη σημεία μιας ιστοσελίδας.

Τις περισσότερες των περιπτώσεων πρόκειται για υπογραμμισμένο κείμενο με διαφορετικό χρώμα από το κείμενο της ιστοσελίδας ή σε διαφορετικό χρώμα φοντού ή ακομα μπορεί να είναι κάποιο πολυμέσο όπως εικόνα. Ο Υπερσύνδεσμος είναι το μέσο που βοηθάει τον χρήστη του παγκόσμιου ιστού να εναλλαζετέ τις ιστιοσελίδες με κάποια λογική σειρά και να εμβαθύνει την πληροφορία, την οποία αναζητεί.

# 2.2 Η Γλώσσα Προγραμματισμού HTML

Οι ιστιοσελίδες που εμφανίζονται στο διαδίκτυο είναι αρχεία γραμμένα με την γλώσσα προγραμματισμού **HTML**, η οποία είναι το ακρωνύμιο των λέξεων **Hyper Text Markup Language** και βασίζεται στην γλώσσα **SGML (Standard Generalized Markup Language)** που είναι ένα τεράστιο σύστημα επεξεργασίας εγγράφων.

Η γλώσσα HTML ορίζεται από ένα σύνολο εντολών, οι οποίες περικλείονται στα σύμβολα <>, που αποκαλούνται ετικέτα **(Tag)** και καθορίζουν την μορφή της ιστοσελίδας. Οι ιστοσελίδες που έχουν γραφτεί με την HTML είναι απλά αρχεία κειμένου σε μορφή **ASCII** και δεν χρειάζονται κάποιο ειδικό πρόγραμμα για να διαβαστούν, αλλά μπορούν να διαβαστούν από οποιονδήποτε συντακτή που υποστηρίζει απλό κείμενο, όπως είναι το **Notepad**, το **Wordpad** και το **Write**. Οι φυλλομετρητές εκτός από τη λειτουργία της ανάκτησης ιστοσελίδων από το παγκόσμιο ιστό, λειτουργούν επίσης και σαν μορφοποιητές για την HTML, δηλαδή ο φυλλομετρητής διερμηνεύει τις ετικέτες της HTML και μορφοποιεί το κείμενο και τις εικόνες στο παράθυρο του χρήστη.

## 2.2.1 Οι Ετικέτες (Tags)

Οι ετικέτες **(Tags)** της **HTML** καθορίζουν τα περιεχόμενα, τη δομή και τη μορφοποίηση της ιστοσελίδας. Η βασική μορφή των περισσότερων ετικετών είναι η εξής:

**< όνομα Tag> κείμενο </ όνομα Tag>**

Η γενική μορφή των ετικετών της HTML είναι μια ετικέτα αρχής ή αλλιώς ανοίγματος και μια ετικέτα τέλους ή αλλιώς κλεισίματος, στις οποίες περικλείεται κείμενο που το επηρεάζουν ανάλογα με τη λειτουργία της ετικέτας. Η ετικέτα αρχής ενεργοποιεί μια λειτουργία, για το κείμενο που περικλείει και η ετικέτα τέλους απενεργοποοιεί τη λειτουργία. Οι ετικέτες αρχής και τέλους έχουν το ίδιο όνομα, η διαφορά τους είναι ο χαρακτήρας /, που περιέχεται στις ετικέτες τέλους σαν πρόθεμα.

### 2.2.1.1 Η Ετικέτα <HTML>

Η ετικέτα <HTML> ελέγχει την δομή της ιστοσελίδας και καθορίζει ότι το περιεχόμενο της περιέχει κώδικα γραμμένο στη γλώσσα HTML. Όλες οι εντολές και το κείμενο πρέπει να τοποθετείται ανάμεσα στις ετικέτες αρχής και τέλους <HTML>, όπως φαίνεται παρακάτω:

```
<HTML>
    ... κείμενο και εντολές της ιστιοσελίδας...
</HTML>
```

### 2.2.1.2 Η Ετικέτα <HEAD>

Ανάμεσα στις ετικέτες αρχής και τέλους της <HEAD> περικλείεται ο πρόλογος για το υπόλοιπο της ιστοσελίδας. Στην ετικέτα αυτή τοποθετείται συνήθως ο τίτλος της ιστοσελίδας και δεν τοποθετείται ποτέ κείμενο.

```
<HTML>
    <HEAD>
    </HEAD>
        ...
</HTML>
```

### 2.2.1.3 Η Ετικέτα <BODY>

Στην ετικέτα <BODY> περικλείονται όλα τα υπόλοιπα στοιχεία της ιστιοσελίδας, όπως κείμενο, εικόνες, υπερσύνδεσμοι κ.α.

```
<HTML>
    <HEAD>
    </HEAD>
```

```
<BODY>
    ...
</BODY>
</HTML>
```

### 2.2.1.4 Η Ετικέτα <TITLE>

Ο τίτλος της ιστοσελίδας περικλείεται ανάμεσα στην ετικέτα <TITLE> και περιγράφει το περιεχόμενο και τον σκοπό της ιστιοσελίδας. Ο τίτλος εμφανίζεται στη γραμμή τίτλου του φυλλομετρητή και περικλείεται πάντα μέσα στην ετικέτα <HEAD>.

```
<HTML>
    <HEAD>
        <TITLE>
            Ο τίτλος της ιστιοσελίδας
        </TITLE>
    </HEAD>
    <BODY>
        ...
    </BODY>
</HTML>
```

### 2.2.1.5 Οι Ετικέτες Επικεφαλίδων

Στην HTML για να ορίσουμε επικεφαλίδες (headings) χρησιμοποιούμε την ετικέτα <H> μαζί με ένα αριθμό <H1>, ο αριθμός δηλώνει το μέγεθος της επικεφαλίδας. Στην HTML ορίζονται 6 μεγέθη επικεφαλίδας, καθώς αυξάνουμε από H1 σε H6 ελαττώνεται το μέγεθος της επικεφαλίδας.

```
<H1> Η μεγαλύτερη επικεφαλίδα </H1>
```

**<H2> Η δεύτερη μεγαλύτερη επικεφαλίδα </H2>**

**<H3> Η τρίτη μεγαλύτερη επ και δεν λύνει 1/7 του ικεφαλίδα </H3>**

**<H4> Η τέταρτη μεγαλύτερη επικεφαλίδα </H4>**

**<H5> Η πέμπτη μεγαλύτερη επικεφαλίδα </H5>**

**<H6> Η μικρότερη επικεφαλίδα </H6>**

### 2.2.1.6 Η Ετικέτα <P>

Για να ορίσουμε παράγραφο στην HTML περικλείουμε ολόκληρη την παράγραφο-ανάμεσα στις ετικέτες αρχής και τέλους της <P>. Η χρήση της ετικέτας τέλους </P> είναι προαιρετική, καθώς μπορούμε να αλλάξουμε παράγραφο χρησιμοποιώντας την ετικέτα <P> στην αρχή της νέας παραγράφου.

**<P> ...ολόκληρη η παράγραφος... </P>**

### 2.2.1.7 Οι Ετικέτες Μορφοποίησης Κειμένου

Οι ετικέτες μορφοποίησης κείμενο που καθορίζουν την μορφή του κειμένου, το οποίο περικλείεται ανάμεσα στις ετικέτες αρχής και τέλους τους και μπορούν να χρησιμοποιηθούν ταυτόχρονα στο ίδιο κείμενο. Οι ετικέτες μορφοποίησης κειμένων είναι οι εξής:

- **<B>**, έντονη γραφή
- **<I>**, πλάγια γραφή
- **<U>**, υπογραμμισμένο κείμενο
- **<S>**, διακριτή διαγραφή
- **<BIG>**, μεγαλύτερο μέγεθος κειμένου
- **<SMALL>**, μικρότερο μέγεθος κειμένου
- **<SUB>**, δείκτης
- **<SUP>**, εκθέτης

**<B> <I> ...μορφοποιημένο κείμενο...</B> </I>**

### 2.2.1.8 Η Ετικέτα <HR>

Η ετικέτα <HR> χρησιμοποιείται για την δημιουργία μιας οριζόντιας γραμμής (γραφικό), η οποία χρησιμοποιείται για τον οπτικό διαχωρισμό των ενοτήτων της ιστιοσελίδας. Η ετικέτα <HR> δεν έχει αντίστοιχη ετικέτα τέλους και δεν χρησιμοποιείται σε κείμενο. Ακόμα περιέχει και κάποιους παράμετρους, οι οποίοι μας βοηθάνε να σχεδιάζουμε οριζόντιες γραμμές της αρεσκείας μας. Οι παράμετροι της ετικέτας <HR> είναι οι εξής:

- **SIZE**, ο αριθμός των pixels του πάχους της οθόνης (προκαθορισμένη τιμή είναι το 2)
- **WIDTH**, το πλάτος της γραμμής (προκαθορισμένη τιμή είναι το 100%)
- **ALIGN**, χρησιμοποιείται μαζί με την παράμετρο WIDTH και ορίζει απο που θα αρχίσει γραμμή ( προκαθορισμένη τιμή είναι το CENTER)
- **NOSHADE**, η οριζόντια γραμμή θα σχεδιαστεί σε μαύρο χρώμα

**<HR NOSHADE ALIGN=RIGHT SIZE=6 WIDTH=80%>**

### 2.2.1.9 Η Ετικέτα <BR>

Η Ετικέτα <BR> έχει την λειτουργία που εχεί το πλήκτρο Enter σ' έναν επεξεργαστή κειμένου. Δηλαδή ο φυλλομετρητής όταν συναντήσει την ετικέτα <BR>, ξεκινά το αμέσως επόμενο κείμενο από το αριστερό περιθώριο της επόμενης γραμμής. Η ετικέτα <BR> χρησιμοποιείται χωρίς την αντίστοιχη ετικέτα τέλους της.

### 2.2.1.10 Η Ετικέτα <BGSOUND>

Η Ετικέτα <BGSOUND> χρησιμοποιείται για να προστεθεί στην ιστιοσελίδα ήχος. Περιλαμβάνει και μια παράμετρο την LOOP, η οποία καθορίζει πόσες φορές θα επαναληφθεί ο ήχος. Η σύνταξη της είναι η εξής:

**<BGSOUND SRC="Mozart.mid" LOOP=1>**

### 2.2.2 Στοίχιση Κειμένου

Για να στοιχίσουμε ένα συγκεκριμένο μέρος ενός κειμένου χρησιμοποιούμε την παράμετρο ALIGN, η οποία μπορεί να πάρει τις εξης τρείς τιμές: LEFT, RIGHT και CENTER. Η παράμετρος ALIGN μπορεί να χρησιμοποιηθεί στις ετικέτες αρχής της <H> για επικεφαλίδα και της<P> για παράγραφο.

**<H1 ALIGN=CENTER> ...Επικεφαλίδα... </H1>**
**<P ALIGN=RIGHT> ...Παράγραφος... </P>**

### 2.2.2.1 Η Ετικέτα <DIV>

Η Ετικέτα <DIV> είναι μια πιο ευέλικτη μέθοδος για τη στοίχιση κειμένου. Περιλαμβάνει και αυτή την παράμετρο ALIGN, αλλά αντί να στοιχίζει το κείμενο μιας συγκεκριμένης ετικέτας, στοιχίζει το κείμενο όλων των ετικετών που περικλείονται μεταξύ των ετικετών αρχής και τέλους της.

**<DIV ALIGN=CENTER>**
**<H1> ...Επικεφαλίδα... </H1>**
**<P> ...Παράγραφος... </P>**
**</DIV>**

### 2.2.2.2 Η Ετικέτα <CENTER>

Άλλη μια ευέλικτη μέθοδος για την στοίχιση κειμένου είναι η ετικέτα <CENTER>. Έχει την ίδια λειτουργία με την ετικέτα <DIV>, αλλά χρησιμοποιείται μόνο για κεντράρισμα.

**<CENTER>**
**<H1> ...Επικεφαλίδα... </H1>**
**<P> ...Παράγραφος... </P>**
**</CENTER>**

## 2.2.3 Οι Λίστες (Lists)

Η HTML υποστηρίζει τα εξής πέντε είδη λιστών:

- **Αριθμημένες λίστες**, οι οποίες χρησιμοποιούν αριθμούς για τα στοιχεία τους.
- **Λίστες κουκκίδων**, οι οποίες χρησιμοποιούν μια κουκκίδα για κάθε στοιχείο.
- **Λίστες γλωσσαριού**, οι οποίες για κάθε στοιχείο χρησιμοποιούν έναν όρο και έναν ορισμό.
- **Λίστες σε μορφή μένου**, οι οποίες χρησιμοποιούνται για ειδικές διατάξεις.
- **Λίστες καταλόγου**, οι οποίες χρησιμοποιούνται για την παρουσίαση σύντομων στοιχείων.

### 2.2.3.1 Οι Αριθμημένες Λίστες

Στις αριθμημένες λίστες τα στοιχεία τους είναι αριθμημένα. Περικλείονται μεταξύ της ετικέτας αρχής <OL> και της αντίστοιχης ετικέτας τέλους </OL>, ενώ κάθε στοιχείο τους ξεκινά με την ετικέτα <LI>. Δεν υπάρχει αντίστοιχη ετικέτα τέλους </LI>.

<OL>
<LI> Στοιχείο 1
<LI> Στοιχείο 2
<LI> Στοιχείο 3
</OL>

### 2.2.3.2 Οι Μη Αριθμημένες Λίστες

Στις μη αριθμημένες λίστες τα στοιχεία τους εμφανίζονται με οποιαδήποτε σειρά. Περικλείονται μεταξύ της ετικέτας-αρχής <UL> και της ετικέτας τέλους </UL>, ενώ κάθε στοιχείο τους ξεκινά με την ετικέτα <LI>, όπως και των αριθμημένων. Συνήθως οι λίστες αυτές μορφοποιούνται με κουκκίδες ή κάποιο άλλο σύμβολο.

<UL>
<LI> Στοιχείο 1
<LI> Στοιχείο 2
<LI> Στοιχείο 3
</UL>

### 2.2.4 Οι Σύνδεσμοι (Links)

Για τη δημιουργία ενός συνδέσμου στην γλώσσα HTML χρειαζόμαστε τα εξής δύο στοιχεία:

- Την διαδρομή του αρχείου στον τοπικό δίσκο ή το URL, που θέλουμε να δημιουργήσουμε ως συνδέσμο.
- Το κείμενο το οποίο θα λειτουργήσει σαν σύνδεσμος και κάνοντας κλικ πάνω του θα μας οδηγεί στο συγκεκριμένο αρχείο ή URL.

Για να κατασκευάσουμε ένα σύνδεσμο σε μια ιστοσελίδα, χρησιμοποιούμε την ετικέτα <A> μαζί με τις παραμέτρους NAME, HREF και TITLE, οι οποίες περιγράφουν τον συνδέσμο. Στην παράμετρο HREF καθορίζεται η διαδρομή του αρχείου στον τοπικό

δίσκο ή το URL,που θα δείχνει ο σύνδεσμος, ενώ η χρήση των παραμέτρων NAME και TITLE είναι προαιρετική και μπορεί να παραληφθεί. Ακόμα το κείμενο που περικλείεται μεταξύ των ετικετών τέλους και αρχής <A> και </A>, θα είναι ο σύνδεσμος στην ιστιοσελίδα. Η μορφή ενός συνδέσμου είναι η εξής:

**<A NAME="Up" HREF="…/menu.html" TITLE="Care"> Main Menu </A>**

## 2.2.5 Εισαγωγή Εικόνας

Η ετικέτα για την εισαγωγή εικόνας σε μια ιστιοσελίδα είναι η <IMG>, η οποία δεν έχει αντίστοιχη ετικέτα τέλους, αλλά έχει πάρα πολλές παραμέτρους. Η πιο βασική παράμετρος της είναι η SRC (source), η οποία καθορίζει την διαδρομή στο τοπικό δίσκο του αρχείου της εικόνας, που θέλουμε να εισάγουμε στην ιστιοσελίδα. Η HTML υποστηρίζει αρχεία εικόνων σε μορφή GIF και JPEG.

**<IMG SRC="…/image.JPEG">**

## 2.2.6 Καθορισμός Χρώματος Φόντου

Ο καθορισμός του χρώματος Φόντου της ιστοσελίδας, γίνεται με την προσθήκη της παραμέτρου BGCOLOR στην ετικέτα <BODY>. Για να ορίσουμε χρώμα στην παράμετρο BGCOLOR χρησιμοποιούμε το 16δικό κωδικό του χρώματος, τον οποίο μπορούμε να τα βρούμε από διάφορα ειδικά προγράμματα ζωγραφικής ή εναλλακτικά να χρησιμοποιήσουμε την λέξη του χρώματος.

**<BODY BGCOLOR="#FFFFFF">**

**<BODY BGCOLOR=red>**

## 2.2.7 Καθορισμός Χρώματος Κειμένου

Ο καθορισμός του χρώματος του κειμένου και των συνδέσμων σε μια ιστοσελίδα, γίνεται με την προσθήκη κάποιων ειδικών παραμέτρων στην ετικέτα <BODY>. Οι παράμετροι αυτοί είναι οι εξής:

- **TEXT**, η οποία καθορίζει το χρώμα ολόκληρου του κειμένου της ιστοσελίδας.
- **LINK**, η οποία καθορίζει το χρώμα των συνδέσμων που δεν έχουν επισκεφθεί ακόμα από το χρήστη, η προκαθορισμένη τιμή είναι το μπλε.
- **VLINK**, η οποία καθορίζει το χρώμα των συνδέσμων που έχουν επισκεφθεί από χρήστη, η προκαθορισμένη τιμή είναι το μόβ.
- **ALINK**, η οποία καθορίζει το χρώμα του συνδέσμου όταν τον κλικάρει ο χρήστης, η προκαθορισμένη τιμή είναι το κόκκινο.

## 2.2.8 Κινούμενο Κείμενο <MARGUEE>

Για να δημιουργήσουμε κινούμενο κείμενο με την HTML, χρησιμοποιούμε την ετικέτα <MARGUEE>. Περικλείουμαι το κείμενο που θέλουμε μεταξύ των ετικετών αρχής και τέλους της <MARGUEE> και αυτό εμφανίζεται στον φυλλομετρητή μας κοινούμενο.

**<MARGUEE> ... κινούμενο κείμενο... </MARGUEE>**

# 2.3 Η Τεχνολογία ASP

Το ιδιαίτερο χαρακτηριστικό της γλώσσας προγραμματισμού **ASP (Active Server Pages)** είναι ότι ο κώδικας ενός αρχείου, που είναι γραμμένο με τη γλώσσα αυτή πρώτα μεταγλωτίζεται στον Web Server και ύστερα φορτώνεται σαν ένα κανονικό HTML αρχείο, χωρίς να έχει την δυνατότητα ο χρήστης να δει τον αρχικό κώδικα. Η τεχνολογία της γλώσσας ASP χρησιμοποιείται για να δημιουργήσουμε δυναμικές ιστιοσελίδες (Dynamic Web Pages).

Το αρχείο που είναι γραμμένο στη γλώσσα ASP είναι ακριβώς το ίδιο με ένα αρχείο που είναι γραμμένα στη γλώσσα HTML. Περιέχει κείμενο, HTML Tags και Scripts, τα οποία εκτελούνται στον Web Server και η κατάληξη του είναι .asp. Τα βασικά πλεονεκτήματα της τεχνολογίας ASP είναι τα εξής:

- Τροποποιεί δυναμικά το περιεχόμενο της ιστιοσελίδας,
- έχει πρόσβαση σε δεδομένα ή βάσεις δεδομένων,
- κάνει την ιστιοσελίδα πιο προσιτή στους χρήστες,
- παρέχει ασφάλεια, αφού ο κώδικας της δεν εμφανίζεται,
- και τέλος,ελαχιστοποιεί την κυκλοφορία στο διαδίκτυο (Network Traffic).

## 2.3.1 Η Σύνταξη της ASP

Το αρχείο που είναι γραμμένο στη γλώσσα προγραμματισμού ASP περιέχει Tags της HTML και Server Scripts, τα οποία περικλείονται ανάμεσα στις ετικέτες <% και %>. Τα Server Scripts περιέχουν εντολές, τελεστές, διαδικασίες και γενικά οποιεσδήποτε εκφράσεις που είναι έγκυρες για την γλώσσα ASP.

Παρακάτω ακολουθεί ένα απλό παράδειγμα, το οποίο επιστρέφει στον φυλλομετρητή μας το κείμενο Hello World, μέσω της μεθόδου write του αντικειμένου response της ASP:

```
<HTML>
    <BODY>
        <%
        Response.write"Hello World"
        %>
    </BODY>
</HTML>
```

## 2.3.2 Γλώσσες Συγγραφής (Scripting Languages)

Στην γλώσσα προγραμματισμού ASP έχουμε τη δυνατότητα να χρησιμοποιήσουμε διάφορες γλώσσες συγγραφής. Η προκαθορισμένη γλώσσας συγγραφής που χρησιμοποιείται είναι η VBScript. Άλλες γλώσσες συγγραφής που μπορούμε να χρησιμοποιήσουμε είναι η Javascript, η οποία ξεχωρίζει τα πεζά από τα κεφαλαία γράμματα δηλαδή είναι η γλώσσα case sensitive και θα πρέπει να προσέχουμε πώς γράφουμε τον κώδικα μας όταν την χρησιμοποιούμε, η Perl, η REXX και η Python που για να τις χρησιμοποιήσουμε χρειάζεται να εγκαταστήσουμε τις αντίστοιχες μηχανές συγγραφής (Scripting Engines).

Για να καθορίσουμε ποια γλώσσα συγγραφής θέλουμε να χρησιμοποιήσουμε, προσθέτουμε στην αρχή του κώδικα μας την εξής εντολή:

```
<%@ language="javascript"%>
```

### 2.3.3 Το Μοντέλο Αντικειμένων της ASP

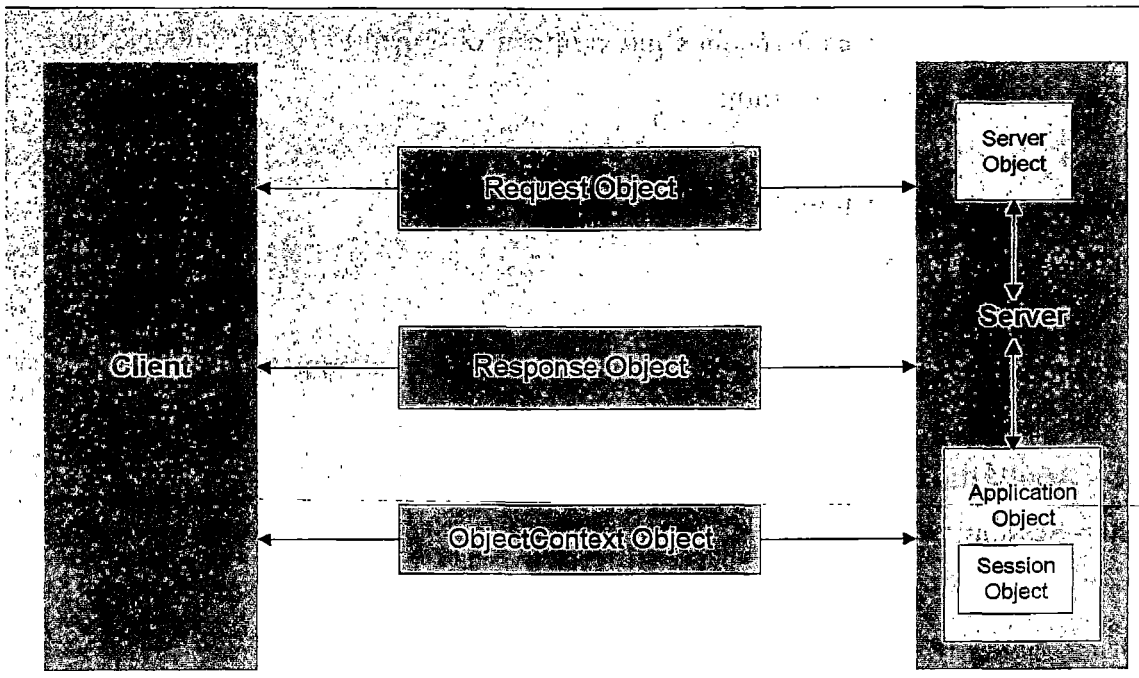
Η ASP, όπως οι περισσότερες τεχνολογίες της Microsoft, χρησιμοποιεί το Συστατικό Μοντέλο Αντικειμένων (Component Object Model – COM), για να εκθέσει λειτουργικότητα σε καταναλωτικές εφαρμογές. Η ASP είναι στην πραγματικότητα μία επέκταση του web server μας που επιτρέπει το scripting στον server. Ταυτόχρονα παρέχει ένα σύνολο από αντικείμενα και συστατικά, που χειρίζονται την αλληλεπίδραση ανάμεσα στον web server και τον browser. Τα αντικείμενα αυτά αποτελούν το Active Server Pages Object Model και τα χειρίζονται scripting γλώσσες.

Αυτό το μοντέλο αντικειμένων αποτελείται από έξι κυρία αντικείμενα, καθένα με διακριτές ιδιότητες και μεθόδους. Τα αντικείμενα αυτά είναι τα εξής:

- Request (Αίτηση)
- Response (Ανταπόκριση)
- Application (Εφαρμογή)
- Session (Σύνοψη)
- Server (Εξυπηρετητής)
- ObjectContext (Γενικό πλαίσιο αντικειμένου)

καθένα από τα αντικείμενα, εκτός των αντικειμένων Server και ObjectContext, μπορεί να χρησιμοποιεί συλλογές (Collections) για να αποθηκεύει δεδομένα.

Στο παρακάτω σχήμα φαίνεται η διαίρεση της ASP σε έξι διαφορετικά αντικείμενα, τα οποία χειρίζονται το δικό τους τμήμα αλληλεπίδρασης μεταξύ του Client και του Server. Στην καρδιά της αλληλεπίδρασης ανάμεσα στον Client και στον Server είναι τα αντικείμενα Request και Response, που σχετίζονται με την HTTP αίτηση και ανταπόκριση.




**2.4: Το μοντέλο των αντικειμένων της ASP**

## 2.3.4 Οι Συλλογές (Collections) της ASP

Οι συλλογές στην ASP μοιάζουν πολύ με τις συνονόματές τους στην VBScript. Λειτουργούν ως αποθηκευτικοί χώροι δεδομένων που αποθηκεύουν τα δεδομένα τους με ένα τρόπο παρόμοιο με αυτόν των πινάκων. Η πληροφορία αποθηκεύεται σε ζεύγη της μορφής (όνομα, τιμή).

Τα αντικείμενα Application και Session έχουν μία ιδιότητα συλλογής με το όνομα Contents. Αυτή η συλλογή παραλλαγών μπορεί να συγκρατήσει οποιαδήποτε πληροφορία θέλουμε να τοποθετήσουμε σ' αυτήν. Χρησιμοποιώντας τις συλλογές αυτές μας δίνεται η δυνατότητα να μοιράζουμε πληροφορίες μεταξύ ιστοσελίδων.

Παρακάτω ακολουθεί ένα παράδειγμα συλλόγων, για να γίνει κατανοητή η λειτουργία τους. Για να τοποθετήσουμε μια τιμή στη συλλογή, απλά της καθορίζουμε ένα κλειδί και μετα θέτουμε την τιμή:

**Application("Name") = "John"**
**Session("Height") = 1.80**

Για να ανακτήσουμε τιμές από τις συλλογές, αντιστρέφουμε τη κλήση:

**sName = Application("Name")**
**sHeight = Session("Height")**

# 2.4 Η Γλώσσα Διαχείρισης Βάσεων Δεδομένων SQL

Για να μπορέσουμε να δημιουργήσουμε και να διαχειριστούμε μια βάση δεδομένων, μπορούμε να χρησιμοποιήσουμε ειδικές γλώσσες προγραμματισμού, τις λεγόμενες γλώσσες ερωταπαντήσεων (query languages). Είναι γλώσσες μη διαδικαστικές, τέταρτης γενιάς ($4^{th}$ generation languages). Εμείς απλά διατυπώνουμε με απλές και κατανοητές εντολές το τι πληροφορίες ζητάμε και το ΣΔΒΔ (Σύστημα Διαχείρισης Βάσεων Δεδομένων) αναλαμβάνει να μας απαντήσει. Η SQL (Structured Query Language) είναι σήμερα η πιο δημοφιλής και πιο διαδεδομένη γλώσσα ανάπτυξης και διαχείρισης σχεσιακών βάσεων δεδομένων.

Η SQL αποτελείται από εντολές με τα ορίσματά τους, τις οποίες μπορούμε να χρησιμοποιήσουμε με συγκεκριμένους κανόνες σύνταξης για να πάρουμε τα αποτελέσματα που θέλουμε. Με την SQL μπορούμε να δημιουργήσουμε μια βάση δεδομένων και τους πίνακές της με τα αντίστοιχα πεδία, να καταχωρήσουμε δεδομένα στους πίνακες, να τροποποιήσουμε και να διαγράψουμε τα δεδομένα αυτά, να αλλάξουμε τη δομή των πινάκων με προσθήκη και διαγραφή πεδίων και να εμφανίσουμε πληροφορίες (συνδυασμούς από δεδομένα).

Η γλώσσα SQL χωρίζεται σε δύο τμήματα :

- Τη γλώσσα ορισμού δεδομένων (Data Definition Language – DDL), η οποία περιέχει τις απαραίτητες εντολές για τον ορισμό και την τροποποίηση του σχεσιακού σχήματος καθώς και για τη δημιουργία, την τροποποίηση και τη διαγραφή σχέσεων. Περιέχει ακόμη τις εντολές δημιουργίας και επεξεργασίας όψεων και ορισμού περιορισμών ακεραιότητας.
- Τη γλώσσα χειρισμού δεδομένων (Data Manipulation Language - DML), η οποία περιέχει τις απαραίτητες εντολές για την εμφάνιση (αναζήτηση) δεδομένων καθώς και για την καταχώρηση, τροποποίηση και διαγραφή των εγγραφών (πλειάδων) μιας σχέσης.

### 2.4.1 Οι Τύποι Δεδομένων της SQL

Η SQL υποστηρίζει τους εξής τύπους δεδομένων για τα πεδία μιας σχέσης:

- **char(n)**, ένα αλφαριθμητικό (string) με n ακριβώς χαρακτήρες.
- **varchar(n)**, ένα αλφαριθμητικό (string) με μεταβλητό μήκος και με n το πολύ χαρακτήρες.
- **int**, ακέραιος αριθμός.
- **smallint**, ακέραιος αριθμός με μικρές τιμές.
- **Numeric(p,d)**, αριθμός με p ψηφία, από τα οποία τα d είναι δεκαδικά.
- **real**, αριθμός κινητής υποδιαστολής απλής ακρίβειας.
- **double precision**, αριθμός κινητής υποδιαστολής διπλής ακρίβειας.
- **float(n)**, αριθμός κινητής υποδιαστολής με ακρίβεια n ψηφίων.
- **date**, ημερομηνία (ημέρα, μήνας, έτος).
- **time**, ώρα (ώρα, λεπτά, δευτερόλεπτα).

### 2.4.2 Δημιουργία, Τροποποίηση και Διαγραφή Σχέσεων (Πινάκων)

Για να δημιουργήσουμε μια σχέση (πίνακα) χρησιμοποιούμε την εντολή create table. Για παράδειγμα, για να δημιουργήσουμε τη σχέση ΦΟΙΤΗΤΗΣ δίνουμε τις εξής εντολές :

```
create table ΦΟΙΤΗΤΗΣ
    (Α.Μ_Φοιτητή        intenger not null,
    Επώνυμο             char(20) not null,
    Όνομα               char(15),
    Ημ/νία Εγγραφής     date,
    primary key (Α.Μ_Φοιτητή),
    check (Α.Μ_Φοιτητή>1000));
```

Μετά την εντολή create table γράφουμε το όνομα της σχέσης (πίνακα) και ακολουθεί μια παρένθεση που περιέχει τα ονόματα των πεδίων με τους τύπους δεδομένων τους.

- Η δήλωση **not null** σημαίνει ότι το συγκεκριμένο πεδίο θα πρέπει να έχει οπωσδήποτε κάποια τιμή. Προσοχή : άλλο πράγμα είναι η τιμή 0 ή η τιμή του κενού χαρακτήρα (space), που είναι κάποια τιμή, και άλλο πράγμα είναι η μη ύπαρξη τιμής.
- Με τη δήλωση **primary key** ορίζουμε το πεδίο κλειδί (πρωτεύον κλειδί) της σχέσης. Αν, αργότερα κατά την εισαγωγή τιμών, δώσουμε μια τιμή null στο πρωτεύον κλειδί ή μια τιμή που ήδη υπάρχει σε κάποια άλλη πλειάδα, τότε θα εμφανισθεί ένα μήνυμα λάθους.
- Με τον όρο **check** μπορούμε να δηλώσουμε μια συνθήκη για την περιοχή των τιμών ενός πεδίου. Είδαμε τη χρήση του όρου check για να ελέγξει αν η τιμή ενός κωδικού είναι μεγαλύτερη από 1000.

Για να τροποποιήσουμε μια σχέση και να προσθέσουμε ένα πεδίο, δίνουμε την εντολή **alter table** και το όρισμα **add**, ως εξής :

**alter table ΦΟΙΤΗΤΗΣ add (Τηλέφωνο char(10));**

Τα καινούργια πεδία που προστίθενται σε μια σχέση, έχουν αρχικά τιμές ίσες με null. Για να τροποποιήσουμε μια σχέση και να διαγράψουμε ένα πεδίο, δίνουμε την εντολή **alter table** και το όρισμα **drop**, ως εξής :

**alter table ΦΟΙΤΗΤΗΣ drop Τηλέφωνο;**

Για να διαγράψουμε τελείως μια σχέση από τη βάση δεδομένων στην οποία ανήκει, μαζί με τις τιμές των εγγραφών της, δίνουμε την εντολή **drop table**, ως εξής :

**drop table ΦΟΙΤΗΤΗΣ;**

Υπάρχει και η εντολή **delete from ΦΟΙΤΗΤΗΣ**, με την οποία μπορούμε να διαγράψουμε τις τιμές (περιεχόμενα, εγγραφές) μιας σχέσης, αλλά όχι την ίδια τη σχέση.

### 2.4.3 Ακεραιότητα Αναφορών (Referential Integrity)

Με τον όρο ακεραιότητα αναφορών (referential integrity) αναφερόμαστε στην ιδιότητα όπου οι τιμές ορισμέΘέση του θα χρόνων νων πεδίων μιας σχέσης υπάρχουν και σε αντίστοιχα πεδία σε κάποια άλλη σχέση. Για παράδειγμα, αν έχουμε τις σχέσεις **ΦΟΙΤΗΤΗΣ** και ΜΑΘΗΜΑ, τότε η συσχέτιση μεταξύ τους υλοποιείται με την τρίτη σχέση ΕΓΓΡΑΦΗ, όπου εμφανίζεται το ποιοι φοιτητές παρακολουθούν το συγκεκριμένο μάθημα και φυσικά τι βαθμολογία τους σ' αυτό.

Η ακεραιότητα αναφορών έρχεται να επιλύσει το πρόβλημα της διαγραφής μιας πλειάδας από τη σχέση ΜΑΘΗΜΑ, οπότε οι φοιτητές που παρακολουθούσαν το συγκεκριμένο μάθημα θα φαίνονται ξεκρέμαστοι, καθώς επίσης και το πρόβλημα της τροποποίησης του πεδίου κλειδιού (Κωδικός_Μαθήματος) μιας πλειάδας από τη σχέση ΜΑΘΗΜΑ, οπότε και πάλι οι φοιτητές που παρακολουθούσαν το συγκεκριμένο μάθημα θα φαίνονται ξεκρέμαστοι.

Για να αποφύγουμε τέτοιες καταστάσεις, καταφεύγουμε στην έννοια του ξένου κλειδιού (foreign key), όπου στη σχέση ΕΓΓΡΑΦΗ, το πεδίο Κωδικός_Φοιτητή είναι ξένο κλειδί αλλά και πρωτεύον κλειδί στη σχέση ΜΑΘΗΤΗΣ, ενώ το πεδίο Κωδικός_ Μαθήματος είναι ξένο κλειδί στη σχέση ΕΓΓΡΑΦΗ αλλά και πρωτεύον κλειδί στη σχέση ΜΑΘΗΜΑ.

Με τη δήλωση της ακεραιότητας αναφορών, αν διαγράψουμε μια πλειάδα ενός πεδίου που είναι πρωτεύον κλειδί σε κάποια σχέση, τότε θα διαγραφούν και όλες οι αντίστοιχες εγγραφές (πλειάδες) στη σχέση όπου το ίδιο πεδίο είναι ξένο κλειδί. Αυτό σημαίνει πρακτικά ότι αν διαγράψουμε έναν φοιτητή από τη σχέση **ΦΟΙΤΗΤΗΣ**, τότε θα διαγραφούν και όλες οι εγγραφές του σε μαθήματα από τη σχέση ΕΓΓΡΑΦΗ. Επίσης, αν τροποποιήσουμε την τιμή ενός πεδίου που είναι πρωτεύον κλειδί σε κάποια σχέση, τότε θα ενημερωθούν αυτόματα και όλες οι αντίστοιχες εγγραφές (πλειάδες) στη σχέση όπου το ίδιο πεδίο είναι ξένο κλειδί. Αυτό σημαίνει πρακτικά ότι αν τροποποιήσουμε τον κωδικό ενός μαθήματος από τη σχέση ΜΑΘΗΜΑ, τότε θα ενημερωθούν αυτόματα και όλες οι εγγραφές των φοιτητών στο μάθημα αυτό που υπάρχουν στη σχέση ΕΓΓΡΑΦΗ.

Για να δηλώσουμε ένα ξένο κλειδί και την αναφορά του, χρησιμοποιούμε τις εντολές **foreign key** και **references** όταν δημιουργούμε μια σχέση, ως εξής :

```
create table ΕΓΓΡΑΦΗ
        (Κωδικός_Φοτητή          integer not null,
        Κωδικός_Μαθήματος        integer not null,
        Βαθμολογία               float,
        primary key(Κωδικός_Φοιτητή, Κωδικός_Μαθήματος)
        foreign key(Κωδικός_Φοιτητή) references ΦΟΙΤΗΤΗΣ,
        foreign key(Κωδικός_Μαθήματος) references ΜΑΘΗΜΑ);
```

Σύμφωνα με τον παραπάνω τρόπο δημιουργίας της σχέσης **ΕΓΓΡΑΦΗ**, αν επιχειρήσουμε να διαγράψουμε έναν φοιτητή που έχει παρακολουθήσει κάποιο μάθημα ή ένα μάθημα που έχουν παρακολουθήσει κάποιοι φοιτητές, το σύστημα δεν θα μας αφήσει να το κάνουμε γιατί έχουμε παραβίαση της ακεραιότητας αναφοράς. Το ίδιο πρόβλημα θα παρουσιασθεί και αν προσπαθήσουμε να τροποποιήσουμε τον κωδικό ενός φοιτητή που έχει παρακολουθήσει κάποιο μάθημα ή τον κωδικό ενός μαθήματος το οποίο έχουν παρακολουθήσει κάποιοι φοιτητές.

Για να μπορέσουμε να αντιμετωπίσουμε την ανάγκη διαγραφής μιας πλειάδας ή τροποποίησης του πεδίου κλειδιού χωρίς να έχουμε παραβίαση της ακεραιότητας αναφοράς, χρησιμοποιούμε τις εντολές **on delete cascade** και **on update cascade** αντίστοιχα, ως εξής :

```
create table ΕΓΓΡΑΦΗ
        (Κωδικός_Φοτητή          integer not null,
        Κωδικός_Μαθήματος        integer not null,
        Βαθμολογία               float,
        primary key(Κωδικός_Φοιτητή, Κωδικός_Μαθήματος)
        foreign key(Κωδικός_Φοιτητή) references ΦΟΙΤΗΤΗΣ
                on delete cascade
                on update cascade,
        foreign key(Κωδικός_Μαθήματος) references ΜΑΘΗΜΑ
                on delete cascade
                on update cascade);
```

## 2.4.4 Οι Όψεις (Views)

Η **όψη (view)** είναι μια σχέση μιας βάσης δεδομένων που δεν έχει δημιουργηθεί με κάποια εντολή create table. Με τις όψεις μπορούμε να εμφανίζουμε ορισμένα μόνο πεδία μιας σχέσης ή και κάποιες άλλες πληροφορίες που αν δεν ήταν οι όψεις θα έπρεπε να δίνουμε κάθε φορά πολύπλοκες εντολές για να δούμε τις πληροφορίες που θέλουμε. Μπορούμε να δημιουργούμε, να τροποποιούμε και να διαγράφουμε όσες όψεις θέλουμε, χωρίς να επηρεάζονται καθόλου τα δεδομένα των σχέσεων στις οποίες αναφέρονται οι όψεις.

Για να δημιουργήσουμε μια όψη, δίνουμε την εντολή **create view as**, ως εξής :

**create view ΦΟΙΤΗΤΗΣ _1 as**
**(Select Α.Μ_Φοιτητή, Επώνυμο, Ημ/νία Εγγραφής**
**From ΦΟΙΤΗΤΗΣ );**

Μετά την εντολή as γράφουμε μέσα σε παρένθεση την εντολή SQL της οποίας το αποτέλεσμα θα δημιουργήσει την όψη. Η παραπάνω όψη εμφανίζει λίγα μόνο από τα πεδία της σχέσης ΑΘΛΗΤΗΣ. Η επόμενη όψη εμφανίζει την μέση τιμή των βαθμολογιών (μ.ο) των φοιτητών :

**Create view ΦΟΙΤΗΤΗΣ _2 as**
**(Select avg(μ.ο)**
**From ΦΟΙΤΗΤΗΣ);**

Για να διαγράψουμε μια όψη, δίνουμε την εντολή **drop view**, ως εξής :

**Drop view ΦΟΙΤΗΤΗΣ _1;**

## 2.4.5 Οι Συναρτήσεις Ομαδοποίησης

Η SQL χρησιμοποιεί μερικές πολύ χρήσιμες συναρτήσεις, που ονομάζονται συναρτήσεις ομαδοποίησης (aggregate functions) και οι οποίες δέχονται ένα σύνολο τιμών και επιστρέφουν μία τιμή. Οι συναρτήσεις αυτές είναι οι εξής :

| **Συνάρτηση Ομαδοποίησης** | **Αντίστοιχος Όρος στην SQL** |
|---|---|
| Απαρίθμηση | Count |
| Άθροισμα | Sum |
| Μέσος Όρος | Avg |
| Μέγιστη Τιμή | Max |
| Ελάχιστη Τιμή | Min |

Οι συναρτήσεις Sum και Avg εργάζονται μόνο με αριθμητικές τιμές, ενώ οι υπόλοιπες συναρτήσεις μπορούν να δεχθούν και αλφαριθμητικές τιμές. Για να βρούμε τον συνολικό αριθμό των φοιτητών, δίνουμε την εξής εντολή :

**Select count(*)**
**From ΦΟΙΤΗΤΗΣ;**

Το αποτέλεσμα θα είναι ένας μόνο αριθμός, δηλ. ο συνολικός αριθμός των φοιτητών (εγγραφών) της σχέσης ΦΟΙΤΗΤΗΣ. Για να βρούμε τον φοιτητή που έχει την καλύτερη βαθμολογία, δίνουμε την εξής εντολή :

**Select Α.Μ_Φοιτητή, Επώνυμο, Όνομα, max(μ.ο)**
**From ΦΟΙΤΗΤΗΣ;**

Εδώ η συνάρτηση max() δέχεται σαν είσοδο όλες τις αριθμητικές τιμές του πεδίου μ.ο και επιστρέφει την μέγιστη τιμή μαζί με τα στοιχεία του φοιτητή που έχει την

καλύτερη βαθμολογία. Για να βρούμε τον αθλητή που έχει την χαμηλότερη βαθμολογία, δίνουμε την εξής εντολή :

**Select A.M_Φοιτητή, Επώνυμο, Όνομα, min(μ.ο)**
**From ΦΟΙΤΗΤΗΣ;**

Εδώ η συνάρτηση min() δέχεται σαν είσοδο όλες τις αριθμητικές τιμές του πεδίου μ.ο και επιστρέφει την ελάχιστη τιμή μαζί με τα στοιχεία του φοιτητή που έχει τη χαμηλότερη βαθμολογία. Για να βρούμε τον μέσο όρο των βαθμολογιών των φοιτητών, δίνουμε την εξής εντολή :

**Select avg(μ.ο)**
**From ΦΟΙΤΗΤΗΣ;**

Εδώ η συνάρτηση avg() δέχεται σαν είσοδο όλες τις αριθμητικές τιμές του πεδίου μ.ο και επιστρέφει την μέση τιμή των των βαθμολογιών όλων των φοιτητών. Και εδώ το αποτέλεσμα θα είναι ένας μόνο αριθμός.

## 2.4.6 Οι Βασικότερες Εντολές της SQL

Παρακάτω ακολουθούν οι βασικότερες εντολές της SQL (Structured Query Language), καθώς και επεξήγηση τους:

- **Select**, χρησιμοποιείται για την ανάκληση δεδομένων απ' τη βάση δεδομένων.
- **As**, με τον όρο as μπορούμε να μετονομάσουμε πεδία ή σχέσεις αλλά μόνο προσωρινά για τις ανάγκες μιας εντολής Select και όχι μόνιμα.
- **Order By**, με τον όρο order by μπορούμε να ταξινομήσουμε το αποτέλεσμα μιας εντολής Select ως προς κάποιο πεδίο.
- **Group By**, με τον όρο group by μπορούμε να απομονώσουμε τις εγγραφές μιας σχέσης σε ανεξάρτητα υποσύνολα και μετά να εφαρμόσουμε μια συνάρτηση ομαδοποίησης σε κάθε υποσύνολο.

- **Intersect**, Επειδή οι σχέσεις αντιστοιχούν σε σύνολα, μπορούμε να χρησιμοποιήσουμε τις πράξεις μεταξύ συνόλων και στις σχέσεις. Η πράξη της τομής στη σχεσιακή άλγεβρα επιτυγχάνεται στην SQL με τον όρο Intersect. Για να μπορέσουμε να εφαρμόσουμε την πράξη της τομής, όπως και τις πράξεις της ένωσης αλλά και της διαφοράς, ανάμεσα σε δύο σχέσεις, θα πρέπει αυτές να είναι συμβατές, δηλ. να έχουν τα ίδια πεδία ή πεδία με το ίδιο πεδίο ορισμού.
- **Union**, επιτυγχάνεται η πράξη της ένωσης στη σχεσιακή άλγεβρα.
- **Except**, επιτυγχάνεται η πράξη της διαφοράς στη σχεσιακή άλγεβρα.
- **In**, ελέγχουμέ αν μια πλειάδα ανήκει σε μια σχέση η οποία δημιουργείται από μια φωλιασμένη εντολή.
- **Not In**, ελέγχουμε αν μια πλειάδα δεν ανήκει σε μια σχέση η οποία δημιουργείται από μια φωλιασμένη εντολή.
- **Some**, χρησιμοποιείται για να απαντήσουμε σε ερωτήματα όπου η τιμή ενός πεδίου θέλουμε να συγκριθεί με την τιμή ενός πεδίου μίας τουλάχιστον πλειάδας μιας σχέσης που προκύπτει από μια φωλιασμένη εντολή.
- **All**, χρησιμοποιείται για να απαντήσουμε σε ερωτήματα όπου η τιμή ενός πεδίου θέλουμε να συγκριθεί με την τιμή ενός πεδίου όλων των πλειάδων μιας σχέσης που προκύπτει από μια φωλιασμένη εντολή.
- **Insert Into**, χρησιμοποιείται για να καταχωρήσουμε πλειάδες σε μια ήδη υπάρχουσα σχέση.
- **Delete From**, χρησιμοποιείται για να διαγράψουμε ολόκληρες πλειάδες και όχι μεμονωμένα πεδία (στήλες).
- **Update**, χρησιμοποιείται για να τροποποιήσουμε την τιμή κάποιων πεδίων από ορισμένες ή και απ' όλες τις πλειάδες μιας σχέσης.

# 2.5 Ασφάλεια Πληροφοριακών Συστημάτων

Για να είναι αποδοτικότερα τα πληροφοριακά συστήματα που είναι βασισμένα στις ηλεκτρονικές πληρωμές, πρέπει να παρέχουν υπηρεσίες ασφάλειας, όπως ακεραιότητα των δεδομένων που συναλλάσσονται, πιστοποίηση μεταξύ των συναλλασσόμενων, κ.α. Τα πληροφοριακά συστήματα ηλεκτρονικών πληρωμών παρέχουν υπηρεσίες ασφάλειας, με την χρήση κρυπτογραφικών τεχνικών και ψηφιακών πιστοποιητικών.

Παρακάτω αναλύονται ειδικότερα τα **κρυπτοσυστήματα** και τα **ψηφιακά πιστοποιητικά**.

## 2.5.1 Κρυπτοσυστήματα

Το κρυπτοσύστημα βασίζεται σ' έναν αλγόριθμο κρυπτογράφησης και στα κρυπτογραφικά κλειδιά που χρησιμοποιεί. Μπορεί να οριστεί ως ένα ζεύγος μετασχηματισμών, την κρυπτογράφηση (encrypt) και την αποκρυπτογράφηση (decrypt). Η κρυπτογράφηση είναι μια διαδικασία που εφαρμόζεται στα δεδομένα (plaintext) και τα μετασχηματίζει σε ακατανόητα δεδομένα (ciphertext). Η αποκρυπτογράφηση εφαρμόζεται στα ακατανόητα δεδομένα (ciphertext) και τα μετασχηματίζει στα αρχικά δεδομένα.

Ο μετασχηματισμός της κρυπτογράφησης δέχεται ως είσοδο τα δεδομένα (plaintext) και το κλειδί κρυπτογράφησης. Παρόμοια ο μετασχηματισμός της αποκρυπτογράφησης δέχεται ως είσοδο τα κρυπτογραφημένα δεδομένα (ciphertext) και το κλειδί αποκρυπτογράφησης. Σε κάθε περίπτωση το κλειδί είναι μία τυχαία ακολουθία από bits, το πλήθος των οποίων καθορίζει το συγκεκριμένο κρυπτοσύστημα.

Τα κρυπτοσυστήματα, κατά την συναλλαγή τους, παρέχουν τις εξής υπηρεσίες:

- μυστικότητα των δεδομένων (data confidentiality),
- ακεραιότητα των δεδομένων (data integrity),
- πιστοποίηση χρηστών (user authentication) και
- αδυναμία απάρνησης.

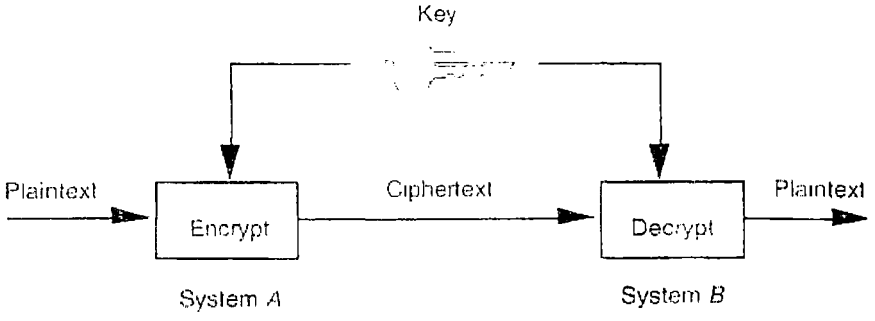
Υπάρχουν δύο κατηγορίες αλγορίθμων κρυπτογράφησης. Οι συμμετρικοί ή ιδιωτικού κλειδιού αλγόριθμοι και οι αλγόριθμοι δημοσίου κλειδιού. Κάθε μία κατηγορία έχει συγκεκριμένα πλεονεκτήματα και χρησιμοποιείται με διαφορετικό τρόπο.

### 2.5.1.1 Κρυπτοσυστήματα συμμετρικού κλειδιού

Τα συμμετρικά κρυπτοσυστήματα χρησιμοποιούνται σε εμπορικές εφαρμογές από τη δεκαετία του 70 και έχουν το χαρακτηριστικό, ότι το ίδιο κλειδί χρησιμοποιείται για την κρυπτογράφηση και την αποκρυπτογράφηση των δεδομένων. Το βασικό πλεονέκτημα των κρυπτοσυστημάτων είναι η ταχύτητα τους, σε σχέση με τα κρυπτοσυστήματα δημόσιου κλειδιού. Ένα κρυπτοσύστημα συμμετρικού κλειδιού λειτουργεί με τον εξής τρόπο:

- Δύο συστήματα Α και Β πρέπει να επικοινωνήσουν με ασφάλεια
- Με κάποιο τρόπο και οι δύο αποκτούν το μυστικό κλειδί, το οποίο είναι άγνωστο σε όλους εκτός από τα Α και Β.
- Χρησιμοποιώντας το συμμετρικό κλειδί μπορούν να κρυπτογραφήσουν και να αποκρυπτογραφήσουν το μήνυμα μόνο αυτοί.

Η διαδικασία κρυπτογράφησης συμμετρικού κλειδιού, παρουσιάζεται στο παρακάτω σχήμα




**2.5: Κρυπτογράφηση συμμετρικού κλειδιού**

Η ασφάλεια του συμμετρικού κρυπτοσυστήματος βασίζεται στο κρυπτογραφικό κλειδί που είναι γνωστό μόνο στα μέρη που επικοινωνούν. Γι' αυτό το λόγο, το κλειδί πρέπει να παραμένει μυστικό και να προστατεύεται από υποκλοπή και μεταβολή.

Ένα συμμετρικό κρυπτοσύστημα λειτουργεί, είτε ως block cipher, είτε ως stream cipher. Σε block cipher η συνάρτηση κρυπτογράφησης λειτουργεί σε σταθερού μήκους block plaintext, n bits και παράγει σταθερού μήκους block ciphertext, το οποίο έχει μήκος n bits. Όμοια λειτουργεί και η συνάρτηση αποκρυπτογράφησης. Από την άλλη μεριά, το stream cipher λειτουργεί σε μία ακολουθία από bits μεταβλητού μήκους, παράγοντας ciphertext του ίδιου μήκους. Στην πραγματικότητα το stream cipher επεξεργάζεται τα δεδομένα σαν ακολουθίες χαρακτήρων, όπου χαρακτήρας μπορεί να θεωρηθεί ένα bit ή ένας μικρός αριθμός bits.

### 2.5.1.1.1 Ο αλγόριθμος Data Encryption Standard (DES)

Το πρώτο συμμετρικό κρυπτοσύστημα που χρησιμοποιήθηκε ευρέως ήταν το Data Encryption Standard (DES). Αρχικά ο αλγόριθμος χρησιμοποιήθηκε για την προστασία κυβερνητικής πληροφορίας και εμπορικές συναλλαγές αλλά αργότερα επεκτάθηκε η χρήση του και σε άλλους τομείς. Ο DES λειτουργεί ως block cipher σε blocks δεδομένων των 64 bits και χρησιμοποιεί κλειδί των 56 bits.

Γενικά ο DES θεωρείται καλό κρυπτοσύστημα διότι δεν έχει βρεθεί άλλη μέθοδος παραβίασης του, πέραν της εξαντλητικής αναζήτησης του κλειδιού, η οποία μέχρι πρόσφατα δεν θεωρούνταν εφικτή λόγω των υπολογιστικών πόρων που απαιτούσε. Παρόλα αυτά η τεχνολογία έχει προχωρήσει αρκετά, σε σημείο που να είναι αμφισβητήσιμη η ασφάλεια που παρέχει ένα οποιοδήποτε block cipher με κλειδί μήκους 56 bits. Για το λόγο αυτό έχουν αναπτυχθεί εναλλακτικές λύσεις για την αντικατάσταση του DES, οι οποίες είναι:

- **Triple-DES.**

  Η αποτελεσματικότητα του DES μπορεί να βελτιωθεί σημαντικά χρησιμοποιώντας τεχνικές πολλαπλής κρυπτογράφησης. Το αρχικό μήνυμα, σε blocks των 64 bits, κρυπτογραφείται χρησιμοποιώντας ένα κλειδί α, στη συνέχεια αποκρυπτογραφείται χρησιμοποιώντας ένα κλειδί β

και το αποτέλεσμα κρυπτογραφείται χρησιμοποιώντας ένα κλειδί γ. Δύο ή τρία κλειδιά χρησιμοποιούνται (μερικές φορές τα κλειδιά α και γ είναι ίδια). Το αποτέλεσμα που προκύπτει είναι πολλές τάξεις μεγέθους ισχυρότερο του DES.

- **SKIPJACK.**

  Είναι ένα κρυπτοσύστημα που προτάθηκε από το υπουργείο Άμυνας των Η.Π.Α. και που παρέχει τη δυνατότητα των οργάνων του νόμου να αποκρυπτογραφούν το μήνυμα όταν πρέπει. Το SKIPJACK είναι ένα 64-bits block cipher το οποίο χρησιμοποιεί ένα κλειδί μήκους 80-bits. Παρόλα αυτά επειδή τα τεχνικά χαρακτηριστικά του είναι απόρρητα δεν είναι πιθανό να χρησιμοποιηθεί για εμπορικούς σκοπούς.

- **Εμπορικοί Αλγόριθμοι.**

  Πολλοί αλγόριθμοι έχουν κατασκευαστεί από εταιρίες και χρησιμοποιούνται σε εμπορικά προϊόντα. Παραδείγματα είναι ο IDEA (International Data Encryption Algorithm) από την Ascom-Tech, οι RC2, RC4 και RC5 από την RSA Data Security

### 2.5.1.2 Κρυπτοσυστήματα Δημοσίου Κλειδιού

Στα κρυπτοσυστήματα δημόσιου κλειδιού τα μέλη που παίρνουν μέρος, χρησιμοποιούν ένα ζεύγος κρυπτογραφικών κλειδιών για την κρυπτογράφηση και την αποκρυπτογράφηση, σε αντίθεση με τα κρυπτοσυστήματα συμμετρικού κλειδιού που χρησιμοποιούν ένα κλειδί. Για την κρυπτογράφηση χρησιμοποιείται το ιδιωτικό κλειδί (private key) , το οποίο είναι γνωστό μόνο στο μέλος που κρυπτογραφεί την πληροφορία. Για την αποκρυπτογράφηση χρησιμοποιείται το δημόσιο κλειδί, το οποίο αποστέλλεται από το μέλος που κρυπτογράφησε την πληροφορία στα μέλη που θέλει να την αποκρυπτογραφήσουν.

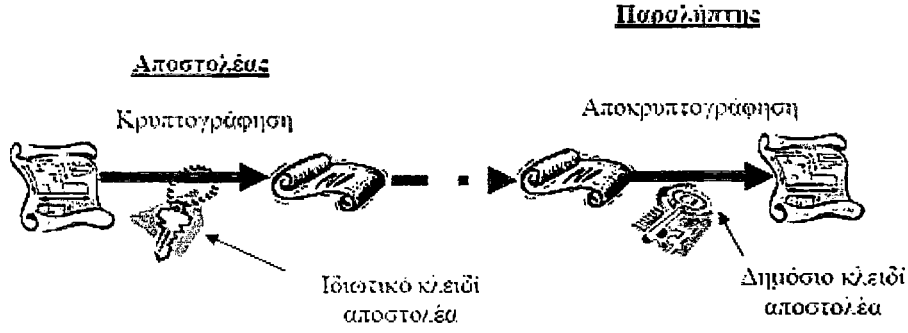
Για την ασφάλεια των κρυπτοσυστημάτων δημόσιου κλειδιού πρέπει να προστατεύεται το ιδιωτικό κλειδί από υποκλοπή ή τροποποίηση και το δημόσιο κλειδί μόνο από τροποποίηση. Το πλεονέκτημα αυτών των κρυπτοσυστημάτων είναι ότι, για την επικοινωνία δύο μελών δεν χρειάζεται το ένα μέλος να γνωρίζει το ιδιωτικό κλειδί

του αλλού, αλλά έχουν το μειονέκτημα να είναι πιο αργά απ' τα κρυπτοσυστήματα συμμετρικού κλειδιού.

Ένα κρυπτοσύστημα δημόσιου κλειδιού μπορεί να λειτουργήσει με τρεις διαφορετικούς τρόπους, οι οποίοι είναι:

**1ος Τρόπος κρυπτογράφησης δημόσιου κλειδιού:**

Το μέλος που κρυπτογραφεί την πληροφορία, ο αποστολέας κρυπτογραφεί την πληροφορία με την χρησιμοποίηση του ιδιωτικού του κλειδιού. Το μέλος στο οποίο αποστέλλεται η κρυπτογραφημένη πληροφορία, ο παραλήπτης έχει παραλάβει με κάποιον τρόπο το δημόσιο κλειδί του αποστολέα και μπορεί να αποκρυπτογραφήσει την κρυπτογραφημένη πληροφορία. Ο τρόπος αυτός, παρουσιάζεται στο παρακάτω σχήμα:

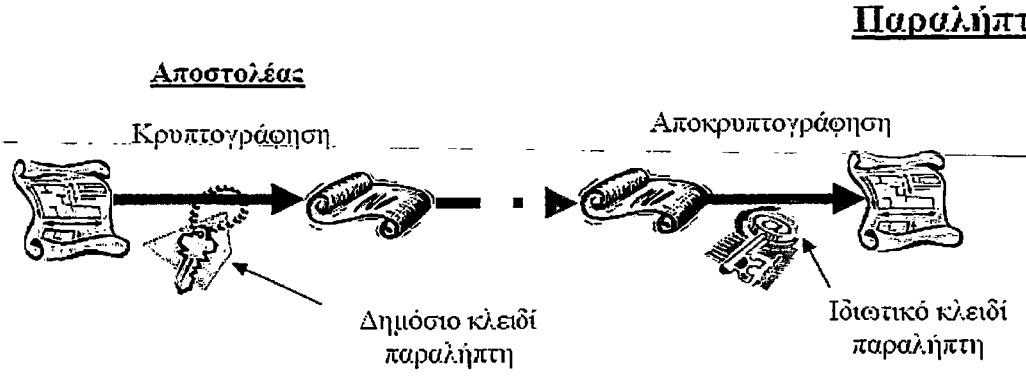



**2.6: 1ος Τρόπος κρυπτογράφησης δημόσιου κλειδιού**

Μ' αυτόν τον τρόπο πιστοποιείται η ταυτότητα του αποστολέα, αφού μόνο αυτός κατέχει το ιδιωτικό του κλειδί, με το οποίο κρυπτογραφήθηκε η πληροφορία, αλλά δεν εξασφαλίζεται η εμπιστευτικότητα της πληροφορίας, διότι οποιοσδήποτε μπορεί να αποκρυπτογραφήσει την κρυπτογραφημένη πληροφορία, χρησιμοποιώντας το δημόσιο κλειδί του παραλήπτη.

## 2ος Τρόπος κρυπτογράφησης δημόσιου κλειδιού:

Η πληροφορία κρυπτογραφείται με την χρησιμοποίηση του δημόσιου κλειδιού του παραλήπτη, το οποίο έχει παραλάβει ο αποστολέας με κάποιο τρόπο. Ο παραλήπτης αποκρυπτογραφεί την κρυπτογραφημένη πληροφορία, με την χρησιμοποίηση του ιδιωτικού του κλειδιού. Ο τρόπος αυτός, παρουσιάζεται στο παρακάτω σχήμα:

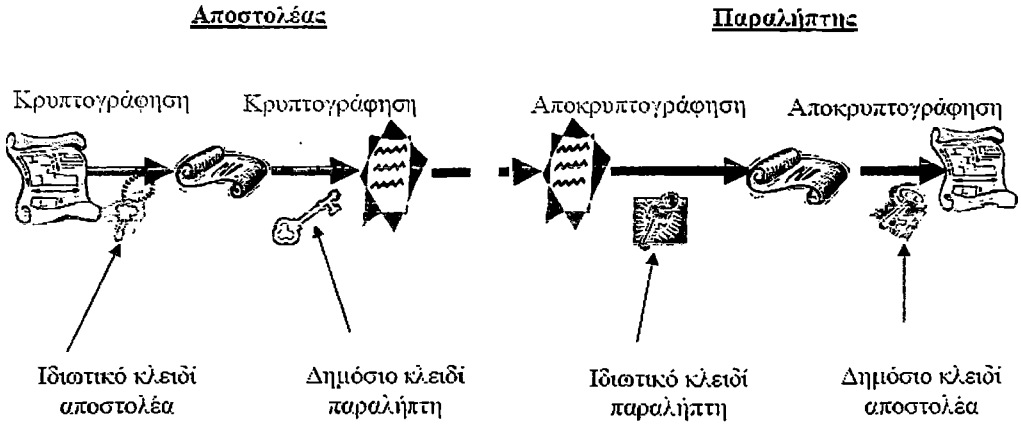



**2.7: 2ος Τρόπος κρυπτογράφησης δημόσιου κλειδιού**

Μ' αυτόν τον τρόπο εξασφαλίζεται η εμπιστευτικότητα της πληροφορίας, διότι μόνο ο παραλήπτης μπορεί να αποκρυπτογραφήσει την κρυπτογραφημένη πληροφορία αφού μόνο αυτός κατέχει το ιδιωτικό του κλειδί. Δεν πιστοποιείται όμως η ταυτότητα του αποστολέα, διότι οποιοσδήποτε μπορεί να κρυπτογραφήσει με το δημόσιο κλειδί του παραλήπτη.

## 3ος Τρόπος κρυπτογράφησης δημόσιου κλειδιού:

Ο αποστολέας κρυπτογραφεί την πληροφορία, με την χρησιμοποίηση του ιδιωτικού του κλειδιού αρχικά και στη συνέχεια με το δημόσιο κλειδί του παραλήπτη. Ο παραλήπτης αποκρυπτογραφεί την κρυπτογραφημένη πληροφορία, με την χρησιμοποίηση του ιδιωτικού του κλειδιού αρχικά και στη συνέχεια με το δημόσιο κλειδί του αποστολέα. Ο τρόπος αυτός, παρουσιάζεται στο παρακάτω σχήμα:




**2.8: 3ος Τρόπος κρυπτογράφησης δημόσιου κλειδιού**

Μ' αυτόν τον τρόπο, που είναι ο συνδυασμός των δύο προαναφερόμενων τρόπων, πιστοποιείται η ταυτότητα του αποστολέα και εξασφαλίζεται η εμπιστευτικότητα της πληροφορίας.

### 2.5.1.2.1 Ο αλγόριθμος RSA

Το RSA είναι ένα κρυπτοσύστημα που μπορεί να χρησιμοποιηθεί, τόσο για την κρυπτογράφηση δεδομένων, όσο και για την εξασφάλιση της αυθεντικότητας του αποστολέα. Ονομάστηκε έτσι από τα αρχικά των ονομάτων των ανθρώπων που τον επινόησαν, Ron Rivest, Adi Shamir και Len Adleman, το 1978.

Η ασφάλεια του RSA έγκειται στο γεγονός ότι, η εύρεση δύο μεγάλων πρώτων αριθμών είναι σχετικά εύκολη, η παραγοντοποίηση του γινομένου τους είναι δύσκολη. Αν οι αριθμοί είναι αρκετά μεγάλοι, η παραγοντοποίηση απαιτεί τεράστιους υπολογιστικούς πόρους στο βαθμό να θεωρείται το πρόβλημα υπολογιστικά δύσκολο.

Η ισχύς του RSA συχνά αμφισβητείται. Έχει έναν πολύ προφανή τρόπο να σπάσει, την παραγοντοποίηση με κάποια από τις γνωστές μεθόδους. Η ισχύς του επομένως, εξαρτάται από το χρόνο και το κόστος που απαιτείται για να γίνει αυτό. Αυτό που κάνει το RSA βιώσιμο στο μέλλον είναι το γεγονός ότι μία μικρή αύξηση του μεγέθους του υπολοίπου, συνεπάγεται σε μεγάλη αύξηση στην προσπάθεια για την παραγοντοποίησή του (με την υπάρχουσα τεχνολογία αύξηση τριών bits του υπολοίπου συνεπάγεται σε διπλασιασμό της δυσκολίας παραγοντοποίησης του). Για ικανοποιητική προστασία υπόλοιπο μήκους 1024-bits είναι αρκετό για τις περισσότερες εμπορικές

εφαρμογές. Όπου χρειάζεται περισσότερη ασφάλεια χρησιμοποιείται υπόλοιπο μήκους 2048-bits.

### 2.5.1.3 Ψηφιακές Υπογραφές

Οι ψηφιακές υπογραφές είναι κρυπτογραφικοί μηχανισμοί, και σαν βασικό στόχο έχουν να επιβεβαιώσουν την ταυτότητα του αποστολέα, όπως και οι γραπτές υπογραφές. Εάν ο αποστολέας χρησιμοποιήσει ψηφιακή υπογραφή σε κάποιο μήνυμα που αποστέλλει, δεν μπορεί να αρνηθεί ύστερα ότι έστειλε το συγκεκριμένο μήνυμα. Τα συστήματα ηλεκτρονικών υπογραφών βασίζονται στις κρυπτογραφικές τεχνικές και στην μυστικότητα των κρυπτογραφικών κλειδιών.

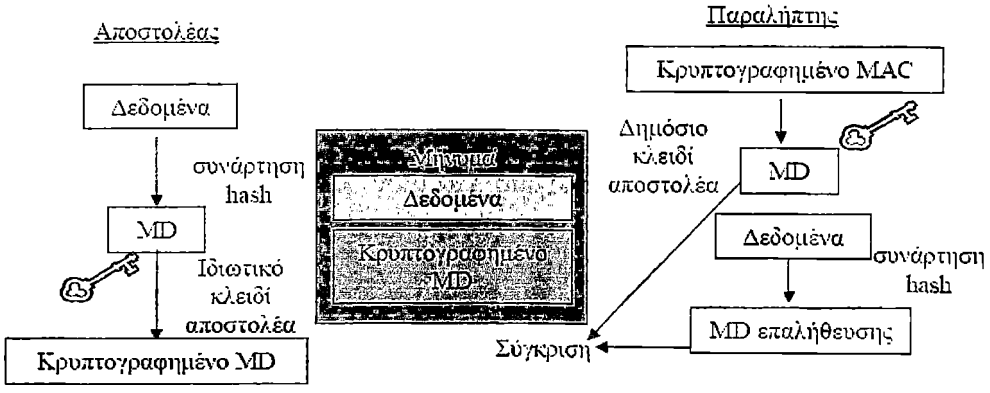
Για την υλοποίηση των ψηφιακών υπογραφών, χρησιμοποιούνται:

- **Κρυπτοσυστήματα συμμετρικού κλειδιού**

Στα συστήματα μυστικού κλειδιού, οι ηλεκτρονικές υπογραφές μπορούν να υλοποιηθούν με την χρήση του κωδικού πιστοποίησης δεδομένων (MAC). Για παράδειγμα, για ένα ζευγάρι οντοτήτων που επικοινωνεί με χρήση ενός κοινού μυστικού κλειδιού, όταν μία οντότητα λάβει ένα αρχείο που έχει υπογραφεί με έναν κωδικό πιστοποίησης μηνύματος, τότε μπορεί εύκολα να επιβεβαιώσει, με χρήση του κοινού κρυπτογραφικού κλειδιού, ότι το μήνυμα έχει σταλθεί από την άλλη οντότητα, και δεν έχει τροποποιηθεί στη διαδρομή. Αυτή η απλή διαδικασία έχει το μειονέκτημα ότι προϋποθέτει εμπιστοσύνη μεταξύ των δύο οντοτήτων που επικοινωνούν. Πιο προηγμένες τεχνικές μπορούν να χρησιμοποιηθούν για άρση της υπόθεσης για εμπιστοσύνη μεταξύ των δύο οντοτήτων. Αξίζει να σημειωθεί ότι η ομοσπονδιακή κυβέρνηση των Η.Π.Α. έχει ήδη εγκρίνει τη χρήση της τεχνολογίας κωδικού πιστοποίησης μηνύματος σαν εναλλακτικό της γραπτής υπογραφής

### ➢ Κρυπτοσυστήματα δημόσιου κλειδιού

Τα συστήματα δημόσιου κλειδιού είναι τα πλέον κατάλληλα για την υλοποίηση ηλεκτρονικών υπογραφών, που σε αυτή την περίπτωση πολλές φορές ονομάζονται και ψηφιακές υπογραφές (digital signatures). Τα δεδομένα υπογράφονται (κρυπτογραφούνται) με χρήση του ιδιωτικού κλειδιού του αποστολέα. Για να επιταχυνθεί η διαδικασία, συνήθως κρυπτογραφείται μόνο η σύνοψη των δεδομένων (message digest) που αντιστοιχείται στα δεδομένα από μία ασφαλή hash συνάρτηση. Η κρυπτογραφημένη σύνοψη του μηνύματος ονομάζεται ψηφιακή υπογραφή, και αποθηκεύεται ή μεταδίδεται μαζί με τα δεδομένα. Η ταυτότητα του αποστολέα, καθώς και η ακεραιότητα των δεδομένων μπορούν εύκολα να πιστοποιηθούν από οποιονδήποτε, με χρήση του δημόσιου κλειδιού του αποστολέα. Το χαρακτηριστικό αυτό είναι ιδιαίτερα επιθυμητό όταν για παράδειγμα μια εταιρεία προωθεί λογισμικό, το οποίο είναι πιστοποιημένο ότι δεν περιέχει ιούς ή άλλο επιβλαβή κώδικα (malicious code). Η εταιρεία μπορεί να υπογράψει ψηφιακά το μήνυμα, και κάθε παραλήπτης μπορεί να χρησιμοποιήσει το δημόσιο κλειδί της εταιρείας ώστε να επιβεβαιώσει ότι το λογισμικό που παρέλαβε συνεχίζει να είναι αυτό που απέστειλε η εταιρεία, και επομένως μπορεί να εγκατασταθεί άφοβα. Το NIST έχει δημοσιεύσει τις τυποποιήσεις FIPS 186, Digital Signature Standard, και FIPS 180, Secure Hash Standard (SHS), σχετικά με την υλοποίηση ψηφιακών υπογραφών και ασφαλών hash συναρτήσεων. Η διαδικασία σύνοψης του μηνύματος, παρουσιάζεται στο παρακάτω σχήμα:



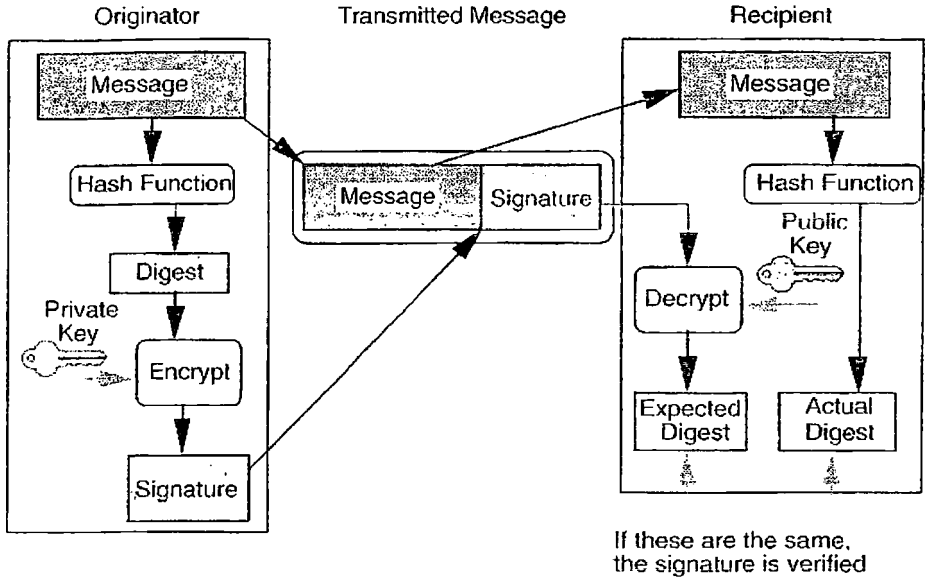

**2.9: Η διαδικασία σύνοψής του μηνύματος**

### 2.5.1.3.1 RSA Ψηφιακές Υπογραφές

Ο απλοποιημένος τρόπος λειτουργίας της RSA ψηφιακής υπογραφής είναι ο παρακάτω. Ο αποστολέας δημιουργεί μία κρυπτογραφημένη έκδοση του μηνύματος, χρησιμοποιώντας τον αλγόριθμο RSA στη λειτουργία αυθεντικότητας (δηλαδή το κλειδί της κρυπτογράφησης είναι το ιδιωτικό κλειδί του αποστολέα). Η κρυπτογραφημένη έκδοση του μηνύματος προσαρτάται μαζί με την κανονική και αποστέλλεται. Ο παραλήπτης χρησιμοποιώντας το δημόσιο κλειδί που αντιστοιχεί στον εν λόγω αποστολέα αποκρυπτογραφεί το κρυπτογραφημένο τμήμα του μηνύματος και το συγκρίνει με το κανονικό. Αν είναι ίδια τότε ο παραλήπτης είναι σίγουρος ότι το μήνυμα δεν παραποιήθηκε στην πορεία.

Η παραπάνω διαδικασία έχει ένα μειονέκτημα. Συνολικά, κάθε μήνυμα που αποστέλλεται περιέχει την πληροφορία δύο φορές, κρυπτογραφημένη και κανονική. Επιπλέον έχουμε και υπολογιστικό κόστος αφού η κρυπτογράφηση και αποκρυπτογράφηση γίνονται σε όλο το μήνυμα, όσο μεγάλο και αν είναι αυτό. Η λύση είναι η εφαρμογή μιας συνάρτησης hash στο μήνυμα πριν κρυπτογραφηθεί. Οι συναρτήσεις hash είναι μονόδρομες συναρτήσεις που μπορούν να απεικονίσουν ένα πολύ μεγάλο διάστημα σε ένα πολύ μικρό με μοναδικό τρόπο. Για παράδειγμα ένα μήνυμα μπορεί να έχει μήκος χιλιάδων ή εκατομμυρίων bits. Χρησιμοποιώντας μια συνάρτηση hash μπορούμε να παράγουμε μια σύνοψη του μηνύματος (digest), έστω 128 bits. Λόγω της φύσης της συνάρτησης hash έστω και ένα bit να αλλάξει στο μήνυμα, η σύνοψη αλλάζει.

Έτσι χρησιμοποιείται ο αλγόριθμος RSA για να κρυπτογραφήσει τη σύνοψη του μηνύματος. Όταν φτάνει το μήνυμα στον παραλήπτη αυτός υπολογίζει πάλι τη σύνοψη, αποκρυπτογραφεί τη σύνοψη που έλαβε και τις συγκρίνει. Αν είναι ίδιες τότε το μήνυμα δεν έχει υποστεί παραποίηση. Επίσης είναι σίγουρος και για τον αποστολέα του μηνύματος. Η διαδικασία της RSA ψηφιακής υπογραφής, παρουσιάζεται στο παρακάτω σχήμα:




**2.10: Η διαδικασία της RSA ψηφιακής υπογραφής**

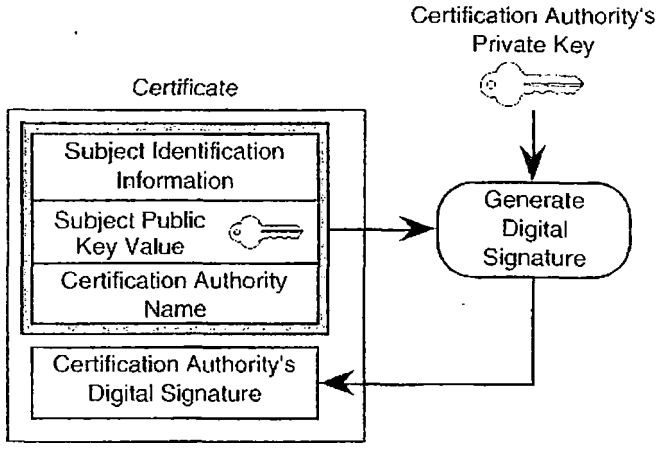
## 2.5.2 Ψηφιακά Πιστοποιητικά

Τα ψηφιακά πιστοποιητικά είναι ηλεκτρονικά έγγραφα και η βασική χρήση τους είναι η πιστοποίηση ενός δικτυακού τόπου στο World Wide Web και η σχέση του με το δημόσιο κλειδί του.

Ένας δικτυακός τόπος για την απόκτηση ενός ηλεκτρονικού πιστοποιητικού, πρέπει να απευθυνθεί σε μια Αρχή Πιστοποίησης (Certificate Authority), η οποία επιβεβαιώνει την ταυτότητα του και του εκδίδει ένα ηλεκτρονικό πιστοποιητικό, το οποίο περιλαμβάνει:

- το όνομα και πληροφορίες για τον κάτοχο του δικτυακού τόπου,
- το δημόσιο κλειδί του κατόχου,
- την ημερομηνία έναρξης και λήξης του πιστοποιητικού,
- το όνομα και την ψηφιακή υπογραφή της Αρχής Πιστοποίησης που το έκδωσε,
- τον σειριακό αριθμό του πιστοποιητικού, κ.ά.

Στο παρακάτω σχήμα παρουσιάζεται η διαδικασία κατασκευής ψηφιακού πιστοποιητικού:




**2.11: Η διαδικασία κατασκευής ψηφιακού πιστοποιητικού**

Η χρήση ψηφιακών πιστοποιητικών και κρυπτογραφικών τεχνικών, που περιγράψαμε στην παράγραφο 2.1, παρέχουν υπηρεσίες ασφάλειας στον τρόπο χρήσης του διαδικτύου για συναλλαγές, όπως:

- πιστοποίηση των αγοραστών ( Web clients),
- πιστοποίηση των εμπόρων (Web servers),
- ακεραιότητα των δεδομένων που συναλλάσσονται,
- αποφυγή προστριβών μεταξύ των μερών που συναλλάσσονται και
- εξουσιοδοτημένη πρόσβαση σε προσωπικά δεδομένα.

### 2.5.2.1 Διάκριση ψηφιακών πιστοποιητικών

Τα ψηφιακά πιστοποιητικά διακρίνονται ανάλογα με τον τύπο τους, το είδος της πληροφορίας που περιέχουν και την χρήση τους.

Ανάλογα με τον τύπο τους, τα ψηφιακά πιστοποιητικά διακρίνονται, σε:

- **Client SSL certificates.**

  Αυτά τα ψηφιακά πιστοποιητικά χρησιμοποιούνται, για την πιστοποίηση των αγοραστών στους εμπόρους, μέσω του πρωτοκόλλου SSL.

- **Server SSL certificates.**

  Αυτά τα ψηφιακά πιστοποιητικά χρησιμοποιούνται, για την πιστοποίηση των εμπόρων στους αγοραστές, μέσω του πρωτοκόλλου SSL.

- **S/MIME certificates.**
  Αυτά τα ψηφιακά πιστοποιητικά χρησιμοποιούνται, για την υπογραφή και κρυπτογράφηση μηνυμάτων ηλεκτρονικού ταχυδρομείου (e-mail).
- **Object-signing certificates.**
  Αυτά τα ψηφιακά πιστοποιητικά χρησιμοποιούνται, για την πιστοποίηση υπογεγραμμένου κώδικα σε Java, C++ κ.ά.
- **CA certificates.**
  Αυτά τα ψηφιακά πιστοποιητικά χρησιμοποιούνται, για την πιστοποίηση της Αρχής Πιστοποίησης και άλλων διάφορων φορέων.

Ανάλογα με το είδος της πληροφορίας που περιέχουν, τα ψηφιακά πιστοποιητικά διακρίνονται, σε:

- **Πιστοποιητικά ταυτότητας (Identity certificates).**
  Αυτά τα ψηφιακά πιστοποιητικά χρησιμοποιούνται, για την πιστοποίηση κάποιου μέλους που συμμετέχει στην συναλλαγή.
- **Πιστοποιητικά χαρακτηριστικών (Attribute certificates).**
  Αυτά τα ψηφιακά πιστοποιητικά χρησιμοποιούνται, για την περιγραφή των ιδιοτήτων ενός μέλους της συναλλαγής.

Ανάλογα με την χρήση τους, τα ψηφιακά πιστοποιητικά διακρίνονται, σε:

- **Σύνοδοι με βάση το πρωτόκολλο SSL (SSL Protocol)**
- **Υπογεγραμμένο και κρυπτογραφημένο ηλεκτρονικό ταχυδρομείο (Signed and Encrypted Email)**
- **Εφάπαξ διαδικασία εισόδου στο σύστημα (Single and Sign-On)**
- **Υπογραφή ηλεκτρονικών φορμών (Form Signing)**
- **Υπογραφή αντικειμένων (Object Signing)**
- **Έλεγχος προσπέλασης (Access Control)**

### 2.5.2.2 Αρχές Πιστοποίησης

Οι Αρχές Πιστοποίησης (Certification Authorities) είναι φορείς, που πιστοποιούν τις ταυτότητες των μελών που συμμετέχουν σε μια συναλλαγή, εκδίδοντας τα αντίστοιχά ψηφιακά πιστοποιητικά τους. Μια Αρχή Πιστοποίησης, μπορεί να είναι:

- ένας Έμπιστος Τρίτος Φορέας (Trusted Third Party- TTP) ή
- ένα τμήμα ενός οργανισμού, που λειτουργεί στα πλαίσια του.

# 3 Σχεδιασμός του συστήματος

## 3.1 Unified Modeling Language (UML)

Η Ενοποιημένη Γλώσσα Μοντελοποίησης (UML) είναι μια οπτική (visual) γλώσσα μοντελοποίησης με εκφραστικές δυνατότητες για την: απεικόνιση (visualization), προδιαγραφή (specification), τεκμηρίωση (documentation) και κατασκευή (construction) των προϊόντων της διαδικασίας ανάπτυξης συστημάτων. Αποτελεί σήμερα βιομηχανικό πρότυπο το οποίο αναπτύχθηκε στο πλαίσιο του οργανισμού OMG και βρίσκεται σε συνεχή εξέλιξη. Ανήκει δε στην κατηγορία των αντικειμενοστρεφών γλωσσών μοντελοποίησης.

### 3.1.1 Οι Στόχοι της UML

Σύμφωνα με τους Booch, Jacobson και Rumbaugh, που είναι γνωστοί στη βιβλιόγραφια και ως The Three Amigos, η UML καλείται να διεκπεραιώσει τους εξής στόχους:

- Να παρέχει στους χρήστες μιας έτοιμη προς χρήση, εκφραστική, οπτική γλώσσα μοντέλοποίησης, έτσι ώστε να μπορούν να αναπτύξουν και να ανταλλάξουν μοντέλα που είναι κατανοητά από όλους.
- Να διαθέτει μηχανισμούς επεκτασιμότητας και εξειδίκευσης, έτσι ώστε να είναι δυνατή η επέκταση των βασικών εννοιών της.
- Να είναι ανεξάρτητη από γλώσσες προγραμματισμού και διαδικασίες ανάπτυξης.
- Να διαθέτει κάποιες βασικές έννοιες που θα διευκολύνουν την κατανόηση της.

- Να διευκολύνει την ανάπτυξη εργαλείων που να την υποστηρίζουν.
- Να υποστηρίζει τα υψηλότερα επίπεδα της διαδικασίας ανάπτυξης.
- Να περιλαμβάνει, στο βασικό της πυρήνα, τις καλύτερες πρακτικές και μεθοδολογίες ανάπτυξης.

### 3.1.2 Τα Διαγράμματα της UML

Η UML ορίζει δώδεκα τύπους διαγραμμάτων, οι οποίοι διαιρούνται σε τρεις κατηγορίες, τα διαγράμματα αυτά είναι τα εξής:

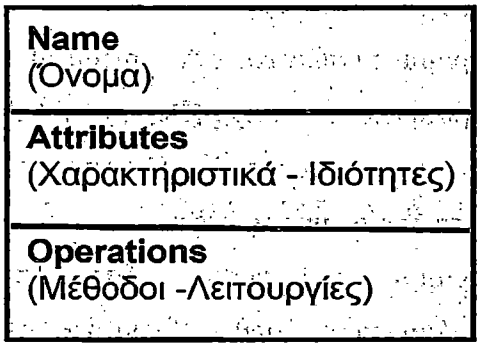
- Στην πρώτη κατηγορία συμπεριλαμβάνονται τα στατικά διαγράμματα εφαρμογών, τα οποία είναι τα εξής:
  - ✓ Τα διαγράμματα κλάσεων (Class Diagrams)
  - ✓ Τα διαγράμματα αντικειμένων (Object Diagrams)
  - ✓ Τα διαγράμματα συστατικών (Component Diagrams)
  - ✓ Τα διαγράμματα ανάπτυξης (Deployment Diagrmas)
- Στην δεύτερη κατηγορία συμπεριλαμβάνονται τα δυναμικά διαγράμματα συμπεριφοράς, τα οποία είναι τα εξής:
  - ✓ Τα διαγράμματα περιπτώσεων χρήσης (Use Case Diagrams)
  - ✓ Τα διαγράμματα ακολουθίας (Sequence Diagrams)
  - ✓ Τα διαγράμματα δραστηριότητας (Activity Diagrams)
  - ✓ Τα διαγράμματα συνεργασίας (Collaboration Diagrams)
  - ✓ Τα διαγράμματα καταστάσεων (Statechart Diagrams)
- Στην τρίτη κατηγορία συμπεριλαμβάνονται τα διαγράμματα για την οργάνωση και την διαχείριση εφαρμογών, τα οποία είναι τα εξής:
  - ✓ Τα πακέτα (Packages)
  - ✓ Τα υποσυστήματα (Subsystems)
  - ✓ Τα μοντέλα (Models)

Παρακάτω θα αναλυθούν από την πρώτη κατηγορία, τα διαγράμματα κλάσεων (Class Diagrams) και από την δεύτερη κατηγορία, τα διαγράμματα περιπτώσεων χρήσης (Use Case Diagrams).

### 3.1.2.1 Τα Διαγράμματα Κλάσεων (Class Diagrams)

Γενικά λόγια για τις κλάσεις:

- Οι κλάσεις, τα αντικείμενα και οι μεταξύ τους συσχετίσεις είναι τα πρωταρχικά στοιχεία μοντελοποίησης στην αντικειμενοστραφή θεώρηση.
- Οι κλάσεις και τα αντικείμενα περιγράφουν τι υπάρχει μέσα στο σύστημα που περιγράφουμε.
- Μια κλάση είναι μια περιγραφή ενός τύπου αντικειμένου: περιγράφει τα χαρακτηριστικά και τη συμπεριφορά του συγκεκριμένου τύπου αντικειμένου.
- Όλα τα αντικείμενα είναι στιγμιότυπα μιας κλάσης.




**3.1: Διάγραμμά κλάσης**

Τα Χαρακτηριστικά (Attributes) τών κλάσεων:

- Η γενική περιγραφή ενός χαρακτηριστικού είναι :
  - ✓ **Ορατότητα Όνομα: Τύπος = Αρχική Τιμή {Property String}**
- **Ορατότητα:** Περιγράφει αν το χαρακτηριστικό είναι ορατό και αν μπορούν να αναφερθούν σε αυτό άλλες κλάσεις, εκτός από αυτή στην οποία ορίζεται (- (ιδιωτικό) + (δημόσιο) # (προστατευόμενο)).
- **Τύπος:** Μπορεί να είναι πρωταρχικός τύπος όπως ακέραιος, πραγματικός, αλφαριθμητικός, κ.λ.π. Μπορεί να είναι και μια κλάση αντικειμένων.

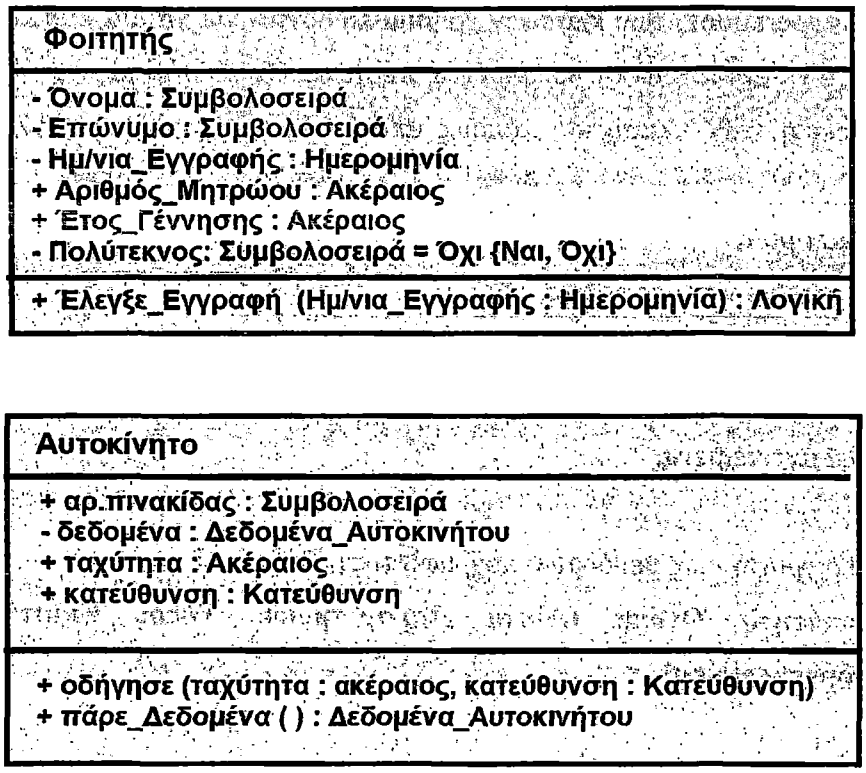
Οι μέθοδοι (Operations) των κλάσεων χρησιμοποιούνται για να διαχειριζόμαστε τα χαρακτηριστικά ή να εκτελούμε συγκεκριμένες ενέργειες. Οι μέθοδοι περιγράφουν τι υπηρεσίες προσφέρει η κάθε κλάση και κάποιες από αυτές παρέχουν την κατάλληλη διασύνδεση.

Οι μέθοδοι μπορούν:

- Να παίρνουν πληροφορίες.
- Να ενημερώνουν.
- Να κάνουν κάποιες ενέργειες πάνω στο αντικείμενο ή/και να καλούν άλλα αντικείμενα.

Η γενική περιγραφή μιας μεθόδου (υπογραφή της μεθόδου) είναι:

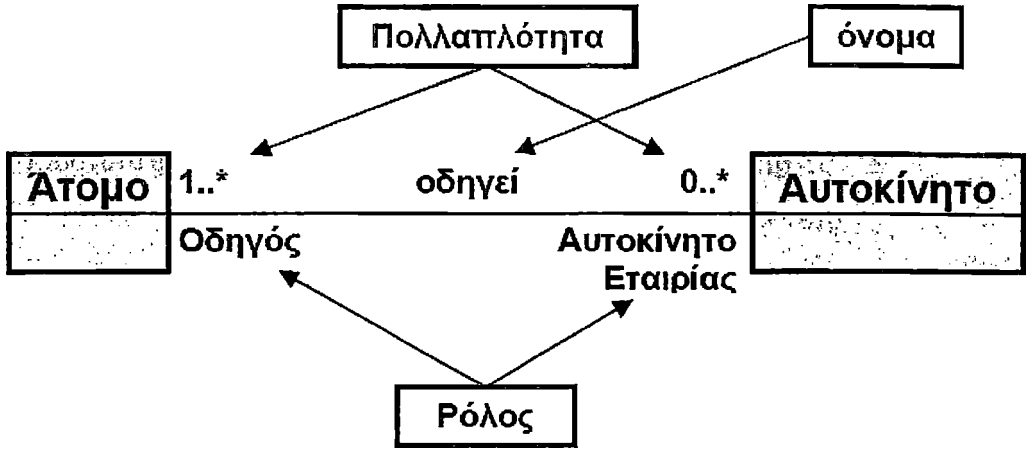
- **Ορατότητα Όνομα (Λίστα Παραμέτρων): Τύπος Επιστροφής {property string}**




**3.2: Παράδειγμα κλάσεων**

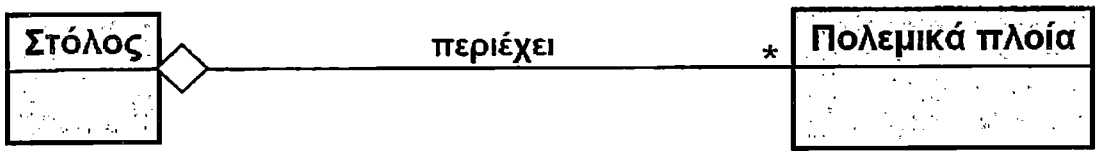
Οι Σχέσεις μεταξύ των Κλάσεων είναι οι εξής:

- Συσχέτιση (Association): είναι μια σημασιολογική σχέση μεταξύ των αντικειμένων δύο ή περισσότερων κλάσεων.



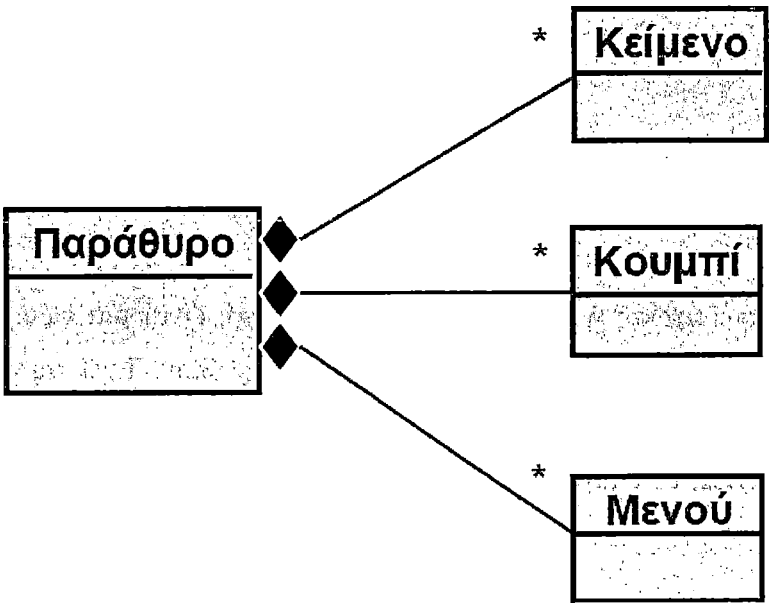

**3.3: Συσχέτιση κλάσεων**

- Συσσωμάτωση ή Συνάθροιση (Aggregation): είναι μία ειδική περίπτωση συσχέτισης που συσχετίζει το μέρος με το όλον. Έχει τις ιδιότητες της μεταβατικότητας και της αντισυμμετρικότητας. Το μέρος μπορεί να συνεχίζει να υπάρχει και εφόσον καταστραφεί το όλον.



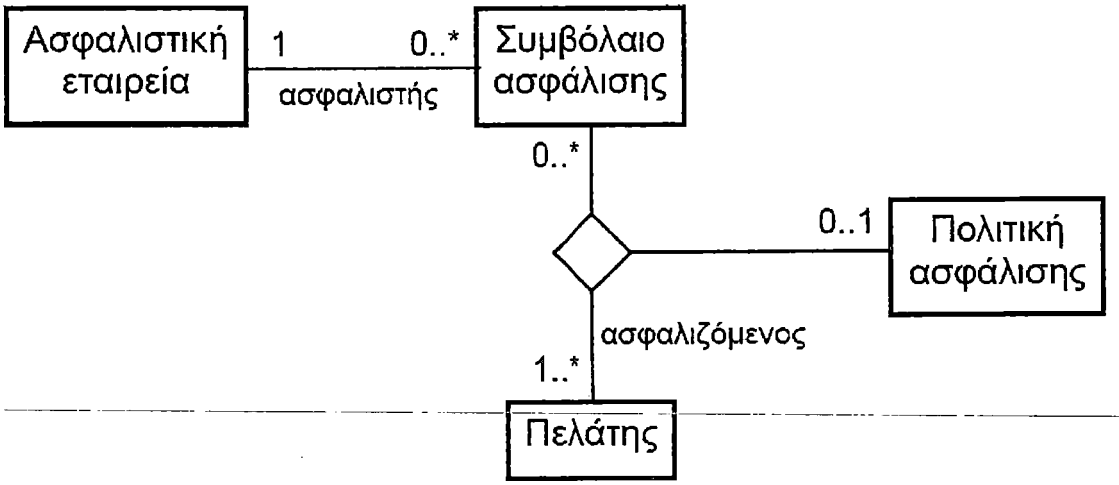

**3.4: Συσσωμάτωση κλάσεων**

- Σύνθεση (Composition): είναι μία ειδική περίπτωση συσχέτισης που:
  - ✓ Παρουσιάζει ισχυρή ιδιοκτησία των τμημάτων της.
  - ✓ Τα τμήματα που «ζουν» μέσα στο σύνολο και θα καταστραφούν, όταν καταστραφεί και το σύνολο.



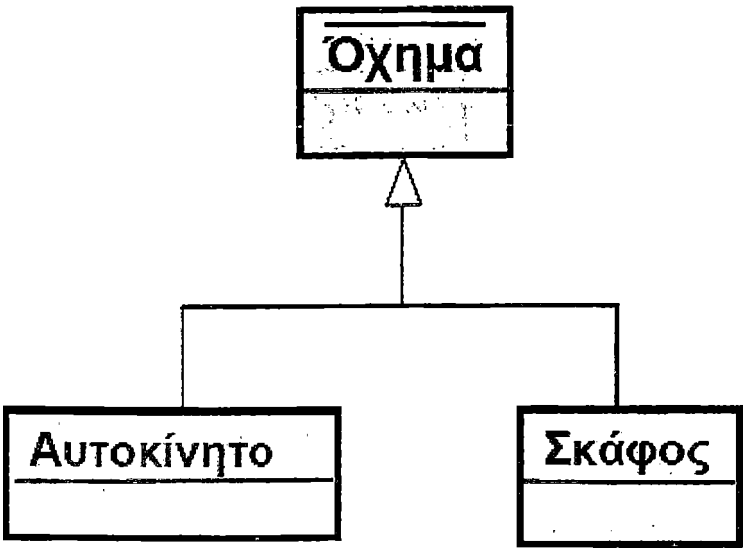

**3.5: Σύνθεση κλάσεων**

- Ν-αδική Συσχέτιση (N-ary Association): συσχετίζει Ν κλάσεις.



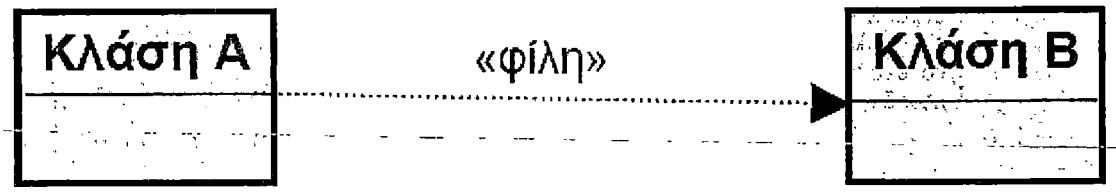

**3.6: Ν-αδική Συσχέτιση κλάσεων**

- Γενίκευση (Generalization): είναι η σχέση ανάμεσα σ' ένα πιο γενικό (υπερκλάση) και σ' ένα πιο εξειδικευμένοστοιχείο (υποκλάση). Το ειδικό στοιχείο:
  - ✓ Θα πρέπει να περιέχει μόνο επιπρόσθετες πληροφορίες.
  - ✓ Κληρονομεί όλα τα χαρακτηριστικά του γενικού.
  - ✓ Μπορεί να έχει επιπλέον χαρακτηριστικά.




**3.7: Γενίκευση κλάσεων**

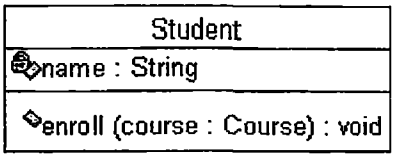
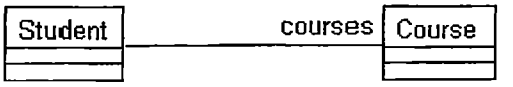
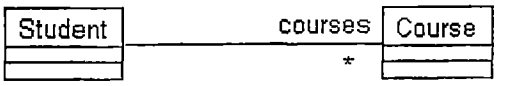
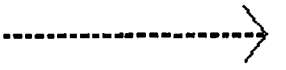
- Εξάρτηση (Dependency): σχέση ανάμεσα σε ανεξάρτητο κι εξαρτημένο στοιχείο. Μια ενδεχόμενη αλλαγή στο ανεξάρτητο στοιχείο θα επηρεάσει το εξαρτημένο στοιχείο. Παραδείγματα:
  - ✓ Μια κλάση παίρνει το αντικείμενο κάποιας άλλης κλάσης σαν παράμετρο.
  - ✓ Μια κλάση καλεί μία μέθοδο κλάσης μιας άλλης κλάσης.

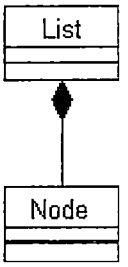
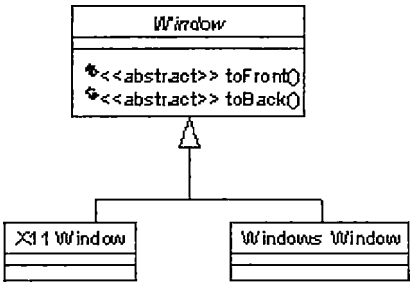


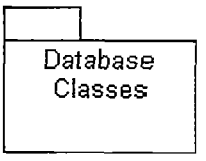

**3.8: Εξάρτηση κλάσεων**

Όταν χρησιμοποιείται μια ετικέτα όπως στο παράδειγμά μας «φίλη», είναι ένα στερεότυπο που προσδιορίζει το είδος της εξάρτησης.

Στον παρακάτω πίνακα που ακολουθεί συνοψίζονται τα βασικά διαγραμματικά στοιχεία, που συναντάμε στα διαγράμματα κλάσεων. Καθώς και η περιγραφή τους και από ένα παράδειγμα σύνταξης:

| ΚΑΤΑΣΚΕΥΗ | ΠΕΡΙΓΡΑΦΗ | ΠΑΡΑΔΕΙΓΜΑ ΣΥΝΤΑΞΗΣ |
|---|---|---|
| **Τάξη** | Αποτελεί την βάση της κατασκευής οποιουδήποτε αντικειμενοστραφούς συστήματος. Ενσωματώνει (encapsulate) δεδομένα και τις λειτουργίες που επενεργούν στα δεδομένα αυτά. Στο πάνω διαμέρισμα απεικονίζονται οι ιδιότητες και στο κάτω διαμέρισμα οι μέθοδοι |  |
| **Συσχέτιση** | Οι συσχετίσεις αναπαριστούν σχέσεις μεταξύ των περιστάσεων των τάξεων (Ένας άνθρωπος εργάζεται για μια εταιρεία. Μια εταιρεία έχει ένα αριθμό γραφείων). |  |
| **Ρόλος** | Ένα πέρας μπορεί να ονομαστεί με μια ετικέτα. Αυτή η ετικέτα ονομάζεται όνομα ρόλου (role name). (Τα πέρατα των συσχετίσεων συχνά καλούνται **ρόλοι**). |  |
| **Πολλαπλότητα** | Το πέρας μιας συσχέτισης έχει **πολλαπλότητα (multiplicity),** που είναι μια ένδειξη για το πόσα αντικείμενα μπορούν να συμμετέχουν σ' αυτή τη σχέση. Αυτή συνήθως είναι 1 (ακριβώς 1), 0.. 1 (προαιρετική συσχέτιση), * (πολλά αντικείμενα) ή και κάποιος συγκεκριμένος αριθμός (π.χ. 11) |  |
| **Εξάρτηση** | Η εξάρτηση δηλώνει πως μια αλλαγή σε μιαν οντότητα θα επηρεάσει μιαν άλλη αλλά όχι απαραίτητα και το αντίστροφο. Παριστάνεται με μια διακεκομμένη γραμμή με ανοιχτό βέλος που δείχνει προς την οντότητα που υπάρχει |  |

| | | |
|---|---|---|
| | εξάρτηση. | |
| **Σύνθεση** | Η **σύνθεση (composition)** είναι και αυτή μια σχέση όλου και μερών που το αποτελούν. Μόνο που αυτή τη φορά τα μέρη είναι περισσότερο έντονα συνδεδεμένα με το όλο. **Συγκεκριμένα η διάρκεια ζωής των μερών περιέχεται στην διάρκεια ζωής του όλου, δηλαδή τα μέρη δεν ζουν χωρίς το όλο στο οποίο περιέχονται.** Επίσης κάθε μέρος ανήκει σε ένα και μόνο όλο. Αυτό το τελευταίο συνεπάγεται ότι τα μέρη είναι επίσης ιδιωτικά στο όλο που τα περιέχει. Η σύνθεση παριστάνεται με ένα μαύρο διαμάντι, όπως φαίνεται στο σχήμα. |  |
| **Γενίκευση**<br><br>**(Αφηρημένες Τάξεις)** | Η γενίκευση δηλώνει μια σχέση ανάμεσα σε κάτι γενικό (τη βασική κλάση ή αλλιώς γονέα) και κάτι ειδικό (μιαν υποκλάση ή αλλιώς παιδί της). Παριστάνεται με μια συνεχή γραμμή με κλειστό βέλος που δείχνει προς τη βασική κλάση: Μία **αφηρημένη τάξη (abstract class)** αφήνει ορισμένες λειτουργίες της αόριστες, οι οποίες και ονομάζονται αφηρημένες λειτουργίες και απλά τις δηλώνει. Σε αντίθεση όμως με μια διασύνδεση μια αφηρημένη τάξη μπορεί να παρέχει και την υλοποίηση κάποιων λειτουργιών. Μια αφηρημένη τάξη συμβολίζεται με το στερεότυπο «abstract» για την τάξη και τις μεθόδους της που είναι αφηρημένες ή το όνομα της τάξης και τα ονόματα των αφηρημένων λειτουργιών της εμφανίζονται με πλάγια γραφή (*italics*). Όπως και από μια διασύνδεση έτσι και |  |

| | | |
|---|---|---|
| | από μια αφηρημένη τάξη δεν μπορούν να δηλωθούν αντικείμενα. Μια τάξη που κληρονομεί από μια αφηρημένη τάξη πρέπει να υλοποιήσει όλες τις αφηρημένες λειτουργίες της αφηρημένης υπερτάξης της διαφορετικά θα είναι και αυτή αφηρημένη | |
| **Πακέτα** | Αποτελούν συλλογές τάξεων με σκοπό τον λογικό διαχωρισμό των τάξεων και την μείωση των εξαρτήσεων. Αν η κάθε τάξη ανήκει σε ένα πακέτο και τεκμηριώσουμε τις εξαρτήσεις μεταξύ των πακέτων, τότε οποιαδήποτε αλλαγή σε μια τάξη μπορεί να επιφέρει αλλαγές μόνο στα πακέτα που εξαρτώνται από το πακέτο της τάξης που άλλαξε. Ένα πακέτο συμβολίζεται όπως στην εικόνα. |  |

**3.9: Τα βασικά διαγραμματικά στοίχεια των κλάσεων**

### 3.1.2.2 Τα Διαγράμματα Περιπτώσεων Χρήσης (Use Case Diagrams)

Τα διάγραμματα περιπτώσεων χρήσης χρησιμοποιούνται για να μοντελοποιήσουν το πλαίσιο λειτουργίας του συστήματος, καθώς και τις προδιαγραφές του. Τα διάγραμματα περιπτώσεων χρήσης περιλαμβάνουν τα εξής:

- Περιπτώσεις χρήσης.
- Δρώντες (Actors), που δρούν έξω από το σύστημα.
- Σχέσεις εξάρτησης, γενίκευσης, σύνδεσης
- Τα όρια του συστήματος

Το μοντέλο περιπτώσεων χρήσης είναι μία άποψη του συστήματος που αποδίδει έμφαση στην λειτουργικότητα ενός συστήματος, όπως αυτή είναι ορατή από τους εξωτερικούς χρήστες του συστήματος. Μια περίπτωση χρήσης διαμερίζει την λειτουργικότητα ενός συστήματος σε συναλλαγές («περιπτώσεις χρήσης») που έχουν νόημα για τους χρήστες του συστήματος («ρόλους»).

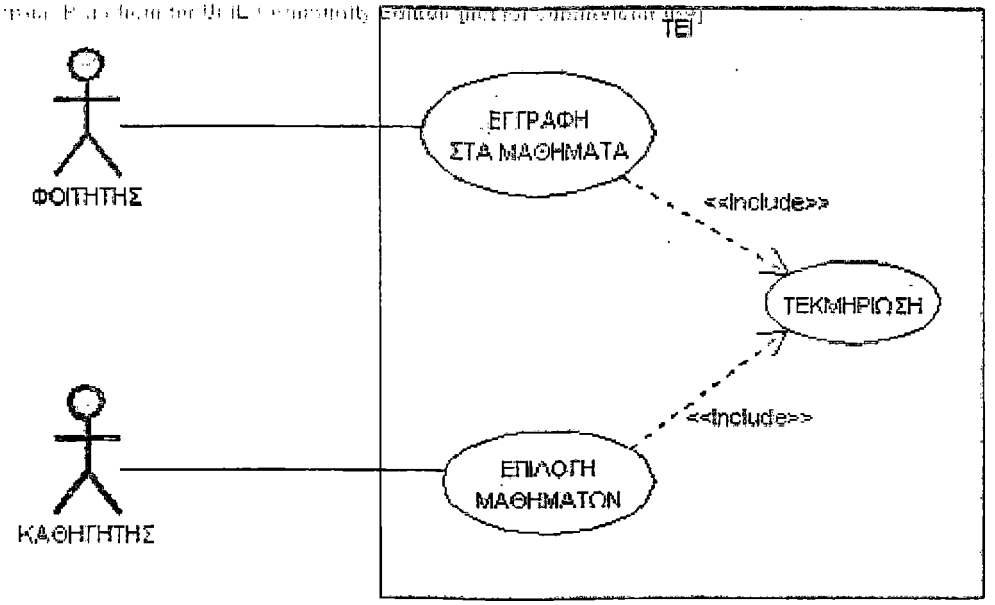
Στον παρακάτω πίνακα που ακολουθεί συνοψίζονται τα βασικά διαγραμματικά στοιχεία, που συναντάμε στα διαγράμματα περιπτώσεων χρήσης. Καθώς και η λειτουργία τους:

| Σύμβολο | Ονομασία | Λειτουργία |
|---|---|---|
|  | **Note**<br>**Σημείωση** | Χρησιμοποιείται για το σχολιασμό επιπρόσθετων εξηγήσεων, προδιαγραφών, απαιτήσεων σε ένα στοιχείο ή μια σύνδεση του διαγράμματος. |
|  | **Anchor**<br>**Άγκυρα** | Χρησιμοποιείται για να συνδέσει ένα στοιχείο του διαγράμματος και μια σημείωση. |
|  | **Package**<br>**Πακέτο** | Είναι ένας μηχανισμός για να οργανώνουμε στοιχεία σε ομάδες. Τα πακέτα μπορεί να αποτελούνται από άλλα πακέτα. Ένα πακέτο μπορεί να περιέχει επίσης δευτερεύοντα πακέτα και κανονικά στοιχεία μοντέλου. |
|  | **System**<br>**Σύστημα** | Μια ομάδα περιπτώσεων χρήσεων με κοινό σκοπό. Κάθε διάγραμμα περιπτώσεων χρήσης μπορεί να έχει μόνο ένα σύστημα. |
|  | **Use Case**<br>**Περίπτωση Χρήσης** | Περιγράφει τις ιδιότητες ενός συστήματος. Περιέχει σενάρια, που περιγράφουν μια ποικιλία ενεργειών αρκετά ξεκάθαρη για να γίνει κατανοητή από κάποιο τρίτο. |
|  | **Actor**<br>**Ρόλος - Ηθοποιός** | Απαρτίζει ένα σετ από ρόλους τους οποίους παίζουν οι χρήστες του συστήματος όταν αλληλεπιδρούν με τις περιπτώσεις χρήσεις του συστήματος. Ένας ηθοποιός συμμετέχει σε μια ή περισσότερες περιπτώσεις χρήσης για να πετύχει ένα γενικό σκοπό. Ένας ηθοποιός μπορεί να παίζει το ρόλο ενός ανθρώπου, μιας συσκευής ή ενός άλλου συστήματος. |
|  | **Association**<br>**Συσχέτιση** | Χρησιμοποιείται για να συσχετίσει ένα ηθοποιό με μια περίπτωση χρήσης. Δείχνει τη συμμετοχή ενός ηθοποιού σε μια περίπτωση χρήσης. |
|  | **Include**<br>**Περιεκτικότητα** | Είναι ένας δεσμός που συνδέει μια περιεχόμενη περίπτωση χρήσης με μια βασική περίπτωση χρήσης. |
|  | **Extend**<br>**Επέκταση** | Είναι ένας δεσμός που συνδέει μια επέκταση μιας περίπτωσης χρήσης με μια περίπτωση χρήσης βάσης. |
|  | **Dependency**<br>**Εξάρτηση** | Μια σύνδεση εξάρτησης είναι μία σημαντική σχέση μεταξύ δύο στοιχείων. Δείχνει πότε μια αλλαγή που συμβαίνει σε ένα στοιχείο, μπορεί να αλλάξει υποχρεωτικά ένα άλλο στοιχείο. |

**3.10: Τα βασικά διαγραμματικά στοίχεια των περιπτώσεων χρήσης**

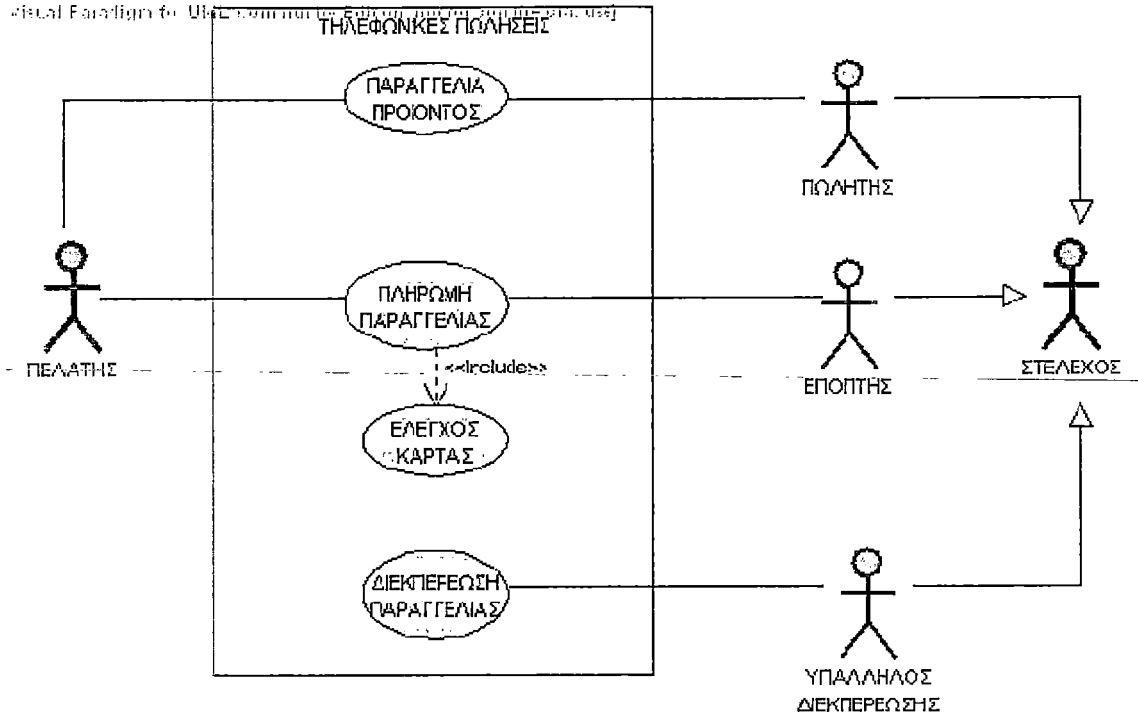
Παρακάτω ακολουθούν τρία παραδείγματα περιπτώσεων χρήσης, με σκοπό να γίνουν πιο κατανοητά:

- **Παράδειγμα 1°: Εγγραφή σε μάθημα**



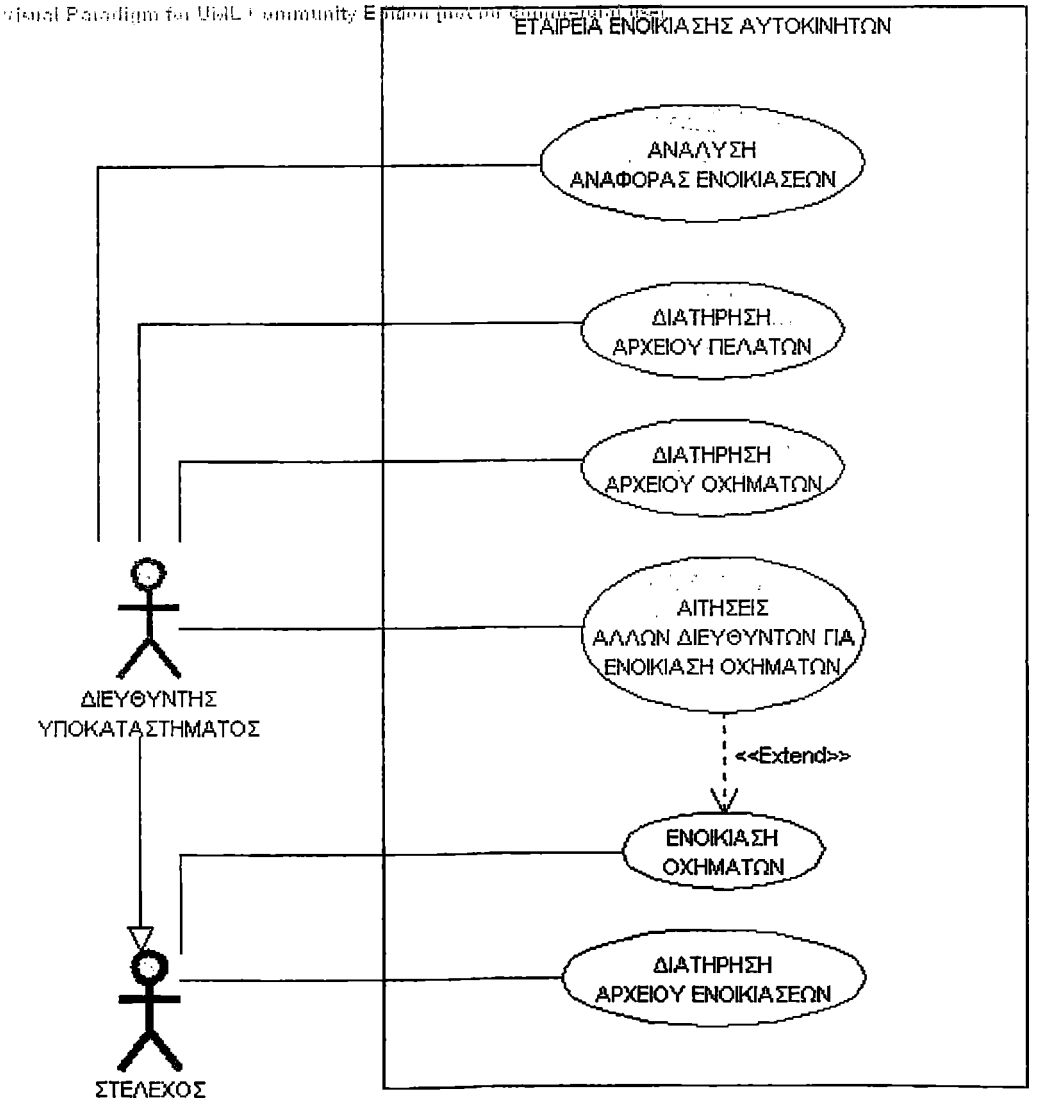

**3.11: Διάγραμμα περίπτωσης χρήσης για εγγραφή σε μάθημα**

- **Παράδειγμα 2$^{ο}$: Τηλεφωνική πώληση**




**3.12: Διάγραμμα περίπτωσης χρήσης για τηλεφωνική πώληση**

- **Παράδειγμα 3ο: Δαχείριση εταιρείας ενοικίασης αυτοκινήτων**




**3.13: Διάγραμμα περίπτωσης χρήσης για διαχείριση εταιρείας ενοικίασης αυτοκινήτων**

# 3.2 Διαγράμματα Ροής Δεδομένων

Τα **Διαγράμματα Ροής Δεδομένων - ΔΡΔ (Data Flow Diagrams - DFDs)** αναπαριστούν ένα σύστημα λογισμικού με βάση τα δεδομένα που παράγονται ή διακινούνται σ' αυτό. Αποτελούν μια λογική αναπαράσταση του συστήματος, χωρίς να περιέχουν πληροφορίες για το υλικό, το λογισμικό ή τα αρχεία που το αποτελούν. Για τους λόγους αυτούς, τα ΔΡΔ είναι κατάλληλα για την κατανόηση της λειτουργίας του συστήματος και από μη ειδικούς.

Σε κάθε ΔΡΔ αναπαρίσταται η πορεία (ροή) που ακολουθεί κάθε τμήμα δεδομένων από το σημείο δημιουργίας του, έως το σημείο εξόδου από το σύστημα.

## 3.2.1 Οι Οντότητες των ΔΡΔ

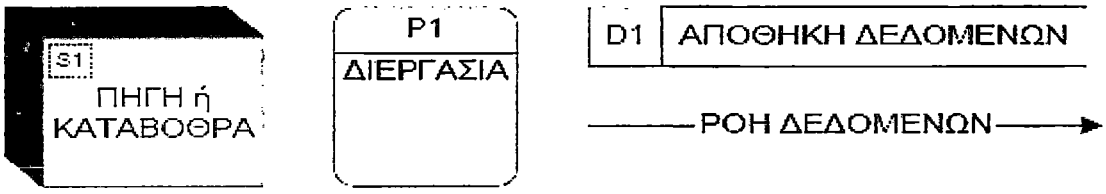
Οι βασικές οντότητές που χρησιμοποιούνται πιο συχνά για την κατασκευή των Διαγραμμάτων Ροής Δεδομένων είναι οι εξής:

- **Ροή Δεδομένων (Data Flow)**
- **Πηγή Δεδομένων (Data Source)**
- **Καταβόθρα Δεδομένων (Data Sink)**
- **Διεργασία (Process)**
- **Αποθήκη Δεδομένων (Data store)**

Η βασική οντότητα είναι η **ροή δεδομένων (data flow)**, η οποία απεικονίζει την ανταλλαγή δεδομένων ανάμεσα στα συστατικά μέρη του συστήματος. Τα δεδομένα εισέρχονται στο σύστημα από τις **πηγές δεδομένων (data sources)** και εγκαταλείπουν το σύστημα στις **καταβόθρες δεδομένων (data sinks).** Οι δύο αυτές οντότητες αναπαριστούν το περιβάλλον του συστήματος.

Σύμφωνα με τα ΔΡΔ, ένα σύστημα θεωρείται ότι επεξεργάζεται πληροφορίες που παράγουν οι πηγές και στέλνει τα αποτελέσματα στις καταβόθρες. Οι οντότητες του συστήματος που εκτελούν την επεξεργασία καλούνται **διεργασίες (processes).** Μέσα σ' ένα σύστημα λογισμικού, οι ροές δεδομένων εισέρχονται σε μια διεργασία, υφίστανται την κατάλληλη επεξεργασία και εξέρχονται από αυτή.

Τέλος, πολλές φορές είναι απαραίτητη η προσωρινή ή όχι αποθήκευση των δεδομένων σε οντότητες που λέγονται **αποθήκες δεδομένων (data stores).** Με τις αποθήκες δεδομένων επικοινωνούν μόνο οι διεργασίες, οι οποίες είτε γράφουν είτε διαβάζουν δεδομένα (δηλαδή δεν επιτρέπεται να συνδεθούν απευθείας δύο αποθήκες δεδομένων χρησιμοποιώντας μια ροή δεδομένων). Παρακάτω ακολουθούν τα συμβολά των οντοτήτων των ΔΡΔ:




**3.14: Τα σύμβολα των οντοτητών των ΔΡΔ**

Ας σημειωθεί ότι μια διεργασία δεν αναπαριστά απαραίτητα ένα πρόγραμμα υπολογιστή. Μπορεί να αναπαριστά ένα πρόγραμμα, αλλά και μια ομάδα προγραμμάτων, ένα τμήμα του λογισμικού, ή και μια εργασία που εκτελείται χωρίς το λογισμικό. Ομοίως, μια αποθήκη δεδομένων δεν αντιστοιχεί σε ένα αρχείο, αλλά μπορεί να αναπαριστά ένα αρχείο, αλλά και μια βάση δεδομένων, μια εγγραφή ή ένα πεδίο. Μια ροή δεδομένων αναπαριστά δεδομένα που βρίσκονται σε κίνηση, ενώ σε μια αποθήκη δεδομένων, αυτά βρίσκονται αποθηκευμένα: πρόκειται για δύο διαφορετικές καταστάσεις της ίδιας οντότητας (δεδομένα).

## 3.2.2 Λεξικό Δεδομένων

Ενα ΔΡΔ δεν είναι από μόνο του ικανό να προδιαγράψει τις απαιτήσεις από ένα σύστημα λογισμικού, αφού δεν περιέχει περιγραφές των οντοτήτων που το αποτελούν, όπως τί ακριβώς περιέχεται σε μια αποθήκη δεδομένων ή τί κάνει μια διεργασία κ.α.

Οι πληροφορίες αυτές περιέχονται συνήθως στο **λεξικό δεδομένων - ΛΔ (Data Dictionary)**, ή αλλιώς λεξικό απαιτήσεων, το οποίο συνοδεύει ένα ΔΡΔ. Το ΛΔ ορίζεται ως "ένας οργανωμένος κατάλογος όλων των στοιχείων δεδομένων που ενυπάρχουν στο σύστημα, με ακριβείς και αυστηρούς ορισμούς ώστε τόσο ο τελικός χρήστης όσο και ο μηχανικός λογισμικού να αντιλαμβάνονται με τον ίδιο τρόπο τις εισόδους και εξόδους, τα περιεχόμενα των αποθηκών δεδομένων και τους ενδιάμεσους υπολογισμούς".

Ενα ΛΔ είναι μια συλλογή δεδομένων για τα δεδομένα και εξυπηρετεί πολλούς και σημαντικούς σκοπούς, όπως:

- αποτελεί ένα κοινό και κατανοητό έγγραφο αναφοράς τόσο για τους τεχνικούς, όσο και τους τελικούς χρήστες
- χρησιμοποιείται για τη βελτίωση της διασύνδεσης και της επικοινωνίας μεταξύ διαφορετικών τμημάτων λογισμικού (τα οποία μπορεί να αναπτύσσονται από διαφορετικές ομάδες)
- διευκολύνει την κατοπινή αναβάθμιση του συστήματος λογισμικού
- μπορεί να χρησιμοποιηθεί για τον έλεγχο της ορθότητας και της σωστής επεξεργασίας των δεδομένων
- αποτελεί το πρώτο βήμα δημιουργίας των βάσεων δεδομένων του συστήματος

Το Λ.Δ αποτελεί μια συλλογή καρτελών, κάθε μια καρτέλα περιγράφει ένα στοιχείο δεδομένων. Ετσι, είναι εύκολη η ενημέρωση του Λ.Δ με τυχόν αλλαγές, η νέα καρτέλα για το στοιχείο δεδομένων τοποθετείται στη θέση της παλαιάς. Ενας ενδεικτικός κατάλογος με πληροφορίες που συνήθως διατηρούνται για κάθε στοιχείο δεδομένων ενός Λ.Δ είναι:

- Γενικές πληροφορίες: περιλαμβάνουν το όνομα του στοιχείου, πιθανά ψευδώνυμα (διαφορετικά ονόματα που όμως περιγράφουν ουσιαστικά το ίδιο στοιχείο) καιμια λεκτική περιγραφή
- Πληροφορίες μορφής: περιγράφουν χαρακτηριστικά αποθήκευσης και εμφάνισης του στοιχείου (π.χ. τύπος δεδομένων, μέγεθος, κ.α.)
- Χαρακτηριστικά ελέγχου και χρήσης: περιγράφουν τους σωστούς τρόπους χρήσης του στοιχείου (π.χ. πεδίο τιμών, συχνότητα και εμβέλεια χρήσης,

τιμές υπό συνθήκη, κ.α.), και δίνουν στοιχεία ελέγχου της ορθότητας χρήσης (πότε και από πού παράγεται, πού χρησιμοποιείται, ποιά διεργασία έχει δικαίωμα προσπέλασης, τροποποίησης ή διαγραφής, κ.α.)

- Πληροφορίες για την ομάδα δεδομένων στην οποία ανήκει το στοιχείο (π.χ. σε ποιές δομές δεδομένων ανήκει ή αποθηκεύεται, κ.α.)

Παρακάτω ακολουθεί ένα παράδειγμα Λεξικού Δεδομένων και ποιές πληροφορίες καταγράφονται σ' αυτό:

| **Γενικές** |
|---|
| Ονομα |
| Συνώνυμα |
| Περιγραφή |

| **Μορφή** |
|---|
| Τύπος δεδομένων |
| Μέγεθος |

| **Πληροφορίες ομάδας** |
|---|
| Υπερκείμενη δομή |
| Υποκείμενες δομές |
| Δομές επανάληψης |
| Θέση φυσικής αποθήκευσης |

| **Πληροφορίες ελέγχου** |
|---|
| Παράγεται από: |
| Ημερομηνία παραγωγής |
| Χρησιμοποιείται από: |
| Δικαιώματα τροποποίησης |
| Δικαιώματα προσπέλασης |

| **Χαρακτηριστικά χρήσης** |
|---|
| Πεδίο τιμών |
| Συχνότητα χρήσης |
| Εμφάνιση |
| Εμβέλεια |
| Τιμές υπό συνθήκη |

**3.15: Λεξικό Δεδομένων**

Τα στοιχεία δεδομένων μπορεί να είναι στοιχειώδη ή σύνθετα, δηλαδή να αποτελούνται από άλλα στοιχεία δεδομένων. Τα μεν πρώτα περιγράφονται με την ερμηνεία του πεδίου τιμών τους, ενώ τα δεύτερα εκφράζονται σε σχέση με τα συστατικά τους μέρη.

Υπάρχουν γενικά, τρεις τρόποι παραγωγής σύνθετων δεδομένων, οι οποίοι είναι οι εξής:

- ως μια ακολουθία στοιχείων δεδομένων: το στοιχείο Α **είναι** στοιχείο Ρ **και** στοιχείο Q **και** στοιχείο R
- ως επιλογή ενός στοιχείου από ένα σύνολο στοιχείων: το στοιχείο Α **είναι** στοιχείο Ρ **είτε** στοιχείο Q **είτε** στοιχείο R
- ως ομάδα προερχόμενη από την επανάληψη ενός στοιχείου: το στοιχείο Α **είναι επανάληψη του** στοιχείο Ρ

### 3.2.3 Κατασκευή ΔΡΔ

Ενα σύστημα μπορεί να αναπαρασταθεί με ΔΡΔ σε οποιοδήποτε επίπεδο λεπτομέρειας. Συνήθως, στο πρώτο βήμα, το Διάγραμμα Μηδενικού Επιπέδου, αναπαριστούμε το σύστημα με ένα ΔΡΔ που έχει μόνο μια διεργασία, στην οποία εισέρχονται οι ροές δεδομένων που προέρχονται από το περιβάλλον του συστήματος, και από την οποία εξέρχονται οι ροές δεδομένων προς το περιβάλλον. Το περιβάλλον αναπαρίσταται με πηγές και καταβόθρες δεδομένων.

Στη συνέχεια, η διεργασία αυτή αναλύεται σε άλλες διεργασίες, και κάθε διεργασία αναλύεται με τη σειρά της, έως ότου φτάσουμε σε ένα ικανοποιητικό επίπεδο λεπτομέρειας. Για παράδειγμα, ένα σημείο τερματισμού της ανάλυσης ορίζεται ως το επίπεδο όπου κάθε περιεχόμενη διεργασία μπορεί να περιγραφεί με ακρίβεια το πολύ σε μια σελίδα. Η διαδικασία κατασκευής ενός ΔΡΔ φαίνεται καλύτερα στη μελέτη περίπτωσης που ακολουθεί.

#### 3.2.3.1 Μελέτη Περίπτωσης

Έχουμε ένα Πληροφοριακό Σύστημα, το οποίο διαχειρίζεται τις παραγγελίες πελατών σ' ένα ηλεκτρονικό κατάστημα. Σ' αυτό το Π.Σ θα πρέπει αρχικά ο πελάτης να έχει την δυνατότητα να παραγγείλει τα προιόντα που θέλει. Το Π.Σ θα πρέπει να έχει την

δυνατότητα αποστολής των προιόντων στον αγοραστή, αφού πρώτα έχει προπληρώσει με κάποιο τρόπο.

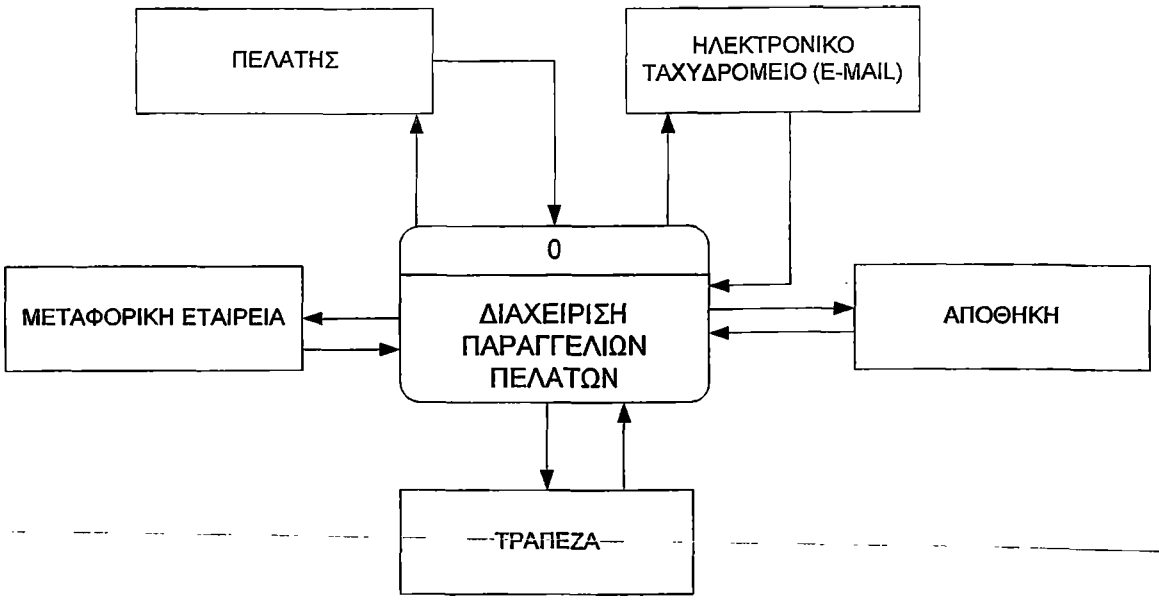
Επίσης θα πρέπει το Π.Σ να ελέγχει την ορθότητα των παραγγελίων του κάθε πελάτη και τα στοιχεία πο τον χαρακτηρίζουν. Το Π.Σ θα πρέπει να έχει την δυνατότητα να αποθηκεύει τις εξής πληροφορίες:

- Τις παραγγελίες των πελατών.
- Τα οικονομικά στοιχεία των πελατών, δηλαδή αν έχει πληρώσει ή αν χρωστάει ο πελάτης απο προηγούμενες αγορές.

### 3.2.3.2 ΔΡΔ Μηδενικού Επιπέδου

Στο ΔΡΔ Μηδενικού Επιπέδου απεικονίζεται μία διεργασία η **Διαχείριση Παραγγέλιων Πελατών**, που είναι συνολικά το σύνολο των διεργασίων του Π.Σ που αναλύουμε και οι εξωτερικές οντότητες που επικοινωνούν μ' αυτό, που είναι οι εξής:

- Πελάτης
- Ηλεκτρονικό Ταχυδρομείο (E-Mail)
- Μεταφορική Εταιρεία
- Τράπεζα
- Αποθήκη




**3.16: ΔΡΔ Μηδενικού Επιπέδου για την διεργασία διαχείρισης παραγγελιών πελατών**

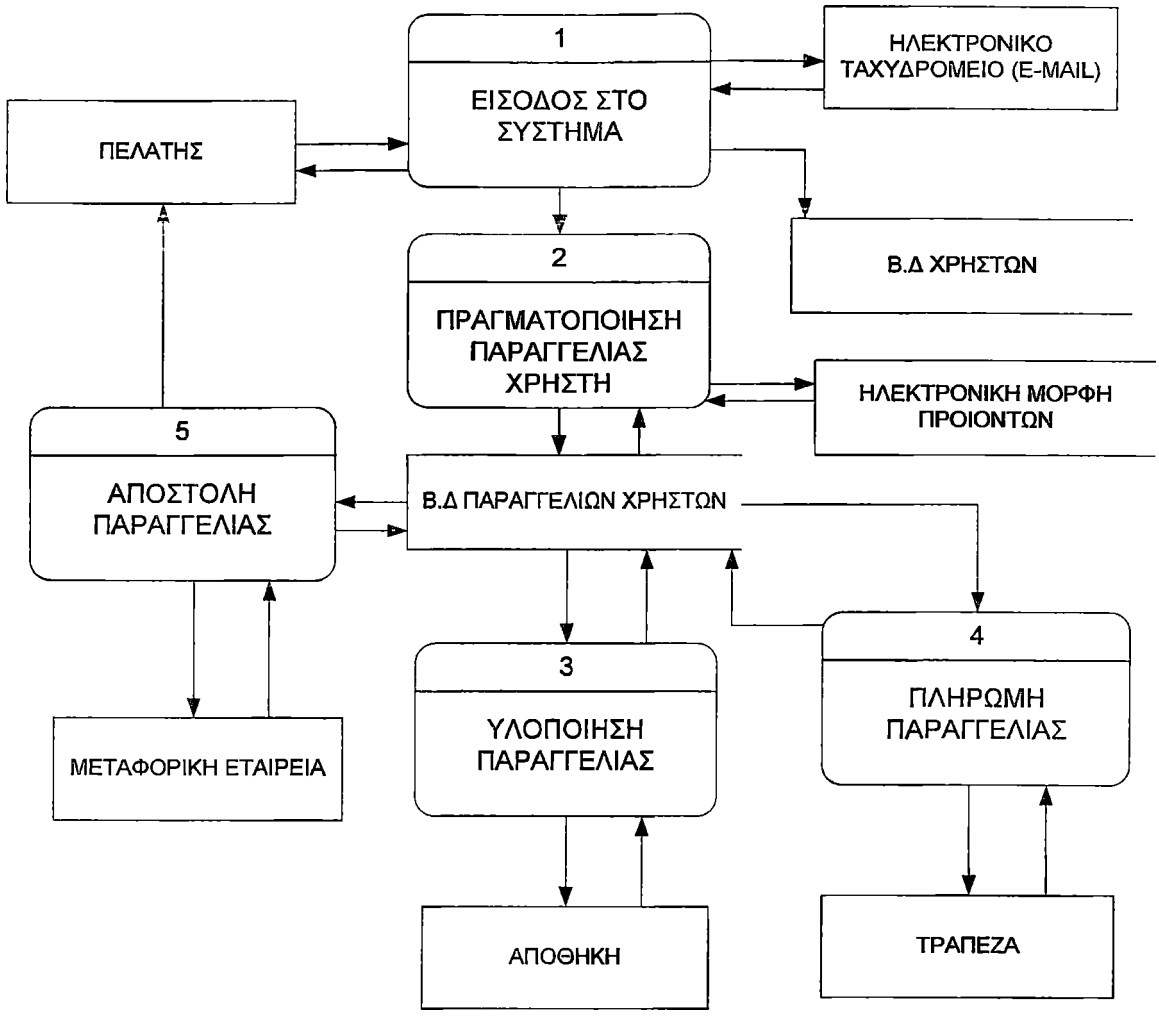
### 3.2.3.3 ΔΡΔ Πρώτου Επιπέδου

Στο ΔΡΔ Πρώτου Επιπέδου αναλύεται η διεργασία **Διαχείρισης Παραγγέλιων Πελατών** στις εξής διέργασίες:

- **Είσοδος Στο Σύστημα**
- **Πραγματοποίηση Παραγγελίας Χρήστη**
- **Υλοποίηση Παραγγέλίας**
- **Πληρωμή Παραγγελίας**
- **Αποστολή Παραγγελίας**

Ακόμα απεικονίζονται οι αποθήκες δεδομένων, οι οποίες είναι οι εξής:

- **Β.Δ Χρηστών**
- **Β.Δ Παραγγελιών Χρηστών**
- **Ηλεκτρονική Μορφή Προιόντων**




**3.17: ΔΡΔ Πρώτου Επιπέδου για την διεργασία διαχείρισης παραγγελιών πελατών**

### 3.2.3.4 ΔΡΔ Δευτέρου Επιπέδου

Στα ΔΡΔ Δευτέρου Επιπέδου, αναλύεται ξεχωριστά κάθε διεργασία του πρώτου επιπέδου. Όταν επιχειρείται ένα τέτοιο βήμα ανάλυσης μιας διεργασίας, ο αναλύτης πρέπει να προσέξει τα εξής σημεία:

- κάθε νέο διάγραμμα πρέπει να διατηρεί "τη συνέχεια της ροής". Αυτό σημαίνει ότι όλες οι ροές που εισέρχονταν σε ή εξέρχονταν από μια διεργασία σ' ένα ΔΡΔ επιπέδου 1, πρέπει να εμφανίζονται εισερχόμενες ή εξερχόμενες σε διεργασίες του ΔΡΔ επιπέδου 2, όπου αναλύεται αυτή η διεργασία.
- δεν πρέπει να υπάρχουν διεργασίες που "καταπίνουν" δεδομένα χωρίς να παράγουν έξοδο, ούτε πρέπει να υπάρχουν διεργασίες που παράγουν δεδομένα με "μαγικό" τρόπο, χωρίς να δέχονται καμία είσοδο.
- μια αποθήκη δεδομένων πρέπει να περιέχει όλα τα δεδομένα που, με διάφορες ροές δεδομένων, φαίνεται ότι εισέρχονται σε ή εξέρχονται από αυτή.

Παρακάτω ακολουθούν τα ΔΡΔ Δευτέρου επιπέδου για τις εξής διεργασίες:

- Είσοδος Στο Σύστημα
- Πραγματοποίηση Παραγγελίας Χρήστη
- Πληρωμή της Παραγγελίας
- Αποστολή Παραγγελίας

### 3.2.3.4.1 Για Την Διεργασία Είσοδος Στο Σύστημα

Η Διεργασία Είσοδος Στο Σύστημα αναλύεται στις εξής διεργασίες:

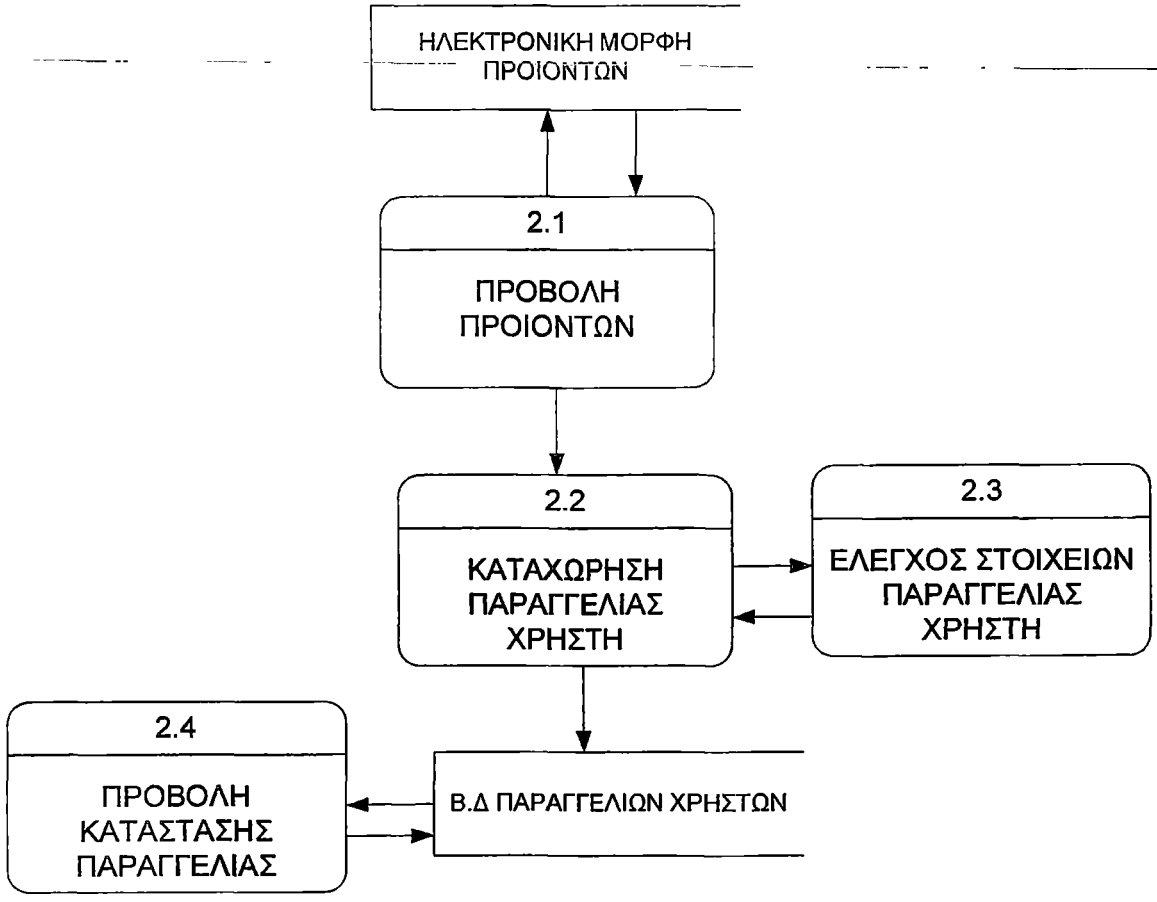
- Εγγραφή Στο Σύστημα
- Έλεγχος Ορθότητας Στοιχείων Πελάτη
- Ταυτοποίηση και Πιστοποίηση


**3.18: ΔΡΔ Δευτέρου Επιπέδου για την διεργασία είσοδος στο σύστημα**

### 3.2.3.4.2 Για Την Διεργασία Πραγματοποίηση Παραγγελίας Χρήστη

Η Διεργασία Πραγματοποίηση Παραγγελίας Χρήστη αναλύεται στις εξής διεργασίες:

- Προβολή Προιόντων
- Καταχώρηση Παραγγελίας Χρήστη
- Έλεγχος Στοιχείων Παραγγελίας Χρήστη
- Προβολή Κατάστασης Παραγγελίας

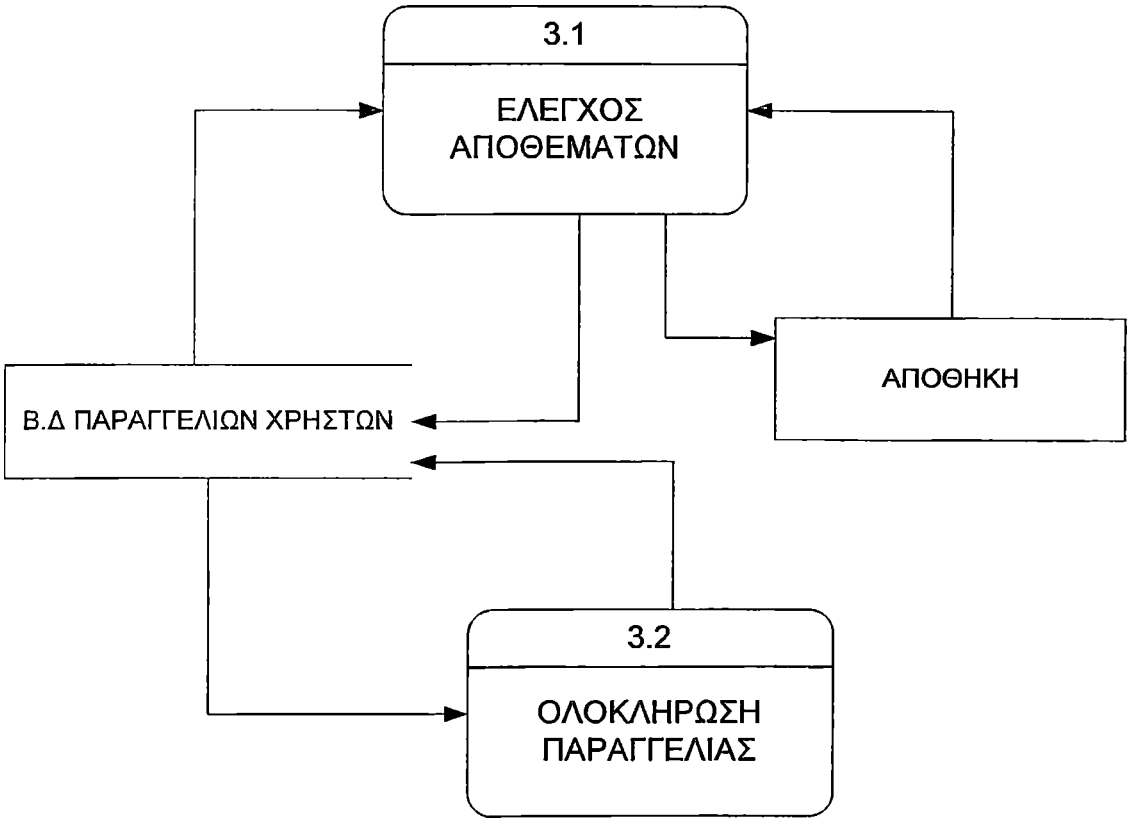



**3.19: ΔΡΔ δευτέρου επιπέδου για την διεργασία πραγματοποίηση παραγγελίας χρήστη**

### 3.2.3.4.3 Για Την Διεργασία Πληρωμή της Παραγγελίας

Η Διεργασία Πληρωμή της Παραγγελίας, αναλύεται στις εξής διεργασίες:

- Έλεγχος Αποθεμάτων
- Ολοκλήρωση Παραγγελίας

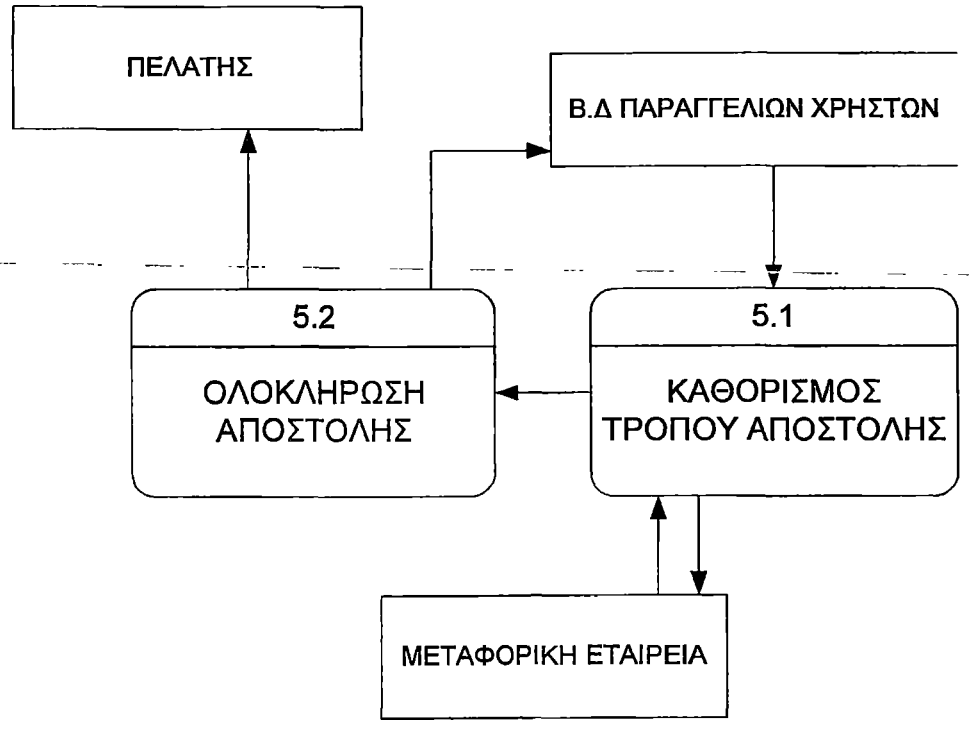



**3.20: ΔΡΔ δευτέρου επιπέδου για την διεργασία πληρωμή της παραγγελίας**

### 3.2.3.4.4 Για Την Διεργασία Αποστολή Παραγγελίας

Η Διεργασία Αποστολή Παραγγελίας, αναλύεται στις εξής διεργασίες:

- Καθορισμός Τρόπου Αποστολής
- Ολοκλήρωση Αποστολής




**3.21: ΔΡΔ δευτέρου επιπέδου για την διεργασία αποστολή παραγγελίας**

# 3.3 Το Μοντέλο Οντοτήτων Συσχετίσων

Ένα σύστημα αυτόματης επεξεργασίας δεδομένων είναι ένα σύστημα το οποίο δέχεται, επεξεργάζεται και παράγει δεδομένα σε προκαθορισμένη - τυποποιημένη μορφή (π.χ. καταστάσεις, φόρμες, τιμολόγια, αποδείξεις, κ.λπ.). Υπάρχουν βέβαια και δεδομένα που αποτελούν μέρος ενός Π.Σ. και έχουν άτυπες μορφές (π.χ. μέσω ανθρώπινης επικοινωνίας). Από τις διάφορες λειτουργίες ενός Π.Σ, ιδιαίτερη έμφαση δίνεται στις λειτουργίες που αφορούν την αυτόματη επεξεργασία των δεδομένων.

Το μοντέλο επεξεργασίας δεδομένων δημιουργείται με βασικό σκοπό την υποστήριξη των τωρινών και μελλοντικών αναγκών της επιχείρησης για επεξεργασία τυποποιημένων δεδομένων. Είναι λοιπόν αναγκαίο να βρεθούν, να οργανωθούν και να παρουσιαστούν όλα τα δεδομένα που χρησιμοποιεί το σύστημα και όχι μόνο αυτά που είναι αποθηκευμένα σε αρχεία. Για το λόγο αυτό το μοντέλο επεξεργασίας δεδομένων μπορεί να χρησιμοποιηθεί για την ανακάλυψη τόσο κανόνων που διέπουν την επιχείρηση όσο και γεγονότων.

## 3.3.1 Σχεσιακή Ανάλυση Δεδομένων

Η σχεσιακή ανάλυση δεδομένων ή κανονικοποίηση των δεδομένων, όπως διαφορετικά λέγεται, στηρίζεται στη μαθηματική θεωρία συνόλων και ξεκίνησε από το έργο του Edgar Codd (1979). Ο Codd και οι άλλοι ερευνητές παρατήρησαν ότι τόσο στις επιχειρήσεις όσο και στους υπολογιστές, κατά τις δεκαετίες του '70 και του '80, τα δεδομένα συλλέγονταν και αποθηκεύονταν κατά μη αποτελεσματικό τρόπο. Για το λόγο αυτό παρουσίασαν μια σειρά από βήματα, τα οποία στηρίζονται στη μαθηματική θεωρία συνόλων έχουν ως αποτέλεσμα την ορθότερη οργάνωση (ομαλοποίηση) των δεδομένων.

Τα βήματα αυτά επιτρέπουν στον αναλυτή να:

- αυξήσει και να βελτιώσει τη γνώση του για τα δεδομένα
- εντοπίσει τυχόν εξαρτήσεις μεταξύ των στοιχειωδών δεδομένων (ιδιοτήτων)
- απομακρύνει τυχόν πλεονάζοντα δεδομένα, επειδή προκαλούν προβλήματα στη συντήρηση των δεδομένων

- επικυρώσει το μοντέλο οντοτήτων – σχέσεων που κατασκευάστηκε με τηνκαθοδική (από πάνω προς τα κάτω) προσέγγιση, με ένα ανοδικό (από κάτω προς τα πάνω) έλεγχο.

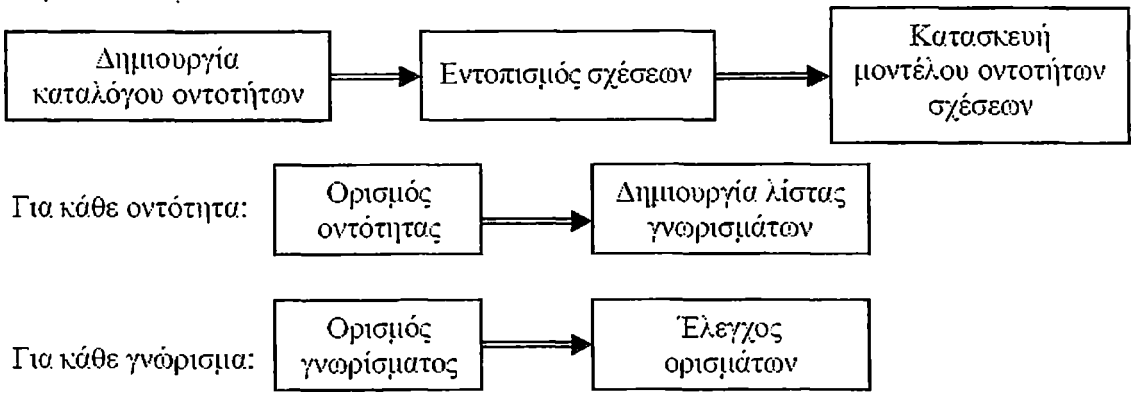
## 3.3.2 Κατασκευή Του Μοντέλου Οντοτήτων Συσχετίσων

Η κατασκευή ενός μοντέλου οντοτήτων σχέσεων δεν ακολουθεί κάποιον συγκεκριμένο αλγόριθμο και μάλιστα εξαρτάται κυρίως από την πείρα και τις ικανότητες του αναλυτή Π.Σ.. Στη συνέχεια περιγράφονται τα βήματα ενός εμπειρικού τρόπου κατασκευής του μοντέλου οντοτήτων - συσχετίσεων.

Τα τρία βασικά χαρακτηριστικά που πρέπει να έχει ένα μοντέλο οντοτήτων - συσχετίσεων, για να είναι αποδοτικό είναι:

- Αναλυτικό (με όλες τις απαραίτητες λεπτομέρειες).
- Λακωνικό (χωρίς πλεονασμούς και αναφορές σε μη σημαντικά στοιχεία).
- Ανεξάρτητο της τεχνολογίας υλοποίησης.

Σε γενικές γραμμές η κατασκευή του μοντέλου οντοτήτων - συσχετίσεων οντοτήτων ακολουθεί την παρακάτω πορεία, που φαίνεται στο παρακάτω σχήμα:




***3.22:* Η πορεία κατασκευής του μοντέλου οντοτήτων - συσχετίσεων**

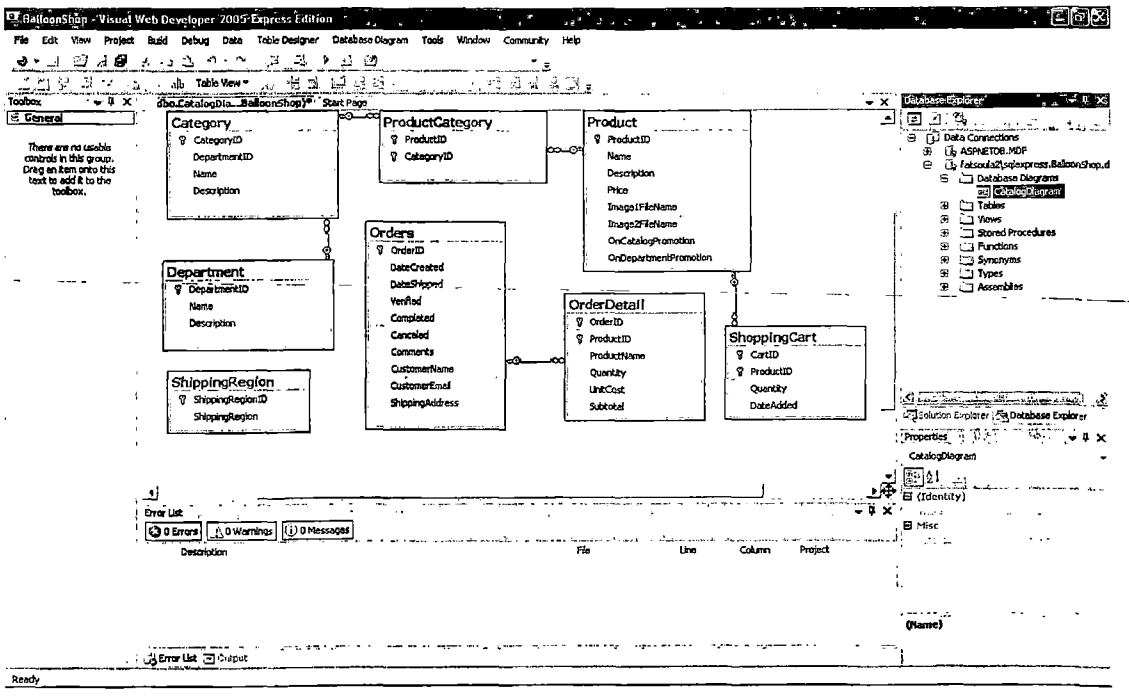
Προφανώς, η δημιουργία ενός καινούριου μοντέλου οντοτήτων - συσχετίσεων θα βασιστεί τουλάχιστον αρχικά στα στοιχεία του υπάρχοντος συστήματος. Έτσι, ένας έμπειρος αναλυτής θα ψάξει για υποψήφιες οντότητες και συσχετίσεις στα εξής:

- Στα αποτελέσματα της διερευνητικής μελέτης.
- Στα αποτελέσματα της μελέτης σκοπιμότητας.
- Στις εξωτερικές οντότητες που έχουν συνδιαλλαγές με την επιχείρηση/οργανισμό.
- Στις δομές δεδομένων των αρχείων του συστήματος.
- Σε υπάρχουσες αναφορές, καταλόγους, πολιτικές του συστήματος.
- Σε υπάρχοντα έντυπα, παραστατικά, φόρμες (τιμολόγια, καταστάσεις, πίνακες, κ.λπ.).
- Σε συνεντεύξεις, ερωτηματολόγια, προσωπική παρατήρηση, μετρήσεις.

Εκμεταλλευόμενος όλα τα παραπάνω ως στοιχεία εισόδου ο αναλυτής προσπαθεί να κατασκευάσει το τελικό μοντέλο οντοτήτων - συσχετίσεων .

### 3.3.2.1 Το Μοντέλο Οντοτήτων Συσχετίσων Της Εφαρμογής μας

Παρακάτω ακολουθεί το μοντέλο οντοτήτων - συσχετίσων της εφαρμογής BalloonShop, η οποία θα παρουσιαστεί στο επόμενο κεφάλαιο.




3.23: Το μοντέλο οντοτήτων – συσχετίσεων της εφαρμογής

# 4 Υλοποίηση του συστήματος

## 4.1 Εργαλεία Υλοποίησης

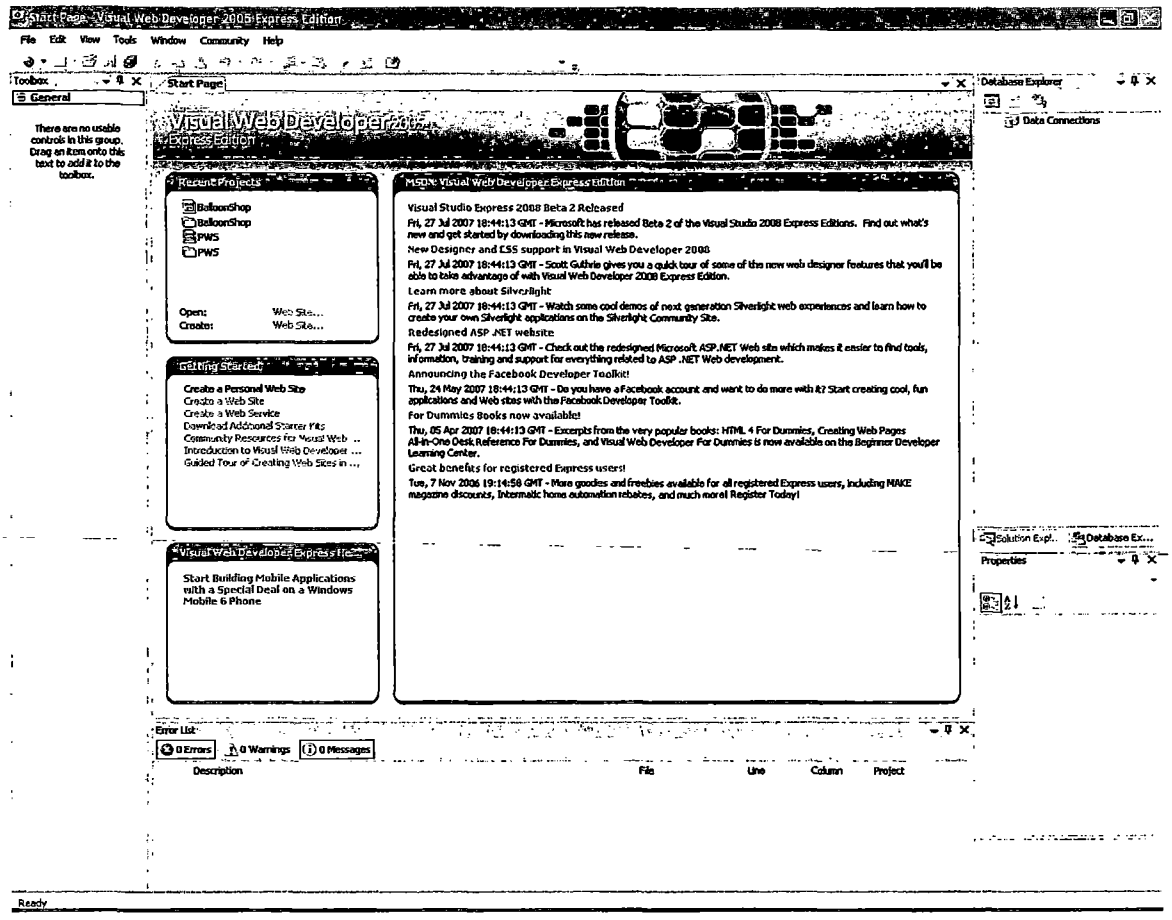
Τα εργαλεία που χρησιμοποιήθηκαν για την υλοποίηση του συστήματος είναι τα εξής και παρουσιάζονται παρακάτω:

- **Microsoft Visual Web Developer 2005 Express Edition**
- **Microsoft SQL Server 2005 Express Edition**
- **SQL Server Management Studio Express**
- **Microsoft Internet Information Services 5.1**

### 4.1.1 Visual Web Developer

Το Visual Web Developer είναι μια γραφική πλατφόρμα δημιουργίας ιστοσελίδων στο διαδίκτυο. Βασίζεται στο δημοφιλές προγραμματιστικό εργαλείο της Microsoft Visual Studio.NET και περιέχει μόνο, τις λειτουργίες του για τη δημιουργία τοποθεσιών ιστού (Web Site). Η τρέχουσα έκδοση του είναι η Microsoft Visual Web Developer 2005 Express Edition και διατίθεται δωρεάν από την εταιρία Microsoft στην ακόλουθη ηλεκτρονική διεύθυνση:

http://msdn2.microsoft.com/en-us/express/aa700797.aspx

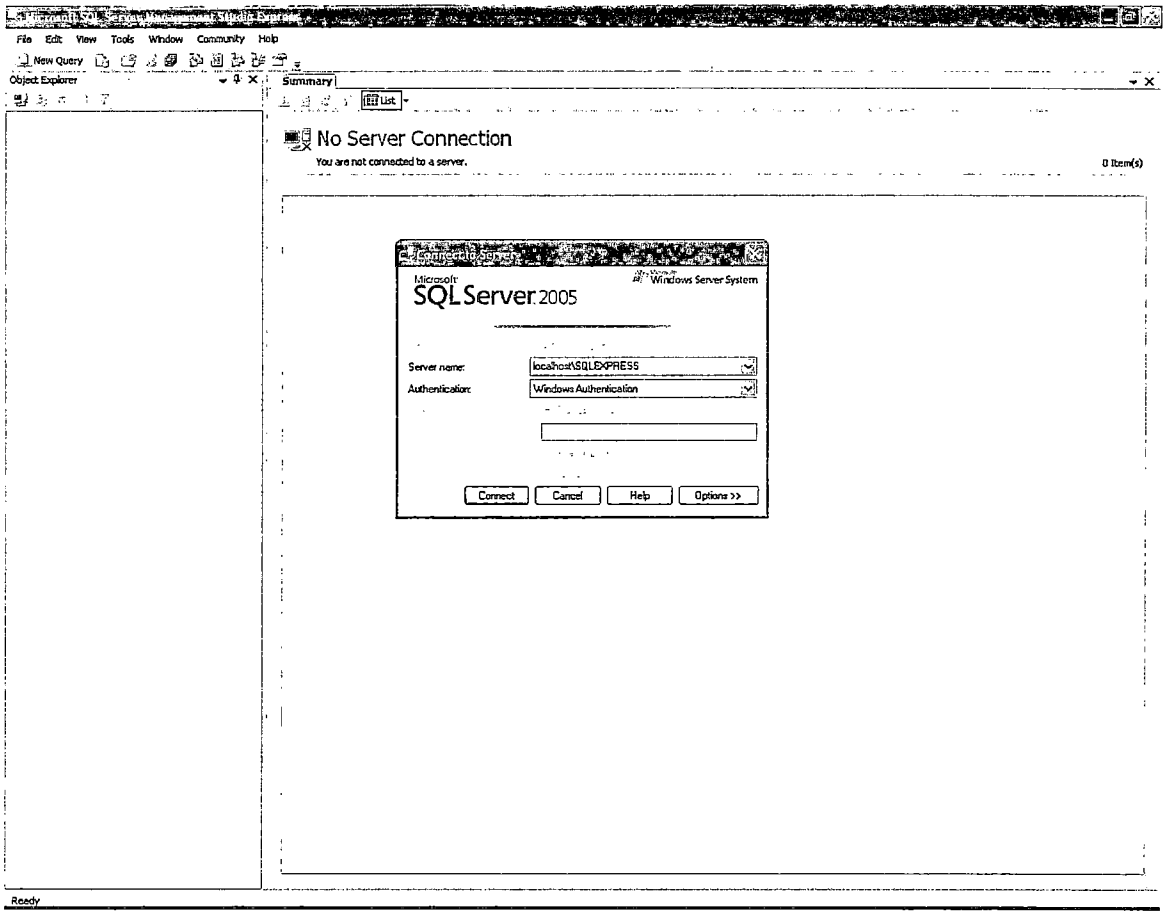



**4.1: Το περιβάλλον του Microsoft Visual Web Developer 2005 Express Edition**

## 4.1.2 SQL Server

Το εργαλείο SQL Server είναι μια πλατφόρμα διαχείρισης βάσεων δεδομένων. Η εταιρία Microsoft διαθέτει δωρεάν το εργαλείο Microsoft SQL Server 2005 Express Edition, που είναι και η τρέχουσα έκδοση του και περιέχει πολλές απ' τις λειτουργίες του SQL Server. Μαζί με το εργαλείο Visual Web Developer, που παρουσιάσαμε παραπάνω αποτελούν την καλύτερη δωρεάν λύση για τη δημιουργία τοποθεσιών ιστού. Ακόμα μπορούμε να ενσωματώσουμε και το εργαλείο SQL Server Management Studio Express, το οποίο είναι ένα γραφικό περιβάλλον διαχείρισης βάσεων δεδομένων και συγγραφής κώδικα SQL. Το Microsoft SQL Server 2005 Express Edition διατίθεται στην ακόλουθη ηλεκτρονική διεύθυνση:

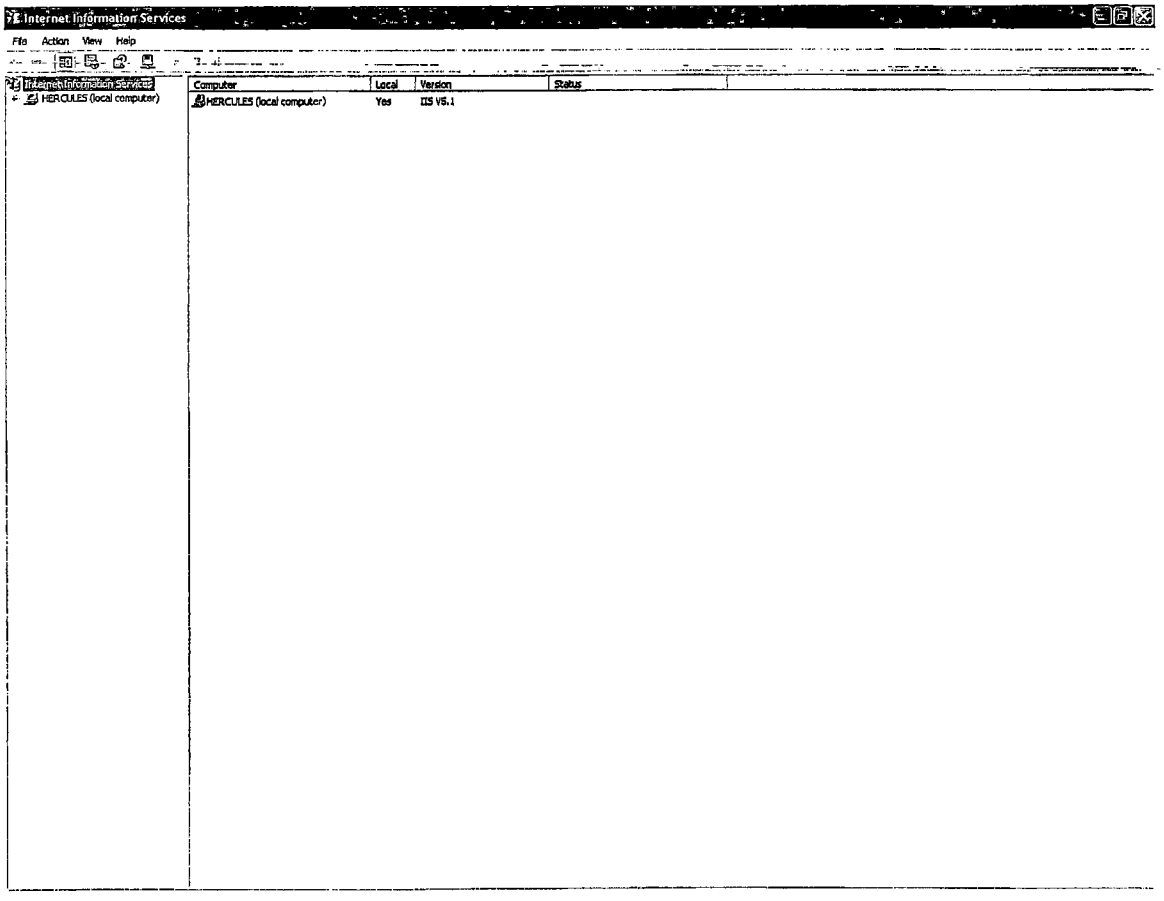
http://msdn2.microsoft.com/en-us/express/aa718378aspx




**4.2 :Το περιβάλλον του Microsoft SQL Server Management Studio Express**

### 4.3.1 Internet Information Services (IIS)

Το εργαλείο Internet Information Services της εταιρίας Microsoft, παρέχει ένα σύνολο υπηρεσιών για εξυπηρετητές (Servers) του διαδικτύου, οι οποίοι χρησιμοποιούν τα Windows της Microsoft ως λειτουργικό σύστημα. Η περιορισμένη έκδοση του IIS 5.1, η οποία χρησιμοποιήθηκε για τη δημιουργία του παρόντος πληροφοριακού συστήματος είναι ενσωματωμένη στο λειτουργικό σύστημα Microsoft Windows XP Professional.




**4.3: Το περιβάλλον του Internet Information Services**

## 4.2 Τεχνολογίες .NET

Οι Τεχνολογίες .NET, οι οποίες παρουσιάζονται παρακάτω είναι οι εξής:

- **.NET Framework**
- **ASP.NET**
- **ADO.NET**
- **Visual C#.NET**

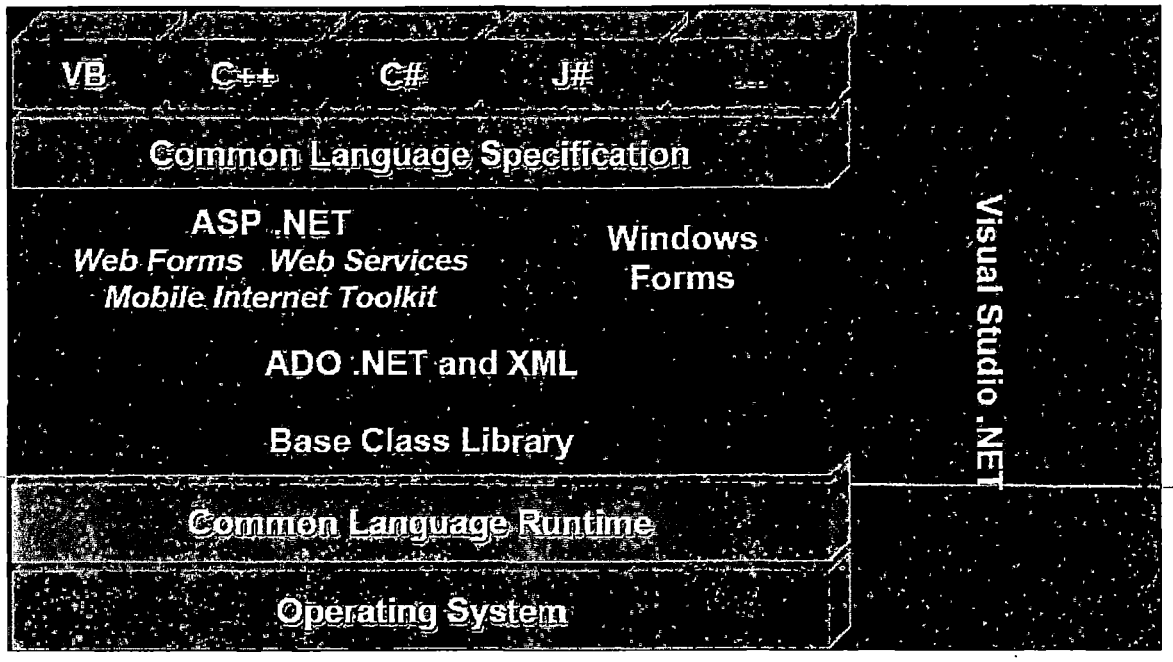
### 4.2.1 .NET Framework

Η τεχνολογία Microsoft .NET Framework είναι η καρδιά της στρατηγικής .NET. Διαχειρίζεται και εκτελεί εφαρμογές και υπηρεσίες διαδικτύου, περιέχει μια βιβλιοθήκη κλάσεων την Framework Class Library (FCL), ενισχύει την ασφάλεια και παρέχει πάρα πολλές προγραμματιστικές δυνατότητες. Οι λεπτομέρειες της τεχνολογίας .NET Framework δίνονται στο Common Language Specification (CLS), που παρέχει πληροφορίες για την αποθήκευση τύπων δεδομένων και των αντικειμένων.

Ο κώδικας των προγραμμάτων μεταγλωτίζεται σε οδηγίες συγκεκριμένες για τη μηχανή στα εξής δύο βήματα:

- **Βήμα 1°:** Ο κώδικας μεταγλωττίζεται σε ενδιάμεσο κώδικα (bytecode), γραμμένο στην γλώσσα Microsoft Intermediate Language (MSIL) και ορίζει οδηγίες για το Common Language Runtime (CLR) που λειτουργεί ως εικονική μηχανή (Virtual Machine).
- **Βήμα 2°:** Μέσω ενός άλλου μεταγλωττιστή στο CLR, ο κώδικας της MSIL μεταγλωττίζεται σε κώδικα μηχανής για μια συγκεκριμένη πλατφόρμα, δημιουργώντας μια εφαρμογή.

Αυτή η αρχιτεκτονική των δύο προηγούμενων βημάτων, παρέχει δυνατότητα μεταφοράς μεταξύ των λειτουργικών συστημάτων, διαλειτουργικότητα ανάμεσα στις γλώσσες προγραμματισμού και έλεγχο στην διαχείριση της μνήμης και της ασφάλειας. Η αρχιτεκτονική της τεχνολογίας .NET Framework παρουσιάζεται παρακάτω:




**4.4: Η Αρχιτεκτονική του .NET Framework**

## 4.2.2 ASP.NET

Η τεχνολογία Microsoft Active Server Pages .NET αποτελείται απ' ένα σύνολο επιμερών τεχνολογιών της .NET Framework για την κατασκευή εφαρμογών διαδικτύου (Web Application) και διαδικτυακών υπηρεσιών (Web Services). Οι ιστοσελίδες ASP.NET εκτελούνται στον διακομιστή Ιστού (Web Server) και δημιουργούν Markups της HTML, της XML κ.α., που στέλνονται σε οποιοδήποτε φύλλομετρητή (Web Browser), υπολογιστή που είναι συνδεδεμένος στο διαδίκτυο (Web Client).

Η τεχνολογία ASP.NET χρησιμοποιεί ένα μοντέλο προγραμματισμού μεταγλωττιζόμενο και γεγονοτοδηγούμενο, το οποίο βελτιώνει την επίδοση και διαχωρίζει το επίπεδο εφαρμογής (Application Layer) και το επίπεδο χρήστη (User Interface).

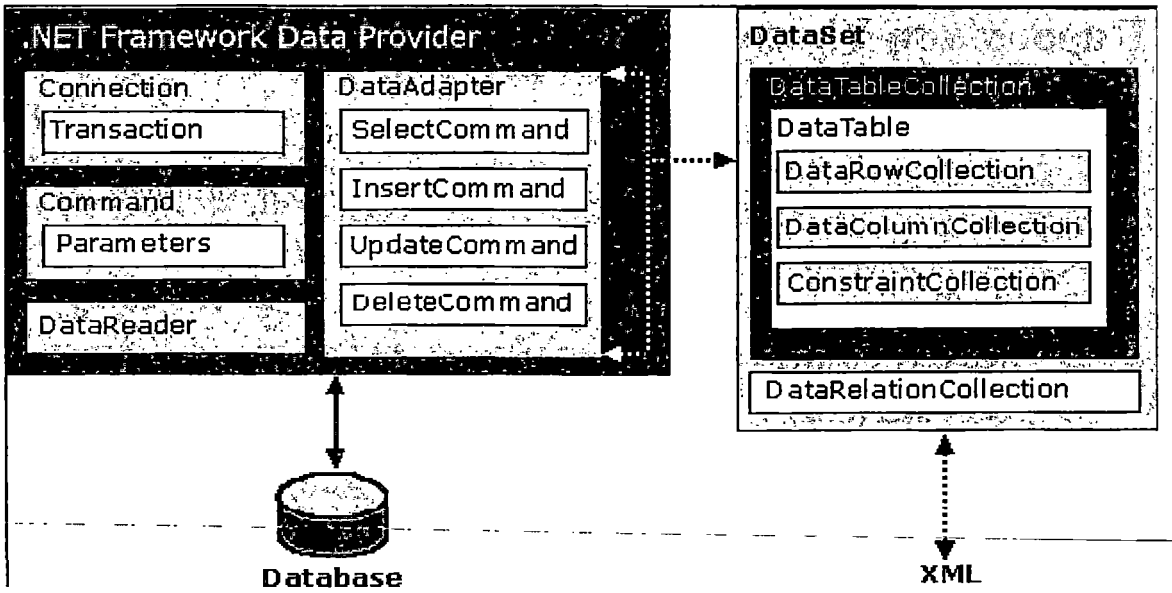
### 4.2.3 ADO.NET

Η τεχνολογία ADO.NET αποτελείται απ' ένα μοντέλο αντικειμένων, το οποίο παρέχει ένα API (Application Programming Interface) για πρόσβαση σε συστήματα βάσεων δεδομένων. Δημιουργήθηκε για το .NET Framework και είναι η επόμενη γενιά των ActiveX Data Objects (ADO), που σχεδιάστηκαν για την αλληλεπίδραση με το Component Object Model (COM) της εταιρίας Microsoft.

Με λίγα λόγια, η τεχνολογία ADO.NET παρέχει ένα επίπεδο μεταξύ των γλωσσών προγραμματισμού ανώτερου επιπέδου και των βάσεων δεδομένων, το οποίο δίνει τη δυνατότητα πρόσβασης στις βάσεις δεδομένων, χωρίς να γνωρίζουμε πως αυτές έχουν υλοποιηθεί. Τα σημαντικότερα αντικείμενα που αποτελούν την τεχνολογία ADO.NET είναι τα εξής:

- Connection: καθορίζει τις ρυθμίσεις για τη σύνδεση σε μια συγκεκριμένη πηγή δεδομένων.
- Command: περιέχει τις εντολές SQL που αποστέλλονται στη βάση δεδομένων.
- DataReader: παρέχει ένα σειριακό τρόπο ανάγνωσης δεδομένων χωρίς τη δυνατότητα τροποποίησης τους και δημιουργείται κατά την εκτέλεση ενός αντικειμένου Command.
- DataSet: αποθηκεύει δεδομένα προσωρινά στην τοπική μνήμη.
- DataAdapter: επιτρέπει τη συμπλήρωση και την ενημέρωση των περιεχομένων ενός αντικειμένου DataSet. Περιέχει τις εξής ιδιότητες τύπου Command, για τη διαχείριση δεδομένων:
  - ✓ DeleteCommand: για την διαγραφή δεδομένων.
  - ✓ InsertCommand: για την εισαγωγή δεδομένων.
  - ✓ SelectCommand: για την ανάκτηση δεδομένων.
  - ✓ UpdateCommand: για την ενημέρωση δεδομένων.

Η αρχιτεκτονική της τεχνολογίας ADO.NET παρουσιάζεται παρακάτω:




**4.5: Η Αρχιτεκτονική του ADO .NET**

## 4.2.4 Visual C#.NET

Η τεχνολογία Visual C#.NET είναι μια αντικείμενοστρεφής γλώσσα προγραμματισμού, που έχει αναπτυχθεί από την εταιρία Microsoft για την πλατφόρμα των τεχνολογιών .NET.

Η Visual C#.NET ανήκει στην ίδια κατηγορία αντικείμενοστρεφών γλωσσών προγραμματισμού με την Java και την C++, συνδυάζοντας την απλότητα της Visual Basic. Στις καινοτομίες που εισάγει εντάσσονται η δυνατότητα Boxing / Unboxing και Versioning και η υποστήριξη του μοντέλου προγραμματισμού που βασίζεται σε Components (Component Basd Programming).

# 4.3 Παρουσίαση Υλοποίησης

Παρακάτω παρατίθεται ο κώδικας της εφαρμογής BalloonShop, χωρισμένος σε κατηγορίες ανάλογα με το τι υλοποίει. Έτσι ο κώδκας διακρίνεται στις εξής κατηγορίες:

- **Master Page**
- **Web Form**
- **Web User Control**
- **Style Sheet (CSS)**
- **Skin File**
- **Class File**
- **Το αρχείο Global.asax**
- **Το αρχείο Web.config**
- **Οι Φάκελοι Images και Product Images**
- **Η Β.Δ ASPNETDB.MDF**
- **Η Β.Δ BalloonShop.dbo**

Τα αρχεία Master Page, Web Form, Web User Control είναι διπλά αρχεία, δηλαδή έχουν ενα ακόμα αρχείο με την ονομασία τους, αλλά με την κατάληξη .cs. Αυτά τα αρχεία είναι Class File και είναι πολυ σημαντικά, καθώς αυτά ''κάνουν ολη την δουλειά'' περιέχουν κώδικα Visual C# που υλοποίει διάφορες λειτουργίες και συμπεριφορές.

Ακόμα στον κώδικα υπάρχουν και σχόλια στα αγγλικά ( // ), για να γίνεται πιο κατανοητός.

## 4.3.1 Master Page

Οι Master Pages είναι ένας εύκολος τρόπος να διατηρήσουμε την ίδια δομή σε ολόκληρη την εφαρμγή μας και να έχουν την ίδια μορφή όλες οι ιστοσελίδες (Web Pages) της. Το μεγάλο πλεονέκτημά τους είναι ότι δεν χρειάζεται να δημιουργήσουμε μια ιστιοσελίδα από την αρχή, αλλά μια ιστιοσελίδα βασισμένη στην Master Page.

Στην εφαρμογή μας έχουμε δύο Master Pages, την **BalloonsShop.master**, που χρησιμοποιείται στις ιστιοσελίδες που διαχειρίζονται οι χρήστες της εφαρμογής και την **Admin.master**, που χρησιμοποιείται στις ιστιοσελίδες που διαχειρίζεται ο διαχειριστής (Administrator) της εφαρμογής.

Στις παρακάτω υποπαραγράφους ακολουθεί ο κώδικας των δύο Master Pages.

### 4.3.1.1 BalloonsShop.master

```
<%@ Master Language="C#" AutoEventWireup="true"
CodeFile="BalloonShop.master.cs"
    Inherits="BalloonShop" %>
<%@ Register Src="UserControls/CartSummary.ascx" TagName="CartSummary"
TagPrefix="uc6" %>
<%@ Register Src="UserControls/UserInfo.ascx" TagName="UserInfo"
TagPrefix="uc5" %>
<%@ Register Src="UserControls/SearchBox.ascx" TagName="SearchBox"
TagPrefix="uc4" %>
<%@ Register Src="UserControls/CategoriesList.ascx"
TagName="CategoriesList" TagPrefix="uc3" %>
<%@ Register Src="UserControls/DepartmentsList.ascx"
TagName="DepartmentsList" TagPrefix="uc2" %>
<%@ Register Src="UserControls/Header.ascx" TagName="Header"
TagPrefix="uc1" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
  <title>BalloonShop</title>
</head>
<body>
  <form id="Form1" runat="server">
    <table cellspacing="0" cellpadding="0" width="770" border="0">
      <tr>
        <td width="220" valign="top">
          <uc5:UserInfo ID="UserInfo1" runat="server" />
          <br />
          <uc2:DepartmentsList ID="DepartmentsList1" runat="server" />
          <br />
          <uc3:CategoriesList ID="CategoriesList1" runat="server" />
          <uc4:SearchBox id="SearchBox1" runat="server">
          </uc4:SearchBox>
            <br />
            <uc6:CartSummary ID="CartSummary1" runat="server" />
        </td>
        <td valign="top">
          <uc1:Header ID="Header1" runat="server" />
          <asp:ContentPlaceHolder ID="contentPlaceHolder"
runat="server">
          </asp:ContentPlaceHolder>
```

```
        </td>
      </tr>
    </table>
  </form>
</body>
</html>
```

### 4.3.1.1.1 BalloonsShop.master.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
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;

public partial class BalloonShop : System.Web.UI.MasterPage
{
    // Website pages considered to be "catalog pages" that the visitor
    // can "Continue Shopping" to
    private static string[] catalogPages = { "~/Default.aspx",
"~/Catalog.aspx", "~/Search.aspx" };

    // Executes when any page based on this master page loads
    protected void Page_Load(object sender, EventArgs e)
    {
        // Don't perform any actions on postback events
        if (!IsPostBack)
        {
            /* Save the latest visited catalog page into the session
              to support "Continue Shopping" functionality */
            // Get the currently loaded page
            string currentLocation =
Request.AppRelativeCurrentExecutionFilePath;
            // If the page is one of those we want the visitor to
"continue shopping"
            // to, then save it to visitor's Session
            for (int i = 0; i < catalogPages.GetLength(0); i++)
                if (String.Compare(catalogPages[i], currentLocation,
true) == 0)
                {
                    // save the current location
                    Session["LastVisitedCatalogPage"] =
Request.Url.ToString();
                    // stop the for loop from continuing
                    break;
                }
        }
```

```
    }
}
```

### 4.3.1.2 Admin.master

```
<%@ Master Language="C#" AutoEventWireup="true"
CodeFile="Admin.master.cs" Inherits="Admin" %>
<%@ Register Src="UserControls/UserInfo.ascx" TagName="UserInfo"
TagPrefix="uc1" %>

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

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
  <form id="form1" runat="server">
    <table cellspacing="0" cellpadding="0" border="0" width="100%">
      <tr valign="top">
        <td width="220">
          <uc1:UserInfo ID="UserInfo1" runat="server" />
        </td>
        <td valign="top">
          <span class="AdminTitle">
            <% Response.Write(BalloonShopConfiguration.SiteName); %>
          </span>
          (<a href="Default.aspx" class="AdminPageText">go back
          to BalloonShop</a>)
          <br />
          <asp:ContentPlaceHolder ID="ContentPlaceHolder1"
runat="server">
          </asp:ContentPlaceHolder>
        </td>
      </tr>
    </table>
    <br />
    <asp:ContentPlaceHolder ID="ContentPlaceHolder2" runat="server">
    </asp:ContentPlaceHolder>
  </form>
</body>
</html>
```

#### 4.3.1.2.1 Admin.master.cs

```
using System;
using System.Data;
using System.Configuration;
```

```
using System.Collections;
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;

public partial class Admin : System.Web.UI.MasterPage
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
}
```

### 4.3.2 Web Form

Οι Web Forms είναι οι φόρμες της εφαρμογή μας και εχούν πολλαπλές λειτουργίες στην εφαρμογή μας. Οι βασικότερες λειτουργίες τους είναι λειτουργίες όπως εισαγωγή νέων προιόντων στην εφαρμογή μας, εγγραφή νέων πελατών, καταχώρηση παρραγγελίας, πληρωμή παρραγγελίας κ.ά.

Παρακάτω ακολουθούν όλες οι Web Forms της εφαρμογής μας, καθώς και οι λειτουργές τους:

- **Catalog.aspx**: όπου εμφανίζονται τα προϊόντα του ηλεκτρονικού καταστήματος.
- **CatalogAdmin.aspx**: όπου εισάγονται ή διαγράφονται προϊόντα από το διαχειριστή του ηλεκτρονικού καταστήματος.
- **Checkout.aspx**: όπου γίνεται η πληρωμή των παραγγελιών
- **CustomerDetails.aspx**: όπου εμφανίζονται οι λεπτομέρειες του πελάτη, όπως τα στοιχεία του και τι έχει επιλέξει να αγοράσει.
- **Default.aspx**: όπου εμφανίζονται προτάσεις για αγορά.
- **Login.aspx**: όπου γίνεται η είσοδος εγγεγραμμένου πελάτη
- **Oooops.aspx**: μήνυμα που εμφανίζεται σε τυχόν λάθος της εφαρμογής.
- **OrdersAdmin.aspx**: όπου γίνεται η διαχείριση των παραγγελιών από τον διαχειριστή του ηλεκτρονικού καταστήματος.

- **Product.aspx**: όπου εμφανίζονται οι λεπτομέρειες ενός προϊόντος.
- **Register.aspx**: όπου γίνεται νέα εγγραφή πελάτη.
- **Search.aspx**: όπου εμφανίζονται τα αποτελέσματα αναζήτησης πελάτη
- **SecurityLibTester.aspx**: τεστ 1$^{ο}$ για την ασφάλεια της εφαρμογής.
- **SecurityLibTester2.aspx**: τεστ 2$^{ο}$ για την ασφάλεια της εφαρμογής.
- **SecurityLibTester3.aspx**: τεστ 3$^{ο}$ για την ασφάλεια της εφαρμογής.
- **ShoppingCart.aspx**: όπου εμφανίζεται το καλάθι αγοράς ενός πελάτη.
- **ShoppingCartAdmin.aspx**: όπου γίνεται η διαχείριση των καλαθιών αγορών από το διαχειριστή του ηλεκτρονικού καταστήματος.

Στις παρακάτω υποπαράγραφος ακολουθεί ο κώδικας των Web Forms.

### 4.3.2.1 Catalog.aspx

```
<%@ Page Language="C#" MasterPageFile="~/BalloonShop.master"
AutoEventWireup="true" CodeFile="Catalog.aspx.cs" Inherits="Catalog"
Title="BalloonShop - The Product Catalog" %>
<%@ Register Src="UserControls/ProductsList.ascx"
TagName="ProductsList" TagPrefix="uc1" %>
<asp:Content ID="content" ContentPlaceHolderID="contentPlaceHolder"
Runat="Server">
  <asp:Label ID="catalogTitleLabel" CssClass="CatalogTitle"
Runat="server" />
  <br />
  <asp:Label ID="catalogDescriptionLabel" CssClass="CatalogDescription"
Runat="server" />
  <br />
  <uc1:ProductsList ID="ProductsList1" runat="server" />
</asp:Content>
```

### 4.3.2.1.1 Catalog.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
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;

public partial class Catalog : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        // don't reload data during postbacks
        if (!IsPostBack)
        {
            PopulateControls();
        }
    }

    // Fill the page with data
    private void PopulateControls()
    {
        // Retrieve DepartmentID from the query string
        string departmentId = Request.QueryString["DepartmentID"];
        // Retrieve CategoryID from the query string
        string categoryId = Request.QueryString["CategoryID"];
        // If browsing a category...
        if (categoryId != null)
        {
            // Retrieve category details and display them
            CategoryDetails cd =
CatalogAccess.GetCategoryDetails(categoryId);
            catalogTitleLabel.Text = cd.Name;
            catalogDescriptionLabel.Text = cd.Description;
            // Set the title of the page
            this.Title = BalloonShopConfiguration.SiteName +
                         " : Category : " + cd.Name;
        }
        // If browsing a department...
        else if (departmentId != null)
        {
            // Retrieve department details and display them
            DepartmentDetails dd =
CatalogAccess.GetDepartmentDetails(departmentId);
            catalogTitleLabel.Text = dd.Name;
            catalogDescriptionLabel.Text = dd.Description;
            // Set the title of the page
            this.Title = BalloonShopConfiguration.SiteName +
                         " : Department : " + dd.Name;
        }
    }
}
```

## 4.3.2.2 CatalogAdmin.aspx

```
<%@ Page Language="C#" MasterPageFile="~/Admin.master"
AutoEventWireup="true" CodeFile="CatalogAdmin.aspx.cs"
Inherits="CatalogAdmin" Title="Untitled Page" %>
```

```
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1"
Runat="Server">
    <span class="AdminTitle">Catalog Admin </span>
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder2"
Runat="Server">
    <asp:PlaceHolder ID="adminPlaceHolder"
runat="server"></asp:PlaceHolder>
</asp:Content>
```

### 4.3.2.2.1 CatalogAdmin.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
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;

public partial class CatalogAdmin : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        // Set the title of the page
        this.Title = BalloonShopConfiguration.SiteName + " : Catalog
Admin";
        // Get DepartmentID from the query string
        string departmentId = Request.QueryString["DepartmentID"];
        // Get CategoryID from the query string
        string categoryId = Request.QueryString["CategoryID"];
        // Get ProductID from the query string
        string productId = Request.QueryString["ProductID"];
        // Load the appropriate control into the place holder
        if (departmentId == null)
        {
            Control c = Page.LoadControl(Request.ApplicationPath +
"/UserControls/DepartmentsAdmin.ascx");
            adminPlaceHolder.Controls.Add(c);
        }
        else if (categoryId == null)
        {
            Control c = Page.LoadControl(Request.ApplicationPath +
"/UserControls/CategoriesAdmin.ascx");
            adminPlaceHolder.Controls.Add(c);
        }
        else if (productId == null)
        {
```

```
            Control c = Page.LoadControl(Request.ApplicationPath +
"/UserControls/ProductsAdmin.ascx");
            adminPlaceHolder.Controls.Add(c);
        }
        else
        {
            Control c = Page.LoadControl(Request.ApplicationPath +
"/UserControls/ProductDetailsAdmin.ascx");
            adminPlaceHolder.Controls.Add(c);
        }
    }
}
```

### 4.3.2.3 Checkout.aspx

```
<%@ Page Language="C#" MasterPageFile="~/BalloonShop.master"
AutoEventWireup="true"
  CodeFile="Checkout.aspx.cs" Inherits="Checkout" Title="BalloonShop :
Checkout" %>

<%@ Register TagPrefix="uc1" TagName="CustomerDetailsEdit"
Src="UserControls/CustomerDetailsEdit.ascx" %>
<asp:Content ID="Content1" ContentPlaceHolderID="contentPlaceHolder"
runat="Server">
  <asp:Label ID="titleLabel" runat="server"
CssClass="ShoppingCartTitle" Text="Your Shopping Cart" /> <br />
  <br />
  <asp:GridView ID="grid" runat="server" AutoGenerateColumns="False"
DataKeyNames="ProductID"
    BorderWidth="1px" Width="100%">
    <Columns>
      <asp:BoundField DataField="Name" HeaderText="Product Name"
ReadOnly="True" SortExpression="Name" />
      <asp:BoundField DataField="Price" DataFormatString="{0:c}"
HeaderText="Price" ReadOnly="True"
        SortExpression="Price" />
      <asp:BoundField DataField="Quantity" HeaderText="Quantity"
ReadOnly="True" SortExpression="Quantity" />
      <asp:BoundField DataField="Subtotal" DataFormatString="{0:c}"
HeaderText="Subtotal"
        ReadOnly="True" SortExpression="Subtotal" />
    </Columns>
  </asp:GridView>
  <asp:Label ID="Label2" runat="server" Text="Total amount: "
CssClass="ProductDescription"></asp:Label>
  <asp:Label ID="totalAmountLabel" runat="server" Text="Label"
CssClass="ProductPrice"></asp:Label>
  <br />
  <br />
  <uc1:CustomerDetailsEdit ID="CustomerDetailsEdit1" runat="server"
Editable="false"
    Title="User Details" />
```

```
  <br />
  <asp:Label ID="InfoLabel" runat="server" CssClass="InfoText" />
  <br />
  <br />
  <asp:Button ID="placeOrderButton" runat="server"
CssClass="ButtonText" Text="Place order"
    OnClick="placeOrderButton_Click" />
</asp:Content>
```

### 4.3.2.3.1 Checkout.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
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;

public partial class Checkout : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        // Set the title of the page
        this.Title = BalloonShopConfiguration.SiteName +
                     " : Checkout";

        if (!IsPostBack)
            PopulateControls();
    }

    // fill controls with data
    private void PopulateControls()
    {
        // get the items in the shopping cart
        DataTable dt = ShoppingCartAccess.GetItems();
        // populate the list with the shopping cart contents
        grid.DataSource = dt;
        grid.DataBind();
        // setup controls
        titleLabel.Text =
          "These are the products in your shopping cart:";
        grid.Visible = true;
        // display the total amount
        decimal amount = ShoppingCartAccess.GetTotalAmount();
        totalAmountLabel.Text = String.Format("{0:c}", amount);

        // check customer details
        bool addressOK = true;
```

```
        bool cardOK = true;
        if (Profile.Address1 + Profile.Address2 == ""
          || Profile.ShippingRegion == ""
          || Profile.ShippingRegion == "1"
          || Profile.Country == "")
        {
            addressOK = false;
        }
        if (Profile.CreditCard == "")
        {
            cardOK = false;
        }

        // report / hide place order button
        if (!addressOK)
        {
            if (!cardOK)
            {
                InfoLabel.Text =
                  "You must provide a valid address and credit card "
                  + "before placing your order.";
            }
            else
            {
                InfoLabel.Text =
                  "You must provide a valid address before placing your
"
                  + "order.";
            }
        }
        else if (!cardOK)
        {
            InfoLabel.Text = "You must provide a credit card before "
              + "placing your order.";
        }
        else
        {
            InfoLabel.Text = "Please confirm that the above details are
"
              + "correct before proceeding.";
        }
        placeOrderButton.Visible = addressOK && cardOK;
    }

    protected void placeOrderButton_Click(object sender, EventArgs e)
    {
        // Store the total amount because the cart
        // is emptied when creating the order
        decimal amount = ShoppingCartAccess.GetTotalAmount();
        // Create the order and store the order ID
        string orderId = ShoppingCartAccess.CreateOrder();
        // Create the PayPal redirect location
        string redirect = "";
        redirect +=

"https://www.paypal.com/xclick/business=youremail@server.com";
```

```
        redirect += "&item_name=BallonShopOrder " + orderId;
        redirect += "&item_number=" + orderId;
        redirect += "&amount=" + String.Format("{0:c} ", amount);
        redirect += "&return=http://www.YourWebSite.com";
        redirect += "&cancel_return=http://www.YourWebSite.com";
        // Redirect to the payment page
        Response.Redirect(redirect);
    }
}
```

## 4.3.2.4 CustomerDetails.aspx

```
<%@ Page Language="C#" MasterPageFile="~/BalloonShop.master"
AutoEventWireup="true"
  CodeFile="CustomerDetails.aspx.cs" Inherits="CustomerDetails" %>

<%@ Register TagPrefix="uc1" TagName="CustomerDetailsEdit"
  Src="UserControls/CustomerDetailsEdit.ascx" %>
<asp:Content ID="Content1" ContentPlaceHolderID="contentPlaceHolder"
runat="Server">
  <uc1:customerdetailsedit id="CustomerDetailsEdit1" runat="server" />
</asp:Content>
```

## 4.3.2.4.1 CustomerDetails.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
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;

public partial class CustomerDetails : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        // Set the title of the page
        this.Title = BalloonShopConfiguration.SiteName +
                     " : Customer Details";
    }
}
```

### 4.3.2.5 Default.aspx

```
<%@ Page Language="C#" MasterPageFile="~/BalloonShop.master"
AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default"
Title="Welcome to BalloonShop!" %>
<%@ Register Src="UserControls/ProductsList.ascx"
TagName="ProductsList" TagPrefix="uc1" %>
<asp:Content ID="content" ContentPlaceHolderID="contentPlaceHolder"
Runat="server">
  <span class="CatalogTitle">Welcome to BalloonShop! </span>
  <br />
  <span class="CatalogDescription">This week we have a special price
for these fantastic products: </span>
  <br />
  <uc1:ProductsList ID="ProductsList1" runat="server" />
</asp:Content>
```

### 4.3.2.5.1 Default.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
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;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
}
```

### 4.3.2.6 Login.aspx

```
<%@ Page Language="C#" MasterPageFile="~/BalloonShop.master"
AutoEventWireup="true"
  CodeFile="Login.aspx.cs" Inherits="Login" %>

<asp:Content ID="Content1" ContentPlaceHolderID="contentPlaceHolder"
runat="Server">
```

```
  <asp:Login ID="login" runat="server">
    <LayoutTemplate>
      <table border="0" cellpadding="1">
        <tr class="UserInfoText">
          <td>
            <table border="0" cellpadding="0">
              <tr>
                <td class="CatalogTitle" align="left" colspan="2">
                  Who Are You?<br />
                  <br />
                </td>
              </tr>
              <tr>
                <td align="right">
                  <asp:Label ID="UserNameLabel" runat="server"
AssociatedControlID="UserName">User Name:</asp:Label></td>
                <td>
                  <asp:TextBox ID="UserName"
runat="server"></asp:TextBox>
                  <asp:RequiredFieldValidator ID="UserNameRequired"
runat="server" ControlToValidate="UserName"
                    ErrorMessage="User Name is required." ToolTip="User
Name is required."
ValidationGroup="Login1">*</asp:RequiredFieldValidator>
                </td>
              </tr>
              <tr>
                <td align="right">
                  <asp:Label ID="PasswordLabel" runat="server"
AssociatedControlID="Password">Password:</asp:Label></td>
                <td>
                  <asp:TextBox ID="Password" runat="server"
TextMode="Password"></asp:TextBox>
                  <asp:RequiredFieldValidator ID="PasswordRequired"
runat="server" ControlToValidate="Password"
                    ErrorMessage="Password is required."
ToolTip="Password is required."
ValidationGroup="Login1">*</asp:RequiredFieldValidator>
                </td>
              </tr>
              <tr>
                <td colspan="2">
                  <asp:CheckBox ID="RememberMe" runat="server"
Text="Remember me next time." />
                </td>
              </tr>
              <tr>
                <td align="center" colspan="2" style="color: red">
                  <asp:Literal ID="FailureText" runat="server"
EnableViewState="False"></asp:Literal>
                </td>
              </tr>
              <tr>
                <td align="right" colspan="2">
                  <asp:Button ID="LoginButton" runat="server"
CommandName="Login" Text="Log In" ValidationGroup="Login1" />
```

```
                </td>
              </tr>
            </table>
          </td>
        </tr>
      </table>
      <span class="InfoText">You must be logged in to place an order.
If you aren't yet
        registered with the site, click
        <asp:HyperLink runat="server" ID="registerLink"
NavigateUrl="~/Register.aspx" Text="here"
          ToolTip="Go to the registration page" CssClass="UserInfoLink"
/>. </span>
    </LayoutTemplate>
  </asp:Login>
</asp:Content>
```

### 4.3.2.6.1 Login.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
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;

public partial class Login : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        // get references to the button, checkbox and textboxes
        TextBox usernameTextBox =
(TextBox)login.FindControl("UserName");
        TextBox passwordTextBox =
(TextBox)login.FindControl("Password");
        CheckBox persistCheckBox =
(CheckBox)login.FindControl("RememberMe");
        Button loginButton = (Button)login.FindControl("LoginButton");
        // tie the two textboxes and the checkbox to the button
        Utilities.TieButton(this.Page, usernameTextBox, loginButton);
        Utilities.TieButton(this.Page, passwordTextBox, loginButton);
        Utilities.TieButton(this.Page, persistCheckBox, loginButton);
        // set the page title
        this.Title = BalloonShopConfiguration.SiteName + " : Login";
        // set focus on the username textbox when the page loads
        usernameTextBox.Focus();
    }
```

```
}
```

### 4.3.2.7 Oooops.aspx

```
<%@ Page Language="C#" MasterPageFile="~/BalloonShop.master"
AutoEventWireup="true" CodeFile="Oooops.aspx.cs" Inherits="Oooops"
Title="BalloonShop - Oooops!" %>
<asp:Content ID="Content1" ContentPlaceHolderID="contentPlaceHolder"
Runat="Server">
<p align="center">
  <span class="CatalogTitle">
    Your request generated an internal error!
  </span>
  <br /><br />
  <span class="CatalogDescription">
    We apologize for the inconvenience! The error has been reported.
  </span>
</p>
</asp:Content>
```

### 4.3.2.7.1 Oooops.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
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;

public partial class Oooops : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
}
```

### 4.3.2.8 OrdersAdmin.aspx

```
<%@ Page Language="C#" MasterPageFile="~/Admin.master"
AutoEventWireup="true" CodeFile="OrdersAdmin.aspx.cs"
  Inherits="OrdersAdmin" Title="Untitled Page" %>

<%@ Register Src="UserControls/OrderDetailsAdmin.ascx"
TagName="OrderDetailsAdmin"
  TagPrefix="uc1" %>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1"
runat="Server">
  <span class="AdminTitle">Orders Admin</span></asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder2"
runat="Server">
  <span class="AdminPageText">Show the most recent
    <asp:TextBox ID="recentCountTextBox" runat="server" MaxLength="4"
Width="40px" Text="20" />
    records
    <asp:Button ID="byRecentGo" runat="server"
CssClass="SmallButtonText" Text="Go" OnClick="byRecentGo_Click"
      CausesValidation="False" /><br />
    Show all records created between
    <asp:TextBox ID="startDateTextBox" runat="server" Width="72px" />
    and
    <asp:TextBox ID="endDateTextBox" runat="server" Width="72px" />
    <asp:Button ID="byDateGo" runat="server" CssClass="SmallButtonText"
Text="Go" OnClick="byDateGo_Click" />
    <br />
    Show all unverified, uncanceled orders
    <asp:Button ID="unverfiedGo" runat="server"
CssClass="SmallButtonText" Text="Go"
      OnClick="unverfiedGo_Click" CausesValidation="False" />
    <br />
    Show all verified, uncompleted orders
    <asp:Button ID="uncompletedGo" runat="server"
CssClass="SmallButtonText" Text="Go"
      OnClick="uncompletedGo_Click" CausesValidation="False" />
    <br />
    <asp:Label ID="errorLabel" runat="server" CssClass="AdminErrorText"
EnableViewState="False"></asp:Label>
    <asp:RangeValidator ID="startDateValidator" runat="server"
ControlToValidate="startDateTextBox"
      Display="None" ErrorMessage="Invalid start date"
MaximumValue="1/1/2009" MinimumValue="1/1/1999"
      Type="Date"></asp:RangeValidator>
    <asp:RangeValidator ID="endDateValidator" runat="server"
ControlToValidate="endDateTextBox"
      Display="None" ErrorMessage="Invalid end date"
MaximumValue="1/1/2009" MinimumValue="1/1/1999"
      Type="Date"></asp:RangeValidator>
    <asp:CompareValidator ID="compareDatesValidator" runat="server"
ControlToCompare="endDateTextBox"
      ControlToValidate="startDateTextBox" Display="None"
ErrorMessage="Start date should be more recent than end date"
      Operator="LessThan" Type="Date"></asp:CompareValidator><br />
```

```
    <asp:ValidationSummary ID="ValidationSummary1" runat="server"
CssClass="AdminErrorText"
      HeaderText="Please fix these errors before submitting requests:"
/>
    <br />
    <asp:GridView ID="grid" runat="server" AutoGenerateColumns="False"
DataKeyNames="OrderID"
OnSelectedIndexChanged="grid_SelectedIndexChanged">
      <Columns>
        <asp:BoundField DataField="OrderID" HeaderText="Order ID"
ReadOnly="True" SortExpression="OrderID" />
        <asp:BoundField DataField="DateCreated" HeaderText="Date
Created" ReadOnly="True"
          SortExpression="DateCreated" />
        <asp:BoundField DataField="DateShipped" HeaderText="Date
Shipped" ReadOnly="True"
          SortExpression="DateShipped" />
        <asp:CheckBoxField DataField="Verified" HeaderText="Verified"
ReadOnly="True" SortExpression="Verified" />
        <asp:CheckBoxField DataField="Completed" HeaderText="Completed"
ReadOnly="True" SortExpression="Completed" />
        <asp:CheckBoxField DataField="Canceled" HeaderText="Canceled"
ReadOnly="True" SortExpression="Canceled" />
        <asp:BoundField DataField="CustomerName" HeaderText="Customer
Name" ReadOnly="True"
          SortExpression="CustomerName" />
        <asp:ButtonField ButtonType="Button" CommandName="Select"
Text="Select" />
      </Columns>
    </asp:GridView>
    <br />
    <uc1:OrderDetailsAdmin EnableViewState="false"
id="orderDetailsAdmin" runat="server">
    </uc1:OrderDetailsAdmin>
  </span>
</asp:Content>
```

## 4.3.2.8.1 OrdersAdmin.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
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;

public partial class OrdersAdmin : System.Web.UI.Page
{
```

```
    protected void Page_Load(object sender, EventArgs e)
    {
        // Set the title of the page
        this.Title = BalloonShopConfiguration.SiteName +
                     " : Orders Admin";
        // associate the check boxes with their buttons
        Utilities.TieButton(this.Page, recentCountTextBox, byRecentGo);
        Utilities.TieButton(this.Page, startDateTextBox, byDateGo);
        Utilities.TieButton(this.Page, endDateTextBox, byDateGo);
    }

    // list the most recent orders
    protected void byRecentGo_Click(object sender, EventArgs e)
    {
        // how many orders to list?
        int recordCount;
        // load the new data into the grid
        if (int.TryParse(recentCountTextBox.Text, out recordCount))
            grid.DataSource = OrdersAccess.GetByRecent(recordCount);
        else
            errorLabel.Text = "<br />Please enter a valid number!";
        // refresh the data grid
        grid.DataBind();
        // no order is selected
        Session["AdminOrderID"] = null;
    }

    // list the orders that happened between specified dates
    protected void byDateGo_Click(object sender, EventArgs e)
    {
        // check if the page is valid (we have date validator controls)
        if ((Page.IsValid) && (startDateTextBox.Text +
endDateTextBox.Text != ""))
        {
            // get the dates
            string startDate = startDateTextBox.Text;
            string endDate = endDateTextBox.Text;
            // load the grid with the requested data
            grid.DataSource = OrdersAccess.GetByDate(startDate,
endDate);
        }
        else
            errorLabel.Text = "<br />Please enter valid dates!";
        // refresh the data grid
        grid.DataBind();
        // no order is selected
        Session["AdminOrderID"] = null;
    }

    // get unverified, uncanceled orders
    protected void unverfiedGo_Click(object sender, EventArgs e)
    {
        // load the grid with the requested data
        grid.DataSource = OrdersAccess.GetUnverifiedUncanceled();
        // refresh the data grid
        grid.DataBind();
```

```
        // no order is selected
        Session["AdminOrderID"] = null;
    }

    // get verified, but uncompleted orders
    protected void uncompletedGo_Click(object sender, EventArgs e)
    {
        // load the grid with the requested data
        grid.DataSource = OrdersAccess.GetVerifiedUncompleted();
        // refresh the data grid
        grid.DataBind();
        // no order is selected
        Session["AdminOrderID"] = null;
    }

    // Load the details of the selected order
    protected void grid_SelectedIndexChanged(object sender, EventArgs
e)
    {
        // Save the ID of the selected order in the session
        Session["AdminOrderID"] =
grid.DataKeys[grid.SelectedIndex].Value.ToString();
    }
}
```

## 4.3.2.9 Product.aspx

```
<%@ Page Language="C#" MasterPageFile="~/BalloonShop.master"
AutoEventWireup="true"
  CodeFile="Product.aspx.cs" Inherits="Product" Title="Untitled Page"
%>

<%@ Register Src="UserControls/ProductRecommendations.ascx"
TagName="ProductRecommendations"
    TagPrefix="uc1" %>
<asp:Content ID="content" ContentPlaceHolderID="contentPlaceHolder"
runat="Server">
  <br />
  <asp:Label CssClass="ProductTitle" ID="titleLabel" runat="server"
Text="Label"></asp:Label>
  <br />
  <br />
  <asp:Image ID="productImage" runat="server" />
  <br />
  <asp:Label CssClass="ProductDescription" ID="descriptionLabel"
runat="server" Text="Label"></asp:Label>
  <br />
  <br />
  <span class="ProductDescription">Price:</span> 
  <asp:Label CssClass="ProductPrice" ID="priceLabel" runat="server"
Text="Label" />
  <br />
```

```
  <asp:Button ID="addToCartButton" runat="server" Text="Add to Cart"
CssClass="SmallButtonText" OnClick="addToCartButton_Click" />
  <asp:Button ID="continueShoppingButton" CssClass="SmallButtonText"
runat="server" Text="Continue Shopping"
OnClick="continueShoppingButton_Click" /><br />
    <br />
    <uc1:ProductRecommendations ID="ProductRecommendations1"
runat="server" />
  </asp:Content>
```

## 4.3.2.9.1 Product.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
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;

public partial class Product : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        // don't reload data during postbacks
        if (!IsPostBack)
        {
            PopulateControls();
        }
    }

    // Fill the control with data
    private void PopulateControls()
    {
        // Retrieve ProductID from the query string
        string productId = Request.QueryString["ProductID"];
        // stores product details
        ProductDetails pd;
        // Retrieve product details
        pd = CatalogAccess.GetProductDetails(productId);
        // Display product details
        titleLabel.Text = pd.Name;
        descriptionLabel.Text = pd.Description;
        priceLabel.Text = String.Format("{0:c}", pd.Price);
        productImage.ImageUrl = "ProductImages/" + pd.Image2FileName;
        // Set the title of the page
        this.Title = BalloonShopConfiguration.SiteName +
                     " : Product : " + pd.Name;
    }
```

```
    // Add the product to cart
    protected void addToCartButton_Click(object sender, EventArgs e)
    {
        // Retrieve ProductID from the query string
        string productId = Request.QueryString["ProductID"];
        // Add the product to the shopping cart
        ShoppingCartAccess.AddItem(productId);
    }

    // Redirects to the previously visited catalog page
    protected void continueShoppingButton_Click(object sender,
EventArgs e)
    {
        // redirect to the last visited catalog page
        object page;
        if ((page = Session["LastVisitedCatalogPage"]) != null)
            Response.Redirect(page.ToString());
        else
            Response.Redirect(Request.ApplicationPath);
    }
}
```

## 4.3.2.10 Register.aspx

```
<%@ Page Language="C#" MasterPageFile="~/BalloonShop.master"
AutoEventWireup="true" CodeFile="Register.aspx.cs" Inherits="Register"
%>
<asp:Content ID="Content1" ContentPlaceHolderID="contentPlaceHolder"
Runat="Server">
  <asp:LoginView ID="LoginView1" runat="server">
    <LoggedInTemplate>
      <span class="CodeInline"><span lang="EN-US" style="font-size:
10pt; color: #008000;
        font-family: Courier New; mso-fareast-font-family: 'Times New
Roman'; mso-bidi-font-family: 'Times New Roman';
        mso-ansi-language: EN-US; mso-fareast-language: EN-US; mso-
bidi-language: AR-SA">
        You are already registered.</span></span>
    </LoggedInTemplate>
    <AnonymousTemplate>
      <asp:CreateUserWizard ID="CreateUserWizard1" runat="server"
BackColor="#F7F6F3" BorderColor="#E6E2D8"
        BorderStyle="Solid" BorderWidth="1px"
CancelDestinationPageUrl="~/Default.aspx"
        ContinueDestinationPageUrl="~/CustomerDetails.aspx"
CreateUserButtonText="Sign Up" Font-Names="Verdana"
        Font-Size="0.8em" OnCreatedUser="CreateUserWizard1_CreatedUser"
PasswordRegularExpressionErrorMessage="Your password must be at least 6
characters long.">
        <SideBarStyle BackColor="#5D7B9D" BorderWidth="0px" Font-
Size="0.9em" VerticalAlign="Top" />
```

```
        <SideBarButtonStyle BorderWidth="0px" Font-Names="Verdana"
ForeColor="White" />
        <NavigationButtonStyle BackColor="#FFFBFF"
BorderColor="#CCCCCC" BorderStyle="Solid"
          BorderWidth="1px" Font-Names="Verdana" ForeColor="#284775" />
        <HeaderStyle BackColor="#5D7B9D" BorderStyle="Solid" Font-
Bold="True" Font-Size="0.9em"
          ForeColor="White" HorizontalAlign="Left" />
        <CreateUserButtonStyle BackColor="#FFFBFF"
BorderColor="#CCCCCC" BorderStyle="Solid"
          BorderWidth="1px" Font-Names="Verdana" ForeColor="#284775" />
        <ContinueButtonStyle BackColor="#FFFBFF" BorderColor="#CCCCCC"
BorderStyle="Solid"
          BorderWidth="1px" Font-Names="Verdana" ForeColor="#284775" />
        <StepStyle BorderWidth="0px" />
        <TitleTextStyle BackColor="#5D7B9D" Font-Bold="True"
ForeColor="White" />
        <WizardSteps>
          <asp:CreateUserWizardStep ID="CreateUserWizardStep1"
runat="server">
          </asp:CreateUserWizardStep>
          <asp:CompleteWizardStep ID="CompleteWizardStep1"
runat="server">
          </asp:CompleteWizardStep>
        </WizardSteps>
      </asp:CreateUserWizard>
    </AnonymousTemplate>
  </asp:LoginView>
</asp:Content>
```

## 4.3.2.10.1 Register.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
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;

public partial class Register : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        // Set the title of the page
        this.Title = BalloonShopConfiguration.SiteName +
                     " : Register";
    }
```

```
    protected void CreateUserWizard1_CreatedUser(object sender,
EventArgs e)
    {
        Roles.AddUserToRole((sender as CreateUserWizard).UserName,
            "Customers");
    }
}
```

## 4.3.2.11 Search.aspx

```
<%@ Page Language="C#" MasterPageFile="~/BalloonShop.master"
AutoEventWireup="true" CodeFile="Search.aspx.cs" Inherits="Search"
Title="Untitled Page" %>

<%@ Register Src="UserControls/ProductsList.ascx"
TagName="ProductsList" TagPrefix="uc1" %>
<asp:Content ID="Content1" ContentPlaceHolderID="contentPlaceHolder"
Runat="Server">
  <asp:Label ID="titleLabel" runat="server"
CssClass="CatalogTitle"></asp:Label><br />
  <asp:Label ID="descriptionLabel" runat="server"
CssClass="CatalogDescription"></asp:Label><br /><br />
  <uc1:productslist id="ProductsList1"
runat="server"></uc1:productslist>
</asp:Content>
```

## 4.3.2.11.1 Search.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
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;

public partial class Search : System.Web.UI.Page
{
    // Fill the form with data
    protected void Page_Load(object sender, EventArgs e)
    {
        // don't reload data during postbacks
        if (!IsPostBack)
        {
```

```
            // fill the table contents
            string searchString = Request.QueryString["Search"];
            titleLabel.Text = "Product Search";
            descriptionLabel.Text = "You searched for <font
color=\"red\">" + searchString + "</font>.";
            // set the title of the page
            this.Title = BalloonShopConfiguration.SiteName +
                         " : Product Search : " + searchString;
        }
    }
}
```

## 4.3.2.12 SecurityLibTester.aspx

```
<%@ Page Language="C#" MasterPageFile="~/BalloonShop.master"
AutoEventWireup="true"
  CodeFile="SecurityLibTester.aspx.cs" Inherits="SecurityLibTester"
Title="SecurityLib Test Page" %>

<asp:Content ID="Content1" ContentPlaceHolderID="contentPlaceHolder"
runat="Server">
  Enter your password:<br />
  <asp:TextBox ID="pwdBox1" runat="server" />
  <br />
  Enter your password again:<br />
  <asp:TextBox ID="pwdBox2" runat="server" />
  <br />
  <asp:Button ID="processButton" runat="server" Text="Process"
OnClick="processButton_Click" />
  <br />
  <asp:Label ID="result" runat="server" />
</asp:Content>
```

## 4.3.2.12.1 SecurityLibTester.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
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.Text;
using SecurityLib;

public partial class SecurityLibTester : System.Web.UI.Page
```

```
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void processButton_Click(object sender, EventArgs e)
    {
        string hash1 = PasswordHasher.Hash(pwdBox1.Text);
        string hash2 = PasswordHasher.Hash(pwdBox2.Text);
        StringBuilder sb = new StringBuilder();
        sb.Append("The hash of the first password is: ");
        sb.Append(hash1);
        sb.Append("<br />The hash of the second password is: ");
        sb.Append(hash2);
        if (hash1 == hash2)
        {
            sb.Append("<br />The passwords match! Welcome!");
        }
        else
        {
            sb.Append("<br />Password invalid. "
              + "Armed guards are on their way.");
        }
        result.Text = sb.ToString();
    }
}
```

## 4.3.2.13 SecurityLibTester2.aspx

```
<%@ Page Language="C#" MasterPageFile="~/BalloonShop.master"
AutoEventWireup="true"
  CodeFile="SecurityLibTester2.aspx.cs" Inherits="SecurityLibTester2"
Title="SecurityLib Test Page 2" %>

<asp:Content ID="Content1" ContentPlaceHolderID="contentPlaceHolder"
runat="Server">
  Enter data to encrypt:<br />
  <asp:TextBox ID="encryptBox" runat="server" />
  <br />
  Enter data to decrypt:<br />
  <asp:TextBox ID="decryptBox" runat="server" />
  <br />
  <asp:Button ID="processButton" runat="server" Text="Process"
OnClick="processButton_Click" />
  <br />
  <asp:Label ID="result" runat="server" />
</asp:Content>
```

### 4.3.2.13.1 SecurityLibTester2.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
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.Text;
using SecurityLib;

public partial class SecurityLibTester2 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void processButton_Click(object sender, EventArgs e)
    {
        string stringToEncrypt = encryptBox.Text;
        string stringToDecrypt = decryptBox.Text;
        string encryptedString =
          StringEncryptor.Encrypt(stringToEncrypt);
        if (stringToDecrypt == "")
        {
            stringToDecrypt = encryptedString;
        }
        string decryptedString =
          StringEncryptor.Decrypt(stringToDecrypt);

        StringBuilder sb = new StringBuilder();
        sb.Append("Encrypted data: ");
        sb.Append(encryptedString);
        sb.Append("<br />Decrypted data: ");
        sb.Append(decryptedString);
        result.Text = sb.ToString();
    }
}
```

### 4.3.2.14 SecurityLibTester3.aspx

```
<%@ Page Language="C#" MasterPageFile="~/BalloonShop.master"
AutoEventWireup="true"
  CodeFile="SecurityLibTester3.aspx.cs" Inherits="SecurityLibTester3"
Title="SecurityLib Test Page 3" %>
```

```
<asp:Content ID="Content1" ContentPlaceHolderID="contentPlaceHolder"
runat="Server">
  Card holder:<br />
  <asp:TextBox ID="cardHolderBox" runat="server" />
  <br />
  Card number:<br />
  <asp:TextBox ID="cardNumberBox" runat="server" />
  <br />
  Issue date:<br />
  <asp:TextBox ID="issueDateBox" runat="server" />
  <br />
  Expiry date:<br />
  <asp:TextBox ID="expiryDateBox" runat="server" />
  <br />
  Issue number:<br />
  <asp:TextBox ID="issueNumberBox" runat="server" />
  <br />
  Card type:<br />
  <asp:TextBox ID="cardTypeBox" runat="server" />
  <br />
  <asp:Button ID="processButton" runat="server" Text="Process"
    OnClick="processButton_Click" />
  <br />
  <asp:Label ID="result" runat="server" />
</asp:Content>
```

### 4.3.2.14.1 SecurityLibTester3.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
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.Text;
using SecurityLib;

public partial class SecurityLibTester3 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void processButton_Click(object sender, EventArgs e)
    {
        SecureCard encryptedCard =
          new SecureCard(cardHolderBox.Text, cardNumberBox.Text,
```

```
            issueDateBox.Text, expiryDateBox.Text, issueNumberBox.Text,
            cardTypeBox.Text);
        string encryptedData = encryptedCard.EncryptedData;
        SecureCard decryptedCard = new SecureCard(encryptedData);
        string decryptedData = string.Format(
          "{0}, {1}, {2}, {3}, {4}, {5}",
          decryptedCard.CardHolder, decryptedCard.CardNumber,
          decryptedCard.IssueDate, decryptedCard.ExpiryDate,
          decryptedCard.IssueNumber, decryptedCard.CardType);

        StringBuilder sb = new StringBuilder();
        sb.Append("Encrypted data:<br />");
        sb.Append("<textarea style=\"width: 400px; height: 150px;\">");
        sb.Append(encryptedData);
        sb.Append("</textarea><br />Decrypted data: ");
        sb.Append(decryptedData);
        result.Text = sb.ToString();
    }
}
```

## 4.3.2.15 ShoppingCart.aspx

```
<%@ Page Language="C#" MasterPageFile="~/BalloonShop.master"
AutoEventWireup="true" CodeFile="ShoppingCart.aspx.cs"
Inherits="ShoppingCart" Title="Untitled Page" %>

<%@ Register Src="UserControls/ProductRecommendations.ascx"
TagName="ProductRecommendations"
    TagPrefix="uc1" %>
<asp:Content ID="Content1" ContentPlaceHolderID="contentPlaceHolder"
Runat="Server">
  <asp:Label ID="titleLabel" runat="server" Text="Your Shopping Cart"
CssClass="ShoppingCartTitle" />
  <br />
  <asp:Label ID="statusLabel" CssClass="AdminPageText" ForeColor="Red"
runat="server" /><br />
  <asp:GridView ID="grid" runat="server" AutoGenerateColumns="False"
DataKeyNames="ProductID" Width="100%" BorderWidth="0px"
OnRowDeleting="grid_RowDeleting">
    <Columns>
      <asp:BoundField DataField="Name" HeaderText="Product Name"
ReadOnly="True" SortExpression="Name" >
        <ControlStyle Width="100%" />
      </asp:BoundField>
      <asp:BoundField DataField="Price" DataFormatString="{0:c}"
HeaderText="Price" ReadOnly="True"
        SortExpression="Price" />
      <asp:TemplateField HeaderText="Quantity">
        <ItemTemplate>
          <asp:TextBox ID="editQuantity" runat="server"
CssClass="GridEditingRow" Width="24px" MaxLength="2"
Text='<%#Eval("Quantity")%>' />
```

```
        </ItemTemplate>
      </asp:TemplateField>
      <asp:BoundField DataField="Subtotal" DataFormatString="{0:c}"
HeaderText="Subtotal"
        ReadOnly="True" SortExpression="Subtotal" />
      <asp:ButtonField ButtonType="Button" CommandName="Delete"
Text="Delete" >
        <ControlStyle CssClass="SmallButtonText " />
      </asp:ButtonField>
    </Columns>
  </asp:GridView>
  <table width="100%">
    <tr>
      <td>
        <span class="ProductDescription">
          Total amount:
        </span>
        <asp:Label ID="totalAmountLabel" runat="server" Text="Label"
CssClass="ProductPrice" />
      </td>
      <td align="right">
        <asp:Button ID="updateButton" runat="server" Text="Update
Quantities" CssClass="SmallButtonText" OnClick="updateButton_Click" />
        <asp:Button ID="checkoutButton" runat="server"
CssClass="SmallButtonText" Text="Proceed to Checkout"
OnClick="checkoutButton_Click" />
      </td>
    </tr>
  </table>
  <br />
  <asp:Button ID="continueShoppingButton" runat="server" Text="Continue
Shopping" CssClass="SmallButtonText"
OnClick="continueShoppingButton_Click" /><br />
    <br />
    <uc1:ProductRecommendations ID="ProductRecommendations1"
runat="server" />
</asp:Content>
```

## 4.3.2.15.1 ShoppingCart.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
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;
```

```
public partial class ShoppingCart : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        // populate the control only on the initial page load
        if (!IsPostBack)
            PopulateControls();
    }

    // fill shopping cart controls with data
    private void PopulateControls()
    {
        // set the title of the page
        this.Title = BalloonShopConfiguration.SiteName + " : Shopping
Cart";
        // get the items in the shopping cart
        DataTable dt = ShoppingCartAccess.GetItems();
        // if the shopping cart is empty...
        if (dt.Rows.Count == 0)
        {
            titleLabel.Text = "Your shopping cart is empty!";
            grid.Visible = false;
            updateButton.Enabled = false;
            checkoutButton.Enabled = false;
            totalAmountLabel.Text = String.Format("{0:c}", 0);
        }
        else
        // if the shopping cart is not empty...
        {
            // populate the list with the shopping cart contents
            grid.DataSource = dt;
            grid.DataBind();
            // setup controls
            titleLabel.Text = "These are the products in your shopping
cart:";
            grid.Visible = true;
            updateButton.Enabled = true;
            checkoutButton.Enabled = true;
            // display the total amount
            decimal amount = ShoppingCartAccess.GetTotalAmount();
            totalAmountLabel.Text = String.Format("{0:c}", amount);
        }
    }

    // remove a product from the cart
    protected void grid_RowDeleting(object sender,
GridViewDeleteEventArgs e)
    {
        // Index of the row being deleted
        int rowIndex = e.RowIndex;
        // The ID of the product being deleted
        string productId = grid.DataKeys[rowIndex].Value.ToString();
        // Remove the product from the shopping cart
        bool success = ShoppingCartAccess.RemoveItem(productId);
        // Display status
        statusLabel.Text = success ? "<br />Product successfully
```

```
removed!<br />" :
                            "<br />There was an error removing the
product!<br />";
        // Repopulate the control
        PopulateControls();
    }

    // update shopping cart product quantities
    protected void updateButton_Click(object sender, EventArgs e)
    {
        // Number of rows in the GridView
        int rowsCount = grid.Rows.Count;
        // Will store a row of the GridView
        GridViewRow gridRow;
        // Will reference a quantity TextBox in the GridView
        TextBox quantityTextBox;
        // Variables to store product ID and quantity
        string productId;
        int quantity;
        // Was the update successful?
        bool success = true;
        // Go through the rows of the GridView
        for (int i = 0; i < rowsCount; i++)
        {
            // Get a row
            gridRow = grid.Rows[i];
            // The ID of the product being deleted
            productId = grid.DataKeys[i].Value.ToString();
            // Get the quantity TextBox in the Row
            quantityTextBox =
(TextBox)gridRow.FindControl("editQuantity");
            // Get the quantity, guarding against bogus values
            if (Int32.TryParse(quantityTextBox.Text, out quantity))
            {
                // Update product quantity
                success = success &&
ShoppingCartAccess.UpdateItem(productId, quantity);
            }
            else
            {
                // if TryParse didn't succeed
                success = false;
            }
            // Display status message
            statusLabel.Text = success ?
              "<br />Your shopping cart was successfully updated!<br
/>" :
              "<br />Some quantity updates failed! Please verify your
cart!<br />";
        }
        // Repopulate the control
        PopulateControls();
    }

    // Redirects to the previously visited catalog page
    // (an alternate to the functionality implemented here is to to
```

```
    // Request.UrlReferrer, although that way you have no control to
    // what pages you forward your visitor back to)
    protected void continueShoppingButton_Click(object sender,
EventArgs e)
    {
        // redirect to the last visited catalog page, or to the
        // main page of the catalog
        object page;
        if ((page = Session["LastVisitedCatalogPage"]) != null)
            Response.Redirect(page.ToString());
        else
            Response.Redirect(Request.ApplicationPath);
    }

    // redirect to the checkout page
    protected void checkoutButton_Click(object sender, EventArgs e)
    {
        string redirect = "Checkout.aspx";
        // Redirect to the checkout page
        Response.Redirect("Checkout.aspx");
    }
}
```

## 4.3.2.16 ShoppingCartAdmin.aspx

```
<%@ Page Language="C#" MasterPageFile="~/Admin.master"
AutoEventWireup="true" CodeFile="ShoppingCartAdmin.aspx.cs"
Inherits="ShoppingCartAdmin" Title="Untitled Page" %>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1"
Runat="Server">
  <span class="AdminTitle">Shopping Cart Admin</span></asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder2"
runat="Server">
  <asp:Label ID="countLabel" runat="server" CssClass="AdminPageText">
      Hello!
  </asp:Label><br />
  <span class="AdminPageText">How many days?</span>
  <asp:DropDownList ID="daysList" runat="server">
    <asp:ListItem Value="0">All shopping carts</asp:ListItem>
    <asp:ListItem Value="1">One</asp:ListItem>
    <asp:ListItem Value="10" Selected="True">Ten</asp:ListItem>
    <asp:ListItem Value="20">Twenty</asp:ListItem>
    <asp:ListItem Value="30">Thirty</asp:ListItem>
    <asp:ListItem Value="90">Ninety</asp:ListItem>
  </asp:DropDownList><br />
  <br />
  <asp:Button ID="countButton" runat="server" Text="Count Old Shopping
Carts" CssClass="Button" OnClick="countButton_Click" />
  <asp:Button ID="deleteButton" runat="server" Text="Delete Old
Shopping Carts" CssClass="Button" OnClick="deleteButton_Click"
/></asp:Content>
```

### 4.3.2.16.1 ShoppingCartAdmin.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
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;

public partial class ShoppingCartAdmin : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        // Set the title of the page
        this.Title = BalloonShopConfiguration.SiteName +
                     " : Shopping Cart Admin";
    }

    // counts old shopping carts
    protected void countButton_Click(object sender, EventArgs e)
    {
        byte days = byte.Parse(daysList.SelectedItem.Value);
        int oldItems = ShoppingCartAccess.CountOldCarts(days);
        if (oldItems == -1)
            countLabel.Text = "Could not count the old shopping
carts!";
        else if (oldItems == 0)
            countLabel.Text = "There are no old shopping carts.";
        else
            countLabel.Text = "There are " + oldItems.ToString() +
                              " old shopping carts.";
    }


    // deletes old shopping carts
    protected void deleteButton_Click(object sender, EventArgs e)
    {
        byte days = byte.Parse(daysList.SelectedItem.Value);
        ShoppingCartAccess.DeleteOldCarts(days);
        countLabel.Text = "The old shopping carts were removed from the
database";
    }
}
```

### 4.3.3 Web User Control

Τα Web User Controls είναι συστατικά (**components**), τα οποία μπορούμε να τα χρησιμοποιήσουμε μέσα σε Master Pages, Web Forms κ.α. Ακόμα μπορούμε να τα χρησιμοποίησουμε σ' άλλες εφαρμογές που θα κατασκευάσουμε στο μέλλον.

Παρακάτω ακολουθεί ο κώδικας των Web User Controls της φαρμογής μας.

#### 4.3.3.1 CartSummary.ascx

```
<%@ Control Language="C#" AutoEventWireup="true"
CodeFile="CartSummary.ascx.cs" Inherits="CartSummary" %>
<table border="0" cellpadding="0" cellspacing="1" width="200">
  <tr>
    <td class="CartSummary">
      <b><asp:Label ID="cartSummaryLabel" runat="server" /></b>
      <asp:HyperLink ID="viewCartLink" runat="server"
NavigateUrl="../ShoppingCart.aspx"
        CssClass="CartLink" Text="(view details)" />
      <asp:DataList ID="list" runat="server">
        <ItemTemplate>
          <%# Eval("Quantity") %> x <%# Eval("Name") %>
        </ItemTemplate>
      </asp:DataList>
      <img src="Images/line.gif" border="0" width="99%" height="1" />
      Total:
      <span class="ProductPrice">
        <asp:Label ID="totalAmountLabel" runat="server" />
      </span>
    </td>
  </tr>
</table>
```

#### 4.3.3.1.1 CartSummary.ascx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
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;

public partial class CartSummary : System.Web.UI.UserControl
{
```

```
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    // fill cart summary contents in the PreRender stage
    protected void Page_PreRender(object sender, EventArgs e)
    {
        PopulateControls();
    }

    // fill the controls with data
    private void PopulateControls()
    {
        // get the items in the shopping cart
        DataTable dt = ShoppingCartAccess.GetItems();
        // if the shopping cart is empty...
        if (dt.Rows.Count == 0)
        {
            cartSummaryLabel.Text = "Your shopping cart is empty.";
            totalAmountLabel.Text = String.Format("{0:c}", 0);
            viewCartLink.Visible = false;
            list.Visible = false;
        }
        else
        // if the shopping cart is not empty...
        {
            // populate the list with the shopping cart contents
            list.Visible = true;
            list.DataSource = dt;
            list.DataBind();
            // setup controls
            cartSummaryLabel.Text = "Cart summary ";
            viewCartLink.Visible = true;
            // display the total amount
            decimal amount = ShoppingCartAccess.GetTotalAmount();
            totalAmountLabel.Text = String.Format("{0:c}", amount);
        }
    }

    // we don't want to display the cart summary in the shopping cart
page
    protected void Page_Init(object sender, EventArgs e)
    {
        // get the current page
        string page = Request.AppRelativeCurrentExecutionFilePath;
        // if we're in the shopping cart, don't display the cart
summary
        if (String.Compare(page, "~/ShoppingCart.aspx", true) == 0)
            this.Visible = false;
        else
            this.Visible = true;
    }
}
```

### 4.3.3.2 CategoriesAdmin.ascx

```
<%@ Control Language="C#" AutoEventWireup="true"
CodeFile="CategoriesAdmin.ascx.cs"
  Inherits="CategoriesAdmin" %>
<asp:Label ID="statusLabel" runat="server" CssClass="AdminPageText"
Text="Categories Loaded"></asp:Label><br />
<br />
<asp:Label ID="locationLabel" runat="server" CssClass="AdminPageText"
Text="Displaying categories for department..."></asp:Label>
<asp:LinkButton ID="goBackLink" runat="server" CssClass="AdminPageText"
OnClick="goBackLink_Click">(go back to departments)</asp:LinkButton><br
/>
<br />
<asp:GridView ID="grid" runat="server" AutoGenerateColumns="False"
DataKeyNames="CategoryID"
  Width="100%" OnRowCancelingEdit="grid_RowCancelingEdit"
OnRowDeleting="grid_RowDeleting" OnRowEditing="grid_RowEditing"
OnRowUpdating="grid_RowUpdating">
  <Columns>
    <asp:BoundField DataField="Name" HeaderText="Category Name"
SortExpression="Name" />
    <asp:TemplateField HeaderText="Category Description"
SortExpression="Description">
      <ItemTemplate>
        <asp:Label ID="Label1" runat="server" Text='<%#
Bind("Description") %>'>
        </asp:Label>
      </ItemTemplate>
      <EditItemTemplate>
        <asp:TextBox ID="descriptionTextBox" runat="server"
TextMode="MultiLine" Text='<%# Bind("Description") %>'
          Height="70px" Width="350px" />
      </EditItemTemplate>
    </asp:TemplateField>
    <asp:TemplateField>
      <ItemTemplate>
        <asp:HyperLink runat="server" ID="link" NavigateUrl='<%#
"../CatalogAdmin.aspx?DepartmentID=" +
Request.QueryString["DepartmentID"] + "&amp;CategoryID=" +
Eval("CategoryID")%>'
          Text="View Products">
        </asp:HyperLink>
      </ItemTemplate>
    </asp:TemplateField>
    <asp:CommandField ShowEditButton="True" />
    <asp:ButtonField CommandName="Delete" Text="Delete" />
  </Columns>
</asp:GridView>
<br />
<span class="AdminPageText">Create a new category in this
department:</span>
<table class="AdminPageText" cellspacing="0">
  <tr>
    <td valign="top" width="100">
```

```
      Name:</td>
    <td>
      <asp:TextBox CssClass="AdminPageText" ID="newName" runat="server"
Width="400px" />
    </td>
  </tr>
  <tr>
    <td valign="top" width="100">
      Description:</td>
    <td>
      <asp:TextBox CssClass="AdminPageText" ID="newDescription"
runat="server" Width="400px"
        Height="70px" TextMode="MultiLine" />
    </td>
  </tr>
</table>
<asp:Button ID="createCategory" Text="Create Category" runat="server"
CssClass="AdminButtonText" OnClick="createCategory_Click" />
```

### 4.3.3.2.1 CategoriesAdmin.ascx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
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;

public partial class CategoriesAdmin : System.Web.UI.UserControl
{
    protected void Page_Load(object sender, EventArgs e)
    {
        // Load the grid only the first time the page is loaded
        if (!Page.IsPostBack)
        {
            // Load the categories grid
            BindGrid();
            // Get DepartmentID from the query string
            string departmentId = Request.QueryString["DepartmentID"];
            // Obtain the department's name
            DepartmentDetails dd =
CatalogAccess.GetDepartmentDetails(departmentId);
            string departmentName = dd.Name;
            // Set controls' properties
            statusLabel.ForeColor = System.Drawing.Color.Red;
            locationLabel.Text = "Displaying categories for department
<b> "
              + departmentName + "</b>";
```

```
        }
    }

    // Populate the GridView with data
    private void BindGrid()
    {
        // Get DepartmentID from the query string
        string departmentId = Request.QueryString["DepartmentID"];
        // Get a DataTable object containing the categories
        grid.DataSource =
CatalogAccess.GetCategoriesInDepartment(departmentId);
        // Bind the data grid to the data source
        grid.DataBind();
    }

    // Enter row into edit mode
    protected void grid_RowEditing(object sender, GridViewEditEventArgs
e)
    {
        // Set the row for which to enable edit mode
        grid.EditIndex = e.NewEditIndex;
        // Set status message
        statusLabel.Text = "Editing row # " +
e.NewEditIndex.ToString();
        // Reload the grid
        BindGrid();
    }

    // Cancel edit mode
    protected void grid_RowCancelingEdit(object sender,
GridViewCancelEditEventArgs e)
    {
        // Cancel edit mode
        grid.EditIndex = -1;
        // Set status message
        statusLabel.Text = "Editing canceled";
        // Reload the grid
        BindGrid();
    }

    // Update row
    protected void grid_RowUpdating(object sender,
GridViewUpdateEventArgs e)
    {
        // Retrieve updated data
        string id = grid.DataKeys[e.RowIndex].Value.ToString();
        string name =
((TextBox)grid.Rows[e.RowIndex].Cells[0].Controls[0]).Text;
        string description =
((TextBox)grid.Rows[e.RowIndex].FindControl("descriptionTextBox")).Text
;
        // Execute the update command
        bool success = CatalogAccess.UpdateCategory(id, name,
description);
        // Cancel edit mode
        grid.EditIndex = -1;
```

```
        // Display status message
        statusLabel.Text = success ? "Update successful" : "Update
failed";
        // Reload the grid
        BindGrid();
    }

    // Delete a record
    protected void grid_RowDeleting(object sender,
GridViewDeleteEventArgs e)
    {
        // Get the ID of the record to be deleted
        string id = grid.DataKeys[e.RowIndex].Value.ToString();
        // Execute the delete command
        bool success = CatalogAccess.DeleteCategory(id);
        // Cancel edit mode
        grid.EditIndex = -1;
        // Display status message
        statusLabel.Text = success ? "Delete successful" : "Delete
failed";
        // Reload the grid
        BindGrid();
    }

    // Create a new category
    protected void createCategory_Click(object sender, EventArgs e)
    {
        // Get DepartmentID from the query string
        string departmentId = Request.QueryString["DepartmentID"];
        // Execute the insert command
        bool success = CatalogAccess.CreateCategory(departmentId,
newName.Text, newDescription.Text);
        // Display results
        statusLabel.Text = success ? "Insert successful" : "Insert
failed";
        // Reload the grid
        BindGrid();
    }

    // Redirect to the department's page
    protected void goBackLink_Click(object sender, EventArgs e)
    {
        Response.Redirect(Request.ApplicationPath +
"/CatalogAdmin.aspx");
    }
}
```

### 4.3.3.3 CategoriesList.ascx

```
<%@ Control Language="C#" AutoEventWireup="true"
CodeFile="CategoriesList.ascx.cs" Inherits="CategoriesList" %>
<asp:DataList ID="list" runat="server" CssClass="CategoryListContent"
```

```
Width="200px">
  <ItemTemplate>
     &raquo;
    <asp:HyperLink
      ID="HyperLink1"
      Runat="server"
      NavigateUrl='<%# "../Catalog.aspx?DepartmentID=" +
Request.QueryString["DepartmentID"] + "&CategoryID=" +
Eval("CategoryID")  %>'
      Text='<%# Eval("Name") %>'
      ToolTip='<%# Eval("Description") %>'
      CssClass='<%# Eval("CategoryID").ToString() ==
Request.QueryString["CategoryID"] ? "CategorySelected" :
"CategoryUnselected" %>'>>
    </asp:HyperLink>
     &laquo;
  </ItemTemplate>
  <HeaderTemplate>
    Choose a Category
  </HeaderTemplate>
  <HeaderStyle CssClass="CategoryListHead" />
</asp:DataList>
<asp:Label ID="brLabel" runat="server" Text="" />
```

### 4.3.3.3.1 CategoriesList.ascx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
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;

public partial class CategoriesList : System.Web.UI.UserControl
{
    protected void Page_Load(object sender, EventArgs e)
    {
        // don't reload data during postbacks
        if (!IsPostBack)
        {
            // Obtain the ID of the selected department
            string departmentId = Request.QueryString["DepartmentID"];
            // Continue only if DepartmentID exists in the query string
            if (departmentId != null)
            {
                // Catalog.GetCategoriesInDepartment returns a
DataTable object containing
```

```
                // category data, which is displayed by the DataList
                list.DataSource =
CatalogAccess.GetCategoriesInDepartment(departmentId);
                // Needed to bind the data bound controls to the data
source
                list.DataBind();
                // Make space for the next control
                brLabel.Text = "<br />";
            }
        }
    }
}
```

## 4.3.3.4 CustomerDetailsEdit.ascx

```
<%@ Control Language="C#" AutoEventWireup="true"
CodeFile="CustomerDetailsEdit.ascx.cs" Inherits="CustomerDetailsEdit"
%>
<asp:ObjectDataSource ID="ObjectDataSource1" runat="server"
DataObjectTypeName="ProfileWrapper"
  SelectMethod="GetData" TypeName="ProfileDataSource"
UpdateMethod="UpdateData"></asp:ObjectDataSource>
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
ConnectionString="<%$ ConnectionStrings:BalloonShopConnection %>"
  SelectCommand="SELECT [ShippingRegionID], [ShippingRegion] FROM
[ShippingRegion]"></asp:SqlDataSource>
<asp:FormView ID="FormView1" runat="server"
DataSourceID="ObjectDataSource1">
  <HeaderTemplate>
    <table border="0" cellpadding="4" cellspacing="0"
      class="UserDetailsTable">
      <tr><td colspan="2" class="UserDetailsTableHead">
        <asp:Label runat="server" ID="TitleLabel" /></td></tr>
  </HeaderTemplate>
  <FooterTemplate>
    </table>
  </FooterTemplate>
  <EditItemTemplate>
      <tr><td>Address line 1: </td><td width="350px">
        <asp:TextBox Width="340px" ID="Address1TextBox" runat="server"
          Text='<%# Bind("Address1") %>' />
      </td></tr>
      <tr><td>Address line 2: </td><td>
        <asp:TextBox Width="340px" ID="Address2TextBox" runat="server"
          Text='<%# Bind("Address2") %>' />
      </td></tr>
      <tr><td>City: </td><td>
        <asp:TextBox Width="340px" ID="CityTextBox" runat="server"
          Text='<%# Bind("City") %>' />
      </td></tr>
      <tr><td>Region: </td><td>
        <asp:TextBox Width="340px" ID="RegionTextBox" runat="server"
```

```
      Text='<%# Bind("Region") %>' />
</td></tr>
<tr><td>Zip / Postal Code: </td><td>
  <asp:TextBox Width="340px" ID="PostalCodeTextBox"
    runat="server" Text='<%# Bind("PostalCode") %>' />
</td></tr>
<tr><td>Country: </td><td>
  <asp:TextBox Width="340px" ID="CountryTextBox" runat="server"
    Text='<%# Bind("Country") %>' />
</td></tr>
<tr><td>Shipping Region: </td><td>
  <asp:DropDownList Width="350px" ID="ShippingRegionDropDown"
    runat="server"
    SelectedValue='<%# Bind("ShippingRegion") %>'
    DataSourceID="SqlDataSource1"
    DataTextField="ShippingRegion"
    DataValueField="ShippingRegionID">
  </asp:DropDownList>
</td></tr>
<tr><td>Daytime Phone no: </td><td>
  <asp:TextBox Width="340px" ID="DayPhoneTextBox" runat="server"
    Text='<%# Bind("DayPhone") %>' />
</td></tr>
<tr><td>Evening Phone no: </td><td>
  <asp:TextBox Width="340px" ID="EvePhoneTextBox" runat="server"
    Text='<%# Bind("EvePhone") %>' />
</td></tr>
<tr><td>Mobile Phone no: </td><td>
  <asp:TextBox Width="340px" ID="MobPhoneTextBox" runat="server"
    Text='<%# Bind("MobPhone") %>' />
</td></tr>
<tr><td>Email: </td><td>
  <asp:TextBox Width="340px" ID="EmailBox" runat="server"
    Text='<%# Bind("Email") %>' />
</td></tr>
<tr><td valign="top">Credit Card: </td><td>
  <table cellpadding="0" cellspacing="0" border="0">
    <tr><td width="140px">Cardholder name: </td>
      <td width="200px">
      <asp:TextBox Width="200px" ID="CreditCardHolderLabel"
        runat="server" Text='<%# Bind("CreditCardHolder") %>' />
    </td></tr>
    <tr><td>Card type: </td><td>
      <asp:TextBox Width="200px" ID="CreditCardTypeLabel"
        runat="server" Text='<%# Bind("CreditCardType") %>' />
    </td></tr>
    <tr><td>Card number: </td><td>
      <asp:TextBox Width="200px" ID="CreditCardNumberLabel"
        runat="server" Text='<%# Bind("CreditCardNumber") %>' />
    </td></tr>
    <tr><td>Issue date: </td><td>
      <asp:TextBox Width="200px" ID="CreditCardIssueDateLabel"
        runat="server"
        Text='<%# Bind("CreditCardIssueDate") %>' />
    </td></tr>
    <tr><td>Expiry date: </td><td>
```

```
            <asp:TextBox Width="200px" ID="CreditCardExpiryDateLabel"
              runat="server"
              Text='<%# Bind("CreditCardExpiryDate") %>' />
          </td></tr>
          <tr><td>Issue number: </td><td>
            <asp:TextBox Width="200px" ID="CreditCardIssueNumberLabel"
              runat="server"
              Text='<%# Bind("CreditCardIssueNumber") %>' />
          </td></tr>
        </table>
      </td></tr>
      <tr><td>
        <asp:Button ID="UpdateButton" runat="server"
          CausesValidation="True" CommandName="Update"
          Text="Update" /> <asp:Button ID="UpdateCancelButton"
          runat="server" CausesValidation="False" CommandName="Cancel"
          Text="Cancel" />
      </td></tr>
  </EditItemTemplate>
  <ItemTemplate>
      <tr><td>Address line 1: </td><td width="350px">
        <asp:Label ID="Address1Label" runat="server"
          Text='<%# Bind("Address1") %>' />
      </td></tr>
      <tr><td>Address line 2: </td><td>
        <asp:Label ID="Address2Label" runat="server"
          Text='<%# Bind("Address2") %>' />
      </td></tr>
      <tr><td>City: </td><td>
        <asp:Label ID="CityLabel" runat="server"
          Text='<%# Bind("City") %>' />
      </td></tr>
      <tr><td>Region: </td><td>
        <asp:Label ID="RegionLabel" runat="server"
          Text='<%# Bind("Region") %>' />
      </td></tr>
      <tr><td>Zip / Postal Code: </td><td>
        <asp:Label ID="PostalCodeLabel" runat="server"
          Text='<%# Bind("PostalCode") %>'>
        </asp:Label>
      </td></tr>
      <tr><td>Country: </td><td>
        <asp:Label ID="CountryLabel" runat="server"
          Text='<%# Bind("Country") %>' />
      </td></tr>
      <tr><td>Shipping Region: </td><td>
        <asp:DropDownList Width="350px" ID="ShippingRegionDropDown"
          runat="server"
          SelectedValue='<%# Bind("ShippingRegion") %>'
          DataSourceID="SqlDataSource1"
          DataTextField="ShippingRegion"
          DataValueField="ShippingRegionID"
          enabled="false">
        </asp:DropDownList>
      </td></tr>
      <tr><td>Daytime Phone no: </td><td>
```

```
        <asp:Label ID="DayPhoneLabel" runat="server"
          Text='<%# Bind("DayPhone") %>' />
      </td></tr>
      <tr><td>Evening Phone no: </td><td>
        <asp:Label ID="EvePhoneLabel" runat="server"
          Text='<%# Bind("EvePhone") %>' />
      </td></tr>
      <tr><td>Mobile Phone no: </td><td>
        <asp:Label ID="MobPhoneLabel" runat="server"
          Text='<%# Bind("MobPhone") %>' />
      </td></tr>
      <tr><td>Email: </td><td>
        <asp:Label ID="EmailLabel" runat="server"
          Text='<%# Bind("Email") %>' />
      </td></tr>
      <tr><td>Credit Card: </td><td>
        <asp:Label ID="CreditCardLabel" runat="server"
          Text='<%# Bind("CreditCard") %>' />
      </td></tr>
      <tr><td>
        <asp:Button ID="EditButton" runat="server"
          CausesValidation="False" CommandName="Edit"
          Text="Edit" />
      </td></tr>
  </ItemTemplate>
</asp:FormView>
```

### 4.3.3.4.1 CustomerDetailsEdit.ascx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
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;

public partial class CustomerDetailsEdit : System.Web.UI.UserControl
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    public bool Editable
    {
        get
        {
            if (ViewState["editable"] != null)
```

### 4.3.3.5 DepartmentsAdmin.ascx

```
<%@ Control Language="C#" AutoEventWireup="true"
CodeFile="DepartmentsAdmin.ascx.cs"
Inherits="UserControls_DepartmentsAdmin" %>
<asp:Label ID="statusLabel" runat="server" CssClass="AdminPageText"
Text="Departments Loaded"></asp:Label><br />
<br />
<asp:Label ID="locationLabel" runat="server" CssClass="AdminPageText"
Text="These are your departments:"></asp:Label><br />
<br />
<asp:GridView ID="grid" runat="server" DataKeyNames="DepartmentID"
Width="100%" AutoGenerateColumns="False"
OnRowCancelingEdit="grid_RowCancelingEdit"
OnRowDeleting="grid_RowDeleting" OnRowEditing="grid_RowEditing"
OnRowUpdating="grid_RowUpdating">
    <Columns>
        <asp:BoundField DataField="Name" HeaderText="Department Name"
SortExpression="Name" />
        <asp:TemplateField HeaderText="Department Description"
SortExpression="Description">
            <EditItemTemplate>
                <asp:TextBox ID="descriptionTextBox" runat="server"
Text='<%# Bind("Description") %>' Height="70px" TextMode="MultiLine"
Width="350px"></asp:TextBox>
            </EditItemTemplate>
            <ItemTemplate>
                <asp:Label ID="Label1" runat="server" Text='<%#
Bind("Description") %>'></asp:Label>
            </ItemTemplate>
        </asp:TemplateField>
        <asp:HyperLinkField DataNavigateUrlFields="DepartmentID"
DataNavigateUrlFormatString="../CatalogAdmin.aspx?DepartmentID={0}"
            Text="View Categories" />
        <asp:CommandField ShowEditButton="True" />
        <asp:ButtonField CommandName="Delete" Text="Delete" />
    </Columns>
</asp:GridView>
<br />
<span class="AdminPageText">Create a new department:</span>
<table class="AdminPageText" cellspacing="0">
  <tr>
    <td valign="top" width="100">Name:
    </td>
    <td>
      <asp:TextBox cssClass="AdminPageText" ID="newName" Runat="server"
Width="400px" />
    </td>
  </tr>
  <tr>
    <td valign="top" width="100">Description:
    </td>
    <td>
      <asp:TextBox cssClass="AdminPageText" ID="newDescription"
Runat="server" Width="400px" Height="70px" TextMode="MultiLine"/>
```

```
            {
                return (bool)ViewState["editable"];
            }
            else
            {
                return true;
            }
        }
        set
        {
            ViewState["editable"] = value;
        }
    }

    public string Title
    {
        get
        {
            if (ViewState["title"] != null)
            {
                return ViewState["title"] as string;
            }
            else
            {
                return "Edit User Details";
            }
        }
        set
        {
            ViewState["title"] = value;
        }
    }

    protected override void OnPreRender(EventArgs e)
    {
        // Find and set title text
        Label TitleLabel =
           FormView1.FindControl("TitleLabel") as Label;
        if (TitleLabel != null)
        {
            TitleLabel.Text = Title;
        }

        // Find and set edit button visibility
        Button EditButton =
           FormView1.FindControl("EditButton") as Button;
        if (EditButton != null)
        {
            EditButton.Visible = Editable;
        }
    }
}
```

```
    </td>
  </tr>
</table>
<asp:Button ID="createDepartment" Text="Create Department"
Runat="server"
CssClass="AdminButtonText" OnClick="createDepartment_Click" />
```

### 4.3.3.5.1 DepartmentsAdmin.ascx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
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;

public partial class UserControls_DepartmentsAdmin :
System.Web.UI.UserControl
{
    protected void Page_Load(object sender, EventArgs e)
    {
        // Load the grid only the first time the page is loaded
        if (!Page.IsPostBack)
        {
            // Load the departments grid
            BindGrid();
            // Set control properties
            statusLabel.ForeColor = System.Drawing.Color.Red;
        }
    }

    // Populate the GridView with data
    private void BindGrid()
    {
        // Get a DataTable object containing the catalog departments
        grid.DataSource = CatalogAccess.GetDepartments();
        // Bind the data bound controls to the data source
        grid.DataBind();
    }
    // Enter row into edit mode
    protected void grid_RowEditing(object sender, GridViewEditEventArgs
e)
    {
        // Set the row for which to enable edit mode
        grid.EditIndex = e.NewEditIndex;
        // Set status message
        statusLabel.Text = "Editing row # " +
e.NewEditIndex.ToString();
```

```
        // Reload the grid
        BindGrid();
    }
    // Cancel edit mode
    protected void grid_RowCancelingEdit(object sender,
GridViewCancelEditEventArgs e)
    {
        // Cancel edit mode
        grid.EditIndex = -1;
        // Set status message
        statusLabel.Text = "Editing canceled";
        // Reload the grid
        BindGrid();
    }
    // Update row
    protected void grid_RowUpdating(object sender,
GridViewUpdateEventArgs e)
    {
        // Retrieve updated data
        string id = grid.DataKeys[e.RowIndex].Value.ToString();
        string name =
((TextBox)grid.Rows[e.RowIndex].Cells[0].Controls[0]).Text;
        string description =
((TextBox)grid.Rows[e.RowIndex].FindControl("descriptionTextBox")).Text
;
        // Execute the update command
        bool success = CatalogAccess.UpdateDepartment(id, name,
description);
        // Cancel edit mode
        grid.EditIndex = -1;
        // Display status message
        statusLabel.Text = success ? "Update successful" : "Update
failed";
        // Reload the grid
        BindGrid();
    }
    // Delete a record
    protected void grid_RowDeleting(object sender,
GridViewDeleteEventArgs e)
    {
        // Get the ID of the record to be deleted
        string id = grid.DataKeys[e.RowIndex].Value.ToString();
        // Execute the delete command
        bool success = CatalogAccess.DeleteDepartment(id);
        // Cancel edit mode
        grid.EditIndex = -1;
        // Display status message
        statusLabel.Text = success ? "Delete successful" : "Delete
failed";
        // Reload the grid
        BindGrid();
    }
    // Create a new department
    protected void createDepartment_Click(object sender, EventArgs e)
    {
        // Execute the insert command
```

```
        bool success = CatalogAccess.AddDepartment(newName.Text,
newDescription.Text);
        // Display status message
        statusLabel.Text = success ? "Insert successful" : "Insert
failed";
        // Reload the grid
        BindGrid();
    }
}
```

## 4.3.3.6 DepartmentsList.ascx

```
<%@ Control Language="C#" AutoEventWireup="true"
CodeFile="DepartmentsList.ascx.cs" Inherits="DepartmentsList" %>
<asp:DataList ID="list" runat="server" Width="200px">
  <ItemTemplate>
     &raquo;
    <asp:HyperLink
      ID="HyperLink1"
      Runat="server"
      NavigateUrl='<%# "../Catalog.aspx?DepartmentID=" +
Eval("DepartmentID")%>'
      Text='<%# Eval("Name") %>'
      ToolTip='<%# Eval("Description") %>'
      CssClass='<%# Eval("DepartmentID").ToString() ==
Request.QueryString["DepartmentID"] ? "DepartmentSelected" :
"DepartmentUnselected" %>'>
    </asp:HyperLink>
     &laquo;
 </ItemTemplate>
  <HeaderTemplate>
    Choose a Department
  </HeaderTemplate>
  <ItemStyle CssClass="DepartmentsListContent" />
  <HeaderStyle CssClass="DepartmentListHead" />
</asp:DataList>
```

## 4.3.3.6.1 DepartmentsList.ascx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
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;
```

```
public partial class DepartmentsList : System.Web.UI.UserControl
{
    // Load department details into the DataList
    protected void Page_Load(object sender, EventArgs e)
    {
        // don't reload data during postbacks
        if (!IsPostBack)
        {
            // CatalogAccess.GetDepartments returns a DataTable object
containing
            // department data, which is read in the ItemTemplate of
the DataList
            list.DataSource = CatalogAccess.GetDepartments();
            // Needed to bind the data bound controls to the data
source
            list.DataBind();
        }
    }
}
```

### 4.3.3.7 Header.ascx

```
<%@ Control Language="C#" AutoEventWireup="true"
CodeFile="Header.ascx.cs" Inherits="Header" %>
<p align="center">
  <a href="Default.aspx">
    <img src="Images/BalloonShopLogo.png" border="0">
  </a>
</p>
```

### 4.3.3.7.1 Header.ascx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
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;

public partial class Header : System.Web.UI.UserControl
{
    protected void Page_Load(object sender, EventArgs e)
    {
```

```
    }
}
```

### 4.3.3.8 OrderDetailsAdmin.ascx

```
<%@ Control Language="C#" AutoEventWireup="true"
CodeFile="OrderDetailsAdmin.ascx.cs" Inherits="OrderDetailsAdmin" %>
<asp:Label ID="orderIdLabel" runat="server"
           CssClass="AdminTitle" Text="Order #000" />
<br /><br />
<table class="AdminPageText">
  <tr>
    <td width="130">Total Amount:</td>
    <td>
      <asp:Label ID="totalAmountLabel" runat="server"
CssClass="ProductPrice" />
    </td>
  </tr>
  <tr>
    <td width="130">Date Created:</td>
    <td>
      <asp:TextBox ID="dateCreatedTextBox" runat="server" Width="400px"
/>
    </td>
  </tr>
  <tr>
    <td width="130">Date Shipped:</td>
    <td>
      <asp:TextBox ID="dateShippedTextBox" runat="server" Width="400px"
/>
    </td>
  </tr>
  <tr>
    <td width="130">Verified:</td>
    <td>
      <asp:CheckBox ID="verifiedCheck" runat="server" />
    </td>
  </tr>
  <tr>
    <td width="130">Completed:</td>
    <td>
      <asp:CheckBox ID="completedCheck" runat="server" />
    </td>
  </tr>
  <tr>
    <td width="130">Canceled:</td>
    <td>
      <asp:CheckBox ID="canceledCheck" runat="server" />
    </td>
  </tr>
  <tr>
    <td width="130">Comments:</td>
```

```
    <td>
      <asp:TextBox ID="commentsTextBox" runat="server" Width="400px" />
    </td>
  </tr>
  <tr>
    <td width="130">Customer Name:</td>
    <td>
      <asp:TextBox ID="customerNameTextBox" runat="server"
Width="400px" />
    </td>
  </tr>
  <tr>
    <td width="130">Shipping Address:</td>
    <td>
      <asp:TextBox ID="shippingAddressTextBox" runat="server"
Width="400px" />
    </td>
  </tr>
  <tr>
    <td width="130">Customer Email:</td>
    <td>
      <asp:TextBox ID="customerEmailTextBox" runat="server"
Width="400px" />
    </td>
  </tr>
</table>
<br />
<asp:Button ID="editButton" runat="server" CssClass="SmallButtonText"
            Text="Edit" Width="100px" OnClick="editButton_Click" />
<asp:Button ID="updateButton" runat="server" CssClass="SmallButtonText"
            Text="Update" Width="100px" OnClick="updateButton_Click" />
<asp:Button ID="cancelButton" runat="server" CssClass="SmallButtonText"
            Text="Cancel" Width="100px" OnClick="cancelButton_Click"
/><br />
<asp:Button ID="markVerifiedButton" runat="server"
CssClass="SmallButtonText"
            Text="Mark Order as Verified" Width="310px"
OnClick="markVerifiedButton_Click" /><br />
<asp:Button ID="markCompletedButton" runat="server"
CssClass="SmallButtonText"
            Text="Mark Order as Completed" Width="310px"
OnClick="markCompletedButton_Click" /><br />
<asp:Button ID="markCanceledButton" runat="server"
CssClass="SmallButtonText"
            Text="Mark Order as Canceled" Width="310px"
OnClick="markCanceledButton_Click" /><br />
<br />
<asp:Label ID="Label13" runat="server" CssClass="AdminPageText"
Text="The order contains these items:" />
<br />
<asp:GridView ID="grid" runat="server" AutoGenerateColumns="False"
BackColor="White" Width="100%">
  <Columns>
    <asp:BoundField DataField="ProductID" HeaderText="Product ID"
                    ReadOnly="True" SortExpression="ProductID" />
    <asp:BoundField DataField="ProductName" HeaderText="Product Name"
```

```
                    ReadOnly="True" SortExpression="ProductName" />
    <asp:BoundField DataField="Quantity" HeaderText="Quantity"
                    ReadOnly="True" SortExpression="Quantity" />
    <asp:BoundField DataField="UnitCost" HeaderText="Unit Cost"
                    ReadOnly="True" SortExpression="UnitCost" />
    <asp:BoundField DataField="Subtotal" HeaderText="Subtotal"
                    ReadOnly="True" SortExpression="Subtotal" />
  </Columns>
</asp:GridView>
```

### 4.3.3.8.1 OrderDetailsAdmin.ascx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
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;

public partial class OrderDetailsAdmin : System.Web.UI.UserControl
{
    // edit mode by default is false
    private bool editMode = false;

    // set up the form
    protected void Page_PreRender(object sender, EventArgs e)
    {
        // check if we must display order details
        if (Session["AdminOrderID"] != null)
        {
            // fill constituent controls with data
            PopulateControls();
            // set edit mode
            SetEditMode(editMode);
        }
        else
            //   Hide
            this.Visible = false;
    }

    // populate the form with data
    private void PopulateControls()
    {
        // obtain order ID from the session
        string orderId = Session["AdminOrderID"].ToString();
        // obtain order info
        OrderInfo orderInfo = OrdersAccess.GetInfo(orderId);
        // populate labels and text boxes with order info
```

```
        orderIdLabel.Text = "Displaying Order #" + orderId;
        totalAmountLabel.Text = String.Format("{0:c}",
orderInfo.TotalAmount);
        dateCreatedTextBox.Text = orderInfo.DateCreated;
        dateShippedTextBox.Text = orderInfo.DateShipped;
        verifiedCheck.Checked = orderInfo.Verified;
        completedCheck.Checked = orderInfo.Completed;
        canceledCheck.Checked = orderInfo.Canceled;
        commentsTextBox.Text = orderInfo.Comments;
        customerNameTextBox.Text = orderInfo.CustomerName;
        shippingAddressTextBox.Text = orderInfo.ShippingAddress;
        customerEmailTextBox.Text = orderInfo.CustomerEmail;
        // by default the Edit button is enabled, and the
        // Update and Cancel buttons are disabled
        editButton.Enabled = true;
        updateButton.Enabled = false;
        cancelButton.Enabled = false;
        // Decide which one of the other three buttons
        // should be enabled and which should be disabled
        if (canceledCheck.Checked || completedCheck.Checked)
        {
            // if the order was canceled or completed ...
            markVerifiedButton.Enabled = false;
            markCompletedButton.Enabled = false;
            markCanceledButton.Enabled = false;
        }
        else if (verifiedCheck.Checked)
        {
            // if the order was not canceled but is verified ...
            markVerifiedButton.Enabled = false;
            markCompletedButton.Enabled = true;
            markCanceledButton.Enabled = true;
        }
        else
        {
            // if the order was not canceled and is not verified ...
            markVerifiedButton.Enabled = true;
            markCompletedButton.Enabled = false;
            markCanceledButton.Enabled = true;
        }
        // fill the data grid with order details
        grid.DataSource = OrdersAccess.GetDetails(orderId);
        grid.DataBind();
    }

    // enable or disable edit mode
    private void SetEditMode(bool enable)
    {
        dateCreatedTextBox.Enabled = enable;
        dateShippedTextBox.Enabled = enable;
        verifiedCheck.Enabled = enable;
        completedCheck.Enabled = enable;
        canceledCheck.Enabled = enable;
        commentsTextBox.Enabled = enable;
        customerNameTextBox.Enabled = enable;
        shippingAddressTextBox.Enabled = enable;
```

```
        customerEmailTextBox.Enabled = enable;
        editButton.Enabled = !enable;
        updateButton.Enabled = enable;
        cancelButton.Enabled = enable;
    }

    // enter edit mode
    protected void editButton_Click(object sender, EventArgs e)
    {
        editMode = true;
    }

    // cancel edit mode
    protected void cancelButton_Click(object sender, EventArgs e)
    {
        // don't need to do anything, editMode will be set to false by
default
    }

    // update order information
    protected void updateButton_Click(object sender, EventArgs e)
    {
        // Store the new order details in an OrderInfo object
        OrderInfo orderInfo = new OrderInfo();
        string orderId = Session["AdminOrderID"].ToString();
        orderInfo.OrderID = Int32.Parse(orderId);
        orderInfo.DateCreated = dateCreatedTextBox.Text;
        orderInfo.DateShipped = dateShippedTextBox.Text;
        orderInfo.Verified = verifiedCheck.Checked;
        orderInfo.Completed = completedCheck.Checked;
        orderInfo.Canceled = canceledCheck.Checked;
        orderInfo.Comments = commentsTextBox.Text;
        orderInfo.CustomerName = customerNameTextBox.Text;
        orderInfo.ShippingAddress = shippingAddressTextBox.Text;
        orderInfo.CustomerEmail = customerEmailTextBox.Text;
        // try to update the order
        try
        {
            // Update the order
            OrdersAccess.Update(orderInfo);
        }
        catch (Exception ex)
        {
            // In case of an error, we simply ignore it
        }
        // Exit edit mode
        SetEditMode(false);
        // Update the form
        PopulateControls();
    }

    // mark order as verified
    protected void markVerifiedButton_Click(object sender, EventArgs e)
    {
        // obtain the order ID from the session
        string orderId = Session["AdminOrderID"].ToString();
```

```
        // mark order as verified
        OrdersAccess.MarkVerified(orderId);
        // update the form
        PopulateControls();
    }

    // Mark order as completed
    protected void markCompletedButton_Click(object sender, EventArgs
e)
    {
        // obtain the order ID from the session
        string orderId = Session["AdminOrderID"].ToString();
        // mark the order as completed
        OrdersAccess.MarkCompleted(orderId);
        // update the form
        PopulateControls();
    }

    // mark order as canceled
    protected void markCanceledButton_Click(object sender, EventArgs e)
    {
        // obtain the order ID from the session
        string orderId = Session["AdminOrderID"].ToString();
        // mark the order as canceled
        OrdersAccess.MarkCanceled(orderId);
        // update the form
        PopulateControls();
    }
}
```

### 4.3.3.9 ProductDetailsAdmin.ascx

```
<%@ Control Language="C#" AutoEventWireup="true"
CodeFile="ProductDetailsAdmin.ascx.cs" Inherits="ProductDetailsAdmin"
%>
<span class="AdminPageText">
  <asp:Label ID="productNameLabel" runat="server" CssClass="AdminTitle"
/>
  <asp:HyperLink ID="goBackLink" runat="server">(go back to
products)</asp:HyperLink>
  <br /><br />
  <asp:Label ID="statusLabel" runat="server" CssClass="AdminPageText"
Text="Product Details Loaded" ForeColor="Red"></asp:Label><br />
  <br />
Product belongs to these categories:
  <asp:Label ID="categoriesLabel" runat="server"></asp:Label>
  <br />
Remove product from this category:
  <asp:DropDownList ID="categoriesListRemove" runat="server">
  </asp:DropDownList>
  <asp:Button ID="removeButton" runat="server" Text="Go!"
OnClick="removeButton_Click" />
```

```
  <asp:Button ID="deleteButton" runat="server" Text="DELETE FROM
CATALOG" OnClick="deleteButton_Click" /><br />
Assign product to this category:
  <asp:DropDownList ID="categoriesListAssign" runat="server">
  </asp:DropDownList>
  <asp:Button ID="assignButton" runat="server" Text="Go!"
OnClick="assignButton_Click" />
  <br />
  <asp:Label ID="moveLabel" runat="server" Text="Move product to this
category:"/>
  <asp:DropDownList ID="categoriesListMove" runat="server">
  </asp:DropDownList>
  <asp:Button ID="moveButton" runat="server" Text="Go!"
OnClick="moveButton_Click" />
  <br />
Image1 file name:
  <asp:Label ID="image1FileNameLabel" runat="server"></asp:Label>
  <asp:FileUpload ID="image1FileUpload" runat="server" />
  <asp:Button ID="upload1Button" runat="server" Text="Upload"
OnClick="upload1Button_Click" /><br />
  <asp:Image ID="image1" runat="server" />
  <br />
Image2 file name:
  <asp:Label ID="image2FileNameLabel" runat="server"></asp:Label>
  <asp:FileUpload ID="image2FileUpload" runat="server" />
  <asp:Button ID="upload2Button" runat="server" Text="Upload"
OnClick="upload2Button_Click" /><br />
  <asp:Image ID="image2" runat="server" />
</span>
```

### 4.3.3.9.1 ProductDetailsAdmin.ascx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
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;

public partial class ProductDetailsAdmin : System.Web.UI.UserControl
{
    // store product, category and department IDs as class members
    private string currentProductId, currentCategoryId,
currentDepartmentId;

    protected void Page_Load(object sender, EventArgs e)
    {
        // Get DepartmentID, CategoryID, ProductID from the query
```

```
string
        // and save their values
        currentDepartmentId = Request.QueryString["DepartmentID"];
        currentCategoryId = Request.QueryString["CategoryID"];
        currentProductId = Request.QueryString["ProductID"];
        // Assign buttons to the combo boxes
        Utilities.TieButton(this.Page, categoriesListRemove,
removeButton);
        Utilities.TieButton(this.Page, categoriesListAssign,
assignButton);
        Utilities.TieButton(this.Page, categoriesListMove, moveButton);
        // Fill the controls with data only on the initial page load
        if (!IsPostBack)
        {
            // Fill controls with data
            PopulateControls();
        }
    }

    // Populate the controls
    private void PopulateControls()
    {
        // Set the "go back to products" link
        goBackLink.NavigateUrl = Request.ApplicationPath +

String.Format("/CatalogAdmin.aspx?DepartmentID={0}&CategoryID={1}",
                              currentDepartmentId, currentCategoryId);
        // Retrieve product details and category details from database
        ProductDetails productDetails =
CatalogAccess.GetProductDetails(currentProductId);
        CategoryDetails categoryDetails =
CatalogAccess.GetCategoryDetails(currentCategoryId);
        // Set up labels and images
        productNameLabel.Text = productDetails.Name;
        moveLabel.Text = "Move product from category <b>" +
categoryDetails.Name + "</b> to this category: ";
        image1.ImageUrl = Request.ApplicationPath + "/ProductImages/" +
productDetails.Image1FileName;
        image2.ImageUrl = Request.ApplicationPath + "/ProductImages/" +
productDetails.Image2FileName;
        // Clear form
        categoriesLabel.Text = "";
        categoriesListAssign.Items.Clear();
        categoriesListMove.Items.Clear();
        categoriesListRemove.Items.Clear();
        // Fill categoriesLabel and categoriesListRemove with data
        string categoryId, categoryName;
        DataTable productCategories =
CatalogAccess.GetCategoriesWithProduct(currentProductId);
        for (int i = 0; i < productCategories.Rows.Count; i++)
        {
            // obtain category id and name
            categoryId =
productCategories.Rows[i]["CategoryId"].ToString();
            categoryName =
productCategories.Rows[i]["Name"].ToString();
```

```
            // add a link to the category admin page
            categoriesLabel.Text += (categoriesLabel.Text == "" ? "" :
", ") +
                "<a href=\"" + Request.ApplicationPath +
"/CatalogAdmin.aspx" +
                "?DepartmentID=" +
CatalogAccess.GetCategoryDetails(currentCategoryId).DepartmentId +
                "&CategoryID=" + categoryId + "\">" +
                categoryName + "</a>";
            // populate the categoriesListRemove combo box
            categoriesListRemove.Items.Add(new ListItem(categoryName,
categoryId));
        }
        // Delete from catalog or remove from category?
        if (productCategories.Rows.Count > 1)
        {
            deleteButton.Visible = false;
            removeButton.Enabled = true;
        }
        else
        {
            deleteButton.Visible = true;
            removeButton.Enabled = false;
        }
        // Fill categoriesListMove and categoriesListAssign with data
        productCategories =
CatalogAccess.GetCategoriesWithoutProduct(currentProductId);
        for (int i = 0; i < productCategories.Rows.Count; i++)
        {
            // obtain category id and name
            categoryId =
productCategories.Rows[i]["CategoryId"].ToString();
            categoryName =
productCategories.Rows[i]["Name"].ToString();
            // populate the list boxes
            categoriesListAssign.Items.Add(new ListItem(categoryName,
categoryId));
            categoriesListMove.Items.Add(new ListItem(categoryName,
categoryId));
        }
    }

    // Remove the product from a category
    protected void removeButton_Click(object sender, EventArgs e)
    {
        // Check if a category was selected
        if (categoriesListRemove.SelectedIndex != -1)
        {
            // Get the category ID that was selected in the
DropDownList
            string categoryId =
categoriesListRemove.SelectedItem.Value;
            // Remove the product from the category
            bool success =
CatalogAccess.RemoveProductFromCategory(currentProductId, categoryId);
            // Display status message
```

```
            statusLabel.Text = success ? "Product removed successfully"
: "Product removal failed";
            // Refresh the page
            PopulateControls();
        }
        else
            statusLabel.Text = "You need to select a category";
    }

    // delete a product from the catalog
    protected void deleteButton_Click(object sender, EventArgs e)
    {
        // Delete the product from the catalog
        CatalogAccess.DeleteProduct(currentProductId);
        // Need to go back to the categories page now
        Response.Redirect(Request.ApplicationPath +
"/CatalogAdmin.aspx" +
                "?DepartmentID=" + currentDepartmentId +
                "&CategoryID=" + currentCategoryId);
    }

    // assign the product to a new category
    protected void assignButton_Click(object sender, EventArgs e)
    {
        // Check if a category was selected
        if (categoriesListAssign.SelectedIndex != -1)
        {
            // Get the category ID that was selected in the
DropDownList
            string categoryId =
categoriesListAssign.SelectedItem.Value;
            // Assign the product to the category
            bool success =
CatalogAccess.AssignProductToCategory(currentProductId, categoryId);
            // Display status message
            statusLabel.Text = success ? "Product assigned
successfully" : "Product assignation failed";
            // Refresh the page
            PopulateControls();
        }
        else
            statusLabel.Text = "You need to select a category";
    }

    // move the product to another category
    protected void moveButton_Click(object sender, EventArgs e)
    {
        // Check if a category was selected
        if (categoriesListMove.SelectedIndex != -1)
        {
            // Get the category ID that was selected in the
DropDownList
            string newCategoryId =
categoriesListMove.SelectedItem.Value;
            // Move the product to the category
            bool success =
```

```
CatalogAccess.MoveProductToCategory(currentProductId,
currentCategoryId, newCategoryId);
            // In case the operation was successful, reload the page,
            // so the new category will reflect in the query string
            if (!success)
                statusLabel.Text = "Couldn't move the product to the
specified category";
            else
                Response.Redirect(Request.ApplicationPath +
"/CatalogAdmin.aspx" +
                      "?DepartmentID=" + currentDepartmentId +
                      "&CategoryID=" + newCategoryId +
                      "&ProductID=" + currentProductId);
        }
        else
            statusLabel.Text = "You need to select a category";
    }

    // upload product's first image
    protected void upload1Button_Click(object sender, EventArgs e)
    {
        // proceed with uploading only if the user selected a file
        if (image1FileUpload.HasFile)
        {
            try
            {
                string fileName = image1FileUpload.FileName;
                string location = Server.MapPath("./ProductImages/") +
fileName;
                // save image to server
                image1FileUpload.SaveAs(location);
                // update database with new product details
                ProductDetails pd =
CatalogAccess.GetProductDetails(currentProductId);
                CatalogAccess.UpdateProduct(currentProductId, pd.Name,
pd.Description, pd.Price.ToString(), fileName, pd.Image2FileName,
pd.OnDepartmentPromotion.ToString(), pd.OnCatalogPromotion.ToString());
                // reload the page
                Response.Redirect(Request.ApplicationPath +
"/CatalogAdmin.aspx" +
                      "?DepartmentID=" + currentDepartmentId +
                      "&CategoryID=" + currentCategoryId +
                      "&ProductID=" + currentProductId);
            }
            catch
            {
                statusLabel.Text = "Uploading image 1 failed";
            }
        }
    }

    // upload product's second image
    protected void upload2Button_Click(object sender, EventArgs e)
    {
        // proceed with uploading only if the user selected a file
        if (image2FileUpload.HasFile)
```

```
        {
            try
            {
                string fileName = image2FileUpload.FileName;
                string location = Server.MapPath("./ProductImages/") +
fileName;
                // save image to server
                image2FileUpload.SaveAs(location);
                // update database with new product details
                ProductDetails pd =
CatalogAccess.GetProductDetails(currentProductId);
                CatalogAccess.UpdateProduct(currentProductId, pd.Name,
pd.Description, pd.Price.ToString(), pd.Image1FileName, fileName,
pd.OnDepartmentPromotion.ToString(), pd.OnCatalogPromotion.ToString());
                // reload the page
                Response.Redirect(Request.ApplicationPath +
"/CatalogAdmin.aspx" +
                          "?DepartmentID=" + currentDepartmentId +
                          "&CategoryID=" + currentCategoryId +
                          "&ProductID=" + currentProductId);
            }
            catch
            {
                statusLabel.Text = "Uploading image 2 failed";
            }
        }
    }
}
```

## 4.3.3.10 ProductRecommendation.ascx

```
<%@ Control Language="C#" AutoEventWireup="true"
CodeFile="ProductRecommendations.ascx.cs"
Inherits="ProductRecommendations" %>
<asp:Label ID="recommendationsHeader" runat="server"
CssClass="RecommendationHead" />
<asp:DataList ID="list" runat="server">
  <ItemTemplate>
    <table cellpadding="0" cellspacing="0">
      <tr>
        <td width="170px">
          <a class="RecommendationLink"
href='Product.aspx?ProductID=<%# Eval("ProductID") %>'>
            <%# Eval("Name") %>
          </a>
        </td>
        <td class="RecommendationText" valign="top">
          <%# Eval("Description") %>
        </td>
      </tr>
    </table>
  </ItemTemplate>
```

```
</asp:DataList>
```

### 4.3.3.10.1 ProductRecommendation.ascx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
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;

public partial class ProductRecommendations : System.Web.UI.UserControl
{
    protected void Page_PreRender(object sender, EventArgs e)
    {
        // Get the currently loaded page
        string currentLocation =
Request.AppRelativeCurrentExecutionFilePath;
        // If we're in Product.aspx...
        if (currentLocation == "~/Product.aspx")
        {
            // get the product ID
            string productId = Request.QueryString["ProductID"];
            // get product recommendations
            DataTable table;
            // display recommendations
            table = CatalogAccess.GetRecommendations(productId);
            list.DataSource = table;
            list.DataBind();
            // display header
            if (table.Rows.Count > 0)
                recommendationsHeader.Text =
                  "Customers who bought this product also bought:";
            else
                recommendationsHeader.Text = "";
        }
        // If we're in ShoppingCart.aspx...
        else if (currentLocation == "~/ShoppingCart.aspx")
        {
            // get product recommendations
            DataTable table;
            // display recommendations
            table = ShoppingCartAccess.GetRecommendations();
            list.DataSource = table;
            list.DataBind();
            // display header
            if (table.Rows.Count > 0)
                recommendationsHeader.Text =
```

```
                "Customers who bought these products also bought:";
            else
                recommendationsHeader.Text = "";
        }
    }
}
```

## 4.3.3.11 ProductsAdmin.ascx

```
<%@ Control Language="C#" AutoEventWireup="true"
CodeFile="ProductsAdmin.ascx.cs"
  Inherits="ProductsAdmin" %>
<asp:Label ID="statusLabel" runat="server" CssClass="AdminPageText"
Text="Products Loaded"></asp:Label><br />
<br />
<asp:Label ID="locationLabel" runat="server" CssClass="AdminPageText"
Text="Displaying products for category..."></asp:Label>
<asp:LinkButton ID="goBackLink" runat="server" CssClass="AdminPageText"
OnClick="goBackLink_Click">(go back to categories)</asp:LinkButton><br
/>
<br />
<asp:GridView ID="grid" runat="server" AutoGenerateColumns="False"
DataKeyNames="ProductID"
  Width="100%" OnRowCancelingEdit="grid_RowCancelingEdit"
OnRowEditing="grid_RowEditing" OnRowUpdating="grid_RowUpdating">
  <Columns>
    <asp:ImageField DataImageUrlField="Image1FileName"
DataImageUrlFormatString="../ProductImages/{0}"
      HeaderText="Product Image" ReadOnly="True">
    </asp:ImageField>
    <asp:TemplateField HeaderText="Product Name" SortExpression="Name">
      <ItemTemplate>
        <asp:Label ID="Label5" runat="server" Text='<%# Bind("Name")
%>'></asp:Label>
      </ItemTemplate>
      <EditItemTemplate>
        <asp:TextBox ID="nameTextBox" runat="server" Width="97%"
CssClass="GridEditingRow" Text='<%# Bind("Name") %>'></asp:TextBox>
      </EditItemTemplate>
    </asp:TemplateField>
    <asp:TemplateField HeaderText="Product Description"
SortExpression="Description">
      <ItemTemplate>
        <asp:Label ID="Label1" runat="server" Text='<%#
Bind("Description") %>'></asp:Label>
      </ItemTemplate>
      <EditItemTemplate>
        <asp:TextBox ID="descriptionTextBox" runat="server" Text='<%#
Bind("Description") %>'
          Height="100px" Width="97%" CssClass="GridEditingRow"
TextMode="MultiLine" />
      </EditItemTemplate>
```

```
    </asp:TemplateField>
    <asp:TemplateField HeaderText="Price" SortExpression="Price">
      <ItemTemplate>
        <asp:Label ID="Label2" runat="server" Text='<%#
String.Format("{0:0.00}", Eval("Price")) %>'></asp:Label>
      </ItemTemplate>
      <EditItemTemplate>
        <asp:TextBox ID="priceTextBox" runat="server" Width="45px"
Text='<%# String.Format("{0:0.00}", Eval("Price")) %>'></asp:TextBox>
      </EditItemTemplate>
    </asp:TemplateField>
    <asp:TemplateField HeaderText="Image1 File"
SortExpression="Image1FileName">
      <ItemTemplate>
        <asp:Label ID="Label3" runat="server" Text='<%#
Bind("Image1FileName") %>'></asp:Label>
      </ItemTemplate>
      <EditItemTemplate>
        <asp:TextBox ID="image1TextBox" Width="80px" runat="server"
Text='<%# Bind("Image1FileName") %>'></asp:TextBox>
      </EditItemTemplate>
    </asp:TemplateField>
    <asp:TemplateField HeaderText="Image2 File"
SortExpression="Image2FileName">
      <ItemTemplate>
        <asp:Label ID="Label4" runat="server" Text='<%#
Bind("Image2FileName") %>'></asp:Label>
      </ItemTemplate>
      <EditItemTemplate>
        <asp:TextBox ID="image2TextBox" Width="80px" runat="server"
Text='<%# Bind("Image2FileName") %>'></asp:TextBox>
      </EditItemTemplate>
    </asp:TemplateField>
    <asp:CheckBoxField DataField="OnDepartmentPromotion"
HeaderText="Dept. prom." SortExpression="OnDepartmentPromotion" />
    <asp:CheckBoxField DataField="OnCatalogPromotion" HeaderText="Cat.
prom." SortExpression="OnCatalogPromotion" />
    <asp:TemplateField>
      <ItemTemplate>
        <asp:HyperLink runat="server" Text="Select" NavigateUrl='<%#
"../CatalogAdmin.aspx?DepartmentID=" +
Request.QueryString["DepartmentID"] + "&amp;CategoryID=" +
Request.QueryString["CategoryID"] + "&amp;ProductID=" +
Eval("ProductID") %>' ID="HyperLink1">
        </asp:HyperLink>
      </ItemTemplate>
    </asp:TemplateField>
    <asp:CommandField ShowEditButton="True" />
  </Columns>
</asp:GridView>
<br />
<span class="AdminPageText">Create a new product and assign it to this
category:</span>
<table class="AdminPageText" cellspacing="0">
  <tr>
    <td width="100" valign="top">Name:</td>
```

```
    <td>
      <asp:TextBox cssClass="AdminPageText" ID="newName" Runat="server"
Width="400px" />
    </td>
  </tr>
  <tr>
    <td width="100" valign="top">Description:</td>
    <td>
      <asp:TextBox cssClass="AdminPageText" ID="newDescription"
Runat="server" Width="400px" Height="70px" TextMode="MultiLine"/>
    </td>
  </tr>
  <tr>
    <td width="100" valign="top">Price:</td>
    <td>
      <asp:TextBox cssClass="AdminPageText" ID="newPrice"
Runat="server" Width="400px">0.00</asp:TextBox>
    </td>
  </tr>
  <tr>
    <td width="100" valign="top">Image1 File:</td>
    <td>
      <asp:TextBox cssClass="AdminPageText" ID="newImage1FileName"
Runat="server" Width="400px">Generic1.png</asp:TextBox>
    </td>
  </tr>
  <tr>
    <td width="150" valign="top">Image2 File:</td>
    <td>
      <asp:TextBox cssClass="AdminPageText" ID="newImage2FileName"
Runat="server" Width="400px">Generic2.png</asp:TextBox>
    </td>
  </tr>
  <tr>
    <td width="150" valign="top">Dept. Promotion:</td>
    <td>
      <asp:CheckBox ID="newOnDepartmentPromotion" Runat="server" />
    </td>
  </tr>
  <tr>
    <td width="150" valign="top">Catalog Promotion:</td>
    <td>
      <asp:CheckBox ID="newOnCatalogPromotion" Runat="server" />
    </td>
  </tr>
</table>
<asp:Button ID="createProduct" CssClass="AdminButtonText"
  Runat="server" Text="Create Product" OnClick="createProduct_Click" />
```

### 4.3.3.11.1 ProductsAdmin.ascx.cs

```
using System;
```

```
using System.Data;
using System.Configuration;
using System.Collections;
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;

public partial class ProductsAdmin : System.Web.UI.UserControl
{
    protected void Page_Load(object sender, EventArgs e)
    {
        // Load the grid only the first time the page is loaded
        if (!Page.IsPostBack)
        {
            // Get CategoryID from the query string
            string categoryId = Request.QueryString["CategoryID"];
            // Obtain the category's name
            CategoryDetails cd =
CatalogAccess.GetCategoryDetails(categoryId);
            string categoryName = cd.Name;
            // Set controls' properties
            statusLabel.ForeColor = System.Drawing.Color.Red;
            locationLabel.Text = "Displaying products for category <b>
"
                                 + categoryName + "</b>";
            // Load the products grid
            BindGrid();
        }
    }

    // Populate the GridView with data
    private void BindGrid()
    {
        // Get CategoryID from the query string
        string categoryId = Request.QueryString["CategoryID"];
        // Get a DataTable object containing the products
        grid.DataSource =
CatalogAccess.GetAllProductsInCategory(categoryId);
        // Needed to bind the data bound controls to the data source
        grid.DataBind();
    }

    // Enter row into edit mode
    protected void grid_RowEditing(object sender, GridViewEditEventArgs
e)
    {
        // Set the row for which to enable edit mode
        grid.EditIndex = e.NewEditIndex;
        // Set status message
        statusLabel.Text = "Editing row # " +
e.NewEditIndex.ToString();
        // Reload the grid
        BindGrid();
```

```
    }

    // Cancel edit mode
    protected void grid_RowCancelingEdit(object sender,
GridViewCancelEditEventArgs e)
    {
        // Cancel edit mode
        grid.EditIndex = -1;
        // Set status message
        statusLabel.Text = "Editing canceled";
        // Reload the grid
        BindGrid();
    }

    // Update a product
    protected void grid_RowUpdating(object sender,
GridViewUpdateEventArgs e)
    {
        // Retrieve updated data
        string id = grid.DataKeys[e.RowIndex].Value.ToString();
        string name =
((TextBox)grid.Rows[e.RowIndex].FindControl("nameTextBox")).Text;
        string description =
((TextBox)grid.Rows[e.RowIndex].FindControl("descriptionTextBox")).Text
;
        string price =
((TextBox)grid.Rows[e.RowIndex].FindControl("priceTextBox")).Text;
        string image1FileName =
((TextBox)grid.Rows[e.RowIndex].FindControl("image1TextBox")).Text;
        string image2FileName =
((TextBox)grid.Rows[e.RowIndex].FindControl("image2TextBox")).Text;
        string onDepartmentPromotion =
((CheckBox)grid.Rows[e.RowIndex].Cells[6].Controls[0]).Checked.ToString
();
        string onCatalogPromotion =
((CheckBox)grid.Rows[e.RowIndex].Cells[7].Controls[0]).Checked.ToString
();
        // Execute the update command
        bool success = CatalogAccess.UpdateProduct(id, name,
description, price, image1FileName, image2FileName,
onDepartmentPromotion, onCatalogPromotion);
        // Cancel edit mode
        grid.EditIndex = -1;
        // Display status message
        statusLabel.Text = success ? "Product update successful" :
"Product update failed";
        // Reload grid
        BindGrid();
    }

    // Create a new product
    protected void createProduct_Click(object sender, EventArgs e)
    {
        // Get CategoryID from the query string
        string categoryId = Request.QueryString["CategoryID"];
        // Execute the insert command
```

```
        bool success = CatalogAccess.CreateProduct(categoryId,
newName.Text, newDescription.Text, newPrice.Text,
newImage1FileName.Text, newImage2FileName.Text,
newOnDepartmentPromotion.Checked.ToString(),
newOnCatalogPromotion.Checked.ToString());
        // Display status message
        statusLabel.Text = success ? "Insert successful" : "Insert
failed";
        // Reload the grid
        BindGrid();
    }

    // Go back to the list of categories
    protected void goBackLink_Click(object sender, EventArgs e)
    {
        // Get DepartmentID from the query string
        string departmentId = Request.QueryString["DepartmentID"];
        // Redirect
        Response.Redirect(Request.ApplicationPath +
"/CatalogAdmin.aspx?DepartmentID=" + departmentId);
    }
}
```

## 4.3.3.12 ProductsList.ascx

```
<%@ Control Language="C#" AutoEventWireup="true"
CodeFile="ProductsList.ascx.cs"
  Inherits="ProductsList" %>
<asp:Label ID="pagingLabel" runat="server" CssClass="PagingText"
Visible="false" />
  
<asp:HyperLink ID="previousLink" runat="server" CssClass="PagingText"
Visible="false">Previous</asp:HyperLink>
  
<asp:HyperLink ID="nextLink" runat="server" CssClass="PagingText"
Visible="false">Next</asp:HyperLink>
<asp:DataList ID="list" Runat="server" RepeatColumns="2"
EnableViewState="False" OnItemCommand="list_ItemCommand">
  <ItemTemplate>
    <table cellPadding="0" align="left">
      <tr height="105">
        <td align="center" width="110">
          <a href='Product.aspx?ProductID=<%# Eval("ProductID")%>'>
            <img width="100" src='ProductImages/<%#
Eval("Image1FileName") %>' border="0"/>
          </a>
        </td>
        <td vAlign="top" width="250">
          <a class="ProductName" href='Product.aspx?ProductID=<%#
Eval("ProductID")%>'>
            <%# Eval("Name") %>
          </a>
```

```
          <br/>
          <span class="ProductDescription">
            <%# Eval("Description") %>
            <br/><br/>
            Price:
          </span>
          <span class="ProductPrice">
            <%# Eval("Price", "{0:c}") %>
          </span>
          <br />
          <asp:Button ID="addToCartButton" runat="server" Text="Add to
Cart" CommandArgument='<%# Eval("ProductID") %>'
CssClass="SmallButtonText"/>
        </td>
      </tr>
    </table>
  </ItemTemplate>
</asp:DataList>
```

## 4.3.3.12.1 ProductsList.ascx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
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.Collections.Specialized;

public partial class ProductsList : System.Web.UI.UserControl
{
    protected void Page_Load(object sender, EventArgs e)
    {
        PopulateControls();
    }

    private void PopulateControls()
    {
        // Retrieve DepartmentID from the query string
        string departmentId = Request.QueryString["DepartmentID"];
        // Retrieve CategoryID from the query string
        string categoryId = Request.QueryString["CategoryID"];
        // Retrieve Page from the query string
        string page = Request.QueryString["Page"];
        if (page == null) page = "1";
        // Retrieve Search string from query string
        string searchString = Request.QueryString["Search"];
        // How many pages of products?
```

```
        int howManyPages = 1;
        // If performing a product search
        if (searchString != null)
        {
            // Retrieve AllWords from query string
            string allWords = Request.QueryString["AllWords"];
            // Perform search
            list.DataSource = CatalogAccess.Search(searchString,
allWords, page, out howManyPages);
            list.DataBind();
        }
        // If browsing a category...
        else if (categoryId != null)
        {
            // Retrieve list of products in a category
            list.DataSource =
CatalogAccess.GetProductsInCategory(categoryId, page, out
howManyPages);
            list.DataBind();
        }
        else if (departmentId != null)
        {
            // Retrieve list of products on department promotion
            list.DataSource =
CatalogAccess.GetProductsOnDepartmentPromotion(departmentId, page, out
howManyPages);
            list.DataBind();
        }
        else
        {
            // Retrieve list of products on catalog promotion
            list.DataSource =
CatalogAccess.GetProductsOnCatalogPromotion(page, out howManyPages);
            list.DataBind();
        }
        // display paging controls
        if (howManyPages > 1)
        {
            // have the current page as integer
            int currentPage = Int32.Parse(page);
            // make controls visible
            pagingLabel.Visible = true;
            previousLink.Visible = true;
            nextLink.Visible = true;
            // set the paging text
            pagingLabel.Text = "Page " + page + " of " +
howManyPages.ToString();
            // create the Previous link
            if (currentPage == 1)
                previousLink.Enabled = false;
            else
            {
                NameValueCollection query = Request.QueryString;
                string paramName, newQueryString = "?";
                for (int i = 0; i < query.Count; i++)
                    if (query.AllKeys[i] != null)
```

```
                        if ((paramName =
query.AllKeys[i].ToString()).ToUpper() != "PAGE")
                            newQueryString += paramName + "=" +
query[i] + "&";
                previousLink.NavigateUrl = Request.Url.AbsolutePath +
newQueryString + "Page=" + (currentPage - 1).ToString();
            }
            // create the Next link
            if (currentPage == howManyPages)
                nextLink.Enabled = false;
            else
            {
                NameValueCollection query = Request.QueryString;
                string paramName, newQueryString = "?";
                for (int i = 0; i < query.Count; i++)
                    if (query.AllKeys[i] != null)
                        if ((paramName =
query.AllKeys[i].ToString()).ToUpper() != "PAGE")
                            newQueryString += paramName + "=" +
query[i] + "&";
                nextLink.NavigateUrl = Request.Url.AbsolutePath +
newQueryString + "Page=" + (currentPage + 1).ToString();
            }
        }
    }

    // fires when an Add to Cart button is clicked
    protected void list_ItemCommand(object source,
DataListCommandEventArgs e)
    {
        // The CommandArgument of the clicked Button contains the
ProductID
        string productId = e.CommandArgument.ToString();
        // Add the product to the shopping cart
        ShoppingCartAccess.AddItem(productId);
    }
}
```

## 4.3.3.13 SearchBox.ascx

```
<%@ Control Language="C#" AutoEventWireup="true"
CodeFile="SearchBox.ascx.cs" Inherits="SearchBox" %>
<table border="0" cellpadding="0" cellspacing="0" width="200px">
  <tr>
    <td class="SearchBoxHead">
      Search the Catalog
    </td>
  </tr>
  <tr>
    <td class="SearchBoxContent">
      <asp:TextBox ID="searchTextBox" Runat="server" Width="128px"
CssClass="SearchBox" BorderStyle="Dotted" MaxLength="100" Height="16px"
/>
      <asp:Button ID="goButton" Runat="server" CssClass="SearchBox"
```

```
Text="Go!" Width="36px" Height="21px" OnClick="goButton_Click" /><br />
      <asp:CheckBox ID="allWordsCheckBox" CssClass="SearchBox"
Runat="server" Text="Search for all words" />
    </td>
  </tr>
</table>
```

### 4.3.3.13.1 SearchBox.ascx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
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;

public partial class SearchBox : System.Web.UI.UserControl
{
  protected void Page_Load(object sender, EventArgs e)
  {
    // don't repopulate control on postbacks
    if (!IsPostBack)
    {
      // tie the search text box to the Go button
      Utilities.TieButton(this.Page, searchTextBox, goButton);
      // load search box controls' values
      string allWords = Request.QueryString["AllWords"];
      string searchString = Request.QueryString["Search"];
      if (allWords != null)
        allWordsCheckBox.Checked = (allWords.ToUpper() == "TRUE");
      if (searchString != null)
        searchTextBox.Text = searchString;
    }
  }

  // Perform the product search
  protected void goButton_Click(object sender, EventArgs e)
  {
    ExecuteSearch();
  }

  // Redirect to the search results page
  private void ExecuteSearch()
  {
    if (searchTextBox.Text.Trim() != "")
      Response.Redirect(Request.ApplicationPath +
                  "/Search.aspx?Search=" + searchTextBox.Text +
                  "&AllWords=" + allWordsCheckBox.Checked.ToString());
```

```
    }
}
```

### 4.3.3.14 UserInfo.ascx

```
<%@ Control Language="C#" AutoEventWireup="true"
CodeFile="UserInfo.ascx.cs" Inherits="UserInfo" %>
<table cellspacing="0" border="0" width="200px"
class="UserInfoContent">
  <tr>
    <td class="UserInfoHead">
      User Info</td>
  </tr>
  <asp:LoginView ID="LoginView1" runat="server">
    <AnonymousTemplate>
      <tr>
        <td>
          <span class="UserInfoText">You are not logged in.</span>
        </td>
      </tr>
      <tr>
        <td>
           &raquo;
          <asp:LoginStatus ID="LoginStatus1" runat="server"
            CssClass="UserInfoLink" />
           &laquo;
        </td>
      </tr>
      <tr>
        <td>
           &raquo;
          <asp:HyperLink runat="server" ID="registerLink"
            NavigateUrl="~/Register.aspx" Text="Register"
            ToolTip="Go to the registration page"
            CssClass="UserInfoLink" />
           &laquo;
        </td>
      </tr>
    </AnonymousTemplate>
    <RoleGroups>
      <asp:RoleGroup Roles="Customers">
        <ContentTemplate>
          <tr>
            <td>
              <asp:LoginName ID="LoginName1" runat="server"
                FormatString="You are logged in as <b>{0}</b>. "
                CssClass="UserInfoText" />
            </td>
          </tr>
          <tr>
            <td>
               &raquo;
```

```
            <asp:LoginStatus ID="LoginStatus1" runat="server"
              CssClass="UserInfoLink" />
             &laquo;
          </td>
        </tr>
        <tr>
          <td>
             &raquo;
            <asp:HyperLink runat="server" ID="detailsLink"
              NavigateUrl="~/CustomerDetails.aspx"
              Text="Edit Details"
              ToolTip="Edit your personal details"
              CssClass="UserInfoLink" />
             &laquo;
          </td>
        </tr>
      </ContentTemplate>
    </asp:RoleGroup>
    <asp:RoleGroup Roles="Administrators">
      <ContentTemplate>
        <tr>
          <td>
            <asp:LoginName ID="LoginName2" runat="server"
FormatString="You are logged in as <b>{0}</b>. "
              CssClass="UserInfoText" />
          </td>
        </tr>
        <tr>
          <td>
             &raquo;
              <asp:LoginStatus ID="LoginStatus2" runat="server"
CssClass="UserInfoLink" />
             &laquo;
          </td>
        </tr>
        <tr>
          <td>
             &raquo;
              <a class="UserInfoLink"
href="CatalogAdmin.aspx">Catalog Admin</a>
             &laquo;
          </td>
        </tr>
        <tr>
          <td>
             &raquo;
              <a class="UserInfoLink"
href="ShoppingCartAdmin.aspx">Shopping Cart Admin</a>
             &laquo;
          </td>
        </tr>
        <tr>
          <td>
             &raquo;
              <a class="UserInfoLink" href="OrdersAdmin.aspx">Orders
Admin</a>
```

```
                 &laquo;
              </td>
            </tr>
          </ContentTemplate>
        </asp:RoleGroup>
      </RoleGroups>
    </asp:LoginView>
</table>
```

### 4.3.3.14.1 UserInfo.ascx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
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;

public partial class UserInfo : System.Web.UI.UserControl
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
}
```

## 4.3.4 Style Sheet (CSS)

Τα Style Sheet (CSS) είναι αρχεία που χρησιμοποιούν την τεχνολογία CSS και καθορίζουν την μορφή που θα έχει η εφαρμογή μας. Καορίζουν παραμέτρους όπως το χρώμα, το φόντο, την στοίχιση κ.α.

### 4.3.4.1 BalloonShop.css

```
.DepartmentListHead
{
  border-right: #01a647 1px solid;
  border-top: #01a647 1px solid;
  border-left: #01a647 1px solid;
  border-bottom: #01a647 1px solid;
  background-color: #30b86e;
```

```
  font-family: Verdana, Arial;
  font-weight: bold;
  font-size: 10pt;
  color: #f5f5dc;
  padding-left: 3px;
  text-align: center;
}
.DepartmentListContent
{
  border-right: #01a647 1px solid;
  border-top: #01a647 1px solid;
  border-left: #01a647 1px solid;
  border-bottom: #01a647 1px solid;
  background-color: #9fe1bb;
  text-align: center;
}
a.DepartmentUnselected
{
  font-family: Verdana, Arial;
  font-weight: bold;
  font-size: 9pt;
  color: #5f9ea0;
  line-height: 25px;
  padding-left: 5px;
  text-decoration: none;
}
a.DepartmentUnselected:hover
{
  padding-left: 5px;
  color: #2e8b57;
}
a.DepartmentSelected
{
  font-family: Verdana, Arial;
  font-weight: bold;
  font-size: 9pt;
  color: #556b2f;
  line-height: 25px;
  padding-left: 5px;
  text-decoration: none;
}
.CategoryListHead
{
  border-right: #ea6d00 1px solid;
  border-top: #ea6d00 1px solid;
  border-left: #ea6d00 1px solid;
  border-bottom: #ea6d00 1px solid;
  background-color: #ef8d0e;
  font-family: Verdana, Arial;
  font-weight: bold;
  font-size: 10pt;
  color: #f5f5dc;
  text-align: center;
}
.CategoryListContent
{
```

```
  border-right: #ea6d00 1px solid;
  border-top: #ea6d00 1px solid;
  border-left: #ea6d00 1px solid;
  border-bottom: #ea6d00 1px solid;
  background-color: #f8c78c;
  text-align: center;
}
a.CategoryUnselected
{
  font-family: Verdana, Arial;
  font-weight: bold;
  font-size: 9pt;
  color: #cd853f;
  line-height: 25px;
  padding-right: 5px;
  padding-left: 5px;
  text-decoration: none
}
a.CategoryUnselected:hover
{
  color: #d2691e;
  padding-right: 5px;
  padding-left: 5px
}
a.CategorySelected
{
  font-family: Verdana, Arial;
  font-weight: bold;
  font-size: 9pt;
  color: #a0522d;
  line-height: 25px;
  padding-right: 5px;
  padding-left: 5px;
  text-decoration: none
}
.CatalogTitle
{
  color: red;
  font-family: 'Trebuchet MS', Comic Sans MS, Arial;
  font-size: 24px;
  font-weight: bold;
}
.CatalogDescription
{
  color: Black;
  font-family: Verdana, Helvetica, sans-serif;
  font-weight: bold;
  font-size: 14px;
}
a.ProductName
{
  color: Red;
  font-family: 'Trebuchet MS';
  text-decoration: none;
  font-weight: bold;
  font-size: 12px;
```

```
}
a.ProductName:hover
{
  text-decoration: underline;
}
.ProductDescription
{
  color: Black;
  font-family: Verdana, Helvetica, sans-serif;
  font-size: 11px;
}
.ProductPrice
{
  color: Black;
  font-family: Verdana, Helvetica, sans-serif;
  font-weight: bold;
  font-size: 11px;
}
.PagingText
{
  font-family: Verdana, Helvetica, sans-serif;
  font-size: 11px;
  color: Black;
}
.ProductTitle
{
  color: Blue;
  font-family: Verdana, Helvetica, sans-serif;
  text-decoration: none;
  font-size: 24px;
  font-weight: bold;
  line-height: 15px;
}
.SearchBoxHead
{
  border-right: #0468a4 1px solid;
  border-top: #0468a4 1px solid;
  border-bottom: #0468a4 1px solid;
  border-left: #0468a4 1px solid;
  background-color: #0583b5;
  font-family: Verdana, Arial;
  font-weight: bold;
  font-size: 10pt;
  color: #f5f5dc;
  text-align: center;
}
.SearchBoxContent
{
  border-right: #0468a4 1px solid;
  border-top: #0468a4 1px solid;
  border-left: #0468a4 1px solid;
  border-bottom: #0468a4 1px solid;
  background-color: #8bc8dd;
  font-family: Arial, Verdana;
  font-size: 9pt;
  color: darkblue;
```

```
  padding-top: 5px;
  padding-left: 12px;
  padding-bottom: 5px;
}
.SearchBox
{
  font-family: Verdana, Helvetica, sans-serif;
  font-size: 9pt;
  margin-bottom: 5px;
}
.UserInfoHead
{
  border-right: #cc6666 1px solid;
  border-top: #cc6666 1px solid;
  border-left: #cc6666 1px solid;
  border-bottom: #cc6666 1px solid;
  background-color: #dc143c;
  font-family: Verdana, Arial;
  font-weight: bold;
  font-size: 10pt;
  color: #f5f5dc;
  padding-left: 3px;
  text-align: center;
}
.UserInfoContent
{
  border-right: #cc6666 1px solid;
  border-top: #cc6666 1px solid;
  border-left: #cc6666 1px solid;
  border-bottom: #cc6666 1px solid;
  background-color: #ffcccc;
  text-align: center;
}
.UserInfoText
{
  font-family: Verdana, Arial;
  font-size: 9pt;
  padding-left: 5px;
  text-decoration: none;
}
a.UserInfoLink
{
  font-family: Verdana, Arial;
  font-weight: bold;
  font-size: 9pt;
  color: #ed486d;
  line-height: 15px;
  padding-left: 5px;
  text-decoration: none;
}
a.UserInfoLink:hover
{
  padding-left: 5px;
  color: #dc143c;
}
.Button
```

```
{
  color: Black;
  font-family: Verdana, Helvetica, sans-serif;
  font-size: 12px;
}
.AdminTitle
{
  color: Black;
  font-family: Verdana, Helvetica, sans-serif;
  text-decoration: none;
  font-size: 21px;
  font-weight: bold;
  line-height: 25px;
}
.AdminPageText
{
  color: Navy;
  font-family: Verdana, Helvetica, sans-serif;
  text-decoration: none;
  font-size: 12px;
}
a.AdminPageText
{
  color: Navy;
  font-family: Verdana, Helvetica, sans-serif;
  text-decoration: none;
  font-size: 12px;
}
a.AdminPageText:hover
{
  color:Red;
}
.Grid
{
  border-color: #E7E7FF;
  width: 100%;
}
.GridHeader
{
  color: White;
  background-color: Navy;
  font-family: Verdana, Helvetica, sans-serif;
  text-decoration: none;
  font-size: 11px;
  text-align: left;
}
.GridRow
{
  color: Navy;
  background-color: #E7E7FF;
  font-family: Verdana, Helvetica, sans-serif;
  text-decoration: none;
  font-size: 11px;
  text-align: left;
}
.GridEditingRow
```

```
{
  color: Navy;
  font-family: Verdana, Helvetica, sans-serif;
  text-decoration: none;
  font-size: 11px;
  text-align: left;
}
.GridAlternateRow
{
  color: Navy;
  background-color: #F7F7F7;
  font-family: Verdana, Helvetica, sans-serif;
  text-decoration: none;
  font-size: 11px;
  text-align: left;
}
.SmallButtonText
{
  color: Black;
  font-family: Verdana, Helvetica, sans-serif;
  font-size: 10px;
}
.CartSummary
{
  border-right: #0468a4 2px solid;
  border-top: #0468a4 2px solid;
  border-left: #0468a4 2px solid;
  border-bottom: #0468a4 2px solid;
  background-color: snow;
  font-family: Arial;
  font-size: 9pt;
  color: Navy;
  padding-top: 3px;
  padding-left: 2px;
  padding-bottom: 5px;
}
a.CartLink
{
  color: Black;
  font-family: Arial;
  text-decoration: none;
  font-size: 12px;
}
a.CartLink:hover
{
  color: Red;
}
.ShoppingCartTitle
{
  color: Red;
  font-family: Verdana, Helvetica, sans-serif;
  font-size: 16px;
}
.AdminErrorText
{
  font-weight: bold;
```

```
  font-size: 12px;
  color: red;
  font-style: italic;
  font-family: Verdana, Helvetica, sans-serif;
}
.RecommendationHead
{
  color: Black;
  font-family: Verdana, Helvetica, sans-serif;
  font-weight: bold;
  font-size: 10px;
}
.RecommendationText
{
  color: Black;
  font-family: Verdana, Helvetica, sans-serif;
  font-size: 10px;
}
.RecommendationLink
{
  color: Black;
  font-family: Verdana, Helvetica, sans-serif;
  text-decoration: underline;
  font-size: 10px;
}
a.RecommendationLink:hover
{
  color: Red;
}
.UserDetailsTable
{
    width: 100%;
    background-color: #ccccff;
    font-family: Verdana, Helvetica, sans-serif;
    font-size: 12px;
    border: Solid 2px Navy;
    line-height: 25px;
}
.UserDetailsTableHead
{
    border-bottom: Navy 2px solid;
    background-color: #666699;
    font-family: Verdana, Arial;
    font-weight: bold;
    font-size: 10pt;
    color: #eeeeff;
    padding-left: 3px;
    text-align: center;
}
.InfoText
{
    font-family: Verdana, Helvetica, sans-serif;
    font-size: 12px;
}
```

### 4.3.5 Skin File

Το **Skin File καθορίζει το στύλ των γραμμών τη εφαρμογή μας.**

### 4.3.5.1 BalloonShop.skin

```
<%--
Default skin template. The following skins are provided as examples
only.

1. Named control skin. The SkinId should be uniquely defined because
   duplicate SkinId's per control type are not allowed in the same
theme.

<asp:GridView runat="server" SkinId="gridviewSkin" BackColor="White" >
   <AlternatingRowStyle BackColor="Blue" />
</asp:GridView>

2. Default skin. The SkinId is not defined. Only one default
   control skin per control type is allowed in the same theme.

<asp:Image runat="server" ImageUrl="~/images/image1.jpg" />
--%>

<asp:GridView runat="server"  CellPadding="4"
AutoGenerateColumns="False">
  <SelectedRowStyle BackColor="#738A9C" Font-Bold="True"
ForeColor="#F7F7F7" />
  <HeaderStyle CssClass="GridHeader" />
  <RowStyle CssClass="GridRow" />
  <AlternatingRowStyle CssClass="GridAlternateRow" />
</asp:GridView>
```

### 4.3.6 Class File

Τα Class File είναι απο τα πιο σημαντικά αρχεία, όπως έχουμε προαναφέρει σε προηγούμενη ενότητα. Θα συμπληρώσουμε ακόμα εδώ, ένα λόγο που είναι τόσο σημαντικά, οτι δηλαδή ''καλούν'' δαδικασίες (Stored Procedures) απ' την βάση δεδομένων για να υλοποίησουν διάφορες λειτουργίες που σχετίζονται με την Β.Δ.

### 4.3.6.1 BalloonShopConfiguration.cs

```
using System;
using System.Configuration;

/// <summary>
/// Repository for BalloonShop configuration settings
/// </summary>
public static class BalloonShopConfiguration
{
    // Store the number of days for shopping cart expiration
    private readonly static int cartPersistDays;
    // Caches the connection string
    private readonly static string dbConnectionString;
    // Caches the data provider name
    private readonly static string dbProviderName;
    // Store the number of products per page
    private readonly static int productsPerPage;
    // Store the product description length for product lists
    private readonly static int productDescriptionLength;
    // Store the name of your shop
    private readonly static string siteName;

    // Initialize various properties in the constructor
    static BalloonShopConfiguration()
    {
        cartPersistDays =
Int32.Parse(ConfigurationManager.AppSettings["CartPersistDays"]);
        dbConnectionString =
ConfigurationManager.ConnectionStrings["BalloonShopConnection"].Connect
ionString;
        dbProviderName =
ConfigurationManager.ConnectionStrings["BalloonShopConnection"].Provide
rName;
        productsPerPage =
Int32.Parse(ConfigurationManager.AppSettings["ProductsPerPage"]);
        productDescriptionLength =
Int32.Parse(ConfigurationManager.AppSettings["ProductDescriptionLength"
]);
        siteName = ConfigurationManager.AppSettings["SiteName"];
    }

    // Returns the number of days for shopping cart expiration
    public static int CartPersistDays
    {
        get
        {
            return cartPersistDays;
        }
    }

    // Returns the connection string for the BalloonShop database
    public static string DbConnectionString
    {
        get
```

```
        {
            return dbConnectionString;
        }
    }

    // Returns the data provider name
    public static string DbProviderName
    {
        get
        {
            return dbProviderName;
        }
    }

    // Returns the maximum number of products to be displayed on a page
    public static int ProductsPerPage
    {
        get
        {
            return productsPerPage;
        }
    }

    // Returns the length of product descriptions in products lists
    public static int ProductDescriptionLength
    {
        get
        {
            return productDescriptionLength;
        }
    }

    // Returns the address of the mail server
    public static string MailServer
    {
        get
        {
            return ConfigurationManager.AppSettings["MailServer"];
        }
    }

    // Send error log emails?
    public static bool EnableErrorLogEmail
    {
        get
        {
            return
bool.Parse(ConfigurationManager.AppSettings["EnableErrorLogEmail"]);
        }
    }

    // Returns the email address where to send error reports
    public static string ErrorLogEmail
    {
        get
        {
```

```
            return ConfigurationManager.AppSettings["ErrorLogEmail"];
        }
    }

    // Returns the length of product descriptions in products lists
    public static string SiteName
    {
        get
        {
            return siteName;
        }
    }
}
```

### 4.3.6.2 CatalogAccess.cs

```
using System;
using System.Data;
using System.Data.Common;

/// <summary>
/// Wraps department details data
/// </summary>
public struct DepartmentDetails
{
    public string Name;
    public string Description;
}

/// <summary>
/// Wraps category details data
/// </summary>
public struct CategoryDetails
{
    public int DepartmentId;
    public string Name;
    public string Description;
}

/// <summary>
/// Wraps product details data
/// </summary>
public struct ProductDetails
{
    public string Name;
    public string Description;
    public decimal Price;
    public string Image1FileName;
    public string Image2FileName;
    public bool OnDepartmentPromotion;
    public bool OnCatalogPromotion;
}
```

```
/// <summary>
/// Product catalog business tier component
/// </summary>
public static class CatalogAccess
{
    static CatalogAccess()
    {
        //
        // TODO: Add constructor logic here
        //
    }

    // Retrieve the list of departments
    public static DataTable GetDepartments()
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "GetDepartments";
        // execute the stored procedure and return the results
        return GenericDataAccess.ExecuteSelectCommand(comm);
    }

    // get department details
    public static DepartmentDetails GetDepartmentDetails(string
departmentId)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "GetDepartmentDetails";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@DepartmentID";
        param.Value = departmentId;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // execute the stored procedure
        DataTable table = GenericDataAccess.ExecuteSelectCommand(comm);
        // wrap retrieved data into a DepartmentDetails object
        DepartmentDetails details = new DepartmentDetails();
        if (table.Rows.Count > 0)
        {
            details.Name = table.Rows[0]["Name"].ToString();
            details.Description =
table.Rows[0]["Description"].ToString();
        }
        // return department details
        return details;
    }

    // Get category details
    public static CategoryDetails GetCategoryDetails(string categoryId)
    {
        // get a configured DbCommand object
```

```
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "GetCategoryDetails";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@CategoryID";
        param.Value = categoryId;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // execute the stored procedure
        DataTable table = GenericDataAccess.ExecuteSelectCommand(comm);
        // wrap retrieved data into a CategoryDetails object
        CategoryDetails details = new CategoryDetails();
        if (table.Rows.Count > 0)
        {
            details.DepartmentId =
Int32.Parse(table.Rows[0]["DepartmentID"].ToString());
            details.Name = table.Rows[0]["Name"].ToString();
            details.Description =
table.Rows[0]["Description"].ToString();
        }
        // return department details
        return details;
    }

    // Get product details
    public static ProductDetails GetProductDetails(string productId)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "GetProductDetails";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@ProductID";
        param.Value = productId;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // execute the stored procedure
        DataTable table = GenericDataAccess.ExecuteSelectCommand(comm);
        // wrap retrieved data into a ProductDetails object
        ProductDetails details = new ProductDetails();
        if (table.Rows.Count > 0)
        {
            // get the first table row
            DataRow dr = table.Rows[0];
            // get product details
            details.Name = dr["Name"].ToString();
            details.Description = dr["Description"].ToString();
            details.Price = Decimal.Parse(dr["Price"].ToString());
            details.Image1FileName = dr["Image1FileName"].ToString();
            details.Image2FileName = dr["Image2FileName"].ToString();
            details.OnDepartmentPromotion =
bool.Parse(dr["OnDepartmentPromotion"].ToString());
            details.OnCatalogPromotion =
bool.Parse(dr["OnCatalogPromotion"].ToString());
```

```
        }
        // return department details
        return details;
    }

    // retrieve the list of categories in a department
    public static DataTable GetCategoriesInDepartment(string
departmentId)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "GetCategoriesInDepartment";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@DepartmentID";
        param.Value = departmentId;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // execute the stored procedure
        return GenericDataAccess.ExecuteSelectCommand(comm);
    }

    // Retrieve the list of products on catalog promotion
    public static DataTable GetProductsOnCatalogPromotion(string
pageNumber, out int howManyPages)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "GetProductsOnCatalogPromotion";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@DescriptionLength";
        param.Value =
BalloonShopConfiguration.ProductDescriptionLength;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@PageNumber";
        param.Value = pageNumber;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@ProductsPerPage";
        param.Value = BalloonShopConfiguration.ProductsPerPage;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@HowManyProducts";
        param.Direction = ParameterDirection.Output;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
```

```
        // execute the stored procedure and save the results in a
DataTable
        DataTable table = GenericDataAccess.ExecuteSelectCommand(comm);
        // calculate how many pages of products and set the out
parameter
        int howManyProducts =
Int32.Parse(comm.Parameters["@HowManyProducts"].Value.ToString());
        howManyPages = (int)Math.Ceiling((double)howManyProducts /

(double)BalloonShopConfiguration.ProductsPerPage);
        // return the page of products
        return table;
    }

    // retrieve the list of products featured for a department
    public static DataTable GetProductsOnDepartmentPromotion(string
departmentId, string pageNumber, out int howManyPages)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "GetProductsOnDepartmentPromotion";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@DepartmentID";
        param.Value = departmentId;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@DescriptionLength";
        param.Value =
BalloonShopConfiguration.ProductDescriptionLength;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@PageNumber";
        param.Value = pageNumber;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@ProductsPerPage";
        param.Value = BalloonShopConfiguration.ProductsPerPage;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@HowManyProducts";
        param.Direction = ParameterDirection.Output;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // execute the stored procedure and save the results in a
DataTable
        DataTable table = GenericDataAccess.ExecuteSelectCommand(comm);
```

```
        // calculate how many pages of products and set the out
parameter
        int howManyProducts =
Int32.Parse(comm.Parameters["@HowManyProducts"].Value.ToString());
        howManyPages = (int)Math.Ceiling((double)howManyProducts /

(double)BalloonShopConfiguration.ProductsPerPage);
        // return the page of products
        return table;
    }

    // retrieve the list of products in a category
    public static DataTable GetProductsInCategory(string categoryId,
string pageNumber, out int howManyPages)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "GetProductsInCategory";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@CategoryID";
        param.Value = categoryId;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@DescriptionLength";
        param.Value =
BalloonShopConfiguration.ProductDescriptionLength;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@PageNumber";
        param.Value = pageNumber;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@ProductsPerPage";
        param.Value = BalloonShopConfiguration.ProductsPerPage;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@HowManyProducts";
        param.Direction = ParameterDirection.Output;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // execute the stored procedure and save the results in a
DataTable
        DataTable table = GenericDataAccess.ExecuteSelectCommand(comm);
        // calculate how many pages of products and set the out
parameter
        int howManyProducts =
```

```
Int32.Parse(comm.Parameters["@HowManyProducts"].Value.ToString());
        howManyPages = (int)Math.Ceiling((double)howManyProducts /

(double)BalloonShopConfiguration.ProductsPerPage);
        // return the page of products
        return table;
    }

    // Search the product catalog
    public static DataTable Search(string searchString, string
allWords, string pageNumber, out int howManyPages)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "SearchCatalog";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@DescriptionLength";
        param.Value =
BalloonShopConfiguration.ProductDescriptionLength;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@AllWords";
        param.Value = allWords.ToUpper() == "TRUE" ? "True" : "False";
        param.DbType = DbType.Boolean;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@PageNumber";
        param.Value = pageNumber;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@ProductsPerPage";
        param.Value = BalloonShopConfiguration.ProductsPerPage;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@HowManyResults";
        param.Direction = ParameterDirection.Output;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);

        // define the maximum number of words
        int howManyWords = 5;
        // transform search string into array of words
        char[] wordSeparators = new char[] { ',', ';', '.', '!', '?',
'-', ' ' };
        string[] words = searchString.Split(wordSeparators,
StringSplitOptions.RemoveEmptyEntries);
        int index = 1;
```

```
        // add the words as stored procedure parameters
        for (int i = 0; i <= words.GetUpperBound(0) && index <=
howManyWords; i++)
            // ignore short words
            if (words[i].Length > 2)
            {
                // create the @Word parameters
                param = comm.CreateParameter();
                param.ParameterName = "@Word" + index.ToString();
                param.Value = words[i];
                param.DbType = DbType.String;
                comm.Parameters.Add(param);
                index++;
            }

        // execute the stored procedure and save the results in a
DataTable
        DataTable table = GenericDataAccess.ExecuteSelectCommand(comm);
        // calculate how many pages of products and set the out
parameter
        int howManyProducts =
Int32.Parse(comm.Parameters["@HowManyResults"].Value.ToString());
        howManyPages = (int)Math.Ceiling((double)howManyProducts /

(double)BalloonShopConfiguration.ProductsPerPage);
        // return the page of products
        return table;
    }

    // Update department details
    public static bool UpdateDepartment(string id, string name, string
description)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "UpdateDepartment";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@DepartmentId";
        param.Value = id;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@DepartmentName";
        param.Value = name;
        param.DbType = DbType.String;
        param.Size = 50;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@DepartmentDescription";
        param.Value = description;
        param.DbType = DbType.String;
```

```
        param.Size = 1000;
        comm.Parameters.Add(param);
        // result will represent the number of changed rows
        int result = -1;
        try
        {
            // execute the stored procedure
            result = GenericDataAccess.ExecuteNonQuery(comm);
        }
        catch
        {
            // any errors are logged in GenericDataAccess, we ingore
them here
        }
        // result will be 1 in case of success
        return (result != -1);
    }

    // Delete department
    public static bool DeleteDepartment(string id)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "DeleteDepartment";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@DepartmentId";
        param.Value = id;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // execute the stored procedure; an error will be thrown by the
        // database in case the department has related categories, in
which case
        // it is not deleted
        int result = -1;
        try
        {
            result = GenericDataAccess.ExecuteNonQuery(comm);
        }
        catch
        {
            // any errors are logged in GenericDataAccess, we ingore
them here
        }
        // result will be 1 in case of success
        return (result != -1);
    }

    // Add a new department
    public static bool AddDepartment(string name, string description)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "AddDepartment";
```

```
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@DepartmentName";
        param.Value = name;
        param.DbType = DbType.String;
        param.Size = 50;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@DepartmentDescription";
        param.Value = description;
        param.DbType = DbType.String;
        param.Size = 1000;
        comm.Parameters.Add(param);
        // result will represent the number of changed rows
        int result = -1;
        try
        {
            // execute the stored procedure
            result = GenericDataAccess.ExecuteNonQuery(comm);
        }
        catch
        {
            // any errors are logged in GenericDataAccess, we ingore
them here
        }
        // result will be 1 in case of success
        return (result != -1);
    }

    // Create a new Category
    public static bool CreateCategory(string departmentId, string name,
string description)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "CreateCategory";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@DepartmentID";
        param.Value = departmentId;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@CategoryName";
        param.Value = name;
        param.DbType = DbType.String;
        param.Size = 50;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@CategoryDescription";
        param.Value = description;
        param.DbType = DbType.String;
```

```
        param.Size = 1000;
        comm.Parameters.Add(param);
        // result will represent the number of changed rows
        int result = -1;
        try
        {
            // execute the stored procedure
            result = GenericDataAccess.ExecuteNonQuery(comm);
        }
        catch
        {
            // any errors are logged in GenericDataAccess, we ingore
them here
        }
        // result will be 1 in case of success
        return (result != -1);
    }

    // Update category details
    public static bool UpdateCategory(string id, string name, string
description)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "UpdateCategory";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@CategoryId";
        param.Value = id;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@CategoryName";
        param.Value = name;
        param.DbType = DbType.String;
        param.Size = 50;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@CategoryDescription";
        param.Value = description;
        param.DbType = DbType.String;
        param.Size = 1000;
        comm.Parameters.Add(param);
        // result will represent the number of changed rows
        int result = -1;
        try
        {
            // execute the stored procedure
            result = GenericDataAccess.ExecuteNonQuery(comm);
        }
        catch
        {
            // any errors are logged in GenericDataAccess, we ingore
```

```
them here
        }
        // result will be 1 in case of success
        return (result != -1);
    }

    // Delete Category
    public static bool DeleteCategory(string id)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "DeleteCategory";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@CategoryId";
        param.Value = id;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // execute the stored procedure; an error will be thrown by the
        // database in case the Category has related categories, in
which case
        // it is not deleted
        int result = -1;
        try
        {
            result = GenericDataAccess.ExecuteNonQuery(comm);
        }
        catch
        {
            // any errors are logged in GenericDataAccess, we ingore
them here
        }
        // result will be 1 in case of success
        return (result != -1);
    }

    // retrieve the list of products in a category
    public static DataTable GetAllProductsInCategory(string categoryId)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "GetAllProductsInCategory";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@CategoryID";
        param.Value = categoryId;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // execute the stored procedure and save the results in a
DataTable
        DataTable table = GenericDataAccess.ExecuteSelectCommand(comm);
        return table;
    }
```

```
    // Create a new product
    public static bool CreateProduct(string categoryId, string name,
string description, string price, string image1FileName, string
image2FileName, string onDepartmentPromotion, string
onCatalogPromotion)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "CreateProduct";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@CategoryID";
        param.Value = categoryId;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@ProductName";
        param.Value = name;
        param.DbType = DbType.String;
        param.Size = 50;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@ProductDescription";
        param.Value = description;
        param.DbType = DbType.AnsiString;
        param.Size = 5000;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@ProductPrice";
        param.Value = price;
        param.DbType = DbType.Decimal;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@Image1FileName";
        param.Value = image1FileName;
        param.DbType = DbType.String;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@Image2FileName";
        param.Value = image2FileName;
        param.DbType = DbType.String;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@OnDepartmentPromotion";
        param.Value = onDepartmentPromotion;
        param.DbType = DbType.Boolean;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
```

```
        param.ParameterName = "@OnCatalogPromotion";
        param.Value = onCatalogPromotion;
        param.DbType = DbType.Boolean;
        comm.Parameters.Add(param);
        // result will represent the number of changed rows
        int result = -1;
        try
        {
            // execute the stored procedure
            result = GenericDataAccess.ExecuteNonQuery(comm);
        }
        catch
        {
            // any errors are logged in GenericDataAccess, we ingore
them here
        }
        // result will be 1 in case of success
        return (result >= 1);
    }

    // Update an existing product
    public static bool UpdateProduct(string productId, string name,
string description, string price, string image1FileName, string
image2FileName, string onDepartmentPromotion, string
onCatalogPromotion)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "UpdateProduct";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@ProductID";
        param.Value = productId;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@ProductName";
        param.Value = name;
        param.DbType = DbType.String;
        param.Size = 50;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@ProductDescription";
        param.Value = description;
        param.DbType = DbType.AnsiString;
        param.Size = 5000;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@ProductPrice";
        param.Value = price;
        param.DbType = DbType.Decimal;
        comm.Parameters.Add(param);
```

```
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@Image1FileName";
        param.Value = image1FileName;
        param.DbType = DbType.String;
        param.Size = 50;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@Image2FileName";
        param.Value = image2FileName;
        param.DbType = DbType.String;
        param.Size = 50;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@OnDepartmentPromotion";
        param.Value = onDepartmentPromotion;
        param.DbType = DbType.Boolean;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@OnCatalogPromotion";
        param.Value = onCatalogPromotion;
        param.DbType = DbType.Boolean;
        comm.Parameters.Add(param);
        // result will represent the number of changed rows
        int result = -1;
        try
        {
            // execute the stored procedure
            result = GenericDataAccess.ExecuteNonQuery(comm);
        }
        catch
        {
            // any errors are logged in GenericDataAccess, we ingore
them here
        }
        // result will be 1 in case of success
        return (result != -1);
    }

    // get categories that contain a specified product
    public static DataTable GetCategoriesWithProduct(string productId)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "GetCategoriesWithProduct";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@ProductID";
        param.Value = productId;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // execute the stored procedure
```

```
        return GenericDataAccess.ExecuteSelectCommand(comm);
    }

    // get categories that do not contain a specified product
    public static DataTable GetCategoriesWithoutProduct(string
productId)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "GetCategoriesWithoutProduct";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@ProductID";
        param.Value = productId;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // execute the stored procedure
        return GenericDataAccess.ExecuteSelectCommand(comm);
    }

    // assign a product to a new category
    public static bool AssignProductToCategory(string productId, string
categoryId)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "AssignProductToCategory";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@ProductID";
        param.Value = productId;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@CategoryID";
        param.Value = categoryId;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // result will represent the number of changed rows
        int result = -1;
        try
        {
            // execute the stored procedure
            result = GenericDataAccess.ExecuteNonQuery(comm);
        }
        catch
        {
            // any errors are logged in GenericDataAccess, we ingore
them here
        }
        // result will be 1 in case of success
        return (result != -1);
    }
```

```
    // move product to a new category
    public static bool MoveProductToCategory(string productId, string
oldCategoryId, string newCategoryId)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "MoveProductToCategory";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@ProductID";
        param.Value = productId;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@OldCategoryID";
        param.Value = oldCategoryId;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@NewCategoryID";
        param.Value = newCategoryId;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // result will represent the number of changed rows
        int result = -1;
        try
        {
            // execute the stored procedure
            result = GenericDataAccess.ExecuteNonQuery(comm);
        }
        catch
        {
            // any errors are logged in GenericDataAccess, we ingore
them here
        }
        // result will be 1 in case of success
        return (result != -1);
    }

    // removes a product from a category
    public static bool RemoveProductFromCategory(string productId,
string categoryId)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "RemoveProductFromCategory";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@ProductID";
        param.Value = productId;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
```

```
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@CategoryID";
        param.Value = categoryId;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // result will represent the number of changed rows
        int result = -1;
        try
        {
            // execute the stored procedure
            result = GenericDataAccess.ExecuteNonQuery(comm);
        }
        catch
        {
            // any errors are logged in GenericDataAccess, we ingore
them here
        }
        // result will be 1 in case of success
        return (result != -1);
    }

    // deletes a product from the product catalog
    public static bool DeleteProduct(string productId)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "DeleteProduct";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@ProductID";
        param.Value = productId;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // result will represent the number of changed rows
        int result = -1;
        try
        {
            // execute the stored procedure
            result = GenericDataAccess.ExecuteNonQuery(comm);
        }
        catch
        {
            // any errors are logged in GenericDataAccess, we ingore
them here
        }
        // result will be 1 in case of success
        return (result != -1);
    }

    // gets product recommendations
    public static DataTable GetRecommendations(string productId)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
```

```
        // set the stored procedure name
        comm.CommandText = "GetProductRecommendations";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@ProductID";
        param.Value = productId;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@DescriptionLength";
        param.Value =
BalloonShopConfiguration.ProductDescriptionLength;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // execute the stored procedure
        return GenericDataAccess.ExecuteSelectCommand(comm);
    }
}
```

## 4.3.6.3 GenericDataAccess.cs

```
using System;
using System.Data;
using System.Data.Common;
using System.Configuration;

/// <summary>
/// Class contains generic data access functionality to be accessed
from
/// the business tier
/// </summary>
public static class GenericDataAccess
{
    // static constructor
    static GenericDataAccess()
    {
        //
        // TODO: Add constructor logic here
        //
    }

    // execute a command and return the results as a DataTable object
    public static DataTable ExecuteSelectCommand(DbCommand command)
    {
        // The DataTable to be returned
        DataTable table;
        // Execute the command making sure the connection gets closed
in the end
        try
        {
            // Open the data connection
```

```
            command.Connection.Open();
            // Execute the command and save the results in a DataTable
            DbDataReader reader = command.ExecuteReader();
            table = new DataTable();
            table.Load(reader);
            // Close the reader
            reader.Close();
        }
        catch (Exception ex)
        {
            Utilities.LogError(ex);
            throw ex;
        }
        finally
        {
            // Close the connection
            command.Connection.Close();
        }
        return table;
    }

    // execute an update, delete, or insert command
    // and return the number of affected rows
    public static int ExecuteNonQuery(DbCommand command)
    {
        // The number of affected rows
        int affectedRows = -1;
        // Execute the command making sure the connection gets closed
in the end
        try
        {
            // Open the connection of the command
            command.Connection.Open();
            // Execute the command and get the number of affected rows
            affectedRows = command.ExecuteNonQuery();
        }
        catch (Exception ex)
        {
            // Log eventual errors and rethrow them
            Utilities.LogError(ex);
            throw ex;
        }
        finally
        {
            // Close the connection
            command.Connection.Close();
        }
        // return the number of affected rows
        return affectedRows;
    }

    // execute a select command and return a single result as a string
    public static string ExecuteScalar(DbCommand command)
    {
        // The value to be returned
        string value = "";
```

```
        // Execute the command making sure the connection gets closed
in the end
        try
        {
            // Open the connection of the command
            command.Connection.Open();
            // Execute the command and get the number of affected rows
            value = command.ExecuteScalar().ToString();
        }
        catch (Exception ex)
        {
            // Log eventual errors and rethrow them
            Utilities.LogError(ex);
            throw ex;
        }
        finally
        {
            // Close the connection
            command.Connection.Close();
        }
        // return the result
        return value;
    }

    // creates and prepares a new DbCommand object on a new connection
    public static DbCommand CreateCommand()
    {
        // Obtain the database provider name
        string dataProviderName =
BalloonShopConfiguration.DbProviderName;
        // Obtain the database connection string
        string connectionString =
BalloonShopConfiguration.DbConnectionString;
        // Create a new data provider factory
        DbProviderFactory factory =
DbProviderFactories.GetFactory(dataProviderName);
        // Obtain a database specific connection object
        DbConnection conn = factory.CreateConnection();
        // Set the connection string
        conn.ConnectionString = connectionString;
        // Create a database specific command object
        DbCommand comm = conn.CreateCommand();
        // Set the command type to stored procedure
        comm.CommandType = CommandType.StoredProcedure;
        // Return the initialized command object
        return comm;
    }
}
```

### 4.3.6.4 OrdersAccess.cs

```
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.Data.Common;

/// <summary>
/// Wraps order data
/// </summary>
public struct OrderInfo
{
    public int OrderID;
    public decimal TotalAmount;
    public string DateCreated;
    public string DateShipped;
    public bool Verified;
    public bool Completed;
    public bool Canceled;
    public string Comments;
    public string CustomerName;
    public string ShippingAddress;
    public string CustomerEmail;
}

/// <summary>
/// Summary description for OrdersAccess
/// </summary>
public class OrdersAccess
{
    public OrdersAccess()
    {
        //
        // TODO: Add constructor logic here
        //
    }

    // Retrieve the recent orders
    public static DataTable GetByRecent(int count)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "OrdersGetByRecent";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@Count";
        param.Value = count;
        param.DbType = DbType.Int32;
```

```
        comm.Parameters.Add(param);
        // return the result table
        DataTable table = GenericDataAccess.ExecuteSelectCommand(comm);
        return table;
    }

    // Retrieve orders that have been placed in a specified period of
time
    public static DataTable GetByDate(string startDate, string endDate)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "OrdersGetByDate";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@StartDate";
        param.Value = startDate;
        param.DbType = DbType.Date;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@EndDate";
        param.Value = endDate;
        param.DbType = DbType.Date;
        comm.Parameters.Add(param);
        // return the result table
        DataTable table = GenericDataAccess.ExecuteSelectCommand(comm);
        return table;
    }

    // Retrieve orders that need to be verified or canceled
    public static DataTable GetUnverifiedUncanceled()
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "OrdersGetUnverifiedUncanceled";
        // return the result table
        DataTable table = GenericDataAccess.ExecuteSelectCommand(comm);
        return table;
    }

    // Retrieve orders that need to be shipped/completed
    public static DataTable GetVerifiedUncompleted()
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "OrdersGetVerifiedUncompleted";
        // return the result table
        DataTable table = GenericDataAccess.ExecuteSelectCommand(comm);
        return table;
    }

    // Retrieve order information
```

```
    public static OrderInfo GetInfo(string orderID)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "OrderGetInfo";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@OrderID";
        param.Value = orderID;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // obtain the results
        DataTable table = GenericDataAccess.ExecuteSelectCommand(comm);
        DataRow orderRow = table.Rows[0];
        // save the results into an OrderInfo object
        OrderInfo orderInfo;
        orderInfo.OrderID =
Int32.Parse(orderRow["OrderID"].ToString());
        orderInfo.TotalAmount =
Decimal.Parse(orderRow["TotalAmount"].ToString());
        orderInfo.DateCreated = orderRow["DateCreated"].ToString();
        orderInfo.DateShipped = orderRow["DateShipped"].ToString();
        orderInfo.Verified =
bool.Parse(orderRow["Verified"].ToString());
        orderInfo.Completed =
bool.Parse(orderRow["Completed"].ToString());
        orderInfo.Canceled =
bool.Parse(orderRow["Canceled"].ToString());
        orderInfo.Comments = orderRow["Comments"].ToString();
        orderInfo.CustomerName = orderRow["CustomerName"].ToString();
        orderInfo.ShippingAddress =
orderRow["ShippingAddress"].ToString();
        orderInfo.CustomerEmail = orderRow["CustomerEmail"].ToString();
        // return the OrderInfo object
        return orderInfo;
    }

    // Retrieve the order details (the products that are part of that
order)
    public static DataTable GetDetails(string orderID)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "OrderGetDetails";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@OrderID";
        param.Value = orderID;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // return the results
        DataTable table = GenericDataAccess.ExecuteSelectCommand(comm);
        return table;
    }
```

```
// Update an order
public static void Update(OrderInfo orderInfo)
{
    // get a configured DbCommand object
    DbCommand comm = GenericDataAccess.CreateCommand();
    // set the stored procedure name
    comm.CommandText = "OrderUpdate";
    // create a new parameter
    DbParameter param = comm.CreateParameter();
    param.ParameterName = "@OrderID";
    param.Value = orderInfo.OrderID;
    param.DbType = DbType.Int32;
    comm.Parameters.Add(param);
    // create a new parameter
    param = comm.CreateParameter();
    param.ParameterName = "@DateCreated";
    param.Value = orderInfo.DateCreated;
    param.DbType = DbType.DateTime;
    comm.Parameters.Add(param);
    // The DateShipped parameter is sent only if data is available
    if (orderInfo.DateShipped.Trim() != "")
    {
        param = comm.CreateParameter();
        param.ParameterName = "@DateShipped";
        param.Value = orderInfo.DateShipped;
        param.DbType = DbType.DateTime;
        comm.Parameters.Add(param);
    }
    // create a new parameter
    param = comm.CreateParameter();
    param.ParameterName = "@Verified";
    param.Value = orderInfo.Verified;
    param.DbType = DbType.Byte;
    comm.Parameters.Add(param);
    // create a new parameter
    param = comm.CreateParameter();
    param.ParameterName = "@Completed";
    param.Value = orderInfo.Completed;
    param.DbType = DbType.Byte;
    comm.Parameters.Add(param);
    // create a new parameter
    param = comm.CreateParameter();
    param.ParameterName = "@Canceled";
    param.Value = orderInfo.Canceled;
    param.DbType = DbType.Byte;
    comm.Parameters.Add(param);
    // create a new parameter
    param = comm.CreateParameter();
    param.ParameterName = "@Comments";
    param.Value = orderInfo.Comments;
    param.DbType = DbType.String;
    comm.Parameters.Add(param);
    // create a new parameter
    param = comm.CreateParameter();
    param.ParameterName = "@CustomerName";
```

```
        param.Value = orderInfo.CustomerName;
        param.DbType = DbType.String;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@ShippingAddress";
        param.Value = orderInfo.ShippingAddress;
        param.DbType = DbType.String;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@CustomerEmail";
        param.Value = orderInfo.CustomerEmail;
        param.DbType = DbType.String;
        comm.Parameters.Add(param);
        // return the results
        GenericDataAccess.ExecuteNonQuery(comm);
    }

    // Mark an order as verified
    public static void MarkVerified(string orderId)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "OrderMarkVerified";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@OrderID";
        param.Value = orderId;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // return the results
        GenericDataAccess.ExecuteNonQuery(comm);
    }

    // Mark an order as completed
    public static void MarkCompleted(string orderId)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "OrderMarkCompleted";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@OrderID";
        param.Value = orderId;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // return the results
        GenericDataAccess.ExecuteNonQuery(comm);
    }

    // Mark an order as canceled
    public static void MarkCanceled(string orderId)
    {
```

```
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "OrderMarkCanceled";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@OrderID";
        param.Value = orderId;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // return the results
        GenericDataAccess.ExecuteNonQuery(comm);
    }


}
```

## 4.3.6.5 ProfileDataSource.cs

```
using System;
using System.Collections.Generic;

/// <summary>
/// A further wrapper around ProfileWrapper, exposing data
/// in a form usable by ObjectDataSource.
/// </summary>
public class ProfileDataSource
{
    public ProfileDataSource()
    {
    }

    public List<ProfileWrapper> GetData()
    {
        List<ProfileWrapper> data = new List<ProfileWrapper>();
        data.Add(new ProfileWrapper());
        return data;
    }

    public void UpdateData(ProfileWrapper newData)
    {
        newData.UpdateProfile();
    }
}
```

## 4.3.6.6 ProfileWrapper.cs

```
using System;
using System.Web;
using System.Web.Security;
```

```
using SecurityLib;

/// <summary>
/// A wrapper around profile information, incuding
/// credit card encryption functionality.
/// </summary>
public class ProfileWrapper
{
    private string address1;
    private string address2;
    private string city;
    private string region;
    private string postalCode;
    private string country;
    private string shippingRegion;
    private string dayPhone;
    private string evePhone;
    private string mobPhone;
    private string email;
    private string creditCard;
    private string creditCardHolder;
    private string creditCardNumber;
    private string creditCardIssueDate;
    private string creditCardIssueNumber;
    private string creditCardExpiryDate;
    private string creditCardType;

    public string Address1
    {
        get
        {
            return address1;
        }
        set
        {
            address1 = value;
        }
    }
    public string Address2
    {
        get
        {
            return address2;
        }
        set
        {
            address2 = value;
        }
    }
    public string City
    {
        get
        {
            return city;
        }
        set
```

```
        {
            city = value;
        }
    }
    public string Region
    {
        get
        {
            return region;
        }
        set
        {
            region = value;
        }
    }
    public string PostalCode
    {
        get
        {
            return postalCode;
        }
        set
        {
            postalCode = value;
        }
    }
    public string Country
    {
        get
        {
            return country;
        }
        set
        {
            country = value;
        }
    }
    public string ShippingRegion
    {
        get
        {
            return shippingRegion;
        }
        set
        {
            shippingRegion = value;
        }
    }
    public string DayPhone
    {
        get
        {
            return dayPhone;
        }
        set
        {
```

```
            dayPhone = value;
        }
    }
    public string EvePhone
    {
        get
        {
            return evePhone;
        }
        set
        {
            evePhone = value;
        }
    }
    public string MobPhone
    {
        get
        {
            return mobPhone;
        }
        set
        {
            mobPhone = value;
        }
    }
    public string Email
    {
        get
        {
            return email;
        }
        set
        {
            email = value;
        }
    }
    public string CreditCard
    {
        get
        {
            return creditCard;
        }
        set
        {
            creditCard = value;
        }
    }
    public string CreditCardHolder
    {
        get
        {
            return creditCardHolder;
        }
        set
        {
            creditCardHolder = value;
```

```
        }
    }
    public string CreditCardNumber
    {
        get
        {
            return creditCardNumber;
        }
        set
        {
            creditCardNumber = value;
        }
    }
    public string CreditCardIssueDate
    {
        get
        {
            return creditCardIssueDate;
        }
        set
        {
            creditCardIssueDate = value;
        }
    }
    public string CreditCardIssueNumber
    {
        get
        {
            return creditCardIssueNumber;
        }
        set
        {
            creditCardIssueNumber = value;
        }
    }
    public string CreditCardExpiryDate
    {
        get
        {
            return creditCardExpiryDate;
        }
        set
        {
            creditCardExpiryDate = value;
        }
    }
    public string CreditCardType
    {
        get
        {
            return creditCardType;
        }
        set
        {
            creditCardType = value;
        }
```

```
}

public ProfileWrapper()
{
    ProfileCommon profile =
      HttpContext.Current.Profile as ProfileCommon;
    address1 = profile.Address1;
    address2 = profile.Address2;
    city = profile.City;
    region = profile.Region;
    postalCode = profile.PostalCode;
    country = profile.Country;
    shippingRegion =
      (profile.ShippingRegion == null
      || profile.ShippingRegion == ""
      ? "1"
      : profile.ShippingRegion);
    dayPhone = profile.DayPhone;
    evePhone = profile.EvePhone;
    mobPhone = profile.MobPhone;
    email = Membership.GetUser(profile.UserName).Email;
    try
    {
        SecureCard secureCard = new SecureCard(profile.CreditCard);
        creditCard = secureCard.CardNumberX;
        creditCardHolder = secureCard.CardHolder;
        creditCardNumber = secureCard.CardNumber;
        creditCardIssueDate = secureCard.IssueDate;
        creditCardIssueNumber = secureCard.IssueNumber;
        creditCardExpiryDate = secureCard.ExpiryDate;
        creditCardType = secureCard.CardType;
    }
    catch
    {
        creditCard = "Not entered.";
    }
}

public void UpdateProfile()
{
    ProfileCommon profile =
       HttpContext.Current.Profile as ProfileCommon;
    profile.Address1 = address1;
    profile.Address2 = address2;
    profile.City = city;
    profile.Region = region;
    profile.PostalCode = postalCode;
    profile.Country = country;
    profile.ShippingRegion = shippingRegion;
    profile.DayPhone = dayPhone;
    profile.EvePhone = evePhone;
    profile.MobPhone = mobPhone;
    profile.CreditCard = creditCard;
    Membership.GetUser(profile.UserName).Email = email;
    try
    {
```

```
            SecureCard secureCard = new SecureCard(
               creditCardHolder, creditCardNumber,
               creditCardIssueDate, creditCardExpiryDate,
               creditCardIssueNumber, creditCardType);
            profile.CreditCard = secureCard.EncryptedData;
        }
        catch
        {
            creditCard = "";
        }
    }
}
```

## 4.3.6.7 ShoppingCartAccess.cs

```
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.Data.Common;

/// <summary>
/// Supports Shopping Cart functionality
/// </summary>
public class ShoppingCartAccess
{
    public ShoppingCartAccess()
    {
        //
        // TODO: Add constructor logic here
        //
    }

    // returns the shopping cart ID for the current user
    private static string shoppingCartId
    {
        get
        {
            // get the current HttpContext
            HttpContext context = HttpContext.Current;
            // try to retrieve the cart ID from the user session object
            string cartId = "";
            object cartIdSession =
context.Session["BalloonShop_CartID"];
            if (cartIdSession != null)
                cartId = cartIdSession.ToString();
            // if the ID exists in the current session...
```

```
            if (cartId != "")
                // return its value
                return cartId;
            else
            // if the cart ID isn't in the session...
            {
                // check if the cart ID exists as a cookie
                if (context.Request.Cookies["BalloonShop_CartID"] !=
null)
                {
                    // if the cart exists as a cookie, use the cookie
to get its value
                    cartId =
context.Request.Cookies["BalloonShop_CartID"].Value;
                    // save the id to the session, to avoid reading the
cookie next time
                    context.Session["BalloonShop_CartID"] = cartId;
                    // return the id
                    return cartId;
                }
                else
                // if the cart ID doesn't exist in the cookie as well,
generate a new ID
                {
                    // generate a new GUID
                    cartId = Guid.NewGuid().ToString();
                    // create the cookie object and set its value
                    HttpCookie cookie = new
HttpCookie("BalloonShop_CartID", cartId.ToString());
                    // set the cookie's expiration date
                    int howManyDays =
BalloonShopConfiguration.CartPersistDays;
                    DateTime currentDate = DateTime.Now;
                    TimeSpan timeSpan = new TimeSpan(howManyDays, 0, 0,
0);
                    DateTime expirationDate =
currentDate.Add(timeSpan);
                    cookie.Expires = expirationDate;
                    // set the cookie on client's browser
                    context.Response.Cookies.Add(cookie);
                    // save the ID to the Session as well
                    context.Session["BalloonShop_CartID"] = cartId;
                    // return the CartID
                    return cartId.ToString();
                }
            }
        }
    }

    // Add a new shopping cart item
    public static bool AddItem(string productId)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "ShoppingCartAddItem";
```

```
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@CartID";
        param.Value = shoppingCartId;
        param.DbType = DbType.String;
        param.Size = 36;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@ProductID";
        param.Value = productId;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // returns true in case of success or false in case of an error
        try
        {
            // execute the stored procedure and return true if it
executes
            // successfully, or false otherwise
            return (GenericDataAccess.ExecuteNonQuery(comm) != -1);
        }
        catch
        {
            // prevent the exception from propagating, but return false
to
            // signal the error
            return false;
        }
    }

    // Update the quantity of a shopping cart item
    public static bool UpdateItem(string productId, int quantity)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "ShoppingCartUpdateItem";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@CartID";
        param.Value = shoppingCartId;
        param.DbType = DbType.String;
        param.Size = 36;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@ProductID";
        param.Value = productId;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@Quantity";
        param.Value = quantity;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
```

```
        // returns true in case of success or false in case of an error
        try
        {
            // execute the stored procedure and return true if it
executes
            // successfully, or false otherwise
            return (GenericDataAccess.ExecuteNonQuery(comm) != -1);
        }
        catch
        {
            // prevent the exception from propagating, but return false
to
            // signal the error
            return false;
        }
    }

    // Remove a shopping cart item
    public static bool RemoveItem(string productId)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "ShoppingCartRemoveItem";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@CartID";
        param.Value = shoppingCartId;
        param.DbType = DbType.String;
        param.Size = 36;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@ProductID";
        param.Value = productId;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // returns true in case of success or false in case of an error
        try
        {
            // execute the stored procedure and return true if it
executes
            // successfully, or false otherwise
            return (GenericDataAccess.ExecuteNonQuery(comm) != -1);
        }
        catch
        {
            // prevent the exception from propagating, but return false
to
            // signal the error
            return false;
        }
    }

    // Retrieve shopping cart items
    public static DataTable GetItems()
```

```
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "ShoppingCartGetItems";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@CartID";
        param.Value = shoppingCartId;
        param.DbType = DbType.String;
        param.Size = 36;
        comm.Parameters.Add(param);
        // return the result table
        DataTable table = GenericDataAccess.ExecuteSelectCommand(comm);
        return table;
    }

    // Retrieve shopping cart items
    public static decimal GetTotalAmount()
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "ShoppingCartGetTotalAmount";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@CartID";
        param.Value = shoppingCartId;
        param.DbType = DbType.String;
        param.Size = 36;
        comm.Parameters.Add(param);
        // return the result table
        return Decimal.Parse(GenericDataAccess.ExecuteScalar(comm));
    }

    // Counts old shopping carts
    public static int CountOldCarts(byte days)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "ShoppingCartCountOldCarts";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@Days";
        param.Value = days;
        param.DbType = DbType.Byte;
        comm.Parameters.Add(param);
        // execute the procedure and return number of old shopping
carts
        try
        {
            return Byte.Parse(GenericDataAccess.ExecuteScalar(comm));
        }
        catch
        {
```

```
            return -1;
        }
    }

    // Deletes old shopping carts
    public static bool DeleteOldCarts(byte days)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "ShoppingCartDeleteOldCarts";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@Days";
        param.Value = days;
        param.DbType = DbType.Byte;
        comm.Parameters.Add(param);
        // execute the procedure and return true if no problem occurs
        try
        {
            GenericDataAccess.ExecuteNonQuery(comm);
            return true;
        }
        catch
        {
            return false;
        }
    }

    // Create a new order from the shopping cart
    public static string CreateOrder()
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "CreateOrder";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@CartID";
        param.Value = shoppingCartId;
        param.DbType = DbType.String;
        param.Size = 36;
        comm.Parameters.Add(param);
        // return the result table
        return GenericDataAccess.ExecuteScalar(comm);
    }

    // gets product recommendations for the shopping cart
    public static DataTable GetRecommendations()
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "GetShoppingCartRecommendations";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
```

```
        param.ParameterName = "@CartID";
        param.Value = shoppingCartId;
        param.DbType = DbType.String;
        param.Size = 36;
        comm.Parameters.Add(param);
        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@DescriptionLength";
        param.Value =
BalloonShopConfiguration.ProductDescriptionLength;
        param.DbType = DbType.Int32;
        comm.Parameters.Add(param);
        // execute the stored procedure
        return GenericDataAccess.ExecuteSelectCommand(comm);
    }
}
```

## 4.3.6.8 Utilities.cs

```
using System;
using System.Net.Mail;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;


/// <summary>
/// Class contains miscellaneous functionality
/// </summary>
public static class Utilities
{
    static Utilities()
    {
        //
        // TODO: Add constructor logic here
        //
    }

    // Generic method for sending emails
    public static void SendMail(string from, string to, string subject,
string body)
    {
        // Configure mail client (may need additional
        // code for authenticated SMTP servers)
        SmtpClient mailClient = new
SmtpClient(BalloonShopConfiguration.MailServer);
        // Create the mail message
        MailMessage mailMessage = new MailMessage(from, to, subject,
body);
        /*
           // For SMTP servers that require authentication
```

```
message.Fields.Add("http://schemas.microsoft.com/cdo/configuration/smtp
authenticate", 1);

message.Fields.Add("http://schemas.microsoft.com/cdo/configuration/send
username", "SmtpHostUserName");

message.Fields.Add("http://schemas.microsoft.com/cdo/configuration/send
password", "SmtpHostPassword");
          */
        // Send mail
        mailClient.Send(mailMessage);
    }

    // Send error log mail
    public static void LogError(Exception ex)
    {
        // get the current date and time
        string dateTime = DateTime.Now.ToLongDateString() + ", at "
                        + DateTime.Now.ToShortTimeString();
        // stores the error message
        string errorMessage = "Exception generated on " + dateTime;
        // obtain the page that generated the error
        System.Web.HttpContext context =
System.Web.HttpContext.Current;
        errorMessage += "\n\n Page location: " +
context.Request.RawUrl;
        // build the error message
        errorMessage += "\n\n Message: " + ex.Message;
        errorMessage += "\n\n Source: " + ex.Source;
        errorMessage += "\n\n Method: " + ex.TargetSite;
        errorMessage += "\n\n Stack Trace: \n\n" + ex.StackTrace;
        // send error email in case the option is activated in
Web.Config
        if (BalloonShopConfiguration.EnableErrorLogEmail)
        {
            string from = "noreply@cristiandarie.ro";
            string to = BalloonShopConfiguration.ErrorLogEmail;
            string subject = BalloonShopConfiguration.SiteName + "
error report";
            string body = errorMessage;
            SendMail(from, to, subject, body);
        }
    }

    // Configures what button to be clicked when the uses presses Enter
in a
    // textbox. The text box doesn't have to be a TextBox control, but
it must
    // be derived from either HtmlControl or WebControl, and the HTML
control it
    // generates should accept an 'onkeydown' attribute. The HTML
generated by
    // the button must support the 'Click' event
    public static void TieButton(Page page, Control TextBoxToTie,
Control ButtonToTie)
    {
```

```
        // Init jscript
        string jsString = "";

        // Check button type and get required jscript
        if (ButtonToTie is LinkButton)
        {
            jsString = "if ((event.which && event.which == 13) ||
(event.keyCode && event.keyCode == 13)) {"
                +
page.ClientScript.GetPostBackEventReference(ButtonToTie,
"").Replace(":", "$") + ";return false;} else return true;";
        }
        else if (ButtonToTie is ImageButton)
        {
            jsString = "if ((event.which && event.which == 13) ||
(event.keyCode && event.keyCode == 13)) {"
                +
page.ClientScript.GetPostBackEventReference(ButtonToTie,
"").Replace(":", "$") + ";return false;} else return true;";
        }
        else
        {
            jsString = "if ((event.which && event.which == 13) ||
(event.keyCode && event.keyCode == 13)) {document."
                + "forms[0].elements['" +
ButtonToTie.UniqueID.Replace(":", "_") + "'].click();return false;}
else return true; ";
        }

        // Attach jscript to the onkeydown attribute - we have to cater
for HtmlControl or WebControl
        if (TextBoxToTie is HtmlControl)
        {
            ((HtmlControl)TextBoxToTie).Attributes.Add("onkeydown",
jsString);
        }
        else if (TextBoxToTie is WebControl)
        {
            ((WebControl)TextBoxToTie).Attributes.Add("onkeydown",
jsString);
        }
    }
}
```

## 4.3.6.9 PasswordHasher.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Security.Cryptography;

namespace SecurityLib
```

```
{
    public static class PasswordHasher
    {
        private static SHA1Managed hasher = new SHA1Managed();

        public static string Hash(string password)
        {
            // convert password to byte array
            byte[] passwordBytes =
              System.Text.ASCIIEncoding.ASCII.GetBytes(password);

            // generate hash from byte array of password
            byte[] passwordHash = hasher.ComputeHash(passwordBytes);

            // convert hash to string
            return Convert.ToBase64String(passwordHash, 0,
               passwordHash.Length);
        }
    }
}
```

## 4.3.6.10 SecureCard.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Xml;

namespace SecurityLib
{
    public class SecureCard
    {
        private bool isDecrypted = false;
        private bool isEncrypted = false;
        private string cardHolder;
        private string cardNumber;
        private string issueDate;
        private string expiryDate;
        private string issueNumber;
        private string cardType;
        private string encryptedData;
        private XmlDocument xmlCardData;

        private SecureCard()
        {
            // private default constructor
        }

        public SecureCard(string newEncryptedData)
        {
            // constructor for use with encrypted data
            encryptedData = newEncryptedData;
```

```
        DecryptData();
    }

    public SecureCard(string newCardHolder,
      string newCardNumber, string newIssueDate,
      string newExpiryDate, string newIssueNumber,
      string newCardType)
    {
        // constructor for use with decrypted data
        cardHolder = newCardHolder;
        cardNumber = newCardNumber;
        issueDate = newIssueDate;
        expiryDate = newExpiryDate;
        issueNumber = newIssueNumber;
        cardType = newCardType;
        EncryptData();
    }

    private void CreateXml()
    {
        // encode card details as XML document
        xmlCardData = new XmlDocument();
        XmlElement documentRoot =
          xmlCardData.CreateElement("CardDetails");
        XmlElement child;

        child = xmlCardData.CreateElement("CardHolder");
        child.InnerXml = cardHolder;
        documentRoot.AppendChild(child);

        child = xmlCardData.CreateElement("CardNumber");
        child.InnerXml = cardNumber;
        documentRoot.AppendChild(child);

        child = xmlCardData.CreateElement("IssueDate");
        child.InnerXml = issueDate;
        documentRoot.AppendChild(child);

        child = xmlCardData.CreateElement("ExpiryDate");
        child.InnerXml = expiryDate;
        documentRoot.AppendChild(child);

        child = xmlCardData.CreateElement("IssueNumber");
        child.InnerXml = issueNumber;
        documentRoot.AppendChild(child);

        child = xmlCardData.CreateElement("CardType");
        child.InnerXml = cardType;
        documentRoot.AppendChild(child);
        xmlCardData.AppendChild(documentRoot);
    }

    private void ExtractXml()
    {
        // get card details out of XML document
        cardHolder =
```

```
            xmlCardData.GetElementsByTagName(
              "CardHolder").Item(0).InnerXml;
        cardNumber =
            xmlCardData.GetElementsByTagName(
              "CardNumber").Item(0).InnerXml;
        issueDate =
            xmlCardData.GetElementsByTagName(
              "IssueDate").Item(0).InnerXml;
        expiryDate =
            xmlCardData.GetElementsByTagName(
              "ExpiryDate").Item(0).InnerXml;
        issueNumber =
            xmlCardData.GetElementsByTagName(
              "IssueNumber").Item(0).InnerXml;
        cardType =
            xmlCardData.GetElementsByTagName(
              "CardType").Item(0).InnerXml;
    }

    private void EncryptData()
    {
        try
        {
            // put data into XML doc
            CreateXml();

            // encrypt data
            encryptedData =
              StringEncryptor.Encrypt(xmlCardData.OuterXml);

            // set encrypted flag
            isEncrypted = true;
        }
        catch
        {
            throw new SecureCardException("Unable to encrypt
data.");
        }
    }

    private void DecryptData()
    {
        try
        {
            // decrypt data
            xmlCardData = new XmlDocument();
            xmlCardData.InnerXml =
              StringEncryptor.Decrypt(encryptedData);

            // extract data from XML
            ExtractXml();

            // set decrypted flag
            isDecrypted = true;
        }
        catch
```

```
            {
                throw new SecureCardException("Unable to decrypt
data.");
            }
        }

        public string CardHolder
        {
            get
            {
                if (isDecrypted)
                {
                    return cardHolder;
                }
                else
                {
                    throw new SecureCardException("Data not
decrypted.");
                }
            }
        }

        public string CardNumber
        {
            get
            {
                if (isDecrypted)
                {
                    return cardNumber;
                }
                else
                {
                    throw new SecureCardException("Data not
decrypted.");
                }
            }
        }

        public string CardNumberX
        {
            get
            {
                if (isDecrypted)
                {
                    return "XXXX-XXXX-XXXX-"
                       + cardNumber.Substring(cardNumber.Length - 4,
4);
                }
                else
                {
                    throw new SecureCardException("Data not
decrypted.");
                }
            }
        }
```

```
        public string IssueDate
        {
            get
            {
                if (isDecrypted)
                {
                    return issueDate;
                }
                else
                {
                    throw new SecureCardException("Data not
decrypted.");
                }
            }
        }

        public string ExpiryDate
        {
            get
            {
                if (isDecrypted)
                {
                    return expiryDate;
                }
                else
                {
                    throw new SecureCardException("Data not
decrypted.");
                }
            }
        }

        public string IssueNumber
        {
            get
            {
                if (isDecrypted)
                {
                    return issueNumber;
                }
                else
                {
                    throw new SecureCardException("Data not
decrypted.");
                }
            }
        }

        public string CardType
        {
            get
            {
                if (isDecrypted)
                {
                    return cardType;
                }
```

```
                else
                {
                    throw new SecureCardException("Data not
decrypted.");
                }
            }
        }

        public string EncryptedData
        {
            get
            {
                if (isEncrypted)
                {
                    return encryptedData;
                }
                else
                {
                    throw new SecureCardException("Data not
decrypted.");
                }
            }
        }
    }
}
```

## 4.3.6.11 SecureCardException.cs

```
using System;

namespace SecurityLib
{
    public class SecureCardException : Exception
    {
        public SecureCardException(string message)
            : base(message)
        {
        }
    }
}
```

## 4.3.6.12 StringEncryptor.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Security.Cryptography;
using System.IO;
```

```
namespace SecurityLib
{
    public static class StringEncryptor
    {
        public static string Encrypt(string sourceData)
        {
            // set key and initialization vector values
            byte[] key = new byte[] { 1, 2, 3, 4, 5, 6, 7, 8 };
            byte[] iv = new byte[] { 1, 2, 3, 4, 5, 6, 7, 8 };
            try
            {
                // convert data to byte array
                byte[] sourceDataBytes =

System.Text.ASCIIEncoding.ASCII.GetBytes(sourceData);

                // get target memory stream
                MemoryStream tempStream = new MemoryStream();

                // get encryptor and encryption stream
                DESCryptoServiceProvider encryptor =
                   new DESCryptoServiceProvider();
                CryptoStream encryptionStream =
                   new CryptoStream(tempStream,
                      encryptor.CreateEncryptor(key, iv),
                      CryptoStreamMode.Write);

                // encrypt data
                encryptionStream.Write(sourceDataBytes, 0,
                   sourceDataBytes.Length);
                encryptionStream.FlushFinalBlock();

                // put data into byte array
                byte[] encryptedDataBytes = tempStream.GetBuffer();

                // convert encrypted data into string
                return Convert.ToBase64String(encryptedDataBytes, 0,
                   (int)tempStream.Length);
            }
            catch
            {
                throw new StringEncryptorException(
                   "Unable to encrypt data.");
            }
        }

        public static string Decrypt(string sourceData)
        {
            // set key and initialization vector values
            byte[] key = new byte[] { 1, 2, 3, 4, 5, 6, 7, 8 };
            byte[] iv = new byte[] { 1, 2, 3, 4, 5, 6, 7, 8 };
            try
            {
                // convert data to byte array
                byte[] encryptedDataBytes =
```

```
                Convert.FromBase64String(sourceData);

            // get source memory stream and fill it
            MemoryStream tempStream =
                new MemoryStream(encryptedDataBytes, 0,
                    encryptedDataBytes.Length);

            // get decryptor and decryption stream
            DESCryptoServiceProvider decryptor =
                new DESCryptoServiceProvider();
            CryptoStream decryptionStream =
                new CryptoStream(tempStream,
                    decryptor.CreateDecryptor(key, iv),
                    CryptoStreamMode.Read);

            // decrypt data
            StreamReader allDataReader =
                new StreamReader(decryptionStream);
            return allDataReader.ReadToEnd();
        }
        catch
        {
            throw new StringEncryptorException(
                "Unable to decrypt data.");
        }
    }
  }
}
```

## 4.3.6.13 StringEncryptorException.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace SecurityLib
{
    public class StringEncryptorException : Exception
    {
        public StringEncryptorException(string message)
            : base(message)
        {
        }
    }
}
```

## 4.3.7 Clobal.asax

```
<%@ Application Language="C#" %>

<script runat="server">

    void Application_Start(object sender, EventArgs e)
    {
        // Code that runs on application startup

    }

    void Application_End(object sender, EventArgs e)
    {
        //  Code that runs on application shutdown

    }

    void Application_Error(object sender, EventArgs e)
    {
        // Log ass unhandled errors
        Utilities.LogError(Server.GetLastError());

    }

    void Session_Start(object sender, EventArgs e)
    {
        // Code that runs when a new session is started

    }

    void Session_End(object sender, EventArgs e)
    {
        // Code that runs when a session ends.
        // Note: The Session_End event is raised only when the
sessionstate mode
        // is set to InProc in the Web.config file. If session mode is
set to StateServer
        // or SQLServer, the event is not raised.

    }

</script>
```

## 4.3.8 Web.config

Το σημαντικότερο αρχείο της εφαρμογής μιας και δημιοργείται μόνο του μετα την πρωτη φορα που θα ''τρέξει'' η εφαρμογή. Καθορίζει διάφορους παραμέτρους, ένας απο τους σημαντικότερους είναι η ένωση με την βάση δεδομένων.

```
<?xml version="1.0"?>
<!--
    Note: As an alternative to hand editing this file you can use the
    web admin tool to configure settings for your application. Use
    the Website->Asp.Net Configuration option in Visual Studio.
    A full list of settings and comments can be found in
    machine.config.comments usually located in
    \Windows\Microsoft.Net\Framework\v2.x\Config
-->
<configuration
xmlns="http://schemas.microsoft.com/.NetConfiguration/v2.0">
  <appSettings>
    <add key="CartPersistDays" value="10" />
    <add key="MailServer" value="localhost" />
    <add key="EnableErrorLogEmail" value="false" />
    <add key="ErrorLogEmail" value="cristian_darie@yahoo.com" />
    <add key="ProductsPerPage" value="6"/>
    <add key="ProductDescriptionLength" value="60"/>
    <add key="SiteName" value="BalloonShop"/>
  </appSettings>

  <connectionStrings>
    <add name="BalloonShopConnection"
connectionString="Server=(local)\SqlExpress;Integrated
Security=True;Database=BalloonShop"
providerName="System.Data.SqlClient"/>
  </connectionStrings>

  <!-- Only administrators are allowed to access CatalogAdmin.aspx -->
  <location path="CatalogAdmin.aspx">
    <system.web>
      <authorization>
        <allow roles="Administrators" />
        <deny users="*"  />
      </authorization>
    </system.web>
  </location>

  <!-- Only administrators are allowed to access ShoppingCartAdmin.aspx
-->
  <location path="ShoppingCartAdmin.aspx">
    <system.web>
      <authorization>
        <allow roles="Administrators" />
        <deny users="*" />
      </authorization>
    </system.web>
  </location>

  <!-- Only administrators are allowed to access OrdersAdmin.aspx -->
  <location path="OrdersAdmin.aspx">
    <system.web>
      <authorization>
        <allow roles="Administrators" />
        <deny users="*" />
```

```
    </authorization>
  </system.web>
</location>

<!-- Only existing customers can access CustomerDetails.aspx -->
<location path="CustomerDetails.aspx">
  <system.web>
    <authorization>
      <allow roles="Customers" />
      <deny users="*" />
    </authorization>
  </system.web>
</location>

<!-- Only existing customers can access Checkout.aspx -->
<location path="Checkout.aspx">
  <system.web>
    <authorization>
      <allow roles="Customers" />
      <deny users="*" />
    </authorization>
  </system.web>
</location>


<system.web>
  <profile>
    <properties>
      <add name="CreditCard" type="System.String" />
      <add name="Address1" type="System.String" />
      <add name="Address2" type="System.String" />
      <add name="City" type="System.String" />
      <add name="Region" type="System.String" />
      <add name="PostalCode" type="System.String" />
      <add name="Country" type="System.String" />
      <add name="ShippingRegion" type="System.String" />
      <add name="DayPhone" type="System.String" />
      <add name="EvePhone" type="System.String" />
      <add name="MobPhone" type="System.String" />
    </properties>
  </profile>

  <membership>
    <providers>
      <remove name="AspNetSqlMembershipProvider" />
      <add name="AspNetSqlMembershipProvider"
        type="System.Web.Security.SqlMembershipProvider,
        System.Web,
        Version=2.0.0.0, Culture=neutral,
        PublicKeyToken=b03f5f7f11d50a3a"
        connectionStringName="LocalSqlServer"
        enablePasswordRetrieval="false" enablePasswordReset="true"
        requiresQuestionAndAnswer="true" applicationName="/"
        requiresUniqueEmail="false" passwordFormat="Hashed"
        maxInvalidPasswordAttempts="5" passwordAttemptWindow="10"
        minRequiredPasswordLength="6"
```

```
        minRequiredNonalphanumericCharacters="0"
        passwordStrengthRegularExpression="" />
    </providers>
  </membership>

  <roleManager enabled="true" />
  <pages theme="BalloonShopDefault"/>
  <!--
        Set compilation debug="true" to insert debugging
        symbols into the compiled page. Because this
        affects performance, set this value to true only
        during development.
    -->
  <compilation debug="true"/>
  <!--
        The <authentication> section enables configuration
        of the security authentication mode used by
        ASP.NET to identify an incoming user.
    -->
  <authentication mode="Forms">
    <forms name="BalloonShopLogin"
      loginUrl="Login.aspx" path="/" protection="All" timeout="60">
    </forms>
  </authentication>

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

    <customErrors mode="RemoteOnly"
defaultRedirect="GenericErrorPage.htm">
        <error statusCode="403" redirect="NoAccess.htm"/>
        <error statusCode="404" redirect="FileNotFound.htm"/>
    </customErrors>
    -->
  <globalization requestEncoding="utf-8" responseEncoding="utf-8"
                 culture="en-US"/>
 </system.web>
</configuration>
```

### 4.3.9 Οι Φάκελοι Images και Product Images

Οι Φάκελοι Images και Product Images δεν περιέχουν κώδικα, απλά περιέχουν τις φωτογραφίες της εφαρμογής και των προιόντων της εφαρμογής αντιστοιχά lol.

### 4.3.10 ASPNETDB.MDF

Η Β.Δ , η οποία διαχειρίζεται τους χρήστες της εφαρμογής. Κατασκευάζεται αυτόματα μέσω απο το περιβάλλον του Visual Web Developer. Άλλη μία νεοτροπία της τεχνολογίας .NET.

### 4.3.11 BalloonShop.dbo

Η Β.Δ της εφαρμογής μας. Παρακάτω παρατίθενται τα Scripts για την δημιοργία των:

- Πινάκων (Tables)
- Διαδικασιών (Stored Procedures)
- Συναρτήσεών (Functions)

### 4.3.11.1 Πίνακες (Tables)

```
CREATE TABLE Category(
      CategoryID INT IDENTITY(1,1) NOT NULL,
      DepartmentID INT NOT NULL,
      Name VARCHAR(50) NOT NULL,
      Description VARCHAR(1000) NULL,
 CONSTRAINT PK_Category_1 PRIMARY KEY CLUSTERED (CategoryID ASC)

GO

CREATE TABLE Department(
      DepartmentID INT IDENTITY(1,1) NOT NULL,
      Name VARCHAR(50) NOT NULL,
      Description VARCHAR(1000) NULL,
 CONSTRAINT PK_Department PRIMARY KEY CLUSTERED (DepartmentID ASC)
)

GO

CREATE TABLE OrderDetail(
      OrderID INT NOT NULL,
      ProductID INT NOT NULL,
      ProductName VARCHAR(50) NOT NULL,
      Quantity INT NOT NULL,
      UnitCost MONEY NOT NULL,
      Subtotal  AS (Quantity*UnitCost),
 CONSTRAINT PK_OrderDetail PRIMARY KEY CLUSTERED(OrderID ASC, ProductID
ASC)
```

```
GO

CREATE TABLE Orders(
      OrderID INT IDENTITY(1,1) NOT NULL,
      DateCreated SMALLDATETIME NOT NULL CONSTRAINT
DF_Orders_DateCreated  DEFAULT (getdate()),
      DateShipped SMALLDATETIME NULL,
      Verified BIT NOT NULL CONSTRAINT DF_Orders_Verified DEFAULT
((0)),
      Completed BIT NOT NULL CONSTRAINT DF_Orders_Completed DEFAULT
((0)),
      Canceled BIT NOT NULL CONSTRAINT DF_Orders_Canceled DEFAULT
((0)),
      Comments VARCHAR(1000) NULL,
      CustomerName VARCHAR(50) NULL,
      CustomerEmail VARCHAR(50) NULL,
      ShippingAddress VARCHAR(500) NULL,
 CONSTRAINT PK_Orders PRIMARY KEY CLUSTERED(OrderID ASC)
)


GO

CREATE TABLE Product(
      ProductID INT IDENTITY(1,1) NOT NULL,
      Name VARCHAR(50) NOT NULL,
      Description VARCHAR(5000) NOT NULL,
      Price MONEY NOT NULL,
      Image1FileName VARCHAR(50) NULL,
      Image2FileName VARCHAR(50) NULL,
      OnCatalogPromotion BIT NOT NULL,
      OnDepartmentPromotion BIT NOT NULL,
 CONSTRAINT PK_Product PRIMARY KEY CLUSTERED (ProductID ASC)
)


GO

CREATE TABLE ProductCategory(
      ProductID INT NOT NULL,
      CategoryID INT NOT NULL,
 CONSTRAINT PK_ProductCategory PRIMARY KEY CLUSTERED (ProductID ASC,
CategoryID ASC)
)


GO

CREATE TABLE ShoppingCart(
      CartID char(36) NOT NULL,
      ProductID INT NOT NULL,
```

```
      Quantity INT NOT NULL,
      DateAdded SMALLDATETIME NOT NULL,
 CONSTRAINT PK_ShoppingCart PRIMARY KEY CLUSTERED (CartID ASC,
ProductID ASC)
```

## 4.3.11.2 Διαδικασίες (Stored Procedures)

```
CREATE PROCEDURE GetDepartments AS
SELECT DepartmentID, Name, Description
FROM Department

GO

CREATE PROCEDURE GetCategoriesInDepartment
(@DepartmentID INT)
AS
SELECT CategoryID, Name, Description
FROM Category
WHERE DepartmentID = @DepartmentID

GO

CREATE PROCEDURE GetDepartmentDetails
 @DepartmentID INT)
AS
SELECT Name, Description
FROM Department
WHERE DepartmentID = @DepartmentID

GO

CREATE PROCEDURE GetCategoryDetails
(@CategoryID INT)
AS
SELECT DepartmentID, Name, Description
FROM Category
WHERE CategoryID = @CategoryID

GO

CREATE PROCEDURE GetProductsOnDepartmentPromotion
(@DepartmentID INT,
@DescriptionLength INT,
@PageNumber INT,
@ProductsPerPage INT,
@HowManyProducts INT OUTPUT)
AS

-- declare a new TABLE variable
DECLARE @Products TABLE
(RowNumber INT,
 ProductID INT,
```

```
 Name VARCHAR(50 ,
 Description VARCHAR(5000),
 Price MONEY,
 Image1FileName VARCHAR(50),
 Image2FileName VARCHAR(50),
 OnDepartmentPromotion BIT,
 OnCatalogPromotion BIT)

-- populate the table variable with the complete list of products
INSERT INTO @Products
SELECT ROW_NUMBER() OVER (ORDER BY ProductID) AS Row,
       ProductID, Name, SUBSTRING(Description, 1, @DescriptionLength) +
'...' AS Description,
       Price, Image1FileName, Image2FileName, OnDepartmentPromotion,
OnCatalogPromotion
FROM
  (SELECT DISTINCT Product.ProductID, Product.Name,
   SUBSTRING(Product.Description, 1, @DescriptionLength) + '...' AS
Description,
   Price, Image1FileName, Image2FileName, OnDepartmentPromotion,
OnCatalogPromotion
   FROM Product INNER JOIN ProductCategory
                        ON Product.ProductID = ProductCategory.ProductID
              INNER JOIN Category
                        ON ProductCategory.CategoryID =
Category.CategoryID
   WHERE Product.OnDepartmentPromotion = 1
    AND Category.DepartmentID = @DepartmentID
  ) AS ProductOnDepPr

-- return the total number of products using an OUTPUT variable
SELECT @HowManyProducts = COUNT(ProductID) FROM @Products

-- extract the requested page of products
SELECT ProductID, Name, Description, Price, Image1FileName,
       Image2FileName, OnDepartmentPromotion, OnCatalogPromotion
FROM @Products
WHERE RowNumber > (@PageNumber - 1) * @ProductsPerPage
  AND RowNumber <= @PageNumber * @ProductsPerPage

GO

CREATE PROCEDURE GetProductsOnCatalogPromotion
(@DescriptionLength INT,
@PageNumber INT,
@ProductsPerPage INT,
@HowManyProducts INT OUTPUT)
AS

-- declare a new TABLE variable
DECLARE @Products TABLE
(RowNumber INT,
 ProductID INT,
 Name VARCHAR(50),
 Description VARCHAR(5000),
 Price MONEY,
```

```
 Image1FileName VARCHAR(50),
 Image2FileName VARCHAR(50),
 OnDepartmentPromotion BIT,
 OnCatalogPromotion BIT

-- populate the table variable with the complete list of products
INSERT INTO @Products
SELECT ROW_NUMBER() OVER (ORDER BY Product.ProductID),
       ProductID, Name,
       SUBSTRING(Description, 1, @DescriptionLength) + '...' AS
Description,
       Price, Image1FileName, Image2FileName, OnDepartmentPromotion,
OnCatalogPromotion
FROM Product
WHERE OnCatalogPromotion = 1

-- return the total number of products using an OUTPUT variable
SELECT @HowManyProducts = COUNT(ProductID) FROM @Products

-- extract the requested page of products
SELECT ProductID, Name, Description, Price, Image1FileName,
       Image2FileName, OnDepartmentPromotion, OnCatalogPromotion
FROM @Products
WHERE RowNumber > (@PageNumber - 1) * @ProductsPerPage
  AND RowNumber <= @PageNumber * @ProductsPerPage

GO

CREATE PROCEDURE GetProductsInCategory
 @CategoryID INT,
@DescriptionLength INT,
@PageNumber INT,
@ProductsPerPage INT,
@HowManyProducts INT OUTPUT
AS

-- declare a new TABLE variable
DECLARE @Products TABLE
(RowNumber INT,
 ProductID INT,
 Name VARCHAR(50),
 Description VARCHAR(5000),
 Price MONEY,
 Image1FileName VARCHAR(50),
 Image2FileName VARCHAR(50),
 OnDepartmentPromotion BIT,
 OnCatalogPromotion BIT)

-- populate the table variable with the complete list of products
INSERT INTO @Products
SELECT ROW_NUMBER() OVER (ORDER BY Product.ProductID),
       Product.ProductID, Name,
       SUBSTRING(Description, 1, @DescriptionLength) + '...' AS
Description,
       Price, Image1FileName, Image2FileName, OnDepartmentPromotion,
OnCatalogPromotion
```

```
FROM Product INNER JOIN ProductCategory
  ON Product.ProductID = ProductCategory.ProductID
WHERE ProductCategory.CategoryID = @CategoryID

-- return the total number of products using an OUTPUT variable
SELECT @HowManyProducts = COUNT(ProductID) FROM @Products

-- extract the requested page of products
SELECT ProductID, Name, Description, Price, Image1FileName,
       Image2FileName, OnDepartmentPromotion, OnCatalogPromotion
FROM @Products
WHERE RowNumber > (@PageNumber - 1) * @ProductsPerPage
  AND RowNumber <= @PageNumber * @ProductsPerPage

GO

CREATE PROCEDURE GetProductDetails
(@ProductID INT)
AS
SELECT Name, Description, Price, Image1FileName, Image2FileName,
OnDepartmentPromotion, OnCatalogPromotion
FROM Product
WHERE ProductID = @ProductID
RETURN

GO

CREATE PROCEDURE SearchCatalog
(@DescriptionLength INT,
 @PageNumber TINYINT,
 @ProductsPerPage TINYINT,
 @HowManyResults SMALLINT OUTPUT,
 @AllWords BIT,
 @Word1 VARCHAR(15) = NULL,
 @Word2 VARCHAR(15) = NULL,
 @Word3 VARCHAR(15) = NULL,
 @Word4 VARCHAR(15) = NULL,
 @Word5 VARCHAR(15) = NULL)
AS

/* Create the table variable that will contain the search results */
DECLARE @Products TABLE
(RowNumber SMALLINT IDENTITY (1,1) NOT NULL,
 ProductID INT,
 Name VARCHAR(50),
 Description VARCHAR(1000),
 Price MONEY,
 Image1FileName VARCHAR(50),
 Image2FileName VARCHAR(50),
 Rank INT)

/* Populate @Products for an any-words search */
IF @AllWords = 0
   INSERT INTO @Products
   SELECT ProductID, Name,
          SUBSTRING(Description, 1, @DescriptionLength) + '...' AS
```

```
Description,
          Price, Image1FileName, Image2FileName,
          3 * dbo.WordCount(@Word1, Name) + dbo.WordCount(@Word1,
Description) +
          3 * dbo.WordCount(@Word2, Name) + dbo.WordCount(@Word2,
Description) +
          3 * dbo.WordCount(@Word3, Name) + dbo.WordCount @Word3,
Description) +
          3 * dbo.WordCount(@Word4, Name) + dbo.WordCount(@Word4,
Description) +
          3 * dbo.WordCount(@Word5, Name) + dbo.WordCount(@Word5,
Description)
          AS Rank
   FROM Product
   ORDER BY Rank DESC

/* Populate @Products for an all-words search */
IF @AllWords = 1
   INSERT INTO @Products
   SELECT ProductID, Name,
          SUBSTRING(Description, 1, @DescriptionLength) + '...' AS
Description,
          Price, Image1FileName, Image2FileName,
          (3 * dbo.WordCount(@Word1, Name) + dbo.WordCount(@Word1,
Description)) +
          CASE
            WHEN @Word2 IS NULL THEN 1
            ELSE 3 * dbo.WordCount(@Word2, Name) +
dbo.WordCount(@Word2, Description)
          END +
          CASE
            WHEN @Word3 IS NULL THEN 1
            ELSE 3 * dbo.WordCount(@Word3, Name) +
dbo.WordCount(@Word3, Description)
          END +
          CASE
            WHEN @Word4 IS NULL THEN 1
            ELSE 3 * dbo.WordCount(@Word4, Name) +
dbo.WordCount(@Word4, Description)
          END +
          CASE
            WHEN @Word5 IS NULL THEN 1
            ELSE 3 * dbo.WordCount(@Word5, Name) +
dbo.WordCount(@Word5, Description)
          END
          AS Rank
   FROM Product
   ORDER BY Rank DESC

/* Save the number of searched products in an output variable */
SELECT @HowManyResults = COUNT(*)
FROM @Products
WHERE Rank > 0

/* Send back the requested products */
SELECT ProductID, Name, Description, Price, Image1FileName,
```

```
Image2FileName, Rank
FROM @Products
WHERE Rank > 0
  AND RowNumber BETWEEN (@PageNumber-1) * @ProductsPerPage + 1
                    AND @PageNumber * @ProductsPerPage
ORDER BY Rank DESC

GO

CREATE PROCEDURE AddDepartment
(@DepartmentName VARCHAR(50),
@DepartmentDescription VARCHAR(1000))
AS
INSERT INTO Department (Name, Description)
VALUES (@DepartmentName, @DepartmentDescription)

GO

CREATE PROCEDURE UpdateDepartment
(@DepartmentID INT,
@DepartmentName VARCHAR(50),
@DepartmentDescription VARCHAR(1000))
AS
UPDATE Department
SET Name = @DepartmentName, Description = @DepartmentDescription
WHERE DepartmentID = @DepartmentID

GO

CREATE PROCEDURE DeleteDepartment
(@DepartmentID INT)
AS
DELETE FROM Department
WHERE DepartmentID = @DepartmentID

GO

CREATE PROCEDURE CreateCategory
(@DepartmentID INT,
@CategoryName VARCHAR(50),
@CategoryDescription VARCHAR(50))
AS
INSERT INTO Category (DepartmentID, Name, Description)
VALUES (@DepartmentID, @CategoryName, @CategoryDescription)

GO

CREATE PROCEDURE UpdateCategory
(@CategoryID INT,
@CategoryName VARCHAR(50),
@CategoryDescription VARCHAR(1000))
AS
UPDATE Category
SET Name = @CategoryName, Description = @CategoryDescription
WHERE CategoryID = @CategoryID
```

```
GO

CREATE PROCEDURE DeleteCategory
(@CategoryID INT)
AS
DELETE FROM Category
WHERE CategoryID = @CategoryID

GO

CREATE PROCEDURE GetAllProductsInCategory
(@CategoryID INT)
AS
SELECT Product.ProductID, Name, Description, Price, Image1FileName,
       Image2FileName, OnDepartmentPromotion, OnCatalogPromotion
FROM Product INNER JOIN ProductCategory
  ON Product.ProductID = ProductCategory.ProductID
WHERE ProductCategory.CategoryID = @CategoryID

GO

CREATE PROCEDURE CreateProduct
(@CategoryID INT,
 @ProductName VARCHAR(50),
 @ProductDescription VARCHAR(1000),
 @ProductPrice MONEY,
 @Image1FileName VARCHAR(50),
 @Image2FileName VARCHAR(50),
 @OnDepartmentPromotion BIT,
 @OnCatalogPromotion BIT)
AS
-- Declare a variable to hold the generated product ID
DECLARE @ProductID INT
-- Create the new product entry
INSERT INTO Product
     (Name,
      Description,
      Price,
      Image1FileName,
      Image2FileName,
      OnDepartmentPromotion,
      OnCatalogPromotion )
VALUES
     (@ProductName,
      @ProductDescription,
      @ProductPrice,
      @Image1FileName,
      @Image2FileName,
      @OnDepartmentPromotion,
      @OnCatalogPromotion)
-- Save the generated product ID to a variable
SELECT @ProductID = @@Identity
-- Associate the product with a category
INSERT INTO ProductCategory (ProductID, CategoryID)
VALUES (@ProductID, @CategoryID)
```

```
GO

CREATE PROCEDURE UpdateProduct
(@ProductID INT,
 @ProductName VARCHAR(50),
 @ProductDescription VARCHAR(5000),
 @ProductPrice MONEY,
 @Image1FileName VARCHAR(50),
 @Image2FileName VARCHAR(50),
 @OnDepartmentPromotion BIT,
 @OnCatalogPromotion BIT)
AS
UPDATE Product
SET Name = @ProductName,
    Description = @ProductDescription,
    Price = @ProductPrice,
    Image1FileName = @Image1FileName,
    Image2FileName = @Image2FileName,
    OnDepartmentPromotion = @OnDepartmentPromotion,
    OnCatalogPromotion = @OnCatalogPromotion
WHERE ProductID = @ProductID

GO

CREATE PROCEDURE MoveProductToCategory
(@ProductID INT, @OldCategoryID INT, @NewCategoryID INT)
AS
UPDATE ProductCategory
SET CategoryID = @NewCategoryID
WHERE CategoryID = @OldCategoryID
  AND ProductID = @ProductID

GO

CREATE PROCEDURE AssignProductToCategory
(@ProductID INT, @CategoryID INT)
AS
INSERT INTO ProductCategory (ProductID, CategoryID)
VALUES (@ProductID, @CategoryID)

GO

CREATE PROCEDURE RemoveProductFromCategory
(@ProductID INT, @CategoryID INT)
AS
DELETE FROM ProductCategory
WHERE CategoryID = @CategoryID AND ProductID = @ProductID

GO

CREATE PROCEDURE DeleteProduct
(@ProductID INT)
AS
DELETE FROM ProductCategory WHERE ProductID=@ProductID
DELETE FROM Product where ProductID=@ProductID
```

```
GO

CREATE PROCEDURE GetCategoriesWithProduct
(@ProductID INT)
AS
SELECT Category.CategoryID, Name
FROM Category INNER JOIN ProductCategory
ON Category.CategoryID = ProductCategory.CategoryID
WHERE ProductCategory.ProductID = @ProductID

GO

CREATE PROCEDURE GetCategoriesWithoutProduct
(@ProductID INT)
AS
SELECT CategoryID, Name
FROM Category
WHERE CategoryID NOT IN
    (SELECT Category.CategoryID
     FROM Category INNER JOIN ProductCategory
     ON Category.CategoryID = ProductCategory.CategoryID
     WHERE ProductCategory.ProductID = @ProductID)

GO

CREATE PROCEDURE ShoppingCartAddItem
 @CartID char(36),
 @ProductID INT)
AS
IF EXISTS
         (SELECT CartID
          FROM ShoppingCart
          WHERE ProductID = @ProductID AND CartID = @CartID)
    UPDATE ShoppingCart
    SET Quantity = Quantity + 1
    WHERE ProductID = @ProductID AND CartID = @CartID
ELSE
    IF EXISTS (SELECT Name FROM Product WHERE ProductID=@ProductID)
        INSERT INTO ShoppingCart (CartID, ProductID, Quantity,
DateAdded)
        VALUES (@CartID, @ProductID, 1, GETDATE())

GO

CREATE PROCEDURE ShoppingCartGetItems
(@CartID char(36))
AS
SELECT Product.ProductID, Product.Name, Product.Price,
ShoppingCart.Quantity,
       Product.Price * ShoppingCart.Quantity AS Subtotal
FROM ShoppingCart INNER JOIN Product
ON ShoppingCart.ProductID = Product.ProductID
WHERE ShoppingCart.CartID = @CartID

GO
```

```
CREATE PROCEDURE ShoppingCartGetTotalAmount
(@CartID char(36))
AS
SELECT ISNULL(SUM(Product.Price * ShoppingCart.Quantity), 0)
FROM ShoppingCart INNER JOIN Product
ON ShoppingCart.ProductID = Product.ProductID
WHERE ShoppingCart.CartID = @CartID

GO

CREATE PROCEDURE ShoppingCartRemoveItem
(@CartID char(36),
 @ProductID INT)
AS
DELETE FROM ShoppingCart
WHERE CartID = @CartID and ProductID = @ProductID

GO

CREATE Procedure ShoppingCartUpdateItem
(@CartID char(36),
 @ProductID INT,
 @Quantity INT)
As
IF @Quantity <= 0
  EXEC ShoppingCartRemoveItem @CartID, @ProductID
ELSE
  UPDATE ShoppingCart
  SET Quantity = @Quantity, DateAdded = GETDATE()
  WHERE ProductID = @ProductID AND CartID = @CartID

GO

ALTER PROCEDURE DeleteProduct
(@ProductID INT)
AS
DELETE FROM ShoppingCart WHERE ProductID=@ProductID
DELETE FROM ProductCategory WHERE ProductID=@ProductID
DELETE FROM Product where ProductID=@ProductID

GO

CREATE PROCEDURE ShoppingCartDeleteOldCarts
(@Days smallINT)
AS
DELETE FROM ShoppingCart
WHERE CartID IN
   (SELECT CartID
    FROM ShoppingCart
    GROUP BY CartID
    HAVING MIN(DATEDIFF(dd,DateAdded,GETDATE())) >= @Days)

GO

CREATE PROCEDURE ShoppingCartCountOldCarts
 @Days smallINT
```

```
AS
SELECT COUNT(CartID)
FROM ShoppingCart
WHERE CartID IN
  (SELECT CartID
   FROM ShoppingCart
   GROUP BY CartID
   HAVING MIN(DATEDIFF(dd,DateAdded,GETDATE())) >= @Days)

GO

CREATE PROCEDURE CreateOrder
(@CartID char(36))
AS
/* Insert a new record INTo Orders */
DECLARE @OrderID INT
INSERT INTO Orders DEFAULT VALUES
/* Save the new Order ID */
SET @OrderID = @@IDENTITY
/* Add the order details to OrderDetail */
INSERT INTO OrderDetail
      OrderID, ProductID, ProductName, Quantity, UnitCost)
SELECT
     @OrderID, Product.ProductID, Product.Name,
     ShoppingCart.Quantity, Product.Price
FROM Product JOIN ShoppingCart
ON Product.ProductID = ShoppingCart.ProductID
WHERE ShoppingCart.CartID = @CartID
/* Clear the shopping cart */
DELETE FROM ShoppingCart
WHERE CartID = @CartID
/* Return the Order ID */
SELECT @OrderID

GO

CREATE PROCEDURE OrderGetDetails
(@OrderID INT)
AS
SELECT Orders.OrderID,
       ProductID,
       ProductName,
       Quantity,
       UnitCost,
       Subtotal
FROM OrderDetail JOIN Orders
ON Orders.OrderID = OrderDetail.OrderID
WHERE Orders.OrderID = @OrderID

GO

CREATE PROCEDURE OrdersGetByRecent
(@Count smallINT)
AS
-- Set the number of rows to be returned
SET ROWCOUNT @Count
```

```
-- Get list of orders
SELECT OrderID, DateCreated, DateShipped,
       Verified, Completed, Canceled, CustomerName
FROM Orders
ORDER BY DateCreated DESC
-- Reset rowcount value
SET ROWCOUNT 0

GO

CREATE PROCEDURE OrdersGetByDate
(@StartDate SMALLDATETIME,
 @EndDate SMALLDATETIME)
AS
SELECT OrderID, DateCreated, DateShipped,
       Verified, Completed, Canceled, CustomerName
FROM Orders
WHERE DateCreated BETWEEN @StartDate AND @EndDate
ORDER BY DateCreated DESC

GO

CREATE PROCEDURE OrdersGetUnverifiedUncanceled
AS
SELECT OrderID, DateCreated, DateShipped,
       Verified, Completed, Canceled, CustomerName
FROM Orders
WHERE Verified=0 AND Canceled=0
ORDER BY DateCreated DESC

GO

CREATE PROCEDURE OrdersGetVerifiedUncompleted
AS
SELECT OrderID, DateCreated, DateShipped,
       Verified, Completed, Canceled, CustomerName
FROM Orders
WHERE Verified=1 AND Completed=0
ORDER BY DateCreated DESC

GO

CREATE PROCEDURE OrderGetInfo
(@OrderID INT)
AS
SELECT OrderID,
      (SELECT ISNULL(SUM(Subtotal), 0) FROM OrderDetail WHERE OrderID =
@OrderID)
       AS TotalAmount,
       DateCreated,
       DateShipped,
       Verified,
       Completed,
       Canceled,
       Comments,
       CustomerName,
```

```
        ShippingAddress,
        CustomerEmail
FROM Orders
WHERE OrderID = @OrderID

GO

CREATE PROCEDURE OrderUpdate
 @OrderID INT,
 @DateCreated SMALLDATETIME,
 @DateShipped SMALLDATETIME = NULL,
 @Verified BIT,
 @Completed BIT,
 @Canceled BIT,
 @Comments VARCHAR(200),
 @CustomerName VARCHAR(50),
 @ShippingAddress VARCHAR(200),
 @CustomerEmail VARCHAR(50)
AS
UPDATE Orders
SET DateCreated=@DateCreated,
    DateShipped=@DateShipped,
    Verified=@Verified,
    Completed=@Completed,
    Canceled=@Canceled,
    Comments=@Comments,
    CustomerName=@CustomerName,
    ShippingAddress=@ShippingAddress,
    CustomerEmail=@CustomerEmail
WHERE OrderID = @OrderID

GO

CREATE PROCEDURE OrderMarkVerified
 @OrderID INT
AS
UPDATE Orders
SET Verified = 1
WHERE OrderID = @OrderID

GO

CREATE PROCEDURE OrderMarkCompleted
 @OrderID INT
AS
UPDATE Orders
SET Completed = 1,
    DateShipped = GETDATE()
WHERE OrderID = @OrderID

GO

CREATE PROCEDURE OrderMarkCanceled
 @OrderID INT
AS
UPDATE Orders
```

```
SET Canceled = 1
WHERE OrderID = @OrderID

GO
CREATE PROCEDURE GetProductRecommendations
(@ProductID INT,
 @DescriptionLength INT)
AS
SELECT ProductID,
       Name,
       SUBSTRING(Description, 1, @DescriptionLength) + '...' AS
Description
FROM Product
WHERE ProductID IN
    (
    SELECT TOP 5 od2.ProductID
    FROM OrderDetail od1
    JOIN OrderDetail od2 ON od1.OrderID = od2.OrderID
    WHERE od1.ProductID = @ProductID AND od2.ProductID != @ProductID
    GROUP BY od2.ProductID
    ORDER BY COUNT(od2.ProductID) DESC
    )

GO

CREATE PROCEDURE GetProductRecommendations2
(@ProductID INT,
 @DescriptionLength INT)
AS
--- Returns the product recommendations
SELECT ProductID,
       Name,
       SUBSTRING(Description, 1, @DescriptionLength) + '...' AS
Description
FROM Product
WHERE ProductID IN
   (
   -- Returns the products that were ordered together with @ProductID
   SELECT TOP 5 ProductID
   FROM OrderDetail
   WHERE OrderID IN
      (
      -- Returns the orders that contain @ProductID
      SELECT DISTINCT OrderID
      FROM OrderDetail
      WHERE ProductID = @ProductID
      )
   -- Must not include products that already exist in the visitor''s
cart
   AND ProductID <> @ProductID
   -- Group the ProductID so we can calculate the rank
   GROUP BY ProductID
   -- Order descending by rank
   ORDER BY COUNT(ProductID) DESC
   )
```

```
GO

CREATE PROCEDURE GetShoppingCartRecommendations
(@CartID CHAR(36),
 @DescriptionLength INT)
AS
--- Returns the product recommendations
SELECT ProductID,
       Name,
       SUBSTRING(Description, 1, @DescriptionLength) + '...' AS
Description
FROM Product
WHERE ProductID IN
   (
   -- Returns the products that exist in a list of orders
   SELECT TOP 5 od1.ProductID AS Rank
   FROM OrderDetail od1
     JOIN OrderDetail od2
       ON od1.OrderID=od2.OrderID
     JOIN ShoppingCart sp
       ON od2.ProductID = sp.ProductID
   WHERE sp.CartID = @CartID
        -- Must not include products that already exist in the
visitor''s cart
     AND od1.ProductID NOT IN
      (
      -- Returns the products in the specified shopping cart
      SELECT ProductID
      FROM ShoppingCart
      WHERE CartID = @CartID
      )
   -- Group the ProductID so we can calculate the rank
   GROUP BY od1.ProductID
   -- Order descending by rank
   ORDER BY COUNT(od1.ProductID) DESC
   )

GO

CREATE PROCEDURE GetShoppingCartRecommendations2
(@CartID CHAR(36),
 @DescriptionLength INT)
AS
--- Returns the product recommendations
SELECT ProductID,
       Name,
       SUBSTRING(Description, 1, @DescriptionLength) + '...' AS
Description
FROM Product
WHERE ProductID IN
   (
   -- Returns the products that exist in a list of orders
   SELECT TOP 5 ProductID
   FROM OrderDetail
   WHERE OrderID IN
      (
```

```
      -- Returns the orders that contain certain products
      SELECT DISTINCT OrderID
      FROM OrderDetail
      WHERE ProductID IN
         (
         -- Returns the products in the specified shopping cart
         SELECT ProductID
         FROM ShoppingCart
         WHERE CartID = @CartID
         )
   -- Must not include products that already exist in the visitor''s
cart
   AND ProductID NOT IN
      (
      -- Returns the products in the specified shopping cart
      SELECT ProductID
      FROM ShoppingCart
      WHERE CartID = @CartID
      )
   -- Group the ProductID so we can calculate the rank
   GROUP BY ProductID
   -- Order descending by rank
   ORDER BY COUNT(ProductID) DESC
   )
```

### 4.3.11.3 Συναρτήσεις (Functions)

```
CREATE FUNCTION dbo.WordCount
(@Word VARCHAR(15),
@Phrase VARCHAR(1000))
RETURNS SMALLINT
AS
BEGIN

/* If @Word or @Phrase is NULL the function returns 0 */
IF @Word IS NULL OR @Phrase IS NULL RETURN 0

/* @BiggerWord is a string one character longer than @Word */
DECLARE @BiggerWord VARCHAR(21)
SELECT @BiggerWord = @Word + 'x'

/* Replace @Word with @BiggerWord in @Phrase */
DECLARE @BiggerPhrase VARCHAR(2000)
SELECT @BiggerPhrase = REPLACE (@Phrase, @Word, @BiggerWord)

/* The length difference between @BiggerPhrase and @phrase
   is the number we''re looking for */
RETURN LEN(@BiggerPhrase) - LEN(@Phrase)
END
```

# Βιβλιογραφία


1. David Avison - Guy Fitzgerald, *Προηγμένα Πληροφοριακά Συστήματα : Απο την θεωρία στην πράξη*, Εκδόσεις Νέων Τεχνολογιών, *Επιμέλεια Ελληνικής Έκδοσης : Νικ. Σπ. Βώρος - Γρ. Ν. Μπεληγιάννης - Γ. Αθ. Τσιρογιάννης*

2. Βασίλης Γ. Λαοπόδης, *Ανάπτυξη Πληροφοριακών Συστημάτων : Αναλυσή & Σχεδιασμός Συστημάτων*, Εκδόσεις Νέων Τεχνολογιών, *Αθήνα*

3. Φείδας Χρήστος, *Σημειώσεις Μαθήματος Ανάλυση & Σχεδιασμός Πληροφοριακών Συστημάτων II*, Μεσολόγγι 2006

4. R.Elmasri – S.B.Navathe, *Θεμελιώδεις Αρχές Συστημάτων Βάσεων Δεδομένων*, Εκδόσεις Δίαυλος, Τόμος Α'

5. R.Elmasri – S.B.Navathe, *Θεμελιώδεις Αρχές Συστημάτων Βάσεων Δεδομένων*, Εκδόσεις Δίαυλος, Τόμος Β'

6. Καραγιάννης Γεώργιος, *Σημειώσεις Μαθήματος Ασφάλειά Πληροφοριακών Συστημάτων*, Μεσολόγγι 2003

7. Επιτήδειος Γεώργιος, *Μαθήματα HTML*,ΕΕΕΙ Ελληνική Ένωση Επαγγελματιών Internet

8. Συρμακέσης Σπυρίδων, *Σημειώσεις Μαθήματος Ηλεκτρονικό Εμπόριο*, Μεσολόγγι

9. Σταυρακούδης Αθανάσιος, *Σημειώσεις Μαθήματος Βάσεις Δεδομένων II*, Μεσολόγγι 2005

10. Apress – *Beginning ASP.NET 2.0 E-Commerce in C# 2005* – From Novice to Professional