ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΕΙΡΑΙΑ

Τ.Ε.Ι ΠΕΙΡΑΙΑ

ΑΝΑΠΤΥΞΗ ΔΙΑΔΡΑΣΤΙΚΗΣ ΙΣΤΟΣΕΛΙΔΑΣ ΠΑΡΑΓΓΕΛΙΩΝ ΛΙΑΝΙΚΗΣ ΣΕ ΕΡΓΟΣΤΑΣΙΟ

ΒΑΣΙΛΕΙΟΣ ΠΑΣΣΙΟΣ Α.Μ:37992
ΙΩΑΝΝΗΣ ΑΓΓΕΛΟΠΟΥΛΟΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΓΜΟΓΩΝ
ΤΜΗΜΑ ΑΥΤΟΜΑΤΙΣΜΟΥ

ΙΑΝΟΥΑΡΙΟΣ 2014 (16/01/2014)

ΤΜΗΜΑ ΑΥΤΟΜΑΤΙΣΜΟΥ
Π.ΡΑΛΛΗ & ΘΗΒΩΝ 250,12244 ΑΙΓΑΛΕΩ,ΑΘΗΝΑ-ΕΛΛΑΔΑ
ΤΗΛ. 210-5381488

Table of Contents

Εισαγωγή

Στο παρών σύγγραμμα θα υλοποιήσουμε μια διαδραστική εφαρμογή ιστού (web application), η οποία θα έχει την μορφή μιας ιστοσελίδας (web page) για παραγγελίες λιανικής σε εργοστάσιο. Η εφαρμογή θα υλοποιηθεί πάνω στην πλατφόρμα ASP.net που παρέχεται από το πλαίσιο (framework) .Net της Microsoft, με τη χρήση της γλώσσας προγραμματισμού C#. Επίσης, θα χρησιμοποιηθεί η σουίτα προγραμμάτων του Visual Studio Express και τα δεδομένα της εφαρμογής θα αποθηκεύονται στην βάση δεδομένων (database) του SQL Server Express. Τα δύο αυτά προγράμματα παρέχονται δωρεάν από την ιστοσελίδα της Microsoft. Τέλος, η εφαρμογή εκτελείται πάνω σε ένα διακομιστή ιστού (web server), τον IIS Express που περιέχεται στο Visual Studio Express.

Το σύνολο της εφαρμογής θα αποτελείται από ένα ολοκληρωμένο σύστημα λήψης και διαχείρισης παραγγελίων πελατών. Συγκεκριμένα, ο πελάτης υποβάλει την παραγγελία, το σύστημα ελέγχει τα αποθέματα των προϊόντων και ο διαχειριστής ελέγχει την ομαλή πορεία της παραγγελίας από την συγκέντρωση των απαιτούμενων προϊόντων εώς την ολοκλήρωση της αποστολής αυτών στον πελάτη. Πάνω σ' αυτήν την βασική ιδέα, το σύστημα χωρίζεται σε δύο βασικά υποσυστήματα. Το πρώτο αφορά τον πελάτη, ο οποίος μπορεί να παραγγέλνει προϊόντα της αρεσκείας του και να ελέγχει την κατάσταση των παραγγελιών του ανά πάσα στιγμή. Το δεύτερο αφορά τον διαχειριστή, ο οποίος διαχειρίζεται τα προϊόντα, τα αποθέματα αυτών, τις προσφορές και άλλα στοιχεία, καθώς και την ομαλή ολοκλήρωση των παραγγελίων των πελατών.

Δημιουργία βάσης δεδομένων (Database)

Το Visual Studio Express δημιουργεί αυτομάτως μια βάση δεδομένων (database) σύμφωνα με τις αγγλικές προδιαγραφές. Αυτό όμως δεν μας ικανοποιεί, καθώς επιθυμούμε ελληνικές προδιαγραφές. Για μετατροπή της βάσης δεδομένων (database) από την εξ ορισμού κωδικοποίση αγγλικών σε ελληνικά, προσθέτουμε τους διακόπτες ci και ai, γιατί εκτός από ελληνικά, δεν θέλουμε επίσης να περιορίζουμε τις αναζητήσεις των πελατών από τόνους, κεφαλαία και μικρά. Δηλαδή, η αναζήτηση είναι ανεξάρτητη από γραμματικούς περιορισμούς, δεδομένου ότι οι πελάτες δεν ακολουθούν πάντα τους κανόνες αναζήτησης (άλλοι ψάχνουν με μικρά, άλλοι εξαιρούν τους τόνους για ταχύτητα, ενώ άλλοι προτιμούν τα κεφαλαία). Για να το πραγματοποιήσουμε αυτό, εκτελούμε την ακόλουθη εντολή SQL:

```
// Συγκεκριμένη για την παρούσα βάση.
SELECT name, collation_name FROM sys.databases;
GO
ALTER DATABASE [FACTORYSALES.MDF]  SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
GO
ALTER DATABASE [FACTORYSALES.MDF] COLLATE Greek_100_ci_ai;
GO
ALTER DATABASE [FACTORYSALES.MDF]  SET MULTI_USER;
GO
SELECT name, collation_name FROM sys.databases;
GO

// Σημείωση: full path για την βάση FACTORYSALES.MDF.
```

Δημιουργία δικαιωμάτων διαχειριστή (role)

Το Asp.net membership διαχειρίζεται αυτομάτως τους χρήστες. Συγκεκριμένα κάθε ένας χρήστης που μπαίνει στην εφαρμογή μας χαρακτηρίζεται ως ανώνυμος (anonymous). Αυτός ο χρήστης δεν έχει δικαιώματα χρήσης επι της εφαρμογής παρά

μόνο να βλέπει συγκεκριμένα μέρη που δεν απαιτούν δικαιώματα χρήσης. Τέτοια κομμάτια της εφαρμογής είναι η αρχική σελίδα,η λίστα των προϊόντων, οι πληροφορίες για την εταιρεία και η επικοινωνία με τον διαχειριστή. Αυτός ο νέος χρήστης μπορεί να αποκτήσει δικαιώματα χρήσης εφόσον ολοκληρώσει την εγγραφή το σύστημα. Με τον τρόπο αυτό αποκτάει την δυνατότητα να πραγματοποιεί παραγγελίες και να ελέγχει την κατάσταση τους ανά πάσα στιγμή.

Εξαίρεση αποτελεί ο διαχειριστής, ο οποίος χρειάζεται ειδικά δικαιώματα επί της εφαρμογής, όπως καταχώρηση προϊόντων, διαχείριση αποθεμάτων, διαχείριση αποθηκών, έλεγχος παραγγελιών πελατών και άλλα. Για να το κάνουμε αυτό, πρέπει να ορίσουμε ότι ο συγκεκριμένος χρήστης δηλαδή ο διαχειριστής έχει δικαιώματα γι' αυτό το σκοπό, καθορίζοντας έναν ειδικό ρόλο (role) με την χρήση του ASP.net Configuration Tool. Επειδή, το Visual Studio Express δεν μας δίνει έναν απευθείας τρόπο γι΄ αυτό (η λειτουργία αυτή υπήρχε στις παλιότερες εκδόσεις, αλλά δεν υπάρχει στην παρούσα), πρέπει να εκτελέσουμε μια σειρά εντολών μέσω της κονσόλας (console) για να το πραγματοποιήσουμε. Οι εντολές αυτές είναι οι ακόλουθες:

```
cmd -> cd\program Files\iis express (folder of iisexpress.exe)
iisexpress.exe
/path:c:\windows\microsoft.net\framework\v4.0.30319\asp.netwebadminfiles
/vpath:/asp.netwebadminfiles /port:[port] /clr:4.0 /ntlm
Application running url:
http://localhost:[port]/asp.netwebadminfiles/default.aspx?applicationphysicalpa
th=[application]&applicationurl=/

[application] = Η φυσική διαδρομή των αρχείων. Παράδειγμα
C:\Users\Username\Documents\Visual Studio
2013\Projects\FactorySales\FactorySales
[port] = Η port της εφαρμογής όταν εκτελείται στον broswer. Παράδειγμα: 53448
```

Σύστημα διαχείρισης λογαριασμών χρηστών

Διαχείριση λογαριασμού

Manage.aspx.cs

Ο ακόλουθος κώδικας διαχειρίζεται τους λογαριασμούς των χρηστών και συγκεκριμένα τις αλλαγές κωδικών (passwords):

```
protected void Page_Load()
        {
            // Αλλάζουμε το μήνυμα αποτυχίας αλλαγής κωδικού χρήστη ώστε να
είναι γραμμένο στα ελληνικά.
            chpwd_change_password.ChangePasswordFailureText = "Λανθασμένος
κωδικός χρήστη ή άκυρος. Το μέγεθος του κωδικού πρέπει να είναι το λιγότερο
{0}" +
                " χαρακτήρες, από τους οποίους οι μη αλφαριθμητικοί να είναι
{1}.";

            if (!IsPostBack)
            {
                // Determine the sections to render
                var hasLocalPassword =
OpenAuth.HasLocalPassword(User.Identity.Name);
                setPassword.Visible = !hasLocalPassword;
                changePassword.Visible = hasLocalPassword;

                CanRemoveExternalLogins = hasLocalPassword;

                // Render success message
                var message = Request.QueryString["m"];
```

```
                if (message != null)
                {
                    // Strip the query string from action
                    Form.Action = ResolveUrl("~/Account/Manage");

                    SuccessMessage =
                        message == "ChangePwdSuccess" ? "Ο κωδικός χρήστη
άλλαξε."
                        : message == "SetPwdSuccess" ? "Ο κωδικός χρήστη
ορίστηκε."
                        : message == "RemoveLoginSuccess" ? "Ο εξωτερικός
σύνδεσμος έχει αφαιρεθεί."
                        : String.Empty;
                    successMessage.Visible =
!String.IsNullOrEmpty(SuccessMessage);
                }
            }
        }
```

Εγγραφή χρήστη

Register.aspx.cs

Ο ακόλουθος κώδικας διαχειρίζεται την εγγραφή του χρήστη στο σύστημα. Επιπλέον, προσθέτει μια εγγραφή στον πίνακα διευθύνσεων πελατών (Customer Addresses) όταν ολοκληρωθεί επιτυχώς η εγγραφή:

```
protected void RegisterUser_CreatedUser(object sender, EventArgs e)
        {
            FormsAuthentication.SetAuthCookie(RegisterUser.UserName,
createPersistentCookie: false);

            string continueUrl = RegisterUser.ContinueDestinationPageUrl;

            // Συλλογή των στοιχείων λογαριασμού του χρήστη που μόλις εγγράφηκε
στο σύστημα (Ο χρήστης συνδέεται στο σύστημα αυτόματα, ύστερα από την επικύρωση
των
            // στοιχείων του).
            MembershipUser msu_user =
Membership.GetUser(RegisterUser.UserName);

            // Αν η συλλογή των στοιχείων του είναι επιτυχής, τότε δημιουργούμε
μια εγγραφή στον πίνακα διευθύνσεων πελατών (CustomersAddreesses), όπου θα
            // αποθηκεύονται τα προσωπικά στοιχεία του (Τα στοιχεία αυτά
χρησιμοποιούνται κυρίως για την εκτύπωση αποδείξεων παραγγελιών, αλλά και να
γνωρίζουμε ως
            // πωλητές που θα αποστείλουμε τα προϊόντα που παρήγγειλε ο χρήστης
από εμάς).
            if (msu_user != null)
            {
                sqlds_customer_address.Insert();
            }

            if (!OpenAuth.IsLocalUrl(continueUrl))
            {
                continueUrl = "~/";
            }
            Response.Redirect(continueUrl);
        }

        protected void sqlds_customer_address_Inserting(object sender,
SqlDataSourceCommandEventArgs e)
```

```
        {
            // Συλλογή των στοιχείων λογαριασμού του χρήστη που μόλις εγγράφηκε
στο σύστημα (Ο χρήστης συνδέεται στο σύστημα αυτόματα, ύστερα από την επικύρωση
των
            // στοιχείων του).
            MembershipUser msu_user =
Membership.GetUser(RegisterUser.UserName);

            // Αν η συλλογή των στοιχείων του είναι επιτυχής, τότε θα
δημιουργήσουμε μια εγγραφή στον πίνακα διευθύνσεων πελατών
(CustomersAddreesses). Η εγγραφή
            // αυτή θα είναι συνδεδεμένη με το λογαριασμό χρήστη μέσω ενός
μοναδικού προσδιοριστικού (Unique ID) και χαρακτηρίζει την μοναδικότητα της
εγγραφής ανά
            // εγγεγραμμένο χρήστη (κάθε χρήστης έχει μία μόνο εγγραφή με τα
προσωπικά του στοιχεία).
            if (msu_user != null)
            {
                Guid guid_customer = (Guid)msu_user.ProviderUserKey; //
Μοναδικό προσδιοριστικό (Unique ID) συνδεδεμένου χρήστη.

                // Εισαγωγή στο σύστημα του μοναδικού προσδιοριστικού (Unique
ID) για την δημιουργία της εγγραφής στον διευθύνσεων πελατών
(CustomersAddreesses).
                e.Command.Parameters["@CustomerUserID"].Value = guid_customer;
            }
        }
```

Εργαλεία διαχειριστή

Διαχείριση εκπτώσεων

DiscountsEntry.aspx.cs

Ο ακόλουθος κώδικας επιτρέπει στον διαχειριστή να διαχειρίζεται τις διαθέσιμες εκπτώσεις για τα καταχωρημένα προϊόντα:

```
protected void Page_Load(object sender, EventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν φορτώνεται στον φυλλομετρητή
(browser) μια σελίδα καταχώρησης διαχειριστή.
            CodeClass.uf_v_AdministatorPageLoad(Page, gvw_discounts,
fvw_discount);
        }

        protected void fvw_discount_ItemDeleted(object sender,
FormViewDeletedEventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν διαγραφεί μια εγγραφή σε μια
πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση δεδομένων)
            // μέσω της φόρμας όψεως (FormView), σε μια φόρμα καταχώρησης
διαχειριστή.
            CodeClass.uf_v_AdministratorFormViewItemDeleted(gvw_discounts,
fvw_discount);
        }

        protected void fvw_discount_ItemInserted(object sender,
FormViewInsertedEventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν εισαχθεί μια εγγραφή σε μια
πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση δεδομένων)
```

```
            // μέσω της φόρμας όψεως (FormView), σε μια φόρμα καταχώρησης
διαχειριστή.
            CodeClass.uf_v_AdministratorFormViewItemInserted(gvw_discounts,
fvw_discount);
        }

        protected void fvw_discount_ItemUpdated(object sender,
FormViewUpdatedEventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν ενημερωθεί μια εγγραφή σε μια
πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση δεδομένων)
            // μέσω της φόρμας όψεως (FormView), σε μια φόρμα καταχώρησης
διαχειριστή.
            CodeClass.uf_v_AdministratorFormViewItemUpdated(gvw_discounts,
fvw_discount);
        }

        protected void gvw_discounts_RowDataBound(object sender,
GridViewRowEventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν συνδεθούν τα δεδομένα μιας
εγγραφής από μια πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση
δεδομένων)
            // στον πίνακα όψεως (GridView), σε μια φόρμα καταχώρησης
διαχειριστή.
            CodeClass.uf_v_AdministratorGridViewRowDataBound(gvw_discounts,
fvw_discount);
        }

        protected void sqlds_discount_Deleted(object sender,
SqlDataSourceStatusEventArgs e)
        {
            // Έλεγχος παραμέτρων πηγής δεδομένων (Data Source) για σφάλματα σε
μια σελίδα καταχώρησης διαχειριστή.
            CodeClass.uf_v_AdministatorSqlDataSourceDeleted(e, lbl_message);
        }
```

Διαχείριση παραγγελιών πελατών

Orders.aspx.cs

Ο ακόλουθος κώδικας επιτρέπει στον διαχειριστή να ελέγχει και να διαχειρίζεται την κατάσταση των παραγγελιών των πελατών, καθώς επίσης να εκτυπώνει (μέσω του εκτυπωτικού Order Print) τις παραγγελίες αυτές:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

using FactorySales.Code;

namespace FactorySales.Administrator
{
    public partial class Orders : System.Web.UI.Page
    {
        private bool priv_b_collect; // Μεταβλητή αποθήκευσης της κατάστασης
της συλλογής των προϊόντων.
        private bool priv_b_pricing; // Μεταβλητή αποθήκευσης της κατάστασης
της τιμολόγησης της παραγγελίας.
```

```
        private bool priv_b_dispatch; // Μεταβλητή αποθήκευσης της κατάστασης
της αποστολής της παραγγελίας.
        private bool priv_b_delivery; // Μεταβλητή αποθήκευσης της κατάστασης
της παραλαβής της παραγγελίας.
        private bool priv_b_cancel; // Μεταβλητή αποθήκευσης της κατάστασης της
ακύρωσης της παραγγελίας.
        private bool priv_b_complete; // Μεταβλητή αποθήκευσης της κατάστασης
της ολοκλήρωσης της παραγγελίας.
        private DateTime priv_dt_submit_date; // Μεταβλητή αποθήκευσης της
ημερομηνίας και ώρας (DateTime) υποβολής της παραγγελίας.
        private DateTime priv_dt_collect_date;  // Μεταβλητή αποθήκευσης της
ημερομηνίας και ώρας (DateTime) συλλογής των προϊόντων.
        private DateTime priv_dt_pricing_date;  // Μεταβλητή αποθήκευσης της
ημερομηνίας και ώρας (DateTime) τιμολόγησης της παραγγελίας.
        private DateTime priv_dt_dispatch_date; // Μεταβλητή αποθήκευσης της
ημερομηνίας και ώρας (DateTime) αποστολής της παραγγελίας.
        private DateTime priv_dt_delivery_date;  // Μεταβλητή αποθήκευσης της
ημερομηνίας και ώρας (DateTime) παραλαβής της παραγγελίας.
        private DateTime priv_dt_cancel_date;  // Μεταβλητή αποθήκευσης της
ημερομηνίας και ώρας (DateTime) ακύρωσης της παραγγελίας.
        private DateTime priv_dt_complete_date;  // Μεταβλητή αποθήκευσης της
ημερομηνίας και ώρας (DateTime) ολοκλήρωσης της παραγγελίας.

        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void gvw_orders_SelectedIndexChanged(object sender, EventArgs
e)
        {
            // Εκτελούμε την εντολή sql για να συλλέξουμε τις ημερομηνίες και
ώρες (DateTime) υποβολής παραγγελίας, συλλογής προϊόντων, τιμολόγησης
παραγγελίας,
            // αποστολής παραγγελίας, παραλαβής παραγγελίας και ολοκλήρωσης
παραγγελίας, καθώς και τις καταστάσεις του κάθε σταδίου.
            sqlds_order_status.Select(DataSourceSelectArguments.Empty);

            // Αν για κάποιο λόγο τα στοιχεία της λίστας κουτιών επιλογής
(CheckBoxList) είναι λιγότερα από το αναμενώμενο, σταματάμε την διαδικασία για
να
            // προστατέψουμε το πρόγραμμα από σφάλματα τύπου εκτός ορίων.
            if (cbxl_order_status.Items.Count < 4)
            {
                return;
            }

            // Μεταβολή της επιλογής ενός κουμπίου της λίστας κουτιών επιλογής
(CheckBoxList) σύμφωνα με την κατάσταση της συλλογής προϊόντων, της τιμολόγησης
            // παραγγελίας, της αποστολής παραγγελίας και της παράδοσης
παραγγελίας.
            CodeClass.uf_v_CheckboxDateTimeControl(cbxl_order_status.Items[0],
priv_b_collect);
            CodeClass.uf_v_CheckboxDateTimeControl(cbxl_order_status.Items[1],
priv_b_pricing);
            CodeClass.uf_v_CheckboxDateTimeControl(cbxl_order_status.Items[2],
priv_b_dispatch);
            CodeClass.uf_v_CheckboxDateTimeControl(cbxl_order_status.Items[3],
priv_b_delivery);
        }
```

```
        protected void sqlds_order_status_Selected(object sender,
SqlDataSourceStatusEventArgs e)
        {
            // Λήψη των ημερομηνιών και ώρας (DateTime) της υποβολής
παραγγελίας, της συλλογής προϊόντων, της τιμολόγησης παραγγελίας, της αποστολής
παραγγελίας,
            // της παραλαβής παραγγελίας και της ολοκλήρωσης παραγγελίας από
τον πίνακα παραγγελίων (Orders) και μετατροπή τους στον κατάλληλο τύπο
δεδομένων.
            priv_dt_submit_date =
CodeClass.uf_dt_StringToDateTime(e.Command.Parameters["@SubmitDate"].Value.ToSt
ring());
            priv_dt_collect_date =
CodeClass.uf_dt_StringToDateTime(e.Command.Parameters["@CollectDate"].Value.ToS
tring());
            priv_dt_pricing_date =
CodeClass.uf_dt_StringToDateTime(e.Command.Parameters["@PricingDate"].Value.ToS
tring());
            priv_dt_dispatch_date =
CodeClass.uf_dt_StringToDateTime(e.Command.Parameters["@DispatchDate"].Value.To
String());
            priv_dt_delivery_date =
CodeClass.uf_dt_StringToDateTime(e.Command.Parameters["@DeliveryDate"].Value.To
String());
            priv_dt_complete_date =
CodeClass.uf_dt_StringToDateTime(e.Command.Parameters["@CompleteDate"].Value.To
String());

            // Σύγκριση της ημερομηνίας και ώρας (DateTime) της συλλογής
προϊόντων, της τιμολόγησης παραγγελίας, της αποστολής παραγγελίας, της
παραλαβής
            // παραγγελίας και της ολοκλήρωσης παραγγελίας με την ημερομηνία
και ώρα της υποβολής παραγγελίας και ενεργοποίηση των αντίστοιχων καταστάσεων
            // αληθείας (boolean). Οι καταστάσεις αυτές δηλώνουν ότι το κάθε
στάδιο από αυτά είναι ολοκληρωμένο ή όχι.
            priv_b_collect =
CodeClass.uf_b_DateTimeGreaterEqualCompare(priv_dt_collect_date,
priv_dt_submit_date);
            priv_b_pricing =
CodeClass.uf_b_DateTimeGreaterEqualCompare(priv_dt_pricing_date,
priv_dt_submit_date);
            priv_b_dispatch =
CodeClass.uf_b_DateTimeGreaterEqualCompare(priv_dt_dispatch_date,
priv_dt_submit_date);
            priv_b_delivery =
CodeClass.uf_b_DateTimeGreaterEqualCompare(priv_dt_delivery_date,
priv_dt_submit_date);
            priv_b_complete =
CodeClass.uf_b_DateTimeGreaterEqualCompare(priv_dt_complete_date,
priv_dt_submit_date);
        }

        protected void cbxl_order_status_SelectedIndexChanged(object sender,
EventArgs e)
        {
            // Εκτελούμε την εντολή sql για να συλλέξουμε τις ημερομηνίες και
ώρες (DateTime) υποβολής παραγγελίας, συλλογής προϊόντων, τιμολόγησης
παραγγελίας,
            // αποστολής παραγγελίας, παραλαβής παραγγελίας και ολοκλήρωσης
παραγγελίας, καθώς και τις καταστάσεις του κάθε σταδίου.
            sqlds_order_status.Select(DataSourceSelectArguments.Empty);
```

```
            // Αν η διαδικασία της παραγγελίας έχει ολοκληρωθεί ή ακυρωθεί,
τότε ο διαχειριστής δεν έχει να κάνει κάτι επί της παραγγελίας.
            if ((priv_b_complete == true) || (priv_b_cancel == true))
            {
                // Απενεργοποίηση των κουμπιών (Button) ακύρωσης και μεταβολής
κατάστασης παραγγελίας, καθώς και της λίστας κουτιών επιλογής (CheckBoxList)
                // κατάστασης παραγγελίας.
                cbxl_order_status.Enabled = false;
                btn_order_status.Enabled = false;
                btn_cancel.Enabled = false;
            }
            else // Διαφορετικά, ο διαχειριστής μπορεί να μεταβάλει την
κατάσταση της παραγγελίας σύμφωνα με την πρόοδό της διαδικασίας.
            {
                // Ενεργοποίηση της λίστας κουτιών επιλογής (CheckBoxList)
κατάστασης παραγγελίας.
                cbxl_order_status.Enabled = true;

                // Αν η διαδικασία της παραγγελίας δεν έχει ολοκληρωθεί ή
ακυρωθεί, τότε ενεργοποίησε τα κουμπιά (Button) ακύρωσης και μεταβολής
κατάστασης
                // παραγγελίας
                if ((priv_b_complete == false) && (priv_b_cancel == false))
                {
                    // Ενεργοποίηση των κουμπιών (Button) ακύρωσης και
μεταβολής κατάστασης παραγγελίας.
                    btn_order_status.Enabled = true;
                    btn_cancel.Enabled = true;
                }
            }

            // Αν για κάποιο λόγο τα στοιχεία της λίστας κουτιών επιλογής
(CheckBoxList) είναι λιγότερα από το αναμενόμενο, σταματάμε την διαδικασία για
να
            // προστατέψουμε το πρόγραμμα από σφάλματα τύπου εκτός ορίων.
            if (cbxl_order_status.Items.Count < 4)
            {
                return;
            }

            // Αν ο διαχειριστής επιλέξει το τέταρτο κουτί επιλογής, επέλεξε
και τα υπόλοιπα τρία.
            if (cbxl_order_status.Items[3].Selected == true)
            {
                cbxl_order_status.Items[0].Selected = true;
                cbxl_order_status.Items[1].Selected = true;
                cbxl_order_status.Items[2].Selected = true;
            }
            else if (cbxl_order_status.Items[2].Selected == true) //
Διαφορετικά, αν επιλέξει το τρίτο, τότε επέλεξε και τα δύο προηγούμενα αυτού.
            {
                cbxl_order_status.Items[0].Selected = true;
                cbxl_order_status.Items[1].Selected = true;
            }
            else if (cbxl_order_status.Items[1].Selected == true) // Τέλος, αν
επιλέξει το δεύτερο, τότε επέλεξε μόνο το πρώτο κουτί επιλογής.
            {
                cbxl_order_status.Items[0].Selected = true;
            }
        }

        protected void btn_order_status_Click(object sender, EventArgs e)
```

```
        {
            // Αν για κάποιο λόγο, δεν υπάρχουν διευθύνσεις αποθηκών στην
αναδιπλούμενη λίστα (DropDownList) των διευθύνσεων αποστολής, τότε δεν υπάρχει
νόημα να
            // συνεχίσουμε την διαδικασία.
            if (ddl_storage_address.Items.Count < 1)
            {
                return;
            }

            // Εκτελούμε την εντολή sql για να συλλέξουμε τις ημερομηνίες και
ώρες (DateTime) υποβολής παραγγελίας, συλλογής προϊόντων, τιμολόγησης
παραγγελίας,
            // αποστολής παραγγελίας, παραλαβής παραγγελίας και ολοκλήρωσης
παραγγελίας, καθώς και τις καταστάσεις του κάθε σταδίου.
            sqlds_order_status.Select(DataSourceSelectArguments.Empty);

            // Αν για κάποιο λόγο τα στοιχεία της λίστας κουτιών επιλογής
(CheckBoxList) είναι λιγότερα από το αναμενώμενο, σταματάμε την διαδικασία για
να
            // προστατέψουμε το πρόγραμμα από σφάλματα τύπου εκτός ορίων.
Ομοίως όταν η παραγγελία έχει ολοκληρωθεί.
            if ((priv_b_complete == true) || (cbxl_order_status.Items.Count <
4))
            {
                return;
            }

            // Στην πραγματικότητα η ακόλουθη τιμή δεν είναι null (κενή), αλλά
ίση με την ημερομηνία 1/1/2001 12:00:00 πμ.
            // Η ίδια τιμή υπάρχει και στις στήλες του αντίστοιχου πίνακα στη
βάση δεδομένων, όπου την χρησιμοποιούμε ως "κενή".
            DateTime dt_date_null = new DateTime(2001, 1, 1, 0, 0, 0);

            // Η ώρα του συστήματος αυτή την στιγμή. Η λήψη γίνεται από το
ρολόι του διακομιστή (server) και όχι του χρήστη (client).
            DateTime dt_date_now = DateTime.Now;

            // Ορισμός της ημερομηνίας και ώρας (DateTime) της συλλογής
προϊόντων, της τιμολόγησης παραγγελίας, της αποστολής παραγγελίας και της
παραλαβής
            // παραγγελίας με βάση την επιλογή του αντίστοιχου κουτιού στην
λίστα κουτιών επιλογής (CheckBoxList).
            DateTime dt_collect_date =
CodeClass.uf_dt_SetDateTimeByListItemSelection(cbxl_order_status.Items[0].Selec
ted, priv_dt_collect_date, dt_date_now);
            DateTime dt_pricing_date =
CodeClass.uf_dt_SetDateTimeByListItemSelection(cbxl_order_status.Items[1].Selec
ted, priv_dt_pricing_date, dt_date_now);
            DateTime dt_dispatch_date =
CodeClass.uf_dt_SetDateTimeByListItemSelection(cbxl_order_status.Items[2].Selec
ted, priv_dt_dispatch_date, dt_date_now);
            DateTime dt_delivery_date =
CodeClass.uf_dt_SetDateTimeByListItemSelection(cbxl_order_status.Items[3].Selec
ted, priv_dt_delivery_date, dt_date_now);

            // Σύγκριση της ημερομηνίας και ώρας (DateTime) της συλλογής
προϊόντων, της τιμολόγησης παραγγελίας, της αποστολής παραγγελίας και
            // της παραλαβής παραγγελίας με την ημερομηνία και ώρα της υποβολής
παραγγελίας και ενεργοποίηση των αντίστοιχων καταστάσεων αληθείας (boolean).
            // Οι καταστάσεις αυτές δηλώνουν ότι το κάθε στάδιο από αυτά είναι
ολοκληρωμένο ή όχι.
```

```
            priv_b_collect =
CodeClass.uf_b_DateTimeGreaterEqualCompare(dt_collect_date,
priv_dt_submit_date);
            priv_b_pricing =
CodeClass.uf_b_DateTimeGreaterEqualCompare(dt_pricing_date,
priv_dt_submit_date);
            priv_b_dispatch =
CodeClass.uf_b_DateTimeGreaterEqualCompare(dt_dispatch_date,
priv_dt_submit_date);
            priv_b_delivery =
CodeClass.uf_b_DateTimeGreaterEqualCompare(dt_delivery_date,
priv_dt_submit_date);

            // Μεταβλητή αποθήκευσης της ημερομηνίας και ώρας (DateTime) της
ολοκλήρωσης της παραγγελίας.
            DateTime dt_complete_date = new DateTime();

            // Αν τα τέσσερα βασικά στάδια της παραγγελίας έχουν ολοκληρωθεί,
τότε και η παραγγελία έχει ολοκληρωθεί και ενημερώνουμε την βάση δεδομένων για
αυτό.
            if ((priv_b_collect == true) && (priv_b_pricing == true) &&
(priv_b_dispatch == true) && (priv_b_delivery == true))
            {
                // Η ημερομηνία και ώρα αυτής της στιγμής αποτελεί και την
ημερομηνία και ώρα της ολοκλήρωσης της παραγγελίας.
                dt_complete_date = dt_date_now;

                // Έλεγχος αν είναι επιλεγμένη μια διεύθυνση αποθήκης ως
διεύθυνσης αποστολής στην αναδιπλούμενη λίστα (DropDownList). Στην ουσία είναι
πάντα
                // επιλεγμένη η πρώτη τιμή.
                if (ddl_storage_address.SelectedIndex > -1)
                {
                    // Εισαγωγή στο σύστημα του κωδικού καταχώρησης διεύθυνσης
αποθήκης για την ενημέρωση της εγγραφής της επιλεγμένης παραγγελίας στον
                    // πίνακα παραγγελιών (Orders).

sqlds_order_status.UpdateParameters["StorageAddressID"].DefaultValue =
ddl_storage_address.SelectedValue;
                }
            }

            // Έλεγχος των ημερομηνιών και ωρών (DateTime) της συλλογής
προϊόντων, της τιμολόγησης παραγγελίας, της αποστολής παραγγελίας, της
παραλαβής παραγγελίας
            // και της ολοκλήρωσης παραγγελίας και σύγκριση με την κενή
ημερομηνία και ώρα. Στη συνέχεια την πιο πρόσφατη από αυτές σε μορφή ακολουθίας
χαρακτήρων
            // (string).
            string s_collect_date =
CodeClass.uf_s_DateTimeToString(dt_collect_date, dt_date_null);
            string s_pricing_date =
CodeClass.uf_s_DateTimeToString(dt_pricing_date, dt_date_null);
            string s_dispatch_date =
CodeClass.uf_s_DateTimeToString(dt_dispatch_date, dt_date_null);
            string s_delivery_date =
CodeClass.uf_s_DateTimeToString(dt_delivery_date, dt_date_null);
            string s_complete_date =
CodeClass.uf_s_DateTimeToString(dt_complete_date, dt_date_null);
```

```
            // Εισαγωγή στο σύστημα των ημερομηνιών συλλογής προϊόντων,
τιμολόγησης παραγγελίας, αποστολής παραγγελίας, παραλαβής παραγγελίας, ακύρωσης
παραγγελίας
            // και ολοκλήρωσης παραγγελίας, καθώς και του κωδικού καταχώρησης
διεύθυνσης αποθήκης για την ενημέρωση της εγγραφής της επιλεγμένης παραγγελίας
στον
            // πίνακα παραγγελιών (Orders).
            sqlds_order_status.UpdateParameters["CollectDate"].DefaultValue =
s_collect_date;
            sqlds_order_status.UpdateParameters["PricingDate"].DefaultValue =
s_pricing_date;
            sqlds_order_status.UpdateParameters["DispatchDate"].DefaultValue =
s_dispatch_date;
            sqlds_order_status.UpdateParameters["DeliveryDate"].DefaultValue =
s_delivery_date;
            sqlds_order_status.UpdateParameters["CancelDate"].DefaultValue =
dt_date_null.ToString();
            sqlds_order_status.UpdateParameters["CompleteDate"].DefaultValue =
s_complete_date;

sqlds_order_status.UpdateParameters["StorageAddressID"].DefaultValue =
ddl_storage_address.SelectedValue;

            // Εκτέλεση της εντολής sql για ενημέρωση της εγγραφής της
κατάστασης παραγγελίας στον πίνακα παραγγελίων (Orders).
            sqlds_order_status.Update();

            // Σύνδεση δεδομένων με τον πίνακα όψεως (GridView). Στην ουσία
ξαναφορτώνουμε τα δεδομένα, για να ενημερωθεί ο πίνακας όψεως (GridView)
            // με τις αλλαγές.
            gvw_orders.DataBind();
        }

        protected void btn_cancel_Click(object sender, EventArgs e)
        {
            // Εκτελούμε την εντολή sql για να συλλέξουμε τις ημερομηνίες και
ώρες (DateTime) υποβολής παραγγελίας και ακύρωσης παραγγελίας.
            sqlds_order_status_cancel.Select(DataSourceSelectArguments.Empty);

            // Αν η παραγγελία έχει ακυρωθεί, τότε δεν χρειάζεται να ακυρωθεί
ξανά.
            if (priv_b_cancel == true)
            {
                return;
            }

            // Η ώρα του συστήματος την στιγμή της ακύρωσης της παραγγελίας. Η
λήψη γίνεται από το ρολόι του διακομιστή (server) και όχι του χρήστη (client).
            DateTime dt_cancel_date = DateTime.Now;

            // Εισαγωγή στο σύστημα της ημερομηνίας ακύρωσης παραγγελίας για
την ενημέρωση της εγγραφής της επιλεγμένης παραγγελίας στον πίνακα παραγγελιών
(Orders).

sqlds_order_status_cancel.UpdateParameters["CancelDate"].DefaultValue =
dt_cancel_date.ToString();

            // Εκτέλεση της εντολής sql για ενημέρωση της εγγραφής της
κατάστασης παραγγελίας στον πίνακα παραγγελίων (Orders).
            sqlds_order_status_cancel.Update();
```

```
            // Σύνδεση δεδομένων με τον πίνακα όψεως (GridView). Στην ουσία
ξαναφορτώνουμε τα δεδομένα, για να ενημερωθεί ο πίνακας όψεως (GridView)
            // με τις αλλαγές.
            gvw_orders.DataBind();
        }

        protected void sqlds_order_status_cancel_Selected(object sender,
SqlDataSourceStatusEventArgs e)
        {
            // Λήψη των ημερομηνιών και ώρας (DateTime) της υποβολής
παραγγελίας και της ακύρωσης παραγγελίας από τον πίνακα παραγγελίων (Orders)
και μετατροπή
            // τους στον κατάλληλο τύπο δεδομένων.
            priv_dt_submit_date =
CodeClass.uf_dt_StringToDateTime(e.Command.Parameters["@SubmitDate"].Value.ToSt
ring());
            priv_dt_cancel_date =
CodeClass.uf_dt_StringToDateTime(e.Command.Parameters["@CancelDate"].Value.ToSt
ring());

            // Σύγκριση της ημερομηνίας και ώρας (DateTime) της ακύρωσης
παραγγελίας με την ημερομηνία και ώρα της υποβολής παραγγελίας και ενεργοποίηση
της
            // κατάστασης αληθείας (boolean). Η κατάσταση αυτή δηλώνει ότι το
συγκεκριμένο στάδιο είναι ολοκληρωμένο ή όχι.
            priv_b_cancel =
CodeClass.uf_b_DateTimeGreaterEqualCompare(priv_dt_cancel_date,
priv_dt_submit_date);
        }

        protected void lvw_order_products_DataBound(object sender, EventArgs e)
        {
            // Δημιουργία και ενεργοποίηση του συνδέσμου εκτύπωσης παραγγελίας
(OrderPrint). Ο σύνδεσμος είναι απεργοποιημένος όταν
            // δεν υπάρχει ο κωδικός εγγραφής της παραγγελίας και
ενεργοποιημένος όταν υπάρχει.
            CodeClass.uf_v_CreateOrderPrintHyperLink(gvw_orders,
hlnk_print_order);
        }
    }
}
```

Διαχείριση μονάδων τιμολόγησης

PricingUnitsEntry.aspx.cs

Ο ακόλουθος κώδικας επιτρέπει στον διαχειριστή να ορίζει τις μονάδες τιμολόγησης:

```
protected void Page_Load(object sender, EventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν φορτώνεται στον φυλλομετρητή
(browser) μια σελίδα καταχώρησης διαχειριστή.
            CodeClass.uf_v_AdministatorPageLoad(Page, gvw_pricing_units,
fvw_pricing_unit);
        }

        protected void fvw_pricing_unit_ItemDeleted(object sender,
FormViewDeletedEventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν διαγραφεί μια εγγραφή σε μια
πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση δεδομένων)
```

```
            // μέσω της φόρμας όψεως (FormView), σε μια φόρμα καταχώρησης
διαχειριστή.
            CodeClass.uf_v_AdministratorFormViewItemDeleted(gvw_pricing_units,
fvw_pricing_unit);
        }

        protected void fvw_pricing_unit_ItemInserted(object sender,
FormViewInsertedEventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν εισαχθεί μια εγγραφή σε μια
πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση δεδομένων)
            // μέσω της φόρμας όψεως (FormView), σε μια φόρμα καταχώρησης
διαχειριστή.
            CodeClass.uf_v_AdministratorFormViewItemInserted(gvw_pricing_units,
fvw_pricing_unit);
        }

        protected void fvw_pricing_unit_ItemUpdated(object sender,
FormViewUpdatedEventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν ενημερωθεί μια εγγραφή σε μια
πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση δεδομένων)
            // μέσω της φόρμας όψεως (FormView), σε μια φόρμα καταχώρησης
διαχειριστή.
            CodeClass.uf_v_AdministratorFormViewItemUpdated(gvw_pricing_units,
fvw_pricing_unit);
        }

        protected void gvw_pricing_units_RowDataBound(object sender,
GridViewRowEventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν συνδεθούν τα δεδομένα μιας
εγγραφής από μια πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση
δεδομένων)
            // στον πίνακα όψεως (GridView), σε μια φόρμα καταχώρησης
διαχειριστή.
            CodeClass.uf_v_AdministratorGridViewRowDataBound(gvw_pricing_units,
fvw_pricing_unit);
        }

        protected void sqlds_pricing_unit_Deleted(object sender,
SqlDataSourceStatusEventArgs e)
        {
            // Έλεγχος παραμέτρων πηγής δεδομένων (Data Source) για σφάλματα σε
μια σελίδα καταχώρησης διαχειριστή.
            CodeClass.uf_v_AdministatorSqlDataSourceDeleted(e, lbl_message);
        }
```

Διαχείριση κατηγορίων προϊόντων

ProductCategoriesEntry.aspx.cs

Ο ακόλουθος κώδικας επιτρέπει στον διαχειριστή να καθορίζει τις κατηγορίες των προϊόντων:

```
protected void Page_Load(object sender, EventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν φορτώνεται στον φυλλομετρητή
(browser) μια σελίδα καταχώρησης διαχειριστή.
            CodeClass.uf_v_AdministatorPageLoad(Page, gvw_product_categories,
fvw_product_category);
        }
```

```
        protected void fvw_product_category_ItemDeleted(object sender,
FormViewDeletedEventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν διαγραφεί μια εγγραφή σε μια
πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση δεδομένων)
            // μέσω της φόρμας όψεως (FormView), σε μια φόρμα καταχώρησης
διαχειριστή.

CodeClass.uf_v_AdministratorFormViewItemDeleted(gvw_product_categories,
fvw_product_category);
        }

        protected void fvw_product_category_ItemInserted(object sender,
FormViewInsertedEventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν εισαχθεί μια εγγραφή σε μια
πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση δεδομένων)
            // μέσω της φόρμας όψεως (FormView), σε μια φόρμα καταχώρησης
διαχειριστή.

CodeClass.uf_v_AdministratorFormViewItemInserted(gvw_product_categories,
fvw_product_category);
        }

        protected void fvw_product_category_ItemUpdated(object sender,
FormViewUpdatedEventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν ενημερωθεί μια εγγραφή σε μια
πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση δεδομένων)
            // μέσω της φόρμας όψεως (FormView), σε μια φόρμα καταχώρησης
διαχειριστή.

CodeClass.uf_v_AdministratorFormViewItemUpdated(gvw_product_categories,
fvw_product_category);
        }

        protected void gvw_product_categories_RowDataBound(object sender,
GridViewRowEventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν συνδεθούν τα δεδομένα μιας
εγγραφής από μια πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση
δεδομένων)
            // στον πίνακα όψεως (GridView), σε μια φόρμα καταχώρησης
διαχειριστή.

CodeClass.uf_v_AdministratorGridViewRowDataBound(gvw_product_categories,
fvw_product_category);
        }

        protected void sqlds_product_category_Deleted(object sender,
SqlDataSourceStatusEventArgs e)
        {
            // Έλεγχος παραμέτρων πηγής δεδομένων (Data Source) για σφάλματα σε
μια σελίδα καταχώρησης διαχειριστή.
            CodeClass.uf_v_AdministatorSqlDataSourceDeleted(e, lbl_message);
        }
```

Διαχείριση προϊόντων

ProductsEntry.aspx.cs

Ο ακόλουθος κώδικας επιτρέπει στον διαχειριστή να καταχωρεί νέα προϊόντα ή να διαχειρίζεται τα υπάρχοντα:

```
public partial class ProductsEntry : System.Web.UI.Page
    {
        private int priv_i_pricing_units_count; // Μεταβλητή ελέγχου παρουσίας εγγραφών μονάδων τιμολόγησης.
        private int priv_i_product_categories_count; // Μεταβλητή ελέγχου παρουσίας εγγραφών κατηγοριών προϊόντων.

        protected void Page_Load(object sender, EventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν φορτώνεται στον φυλλομετρητή (browser) μια σελίδα καταχώρησης διαχειριστή.
            CodeClass.uf_v_AdministatorPageLoad(Page, gvw_products, fvw_product);

            // Έλεγχος παρουσίας εγγραφών στους πίνακες μονάδων τιμολόγησης (Pricing Units) και κατηγοριών προϊόντων (Product Categories). Οι πίνακες αυτοί
            // τροφοδοτούν τις αναδιπλούμενες λίστες (DropDownList) των φορμών όψεως (FormView) και είναι απαραίτητες για την καταχώρηση δεδομένων στον πίνακα
            // προϊόντων (Products).
            sqlds_pricing_units_count.Select(DataSourceSelectArguments.Empty);

sqlds_product_categories_count.Select(DataSourceSelectArguments.Empty);

            // Αν δεν υπάρχει κάποια εγγραφή στους πίνακες μονάδων τιμολόγησης (Pricing Units) και κατηγοριών προϊόντων (Product Categories), τότε απενεργοποιούμε
            // τον πίνακα όψεως (GridView) και την φόρμα όψεως (FormView) και ενημέρωνουμε τον διαχειριστή για να καταχωρήσει τα απαραίτητα δεδομένα που απαιτούνται
            // για την λειτουργία αυτής της σελίδας καταχώρησης.
            if ((priv_i_pricing_units_count < 1) || (priv_i_product_categories_count < 1))
            {
                // Απενεργοπόίηση του πίνακα όψεως (GridView) και της φόρμα όψεως (FormView).
                gvw_products.Enabled = false;
                fvw_product.Enabled = false;

                // Δημιουργία του μηνύματος που θα εμφανίσουμε στον διαχειριστή.
                string s_message = "Ο πίνακας όψεως και η φόρμα καταχώρησης απενεργοποιήθηκαν μέχρι να εισαχθεί μια εγγραφή";

                // Στο μήνυμα πρέπει να ενημερώσουμε τον διαχειριστή για την έλλειψη δεδομένων στον πίνακα μονάδων τιμολόγησης (Pricing Units) για να εισάγει μια
                // εγγραφή εκεί.
                if (priv_i_pricing_units_count < 1)
                {
                    s_message = s_message + " στις μονάδες τιμολόγησης";

                    // Διαχωρίζουμε τους πίνακες όταν δεν υπάρχει εγγραφή σε κάποιον από τους υπόλοιπους πέραν του παρόντος.
                    if (priv_i_product_categories_count < 1)
                    {
                        s_message = s_message + " /";
                    }
                }
```

```
                // Στο μήνυμα πρέπει να ενημερώσουμε τον διαχειριστή για την
έλλειψη δεδομένων στον πίνακα κατηγοριών προϊόντων (Products Categories) για να
εισάγει
                // μια εγγραφή εκεί.
                if (priv_i_product_categories_count < 1)
                {
                    s_message = s_message + " στις κατηγορίες προϊόντων";
                }

                // Εμφάνιση του μηνύματος στον διαχειριστή.
                lbl_message.Text = s_message + ".";
            }
        }

        protected void fvw_product_ItemDeleted(object sender,
FormViewDeletedEventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν διαγραφεί μια εγγραφή σε μια
πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση δεδομένων)
            // μέσω της φόρμας όψεως (FormView), σε μια φόρμα καταχώρησης
διαχειριστή.
            CodeClass.uf_v_AdministratorFormViewItemDeleted(gvw_products,
fvw_product);
        }

        protected void fvw_product_ItemInserted(object sender,
FormViewInsertedEventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν εισαχθεί μια εγγραφή σε μια
πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση δεδομένων)
            // μέσω της φόρμας όψεως (FormView), σε μια φόρμα καταχώρησης
διαχειριστή.
            CodeClass.uf_v_AdministratorFormViewItemInserted(gvw_products,
fvw_product);
        }

        protected void fvw_product_ItemUpdated(object sender,
FormViewUpdatedEventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν ενημερωθεί μια εγγραφή σε μια
πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση δεδομένων)
            // μέσω της φόρμας όψεως (FormView), σε μια φόρμα καταχώρησης
διαχειριστή.
            CodeClass.uf_v_AdministratorFormViewItemUpdated(gvw_products,
fvw_product);
        }

        protected void gvw_products_RowDataBound(object sender,
GridViewRowEventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν συνδεθούν τα δεδομένα μιας
εγγραφής από μια πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση
δεδομένων)
            // στον πίνακα όψεως (GridView), σε μια φόρμα καταχώρησης
διαχειριστή.
            CodeClass.uf_v_AdministratorGridViewRowDataBound(gvw_products,
fvw_product);
        }

        protected void sqlds_product_Deleted(object sender,
SqlDataSourceStatusEventArgs e)
        {
```

```
            // Έλεγχος παραμέτρων πηγής δεδομένων (Data Source) για σφάλματα σε
μια σελίδα καταχώρησης διαχειριστή.
            CodeClass.uf_v_AdministatorSqlDataSourceDeleted(e, lbl_message);
        }

        protected void sqlds_pricing_units_count_Selected(object sender,
SqlDataSourceStatusEventArgs e)
        {
            // Ελέγχουμε αν υπάρχουν εγγραφές μέσα στον πίνακα μονάδων
τιμολόγησης (PricingUnits).
            priv_i_pricing_units_count =
(int)e.Command.Parameters["@IDCount"].Value;
        }

        protected void sqlds_product_categories_count_Selected(object sender,
SqlDataSourceStatusEventArgs e)
        {
            // Ελέγχουμε αν υπάρχουν εγγραφές μέσα στον πίνακα κατηγορίων
προϊόντων (ProductCategories).
            priv_i_product_categories_count =
(int)e.Command.Parameters["@IDCount"].Value;
        }
    }
```

Διαχείριση αποθεμάτων

StocksEntry.aspx.cs

Ο ακόλουθος κώδικας επιτρέπει στον διαχειριστή να διαχειρίζεται τα αποθέματα των καταχωρημένων προϊόντων:

```
public partial class StocksEntry : System.Web.UI.Page
    {
        private int priv_i_products_count; // Μεταβλητή ελέγχου παρουσίας
εγγραφών προϊόντων.
        private int priv_i_discounts_count; // Μεταβλητή ελέγχου παρουσίας
εγγραφών εκπτώσεων.
        private int priv_i_storages_count; // Μεταβλητή ελέγχου παρουσίας
εγγραφών αποθηκών.

        protected void Page_Load(object sender, EventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν φορτώνεται στον φυλλομετρητή
(browser) μια σελίδα καταχώρησης διαχειριστή.
            CodeClass.uf_v_AdministatorPageLoad(Page, gvw_stocks, fvw_stock);

            // Έλεγχος παρουσίας εγγραφών στους πίνακες προϊόντων (Products),
εκπτώσεων (Discounts) και αποθηκών (Storages). Οι πίνακες αυτοί τροφοδοτούν τις
            // αναδιπλούμενες λίστες (DropDownList) των φορμών όψεως (FormView)
και είναι απαραίτητες για την καταχώρηση δεδομένων στον πίνακα αποθεμάτων
(Stocks).
            sqlds_products_count.Select(DataSourceSelectArguments.Empty);
            sqlds_discounts_count.Select(DataSourceSelectArguments.Empty);
            sqlds_storages_count.Select(DataSourceSelectArguments.Empty);

            // Αν δεν υπάρχει κάποια εγγραφή στους πίνακες προϊόντων
(Products), εκπτώσεων (Discounts) και αποθηκών (Storages), τότε απενεργοποιούμε
τον πίνακα
            // όψεως (GridView) και την φόρμα όψεως (FormView) και ενημέρωνουμε
τον διαχειριστή για να καταχωρήσει τα απαραίτητα δεδομένα που απαιτούνται για
την
            // λειτουργία αυτής της σελίδας καταχώρησης.
```

```
            if ((priv_i_products_count < 1) || (priv_i_discounts_count < 1) ||
(priv_i_storages_count < 1))
            {
                // Απενεργοποίηση του πίνακα όψεως (GridView) και της φόρμα
όψεως (FormView).
                gvw_stocks.Enabled = false;
                fvw_stock.Enabled = false;

                // Δημιουργία του μηνύματος που θα εμφανίσουμε στον
διαχειριστή.
                string s_message = "Ο πίνακας όψεως και η φόρμα καταχώρησης
απενεργοποιήθηκαν μέχρι να εισαχθεί μια εγγραφή";

                // Στο μήνυμα πρέπει να ενημερώσουμε τον διαχειριστή για την
έλλειψη δεδομένων στον πίνακα προϊόντων (Products) για να εισάγει μια εγγραφή
εκεί.
                if (priv_i_products_count < 1)
                {
                    s_message = s_message + " στα προϊόντα";

                    // Διαχωρίζουμε τους πίνακες όταν δεν υπάρχει εγγραφή σε
κάποιον από τους υπόλοιπους πέραν του παρόντος.
                    if ((priv_i_discounts_count < 1) || (priv_i_storages_count
< 1))
                    {
                        s_message = s_message + " /";
                    }
                }

                // Στο μήνυμα πρέπει να ενημερώσουμε τον διαχειριστή για την
έλλειψη δεδομένων στον πίνακα εκπτώσεων (Discounts) για να εισάγει μια εγγραφή
εκεί.
                if (priv_i_discounts_count < 1)
                {
                    s_message = s_message + " στις εκπτώσεις";

                    // Διαχωρίζουμε τους πίνακες όταν δεν υπάρχει εγγραφή σε
κάποιον από τους υπόλοιπους πέραν του παρόντος.
                    if (priv_i_storages_count < 1)
                    {
                        s_message = s_message + " /";
                    }
                }

                // Στο μήνυμα πρέπει να ενημερώσουμε τον διαχειριστή για την
έλλειψη δεδομένων στον πίνακα αποθηκών (Storages) για να εισάγει μια εγγραφή
εκεί.
                if (priv_i_products_count < 1)
                {
                    s_message = s_message + " στις αποθήκες";
                }

                // Εμφάνιση του μηνύματος στον διαχειριστή.
                lbl_message.Text = s_message + ".";
            }
        }

        protected void fvw_stock_ItemDeleted(object sender,
FormViewDeletedEventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν διαγραφεί μια εγγραφή σε μια
πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση δεδομένων)
```

```
            // μέσω της φόρμας όψεως (FormView), σε μια φόρμα καταχώρησης
διαχειριστή.
            CodeClass.uf_v_AdministratorFormViewItemDeleted(gvw_stocks,
fvw_stock);
        }

        protected void fvw_stock_ItemInserted(object sender,
FormViewInsertedEventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν εισαχθεί μια εγγραφή σε μια
πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση δεδομένων)
            // μέσω της φόρμας όψεως (FormView), σε μια φόρμα καταχώρησης
διαχειριστή.
            CodeClass.uf_v_AdministratorFormViewItemInserted(gvw_stocks,
fvw_stock);
        }

        protected void fvw_stock_ItemUpdated(object sender,
FormViewUpdatedEventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν ενημερωθεί μια εγγραφή σε μια
πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση δεδομένων)
            // μέσω της φόρμας όψεως (FormView), σε μια φόρμα καταχώρησης
διαχειριστή.
            CodeClass.uf_v_AdministratorFormViewItemUpdated(gvw_stocks,
fvw_stock);
        }

        protected void gvw_stocks_RowDataBound(object sender,
GridViewRowEventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν συνδεθούν τα δεδομένα μιας
εγγραφής από μια πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση
δεδομένων)
            // στον πίνακα όψεως (GridView), σε μια φόρμα καταχώρησης
διαχειριστή.
            CodeClass.uf_v_AdministratorGridViewRowDataBound(gvw_stocks,
fvw_stock);
        }

        protected void sqlds_stock_Deleted(object sender,
SqlDataSourceStatusEventArgs e)
        {
            // Έλεγχος παραμέτρων πηγής δεδομένων (Data Source) για σφάλματα σε
μια σελίδα καταχώρησης διαχειριστή.
            CodeClass.uf_v_AdministatorSqlDataSourceDeleted(e, lbl_message);
        }

        protected void sqlds_products_count_Selected(object sender,
SqlDataSourceStatusEventArgs e)
        {
            // Ελέγχουμε αν υπάρχουν εγγραφές μέσα στον πίνακα προϊόντων
(Products).
            priv_i_products_count =
(int)e.Command.Parameters["@IDCount"].Value;
        }

        protected void sqlds_discounts_count_Selected(object sender,
SqlDataSourceStatusEventArgs e)
        {
            // Ελέγχουμε αν υπάρχουν εγγραφές μέσα στον πίνακα εκπτώσεων
(Discounts).
```

```
            priv_i_discounts_count =
(int)e.Command.Parameters["@IDCount"].Value;
        }

        protected void sqlds_storages_count_Selected(object sender,
SqlDataSourceStatusEventArgs e)
        {
            // Ελέγχουμε αν υπάρχουν εγγραφές μέσα στον πίνακα αποθηκών
(Storages).
            priv_i_storages_count =
(int)e.Command.Parameters["@IDCount"].Value;
        }
    }
```

Διαχείριση διευθύνσεων αποθηκών

StoragesAddressEntry.aspx.cs

Ο ακόλουθος κώδικας επιτρέπει στον διαχειριστή να ορίζει τις διευθύνσεις των αποθηκών του εργοστασίου:

```
protected void Page_Load(object sender, EventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν φορτώνεται στον φυλλομετρητή
(browser) μια σελίδα καταχώρησης διαχειριστή.
            CodeClass.uf_v_AdministatorPageLoad(Page, gvw_storages_addresses,
fvw_storage_address);
        }

        protected void fvw_storage_address_ItemDeleted(object sender,
FormViewDeletedEventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν διαγραφεί μια εγγραφή σε μια
πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση δεδομένων)
            // μέσω της φόρμας όψεως (FormView), σε μια φόρμα καταχώρησης
διαχειριστή.

CodeClass.uf_v_AdministratorFormViewItemDeleted(gvw_storages_addresses,
fvw_storage_address);
        }

        protected void fvw_storage_address_ItemInserted(object sender,
FormViewInsertedEventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν εισαχθεί μια εγγραφή σε μια
πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση δεδομένων)
            // μέσω της φόρμας όψεως (FormView), σε μια φόρμα καταχώρησης
διαχειριστή.

CodeClass.uf_v_AdministratorFormViewItemInserted(gvw_storages_addresses,
fvw_storage_address);
        }

        protected void fvw_storage_address_ItemUpdated(object sender,
FormViewUpdatedEventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν ενημερωθεί μια εγγραφή σε μια
πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση δεδομένων)
            // μέσω της φόρμας όψεως (FormView), σε μια φόρμα καταχώρησης
διαχειριστή.
```

```
CodeClass.uf_v_AdministratorFormViewItemUpdated(gvw_storages_addresses,
fvw_storage_address);
        }

        protected void gvw_storages_addresses_RowDataBound(object sender,
GridViewRowEventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν συνδεθούν τα δεδομένα μιας
εγγραφής από μια πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση
δεδομένων)
            // στον πίνακα όψεως (GridView), σε μια φόρμα καταχώρησης
διαχειριστή.

CodeClass.uf_v_AdministratorGridViewRowDataBound(gvw_storages_addresses,
fvw_storage_address);
        }

        protected void sqlds_storage_address_Deleted(object sender,
SqlDataSourceStatusEventArgs e)
        {
            // Έλεγχος παραμέτρων πηγής δεδομένων (Data Source) για σφάλματα σε
μια σελίδα καταχώρησης διαχειριστή.
            CodeClass.uf_v_AdministatorSqlDataSourceDeleted(e, lbl_message);
        }
```

Διαχείριση αποθηκών

StoragesEntry.cs

Ο ακόλουθος κώδικας επιτρέπει στο διαχειριστή να ορίζει τις διαθέσιμες αποθήκες:

```
public partial class StoragesEntry : System.Web.UI.Page
    {
        private int priv_i_storage_address_count;  // Μεταβλητή ελέγχου
παρουσίας εγγραφών διευθύνσεων αποθηκών.

        protected void Page_Load(object sender, EventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν φορτώνεται στον φυλλομετρητή
(browser) μια σελίδα καταχώρησης διαχειριστή.
            CodeClass.uf_v_AdministatorPageLoad(Page, gvw_storages,
fvw_storage);

            // Έλεγχος παρουσίας εγγραφών στον πίνακα διευθύνσεων αποθηκών
(Storages Addresses). Ο πίνακας αυτός τροφοδοτεί την αναδιπλούμενη λίστα
(DropDownList)
            // της φόρμας όψεως (FormView) και είναι απαραίτητη για την
καταχώρηση δεδομένων στον πίνακα αποθηκών (Storages).

sqlds_storage_address_count.Select(DataSourceSelectArguments.Empty);

            // Αν δεν υπάρχει κάποια εγγραφή στον πίνακα διευθύνσεων αποθηκών
(Storages Addresses), τότε απενεργοποιούμε τον πίνακα όψεως (GridView) και την
φόρμα
            // όψεως (FormView) και ενημέρωνουμε τον διαχειριστή για να
καταχωρήσει τα απαραίτητα δεδομένα που απαιτούνται για την λειτουργία αυτής της
σελίδας
            // καταχώρησης.
            if (priv_i_storage_address_count < 1)
            {
```

```
                // Απενεργοποίηση του πίνακα όψεως (GridView) και της φόρμα
όψεως (FormView).
                gvw_storages.Enabled = false;
                fvw_storage.Enabled = false;

                // Εμφάνιση του μηνύματος στον διαχειριστή.
                lbl_message.Text = "Ο πίνακας όψεως και η φόρμα καταχώρησης
απενεργοποιήθηκαν μέχρι να εισαχθεί μια εγγραφή στις διευθύνσεις αποθηκών.";
            }
        }

        protected void fvw_storage_ItemDeleted(object sender,
FormViewDeletedEventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν διαγραφεί μια εγγραφή σε μια
πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση δεδομένων)
            // μέσω της φόρμας όψεως (FormView), σε μια φόρμα καταχώρησης
διαχειριστή.
            CodeClass.uf_v_AdministratorFormViewItemDeleted(gvw_storages,
fvw_storage);
        }

        protected void fvw_storage_ItemInserted(object sender,
FormViewInsertedEventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν εισαχθεί μια εγγραφή σε μια
πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση δεδομένων)
            // μέσω της φόρμας όψεως (FormView), σε μια φόρμα καταχώρησης
διαχειριστή.
            CodeClass.uf_v_AdministratorFormViewItemInserted(gvw_storages,
fvw_storage);
        }

        protected void fvw_storage_ItemUpdated(object sender,
FormViewUpdatedEventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν ενημερωθεί μια εγγραφή σε μια
πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση δεδομένων)
            // μέσω της φόρμας όψεως (FormView), σε μια φόρμα καταχώρησης
διαχειριστή.
            CodeClass.uf_v_AdministratorFormViewItemUpdated(gvw_storages,
fvw_storage);
        }

        protected void gvw_storages_RowDataBound(object sender,
GridViewRowEventArgs e)
        {
            // Διαδικασία που ακολουθείται όταν συνδεθούν τα δεδομένα μιας
εγγραφής από μια πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση
δεδομένων)
            // στον πίνακα όψεως (GridView), σε μια φόρμα καταχώρησης
διαχειριστή.
            CodeClass.uf_v_AdministratorGridViewRowDataBound(gvw_storages,
fvw_storage);
        }

        protected void sqlds_storage_Deleted(object sender,
SqlDataSourceStatusEventArgs e)
        {
            // Έλεγχος παραμέτρων πηγής δεδομένων (Data Source) για σφάλματα σε
μια σελίδα καταχώρησης διαχειριστή.
            CodeClass.uf_v_AdministatorSqlDataSourceDeleted(e, lbl_message);
```

```
        }

        protected void sqlds_storage_address_count_Selected(object sender,
SqlDataSourceStatusEventArgs e)
        {
            // Ελέγχουμε αν υπάρχουν εγγραφές μέσα στον πίνακα διευθύνσεων
αποθηκών (Storages Addresses).
            priv_i_storage_address_count =
(int)e.Command.Parameters["@IDCount"].Value;
        }
    }
```

Διαχείριση χρηστών

Πληροφορίες χρήστη

Information.aspx.cs

Ο ακόλουθος κώδικας επιτρέπει στον χρήστη να διαχειρίζεται τα προσωπικά του στοιχεία. Τα στοιχεία αυτά χρησιμοποιούνται για την αποστολή των προϊόντων στο χώρο του:

```
public partial class Information : System.Web.UI.Page
    {
        private int priv_i_customer_address_affected_rows; // Μεταβλητή ελέγχου
παρουσίας της εγγραφής των προσωπικών στοιχείων του χρήστη.

        protected void Page_Load(object sender, EventArgs e)
        {
            // Ελέγχουμε αν υπάρχει η εγγραφή στον πίνακα διευθύνσεων πελατών
(CustomersAddreesses).
            sqlds_customer_address.Select(DataSourceSelectArguments.Empty);

            // Αν δεν υπάρχει (δεν καταχωρήθηκε για κάποιον λόγο κατά την
εγγραφή ή ο χρήστης είναι καταχωρημένος πριν την δυνατότητα εισαγωγής των
προσωπικών του
            // στοιχείων), τότε δημιουργούμε μια εγγραφή στον πίνακα
διευθύνσεων πελατών (CustomersAddreesses).
            if (priv_i_customer_address_affected_rows < 1)
            {
                sqlds_customer_address.Insert();
            }
        }

        protected void sqlds_customer_address_Inserting(object sender,
SqlDataSourceCommandEventArgs e)
        {
            // Συλλογή των στοιχείων λογαριασμού του χρήστη που είναι
συνδεδεμένος στο σύστημα.
            MembershipUser msu_user =
Membership.GetUser(HttpContext.Current.User.Identity.Name);

            // Αν η συλλογή των στοιχείων του είναι επιτυχής, τότε θα εισάγουμε
στον πίνακα διευθύνσεων πελατών (CustomersAddreesses) τα προσωπικά του
στοιχεία.
            if (msu_user != null)
            {
                Guid guid_customer = (Guid)msu_user.ProviderUserKey; //
Μοναδικό προσδιοριστικό (Unique ID) συνδεδεμένου χρήστη.
```

```
                // Εισαγωγή στο σύστημα του μοναδικού προσδιοριστικού (Unique
ID) για την εισαγωγή της εγγραφής των προσωπικών στοιχείων του χρήστη
                // από τον πίνακα διευθύνσεων πελατών (CustomersAddreesses).
                e.Command.Parameters["@CustomerUserID"].Value = guid_customer;
            }
        }

        protected void sqlds_customer_address_Selecting(object sender,
SqlDataSourceSelectingEventArgs e)
        {
            // Συλλογή των στοιχείων λογαριασμού του χρήστη που είναι
συνδεδεμένος στο σύστημα.
            MembershipUser msu_user =
Membership.GetUser(HttpContext.Current.User.Identity.Name);

            // Αν η συλλογή των στοιχείων του είναι επιτυχής, τότε θα
συλλέξουμε από τον πίνακα διευθύνσεων πελατών (CustomersAddreesses) τα
προσωπικά του στοιχεία.
            if (msu_user != null)
            {
                Guid guid_customer = (Guid)msu_user.ProviderUserKey; //
Μοναδικό προσδιοριστικό (Unique ID) συνδεδεμένου χρήστη.

                // Εισαγωγή στο σύστημα του μοναδικού προσδιοριστικού (Unique
ID) για την επιλογή της εγγραφής των προσωπικών στοιχείων του χρήστη
                // από τον πίνακα διευθύνσεων πελατών (CustomersAddreesses).
                e.Command.Parameters["@CustomerUserID"].Value = guid_customer;
            }
        }

        protected void sqlds_customer_address_Selected(object sender,
SqlDataSourceStatusEventArgs e)
        {
            // Ελέγχουμε αν ο χρήστης έχει εγγραφή μέσα στον πίνακα διευθύνσεων
πελατών (CustomersAddreesses)
            priv_i_customer_address_affected_rows = e.AffectedRows;
        }
    }
```

Εκτύπωση παραγγελιών

OrderPrint.aspx.cs

Ο ακόλουθος κώδικας επιτρέπει στον χρήστη να εκτυπώσει μια παραγγελία μέσω του εκτυπωτικού Order Print (Είναι το ίδιο με αυτό του διαχειριστή, μόνο που ο διαχειριστής μπορεί να εκτυπώσει την παραγγελία οποιουδήποτε πελάτη):

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

using System.Web.Security;
using FactorySales.Code;

namespace FactorySales.Member
{
    public partial class OrderPrint : System.Web.UI.Page
    {
```

```
        private Guid priv_guid_customer_user_id; // Μεταβλητή μοναδικού προσδιοριστικού (Unique ID) του χρήστη κατόχου της επιλεγμένης παραγγελίας προς εκτύπωση.
        private decimal priv_d_total_price; // Μεταβλητή συνολικού αθροίσματος (συνολική τιμή) της αξίας των προϊόντων της επιλεγμένης παραγγελίας προς εκτύπωση.
        private DateTime priv_dt_dispatch_date; // Μεταβλητή ημερομηνίας αποστολής της επιλεγμένης παραγγελίας προς εκτύπωση.
        private DateTime priv_dt_cancel_date; // Μεταβλητή ημερομηνίας ακύρωσης της επιλεγμένης παραγγελίας προς εκτύπωση.

        protected void Page_Load(object sender, EventArgs e)
        {
            // Συλλογή των στοιχείων λογαριασμού του χρήστη που είναι συνδεδεμένος στο σύστημα.
            MembershipUser msu_user = Membership.GetUser(HttpContext.Current.User.Identity.Name);

            // Λήψη από τις μεταβλητές φυλλομετρητή (Query String: μεταβλητές που αποτελούν μέρος του συνδέσμου και περιέχουν δεδομένα) του κωδικού εγγραφής της
            // παραγγελίας που θέλουμε να εκτυπώσουμε.
            int i_id = CodeClass.uf_i_QueryStringToInteger(Request, "id");

            // Αν δεν κατορθώσουμε να πάρουμε τον κωδικό εγγραφής της παραγγελίας προς εκτύπωση, τότε τερματίζουμε την διαδικασία καθώς δεν έχουμε την βασική
            // πληροφορία που χρειαζόμαστε για να δημιουργήσουμε το εκτυπωτικό.
            if (i_id < 1)
            {
                return;
            }

            // Εκτελούμε την εντολή sql για να συλλέξουμε το μοναδικό προσδιοριστικό (Unique ID) του χρήστη κατόχου της παραγγελίας, της ημερομηνίας αποστολής της
            // και της ημερομηνίας ακύρωσης της.
            sqlds_order.Select(DataSourceSelectArguments.Empty);

            // Ελέγχος δικαιωμάτων εκτύπωσης παραγγελίας από τον συνδεδεμένο χρήστη.
            bool b_allow = uf_b_AllowOrderPrint(msu_user);

            // Αν ο χρήστης δεν έχει δικαιώματα εκτύπωσης, τότε τερματίζουμε την λειτουργία, καθώς δεν υπάρχει λόγος να συνεχιστεί.
            if (b_allow == false)
            {
                return;
            }

            // Εκτελούμε την εντολή sql για να συλλέξουμε την συνολική τιμή των προϊόντων της παραγγελίας.
            sqlds_order_total.Select(DataSourceSelectArguments.Empty);

            // Έλεγχος αν μέσα στην λίστα όψεως (ListView) της επιλεγμένης παραγγελίας, υπάρχει η ετικέτα (Label) της συνολικής τιμής των προϊόντων της παραγγελίας.
            if ((Label)lvw_order.FindControl("lbl_total_price") is Label)
            {
                // Λήψη της ετικέτας (Label) της συνολικής τιμής των προϊόντων της παραγγελίας για να ορίσουμε την τιμή της (ακολουθείται αυτή η διαδικασία καθώς
```

```
                // δεν είναι εφικτή η απευθείας ανάθεση τιμής, καθώς
εμπεριέχεται στο σύνολο των αντικειμένων (child objects) της φόρμας ενός άλλου
αντικειμένου
                // (parent object)).
                Label lbl_total_price =
(Label)lvw_order.FindControl("lbl_total_price");

                // Εμφανίζουμε την συνολική τιμή των προϊόντων της παραγγελίας
στην ανάλογη ετικέτα (Label). Η μορφοποίηση (format) "0.00" εξασφαλίζει
                // ότι το αποτέλεσμα θα είναι δεκαδικός με δύο δεκαδικά ψηφία.
                lbl_total_price.Text = priv_d_total_price.ToString("0.00");
            }

            // Στην πραγματικότητα η ακόλουθη τιμή δεν είναι null (κενή), αλλά
ίση με την ημερομηνία 1/1/2001 12:00:00 πμ.
            // Η ίδια τιμή υπάρχει και στις στήλες του αντίστοιχου πίνακα στη
βάση δεδομένων, όπου την χρησιμοποιούμε ως "κενή".
            DateTime dt_date_null = new DateTime(2001, 1, 1, 0, 0, 0);

            // Ορίζουμε εξ ορισμού την κατάσταση της παραγγελίας σε αναμονή.
            string s_dispatch = "Σε αναμονή.";

            // Ελέγχος αν η παραγγελία έχει ακυρωθεί.
            if (priv_dt_cancel_date > dt_date_null)
            {
                s_dispatch = "Η παραγγελία ακυρώθηκε.";
            }
            else if (priv_dt_dispatch_date > dt_date_null) // Αν δεν έχει
ακυρωθεί, τότε ελέγχουμε αν έχει αποσταλεί στον χρήστη.
            {
                s_dispatch = priv_dt_dispatch_date.ToString();
            }

            // Εμφανίζουμε την ημερομηνία αποστολής στην ανάλογη ετικέτα
(Label).
            lbl_dispatch_date.Text = "Ημερομηνία αποστολής: " + s_dispatch;

            // Συλλογή όλων των μορφοποιημένων δεδομένων που σχηματίζουν την
παρούσα παραγγελία και εμφάνιση τους σε μια φόρμα εκτύπωσης.
            // Η φόρμα αυτή δημιουργείται με τη χρήση κώδικα Javascript.
            ClientScript.RegisterClientScriptBlock(this.GetType(), "key",
"window.print()", true);
        }

        protected void sqlds_order_Selected(object sender,
SqlDataSourceStatusEventArgs e)
        {
            // Λήψη του μοναδικού προσδιοριστικού (Unique ID) του χρήστη
κατόχου της παραγγελίας, της ημερομηνίας αποστολής της και της ημερομηνίας
ακύρωσης της
            // (Ουσιαστικά η ημερομηνιά ακύρωσης δηλώνει ότι η παραγγελία
ακυρώθηκε για κάποιον λόγο και υπάρχει για ακυρωμένες παραγγελίες μόνο).
            priv_guid_customer_user_id =
(Guid)e.Command.Parameters["@CustomerUserID"].Value;
            priv_dt_dispatch_date =
(DateTime)e.Command.Parameters["@DispatchDate"].Value;
            priv_dt_cancel_date =
(DateTime)e.Command.Parameters["@CancelDate"].Value;
        }

        protected void sqlds_order_total_Selected(object sender,
SqlDataSourceStatusEventArgs e)
```

```
        {
            // Λήψη του συνολικού αθροίσματος (συνολική τιμή) της αξίας των
προϊόντων της παραγγελίας.
            priv_d_total_price =
Convert.ToDecimal(e.Command.Parameters["@TotalPrice"].Value);
        }

        protected void sqlds_customer_address_Selecting(object sender,
SqlDataSourceSelectingEventArgs e)
        {
            // Συλλογή των στοιχείων λογαριασμού του χρήστη που είναι
συνδεδεμένος στο σύστημα.
            MembershipUser msu_user =
Membership.GetUser(HttpContext.Current.User.Identity.Name);

            // Αν η συλλογή των στοιχείων του είναι επιτυχής, τότε θα
συλλέξουμε το μοναδικό προσδιοριστικό (Unique ID) του συνδεδεμένου χρήστη.
            if (msu_user != null)
            {
                Guid guid_user_id = (Guid)msu_user.ProviderUserKey; // Μοναδικό
προσδιοριστικό (Unique ID) συνδεδεμένου χρήστη.

                // Ελέγχος δικαιωμάτων εκτύπωσης παραγγελίας από τον
συνδεδεμένο χρήστη.
                bool b_allow = uf_b_AllowOrderPrint(msu_user);

                // Αν ο χρήστης έχει δικαιώματα εκτύπωσης της συγκεκριμένης
παραγγελίας, τότε όρισε τον κάτοχο της παρούσας παραγγελίας.
                if (b_allow == true)
                {
                    // Εισαγωγή στο σύστημα του μοναδικού προσδιοριστικού
(Unique ID) για την επιλογή της εγγραφής της παραγγελίας από τον πίνακα
                    // παραγγελιών (Orders).
                    if (HttpContext.Current.User.IsInRole("administrator") ==
false)
                    {
                        // Εισαγωγή στο σύστημα του μοναδικού προσδιοριστικού
(Unique ID) για την επιλογή της εγγραφής διεύθυνσης του χρήστη
                        // από τον πίνακα παραγγελιών (Orders).
                        e.Command.Parameters["@CustomerUserID"].Value =
guid_user_id;
                    }
                    else
                    {
                        // Εισαγωγή στο σύστημα του μοναδικού προσδιοριστικού
(Unique ID) για την επιλογή της εγγραφής διεύθυνσης του χρήστη που
πραγματοποίησε
                        // την παραγγελία. Στην περίπτωση αυτή, ο διαχειριστής
ζητάει από το σύστημα την εκτύπωση μιας παραγγελίας που αναλογεί σε κάποιον
χρήστη.
                        e.Command.Parameters["@CustomerUserID"].Value =
priv_guid_customer_user_id;
                    }
                }
            }
        }

        /// <summary>
        /// Ελέγχος δικαιωμάτων εκτύπωσης παραγγελίας για τον αιτούμενο χρήστη.
        /// </summary>
        /// <param name="msu_user">Ο χρήστης που αιτείται την άδεια.</param>
        /// <returns>Την άδεια εκτύπωσης της παραγγελίας.</returns>
```

```
        private bool uf_b_AllowOrderPrint(MembershipUser msu_user)
        {
            bool b_allow = false; // Μεταβλητή δικαιωμάτων εκτύπωσης
παραγγελίων.

            // Ένας ανύπαρκτος χρήστης δεν έχει ποτέ δικαιώματα για εκτύπωση
παραγγελιών.
            if (msu_user == null)
            {
                return b_allow;
            }

            Guid guid_user_id = (Guid)msu_user.ProviderUserKey; // Μοναδικό
προσδιοριστικό (Unique ID) συνδεδεμένου χρήστη.

            // Έλεγχος αν ο χρήστης είναι διαχειριστής. Οι διαχειριστές έχουν
πάντα δικαιώματα παραγγελιών για όλους τους εγγεγραμμένους χρήστης
            // (π.χ. αποδείξεις παραγγελιών από την έδρα αποστολής του
εμπορεύματος).
            if (HttpContext.Current.User.IsInRole("administrator") == false)
            {
                // Έλεγχος αν ο συνδεδεμένος χρήστης έχει δικαιώματα εκτύπωσης
επί της συγκεκριμένης παραγγελίας.
                if (guid_user_id != priv_guid_customer_user_id)
                {
                    return b_allow;
                }
            }

            // Αν περάσει όλους τους προηγούμενους ελέγχους, τότε ο χρήστης
έχει δικαιώματα εκτύπωσης παραγγελλιών.
            b_allow = true;

            return b_allow;
        }
    }
}
```

Έλεγχος παραγγελιών

Orders.aspx.cs

Ο ακόλουθος κώδικας επιτρέπει στον χρήστη να παρακολουθεί την κατάσταση των παραγγελιών του. Επίσης, ο χρήστης μπορεί να εκτυπώσει μια παραγγελία μέσω του εκτυπωτικού Order Print:

```
        protected void sqlds_orders_Selecting(object sender,
SqlDataSourceSelectingEventArgs e)
        {
            // Συλλογή των στοιχείων λογαριασμού του χρήστη που είναι
συνδεδεμένος στο σύστημα.
            MembershipUser msu_user =
Membership.GetUser(HttpContext.Current.User.Identity.Name);

            // Αν η συλλογή των στοιχείων του είναι επιτυχής, τότε θα
συλλέξουμε από τον πίνακα παραγγελιών (Orders) τις παραγγελιές του χρήστη.
            if (msu_user != null)
            {
                Guid guid_customer = (Guid)msu_user.ProviderUserKey; //
Μοναδικό προσδιοριστικό (Unique ID) συνδεδεμένου χρήστη.
```

```
                // Εισαγωγή στο σύστημα του μοναδικού προσδιοριστικού (Unique
ID) για την επιλογή της εγγραφών παραγγελιών του χρήστη
                // από τον πίνακα παραγγελιών (Orders).
                e.Command.Parameters["@CustomerUserID"].Value = guid_customer;
            }
        }

        protected void lvw_order_DataBound(object sender, EventArgs e)
        {
            // Δημιουργία και ενεργοποίηση του συνδέσμου εκτύπωσης παραγγελίας
(OrderPrint). Ο σύνδεσμος είναι απεργοποιημένος όταν
            // δεν υπάρχει ο κωδικός εγγραφής της παραγγελίας και
ενεργοποιημένος όταν υπάρχει.
            CodeClass.uf_v_CreateOrderPrintHyperLink(gvw_orders,
hlnk_print_order);
        }
```

Λοιπός κώδικας

Βιβλιοθήκη κώδικα

CodeClass.cs

Ο ακόλουθος κώδικας αποτελεί την βιβλιοθήκη κώδικα της εφαρμογής, όπου επαναλβανόμενα τμήματα κώδικα αποθηκεύονται εδώ για χρήση από όλα τα μέρη της εφαρμογής:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

using System.Web.UI.WebControls;
using System.Web.UI;
using System.Web.Security;

namespace FactorySales.Code
{
    public static class CodeClass
    {
        /// <summary>
        /// Λήψη από τις μεταβλητές φυλλομετρητή (Query String: μεταβλητές που
αποτελούν μέρος του συνδέσμου και περιέχουν δεδομένα) της αιτούμενης ακέραιας
τιμής.
        /// </summary>
        /// <param name="httpr_request">Η αίτηση ιστού (Web Request) από όπου
θα ζητήσουμε την μεταβλητή φυλλομετρητή (Query String) που επιθυμούμε.</param>
        /// <param name="s_querystring">Η μεταβλητή φυλλομετρητή (Query String)
που θα λάβουμε.</param>
        /// <returns>Η τιμή που λάβαμε από την μεταβλητή φυλλομετρητή (Query
String) σε ακέραια μορφή (integer).</returns>
        public static int uf_i_QueryStringToInteger(HttpRequest httpr_request,
string s_querystring)
        {
            // Η μεταβλητή αποθήκευσης της ακέραιας τιμής (integer) της
μεταβλητής φυλλομετρητή (Query String) με την εξ ορισμού τιμή της.
            int i_value = 0;

            // Αν δεν υπάρχει έγκυρη μορφή της μεταβλητής φυλλομετρητή (Query
String) που επιθυμούμε να λάβουμε, τότε επέστρεψε την εξ ορισμού τιμή.
```

```
            if ((s_querystring == null) || (s_querystring.Trim() == ""))
            {
                return i_value;
            }

            // Λήψη της μεταβλητής φυλλομετρητή (Query String) από την αίτηση
ιστού (Web Request).
            string s_value = httpr_request.QueryString[s_querystring];

            // Μετατροπή της ακολουθίας χαρακτήρων (string) σε ακέραιο
(integer). Σε περίπτωση αποτυχίας, επιστρέφει 0.
            int.TryParse(s_value, out i_value);

            // Επιστρέφει την τιμή της αιτούμενης μεταβλητή φυλλομετρητή (Query
String) σε ακέραια τιμή (integer).
            return i_value;
        }

        /// <summary>
        /// Μεταβολή της κατάστασης ενός αντικειμένου ιστού (Web Control) κατά
βούληση.
        /// </summary>
        /// <param name="wctrl_control">Το αντικείμενο ιστού (Web Control) που
θέλουμε να μεταβάλουμε την κατάστασή του.</param>
        /// <param name="b_enable">Η μεταβολή της κατάστασης του αντικειμένου
ιστού (Web Control) που επιθυμούμε.</param>
        public static void uf_v_EnableWebControl(WebControl wctrl_control, bool
b_enable)
        {
            // Έλεγχος αν το αντικείμενο ιστού (Web Control) περιέχει δεδομένα.
Αν δεν περιέχει δεδομένα, τότε αποτελεί κενό αντικείμενο και
            // δεν υπάρχει λόγος μεταβολής ορατότητας και κατάστασης.
            if (wctrl_control != null)
            {
                // Μεταβολή ορατότητας και κατάσταση αντικειμένου ιστού σύμφωνα
(Web Control) με την μεταβλητή κατάστασης.
                wctrl_control.Visible = b_enable;
                wctrl_control.Enabled = b_enable;
            }
        }

        /// <summary>
        /// Λήψη της τιμής του υποαντικειμένου (child object) με το ζητούμενο
όνομα και μετατροπή της σε ακέραιο (integer).
        /// </summary>
        /// <param name="s_control_child">Το ζητούμενο όνομα υποαντικειμένου
(child object).</param>
        /// <param name="ctrl_control">Το αντικείμενο (parent object) που
περιέχει το ζητούμενο υποαντικειμένο (child object).</param>
        /// <returns>Επιστρέφει την ακέραια (integer) τιμή του υποαντικειμένου
(child object) με το ζητούμενο όνομα.</returns>
        public static int uf_i_ConvertChildControlValueToInteger(string
s_control_child, Control ctrl_control)
        {
            // Η μεταβλητή αποθήκευσης της ακέραιας (integer) τιμής του
υποαντικειμένου (child object) με την εξ ορισμού τιμή της.
            int i_value = 0;

            // Αν το αντικείμενο (parent object) και το ζητούμενο όνομα
υποαντικειμένου (child object) δεν είναι έγκυρα, τότε τερματίζουμε την
διαδικασία και
            // επιστρέφουμε την εξ ορισμού τιμή της μεταβλητής.
```

```
            if ((s_control_child == null) || (s_control_child.Trim() == "") ||
(ctrl_control == null))
            {
                return i_value;
            }

            // Έλεγχος αν μέσα στο αντικείμενο (parent object), υπάρχει η
ετικέτα (Label) με το ζητούμενο όνομα υποαντικειμένου (child object).
            if (ctrl_control.FindControl(s_control_child) is Label)
            {
                // Λήψη της ετικέτας (Label) του αντικειμένου (control) ενώ
αποτελεί υπόαντικειμενο ενός άλλου. Η διαδικασία αυτή ακολουθείται
                // καθώς δεν είναι εφικτή η απευθείας αλληλεπίδραση, καθώς
εμπεριέχεται στο σύνολο των υπόαντικειμένων (child objects) ενός άλλου
                // αντικειμένου (parent object)).
                Label lbl_control =
(Label)ctrl_control.FindControl(s_control_child);

                // Λήψη της τιμής της ετικέτας (Label) και μετατροπή της σε
ακέραιο (integer).
                i_value = uf_i_ConvertValueToInteger(lbl_control.Text);
            }

            // Επιστρέφει την ακέραια (integer) τιμή του υποαντικειμένου (child
object) με το ζητούμενο όνομα.
            return i_value;
        }

        /// <summary>
        /// Λήψη της τιμής του υποαντικειμένου (child object) με το ζητούμενο
όνομα και μετατροπή της σε δεκαδικό (decimal).
        /// </summary>
        /// <param name="s_control_child">Το ζητούμενο όνομα υποαντικειμένου
(child object).</param>
        /// <param name="ctrl_control">Το αντικείμενο (parent object) που
περιέχει το ζητούμενο υποαντικειμένο (child object).</param>
        /// <returns>Επιστρέφει την δεκαδική (decimal) τιμή του υποαντικειμένου
(child object) με το ζητούμενο όνομα.</returns>
        public static decimal uf_d_ConvertChildControlValueToDecimal(string
s_control_child, Control ctrl_control)
        {
            // Η μεταβλητή αποθήκευσης της δεκαδικής (decimal) τιμής του
υποαντικειμένου (child object) με την εξ ορισμού τιμή της.
            decimal d_value = 0;

            // Αν το αντικείμενο (parent object) και το ζητούμενο όνομα
υποαντικειμένου (child object) δεν είναι έγκυρα, τότε τερματίζουμε την
διαδικασία και
            // επιστρέφουμε την εξ ορισμού τιμή της μεταβλητής.
            if ((s_control_child == null) || (s_control_child.Trim() == "") ||
(ctrl_control == null))
            {
                return d_value;
            }

            // Έλεγχος αν μέσα στο αντικείμενο (parent object), υπάρχει η
ετικέτα (Label) με το ζητούμενο όνομα υποαντικειμένου (child object).
            if (ctrl_control.FindControl(s_control_child) is Label)
            {
                // Λήψη της ετικέτας (Label) του αντικειμένου (control) ενώ
αποτελεί υπόαντικειμενο ενός άλλου. Η διαδικασία αυτή ακολουθείται
```

```
                // καθώς δεν είναι εφικτή η απευθείας αλληλεπίδραση, καθώς
εμπεριέχεται στο σύνολο των υπόαντικειμένων (child objects) ενός άλλου
                // αντικειμένου (parent object)).
                Label lbl_control =
(Label)ctrl_control.FindControl(s_control_child);

                // Λήψη της τιμής της ετικέτας (Label) και μετατροπή της σε
δεκαδικό (decimal).
                d_value = uf_d_ConvertValueToDecimal(lbl_control.Text);
            }

            // Επιστρέφει την δεκαδική (decimal) τιμή του υποαντικειμένου
(child object) με το ζητούμενο όνομα.
            return d_value;
        }

        /// <summary>
        /// Λήψη της τιμής του υποαντικειμένου (child object) με το ζητούμενο
όνομα και μετατροπή της σε ακολουθία χαρακτήρων (string).
        /// </summary>
        /// <param name="s_control_child">Το ζητούμενο όνομα υποαντικειμένου
(child object).</param>
        /// <param name="ctrl_control">Το αντικείμενο (parent object)  που
περιέχει το ζητούμενο υποαντικειμένο (child object).</param>
        /// <param name="s_default_value">Η εξ ορισμού τιμή της μεταβλητής
αποθήκευσης της ακολουθίας χαρακτήρων (string) τιμής του
υποαντικειμένου.</param>
        /// <returns>Επιστρέφει την ακολουθία χαρακτήρων (string) τιμή του
υποαντικειμένου (child object) με το ζητούμενο όνομα..</returns>
        public static string uf_s_ConvertChildControlValueToString(string
s_control_child, Control ctrl_control, string s_default_value)
        {
            // Η μεταβλητή αποθήκευσης της ακολουθίας χαρακτήρων (string) τιμής
του υποαντικειμένου (child object) με την εξ ορισμού τιμή της.
            string s_value = "";

            // Έλεγχος της εγκυρότητας της δοθέντας εξ ορισμού τιμής της
μεταβλητής αποθήκευσης της ακολουθίας χαρακτήρων (string) τιμής του
υποαντικειμένου
            // (child object) και ορισμός της τιμής της μεταβλητής με αυτήν.
            if ((s_default_value != null) && (s_default_value.Trim() != ""))
            {
                s_value = s_default_value;
            }

            // Αν το αντικείμενο (parent object) και το ζητούμενο όνομα
υποαντικειμένου (child object) δεν είναι έγκυρα, τότε τερματίζουμε την
διαδικασία και
            // επιστρέφουμε την εξ ορισμού τιμή της μεταβλητής.
            if ((s_control_child == null) || (s_control_child.Trim() == "") ||
(ctrl_control == null))
            {
                return s_value;
            }

            // Έλεγχος αν μέσα στο αντικείμενο (parent object), υπάρχει η
ετικέτα (Label) με το ζητούμενο υποαντικείμενο (child object).
            if (ctrl_control.FindControl(s_control_child) is Label)
            {
                // Λήψη της ετικέτας (Label) του αντικειμένου (control) ενώ
αποτελεί υπόαντικειμενο ενός άλλου. Η διαδικασία αυτή ακολουθείται
```

```
                // καθώς δεν είναι εφικτή η απευθείας αλληλεπίδραση, καθώς
εμπεριέχεται στο σύνολο των υπόαντικειμένων (child objects) ενός άλλου
                // αντικειμένου (parent object)).
                Label lbl_control =
(Label)ctrl_control.FindControl(s_control_child);

                // Λήψη της τιμής της ετικέτας (Label) και μετατροπή της σε
ακολουθία χαρακτήρων (string).
                s_value = uf_s_ConvertValueToString(lbl_control.Text);
            }

            // Επιστρέφει την ακολουθία χαρακτήρων (string) τιμή του
υποαντικειμένου (child object) με το ζητούμενο όνομα.
            return s_value;
        }

        /// <summary>
        /// Λαμβάνει μια ακολουθία χαρακτήρων (string) από ένα αντικείμενο
(object) ή μια μεταβλητή και την μετατρέπει σε ακέραιο (integer).
        /// </summary>
        /// <param name="obj_value">Η ακολουθία χαρακτήρων αντικειμένου (object
string) που θα τροποποιήσουμε.</param>
        /// <returns>Την τροποποιημένη τιμή σε ακέραιο (integer).</returns>
        public static int uf_i_ConvertValueToInteger(object obj_value)
        {
            // Η μεταβλητή που θα αποθηκεύσει την τροποποιημένη τιμή της
ακολουθίας χαρακτήρων αντικειμένου (object string).
            int i_converted_value = 0;

            // Μετατροπή της τιμής της ακολουθίας χαρακτήρων αντικειμένου
(object string) μόνο όταν το αντικείμενο (object) που την παρέχει είναι έγκυρο.
            if (obj_value != null)
            {
                // Μετατροπή της ακολουθίας χαρακτήρων αντικειμένου (object
string) σε ακέραιο (integer). Σε περίπτωση αποτυχίας, επιστρέφει 0.
                // Η ακολουθία χαρακτήρων αντικειμένου αντιστοιχεί στην τιμή
που μπορεί να έχει κάποιο αντικείμενο (object) ή κάποια μεταβλητή.
                int.TryParse(obj_value.ToString(), out i_converted_value);
            }

            // Επιστρέφει την τροποποιημένη τιμή σε ακέραιο (integer).
            return i_converted_value;
        }

        /// <summary>
        /// Λαμβάνει μια ακολουθία χαρακτήρων (string) από ένα αντικείμενο
(object) ή μια μεταβλητή και την μετατρέπει σε δεκαδικό (decimal).
        /// </summary>
        /// <param name="obj_value">Η ακολουθία χαρακτήρων αντικειμένου (object
string) που θα τροποποιήσουμε.</param>
        /// <returns>Την τροποποιημένη τιμή σε δεκαδικό (decimal).</returns>
        public static decimal uf_d_ConvertValueToDecimal(object obj_value)
        {
            // Η μεταβλητή που θα αποθηκεύσει την τροποποιημένη τιμή της
ακολουθίας χαρακτήρων αντικειμένου (object string).
            decimal d_converted_value = 0;

            // Μετατροπή της τιμής της ακολουθίας χαρακτήρων αντικειμένου
(object string) μόνο όταν το αντικείμενο (object) που την παρέχει είναι έγκυρο.
            if (obj_value != null)
            {
```

```
                // Μετατροπή της ακολουθίας χαρακτήρων αντικειμένου (object
string) σε δεκαδικό (decimal). Σε περίπτωση αποτυχίας, επιστρέφει 0.
                // Η ακολουθία χαρακτήρων αντικειμένου αντιστοιχεί στην τιμή
που μπορεί να έχει κάποιο αντικείμενο (object) ή κάποια μεταβλητή.
                decimal.TryParse(obj_value.ToString(), out d_converted_value);
            }

            // Επιστρέφει την τροποποιημένη τιμή σε δεκαδικό (decimal).
            return d_converted_value;
        }

        /// <summary>
        /// Λαμβάνει μια ακολουθία χαρακτήρων (string) από ένα αντικείμενο
(object) ή μια μεταβλητή και την μετατρέπει σε ακολουθία χαρακτήρων (string).
        /// </summary>
        /// <param name="obj_value">Η ακολουθία χαρακτήρων αντικειμένου (object
string) που θα τροποποιήσουμε.</param>
        /// <returns>Την τροποποιημένη τιμή σε ακολουθία χαρακτήρων
(string).</returns>
        public static string uf_s_ConvertValueToString(object obj_value)
        {
            // Η μεταβλητή που θα αποθηκεύσει την τροποποιημένη τιμή της
ακολουθίας χαρακτήρων αντικειμένου (object string).
            string s_converted_value = "";

            // Μετατροπή της τιμής της ακολουθίας χαρακτήρων αντικειμένου
(object string) μόνο όταν το αντικείμενο (object) που την παρέχει είναι έγκυρο.
            if (obj_value != null)
            {
                // Μετατροπή της ακολουθίας χαρακτήρων αντικειμένου (object
string) σε ακολουθία χαρακτήρων (string). Σε περίπτωση αποτυχίας, επιστρέφει
"".
                // Η ακολουθία χαρακτήρων αντικειμένου αντιστοιχεί στην τιμή
που μπορεί να έχει κάποιο αντικείμενο (object) ή κάποια μεταβλητή.
                s_converted_value = obj_value.ToString();
            }

            // Επιστρέφει την τροποποιημένη τιμή σε ακολουθία χαρακτήρων
(string).
            return s_converted_value;
        }

        /// <summary>
        /// Μετάφραση της απαρίθμησης δεδομένων (enumeration) που δείχνει την
διαθεσιμότητα κάθε εγγραφής παραγγελίας σε ακολουθία χαρακτήρων (string)
ελληνικών.
        /// </summary>
        /// <param name="cen_availability">Η απαρίθμηση δεδομένων (enumeration)
που θα μεταφραστεί.</param>
        /// <returns>Την μεταφρασμένη απαρίθμησης δεδομένων (enumeration) σε
ακολουθία χαρακτήρων (string) ελληνικών.</returns>
        public static string
uf_s_AvailabilityEnumToString(CodeClass.availability cen_availability)
        {
            // Η μεταβλητή ακολουθίας χαρακτήρων (string) που θα αποθηκεύσει
την τιμή της μετάφρασης.
            string s_availability;

            // Μετάφραση της απαρίθμησης δεδομένων (enumeration) σε ακολουθία
χαρακτήρων (string) ελληνικών.
            switch (cen_availability)
            {
```

```
                case CodeClass.availability.available:
                    s_availability = "Διαθέσιμο";
                    break;
                case CodeClass.availability.preorder:
                    s_availability = "Προπαραγγελία";
                    break;
                default:
                    s_availability = "Εξαντλήθηκε";
                    break;
            }

            // Επιστρέφει την μεταφρασμένη απαρίθμησης δεδομένων (enumeration) σε ακολουθία χαρακτήρων (string) ελληνικών.
            return s_availability;
        }

        /// <summary>
        /// Μετάφραση της ακολουθίας χαρακτήρων (string) ελληνικών σε απαρίθμηση δεδομένων (enumeration) που δείχνει την διαθεσιμότητα κάθε εγγραφής παραγγελίας.
        /// </summary>
        /// <param name="s_availability">Η ακολουθία χαρακτήρων (string) ελληνικών που θα μεταφραστεί.</param>
        /// <returns>Την ακολουθία χαρακτήρων (string) ελληνικών σε απαρίθμηση δεδομένων (enumeration).</returns>
        public static CodeClass.availability uf_cen_AvailabilityStringToEnum(string s_availability)
        {
            // Η μεταβλητή απαρίθμησης δεδομένων (enumeration) που θα αποθηκεύσει την τιμή της μετάφρασης.
            CodeClass.availability cen_availability;

            // Μετάφραση της ακολουθίας χαρακτήρων (string) ελληνικών σε απαρίθμηση δεδομένων (enumeration).
            switch (s_availability)
            {
                case "Διαθέσιμο":
                    cen_availability = CodeClass.availability.available;
                    break;
                case "Προπαραγγελία":
                    cen_availability = CodeClass.availability.preorder;
                    break;
                default:
                    cen_availability = CodeClass.availability.stock_out;
                    break;
            }

            // Επιστρέφει την ακολουθία χαρακτήρων (string) ελληνικών σε απαρίθμηση δεδομένων (enumeration).
            return cen_availability;
        }

        /// <summary>
        /// Ελέγχει δύο ημερομηνίες και ώρες (DateTime) και επιστρέφει την πιο πρόσφατη από αυτές σε μορφή ακολουθίας χαρακτήρων (string).
        /// </summary>
        /// <param name="dt_date_current">Η ισχύουσα ημερομηνία και ώρα (DateTime).</param>
        /// <param name="dt_date_compare">Η συγκρινόμενη ημερομηνία και ώρα (DateTime).</param>
        /// <returns>Την πιο πρόσφατη ημερομηνία και ώρα (DateTime) από τις συγκρινόμενες σε ακολουθία χαρακτήρων (string).</returns>
```

```
        public static string uf_s_DateTimeToString(DateTime dt_date_current,
DateTime dt_date_compare)
        {
            // Η μεταβλητή αποθήκευσης της πιο πρόσφατης ημερομηνίας και ώρας
(DateTime).
            string s_value = "";

            // Αν η ισχύουσα ημερομηνία και ώρα (DateTime) είναι μεγαλύτερη της
συγκρινόμενης τότε την επιλέγουμε για την ακολουθία χαρακτήρων (string).
            if (dt_date_current > dt_date_compare)
            {
                s_value = dt_date_current.ToString();
            }
            else // Διαφορετικά, επιλέγουμε την συγκρινόμενη.
            {
                s_value = dt_date_compare.ToString();
            }

            // Επιστρέφει την ακολουθία χαρακτήρων (string) που δηλώνει την πιο
πρόσφατη ημερομηνία και ώρα (DateTime).
            return s_value;
        }

        /// <summary>
        /// Ελέγχει αν μια κατάσταση αληθείας (boolean) και ορίζει την τιμή
ημερομηνίας και ώρας (DateTime) βάση την αιτούμενη ή την εξ ορισμού.
        /// </summary>
        /// <param name="b_selected">Η κατάσταση αληθείας (boolean) που ορίζει
την τιμή ημερομηνίας και ώρας (DateTime)</param>
        /// <param name="dt_default_date"></param>
        /// <param name="dt_set_date"></param>
        /// <returns>Την αιτούμενη ή την εξ ορισμού ημερομηνία και ώρα
(DateTime).</returns>
        public static DateTime uf_dt_SetDateTimeByListItemSelection(bool
b_selected, DateTime dt_default_date, DateTime dt_set_date)
        {
            // Η μεταβλητή αποθήκευσης της τιμής της επιστρεφόμενης ημερομηνίας
και ώρας (DateTime) και ορισμός με την δοθέντα εξ ορισμού
            // ημερομηνίας και ώρας (DateTime).
            DateTime dt_date = dt_default_date;

            // Αν η κατάσταση αληθείας (boolean) είναι ορισμένη στην αλήθεια,
τότε όρισε την ημερομηνία και ώρα (DateTime) στην αιτούμενη.
            if (b_selected == true)
            {
                dt_date = dt_set_date;
            }

            // Επιστρέφει την αιτούμενη ή την εξ ορισμού ημερομηνία και ώρα
(DateTime).
            return dt_date;
        }

        /// <summary>
        /// Μετατροπή μιας ακολουθίας χαρακτήρων (string) σε ημερομηνία και ώρα
(DateTime).
        /// </summary>
        /// <param name="s_date">Η ημερομηνία se ακολουθία χαρακτήρων
(string).</param>
        /// <returns>Η ημερομηνία από ακολουθία χαρακτήρων (string) σε
ημερομηνία και ώρα (DateTime).</returns>
        public static DateTime uf_dt_StringToDateTime(string s_date)
```

```
        {
            // Ορισμός της μεταβλητής ημερομηνίας και ώρας όπου θα αποθηκευτούν
οι αλλαγές.
            DateTime dt_date;

            // Μετατροπή της ακολουθίας χαρακτήρων (string) στην απαιτούμενη
ημερομηνία και ώρα (DateTime). Σε περίπτωση αποτυχίας επιστρέφει 1/1/0001
12:00:00 πμ.
            DateTime.TryParse(s_date, out dt_date);

            // Επιστρέφει την τροποποιημένη τιμή σε ημερομηνία και ώρα
(DateTime).
            return dt_date;
        }

        /// <summary>
        /// Ελέγχος μεταξύ δύο ημερομηνιών και ωρών (DateTime) για αν η πρώτη
είναι μεγαλύτερη ή ίση της δεύτερης.
        /// </summary>
        /// <param name="dt_date_current">Η ημερομηνία και ώρα (DateTime) που
θα ελέγξουμε.</param>
        /// <param name="dt_date_compare">Η ημερομηνία και ώρα (DateTime) με
την οποία θα συγκρίνουμε την ελεγχόμενη.</param>
        /// <returns>Την κατάσταση αληθείας (boolean) σύγκρισης της πρώτης
ημερομηνίας και ώρας (DateTime) με την δεύτερη.</returns>
        public static bool uf_b_DateTimeGreaterEqualCompare(DateTime
dt_date_current, DateTime dt_date_compare)
        {
            // Μεταβλητή συγκρίσεως μεταξύ των δύο ημερομηνιών και ωρών
(DateTime).
            bool b_greater_equal = false;

            // Σύγκριση της τιμής της πρώτης ημερομηνίας και ώρας (DateTime) με
την δεύτερη.
            if (dt_date_current >= dt_date_compare)
            {
                b_greater_equal = true;
            }

            // Επιστρέφει αν η ελεγχόμενη ημερομηνία και ώρα (DateTime) είναι
μεγαλύτερη ή ίση της συγκρινόμενης.
            return b_greater_equal;
        }

        /// <summary>
        /// Μεταβολή της κατάστασης επιλογής και αλληλεπίδρασης ενός
αντικειμένου λίστας (ListItem) σύμφωνα με μία μεταβλήτη μεταβολής κατάστασης.
        /// </summary>
        /// <param name="li_item">Το αντικείμενο λίστας (ListItem) που θα
μεταβάλουμε την κατάστασή του.</param>
        /// <param name="b_found">Η μεταβλητή μεταβολής της κατάστασης του
αντικειμένου λίστας (ListItem)</param>
        public static void uf_v_CheckboxDateTimeControl(ListItem li_item, bool
b_found)
        {
            // Έλεγχος αντικειμένου λίστας (ListItem) για έγκυρα δεδομένα.
            if (li_item == null)
            {
                return;
            }
```

```
            // Αν η μεταβλητή μεταβολής είναι αληθινή (true), τότε επιλέγουμε
το αντικείμενο λίστας (ListItem)
            // αλλά απενεργοποιούμε κάθε αλληλεπίδραση του χρήστη μαζί του.
            if (b_found == true)
            {
                li_item.Selected = true;
                li_item.Enabled = false;
            }
            else // Διαφορετικά, δεν το επιλέγουμε αλλά ενεργοποιούμε την
αλλεπίδραση του χρήστη με αυτό.
            {
                li_item.Selected = false;
                li_item.Enabled = true;
            }
        }

        /// <summary>
        /// Δημιουργεί τον υπερσύνδεσμο (HyperLink) προς την εκτύπωση
παραγγελίας (OrderPrint).
        /// </summary>
        /// <param name="gvw_orders">Ο πίνακας όψεως (GridView) από τον οποίο
θα λάβουμε τον κωδικό εγγραφής της παραγγελίας για εκτύπωση.</param>
        /// <param name="hlnk_print_order">Ο υπερσύνδεσμος (HyperLink) που θα
ενεργοποιήσουμε για την εκτύπωση της παραγγελίας.</param>
        public static void uf_v_CreateOrderPrintHyperLink(GridView gvw_orders,
HyperLink hlnk_print_order)
        {
            // Έλεγχος πίνακα όψεως (GridView) και υπερσυνδέσμου (HyperLink)
για έγκυρα δεδομένα. Τα κενά αντικείμενα (objects) απορρίπτονται
            // και η διαδικασία τερματίζεται, για την προστασία του συστήματος.
            if ((gvw_orders == null) || (hlnk_print_order == null))
            {
                return;
            }

            string s_print_url = ""; // Μεταβλητή αποθήκευσης συνδέσμου προς
την σελίδα εκτύπωσης παραγγελίας (OrderPrint).
            bool b_print_order_hyperlink = false; // Μεταβλητή ενεργοποίησης
συνδέσμου προς την σελίδα εκτύπωσης παραγγελίας (OrderPrint).

            // Η επιλεγμένη γραμμή στον πίνακα όψεως παραγγελίων.
            int i_selected_index = gvw_orders.SelectedIndex;

            // Αν υπάρχει επιλεγμένη γραμμή στον πίνακα όψεως παραγγελίων, τότε
προσπάθησε να δημιουργήσεις τον σύνδεσμο προς την εκτύπωση παραγγελίας
(OrderPrint).
            if ((i_selected_index > -1) &&
(gvw_orders.DataKeys[i_selected_index].Values.Count > 0))
            {
                // Λήψη του κωδικού εγγραφής της επιλεγμένης παραγγελίας από
τον πίνακα όψεως παραγγελίων.
                string s_id =
gvw_orders.DataKeys[i_selected_index].Values[0].ToString();

                // Αν υπάρχει ο κωδικός εγγραφής της επιλεγμένης παραγγελίας,
τότε δημιούργησε τον σύνδεσμο προς την εκτύπωση παραγγελίας (OrderPrint)
                // και ενεργοποίησε τον.
                if ((s_id != null) && (s_id.Trim() != ""))
                {
                    s_print_url = "~/Member/OrderPrint.aspx?id=" + s_id;
                    b_print_order_hyperlink = true;
                }
```

```
            }

            // Δημιουργία και ενεργοποίηση του συνδέσμου εκτύπωσης παραγγελίας
(OrderPrint) σύμφωνα με όσα έχουν πραγματοποιηθεί παραπάνω (απεργοποιημένος
όταν
            // δεν υπάρχει ο κωδικός εγγραφής της παραγγελίας και
ενεργοποιημένος όταν υπάρχει).
            hlnk_print_order.NavigateUrl = s_print_url;
            CodeClass.uf_v_EnableWebControl(hlnk_print_order,
b_print_order_hyperlink);
        }

        /// <summary>
        /// Διαδικασία που ακολουθείται όταν συνδεθούν τα δεδομένα μιας
εγγραφής από μια πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση
δεδομένων)
        /// στον πίνακα όψεως (GridView), σε μια φόρμα καταχώρησης διαχειριστή.
        /// </summary>
        /// <param name="gvw_items">Ο πίνακας όψεως (GridView) που εμφανίζει τα
δεδομένα μας.</param>
        /// <param name="fvw_item">Η φόρμα όψεως (FormView) που διαχειρίζεται
μια επιλεγμένη εγγραφή.</param>
        public static void uf_v_AdministratorGridViewRowDataBound(GridView
gvw_items, FormView fvw_item)
        {
            // Έλεγχος πίνακα όψεως (GridView) και φόρμας όψεως (FormView) για
έγκυρα δεδομένα. Τα κενά αντικείμενα (objects) απορρίπτονται
            // και η διαδικασία τερματίζεται, για την προστασία του συστήματος.
            if ((gvw_items == null) || (fvw_item == null))
            {
                return;
            }

            // Αλλαγή της κατάστασης της φόρμας όψεως (FormView) σε ανάγνωση
μόνο. Κάθε φόρμα όψεως (FormView) έχει την δυνατότητα να ανταλλάσσεται
            // μεταξύ τριών φορμών: της εμφάνισης δεδομένων, της μεταβολής τους
ή στην εισαγωγή νέων.
            fvw_item.ChangeMode(FormViewMode.ReadOnly);

            // Επιλογή της τελευταίας γραμμής της παρούσας σελίδας του πίνακα
όψεως (GridView).
            gvw_items.SelectedIndex = gvw_items.Rows.Count - 1;
        }

        /// <summary>
        /// Διαδικασία που ακολουθείται όταν ενημερωθεί μια εγγραφή σε μια πηγή
αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση δεδομένων)
        /// μέσω της φόρμας όψεως (FormView), σε μια φόρμα καταχώρησης
διαχειριστή.
        /// </summary>
        /// <param name="gvw_items">Ο πίνακας όψεως (GridView) που εμφανίζει τα
δεδομένα μας.</param>
        /// <param name="fvw_item">Η φόρμα όψεως (FormView) που διαχειρίζεται
μια επιλεγμένη εγγραφή.</param>
        public static void uf_v_AdministratorFormViewItemUpdated(GridView
gvw_items, FormView fvw_item)
        {
            // Έλεγχος πίνακα όψεως (GridView) και φόρμας όψεως (FormView) για
έγκυρα δεδομένα. Τα κενά αντικείμενα (objects) απορρίπτονται
            // και η διαδικασία τερματίζεται, για την προστασία του συστήματος.
            if ((gvw_items == null) || (fvw_item == null))
            {
```

```
                return;
            }

            // Σύνδεση δεδομένων με τον πίνακα όψεως (GridView). Στην ουσία
ξαναφορτώνουμε τα δεδομένα, για να ενημερωθεί ο πίνακας όψεως (GridView)
            // με τις αλλαγές.
            gvw_items.DataBind();

            // Αλλαγή της κατάστασης της φόρμας όψεως (FormView) σε ανάγνωση
μόνο. Κάθε φόρμα όψεως (FormView) έχει την δυνατότητα να ανταλλάσσεται
            // μεταξύ τριών φορμών: της εμφάνισης δεδομένων, της μεταβολής τους
ή στην εισαγωγή νέων.
            fvw_item.ChangeMode(FormViewMode.ReadOnly);
        }

        /// <summary>
        /// Διαδικασία που ακολουθείται όταν εισαχθεί μια εγγραφή σε μια πηγή
αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση δεδομένων)
        /// μέσω της φόρμας όψεως (FormView), σε μια φόρμα καταχώρησης
διαχειριστή.
        /// </summary>
        /// <param name="gvw_items">Ο πίνακας όψεως (GridView) που εμφανίζει τα
δεδομένα μας.</param>
        /// <param name="fvw_item">Η φόρμα όψεως (FormView) που διαχειρίζεται
μια επιλεγμένη εγγραφή.</param>
        public static void uf_v_AdministratorFormViewItemInserted(GridView
gvw_items, FormView fvw_item)
        {
            // Έλεγχος πίνακα όψεως (GridView) και φόρμας όψεως (FormView) για
έγκυρα δεδομένα. Τα κενά αντικείμενα (objects) απορρίπτονται
            // και η διαδικασία τερματίζεται, για την προστασία του συστήματος.
            if ((gvw_items == null) || (fvw_item == null))
            {
                return;
            }

            // Σύνδεση δεδομένων με τον πίνακα όψεως (GridView). Στην ουσία
ξαναφορτώνουμε τα δεδομένα, για να ενημερωθεί ο πίνακας όψεως (GridView)
            // με τις αλλαγές.
            gvw_items.DataBind();

            /// Διαδικασία που ακολουθείται όταν συνδεθούν τα δεδομένα μιας
εγγραφής από μια πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση
δεδομένων)
            /// στον πίνακα όψεως (GridView), σε μια φόρμα καταχώρησης
διαχειριστή.
            uf_v_AdministratorGridViewRowDataBound(gvw_items, fvw_item);
        }

        /// <summary>
        /// Διαδικασία που ακολουθείται όταν διαγραφεί μια εγγραφή σε μια πηγή
αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση δεδομένων)
        /// μέσω της φόρμας όψεως (FormView), σε μια φόρμα καταχώρησης
διαχειριστή.
        /// </summary>
        /// <param name="gvw_items">Ο πίνακας όψεως (GridView) που εμφανίζει τα
δεδομένα μας.</param>
        /// <param name="fvw_item">Η φόρμα όψεως (FormView) που διαχειρίζεται
μια επιλεγμένη εγγραφή.</param>
        public static void uf_v_AdministratorFormViewItemDeleted(GridView
gvw_items, FormView fvw_item)
        {
```

```
            // Έλεγχος πίνακα όψεως (GridView) και φόρμας όψεως (FormView) για
έγκυρα δεδομένα. Τα κενά αντικείμενα (objects) απορρίπτονται
            // και η διαδικασία τερματίζεται, για την προστασία του συστήματος.
            if ((gvw_items == null) || (fvw_item == null))
            {
                return;
            }

            // Σύνδεση δεδομένων με τον πίνακα όψεως (GridView). Στην ουσία
ξαναφορτώνουμε τα δεδομένα, για να ενημερωθεί ο πίνακας όψεως (GridView)
            // με τις αλλαγές.
            gvw_items.DataBind();

            // Αν δεν υπάρχουν εγγραφές στον πίνακα όψεως (GridView), τότε
όρισε την φόρμα όψεως (FormView) σε κατάσταση καταχώρησης.
            if (gvw_items.Rows.Count < 1)
            {
                // Αλλαγή της κατάστασης της φόρμας όψεως (FormView) σε
καταχώρηση. Κάθε φόρμα όψεως (FormView) έχει την δυνατότητα να ανταλλάσσεται
                // μεταξύ τριών φορμών: της εμφάνισης δεδομένων, της μεταβολής
τους ή στην εισαγωγή νέων.
                fvw_item.ChangeMode(FormViewMode.Insert);
            }
        }

        /// <summary>
        /// Διαδικασία που ακολουθείται όταν φορτώνεται στον φυλλομετρητή
(browser) μια σελίδα καταχώρησης διαχειριστή.
        /// </summary>
        /// <param name="pg_page">Η σελίδα καταχώρησης διαχειριστή.</param>
        /// <param name="gvw_items">Ο πίνακας όψεως (GridView) που εμφανίζει τα
δεδομένα μας.</param>
        /// <param name="fvw_item">Η φόρμα όψεως (FormView) που διαχειρίζεται
μια επιλεγμένη εγγραφή.</param>
        public static void uf_v_AdministatorPageLoad(Page pg_page, GridView
gvw_items, FormView fvw_item)
        {
            // Έλεγχος σελίδας (Page), πίνακα όψεως (GridView) και φόρμας όψεως
(FormView) για έγκυρα δεδομένα. Τα κενά αντικείμενα (objects) απορρίπτονται
            // και η διαδικασία τερματίζεται, για την προστασία του συστήματος.
            if ((pg_page == null) || (gvw_items == null) || (fvw_item == null))
            {
                return;
            }

            // Η ακόλουθη διαδικασία πραγματοποιείται μόνο όταν φορτώνει η
σελίδα για πρώτη φορά.
            if (pg_page.IsPostBack == false)
            {
                // Έλεγχος του πλήθους των εγγραφών που είναι συνδεδεμένες με
τον πίνακα όψεως (GridView). Αν υπάρχουν εγγραφές, τότε συνδέονται τα διαθέσιμα
                // δεδομένα με τον πίνακα όψεως (GridView), διαφορετικά
ενεργοποιούμε την κατάσταση καταχώρησης στη φόρμα όψεως (FormView).
                if (gvw_items.Rows.Count > 0)
                {
                    // Διαδικασία που ακολουθείται όταν συνδεθούν τα δεδομένα
μιας εγγραφής από μια πηγή αποθήκευσης δεδομένων (π.χ. ένας πίνακας σε μια βάση
                    // δεδομένων) στον πίνακα όψεως (GridView), σε μια φόρμα
καταχώρησης διαχειριστή.
                    uf_v_AdministratorGridViewRowDataBound(gvw_items,
fvw_item);
                }
```

```
                else
                {
                    // Αλλαγή της κατάστασης της φόρμας όψεως (FormView) σε
καταχώρηση. Κάθε φόρμα όψεως (FormView) έχει την δυνατότητα να ανταλλάσσεται
                    // μεταξύ τριών φορμών: της εμφάνισης δεδομένων, της
μεταβολής τους ή στην εισαγωγή νέων.
                    fvw_item.ChangeMode(FormViewMode.Insert);
                }
            }
        }

        /// <summary>
        /// Έλεγχος παραμέτρων πηγής δεδομένων (Data Source) για σφάλματα σε
μια σελίδα καταχώρησης διαχειριστή.
        /// </summary>
        /// <param name="e">Οι παράμετροι της πηγής δεδομένων (Data Source) που
θα ελέγξουμε για σφάλματα.</param>
        /// <param name="lbl_message">Η ετικέτα (Label) που εμφανίζει στο
χρήστη τα σφάλματα.</param>
        public static void
uf_v_AdministatorSqlDataSourceDeleted(SqlDataSourceStatusEventArgs e, Label
lbl_message)
        {
            // Ελέγχος παραμέτρων για έγκυρα δεδομένα. Τα άκυρα δεδομένα
απορρίπτονται και η διαδικασία τερματίζεται, για την προστασία του συστήματος.
            if (e == null)
            {
                return;
            }

            // Χειριζόμαστε εμείς τα όποια θέματα παρουσιαστούν, άρα
απελευθερώνουμε το σύστημα από αυτήν την έννοια.
            e.ExceptionHandled = true;

            // Ελέγχος ετικέτας (Label) για έγκυρα δεδομένα. Τα άκυρα δεδομένα
απορρίπτονται και η διαδικασία τερματίζεται, για την προστασία του συστήματος.
            if (lbl_message == null)
            {
                return;
            }

            // Αν οι παράμετροι εμφανίσουν κάποιο σφάλμα, τότε θα το
εμφανίσουμε στο διαχειριστή μέσω της ετικέτας (Label).
            if (e.Exception != null)
            {
                // Ορίζουμε το μήνυμα σφάλματος των παραμέτρων.
                string s_message = e.Exception.Message;

                // Αλλαγή μυνήματος σε κάποιο πιο φιλικό στην περίπτωση που
επιχειρηθεί η διαγραφή μιας εγγραφής,
                // ενώ χρησιμοποιείται από κάποιον πίνακα στη βάση δεδομένων.
                if (s_message.ToLower().Contains("the delete statement
conflicted with the reference constraint") == true)
                {
                    s_message = "Η διαγραφή δεν μπορεί να πραγματοποιηθεί,
διότι η εγγραφή χρησιμοποιείται από κάποιον πίνακα.";
                }

                // Εμφάνιση του μηνύματος στο διαχειριστή.
                lbl_message.Text = s_message;
            }
        }
```

```
        /// <summary>
        /// Η δομή δεδομένων (structure) που παρουσιάζει τα δεδομένα της κάθε
εγγραφής μιας παραγγελίας.
        /// </summary>
        public struct order
        {
            public int i_id; // Κωδικός εγγραφής.
            public int i_product_code; // Κωδικός προϊόντος.
            public int i_product_id; // Κωδικός καταχώρησης προϊόντος στο
σύστημα.
            public string s_product_name; // Όνομα προϊόντος.
            public int i_storage_id; // Κωδικός καταχώρησης αποθήκης στο
σύστημα.
            public string s_storage_name; // Όνομα αποθήκης.
            public int i_product_quantity; // Ποσότητα προϊόντος.
            public decimal d_product_price; // Τιμή προϊόντος.
            public int i_stock_id; // Κωδικός καταχώρησης αποθέματος στο
σύστημα.
            public decimal d_product_total_price; // Συνολική τιμή προϊόντος.
            public availability cen_availability; // Διαθεσιμότητα προϊόντος.

            public order(int i_id, int i_product_code, int i_product_id, string
s_product_name, int i_storage_id, string s_storage_name, int
i_product_quantity,
                decimal d_product_price, int i_stock_id, decimal
d_product_total_price, availability cen_availability)
            {
                this.i_id = i_id;
                this.i_product_code = i_product_code;
                this.i_product_id = i_product_id;
                this.s_product_name = s_product_name;
                this.i_storage_id = i_storage_id;
                this.s_storage_name = s_storage_name;
                this.i_product_quantity = i_product_quantity;
                this.d_product_price = d_product_price;
                this.i_stock_id = i_stock_id;
                this.d_product_total_price = d_product_total_price;
                this.cen_availability = cen_availability;
            }
        }

        /// <summary>
        /// Η απαρίθμηση δεδομένων (enumeration) που δείχνει την διαθεσιμότητα
κάθε εγγραφής παραγγελίας.
        /// </summary>
        public enum availability
        {
            available, stock_out, preorder
        }
    }
}
```

Αρχική σελίδα

Default.aspx.cs

Ο ακόλουθος κώδικας δείχνει την αρχική σελίδα της εφαρμογής. Στη σελίδα αυτή απαριθμούνται τα συνολικά καταχωρημένα προϊόντα καθώς και οι διαθέσιμες προσφορές:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace FactorySales
{
    public partial class _Default : Page
    {
        private int priv_i_product_count; // Το πλήθος των προϊόντων που είναι
καταχωρημένα στο ηλεκτρονικό μας κατάστημα.

        protected void Page_Load(object sender, EventArgs e)
        {
            // Εκτελούμε την εντολή sql για να συλλέξουμε τα στοιχεία που
χρειαζόμαστε.
            sqlds_products_count.Select(DataSourceSelectArguments.Empty);

            // Αν η ποσότητα των προϊόντων είναι μεγαλύτερη του μηδενός, τότε
εμφανίζουμε στο χρήστη ένα μήνυμα με την ποικιλία των προϊόντων που είναι
διαθέσιμα
            // από το ηλεκτρονικό μας κατάστημα. Το μήνυμα αυτό είναι μόνο για
διαφημιστικούς λόγους και ενημερώνεται δυναμικά κάθε φορά που καταχωρούμε
προϊόντα.
            if (priv_i_product_count > 0)
            {
                lbl_products_count.Text = "Εδώ, θα μπορέσετε να αγοράσετε αυτά
που χρειάζεστε μέσα από μια ποικιλία από " + priv_i_product_count + "
προϊόντα.";
            }
            else // Διαφορετικά, δεν εμφανίζουμε κανένα μήνυμα καθώς δεν
υπάρχει κανένα προϊόν διαθέσιμο προς πώληση.
            {
                lbl_products_count.Text = "";
            }
        }

        protected void sqlds_products_count_Selected(object sender,
SqlDataSourceStatusEventArgs e)
        {
            // Συλλέγουμε το πλήθος των προϊόντων που είναι καταχωρημένα στο
ηλεκτρονικό μας κατάστημα για να εμφανίσουμε στο χρήστη το ανάλογο μήνυμα.
            priv_i_product_count =
(int)e.Command.Parameters["@ProductsCount"].Value;
        }
    }

}
```

Παραγγελίες προϊόντων

Products.aspx.cs

Ο ακόλουθος κώδικας επιτρέπει στους χρήστες να πραγματοποιούν παραγγελίες προϊόντων μέσω ενός καλαθιού αγορών, δεσμεύοντας προϊόντα από το διαθέσιμο απόθεμα ή προπαραγγέλνοντας:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

using System.Collections;
using System.Data;
using System.IO;
using System.Drawing;
using System.Web.Security;
using FactorySales.Code;

namespace FactorySales
{
    public partial class Products : System.Web.UI.Page
    {
        private int priv_i_order_id; // Μεταβλητή αποθήκευσης του κωδικού
εγγραφής της καταχωρημένης παραγγελίας.
        private int priv_i_stock_quantity; // Μεταβλητή αποθήκευσης του
διαθέσιμου αποθέματος του προϊόντος.
        private int priv_i_storage_address_id; // Μεταβλητή αποθήκευσης της
πρώτης διεύθυνσης αποθήκης.

        protected void Page_Load(object sender, EventArgs e)
        {
            // Η ακόλουθη διαδικασία πραγματοποιείται μόνο όταν φορτώνει η
σελίδα για πρώτη φορά.
            if (Page.IsPostBack == false)
            {
                // Δημιουργία του πίνακα δεδομένων (DataTable) του καλαθιού
αγορών και σύνδεση των εγγραφών του με τον αντίστοιχο πίνακα όψεως (GridView).
                uf_v_CartProducts();

                // Λήψη της μεταβλητής φυλλομετρητή (Query String) που διατηρεί
την τιμή της αναζήτησης του πίνακα όψεως (GridView)
                // των προϊόντων (Products) από την αίτηση ιστού (Web Request).
                string s_search = Request.QueryString["search"];

                // Αν υπάρχει τιμή αναζήτησης (ο χρήστης αναζήτησε προϊόντα
βάση κάποιων κριτηρίων), τότε ενημέρωσε το κουτί κειμένου (TextBox) αναζήτησης
με
                // αυτήν την τιμή και σύλλεξε ανάλογα τα στοιχεία από την βάση
δεδομένων.
                if ((s_search != null) && (s_search.Trim() != ""))
                {
                    // Ενημέρωση του κουτιού κειμένου (TextBox) αναζήτησης με
την τιμή που λάβαμε από την μεταβλήτη φυλλομετρητή (Query String).
                    tbx_search.Text = s_search;

                    // Συλλογή από την βάση δεδομένων των προϊόντων που πληρούν
τα κριτήρια που έθεσε ο χρήστης.
                    uf_v_Search();
                }

                // Λήψη της μεταβλητής φυλλομετρητή (Query String) που διατηρεί
την τιμή της ταξινόμησης του πίνακα όψεως (GridView)
                // των προϊόντων (Products) από την αίτηση ιστού (Web Request).
                string s_sort_expression = Request.QueryString["sort"];
```

```
                // Έλεγχος αν υπάρχει έγκυρη τιμή ταξινόμησης. Αν δεν υπάρχει,
τότε δεν χρειάζεται να επέμβουμε στην ταξινόμηση του πίνακα όψεως (GridView)
                // των προϊόντων (Products)
                if ((s_sort_expression != null) && (s_sort_expression.Trim() !=
""))
                {
                    // Λήψη της μεταβλητής φυλλομετρητή (Query String) που
διατηρεί την τιμή της κατεύθυνσης ταξινόμησης του πίνακα όψεως (GridView)
                    // των προϊόντων (Products) από την αίτηση ιστού (Web
Request).
                    string s_sort_direction = Request.QueryString["direction"];

                    // Μεταβλητή αποθήκευσης της ταξινόμησης του του πίνακα
όψεως (GridView) των προϊόντων (Products).
                    SortDirection sdir_sort_direction;

                    // Έλεγχος αν η μεταβλητή φυλλομετρητή (Query String)
περιέχει την ακόλουθη λέξη. Αν την περιέχει, τότε η ταξινόμηση του πίνακα όψεως
                    // (GridView) των προϊόντων (Products) είναι φθίνουσα.
                    if (s_sort_direction.ToLower() == "descending")
                    {
                        sdir_sort_direction = SortDirection.Descending;
                    }
                    else // Διαφορετικά, η ταξινόμηση του πίνακα όψεως
(GridView) των προϊόντων (Products) είναι αύξουσα.
                    {
                        sdir_sort_direction = SortDirection.Ascending;
                    }

                    // Ορίζουμε την ταξινόμηση του πίνακα όψεως (GridView) των
προϊόντων (Products) σύμφωνα με την τιμή της μεταβλητής αποθήκευσης
ταξινόμησης.
                    gvw_products.Sort(s_sort_expression, sdir_sort_direction);
                }

                // Λήψη των μεταβλητών φυλλομετρητή (Query String) που
διατηρούν την τιμή της σελίδας και της γραμμής του πίνακα όψεως (GridView)
                // των προϊόντων (Products) από την αίτηση ιστού (Web Request).
                int i_page = CodeClass.uf_i_QueryStringToInteger(Request,
"page");
                int i_index = CodeClass.uf_i_QueryStringToInteger(Request,
"index");

                // Ορισμός στον πίνακα όψεως (GridView) των προϊόντων
(Products) της επιλεγμένης σελίδας και γραμμής.
                gvw_products.PageIndex = i_page;
                gvw_products.SelectedIndex = i_index;

                // Σύνδεση δεδομένων με τον πίνακα όψεως (GridView). Στην ουσία
ξαναφορτώνουμε τα δεδομένα, για να ενημερωθεί ο πίνακας όψεως (GridView)
                // με τις αλλαγές.
                gvw_products.DataBind();
            }

            // Έλεγχος των αποθεματικών του επιλεγμένου προϊόντος, με
ενεργοποίηση του κουμπιού (Button) εισαγωγής προϊόντος στο καλάθι αγορών και
εμφάνισης
            // μηνύματος πληροφοριών αποθέματος επιλεγμένης εγγραφής προϊόντος
ανάλογα με την περίπτωση.
            uf_v_CartAddButton();
        }
```

```
        /// <summary>
        /// Συλλογή στοιχείων προϊόντων για την δημιουργία μιας νέας εγγραφής παραγγελίας στο καλάθι αγορών του χρήστη.
        /// </summary>
        private void uf_v_AddToCart()
        {
            // Η συλλογή στοιχείων του προϊόντος πραγματοποιείται μόνο όταν υπάρχει εγγραφή στην λίστα όψεως (ListView) πληροφοριών προϊόντος.
            if (lvw_product_info.Items.Count > 0)
            {
                // Λήψη του πρώτου στοιχείου λίστας όψεως (ListViewItem) από το οποίο θα συλλέξουμε τα περισσότερα στοιχεία του επιλεγμένου προϊόντος.
                ListViewItem lvwi_item = lvw_product_info.Items[0];

                // Λήψη της τιμής της ετικέτας (Label) κωδικού καταχώρησης προϊόντος από την λίστα όψεως (ListView) πληροφοριών προϊόντος και μετατροπή της σε
                // ακέραιο (integer).
                int i_product_id = CodeClass.uf_i_ConvertChildControlValueToInteger("IDLabel", lvwi_item);

                // Οι μεταβλητές αποθήκευσης του κωδικού αποθήκης, του κωδικού αποθέματος και της ποσότητας του διαθέσιμου αποθέματος ορισμένες με μηδενικές τιμές.
                int i_storage_id = 0;
                int i_stock_id = 0;
                int i_available_stock_quantity = 0;

                // Λήψη της επιλεγμένης γραμμής του πίνακα όψεως (GridView) αποθεμάτων.
                int i_stock_selected_index = gvw_stocks.SelectedIndex;

                // Στην περίπτωση που υπάρχει διαθέσιμο απόθεμα, ενημερώνουμε τον κωδικό αποθήκης, τον κωδικό αποθέματος και την ποσότητα του διαθέσιμου αποθέματος.
                if (i_stock_selected_index > -1)
                {
                    // Λήψη των τιμών των ετικετών (Label) κωδικού αποθήκης, κωδικού αποθέματος και της ποσότητας διαθέσιμου αποθέματος από την
                    // λίστα όψεως (ListView) πληροφοριών προϊόντος και μετατροπή τους σε ακέραιους (integer).
                    i_storage_id = CodeClass.uf_i_ConvertChildControlValueToInteger("StorageIDLabel", gvw_stocks.SelectedRow);
                    i_available_stock_quantity = CodeClass.uf_i_ConvertChildControlValueToInteger("QuantityLabel", gvw_stocks.SelectedRow);
                    i_stock_id = CodeClass.uf_i_ConvertChildControlValueToInteger("IDLabel", gvw_stocks.SelectedRow);
                }

                // Έλεγχος διαθεσιμότητας αποθέματος για το προϊόν της επιλεγμένης εγγραφής.
                bool b_stock_only = uf_b_ProductStockOnly();

                // Μεταβλητή αποθήκευσης της διαθεσιμότητας ενός προϊόντος και ορισμός της αρχικής της τιμής σε διαθέσιμο.
                CodeClass.availability cen_availability = CodeClass.availability.available;
```

```
                // Αν δεν υπάρχει διαθέσιμο απόθεμα στο προϊόν, τότε ελέγχουμε
αν μπορεί ο χρήστης να τα προπαραγγείλει.
                if (i_available_stock_quantity < 1)
                {
                    // Στην περίπτωση που το προϊόν είναι διαθέσιμο ακόμη κι
όταν δεν υπάρχει απόθεμα, τότε ο χρήστης μπορεί να το προπαραγγείλει.
                    if (b_stock_only == false)
                    {
                        cen_availability = CodeClass.availability.preorder;
                    }
                    else // Διαφορετικά, δεν υπάρχει δυνατότητα αγοράς του
προϊόντος και η διαδικασία τερματίζεται.
                    {
                        return;
                    }
                }

                // Λήψη των εγγραφών παραγγελίας στο καλάθι αγορών του χρήστη.
                CodeClass.order[] cdta_order = uf_cdta_ShoppingCart();

                // Ελέγχος σε κάθε παραγγελίας στο καλάθι αγορών του χρήστη για
όμοιες καταχωρήσεις.
                for (int i = 0; i < cdta_order.Length; i++)
                {
                    // Λήψη μιας παραγγελίας από το σύνολο των παραγγελιών του
χρήστη που υπάρχουν στο καλάθι αγορών του.
                    CodeClass.order cdt_order_check = cdta_order[i];

                    // Ακύρωση της καταχώρησης του προϊόντος καθώς υπάρχει ήδη
στο καλάθι με το συγκεκριμένο αποθεματικό
                    // (ένα προϊόν μπορεί να έχει περισσότερα από ένα
αποθεματικά π.χ. διαφορετικές αποθήκες).
                    if ((cdt_order_check.i_product_id == i_product_id) &&
(cdt_order_check.i_stock_id == i_stock_id))
                    {
                        return;
                    }
                }

                // Λήψη της τιμής της ετικέτας (Label) κωδικού καταχώρησης
προϊόντος από την λίστα όψεως (ListView) πληροφοριών προϊόντος και μετατροπή
της
                // σε ακέραιο (integer).
                int i_product_code =
CodeClass.uf_i_ConvertChildControlValueToInteger("CodeLabel", lvwi_item);

                // Λήψη της τιμής της ετικέτας (Label) ονόματος προϊόντος από
την λίστα όψεως (ListView) πληροφοριών προϊόντος και μετατροπή της
                // σε ακολουθία χαρακτήρων (string).
                string s_product_name =
CodeClass.uf_s_ConvertChildControlValueToString("NameLabel", lvwi_item,
"Άγνωστο");

                // Οι μεταβλητές αποθήκευσης ονόματος αποθήκης και ποσότητας
προϊόντος με εξ ορισμού τιμές.
                string s_storage_name = "Εργοστάσιο";
                int i_product_quantity = 1;

                // Λήψη της τιμής της ετικέτας (Label) ονόματος προϊόντος από
την λίστα όψεως (ListView) πληροφοριών προϊόντος και μετατροπή της
                // σε ακολουθία χαρακτήρων (string).
```

```
                string s_product_category_name =
CodeClass.uf_s_ConvertChildControlValueToString("ProductCategoryNameLabel",
lvwi_item, "Άγνωστη");

                // Μεταβλήτη αποθήκευσης του ονόματος προϊόντος και της
κατηγορίας αυτού, με σύνθεση των επί μέρους μεταβλητών τους.
                string s_product_name_with_category = s_product_name + " (" +
s_product_category_name + ")";

                // Ορισμός της μεταβλητής αποθήκευσης της τιμής του προϊόντος.
                decimal d_product_price;

                // Αν επιλεγεί απόθεμα, τότε λαμβάνουμε την τιμή του προϊόντος
καθώς και το όνομα της αποθήκης που περιέχει το συγκεκριμένο εμπόρευμα.
                if (i_stock_selected_index > -1)
                {
                    // Λήψη της επιλεγμένης εγγραφής αποθέματος από τον πίνακα
όψεως (GridView) αποθεμάτων προϊόντων.
                    GridViewRow gvwr_row = gvw_stocks.SelectedRow;

                    // Λήψη της τιμής της ετικέτας (Label) τιμής προϊόντος από
τον πίνακα όψεως (GridView) αποθεμάτων και μετατροπή της σε δεκαδικό (decimal).
                    d_product_price =
CodeClass.uf_d_ConvertChildControlValueToDecimal("PriceLabel", gvwr_row);

                    // Λήψη της τιμής της ετικέτας (Label) ονόματος αποθήκης
από τον πίνακα όψεως (GridView) αποθεμάτων και μετατροπή της σε ακολουθία
                    // χαρακτήρων (string).
                    s_storage_name =
CodeClass.uf_s_ConvertChildControlValueToString("StorageNameLabel", gvwr_row,
"Εργοστάσιο");
                }
                else // Διαφορετικά, λαμβάνουμε μόνο την τιμή του προϊόντος από
την λίστα όψεως (ListView) πληροφοριών προϊόντος. Αυτό σημαίνει ότι ο χρήστης
                {
// προπαραγγέλνει το συγκεκριμένο προϊόν.
                    // Λήψη της τιμής της ετικέτας (Label) τιμής προϊόντος από
την λίστα όψεως (ListView) πληροφοριών προϊόντος και μετατροπή της
                    // σε δεκαδικό (decimal).
                    d_product_price =
CodeClass.uf_d_ConvertChildControlValueToDecimal("PriceLabel", lvwi_item);
                }

                // Μεταβλητή αποθήκευσης του κωδικού προσωρινής εγγραφής στο
καλάθι αγορών και ορισμός της αρχικής της τιμή στη μονάδα.
                int i_id = 1;

                // Ελέγχουμε αν η συνεδρία (Session) που διατηρεί στην μνήμη
του διακομιστή (server) τον κωδικό προσωρινής εγγραφής στο καλάθι αγορών του
χρήστη
                // είναι ενεργή και στον σωστό τύπο δεδομένων. (Η κάθε συνεδρία
είναι μοναδική για κάθε χρήστη, ασχέτως αν έχουν το ίδιο όνομα κλήσης στον
κώδικα
                // και δεν είναι προσβάσιμες από τον χρήστη παρά μόνο από την
εφαρμογή). Ο κωδικός αυτός χρησιμοποιείται για την εσωτερική οργάνωση του
καλαθιού
                // αγορών και δεν καταχωρείται στην τελική παραγγελία.
                if ((Session["CartID"] != null) && (Session["CartID"] is int))
                {
                    i_id = (int)Session["CartID"];
                    i_id = i_id + 1;
                }
```

```
                // Προσθήκη στην συνεδρία (Session) που διατηρεί στην μνήμη του
διακομιστή (server) τον κωδικό προσωρινής εγγραφής στο καλάθι αγορών του χρήστη
                // την νέα τιμή του κωδικού.
                Session["CartID"] = i_id;

                // Σύνθεση της νεας εγγραφής παραγγελίας από τα στοιχεία που
συλλέξαμε παραπάνω.
                CodeClass.order cdt_order = new CodeClass.order(i_id,
i_product_code, i_product_id, s_product_name_with_category, i_storage_id,
s_storage_name,
                    i_product_quantity, d_product_price, i_stock_id,
d_product_price, cen_availability);

                // Προσθήκη στο καλάθι αγορών μιας νέας εγγραφής παραγγελίας.
                uf_v_CartAddProduct(cdt_order);
            }
        }

        /// <summary>
        /// Δημιουργία του πίνακα δεδομένων (DataTable) του καλαθιού αγορών και
σύνδεση των εγγραφών του με τον αντίστοιχο πίνακα όψεως (GridView).
        /// </summary>
        private void uf_v_CartProducts()
        {
            // Λήψη των εγγραφών παραγγελίας στο καλάθι αγορών του χρήστη.
            CodeClass.order[] cdta_order = uf_cdta_ShoppingCart();

            // Δημιουργούμε τον πίνακα δεδομένων (DataTable) του καλαθιού
αγορών μόνο όταν ο χρήστης έχει παραγγείλει προϊόντα.
            if (cdta_order.Length > 0)
            {
                // Δημιουργία του πίνακα δεδομένων (DataTable) στον οποίο θα
τοποθετήσουμε τις παραγγελίες του καλαθιού αγορών.
                DataTable dt_cart = new DataTable();

                // Δημιουργία των στηλών (columns) του πίνακα δεδομένων
(DataTable) με τα πεδία των δεδομένων που θα καταχωρήσουμε στη συνέχεια.
                dt_cart.Columns.AddRange(new DataColumn[11] { new
DataColumn("ID", typeof(int)), new DataColumn("Code", typeof(int)),
                    new DataColumn("ProductID", typeof(int)), new
DataColumn("ProductName", typeof(string)), new DataColumn("StorageID",
typeof(int)),
                    new DataColumn("StorageName", typeof(string)), new
DataColumn("ProductQuantity", typeof(int)), new DataColumn("ProductPrice",
typeof(decimal)),
                    new DataColumn("StockID", typeof(int)), new
DataColumn("ProductTotalPrice", typeof(decimal)),
                    new DataColumn("ProductAvailability", typeof(string)) });

                // Για κάθε παραγγελία στο καλάθι αγορών του χρήστη,
δημιουργούμε μια εγγραφή στον πίνακα δεδομένων (DataTable) που θα
χρησιμοποιήσουμε για να
                // συσχετίσουμε τα δεδομένα με την πηγή δεδομένων (DataSource)
του καλαθιού αγορών.
                for (int i = 0; i < cdta_order.Length; i++)
                {
                    // Λήψη μιας παραγγελίας από το σύνολο των παραγγελιών του
χρήστη που υπάρχουν στο καλάθι αγορών του.
                    CodeClass.order cdt_order = cdta_order[i];
```

```
                    // Μετάφραση της απαρίθμησης δεδομένων (enumeration) που
δείχνει την διαθεσιμότητα κάθε εγγραφής παραγγελίας σε ακολουθία χαρακτήρων
(string)
                    // ελληνικών.
                    string s_availability =
CodeClass.uf_s_AvailabilityEnumToString(cdt_order.cen_availability);

                    // Προσθήκη της νέας εγγραφής στον πίνακα δεδομένων
(DataTable) του καλαθιού αγορών.
                    dt_cart.Rows.Add(cdt_order.i_id, cdt_order.i_product_code,
cdt_order.i_product_id, cdt_order.s_product_name, cdt_order.i_storage_id,
                        cdt_order.s_storage_name, cdt_order.i_product_quantity,
cdt_order.d_product_price, cdt_order.i_stock_id,
cdt_order.d_product_total_price,
                        s_availability);
                }

                // Σύνδεση της πηγής δεδομένων (DataSource) του καλαθιού αγορών
με τον πίνακα δεδομένων (DataTable) που δημιουργήσαμε πιο πάνω.
                gvw_cart.DataSource = dt_cart;
            }

            // Σύνδεση δεδομένων με τον πίνακα όψεως (GridView). Στην ουσία
ξαναφορτώνουμε τα δεδομένα, για να ενημερωθεί ο πίνακας όψεως (GridView)
            // με τις αλλαγές.
            gvw_cart.DataBind();
        }

        /// <summary>
        /// Προσθήκη στο καλάθι αγορών μιας νέας εγγραφής παραγγελίας.
        /// </summary>
        /// <param name="cdt_order">Η εγγραφή παραγγελίας που θα προσθέσουμε
στο καλάθι αγορών.</param>
        private void uf_v_CartAddProduct(CodeClass.order cdt_order)
        {
            // Αν η παραγγελία δεν έχει κωδικό εγγραφής, τότε δεν είναι έγκυρη
και τερματίζουμε την διαδικασία.
            if (cdt_order.i_id < 1)
            {
                return;
            }

            // Λήψη των εγγραφών παραγγελίας στο καλάθι αγορών του χρήστη.
            CodeClass.order[] cdta_order = uf_cdta_ShoppingCart();

            // Ελέγχουμε αν υπάρχουν εγγραφές στο καλάθι αγορών.
            if (cdta_order.Length > 0)
            {
                // Λήψη του συνολικού πλήθους των εγγραφών του καλαθιού αγορών.
                int i_deck_length = cdta_order.Length;

                // Αυξάνουμε την χωρητικότητα του καλαθιού αγορών για να
χωρέσει η νέα εγγραφή.
                Array.Resize(ref cdta_order, i_deck_length + 1);

                // Προσθέτουμε την νέα εγγραφή στο καλάθι αγορών.
                cdta_order[i_deck_length] = cdt_order;
            }
            else // Διαφορετικά, δημιουργούμε το καλάθι αγορών και προσθέτουμε
την πρώτη εγγραφή.
            {
                cdta_order = new CodeClass.order[1] { cdt_order };
```

```
            }

            // Προσθήκη στην συνεδρία (Session) που διατηρεί στην μνήμη του
διακομιστή (server) το καλάθι αγορών του χρήστη το νέο σύνολο των παραγγελιών
            // ύστερα από την προσθήκη της νέας εγγραφής.
            Session["Cart"] = cdta_order;

            // Δημιουργία του πίνακα δεδομένων (DataTable) του καλαθιού αγορών
και σύνδεση των εγγραφών του με τον αντίστοιχο πίνακα όψεως (GridView).
            uf_v_CartProducts();
        }

        /// <summary>
        /// Ενημέρωση της συγκεκριμένης εγγραφής παραγγελίας στο καλάθι αγορών.
        /// </summary>
        /// <param name="cdt_order">Η εγγραφή παραγγελίας που θα μεταβάλουμε
στο καλάθι αγορών.</param>
        private void uf_v_CartUpdateProduct(CodeClass.order cdt_order)
        {
            // Αν η παραγγελία δεν έχει κωδικό εγγραφής, τότε δεν είναι έγκυρη
και τερματίζουμε την διαδικασία.
            if (cdt_order.i_id < 1)
            {
                return;
            }

            // Λήψη των εγγραφών παραγγελίας στο καλάθι αγορών του χρήστη.
            CodeClass.order[] cdta_order = uf_cdta_ShoppingCart();

            // Ελέγχος σε κάθε παραγγελίας στο καλάθι αγορών του χρήστη για
εύρεση συγκεκριμένης εγγραφής, της οποίας θα μεταβάλουμε τα στοιχεία.
            for (int i = 0; i < cdta_order.Length; i++)
            {
                // Λήψη μιας παραγγελίας από το σύνολο των παραγγελιών του
χρήστη που υπάρχουν στο καλάθι αγορών του.
                CodeClass.order cdt_order_check = cdta_order[i];

                // Ψάχνουμε να βρούμε την εγγραφή που έχει τον ίδιο κωδικό
εγγραφής.
                if (cdt_order.i_id == cdt_order_check.i_id)
                {
                    // Μεταφορά των εγγραφών του πίνακα στοιχείων (array) σε
έναν δυναμικό πίνακα (ArrayList). Ο δυναμικός πίνακας θα μας βοηθήσει να
                    // διαχειριστούμε τις εγγραφές πιο εύκολα μέσω του
εσωτερικού ευρετηρίου (index).
                    ArrayList al_order = new ArrayList(cdta_order);

                    // Προσθέτουμε την ενημερωμένη εγγραφή σε μια ενδιάμεση
θέση του πίνακα, αλλά ακριβώς μετά την θέση της ήδη υπάρχουσας.
                    al_order.Insert(i + 1, cdt_order);

                    // Αφού προσθέσουμε την νέα εγγραφή, αφαιρούμε την παλιά
για να γίνει η ενημέρωση της. Στην ουσία είναι σαν να ανανεώνεται η ίδια
εγγραφή.
                    al_order.RemoveAt(i);

                    // Μεταφορά των στοιχείων του δυναμικού πίνακα (ArrayList)
στον πίνακα στοιχείων (array), αφού δεν τον χρειαζόμαστε πλεον.
                    cdta_order =
(CodeClass.order[])al_order.ToArray(typeof(CodeClass.order));
```

```
                    // Προσθήκη στην συνεδρία (Session) που διατηρεί στην μνήμη
του διακομιστή (server) το καλάθι αγορών του χρήστη το νέο σύνολο των
παραγγελιών
                    // ύστερα από την μεταβολή της επιλεγμένης εγγραφής.
                    Session["Cart"] = cdta_order;

                    // Δημιουργία του πίνακα δεδομένων (DataTable) του καλαθιού
αγορών και σύνδεση των εγγραφών του με τον αντίστοιχο πίνακα όψεως (GridView).
                    uf_v_CartProducts();
                    break; // Σταματάμε την επαναλαμβανόμενη διαδικασία αφού
βρήκαμε την εγγραφή που χρειαζόμασταν.
                }
            }
        }

        /// <summary>
        /// Διαγραφή της συγκεκριμένης εγγραφής παραγγελίας από το καλάθι
αγορών.
        /// </summary>
        /// <param name="cdt_order">Η εγγραφή παραγγελίας που θα αφαιρέσουμε
απο το καλάθι αγορών.</param>
        private void uf_v_CartRemoveProduct(CodeClass.order cdt_order)
        {
            // Αν η παραγγελία δεν έχει κωδικό εγγραφής, τότε δεν είναι έγκυρη
και τερματίζουμε την διαδικασία.
            if (cdt_order.i_id < 1)
            {
                return;
            }

            // Λήψη των εγγραφών παραγγελίας στο καλάθι αγορών του χρήστη.
            CodeClass.order[] cdta_order = uf_cdta_ShoppingCart();

            // Ελέγχος σε κάθε παραγγελίας στο καλάθι αγορών του χρήστη για
εύρεση συγκεκριμένης εγγραφής, την οποίαν θα διαγράψουμε.
            for (int i = 0; i < cdta_order.Length; i++)
            {
                // Λήψη μιας παραγγελίας από το σύνολο των παραγγελιών του
χρήστη που υπάρχουν στο καλάθι αγορών του.
                CodeClass.order cdt_order_check = cdta_order[i];

                // Ψάχνουμε να βρούμε την εγγραφή που έχει τον ίδιο κωδικό
εγγραφής.
                if (cdt_order.i_id == cdt_order_check.i_id)
                {
                    // Μεταφορά των εγγραφών του πίνακα στοιχείων (array) σε
έναν δυναμικό πίνακα (ArrayList). Ο δυναμικός πίνακας θα μας βοηθήσει να
                    // διαχειριστούμε τις εγγραφές πιο εύκολα μέσω του
εσωτερικού ευρετηρίου (index).
                    ArrayList al_order = new ArrayList(cdta_order);

                    // Αφαιρούμε την εγγραφή που μας ζητήθηκε.
                    al_order.Remove(cdt_order);

                    // Μεταφορά των στοιχείων του δυναμικού πίνακα (ArrayList)
στον πίνακα στοιχείων (array), αφού δεν τον χρειαζόμαστε πλεον.
                    cdta_order =
(CodeClass.order[])al_order.ToArray(typeof(CodeClass.order));

                    // Προσθήκη στην συνεδρία (Session) που διατηρεί στην μνήμη
του διακομιστή (server) το καλάθι αγορών του χρήστη το νέο σύνολο των
παραγγελιών
```

```
                    // ύστερα από την διαγραφή της επιλεγμένης εγγραφής.
                    Session["Cart"] = cdta_order;

                    // Δημιουργία του πίνακα δεδομένων (DataTable) του καλαθιού
αγορών και σύνδεση των εγγραφών του με τον αντίστοιχο πίνακα όψεως (GridView).
                    uf_v_CartProducts();
                    break; // Σταματάμε την επαναλαμβανόμενη διαδικασία αφού
βρήκαμε την εγγραφή που χρειαζόμασταν.
                }
            }
        }

        protected void btn_search_Click(object sender, EventArgs e)
        {
            // Συλλογή από την βάση δεδομένων των προϊόντων που πληρούν τα
κριτήρια που έθεσε ο χρήστης.
            uf_v_Search();
        }

        protected void gvw_products_PageIndexChanged(object sender, EventArgs
e)
        {
            // Συλλογή από την βάση δεδομένων των προϊόντων που πληρούν τα
κριτήρια που έθεσε ο χρήστης.
            uf_v_Search();
        }

        protected void gvw_products_Sorted(object sender, EventArgs e)
        {
            // Συλλογή από την βάση δεδομένων των προϊόντων που πληρούν τα
κριτήρια που έθεσε ο χρήστης.
            uf_v_Search();
        }

        protected void gvw_products_SelectedIndexChanged(object sender,
EventArgs e)
        {
            // Λήψη της επιλεγμένης γραμμής του πίνακα όψεως (GridView)
προϊόντων.
            int i_selected_index = gvw_products.SelectedIndex;

            // Επαναφόρτωση της παρούσας σελίδας με μεταβλητές φυλλομετρητή
(Query String) μόνο αν υπάρχει επιλογή προϊόντος στον πίνακα όψεως (GridView).
            if ((i_selected_index > -1) &&
(gvw_products.DataKeys[i_selected_index].Values.Count > 0))
            {
                // Λήψη του κωδικού εγγραφής του επίλεγμένου προϊόντος από τον
πίνακα όψεως (GridView).
                string s_id =
gvw_products.DataKeys[i_selected_index].Values[0].ToString();

                // Λήψη της φυσικής διαδρομής του αρχείου των προϊόντων.
                string s_current_file = Path.GetFileName(Request.PhysicalPath);

                // Επαναφόρτωση της παρούσας σελίδας με μεταβλητές φυλλομετρητή
(Query String), μόνο αν η φυσική διαδρομή του αρχείου και ο κωδικός εγγραφής
του
                // προϊόντος είναι έγκυρα.
                if ((s_id != null) && (s_id.Trim() != "") && (s_current_file !=
null) && (s_current_file.Trim() != ""))
                {
```

```
                    // Δημιουργία μεταβλητής φυλλομετρητή για τον κωδικό
εγγραφής του προϊόντος.
                    s_current_file = s_current_file + "?id=" + s_id;

                    // Λήψη του επιλεγμένου κριτηρίου και της επιλεγμένης
κατεύθυνσης ταξινόμησης των εγγραφών του πίνακα όψεως (GridView).
                    string s_sort_expression = gvw_products.SortExpression;
                    SortDirection sdir_sort_direction =
gvw_products.SortDirection;

                    // Αν ο χρήστης έχει επιλέξει ταξινόμηση στον πίνακα όψεως
(GridView) προϊόντων, τότε δημιουργούμε δύο μεταβλητές φυλλομετρητή (Query
String)
                    // για να τις αποθηκεύσουμε (κριτήριο και κατεύθυνση
ταξινόμησης).
                    if ((s_sort_expression != null) &&
(s_sort_expression.Trim() != ""))
                    {
                        s_current_file = s_current_file + "&sort=" +
s_sort_expression + "&direction=" + sdir_sort_direction.ToString();
                    }

                    // Λήψη της επιλεγμένης σελίδας του πίνακα όψεως
(GridView).
                    int i_page = gvw_products.PageIndex;

                    // Αν ο χρήστης έχει επιλέξει σελίδα στον πίνακα όψεως
(GridView) προϊόντων, τότε δημιουργούμε μια μεταβλητή φυλλομετρητή (Query
String)
                    // για να την αποθηκεύσουμε.
                    if (i_page > -1)
                    {
                        s_current_file = s_current_file + "&page=" +
i_page.ToString();
                    }

                    // Αν ο χρήστης έχει επιλέξει γραμμή στον πίνακα όψεως
(GridView) προϊόντων, τότε δημιουργούμε μια μεταβλητή φυλλομετρητή (Query
String)
                    // για να την αποθηκεύσουμε.
                    if (i_selected_index > -1)
                    {
                        s_current_file = s_current_file + "&index=" +
i_selected_index.ToString();
                    }

                    // Λήψη των κριτηρίων αναζήτησης από το κουτί κειμένου
(TextBox) της αναζήτησης.
                    string s_search = tbx_search.Text;

                    // Αν ο χρήστης έχει γράψει κριτήρια αναζήτησης στο κουτί
κειμένου (TextBox) αναζήτησης, τότε δημιουργούμε μια μεταβλητή φυλλομετρητή
                    // (Query String) για να τα αποθηκεύσουμε.
                    if (s_search.Trim() != "")
                    {
                        s_current_file = s_current_file + "&search=" +
s_search;
                    }

                    // Επαναφόρτωση της παρούσα σελίδας σε μια νέα σελίδα που
θα περιέχει τις μεταβλητές φυλλομετρητή (Query String) που ορίσαμε
προηγουμένως.
```

```
                    // Η σελίδα αυτή θα επιτρέπει στους χρήστες να
κατευθύνονται απευθείας σε ένα συγκεκριμένο προϊόν με τη χρήση ενός συνδέσμου.
                    Response.Redirect(s_current_file, true);
                }
            }

            // Έλεγχος των αποθεματικών του επιλεγμένου προϊόντος, με
ενεργοποίηση του κουμπιού (Button) εισαγωγής προϊόντος στο καλάθι αγορών και
εμφάνισης
            // μηνύματος πληροφοριών αποθέματος επιλεγμένης εγγραφής προϊόντος
ανάλογα με την περίπτωση.
            uf_v_CartAddButton();
        }

        /// <summary>
        /// Έλεγχος των αποθεματικών του επιλεγμένου προϊόντος, με ενεργοποίηση
του κουμπιού (Button) εισαγωγής προϊόντος στο καλάθι αγορών και εμφάνισης
        /// μηνύματος πληροφοριών αποθέματος επιλεγμένης εγγραφής προϊόντος
ανάλογα με την περίπτωση.
        /// </summary>
        private void uf_v_CartAddButton()
        {
            // Σύνδεση δεδομένων με τον πίνακα όψεως (GridView) και την λίστα
όψεως (ListView). Στην ουσία ξαναφορτώνουμε τα δεδομένα, για να ενημερωθούν ο
πίνακας
            // όψεως (GridView) και η λίστα όψεως (ListView) με τις αλλαγές.
            lvw_product_info.DataBind();
            gvw_stocks.DataBind();

            // Δημιουργούμε τις μεταβλητές αποθήκευσης μήνυμα πληροφοριών
αποθέματος επιλεγμένης εγγραφής προϊόντος και της κατάστασης κουμπιού (Button)
εισαγωγής
            // προϊόντος στο καλάθι αγορών. Στη συνέχεια τους δίνουμε την
απενεργοποιημένη τιμή, δηλαδή την ψευδή (false) κατάσταση.
            bool b_visible_button = false;
            bool b_visible_message = false;

            // Ορίζουμε την μεταβλητής αποθήκευσης μηνύματος πληροφοριών
αποθέματος επιλεγμένης εγγραφής προϊόντος με μια εξ ορισμού τιμή.
            string s_message = "Αγορά";

            // Συλλογή των στοιχείων λογαριασμού του χρήστη που είναι
συνδεδεμένος στο σύστημα.
            MembershipUser msu_user =
Membership.GetUser(HttpContext.Current.User.Identity.Name);

            // Δημιουργία νέου μηνύματος πληροφοριών αποθέματος επιλεγμένης
εγγραφής προϊόντος και ενεργοποίηση του κουμπιού (Button) εισαγωγής προϊόντος
στο
            // καλάθι αγορών όταν υπάρχει συνδεδεμένος χρήστης.
            if (msu_user != null)
            {
                // Λήψη της μεταβλητής φυλλομετρητή (Query String) που διατηρεί
την τιμή του κωδικού εγγραφής του προϊόντος του πίνακα όψεως (GridView)
                // των προϊόντων (Products) από την αίτηση ιστού (Web Request).
                int i_id = CodeClass.uf_i_QueryStringToInteger(Request, "id");

                // Έλεγχος αν υπάρχει επιλεγμένο προϊόν για να ελέγξουμε τα
αποθεματικά του.
                if ((gvw_products.SelectedIndex > -1) || (i_id > 0))
                {
```

```
                    // Έλεγχος διαθεσιμότητας αποθέματος για το προϊόν της
επιλεγμένης εγγραφής.
                    bool b_stock_only = uf_b_ProductStockOnly();

                    // Λήψη της επιλεγμένης γραμμής του πίνακα όψεως (GridView)
αποθεμάτων.
                    int i_gridview_selected_index = gvw_stocks.SelectedIndex;

                    // Μεταβλητή αποθήκευσης του διαθέσιμου αποθέματος για το
επιλεγμένο προϊόν.
                    int i_quantity = 0;

                    // Αν υπάρχει επιλεγμένο απόθεμα, τότε τοποθέτησε την
ποσότητα του στην αντίστοιχη μεταβλητή.
                    if (i_gridview_selected_index > -1)
                    {
                        // Λήψη της τιμής της ετικέτας (Label) κωδικού
καταχώρησης προϊόντος από τον πίνακα όψεως (GridView) αποθεμάτων και μετατροπή
της σε
                        // ακέραιο (integer).
                        i_quantity =
CodeClass.uf_i_ConvertChildControlValueToInteger("QuantityLabel",
gvw_stocks.SelectedRow);
                    }

                    // Ενεργοποιούμε το μήνυμα πληροφοριών αποθέματος
επιλεγμένης εγγραφής προϊόντος.
                    b_visible_message = true;

                    // Ελέγχουμε αν το προϊόν είναι διαθέσιμο μόνο όταν υπάρχει
απόθεμα.
                    if (b_stock_only == true)
                    {
                        // Ελέγχουμε αν υπάρχει απόθεμα στο προϊόν.
                        if (gvw_stocks.Rows.Count > 0)
                        {
                            // Ελέγχουμε αν ο χρήστης έχει διαλέξει ένα από τα
διαθέσιμα αποθέματα.
                            if (i_gridview_selected_index > -1)
                            {
                                // Ελέγχουμε αν υπάρχουν προϊόντα διαθέσιμα
στις αποθήκες.
                                if (i_quantity > 0)
                                {
                                    // Ενεργοποιούμε το κουμπί (Button)
εισαγωγής προϊόντος στο καλάθι αγορών, καθώς ο χρήστης μπορεί μόνο να
παραγγείλει
                                    // από το διαθέσιμο απόθεμα (Stock).
                                    b_visible_button = true;
                                    s_message = "Διαθέσιμο μέχρι εξάντλησης
αποθέματος.";
                                }
                                else // Διαφορετικά, δεν υπάρχει διαθέσιμο
απόθεμα για να παραγγείλει ο χρήστης.
                                {
                                    // Πληροφορούμε τον χρήστη ότι δεν μπορεί
να παραγγείλει καθώς το διαθέσιμο απόθεμα έχει εξαντληθεί (Stock out).
                                    s_message = "Το απόθεμα εξαντλήθηκε.";
                                }
                            }
                            else // Διαφορετικά, ο χρήστης μπορεί να
παραγγείλει μόνο από το διαθέσιμο απόθεμα.
```

```
                            {
                                s_message = "Επιλέξτε πρώτα απόθεμα.";
                            }
                        }
                        else // Διαφορετικά, δεν υπάρχει απόθεμα στο προϊόν.
                        {
                            // Πληροφορούμε τον χρήστη ότι δεν μπορεί να
παραγγείλει καθώς το διαθέσιμο απόθεμα έχει εξαντληθεί (Stock out).
                            s_message = "Το απόθεμα εξαντλήθηκε.";
                        }
                    }
                    else // Διαφορετικα, ο χρήστης μπορεί να το παραγγείλει
ακόμη κι όταν το απόθεμα εξαντληθεί (συγκεκριμένα προπαραγγείλει).
                    {
                        // Ελέγχουμε αν υπάρχει απόθεμα στο προϊόν.
                        if (gvw_stocks.Rows.Count > 0)
                        {
                            // Ελέγχουμε αν ο χρήστης έχει διαλέξει ένα από τα
διαθέσιμα αποθέματα.
                            if (i_gridview_selected_index > -1)
                            {
                                // Ελέγχουμε αν υπάρχουν προϊόντα διαθέσιμα
στις αποθήκες.
                                if (i_quantity > 0)
                                {
                                    // Πληροφορούμε τον χρήστη ότι μπορεί να
παραγγείλει από το διαθέσιμο απόθεμα (Available).
                                    s_message = "Αγορά από διαθέσιμο απόθεμα.";
                                }
                                else // Διαφορετικά, δεν υπάρχει διαθέσιμο
απόθεμα αλλά παρέχεται η δυνατότητα στο χρήστη να προπαραγγείλει.
                                {
                                    // Πληροφορούμε τον χρήστη ότι μπορεί να
προπαραγγείλει καθώς το διαθέσιμο απόθεμα έχει εξαντληθεί (Preorder).
                                    s_message = "Μη διαθέσιμο απόθεμα, αλλά με
δυνατότητα προπαραγγελίας.";
                                }
                            }
                            else // Διαφορετικά, ο χρήστης μπορεί να
προπαραγγείλει ή να παραγγείλει από το διαθέσιμο απόθεμα.
                            {
                                // Πληροφορούμε τον χρήστη ότι μπορεί να
προπαραγγείλει ή να παραγγείλει από το διαθέσιμο απόθεμα.
                                s_message = "Επιλέξτε απόθεμα ή
προπαραγγείλετε.";
                            }

                            // Ενεργοποιούμε το κουμπί (Button) εισαγωγής
προϊόντος στο καλάθι αγορών, καθώς ο χρήστης μπορεί να προπαραγγείλει ή
                            // να παραγγείλει από το διαθέσιμο απόθεμα.
                            b_visible_button = true;
                        }
                        else // Διαφορετικά, δεν υπάρχει απόθεμα στο προϊόν.
                        {
                            // Ενεργοποιούμε το κουμπί (Button) εισαγωγής
προϊόντος στο καλάθι αγορών, καθώς ο χρήστης μπορεί μόνο να προπαραγγείλει.
                            b_visible_button = true;
                            s_message = "Μη διαθέσιμο απόθεμα, αλλά με
δυνατότητα προπαραγγελίας.";
                        }
                    }
                }
```

```
            }
            else // Διαφορετικά, εμφανίζουμε μήνυμα στον ανώνυμο χρήστη
(anonymous) να συνδεθεί με το λογαριασμό του.
            {
                // Ενεργοποιούμε το μήνυμα πληροφοριών αποθέματος επιλεγμένης
εγγραφής προϊόντος και θέτουμε την τιμή του. Συγκεκριμένα
                // προτρέπουμε τον χρήστη να συνδεθεί με τον λογαριασμό του για
να πραγματοποιήσει μια παραγγελία.
                b_visible_message = true;
                s_message = "Για παραγγελίες, παρακαλούμε συνδεδείτε με τον
λογαριασμό σας.";
            }

            // Ενεργοποιούμε την ετικέτα (Label) μηνύματος πληροφοριών
αποθέματος επιλεγμένης εγγραφής προϊόντος και το κουμπί (Button) εισαγωγής
προϊόντος στο
            // καλάθι αγορών σύμφωνα με τις καταστάσεις των αντίστοιχων
μεταβλητών.
            CodeClass.uf_v_EnableWebControl(btn_add_to_cart, b_visible_button);
            CodeClass.uf_v_EnableWebControl(lbl_message, b_visible_message);

            // Εμφάνιση του μηνύματος στο χρήστη.
            lbl_message.Text = s_message;
        }

        /// <summary>
        /// Συλλογή από την βάση δεδομένων των προϊόντων που πληρούν τα
κριτήρια που έθεσε ο χρήστης.
        /// </summary>
        private void uf_v_Search()
        {
            // Πραγματοποίηση της διαδικασίας, μόνο όταν υπάρχει αξιοποιήσιμη
τιμή στο κουτί κειμένου () αναζήτησης.
            if (tbx_search.Text.Trim() != "")
            {
                // Λήψη των κριτηρίων αναζήτησης προϊόντων που έθεσε ο χρήστης
μέσω του κουτιού κειμένου (TextBox) αναζήτησης.
                string s_search = tbx_search.Text;

                // Διορθώνει ένα πρόβλημα με το ελληνικό κόμμα στους δεκαδικούς
κατά την χρήση σε εντολές sql.
                if (s_search.Contains(",") == true)
                {
                    // Μετατροπή της τιμής αναζήτησης σε δεκαδικό αριθμό
(decimal).
                    decimal d_search =
CodeClass.uf_d_ConvertValueToDecimal(s_search);

                    // Αν υπάρχει δεκαδική τιμή, τότε την επιλέγουμε με κόμμα
ανεξαρτήτου κουλτούρας (culture), δηλαδή ούτε ελληνικής,
                    // ούτε αγγλικής, ούτε άλλης γλώσσας.
                    if (d_search != 0)
                    {
                        s_search =
d_search.ToString(System.Globalization.CultureInfo.InvariantCulture);
                    }
                }

                // Λήψη της εντολής sql επιλογής εγγραφών προϊόντων.
                string s_select_command = sqlds_products.SelectCommand;
```

```
                // Έλεγχος αν υπάρχουν ήδη κριτήρια στην sql επιλογής εγγραφών
προϊόντων. Αν υπάρχουν, τότε δεν υπάρχει λόγος να τα ξαναορίσουμε.
                if (s_select_command.Contains("WHERE") == false)
                {
                    // Το κομμάτι της νέας εντολής sql που θα επιστρέψει τα
προϊόντα που πληρούν τα κριτήρια αναζήτησης.
                    string s_where_command_part = " WHERE Name LIKE '%" +
s_search + "%'" +
                        " OR Description LIKE '%" + s_search + "%'" +
                        " OR CAST(Code AS VARCHAR) LIKE '%" + s_search + "%'" +
                        " OR PricingUnitName LIKE '%" + s_search + "%'" +
                        " OR CAST(MinPrice AS VARCHAR) LIKE '%" + s_search +
"%'" +
                        " OR CAST(Price AS VARCHAR) LIKE '%" + s_search + "%'"
+
                        " OR ProductCategoryName LIKE '%" + s_search + "%' " +
                        " OR Availability LIKE '%" + s_search + "%'";

                    // Τροποποίηση της υπάρχουσας εντολής sql με το επιπλέον
κομμάτι για να επιλεγούν μόνο τα προϊόντα που πληρούν τα κριτήρια που έθεσε ο
χρήστης.
                    s_select_command = s_select_command + s_where_command_part;

                    // Ενημέρωση της εντολής sql επιλογής εγγραφών προϊόντων με
τη νέα μορφή.
                    sqlds_products.SelectCommand = s_select_command;
                }
            }

            // Σύνδεση δεδομένων με τον πίνακα όψεως (GridView). Στην ουσία
ξαναφορτώνουμε τα δεδομένα, για να ενημερωθεί ο πίνακας όψεως (GridView)
            // με τις αλλαγές.
            gvw_products.DataBind();

            // Καθαρισμός της επιλεγμένης γραμμής στον πίνακα όψεως (GridView)
προϊόντων.
            gvw_products.SelectedIndex = -1;
        }

        protected void gvw_cart_RowEditing(object sender, GridViewEditEventArgs
e)
        {
            // Έλεγχος διαθεσιμότητας αποθέματος για το προϊόν της επιλεγμένης
εγγραφής.
            bool b_stock_only = uf_b_ProductStockOnly();

            // Ενεργοποιούμε το πλαίσιο ομαδοποίησης στοιχείων (Div) με την
λίστα κουμπιών μοναδικής επιλογής (RadioButtonList) του καλαθιού αγορών όταν
υπάρχει
            // απόθεμα αλλα και η δυνατότητα προπαραγγελίας.
            if ((gvw_stocks.Rows.Count > 0) && (b_stock_only == false))
            {
                // Ορίζουμε την επιλογή στην λίστα κουμπιών μοναδικής επιλογής
(RadioButtonList) στο "μέγιστο απόθεμα"
                rbtnl_cart.Items[0].Selected = true;
                rbtnl_cart.Items[1].Selected = false;

                // Εμφανίζουμε το πλαίσιο ομαδοποίησης στοιχείων (Div) που
περιέχει την λίστα κουμπιών μοναδικής επιλογής (RadioButtonList) του καλαθιού
αγορών.
                div_cart_choice.Visible = true;
            }
```

```
            // Δημιουργία του πίνακα δεδομένων (DataTable) του καλαθιού αγορών
και σύνδεση των εγγραφών του με τον αντίστοιχο πίνακα όψεως (GridView).
            uf_v_CartProducts();
            gvw_cart.EditIndex = e.NewEditIndex;

            // Ορίζουμε το χρώμα του φόντου του κελιού της φόρμας αλλαγών του
καλαθιού αγορών στο ίδιο χρώμα με το φόντο του πλασίου αλλαγής στοιχείων.
            // Χρησιμοποιούμε αυτό το τέχνασμα για να δείξουμε στο χρήστη ποια
πεδία είναι ανοιχτά προς μεταβολή.
            gvw_cart.Rows[e.NewEditIndex].Cells[1].BackColor =
gvw_cart.EditRowStyle.BackColor;

            // Σύνδεση δεδομένων με τον πίνακα όψεως (GridView). Στην ουσία
ξαναφορτώνουμε τα δεδομένα, για να ενημερωθεί ο πίνακας όψεως (GridView)
            // με τις αλλαγές.
            gvw_cart.DataBind();
        }

        /// <summary>
        /// Έλεγχος διαθεσιμότητας αποθέματος για το προϊόν της επιλεγμένης
εγγραφής.
        /// </summary>
        /// <returns>Την διαθεσιμότητα αποθέματος του προϊόντος.</returns>
        private bool uf_b_ProductStockOnly()
        {
            // Μεταβλητή αποθήκευσης διαθεσιμότητας προϊόντος με οξ ορισμού
τιμή την διαθεσιμότητα ακόμη και χωρίς απόθεμα.
            bool b_stock_only = false;

            // Έλεγχος αν μέσα στην λίστα όψεως (ListView) των πληροφοριών
προϊόντων, υπάρχει το κουτί επιλογής (CheckBox) της κατάστασης αποθέματος του
            // επιλεγμένου προϊόντος της παραγγελίας.
            if ((lvw_product_info.Items.Count > 0) &&
(lvw_product_info.Items[0].FindControl("StockOnlyCheckBox") is CheckBox))
            {
                // Λήψη του κουτιού επιλογής (CheckBox) της κατάστασης
αποθέματος του επιλεγμένου προϊόντος της παραγγελίας (ακολουθείται αυτή η
διαδικασία
                // καθώς δεν είναι εφικτή η απευθείας αλληλεπίδραση, καθώς
εμπεριέχεται στο σύνολο των αντικειμένων (child objects) της φόρμας ενός άλλου
                // αντικειμένου (parent object)).
                CheckBox cbx_stock_only =
(CheckBox)lvw_product_info.Items[0].FindControl("StockOnlyCheckBox");

                // Αν το κουτί επιλογής (CheckBox) είναι επιλεγμένο, τότε το
προϊόν είναι διαθέσιμο μόνο όταν είναι σε απόθεμα.
                if (cbx_stock_only.Checked == true)
                {
                    b_stock_only = true;
                }
            }

            // Επιστρέφει την διαθεσιμότητα αποθέματος του προϊόντος.
            return b_stock_only;
        }

        protected void gvw_cart_RowUpdating(object sender,
GridViewUpdateEventArgs e)
        {
            int i_row_cells_count = e.NewValues.Count; // Ο μέγιστος αριθμός
κελιών σε μια γραμμή πίνακα όψεως (Gridview).
```

```
            int i_predefined_max_cell = 11; // Ορίζει το μέγιστο αριθμό κελιών
σε μια γραμμή. Λειτουργεί σαν προστασία από σφάλματα κελιών εκτός οριών πίνακα.

            // Προστασία από σφάλματα κελιών εκτός ορίων, σαν πρόσθετο μέτρο σε
περιπτώσεις ανανέωση κώδικα ή απρόβλεπτων καταστάσεων.
            if ((i_row_cells_count < i_predefined_max_cell) ||
(rbtnl_cart.Items.Count < 2))
            {
                return;
            }

            // Μετατροπή της τιμής του πεδίου της ποσότητας προϊόντος από το
καλάθι αγορών σε ακέραιο αριθμό (integer).
            int i_product_quantity =
CodeClass.uf_i_ConvertValueToInteger(e.NewValues[6]);

            // Η μεταβλητή αποθήκευσης του ονόματος αποθήκης με εξ ορισμού
τιμή.
            string s_storage_name = "Εργοστάσιο";

            // Λήψη της τιμής του πεδίου διαθεσιμότητας από τον πίνακα όψεως
(GridView) του καλαθιού και μετάφραση της ακολουθίας χαρακτήρων (string)
ελληνικών σε
            // απαρίθμηση δεδομένων (enumeration) που δείχνει την διαθεσιμότητα
κάθε εγγραφής παραγγελίας.
            CodeClass.availability cen_availability =
CodeClass.uf_cen_AvailabilityStringToEnum(e.NewValues[10].ToString());

            // Αν η ποσότητα του προϊόντος είναι μικρότερη απο 1, τότε την
θέτουμε 1. Με αυτόν τον τρόπο εξασφαλίζουμε ότι η κάθε παραγγελία προϊόντος θα
έχει
            // τουλάχιστον ένα τεμάχιο (ή όποια άλλη μονάδα τιμολόγησης
ορίζεται για το επιλεγμένο προϊόν).
            if (i_product_quantity < 1)
            {
                i_product_quantity = 1;
            }

            // Αν ο χρήστης έχει επιλέξει απόθεμα για το συγκεκριμένο προϊόν,
τότε διαχειρίζομαστε την διαθέσιμη ποσότητα του εμπορεύματος και την
διαθεσιμότητα
            // ανάλογα με την επιλογή του στην λίστα κουμπιών μοναδικής
επιλογής (RadioButtonList) διαχείρισης ποσότητας.
            if (gvw_stocks.SelectedIndex > -1)
            {
                // Λήψη της τιμής της ετικέτας (Label) διαθέσιμου αποθέματος
προϊόντος από τον πίνακα όψεως (GridView) αποθεμάτων και μετατροπή της σε
                // ακέραιο (integer).
                int i_available_stock_quantity =
CodeClass.uf_i_ConvertChildControlValueToInteger("QuantityLabel",
gvw_stocks.SelectedRow);

                // Έλεγχος αν μέσα στην λίστα όψεως (ListView) των πληροφοριών
προϊόντων, υπάρχει το κουτί επιλογής (CheckBox) της κατάστασης αποθέματος του
                // επιλεγμένου προϊόντος της παραγγελίας.
                if ((lvw_product_info.Items.Count > 0) &&
(lvw_product_info.Items[0].FindControl("StockOnlyCheckBox") is CheckBox))
                {
                    // Λήψη του κουτιού επιλογής (CheckBox) της κατάστασης
αποθέματος του επιλεγμένου προϊόντος της παραγγελίας (ακολουθείται αυτή η
διαδικασία
```

```
                    // καθώς δεν είναι εφικτή η απευθείας αλληλεπίδραση, καθώς
εμπεριέχεται στο σύνολο των αντικειμένων (child objects) της φόρμας ενός άλλου
                    // αντικειμένου (parent object)).
                    CheckBox cbx_stock_only =
(CheckBox)lvw_product_info.Items[0].FindControl("StockOnlyCheckBox");

                    // Ελέγχουμε αν ο χρήστης έχει επιλέξει μέγιστο απόθεμα,
ώστε να περιορίσουμε την ποσότητα στην μέγιστη τιμή σε περίπτωση υπέρβασης.
                    if ((i_product_quantity > i_available_stock_quantity) &&
(rbtnl_cart.Items[0].Selected == true))
                    {
                        i_product_quantity = i_available_stock_quantity;
                    }
                    else // Διαφορετικά, προπαραγγέλει αν ξεπεράσει την μέγιστη
ποσότητα του προϊόντος στην αποθήκη.
                    {
                        // Ελέγχουμε αν υπάρχει δυνατότητα προπαραγγελίας και
διαθέσιμο απόθεμα.
                        if ((i_available_stock_quantity > 0) &&
(cbx_stock_only.Enabled == false))
                        {
                            // Επειδή δεν διαθέτουμε τόσο πολύ εμπόρευμα, ο
χρήστης επιθυμεί την προπαραγγελία.
                            if (i_product_quantity >
i_available_stock_quantity)
                            {
                                cen_availability =
CodeClass.availability.preorder;
                            }
                        }
                        else // Διαφορετικά, δεν υπάρχει καμία επίλογή και
τερματίζουμε την διαδικασία.
                        {
                            return;
                        }
                    }
                }
            }

            // Εξαφανίζουμε το πλαίσιο ομαδοποίησης στοιχείων (Div) που
περιέχει την λίστα κουμπιών μοναδικής επιλογής (RadioButtonList) του καλαθιού
αγορών.
            div_cart_choice.Visible = false;

            // Μετατροπή της τιμών των πεδίων του κωδικού προσωρινής εγγραφής,
του κωδικού προϊόντος, του κωδικού εγγραφής προϊόντος, του ονόματος προϊόντος,
            // του κωδικού αποθήκης, της τιμή του προϊόντος και του κωδικού
αποθέματος από το καλάθι αγορών σε ακέραιο αριθμό (integer), σε δεκαδικό αριθμό
            // (decimal) και σε ακολουθία χαρακτήρων (string) ανάλογα με την
κάθε μεταβλητή.
            int i_id = CodeClass.uf_i_ConvertValueToInteger(e.NewValues[0]);
            int i_product_code =
CodeClass.uf_i_ConvertValueToInteger(e.NewValues[1]);
            int i_product_id =
CodeClass.uf_i_ConvertValueToInteger(e.NewValues[2]);
            string s_product_name =
CodeClass.uf_s_ConvertValueToString(e.NewValues[3]);
            int i_storage_id =
CodeClass.uf_i_ConvertValueToInteger(e.NewValues[4]);
            decimal d_product_price =
CodeClass.uf_d_ConvertValueToDecimal(e.NewValues[7]);
```

```
            int i_stock_id =
CodeClass.uf_i_ConvertValueToInteger(e.NewValues[8]);

            // Υπολογισμός της συνολικής αξίας του προϊόντος.
            decimal d_product_total_price = d_product_price *
i_product_quantity;

            // Έλεγχος αν η διαθεσιμότητα είναι στην προπαραγγελία, καθώς τότε
δεν χρειαζόμαστε το όνομα της αποθήκης.
            if (cen_availability != CodeClass.availability.preorder)
            {
                // Μετατροπή της τιμής του πεδίου του ονόματος αποθήκης από το
καλάθι αγορών σε ακολουθία χαρακτήρων (string).
                s_storage_name =
CodeClass.uf_s_ConvertValueToString(e.NewValues[5]);
            }

            // Ενημέρωση της υπάρχουσας εγγραφής παραγγελίας από τα νέα
στοιχεία που συλλέξαμε παραπάνω.
            CodeClass.order cdt_order = new CodeClass.order(i_id,
i_product_code, i_product_id, s_product_name, i_storage_id, s_storage_name,
i_product_quantity,
                d_product_price, i_stock_id, d_product_total_price,
cen_availability);

            // Ενημέρωση της συγκεκριμένης εγγραφής παραγγελίας στο καλάθι
αγορών.
            uf_v_CartUpdateProduct(cdt_order);

            // Κλείνουμε την φόρμα αλλαγών του καλαθιού αγορών (δηλαδή τα
ανοιχτά πεδία της εγγραφής όπου μεταβάλλουμε τις τιμές τους).
            gvw_cart.EditIndex = -1;

            // Σύνδεση δεδομένων με τον πίνακα όψεως (GridView). Στην ουσία
ξαναφορτώνουμε τα δεδομένα, για να ενημερωθεί ο πίνακας όψεως (GridView)
            // με τις αλλαγές.
            gvw_cart.DataBind();
        }

        protected void gvw_cart_RowDataBound(object sender,
GridViewRowEventArgs e)
        {
            // Λήψη της παρούσας γραμμής από τον πίνακα όψεως (GridView) του
καλαθιού αγορών.
            GridViewRow gvwr_row = e.Row;

            int i_row_cells_count = gvwr_row.Cells.Count; // Ο αριθμός κελιών
σε μια γραμμή πίνακα όψεως (Gridview).
            int i_predefined_max_cell = 11; // Ορίζει το μέγιστο αριθμό κελιών
σε μια γραμμή. Λειτουργεί σαν προστασία από σφάλματα κελιών εκτός οριών πίνακα.

            // Προστασία από σφάλματα κελιών εκτός ορίων, σαν πρόσθετο μέτρο σε
περιπτώσεις ανανέωση κώδικα ή απρόβλεπτων καταστάσεων.
            if (i_row_cells_count < i_predefined_max_cell)
            {
                return;
            }

            // Αν ο τύπος του αντικειμένου γραμμής είναι επικεφαλίδα (Header),
τότε ορίζουμε τα ονόματα των στηλών.
            if (gvwr_row.RowType == DataControlRowType.Header)
            {
```

```
                gvwr_row.Cells[2].Text = "Κωδικός";
                gvwr_row.Cells[4].Text = "Προϊόν";
                gvwr_row.Cells[6].Text = "Διανομή";
                gvwr_row.Cells[7].Text = "Ποσότητα";
                gvwr_row.Cells[8].Text = "Τιμή";
                gvwr_row.Cells[10].Text = "Τελική τιμή";
                gvwr_row.Cells[11].Text = "Διαθεσιμότητα";
            }

            // Αν η γραμμή είναι μεταβολής, τότε μορφοποίησε τα κελιά της
παραγγελίας ώστε να επεμβαίνει ο χρήστης μόνο στο κελί της ποσότητας του
προϊόντος.
            if ((gvwr_row.RowState == (DataControlRowState.Edit |
DataControlRowState.Alternate)) || (gvwr_row.RowState ==
DataControlRowState.Edit))
            {
                // Λήψη του φόντου της γραμμής, ώστε να μορφοποιηθούν τα
υπόλοιπα κελιά ως ανενεργά.
                Color clr_back_color = gvw_cart.EditRowStyle.BackColor;

                // Μορφοποίηση των κελιών με τέτοιον τρόπο που ο χρήστης να
μπορεί να επέμβει μόνο στο κελί της ποσότητας του προϊόντος.
                uf_v_SetTextBoxStyle(gvwr_row.Cells[2].Controls, false, 70,
true, clr_back_color, false); // Κωδικός.
                uf_v_SetTextBoxStyle(gvwr_row.Cells[4].Controls, true, 200,
true, clr_back_color, false); // Προϊόν.
                uf_v_SetTextBoxStyle(gvwr_row.Cells[6].Controls, false, 100,
true, clr_back_color, false); // Διανομή.
                uf_v_SetTextBoxStyle(gvwr_row.Cells[7].Controls, false, 50,
false, Color.Empty, true); // Ποσότητα.
                uf_v_SetTextBoxStyle(gvwr_row.Cells[8].Controls, false, 30,
true, clr_back_color, false); // Τιμή.
                uf_v_SetTextBoxStyle(gvwr_row.Cells[10].Controls, false, 50,
true, clr_back_color, false); // Τελική τιμή.
                uf_v_SetTextBoxStyle(gvwr_row.Cells[11].Controls, false, 110,
true, clr_back_color, false); // Διαθεσιμότητα.
            }

            // Απενεργοποιούμε την ορατότητα στις στήλες των κωδικών εγγραφής,
προϊόντος, αποθήκης και αποθέματος.
            gvwr_row.Cells[1].Visible = false; // ID
            gvwr_row.Cells[3].Visible = false; // Product ID
            gvwr_row.Cells[5].Visible = false; // Storage ID
            gvwr_row.Cells[9].Visible = false; // Stock ID
        }

        /// <summary>
        /// Μορφοποίηση ενός κουτιού κειμένου (TextBox) που βρίσκεται μέσα σε
μια συλλογή αντικειμένων.
        /// </summary>
        /// <param name="ctrlc_control_collection">Η συλλογή των αντικειμένων
που περιέχεται το κουτί κειμένου (TextBox) που θέλουμε να
μορφοποιήσουμε.</param>
        /// <param name="b_multiline">Η κατάσταση αληθείας (boolean) για την
ενεργοποίηση των πολλαπλών γραμμών (multiline) στο κουτί κειμένου
(TextBox).</param>
        /// <param name="i_width">Το πλάτος του κουτιού κειμένου
(TextBox).</param>
        /// <param name="b_hide_border">Η κατάσταση αληθείας (boolean) για την
ενεργοποίηση της γραμμής πλαισίου (Border).</param>
        /// <param name="clr_back_color">Το χρώμα του φόντου του
κελιού.</param>
```

```
        /// <param name="b_enable">Η κατάσταση αληθείας (boolean) για την
ενεργοποίηση του αντικειμένου.</param>
        private void uf_v_SetTextBoxStyle(ControlCollection
ctrlc_control_collection, bool b_multiline, int i_width, bool b_hide_border,
Color clr_back_color,
            bool b_enable)
        {
            // Έλεγχος αν μέσα στη συλλογή των αντικειμένων, το πρώτο
αντικείμενο είναι ένα κουτιού κειμένου (TextBox). Το κουτί αυτό είναι αυτό που
εμφανίζεται
            // κατά την μεταβολή των στοιχείων ενός πίνακα όψεως (GridView).
            if ((ctrlc_control_collection.Count > 0) &&
(ctrlc_control_collection[0] is TextBox))
            {
                // Λήψη του κουτιού κειμένου (TextBox) από την συλλογή.
                TextBox tbx_control = (TextBox)ctrlc_control_collection[0];

                // Ενεργοποιούμε την δυνατότητα πολλαπλών γραμμών στο κουτί
κειμένου (TextBox) με βάση την κατάσταση αληθείας (boolean) που μας δώθηκε.
                if (b_multiline == true)
                {
                    tbx_control.TextMode = TextBoxMode.MultiLine;
                }

                // Ενεργοποιούμε την αλληλεπίδραση με του κουτί κειμένου
(TextBox) με βάση την κατάσταση αληθείας (boolean) που μας δώθηκε.
                tbx_control.Enabled = b_enable;

                // Αν μας έχει δωθεί τιμή για το πλάτος του κουτιού κειμένου
(TextBox), τότε την αλλάζουμε σύμφωνα με αυτήν.
                if (i_width > 0)
                {
                    tbx_control.Width = i_width;
                }

                // Αν μας ζητήθηκε να κρύψουμε την γραμμή πλαισίου (Border),
τότε θέτουμε την τιμή μηδέν για να την εξαφανίσουμε εντελώς.
                if (b_hide_border == true)
                {
                    tbx_control.BorderWidth = 0;
                }

                // Αν υπάρχει κάποιο χρώμα για το φόντο του κουτιού κειμένου
(TextBox), τότε ορίζουμε το χρώμα του φόντου του στο χρώμα που μας δώθηκε.
                // Χρησιμοποιούμε αυτό το τέχνασμα για να δείξουμε στο χρήστη
ποια πεδία είναι ανοιχτά προς μεταβολή.
                if (clr_back_color != Color.Empty)
                {
                    tbx_control.BackColor = clr_back_color;
                }
            }
        }

        protected void gvw_cart_RowDeleting(object sender,
GridViewDeleteEventArgs e)
        {
            // Λήψη της επιλεγμένης προς διαγραφή εγγραφής παραγγελίας
προϊόντος από τον πίνακα όψεως (GridView) του καλαθιού αγορών.
            GridViewRow gvwr_row_deleting = gvw_cart.Rows[e.RowIndex];

            int i_row_cells_count = gvwr_row_deleting.Cells.Count; // Ο αριθμός
κελιών σε μια γραμμή πίνακα όψεως (Gridview).
```

```
            int i_predefined_max_cell = 12; // Ορίζει το μέγιστο αριθμό κελιών
σε μια γραμμή. Λειτουργεί σαν προστασία από σφάλματα κελιών εκτός οριών πίνακα.

            // Προστασία από σφάλματα κελιών εκτός ορίων, σαν πρόσθετο μέτρο σε
περιπτώσεις ανανέωση κώδικα ή απρόβλεπτων καταστάσεων.
            if (i_row_cells_count < i_predefined_max_cell)
            {
                return;
            }

            // Μετατροπή των τιμών των πεδίων του κωδικού προσωρινής εγγραφής,
του κωδικού προϊόντος, του κωδικού εγγραφής προϊόντος, του ονόματος προϊόντος,
            // του κωδικού αποθήκης, του ονόματος αποθήκης, της ποσότητας
προϊόντος, της τιμή του προϊόντος, του κωδικού αποθέματος και της συνολικής
αξίας
            // εγγραφής παραγγελίας από τον πίνακα όψεως (GridView) του
καλαθιού αγορών σε ακέραιο αριθμό (integer), σε δεκαδικό αριθμό (decimal) και
σε ακολουθία
            // χαρακτήρων (string) ανάλογα με την κάθε μεταβλητή.
            int i_id =
CodeClass.uf_i_ConvertValueToInteger(gvwr_row_deleting.Cells[1].Text);
            int i_product_code =
CodeClass.uf_i_ConvertValueToInteger(gvwr_row_deleting.Cells[2].Text);
            int i_product_id =
CodeClass.uf_i_ConvertValueToInteger(gvwr_row_deleting.Cells[3].Text);
            string s_product_name =
CodeClass.uf_s_ConvertValueToString(gvwr_row_deleting.Cells[4].Text);
            int i_storage_id =
CodeClass.uf_i_ConvertValueToInteger(gvwr_row_deleting.Cells[5].Text);
            string s_storage_name =
CodeClass.uf_s_ConvertValueToString(gvwr_row_deleting.Cells[6].Text);
            int i_product_quantity =
CodeClass.uf_i_ConvertValueToInteger(gvwr_row_deleting.Cells[7].Text);
            decimal d_product_price =
CodeClass.uf_d_ConvertValueToDecimal(gvwr_row_deleting.Cells[8].Text);
            int i_stock_id =
CodeClass.uf_i_ConvertValueToInteger(gvwr_row_deleting.Cells[9].Text);
            decimal d_product_total_price =
CodeClass.uf_d_ConvertValueToDecimal(gvwr_row_deleting.Cells[10].Text);

            // Λήψη της τιμής του πεδίου διαθεσιμότητας από τον πίνακα όψεως
(GridView) του καλαθιού και μετάφραση της ακολουθίας χαρακτήρων (string)
ελληνικών σε
            // απαρίθμηση δεδομένων (enumeration) που δείχνει την διαθεσιμότητα
κάθε εγγραφής παραγγελίας.
            CodeClass.availability cen_availability =
CodeClass.uf_cen_AvailabilityStringToEnum(gvwr_row_deleting.Cells[11].Text);

            // Σύνθεση της εγγραφής παραγγελίας από τα στοιχεία που συλλέξαμε
παραπάνω, με σκοπό την διαγραφή της από το καλάθι αγορών.
            CodeClass.order cdt_order = new CodeClass.order(i_id,
i_product_code, i_product_id, s_product_name, i_storage_id, s_storage_name,
i_product_quantity,
                d_product_price, i_stock_id, d_product_total_price,
cen_availability);

            // Διαγραφή της συγκεκριμένης εγγραφής παραγγελίας από το καλάθι
αγορών.
            uf_v_CartRemoveProduct(cdt_order);
        }
```

```
        protected void gvw_cart_RowCancelingEdit(object sender,
GridViewCancelEditEventArgs e)
        {
            // Εξαφανίζουμε το πλαίσιο ομαδοποίησης στοιχείων (Div) που
περιέχει την λίστα κουμπιών μοναδικής επιλογής (RadioButtonList) του καλαθιού
αγορών.
            div_cart_choice.Visible = false;

            // Κλείνουμε την φόρμα αλλαγών του καλαθιού αγορών (δηλαδή τα
ανοιχτά πεδία της εγγραφής όπου μεταβάλλουμε τις τιμές τους).
            gvw_cart.EditIndex = -1;

            // Δημιουργία του πίνακα δεδομένων (DataTable) του καλαθιού αγορών
και σύνδεση των εγγραφών του με τον αντίστοιχο πίνακα όψεως (GridView).
            uf_v_CartProducts();
        }

        protected void btn_add_to_cart_Click(object sender, EventArgs e)
        {
            // Συλλογή στοιχείων προϊόντων για την δημιουργία μιας νέας
εγγραφής παραγγελίας στο καλάθι αγορών του χρήστη.
            uf_v_AddToCart();
        }

        protected void lvw_product_info_ItemDataBound(object sender,
ListViewItemEventArgs e)
        {
            // Λήψη του αντικειμένου λίστας όψεως (ListViewItem).
            ListViewItem lvwi_item = e.Item;

            // Έλεγχος αν μέσα στην λίστα όψεως (ListView) των πληροφοριών
προϊόντων, υπάρχει η ετικέτα (Label) του κωδικού καταχώρησης του επιλεγμένου
προϊόντος
            // της παραγγελίας.
            if (lvwi_item.FindControl("IDLabel") is Label)
            {
                // Λήψη της ετικέτας (Label) του κωδικού καταχώρησης του
επιλεγμένου προϊόντος της παραγγελίας (ακολουθείται αυτή η διαδικασία
                // καθώς δεν είναι εφικτή η απευθείας αλληλεπίδραση, καθώς
εμπεριέχεται στο σύνολο των αντικειμένων (child objects) της φόρμας ενός άλλου
                // αντικειμένου (parent object)).
                Label lbl_id = (Label)lvwi_item.FindControl("IDLabel");

                // Απενεργοποιούμε την ετικέτα (Label) κωδικού καταχώρησης του
επιλεγμένου προϊόντος της παραγγελίας.
                CodeClass.uf_v_EnableWebControl(lbl_id, false);
            }

            // Έλεγχος αν μέσα στην λίστα όψεως (ListView) των πληροφοριών
προϊόντων, υπάρχει το κουτί επιλογής (CheckBox) της κατάστασης αποθέματος του
            // επιλεγμένου προϊόντος της παραγγελίας.
            if (lvwi_item.FindControl("StockOnlyCheckBox") is CheckBox)
            {
                // Λήψη του κουτιού επιλογής (CheckBox) της κατάστασης
αποθέματος του επιλεγμένου προϊόντος της παραγγελίας (ακολουθείται αυτή η
διαδικασία
                // καθώς δεν είναι εφικτή η απευθείας αλληλεπίδραση, καθώς
εμπεριέχεται στο σύνολο των αντικειμένων (child objects) της φόρμας ενός άλλου
                // αντικειμένου (parent object)).
                CheckBox cbx_stock_only =
(CheckBox)lvwi_item.FindControl("StockOnlyCheckBox");
```

```
                // Απενεργοποιούμε το κουτί επιλογής (CheckBox) της κατάστασης
αποθέματος του επιλεγμένου προϊόντος της παραγγελίας.
                CodeClass.uf_v_EnableWebControl(cbx_stock_only, false);
            }
        }

        protected void gvw_stocks_SelectedIndexChanged(object sender, EventArgs
e)
        {
            // Έλεγχος των αποθεματικών του επιλεγμένου προϊόντος, με
ενεργοποίηση του κουμπιού (Button) εισαγωγής προϊόντος στο καλάθι αγορών και
εμφάνισης
            // μηνύματος πληροφοριών αποθέματος επιλεγμένης εγγραφής προϊόντος
ανάλογα με την περίπτωση.
            uf_v_CartAddButton();
        }

        protected void gvw_cart_DataBound(object sender, EventArgs e)
        {
            // Εύρεση του πλήθους των εγγραφών στο καλάθι αγορών.
            int i_cart_rows = gvw_cart.Rows.Count;

            // Έλεγχος αν το καλάθι αγορών έχει εγγραφές, για να υπολογιστεί το
σύνολο του κόστους των προϊόντων.
            if (i_cart_rows > 0)
            {
                // Λήψη των εγγραφών παραγγελίας στο καλάθι αγορών του χρήστη.
                CodeClass.order[] cdta_order = uf_cdta_ShoppingCart();

                // Μεταβλητή αποθήκευσης της συνολικής τιμής των προϊόντων.
                decimal d_total_price = 0;

                // Συλλογή των τιμών των εγγραφών σε κάθε παραγγελίας στο
καλάθι αγορών για τον υπολογισμό της συνολικής τιμής του εμπορεύματος.
                for (int i = 0; i < cdta_order.Length; i++)
                {
                    // Λήψη μιας παραγγελίας από το σύνολο των παραγγελιών του
χρήστη που υπάρχουν στο καλάθι αγορών του.
                    CodeClass.order cdt_order_check =
(CodeClass.order)cdta_order[i];

                    // Αν η ποσότητα του προϊόντος της παραγγελίας είναι
θετική, τότε πρόσθεσε την τιμή του εμπορεύματος μαζί με τα υπόλοιπα
υπολογισθέντα.
                    if (cdt_order_check.i_product_quantity > 0)
                    {
                        d_total_price = d_total_price +
cdt_order_check.d_product_total_price;
                    }
                }

                // Εμφάνιση στο χρήστη της συνολικής τιμής της παραγγελίας του.
                lbl_total_price.Text = "Συνολική τιμή: " + d_total_price + "
€";
            }
            else // Διαφορετικά, δεν υπάρχει λόγος να εμφανίσουμε την αξία των
αγορών στο χρήστη.
            {
                lbl_total_price.Text = "";
            }
```

```
            // Εξ ορισμού εμφανίζουμε το μήνυμα που πληροφορεί ότι δεν υπάρχουν
προϊόντα στο καλάθι αγορών και απενεργοποιούμε την ολοκλήρωση της παραγγελίας.
            bool b_order_button = false;
            bool b_order_message = true;
            bool b_order_comments = false;

            // Αν υπάρχουν εγγραφές στο καλάθι αγορών, τότε ενεργοποιούμε την
ολοκλήρωση της παραγγελίας και απενεργοποιούμε το το μήνυμα που πληροφορεί ότι
            // δεν υπάρχουν προϊόντα στο καλάθι αγορών
            if (i_cart_rows > 0)
            {
                b_order_button = true;
                b_order_message = false;
                b_order_comments = true;
            }

            // Ενεργοποιούμε το κουμπί (Button) καταχώρησης παραγγελίας, τις
ετικέτες (Label) μηνύματος καλαθιού αγορών και παρατηρήσεων παραγγελίας και του
            // κουτιού κειμένου (TextBox) παρατηρήσεων παραγγελίας σύμφωνα με
τις καταστάσεις των αντίστοιχων μεταβλητών.
            CodeClass.uf_v_EnableWebControl(btn_order, b_order_button);
            CodeClass.uf_v_EnableWebControl(lbl_cart_message, b_order_message);
            CodeClass.uf_v_EnableWebControl(lbl_order_comments,
b_order_comments);
            CodeClass.uf_v_EnableWebControl(tbx_order_comments,
b_order_comments);
        }

        /// <summary>
        /// Λήψη των εγγραφών παραγγελίας στο καλάθι αγορών του χρήστη.
        /// </summary>
        /// <returns>Τις εγγραφές παραγγελίας στο καλάθι αγορών του
χρήστη.</returns>
        private CodeClass.order[] uf_cdta_ShoppingCart()
        {
            // Μεταβλητή αποθήκευσης των εγγραφών παραγγελίας στο καλάθι αγορών
του χρήστη.
            CodeClass.order[] cdta_order = new CodeClass.order[0];

            // Ελέγχουμε αν η συνεδρία (Session) που διατηρεί στην μνήμη του
διακομιστή (server) το καλάθι αγορών του χρήστη είναι ενεργή και στον σωστό
τύπο
            // δεδομένων. (Η κάθε συνεδρία είναι μοναδική για κάθε χρήστη,
ασχέτως αν έχουν το ίδιο όνομα κλήσης στον κώδικα και δεν είναι προσβάσιμες από
τον
            // χρήστη παρά μόνο από την εφαρμογή).
            if ((Session["Cart"] != null) && (Session["Cart"] is
CodeClass.order[]))
            {
                // Λήψη των εγγραφών παραγγελίας στο καλάθι αγορών του χρήστη.
                cdta_order = (CodeClass.order[])Session["Cart"];
            }

            // Επιστρέφει τις εγγραφές παραγγελίας στο καλάθι αγορών του
χρήστη.
            return cdta_order;
        }

        protected void tmr_session_Tick(object sender, EventArgs e)
        {
            // Ο ακόλουθος μηχανισμός ανανεώνει τις συνεδρίες (Sessions) του
συνδεδεμένου χρήστη στην μνήμη του διακομιστή (server). Στην ουσία τις
```

```
            // επαναπροσδιορίζει λίγο πριν την λήξη του χρόνου ζωής (timeout)
της συνεδρίας που εξ ορισμού είναι τα είκοσι (20) λεπτά. Εμείς θα ανανεώσουμε
μόνο
            // τις σημαντικές, δηλαδή αυτές που σχετίζονται με το καλάθι αγορών
του, καθώς δεν θέλουμε ένας ξεχασμένος πελάτης να χάσει τις επιλογές προϊόντων
που
            // διάλεξε.

            // Ελέγχουμε αν η συνεδρία (Session) που διατηρεί στην μνήμη του
διακομιστή (server) το καλάθι αγορών του χρήστη είναι ενεργή και στον σωστό
τύπο
            // δεδομένων. (Η κάθε συνεδρία είναι μοναδική για κάθε χρήστη,
ασχέτως αν έχουν το ίδιο όνομα κλήσης στον κώδικα και δεν είναι προσβάσιμες από
τον
            // χρήστη παρά μόνο από την εφαρμογή).
            if ((Session["Cart"] != null) && (Session["Cart"] is
CodeClass.order[]))
            {
                // Λήψη και ανανέωση της συνεδρίας (Session) του καλαθιού
αγορών.
                CodeClass.order[] cdta_cart =
(CodeClass.order[])Session["Cart"];
                Session["Cart"] = cdta_cart;
            }

            // Ελέγχουμε αν η συνεδρία (Session) που διατηρεί στην μνήμη του
διακομιστή (server) τον κωδικό προσωρινής εγγραφής στο καλάθι αγορών του χρήστη
είναι
            // ενεργή και στον σωστό τύπο δεδομένων. (Η κάθε συνεδρία είναι
μοναδική για κάθε χρήστη, ασχέτως αν έχουν το ίδιο όνομα κλήσης στον κώδικα και
δεν
            // είναι προσβάσιμες από τον χρήστη παρά μόνο από την εφαρμογή). Ο
κωδικός αυτός χρησιμοποιείται για την εσωτερική οργάνωση του καλαθιού αγορών
και δεν
            // καταχωρείται στην τελική παραγγελία.
            if ((Session["CartID"] != null) && (Session["CartID"] is int))
            {
                // Λήψη και ανανέωση της συνεδρίας (Session) του προσωρινού
κωδικοπυ εγγραφής στο καλάθι αγορών.
                int i_id = (int)Session["CartID"];
                Session["CartID"] = i_id;
            }
        }

        protected void btn_order_Click(object sender, EventArgs e)
        {
            // Συλλογή των στοιχείων λογαριασμού του χρήστη που είναι
συνδεδεμένος στο σύστημα.
            MembershipUser msu_user =
Membership.GetUser(HttpContext.Current.User.Identity.Name);

            // Αν δεν υπάρχει συνδεδεμένος χρήστης, τότε τερματίζουμε την
διαδικασία καταχώρησης παραγγελίας στο σύστημα.
            if (msu_user == null)
            {
                return;
            }

            // Λήψη εγγραφών της παραγγελίας από το καλάθι αγορών (κάθε εγγραφή
περιέχει ένα προϊόν με επιλεγμένες από το χρήστη ποσσότητες).
            CodeClass.order[] cdta_order = uf_cdta_ShoppingCart();
```

```
            // Καταμέτρηση του πλήθους των συνολικών εγγραφών της παραγγελίας.
            int i_order_count = cdta_order.Length;

            // Αν δεν υπάρχουν εγγραφές στην παραγγελία, τότε ο χρήστης δεν
έχει παραγγείλει τίποτα, άρα δεν υπάρχει λόγος να συνεχίσουμε την διαδικασία.
            if (i_order_count < 1)
            {
                return;
            }

            Guid guid_customer = (Guid)msu_user.ProviderUserKey; // Μοναδικό
προσδιοριστικό (Unique ID) συνδεδεμένου χρήστη.

            // Λήψη της πρώτης διεύθυνσης αποθήκης από την βάση δεδομένων. Ο
διαχειριστής μπορεί να μεταβάλει αυτή την τιμή μέσα από τον έλεγχο παραγγελιών
(Orders)
            // στα εργαλεία διαχειριστή.
            sqlds_top_storage_address.Select(DataSourceSelectArguments.Empty);

            // Αν δεν υπάρχουν εγγραφές στις διευθύνσεις αποθήκης, τότε δεν
υπάρχει αποστολέας για το εμπόρευμα και η διαδικασία δεν έχει νόημα να
συνεχιστεί.
            if (priv_i_storage_address_id < 1)
            {
                return;
            }

            // Λήψη των παρατηρήσεων του χρήστη επί της παραγγελίας από το
κουτί κειμένου (TextBox).
            string s_comments = tbx_order_comments.Text;

            // Ελέγχουμε αν οι παρατηρήσεις του χρήστη επί της παραγγελίας
έχουν περιττά κενά για να τα καθαρίσουμε.
            if (s_comments.Trim() == "")
            {
                s_comments = s_comments.Trim();
            }

            // Εισαγωγή στο σύστημα του μοναδικού προσδιοριστικού (Unique ID)
του χρήστη, των παρατηρήσεων επί της παραγγελίας και του κωδικού καταχώρησης
            // διεύθυνσης αποθήκης για την εισαγωγή της νέας εγγραφής
παραγγελίας στον πίνακα παραγγελιών (Orders).
            sqlds_order.InsertParameters["CustomerUserID"].DefaultValue =
guid_customer.ToString();
            sqlds_order.InsertParameters["Comments"].DefaultValue = s_comments;
            sqlds_order.InsertParameters["StorageAddressID"].DefaultValue =
priv_i_storage_address_id.ToString();

            // Εκτέλεση της εντολής sql για εισαγωγή μιας νέας εγγραφής στον
πίνακα παραγγελιών (Orders) και λήψη του κωδικού εγγραφής της καταχωρημένης
            // παραγγελίας.
            int i_order_rows_inserted = sqlds_order.Insert();

            // Αν δεν έχει καταχωρηθεί η παραγγελία στον πίνακα παραγγελίων
(Orders) ή ο κωδικός εγγραφής της καταχωρημένης παραγγελίας δεν είναι έγκυρος,
            // τότε τερματίζουμε την διαδικασία.
            if ((i_order_rows_inserted < 1) || (priv_i_order_id < 1))
            {
                return;
            }
```

```
            // Μεταβλήτη αποθήκευσης του κωδικού παραγγελίας που θα
χρησιμοποιηθεί σε εντολές sql.
            string s_order_id = priv_i_order_id.ToString();

            // Εισαγωγή κάθε παραγγελίας του καλαθιού αγορών του χρήστη στην
βάση δεδομένων, εφόσον είναι έγκυρη βάση των παρακάτω ελέγχων.
            for (int i = 0; i < i_order_count; i++)
            {
                // Λήψη μιας παραγγελίας από το σύνολο των παραγγελιών του
χρήστη που υπάρχουν στο καλάθι αγορών του.
                CodeClass.order cdt_order_check = cdta_order[i];

                // Λήψη της ποσότητα του προϊόντος της παραγγελίας.
                int i_quantity = cdt_order_check.i_product_quantity;

                // Αν για κάποιο λόγο η ποσότητα είναι μικρότερη της μονάδας,
τότε η παραγγελία δεν είναι έγκυρη και προχωράμε στην επόμενη.
                if (i_quantity < 1)
                {
                    continue;
                }

                // Λήψη του κωδικου αποθέματος και της διαθεσιμότητας του
επιλεγμένου προϊόντος της παραγγελίας.
                int i_stock_id = cdt_order_check.i_stock_id;
                CodeClass.availability cen_availability =
cdt_order_check.cen_availability;

                // Μεταβλητή αποθήκευσης κατάστασης αληθείας (boolean) για την
προπαραγγελια και ορισμένη στην ψευδή (false).
                bool b_preorder = false;

                // Μεταβλήτη αποθήκευσης του κωδικού αποθέματος που θα
χρησιμοποιηθεί σε εντολές sql.
                string s_stock_id = i_stock_id.ToString();

                // Μεταβολή του αποθέματος του προϊόντος σύμφωνα με την
ποσότητα της παραγγελίας. Εξαιρείται η προπαραγγελία, η οποία δεν επεμβαίνει
ποτέ στο
                // αποθεματικό μιας αποθήκης.
                switch (cen_availability)
                {
                    case CodeClass.availability.available: // Το προϊόν είναι
διαθέσιμο.
                        // Εισαγωγή στο σύστημα του κωδικού εγγραφής αποθέματος
για την επιλογή της εγγραφής αποθέματος προϊόντων από τον πίνακα αποθεμάτων
(Stocks).

sqlds_product_quantity.SelectParameters["ID"].DefaultValue = s_stock_id;

                        // Λήψη της ποσότητας του διαθέσιμου αποθέματος του
προϊόντος από τον πίνακα αποθεμάτων (Stocks) για τον συγκεκριμένο κωδικό
εγγραφής
                        // αποθέματος.

sqlds_product_quantity.Select(DataSourceSelectArguments.Empty);

                        // Αν το διαθέσιμο απόθεμα είναι μικρότερο της
ποσότητας του προϊόντος της παραγγελίας ή δεν υπάρχει, τοτε η παραγγελία δεν
είναι έγκυρη και
                        // προχωράμε στην επόμενη.
```

```
                        if ((priv_i_stock_quantity < i_quantity) ||
(priv_i_stock_quantity < 1))
                        {
                            continue;
                        }

                        // Αφαίρεση από την ποσότητα του αποθέματος την
ποσότητα παραγγελίας του προϊόντος.
                        int i_quantity_stock = priv_i_stock_quantity -
i_quantity;

                        // Εισαγωγή στο σύστημα του κωδικού εγγραφής αποθέματος
και της διαθέσιμης ποσότητας προϊόντων στο απόθεμα για την ενημέρωση της
εγγραφής
                        // αποθεμάτων προϊόντων στον πίνακα αποθεμάτων
(Stocks).

sqlds_product_quantity.UpdateParameters["ID"].DefaultValue = s_stock_id;

sqlds_product_quantity.UpdateParameters["Quantity"].DefaultValue =
i_quantity_stock.ToString();

                        // Εκτέλεση της εντολής sql για ενημέρωση της εγγραφής
του αποθέματος του προϊόντος στον πίνακα αποθεμάτων (Stocks).
                        sqlds_product_quantity.Update();
                        break;
                    case CodeClass.availability.preorder: // Το προϊόν
προπαραγγέλεται.
                        b_preorder = true;
                        break;
                    default:
                        continue;
                }

                // Εισαγωγή στο σύστημα του κωδικού εγγραφής της παραγγελίας,
του κωδικού προϊόντος, της τιμής προϊόντος (για να υπάρχει σε περιπτώσεις
αλλαγής της
                // τιμής στο προϊόν, όπου φαίνεται πλέον πόσο πλήρωσε ο πελάτης
επί του προϊόντος), της ποσότητας προϊόντος που παραγγέλθηκε και της
διαθεσιμότητας
                // του αποθέματος του προϊόντος για την εισαγωγή μιας νέας
εγγραφής στον πίνακα προϊόντων παραγγελιών (Orders Products).
                sqlds_order_product.InsertParameters["OrderID"].DefaultValue =
s_order_id;
                sqlds_order_product.InsertParameters["ProductID"].DefaultValue
= cdt_order_check.i_product_id.ToString();
                sqlds_order_product.InsertParameters["Price"].DefaultValue =
cdt_order_check.d_product_price.ToString();
                sqlds_order_product.InsertParameters["Quantity"].DefaultValue =
i_quantity.ToString();
                sqlds_order_product.InsertParameters["Preorder"].DefaultValue =
b_preorder.ToString();

                // Εκτέλεση της εντολής sql για εισαγωγή μιας νέας εγγραφής
στον πίνακα προϊόντων παραγγελιών (Orders Products).
                sqlds_order_product.Insert();
            }

            // Καθαρισμός από την μνήμη του διακομιστή (server) των συνεδριών
(Session) του καλαθιού αγορών και του κωδικού προσωρινής εγγραφής στο καλάθι
αγορών.
            Session.Remove("Cart");
```

```
            Session.Remove("CartID");

            // Καθαρισμός της πηγής δεδομένων (DataSource) από το καλάθι
αγορών.
            gvw_cart.DataSource = null;

            // Σύνδεση δεδομένων με τους πίνακες όψεως (GridView). Στην ουσία
ξαναφορτώνουμε τα δεδομένα, για να ενημερωθούν οι πίνακες όψεως (GridView)
            // με τις αλλαγές.
            gvw_cart.DataBind();
            gvw_products.DataBind();
            gvw_stocks.DataBind();
        }

        protected void sqlds_order_Inserted(object sender,
SqlDataSourceStatusEventArgs e)
        {
            // Λήψη από τον πίνακα των παραγγελιών (Orders) του κωδικού
εγγραφής της καταχωρημένης παραγγελίας.
            priv_i_order_id = (int)e.Command.Parameters["@ID"].Value;
        }

        protected void sqlds_product_quantity_Selected(object sender,
SqlDataSourceStatusEventArgs e)
        {
            // Λήψη από τον πίνακα των αποθεμάτων (Stocks) του διαθέσιμου
αποθέματος του προϊόντος.
            priv_i_stock_quantity =
(int)e.Command.Parameters["@Quantity"].Value;
        }

        protected void sqlds_top_storage_address_Selected(object sender,
SqlDataSourceStatusEventArgs e)
        {
            // Λήψη της πρώτης εγγραφής από τον πίνακα διευθύνσεων αποθηκών
(Storages Addresses).
            priv_i_storage_address_id =
(int)e.Command.Parameters["@StorageAddressID"].Value;
        }
    }
}
```

Πλαίσιο ιστοσελίδας

Site.Master

Ο ακόλουθος κώδικας διαχειρίζεται τις λίστες επιλογών (menu) ανάλογα με τα δικαιώματα των συνδεδεμένων χρηστών:

```
            // Συλλογή των στοιχείων λογαριασμού του χρήστη που βρίσκεται
συνδεδεμένος με το σύστημα τώρα.
            MembershipUser msu_user =
Membership.GetUser(HttpContext.Current.User.Identity.Name);

            // Αν κάποιος χρήστης είναι συνδεδεμένος με το σύστημα, τότε
εμφάνιζει την δυνατότητα διαχείρισης του λογαριασμού του.
            // Σημείωση: Οι ανώνυμοι χρήστες δεν θεωρούνται χρήστες με
δικαιώματα διαχείρισης λογαριασμών, επομένως δεν μπορούν να έχουν αυτή την
επιλογή.
            if (msu_user != null)
```

```
            {
                memberlink.Visible = true;
            }

            // Ενεργοποίηση των εργαλείων διαχειριστή μόνο σε χρήστες με τα
καταλλήλα δικαιώματα.
            if (HttpContext.Current.User.IsInRole("administrator") == true)
            {
                adminlink.Visible = true;
            }
```

Βιβλιογραφία

1. Sams Teach Yourself Microsoft Visual C# .Net 2003 in 24 Hours, James Foxall,2004
2. SQL Structured Query Language Β' ΕΚΔΟΣΗ,Διαχείριση σχεσιακών βάσεων δεδομένων μέσω της γλώσσας SQL και εκτεταμένη ανάλυση των εντολών της,Εκδόσεις Nubis,Κωνσταντίνος Α. Καδής,1993
3. Εισαγωγή στα συστήματα βάσεων δεδομένων, C.J.DATE, Εκδόσεις Κλειδάριθμος,1996
4. Πλήρης Εγχειρίδιο του Internet Millenium Edition,Μέρος V, Παράρτημα 27:Χειροκίνητη Δημιουργία Ιστοσελίδων,Εκδόσεις Β.Γκιούρδας,Συγγραφέας Margaret Levine Young,2001
5. Οδηγός της XML με παραδείγματα,Εκδόσεις Γκιούρδας,Benoit Marchal,2001
6. http://en.wikipedia.org/wiki/C_Sharp_(programming_language)
7. http://studentguru.gr/w/tutorials/01-c.aspx
8. http://www.visualstudio.com/en-us/products/visual-studio-express-vs.aspx
9. http://www.w3schools.com/default.asp
10. http://ebooks.programmersheaven.com/csharp_ebook.pdf
11. http://programmersheaven.com/
12. http://msdn.microsoft.com/en-us/library/aa288436(v=vs.71).aspx
13. http://www.codeguru.com/csharp/