ΑΝΩΤΑΤΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ
ΙΔΡΥΜΑ ΛΑΡΙΣΑΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ
ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Κατασκευή Ηλεκτρονικού Καταστήματος με τεχνολογίες Java (JSP, servlets, JDBC)

Παναγιώτης Σάτος
Ανδρέας Δράκος

ΕΠΙΒΛΕΠΩΝ: κ. Γεώργιος Κακαρόντζας, Καθηγητής Εφαρμογών

ΛΑΡΙΣΑ 2005

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή

Λάρισα .... / .... / .....

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1 ……. [ ]

2 ……. [ ]

3 ……. [ ]

Περίληψη

Στο πλαίσιο της εκπόνησης της πτυχιακής μας εργασίας, σε συνεργασία με τον επιβλέπον καθηγητή, αναλάβαμε την κατασκευή ηλεκτρονικού καταστήματος με τεχνολογίες Java. Συγκεκριμένα χρησιμοποιήθηκαν οι τεχνολογίες JSP, JavaServlets και βάση δεδομένων MySQL για την συγκράτηση των δεδομένων.

Ευχαριστίες

Η παρούσα πτυχιακή εργασία είναι αφιερωμένη στην μνήμη του ανθρώπου εκείνου που υπήρξε ο μεγαλύτερος εμπνευστής και δάσκαλος για εμένα. Στον αείμνηστο παππού μου Χρύσανθο.

Παναγιώτης Σάτος

ΠΕΡΙΕΧΟΜΕΝΑ

1. Εισαγωγή

1.1. Περιγραφή της εργασίας - Απλή αναφορά τεχνολογιών

Στην πτυχιακή μας μελέτη χρησιμοποιήθηκαν οι τεχνολογίες JavaServlets, JavaServer Pages και μέσω του JDBC είχαμε επικοινωνία μεταξύ της εφαρμογής και της βάσης δεδομένων MySQL που κρατούσε πληροφορίες για τους χρήστες και τα προϊόντα. Για την καλή εμφάνιση των περιεχομένων της εφαρμογής χρησιμοποιήθηκαν οι τεχνολογίες HTML, JavaScript, CSS καθώς και τα προγράμματα Macromedia FlashMX, CorelDRAW 11, Photoshop 7.0. Επίσης να τονίσουμε ότι οι σελίδες σχεδιάστηκαν μόνο με HTML κώδικα σε notepad χωρίς την χρήση βοηθητικών προγραμμάτων. Η ιστοσελίδα του ηλεκτρονικού μας καταστήματος έχει κατασκευαστεί σε δύο γλώσσες, στην ελληνική και στην αγγλική. Εκτός από πληροφόρηση των επισκεπτών – πελατών, παρέχει υπηρεσίες εγγραφής χρήστη με δυνατότητα παραγγελίας προϊόντων και εισαγωγής θέματος συζήτησης στο forum. Επίσης δίνεται η δυνατότητα στον χρήστη να ανανεώνει τα προσωπικά του στοιχεία και να του αποστέλλεται στο e-mail του ο κωδικός πρόσβασης σε περίπτωση που τον ξεχάσει. Ακόμα μπορεί να παρακολουθεί την εξέλιξη των παραγγελιών του. Από πλευράς διαχείρισης δύναται η δυνατότητα στον διαχειριστή να εισάγει, να ανανεώνει και να διαγράφει προϊόντα από την βάση καθώς και να αποστέλλει ενημερωτικά e-mail στους χρήστες. Επιπλέον μπορεί να κάνει upload txt αρχείο αναγράφοντας νέα του καταστήματος, τα οποία θα εμφανίζονται στην σελίδα των νέων, καθώς και φωτογραφίες των νέων προϊόντων που θα προσθέτει. Επιπλέον έχει την δυνατότητα να διαγράφει χρήστες καθώς και θέματα ή απαντήσεις από το forum της εφαρμογής. Τέλος μπορεί να ανανεώνει την κατάσταση των παραγγελιών των χρηστών ή ακόμα και να τις διαγράφει.

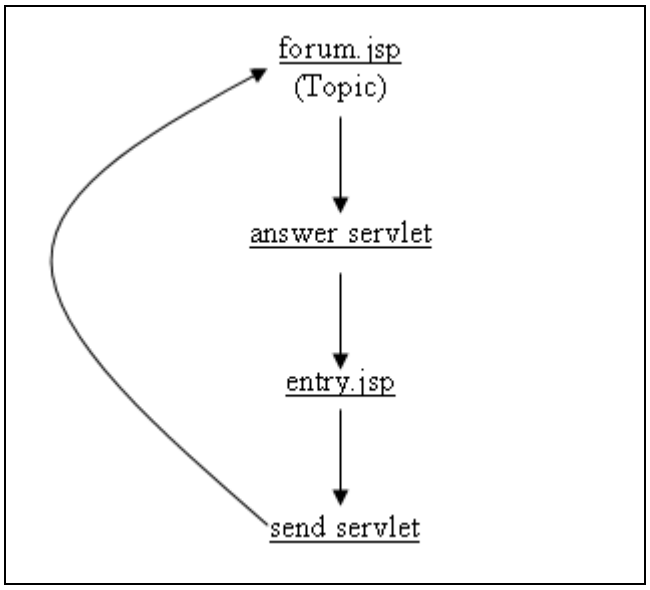
1.2. Ιστορικό της εργασίας

Την πτυχιακή μας μελέτη την αναλάβαμε τον Σεπτέμβριο του 2004 και την ολοκληρώσαμε τον Ιούλιο του 2005. Η υλοποίησή της έγινε σταδιακά. Αρχικά εργαστήκαμε στο θεωρητικό κομμάτι, δηλαδή ποιο θα είναι το θέμα του καταστήματος και τι δυνατότητες θα έχει. Μετά από έρευνά μας αποφασίσαμε την ενσωμάτωση όλων των λειτουργιών που βρήκαμε σε σελίδες διάφορων ηλεκτρονικών καταστημάτων, εκτός από την χρήση της παραγγελίας με πιστωτική κάρτα, κάτι που θα ξέφευγε από τους στόχους της παρούσας μελέτης. Κατά την διάρκεια των εργασιών παρουσιάστηκαν διάφορα προβλήματα τα οποία ξεπεράστηκαν ύστερα από πολύωρες αναζητήσεις στο Διαδίκτυο και στα βιβλία της βιβλιογραφίας μας. Καταλυτικής σημασίας υπήρξαν και οι παρατηρήσεις - διορθώσεις που μας αποστάλθηκαν από τον επιβλέποντα καθηγητή. Όπως γίνεται κατανοητό λόγω του γεγονότος ότι η φύση της μελέτης είναι καθαρά πρακτική, η πλειοψηφία των προβλημάτων ξεπεράστηκε ύστερα από πολλές δοκιμές και τροποποιήσεις του κώδικα. Επομένως δεν είναι εφικτή η αναφορά μας σε συγκεκριμένα προβλήματα και σε τρόπους επίλυσης.

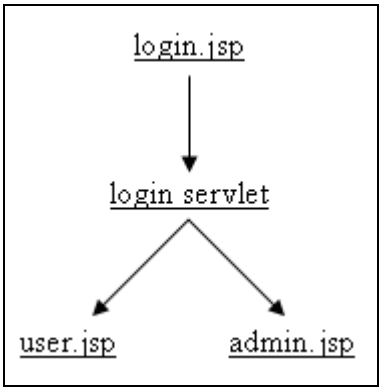
2. Σχεδίαση της εφαρμογής - Διαγράμματα

2.1. **Ολοκληρωμένο διάγραμμα λειτουργίας**

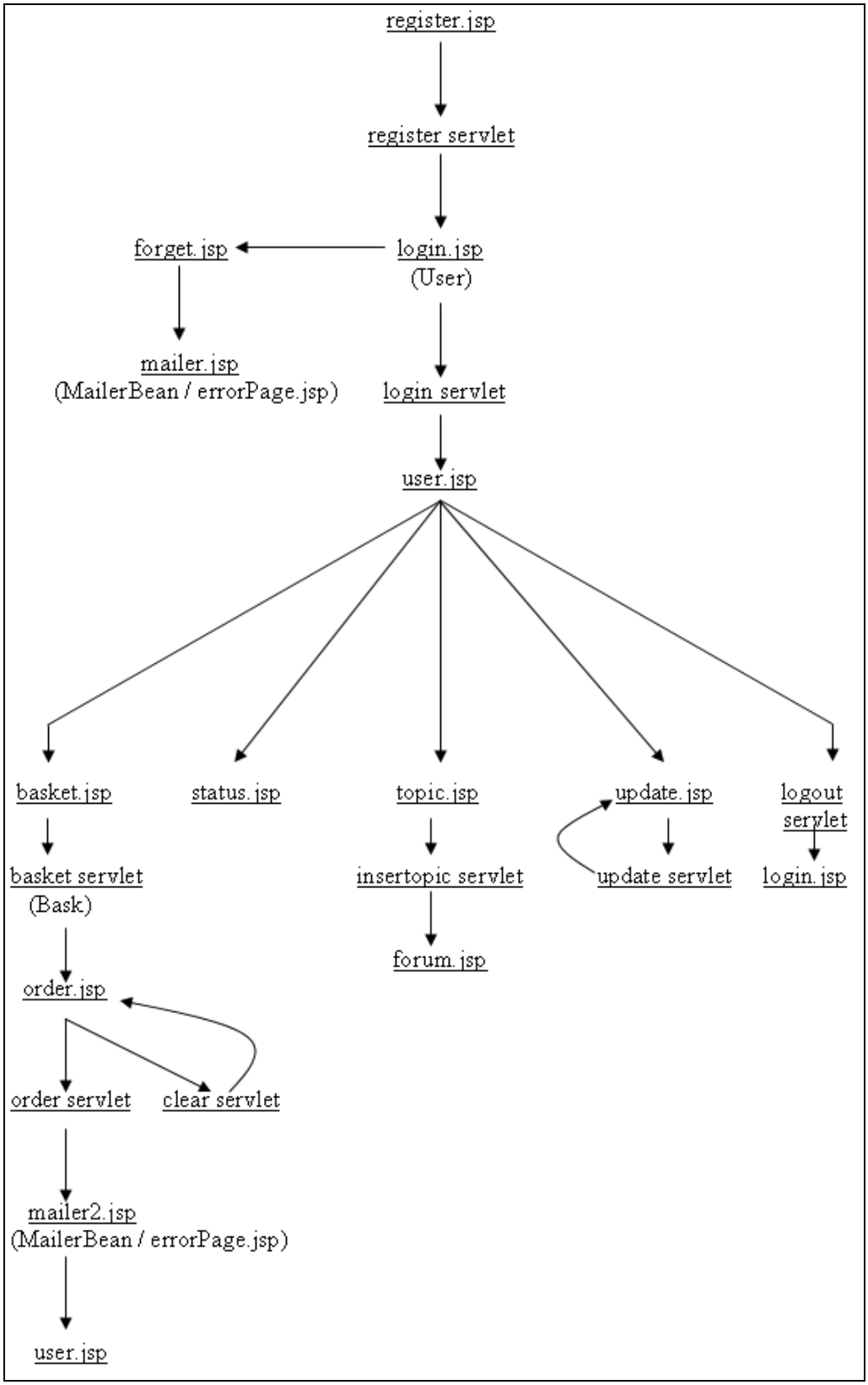
Απάντηση σε θέμα του forum



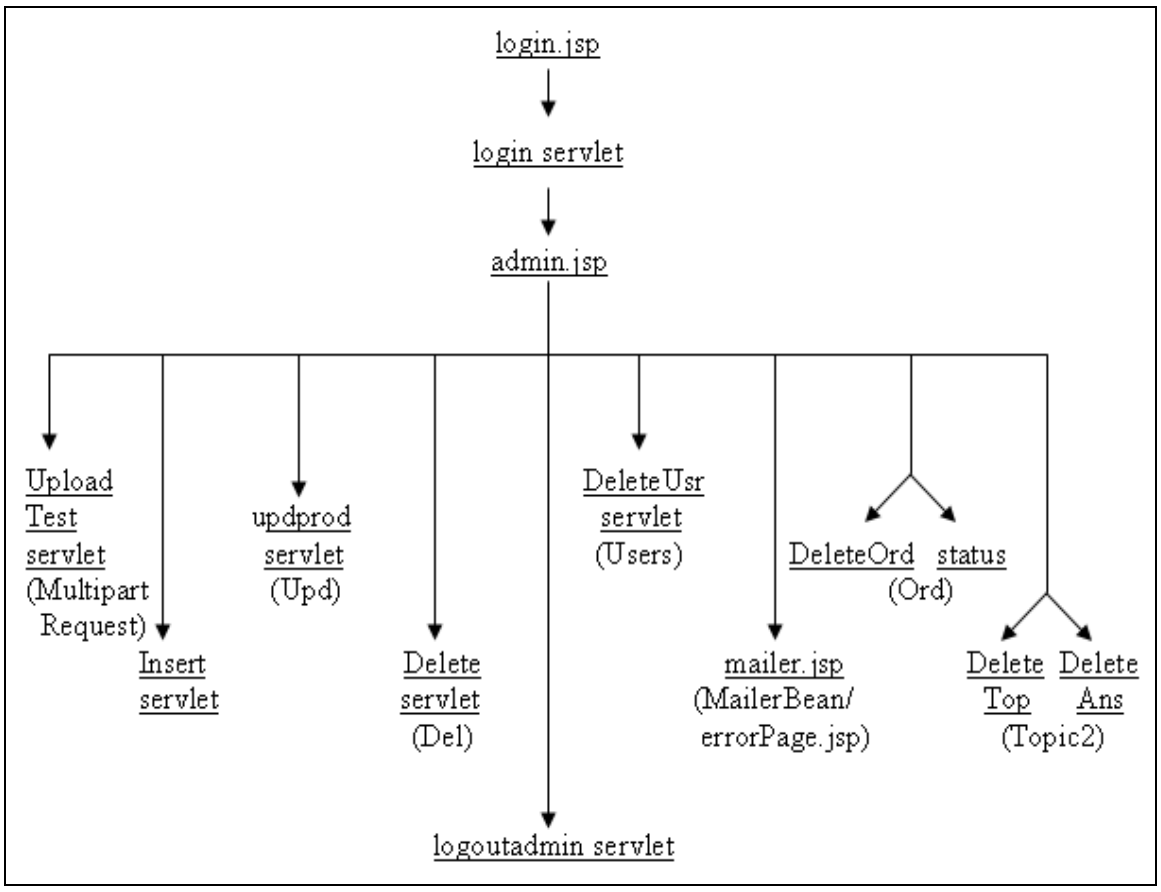
Σύνδεση



Χρήστης

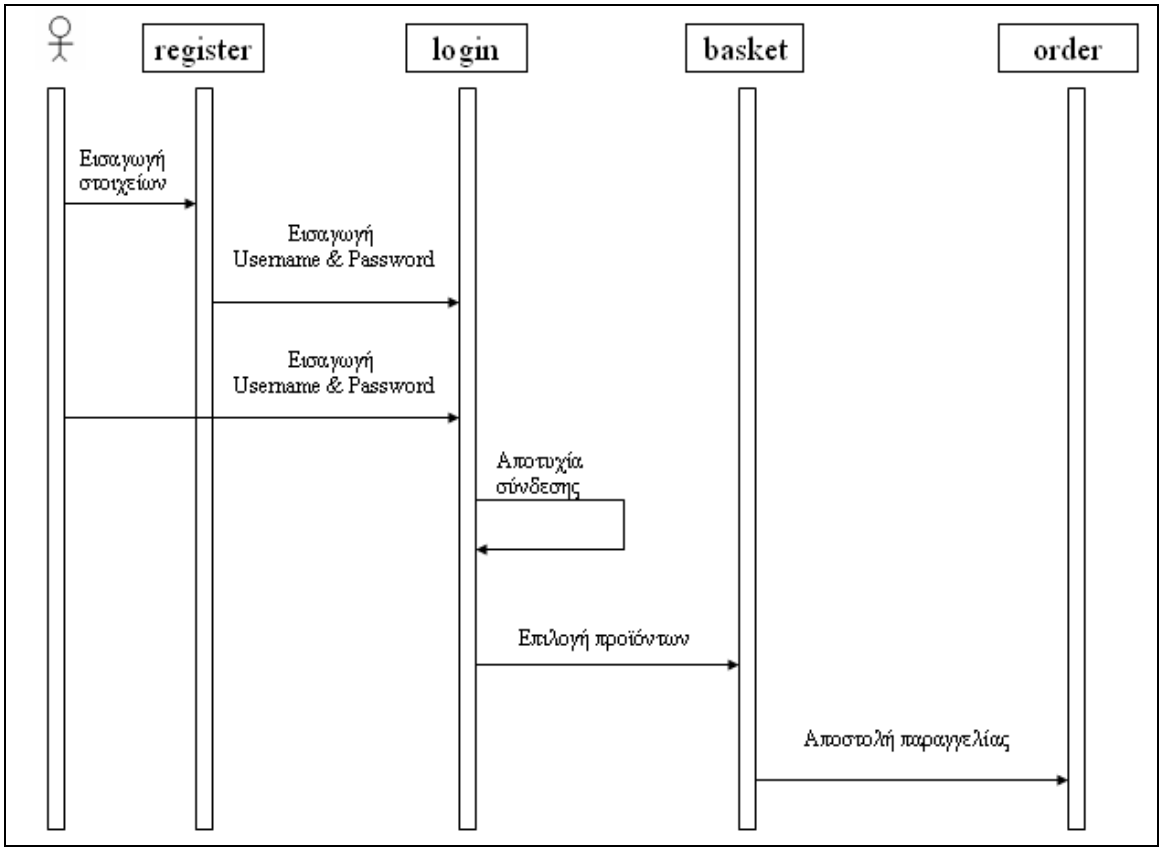

register.jsp
register servlet
forget.jsp
login.jsp
(User)
mailer.jsp
(MailerBean / errorPage.jsp)
login servlet
user.jsp
basket.jsp
status.jsp
topic.jsp
update.jsp
logout
servlet
basket servlet
(Bask)
insertopic servlet
update servlet
login.jsp
forum.jsp
order.jsp
order servlet
clear servlet
mailer2.jsp
(MailerBean / errorPage.jsp)
user.jsp

Διαχειριστής

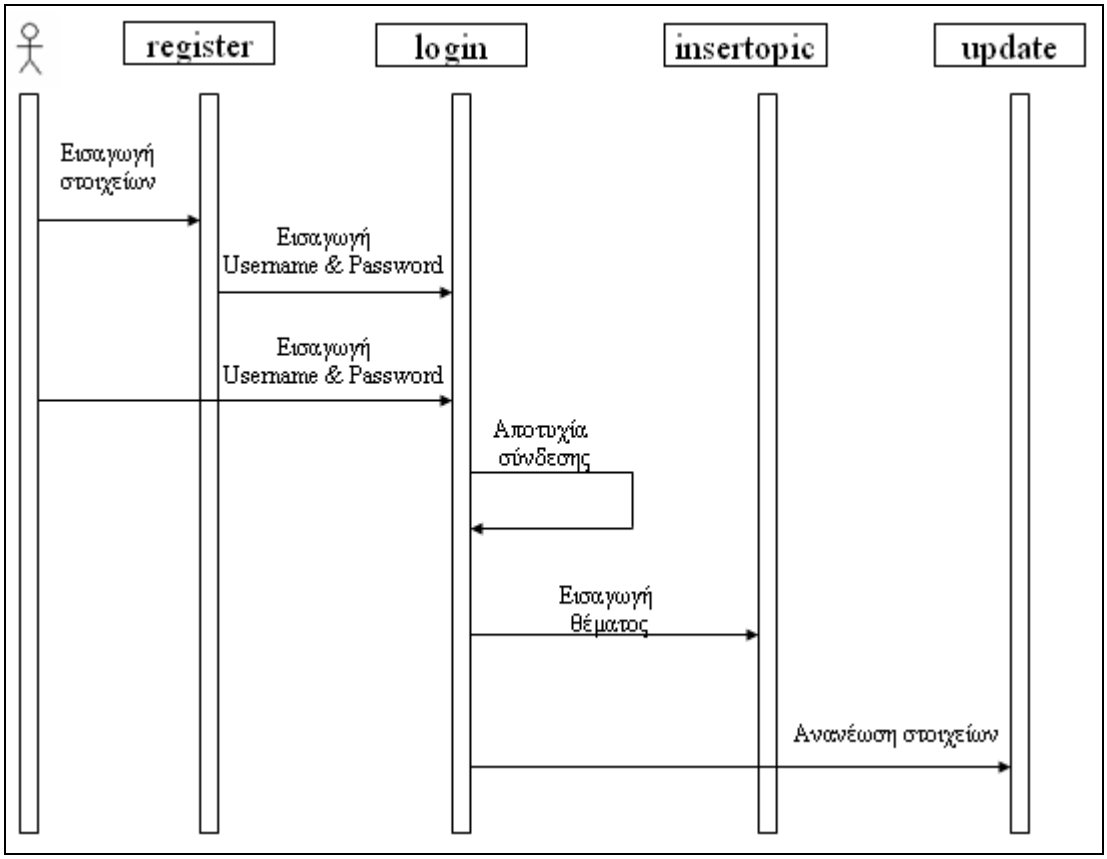


2.2. Διαγράμματα ακολουθίας

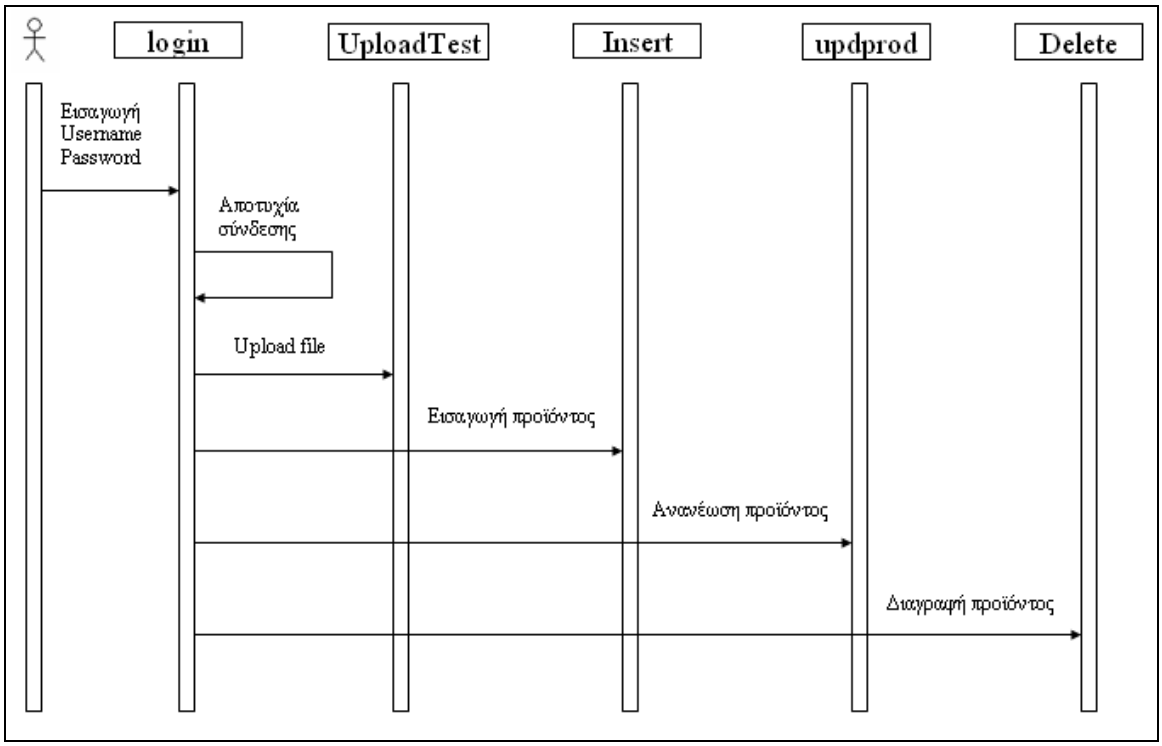
Διάγραμμα ακολουθίας για την παραγγελία προϊόντων
από εγγεγραμμένο ή μελλοντικό χρήστη



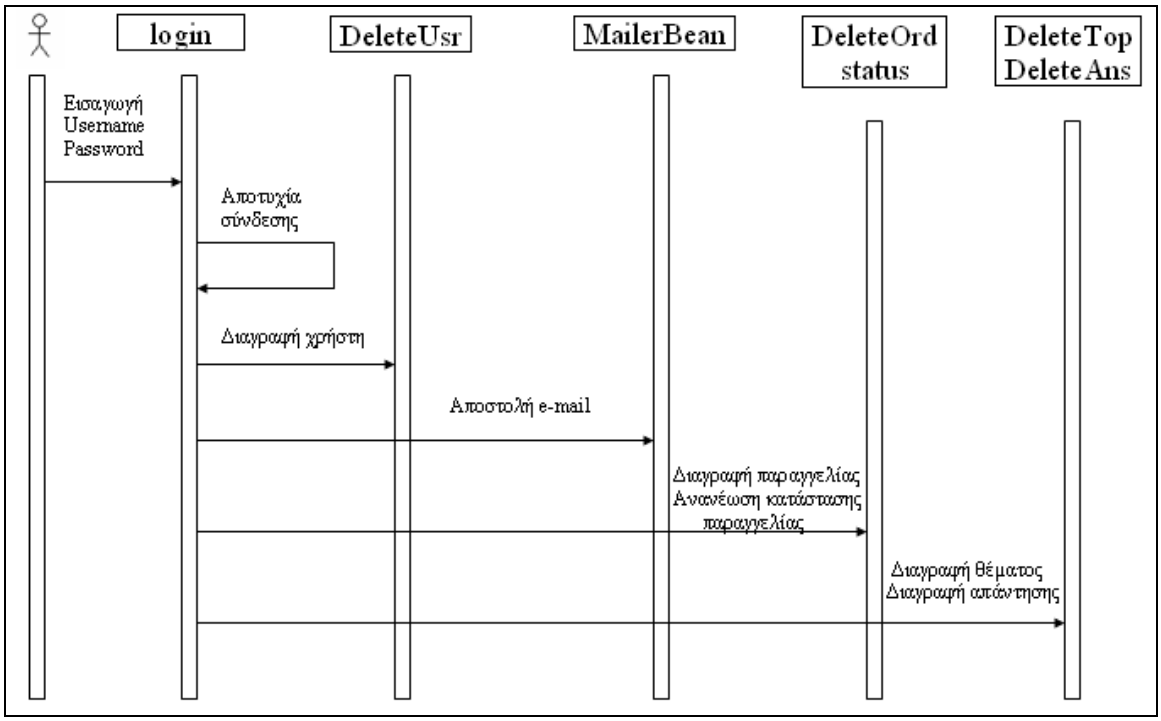
Διάγραμμα ακολουθίας για την εισαγωγή θέματος στο forum & ανανέωση των στοιχείων από εγγεγραμμένο ή μελλοντικό χρήστη



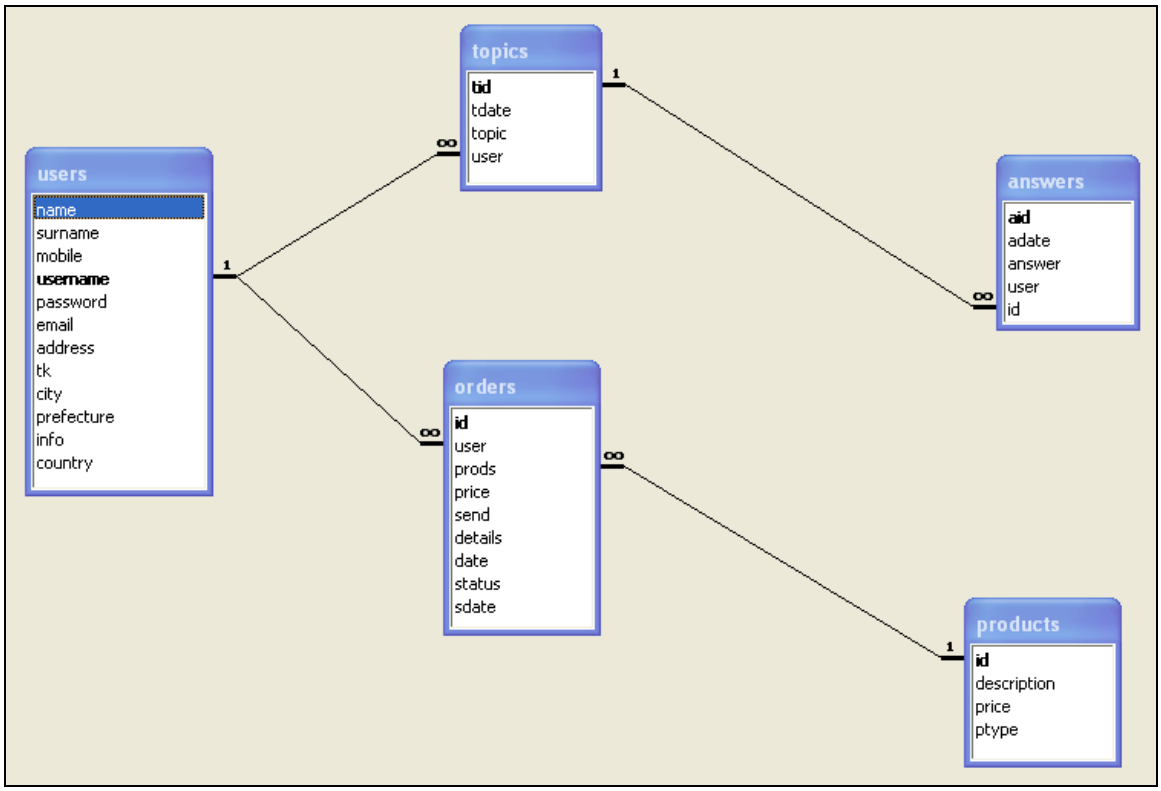
Διάγραμμα ακολουθίας για τις ενέργειες του διαχειριστή



Διάγραμμα ακολουθίας για τις ενέργειες του διαχειριστή



2.3. Διαγράμματα οντοτήτων – συσχετίσεων για την Β.Δ.



3. Αναφορά τεχνολογιών

Πριν μιλήσουμε για τις JSP σελίδες και τα JavaServlets να πούμε δυο λόγια για το αρχείο web.xml του φακέλου WEB-INF. Εντός αυτού του αρχείου δηλώνουμε τον τρόπο που θα καλούνται τα servlets. Η default μορφή είναι η …/servlet/όνομα_servlet. Εμείς αλλάζουμε το mapping ώστε να καλούνται χωρίς την ύπαρξη της λέξης servlet πριν από το όνομα. Ένα παράδειγμα μπορούμε να δούμε στη συνέχεια. Εδώ βέβαια το servlet login έχει παραμέτρους αρχικοποίησης που είναι το username και το password του διαχειριστή. Οι τιμές του δεν αποθηκεύτηκαν σε κάποιο servlet ή στη βάση δεδομένων για μεγαλύτερη ασφάλεια, ενώ θα κληθούν από το servlet login (login_en).

```
<servlet>
  <servlet-name>login</servlet-name>
  <servlet-class>login</servlet-class>

  <init-param>
    <param-name>username</param-name>
    <param-value>admin</param-value>
  </init-param>

  <init-param>
    <param-name>password</param-name>
    <param-value>admin</param-value>
  </init-param>
</servlet>
 .
 .
 .
<servlet-mapping>
  <servlet-name>login</servlet-name>
  <url-pattern>/login</url-pattern>
</servlet-mapping>
```

Επίσης δηλώνουμε το χρόνο ζωής του session και την αρχική σελίδα.

```
<session-config>
  <session-timeout>
       30
    </session-timeout>
</session-config>

<welcome-file-list>
  <welcome-file>
       index.jsp
    </welcome-file>
</welcome-file-list>
```

3.1. Περιγραφή των JSP σελίδων

Τα παρακάτω κομμάτια κώδικα είναι ίδια για όλες τις jsp σελίδες, εκτός από την σελίδα του διαχειριστή.

Αρχικά καθορίζεται η κωδικοποίηση της σελίδας ώστε να εμφανίζονται χωρίς προβλήματα οι ελληνικοί χαρακτήρες. Εισάγουμε τις τάξεις Date του πακέτου util και DateFormat του πακέτου text για την μορφοποίηση και προβολή της ημερομηνίας.

```
<%@ page import="java.util.Date, java.text.DateFormat" %>
<%@ page contentType="text/html; charset=iso-8859-7" %>
```

Σε πολλές από τις σελίδες είναι απαραίτητα και τα ακόλουθα πακέτα τάξεων.

```
<%@ page import="java.io.*" %>
<%@ page import="java.sql.*" %>
<%@ page import="javax.servlet.*" %>
<%@ page import="javax.servlet.http.*" %>
```

Εάν ο χρήστης είναι συνδεδεμένος τότε κάτω από το όνομα του χρήστη εμφανίζεται link για την αποσύνδεσή του, ενώ σε αντίθετη περίπτωση link για την σύνδεσή του.

```
String usr = (String) session.getAttribute("User");
<%
if (usr == null) {
usr = "Ανώνυμος";
}
else {}
%>
Χρήστης: <%= usr %>

<%
String user2 = (String) session.getAttribute("User");
if (user2 == null) {
%>
<a href="login.jsp">σύνδεση</a>
<%
}
else {
%>
<a href="logout">αποσύνδεση</a>
<%
}
%>
```

Με τον κάτωθι κομμάτι κώδικα μορφοποιείται και εμφανίζεται η ημερομηνία που επιστρέφεται από τον server η οποία θα είναι της μορφής: dd/mm/yyyy Αρχικά δημιουργούμε ένα αντικείμενο τύπου Date με όνομα today, το dateOut String και το dateFormatter τύπου DateFormat. Ρυθμίζουμε την μορφή που θέλουμε να έχει η ημερομηνία, την προσαρμόζουμε στο today που έχει την ημερομηνία που επιστρέφει ο server και την αποθηκεύουμε στο dateOut. Στη συνέχεια την εμφανίζουμε.

```
<%
Date today;
String dateOut;
DateFormat dateFormatter;
dateFormatter=DateFormat.getDateInstance(DateFormat.SHORT);
today=new Date();
dateOut=dateFormatter.format(today);
%>
<%= dateOut %>
```

Αφού ο χρήστης είναι συνδεδεμένος πρέπει να μπορεί να έχει πρόσβαση στη σελίδα του (user.jsp) και στη σελίδα του καλαθιού αγορών του (order.jsp) από οποιοδήποτε σελίδα και αν βρίσκεται. Στη συνέχεια μπορούμε να δούμε των κώδικα με το οποίο υλοποιείται.

```
<%
String user3 = (String) session.getAttribute("User");
if (user3 != null) {
%>
<a href="user.jsp">Σελίδα Χρήστη</a>
<a href="order.jsp">Το Καλάθι μου</a>
<%
}
%>
```

Επίσης στις περισσότερες σελίδες εμφανίζονται προειδοποιητικά μηνύματα που μπορούν να προκύψουν από ενέργειες του χρήστη. Τα μηνύματα αυτά αποθηκεύονται στο session και αφού εμφανιστούν αφαιρούνται. Αυτό επιτυγχάνεται με το εξής κώδικα :

```
<%
String errorMessage2 = (String) session.getAttribute("errorMessage2");
if (errorMessage2==null) {
%>
<%
}
else {
%>

<%= errorMessage2 %>
```

```
<%
session.removeAttribute("errorMessage2");
}
%>
```

Τέλος, σελίδες που περιέχουν το κάτωθι τμήμα κώδικα εμφανίζονται μόνο στους συνδεδεμένους χρήστες. Σε περίπτωση που κληθεί χωρίς να υπάρχει συνδεδεμένος χρήστης, τότε τα περιεχόμενα της σελίδας δεν θα εμφανιστούν και ο επισκέπτης θα ανακατευθυνθεί στη σελίδα login.jsp για να συνδεθεί. Όπως γίνεται κατανοητό η ανάγκη για έλεγχο ασφαλείας της σελίδας είναι επιτακτική γιατί σε διαφορετική περίπτωση δεν θα είχε έννοια η σύνδεση και η αποσύνδεση του χρήστη.

```
<%
String usr = (String) session.getAttribute("User");
if (usr == null) {
RequestDispatcher
rd=getServletConfig().getServletContext().getRequestDispatcher("/login.jsp");rd.forwar
d(request,response);
}
else {}
%>
```

Για να πραγματοποιηθεί η σύνδεση με την Βάση Δεδομένων (MySQL) χρησιμοποιείται από τον connector ‘mysql-connector-java-3.0.9-stable-bin.jar’ που είναι αποθηκευμένος στο φάκελο WEB-INF\lib η τάξη Driver. Στη συνέχεια δημιουργείται ένα αντικείμενο τύπου Connection όπου καλεί τη βάση μας ‘satosdb’ ως χρήστης ‘chairman’. (Λεπτομέρειες για τα δικαιώματα του χρήστη ‘chairman’ σελ. 43)

news.jsp (news_en.jsp)

Εδώ μέσω της εντολής jsp:include θα εμφανιστούν τα δεδομένα που περιέχει το αρχείο news.txt. Το αρχείο αυτό αποθηκεύεται στο server μέσω του upload που μπορεί να κάνει ο διαχειριστής. Με αυτό τον τρόπο ο διαχειριστής μπορεί να ανανεώσει δυναμικά τη σελίδα των νέων. Επίσης με την html εντολή 'pre' τα περιεχόμενα του αρχείου θα εμφανίζονται ακριβώς όπως τα έχει εισάγει ο διαχειριστής.

```
<pre><jsp:include page="files/news.txt" flush="true" /></pre>
```

forum.jsp (forum_en.jsp)

Στη σελίδα του forum εμφανίζονται θέματα τα οποία έχουν εισάγει χρήστες και δύναται η δυνατότητα ακόμα και σε απλούς επισκέπτες να απαντήσουν σε κάποιο από αυτά. Για την εμφάνισή τους απαραίτητη είναι η τάξη Topic.

```
<%@ page import="topic.Topic" %>
```

Αφού δημιουργήσουμε την σύνδεση με τη βάση μέσω της μεθόδου executeQuery του Statement αντικειμένου, καλούμε τα θέματα που έχουν εισαχθεί από τους χρήστες ταξινομημένα κατά ημερομηνία. Με την εμφάνιση κάθε θέματος θα εμφανιστούν και οι απαντήσεις του μέσω της τάξης Topic η οποία παίρνει σαν ορίσματα την σύνδεση και τον κωδικό του θέματος.

```
<%
Class.forName("org.gjt.mm.mysql.Driver");
Connection
con=DriverManager.getConnection("jdbc:mysql://localhost/satosdb?user=chairman&p
assword=satos&"+"useUnicode=true&characterEncoding=iso-8859-7");
Statement s = con.createStatement();
ResultSet rs;
String user = (String) session.getAttribute("User");
%>
<%
rs =s.executeQuery("select * from topics order by tdate;");
while (rs.next())
{
Object tdate = rs.getObject("tdate");
Object topic = rs.getObject("topic");
Object tid = rs.getObject("tid");
Object users = rs.getObject("user");
String t=tid.toString();
Topic topics=new Topic(con,t);
String result=topics.toString();
```

```
%>
<%= users %> → <a href="answer?tid=<%= tid %>"> <%= topic %> </a>
<%= tdate %>

<%= result %>

<% } %>
```

entry.jsp (entry_en.jsp)

Με τη φόρμα της σελίδας entry.jsp ο χρήστης μπορεί να απαντήσει σε θέματα του forum. Για την ενέργεια αυτή δεν είναι αναγκαία η σύνδεσή του, όπως αναφέραμε και προηγουμένως.

```
<form method="post" action='send'>
Όνομα <input type="text" name="user" value="<%= usr %>">
Απάντηση <textarea name="answer" rows="4" cols="40"></textarea>
<input type="submit" value=" Εισαγωγή ">
</form>
```

Με το παρακάτω τμήμα κώδικα ο συνδεδεμένος χρήστης δεν μπορεί να τροποποιήσει το όνομά του. Αντίθετα αν είναι απλός χρήστης έχει την δυνατότητα ή να αφήσει το default όνομα "Ανώνυμος" ή να εισάγει ένα της αρεσκείας του.

```
<% if(usr.equals("Ανώνυμος")){ %>
<input type="text" name="user" value="<%= usr %>">
<% } %>

<% if(!usr.equals("Ανώνυμος")){ %>
 <%= usr %>
<input type="hidden" name="user" value="<%= usr %>">
<% } %>
```

products.jsp (products_en.jsp)

Η σελίδα products.jsp εμφανίζει τα προϊόντα χωρίς την δυνατότητα επιλογής προϊόντος, σε αντίθεση με την basket.jsp που αναλύεται στη συνέχεια. Εδώ εισάγεται και η βοηθητική τάξη Prod που θα εμφανίσει τον πίνακα με τα προϊόντα.

```
<%@ page import="prod.Prod" %>
```

Για την εμφάνιση των προϊόντων δημιουργούμε αρχικά ένα αντικείμενο τύπου connection για την δημιουργία της σύνδεσης με τη βάση μας και στη συνέχεια με την χρήση της βοηθητικής τάξης Prod τα εμφανίζουμε.

```
<%
Class.forName("org.gjt.mm.mysql.Driver");
Connection
con=DriverManager.getConnection("jdbc:mysql://localhost/satosdb?user=chairman&p
assword=satos&"+"useUnicode=true&characterEncoding=iso-8859-7");

String que="select * from products order by ptype,id;";
Prod prods=new Prod(con,que);
String result=prods.toString();
%>
<%= result %>
```

register.jsp (register_en.jsp)

Μέσω της σελίδας register.jsp ο επισκέπτης μπορεί να εγγραφεί χρήστης. Αν υπάρχει συνδεδεμένος χρήστης η σελίδα δεν θα εμφανιστεί και θα κατευθυνθεί στην update.jsp (Λεπτομέρειες σελ. 37), αλλιώς στο String ‘usr’ θα αποθηκευτεί η τιμή ‘Ανώνυμος’ η οποία και θα εμφανιστεί πάνω από το link της σύνδεσης.

```
<%
String usr = (String) session.getAttribute("User");
if (usr != null) {
RequestDispatcher
rd=getServletConfig().getServletContext().getRequestDispatcher("/update.jsp");rd.forwa
rd(request,response);
}
else {
usr = "Ανώνυμος";
}
%>
```

Με την παρακάτω φόρμα ο χρήστης εισάγοντας τα προσωπικά του δεδομένα μπορεί να εγγραφεί.

```
<form method="post" action='register'>
Εγγραφή χρήστη

Προσωπικά στοιχεία
Όνομα<input type="text" name="name">
Επώνυμο<input type="text" name="surname">
```

Κινητό<input type="text" name="mobile">
Όνομα χρήστη<input type="text" name="username">
Κωδικός χρήστη<input type="password" name="password">
Επανάληψη κωδικού<input type="password" name="password2">
e-mail<input type="text" name="email">

Αποστολή παραγγελίας
Διεύθυνση<input type="text" name="address">
T.K.<input type="text" name="tk">
Πόλη<input type="text" name="city">
Νομός<input type="text" name="prefecture">
Χώρα<input type="text" name="country">
<input type="checkbox" name="info" checked>Επιθυμώ να λαμβάνω πληροφορίες για προσφορές και νέα προϊόντα μέσω e-mail.
<input class="button" type="submit">
<input class="button" type="reset">
</form>

forget.jsp (forget_en.jsp)

Μέσω της σελίδας αυτής δύναται η δυνατότητα στον χρήστη να του αποσταλεί στο e-mail του ο κωδικός σύνδεσης. Για να εμφανιστεί η σελίδα δεν πρέπει να υπάρχει συνδεδεμένος χρήστης. Έτσι γίνεται έλεγχος αν υπάρχει συνδεδεμένος χρήστης και αν υπάρχει κατευθύνεται στη σελίδα update.jsp. Αυτό συμβαίνει γιατί ο χρήστης που έχει συνδεθεί δεν μπορεί συγχρόνως να έχει ξεχάσει και τον κωδικό του. Η παραπάνω διαδικασία γίνεται με τον παρακάτω κώδικα :

```
<%
String usr = (String) session.getAttribute("User");
if (usr != null) {
RequestDispatcher
rd=getServletConfig().getServletContext().getRequestDispatcher("/update.jsp");rd.forwa
rd(request,response);
}
else {
usr = "Ανώνυμος";
}
%>
```

Στη πρώτη φόρμα ο χρήστης εισάγει το username του και πατώντας το κουμπί υποβολής γίνεται αναζήτηση των στοιχείων από τη βάση για το συγκεκριμένο username. Στην περίπτωση που υπάρχει στη βάση χρήστης με αυτό το username από τις τιμές που θα επιστραφούν αποθηκεύουμε το password και το e-mail σε μεταβλητές τύπου string. Να σημειώσουμε ότι η φόρμα δεν έχει action και ότι η μέθοδος που χρησιμοποιείται είναι η

get και όχι η post. Με αυτό τον τρόπο αποφεύγουμε τη δημιουργία ενός επιπλέον servlet μιας και τα δεδομένα θα επιστραφούν στη παρούσα jsp σελίδα.

```
<form method="get">
<%
Class.forName("org.gjt.mm.mysql.Driver");
Connection
con=DriverManager.getConnection("jdbc:mysql://localhost/satosdb?user=chairman&p
assword=satos&"+"useUnicode=true&characterEncoding=iso-8859-7");
String username = request.getParameter("username");
String password="";
String email="[your e-mail]";

Statement s = con.createStatement();
ResultSet rs =s.executeQuery("select * from users where username='"+username+"';");
if (rs.next())
{
        password = rs.getString("password");
        email = rs.getString("email");
}
%>

Όνομα χρήστη <input type="text" name="username">
<input type="submit" value="  Check  ">
</form>
```

Το e-mail που επιστρέφεται από την προηγούμενη φόρμα θα εμφανιστεί για να πληροφορήσει τον χρήστη που θα αποσταλεί ο κωδικός. Η τιμή του μαζί με την τιμή του κωδικού θα αποθηκευτούν σε πεδία τύπου hidden και μέσω της νέας φόρμας θα αποσταλούν στη σελίδα mailer.jsp (Λεπτομέρειες σελ. 25) η οποία είναι υπεύθυνη να στείλει την τιμή του κωδικού στο e-mail του χρήστη.

```
<form method="post" action='mailer.jsp'>
Ο κωδικός σας θα αποσταλεί στο : <%= email %></td>

<input type="hidden" name="from" value="satos@satos.gr">
<input type="hidden" name="to" value="<%= email %>">
<input type="hidden" name="subject" value="satos.gr - login password">
<input type="hidden" name="message"
        value="Your password is : <%= password %>">

<input type="submit" class="button" value="Confirm">
</form>
```

errorPage.jsp

```
<%@ page isErrorPage="true" %>
<%= exception.getMessage() %>
```

Η σελίδα αυτή θα εμφανίσει μήνυμα όταν από το servlet MailerBean δημιουργηθεί κάποια εξαίρεση.

mailer.jsp

Η σελίδα mailer.jsp στην περίπτωση που θα προκύψει κάποια εξαίρεση, δηλαδή η errorPage.jsp θα περιέχει κάποιο μήνυμα, θα εμφανίσει το περιεχόμενο της errorPage.jsp ενώ σε διαφορετική περίπτωση θα πραγματοποιηθεί η λειτουργία του MailerBean servlet (Λεπτομέρειες σελ. 48) και θα εμφανίσει τα περιεχόμενά της ('*Email has been sent successfully.*').

```
<%@ page errorPage="errorPage.jsp" %>
<jsp:useBean id="mailer" class="com.stardeveloper.bean.test.MailerBean">
<jsp:setProperty name="mailer" property="*"/>
<% mailer.sendMail(); %>
</jsp:useBean>
Email has been sent successfully.
```

login.jsp (login_en.jsp)

Η σελίδα login.jsp είναι η σελίδα που καλείται για τη σύνδεση του χρήστη / διαχειριστή. Στη συνέχεια γίνεται έλεγχος αν υπάρχει συνδεδεμένος χρήστης και αν υπάρχει κατευθύνεται στη σελίδα update.jsp. Διαφορετικά εμφανίζεται η σελίδα login.jsp με την ένδειξη 'Ανώνυμος' στο όνομα του χρήστη (Χρήστης: <%= usr %>) . Η παραπάνω διαδικασία γίνεται με τον παρακάτω κώδικα :

```
<%
String usr = (String) session.getAttribute("User");
if (usr != null) {
RequestDispatcher
rd=getServletConfig().getServletContext().getRequestDispatcher("/update.jsp");rd.forwa
rd(request,response);
}
else {
usr = "Ανώνυμος";
} %>
```

Τέλος, μέσω της παρακάτω φόρμας ο χρήστης ή ο διαχειριστής συνδέονται εισάγοντας το 'Όνομα χρήστη' και τον 'Κωδικό' τους. Η ταυτοποίηση των τιμών που εισάγονται γίνεται στο servlet login (Λεπτομέρειες σελ. 46). Επίσης δίνεται η δυνατότητα στους χρήστες να τους αποστέλλεται στο e-mail τους ο κωδικός σε περίπτωση που τον ξεχάσουν. Για τη λειτουργία αυτή καλείται η forget.jsp. (Λεπτομέρειες σελ. 23)

```
<form method="post" action='login'>
Όνομα χρήστη <input type="text" name="username">
Κωδικός χρήστη <input type="password" name="password">
<input type="submit">
<a href="forget.jsp">Ξεχάσατε τον κωδικό σας?</a>
</form>
```

admin.jsp

Στη σελίδα του διαχειριστή της εφαρμογής αρχικά έχουμε εισάγει τις απαραίτητες βοηθητικές τάξεις.

```
<%@ page import="del.Del" %>
<%@ page import="upd.Upd" %>
<%@ page import="users.Users" %>
<%@ page import="ord.Ord" %>
<%@ page import="topic2.Topic2" %>
```

Η σελίδα αυτή εμφανίζεται εφόσον ο διαχειριστής κάνει login από την σελίδα login.jsp. Σε περίπτωση αποτυχίας σύνδεσης θα ανακατευθυνθούμε στην login.jsp σελίδα. Στη συνέχεια και αφού καλεστεί το servlet login γίνεται έλεγχος των τιμών που δόθηκαν, με τις τιμές που είναι αποθηκευμένες στο αρχείο web.xml (Λεπτομέρειες σελ. 16).

Αφού επαληθευτεί η ταυτοποίηση εκχωρείτε στο session με το όνομα admin η τιμή του ‘username’ (admin) και ανακατευθυνόμαστε στην admin.jsp. Αφού γίνει ο έλεγχος, δηλαδή ότι η τιμή του admin από το session δεν είναι null, εμφανίζεται η σελίδα.

```
<%
String usr = (String) session.getAttribute("admin");
if (usr == null) {
RequestDispatcher
rd=getServletConfig().getServletContext().getRequestDispatcher("/login.jsp");rd.forwar
d(request,response);
}
else  {  }
%>
```

Ο διαχειριστή από την σελίδα του μπορεί να προβεί σε 8 ενέργειες :

1. Upload file / image

Τα δεδομένα του αρχείου θα εμφανιστούν στη σελίδα news.jsp (news_en.jsp).
Αυτό επιτυγχάνεται με την κλήση του servlet UploadTest. (Λεπτομέρειες σελ. 50)

```
<FORM ACTION="UploadTest" ENCTYPE="multipart/form-data" METHOD=POST>
Αναζήτηση αρχείου...<INPUT TYPE=FILE NAME=file>
<input type="submit" value=" Upload ">
</FORM>
```

Το servlet UploadTest χρησιμοποιεί σαν βοηθητικό το servlet MultipartRequest.
Επίσης ο διαχειριστής έχει την δυνατότητα να κάνει ‘upload’ φωτογραφίες νέων προϊόντων δίνοντάς τους σαν όνομα τον κωδικό του προϊόντων με την κατάληξη .JPG

2. Εισαγωγή προϊόντος στη βάση.

Καλείται το servlet Insert. (Λεπτομέρειες σελ. 52)

```
<form method="post" action='Insert'>
ID<input type="text" name="id">
Description<textarea name="desc" rows="6" cols="29"></textarea>
Price<input type="text" name="price" value="0">
Type<input type="text" name="ptype">
<input type="submit" value="Insert">
</form>
```

3. Ανανέωση προϊόντος από τη βάση.

Καλείται το servlet updprod. (Λεπτομέρειες σελ. 54)
Για την εμφάνισή τους χρησιμοποιείται σαν βοηθητικό το servlet Upd. (Λεπτομέρειες σελ. 52) Με την χρήση checkbox μπορεί να γίνει ανανέωση σε περισσότερα από ένα προϊόντα.

```
<%
Class.forName("org.gjt.mm.mysql.Driver");
Connection
con=DriverManager.getConnection("jdbc:mysql://localhost/satosdb?user=chairman&p
assword=satos&"+"useUnicode=true&characterEncoding=iso-8859-7");

String quest="select * from products order by ptype,id;";
Upd uprods=new Upd(con,quest);
String resultpr=uprods.toString();
%>

<%= resultpr %>
```

4. Διαγραφή προϊόντος από τη βάση.

Καλείται το servlet Delete. (Λεπτομέρειες σελ. 57)
Για την εμφάνισή τους χρησιμοποιείται σαν βοηθητικό το servlet Del. (Λεπτομέρειες σελ. 55) Με την χρήση checkbox μπορεί να γίνει διαγραφή σε περισσότερα από ένα προϊόντα.

```
<%
Class.forName("org.gjt.mm.mysql.Driver");
Connection
con=DriverManager.getConnection("jdbc:mysql://localhost/satosdb?user=chairman&p
assword=satos");

String que="select * from products order by ptype;";
Del prods=new Del(con,que);
String result=prods.toString();
%>
<%= result %>
```

5. Εμφάνιση των χρηστών και δυνατότητα διαγραφή τους.

Για την εμφάνιση τους χρησιμοποιούμε την τάξη Users. (Λεπτομέρειες σελ. 57)
ενώ την διαγραφή τους την τάξη DeleteUsr. (Λεπτομέρειες σελ. 59) Με την χρήση checkbox μπορεί να γίνει ανανέωση σε περισσότερους από έναν χρήστες.

```
<%
String que2="select * from users order by surname;";
Users users=new Users(con,que2);
String result2=users.toString();
%>
<%= result2 %>
```

6. Αποστολή e-mail στους χρήστες που θέλουν να δέχονται ενημερωτικά μηνύματα.

Καλείται το η σελίδα mailer.jsp. (Λεπτομέρειες σελ. 25) Να τονίσουμε ότι θα εμφανιστούν τα ονόματα των χρηστών εκείνων που έχουν ενεργοποιημένη την υπηρεσία λήψης ενημερωτικών e-mail.

```
<form action="mailer.jsp" method="post">
<%
String name="";
String surname="";
String email="";
Statement s = con.createStatement();
ResultSet rs =s.executeQuery("select * from users where info='on' order by surname;");
while(rs.next())
{
        name = rs.getString("name");
        surname = rs.getString("surname");
        email = rs.getString("email");
%>
<%= surname %>  <%= name %>  <%= email %>
<%
}
%>

From : satos@satos.gr <input type="hidden" name="from" value="satos@satos.gr">
To   : <input type="text" name="to">
Subject  : <input type="text" name="subject">
Message : <textarea rows="10" cols="80" name="message"></textarea>
<input type="submit" value="Dispatch">
</form>
```

7. Επεξεργασία παραγγελιών.

Έχει την δυνατότητα να βλέπει τις παραγγελίες, να ανανεώνει την κατάστασή τους για την ενημέρωση των πελατών ή ακόμα και να τις διαγράψει.

Σαν βοηθητική τάξη για την εμφάνισή τους χρησιμοποιείται η Ord. (Λεπτομέρειες σελ. 60) Για την ανανέωση των καταστάσεων των παραγγελιών χρησιμοποιείται το servlet status, (Λεπτομέρειες σελ. 60) ενώ για της διαγραφή τους το servlet DeleteOrd. (Λεπτομέρειες σελ. 61)

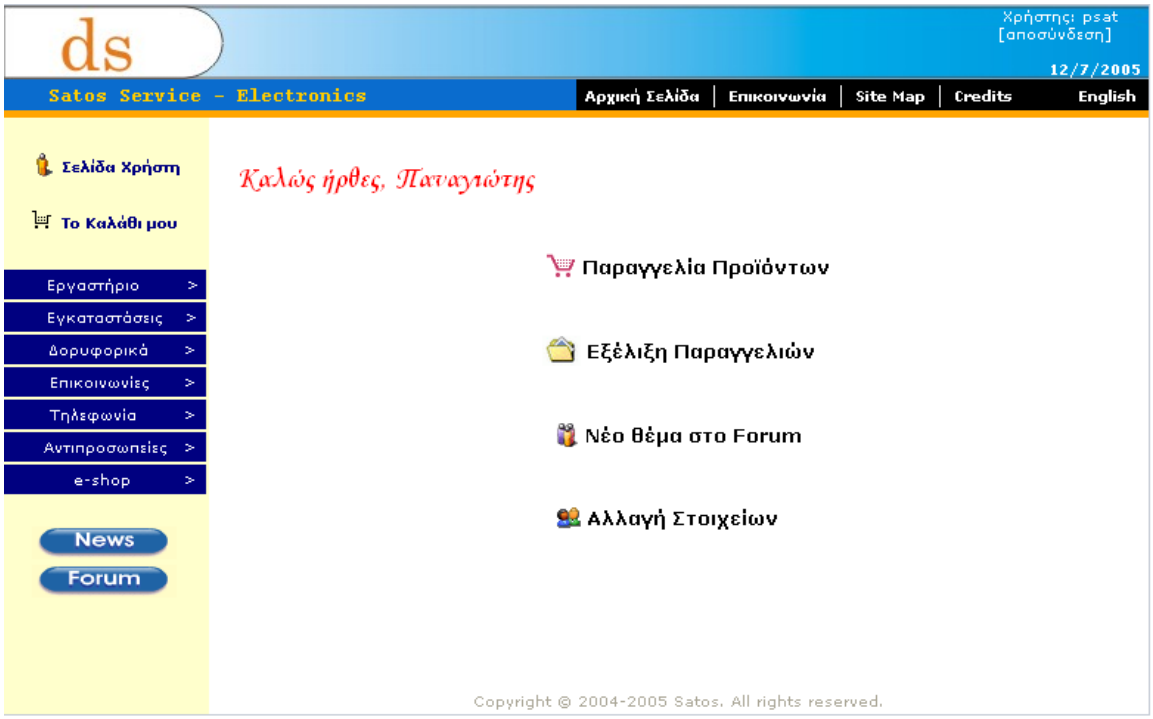
```
<%
String que3="select * from orders order by date desc;";
Ord ord=new Ord(con,que3);
String result3=ord.toString();
%>

<%= result3 %>
```

8. Επεξεργασία των θεμάτων και των απαντήσεων του forum.

Σαν βοηθητική τάξη για την εμφάνισή τους χρησιμοποιείται η Topic2. (Λεπτομέρειες σελ. 61) Για την διαγραφή των θεμάτων χρησιμοποιείται το servlet DeleteTop, (Λεπτομέρειες σελ. 61) ενώ για της διαγραφή των απαντήσεων το servlet DeleteAns. (Λεπτομέρειες σελ. 62) Να επισημάνουμε ότι αν διαγραφεί ένα θέμα, τότε αυτομάτως διαγράφονται και οι απαντήσεις του.

```
<%
rs =s.executeQuery("select * from topics order by tdate;");
while (rs.next())
{
Object topic = rs.getObject("topic");
Object tid = rs.getObject("tid");
Object users1 = rs.getObject("user");

String t=tid.toString();
Topic2 topics=new Topic2(con,t);
String results=topics.toString();
%>
<%= users1 %>  <%= topic %> <a href="DeleteTop?tid=<%= tid %>">delete</a>
<%= results %>
<%
}
%>
```

user.jsp **(user_en.jsp)**



Η user.jsp είναι η σελίδα του χρήστη. Ο χρήστης μπορεί να την καλέσει από οποιαδήποτε σελίδα επιλέγοντας το αντίστοιχο link. Αρχικά δημιουργούμε μία σύνδεση με τη βάση δεδομένων και αντικείμενα τύπου ResultSet, Statement και με τη μέθοδο executeQuery() του δευτέρου εκτελούμε μία sql εντολή. Τα δεδομένα που θα επιστραφούν αποθηκεύονται στο ResultSet και στη συνέχεια τα εμφανίζουμε. Στη συγκεκριμένη σελίδα επιστρέφουμε από τη βάση την τιμή του ονόματος του χρήστη.

```
<%
Class.forName("org.gjt.mm.mysql.Driver");
Connection
con=DriverManager.getConnection("jdbc:mysql://localhost/satosdb?user=chairman&p
assword=satos&"+"useUnicode=true&characterEncoding=iso-8859-7");
Statement s = con.createStatement();
ResultSet rs;
String name="";

rs=s.executeQuery("select * from users where username='"+usr+"';");
while(rs.next())
{
        name = rs.getString("name");
}
%>
Καλώς ήρθες,<%= name %>
```

Από τη σελίδα του ο χρήστης μπορεί να παραγγείλει προϊόντα, να δει την εξέλιξη των παραγγελιών του, να εισάγει θέμα στο forum και να ανανεώσει τα προφίλ του. Οι παραπάνω ενέργειες θα αναπτυχθούν παρακάτω.

Παραγγελία Προϊόντων
Εξέλιξη Παραγγελιών
Νέο θέμα στο Forum
Αλλαγή Στοιχείων

order.jsp (order_en.jsp)

Η order.jsp είναι η σελίδα του καλαθιού αγορών. Όπως και για τη σελίδα του χρήστη έτσι και γι' αυτήν υπάρχει link ώστε να μπορεί από οποιαδήποτε σελίδα να την καλέσει ο χρήστης. Μέσω του παρακάτω link ο χρήστης μπορεί να επιστρέψει στη σελίδα basket.jsp αν θέλει να προσθέσει προϊόντα στο καλάθι.

Συνεχίστε τις αγορές σας

Μέσω της ακόλουθης φόρμας επιτυγχάνουμε το άδειασμα του καλαθιού αγορών. Υπεύθυνο servlet γι'αυτή την ενέργεια είναι το clear (Λεπτομέρειες σελ. 67).

<form method="POST" action="clear">
<input type="submit" value="Αδειασμα">
</form>

Η παρακάτω φόρμα καλεί το servlet order το οποίο θα αναλάβει την αποθήκευση της παραγγελίας του χρήστη στη βάση δεδομένων. Μηνύματα θα ενημερώνουν τον χρήστη για το αν το καλάθι είναι άδειο, ενώ στην περίπτωση που περιέχει προϊόντα εμφανίζονται οι ποσότητες, τα προϊόντα και η τιμή. Αν χρήστης προσπαθήσει να στείλει παραγγελία χωρίς να περιέχει προϊόντα το καλάθι, η αποστολή της παραγγελίας θα αποτύχει και ο χρήστης θα ενημερωθεί με σχετικό μήνυμα. Αν όλα πάνε καλά, μαζί με την παραγγελία μέσω hidden πεδίων, θα αποσταλούν το e-mail του διαχειριστή και η ημερομηνία παραγγελίας ώστε να σταλεί ειδοποιητικό e-mail στον διαχειριστή.

<form method="POST" action="order">

<%
String Message3 = (String) session.getAttribute("Message3");
if (Message3==null) {
%>
<%
}
else {
%>

```
<%= Message3 %>
<%
session.removeAttribute("Message3");
}
%>

<%
String Message2 = (String) session.getAttribute("Message2");
if (Message2==null) {
%>

Το καλάθι είναι άδειο!

 <%
}
else {
%>

Τα Προϊόντα σου:  <%= Message2 %>

<%
}
%>

<%
String Message = (String) session.getAttribute("Message");
if (Message==null) {
%>
<%
}
else {
%>
Τελική Τιμή: <%= Message %> Euro + 19%<%
}
%>
```

Ακόμα ο χρήστης μπορεί να επιλέξει το μέσο αποστολής και να προσθέσει και λεπτομέρειες που θεωρεί σημαντικές για την αποστολή των προϊόντων. (π.χ. εταιρεία courier)

```
Τρόπος Αποστολής:
<select name="send">
<option value="1">Ταχυδρομείο</option>
<option value="2">Courier</option>
<option value="3">Μεταφορική</option>
</select>
```

Λεπτομέρειες: <input type="text" name="details">

<input type="hidden" name="from" value="satos@satos.gr">
<input type="hidden" name="to" value="psatos@satos.gr">
<input type="hidden" name="subject" value="New order!">
<input type="hidden" name="message" value="Order date: <%= dateOut %>">

<input type="submit">
</form>

mailer2.jsp (mailer2_en.jsp)

Και η σελίδα mailer2.jsp στην περίπτωση που θα προκύψει κάποια εξαίρεση θα εμφανίσει το περιεχόμενο της errorPage.jsp ενώ σε διαφορετική περίπτωση θα πραγματοποιηθεί η λειτουργία του MailerBean servlet (Λεπτομέρειες σελ. 48) και θα εμφανίσει τα περιεχόμενά της ('*Data process...*') και ο χρήστης θα επιστρέψει στη σελίδα του.
('*<META HTTP-EQUIV="Refresh" CONTENT="2; URL=user.jsp">*').

<%@ page errorPage="errorPage.jsp" %>
<META HTTP-EQUIV="Refresh" CONTENT="2; URL=user.jsp">
<jsp:useBean id="mailer" class="com.stardeveloper.bean.test.MailerBean">
<jsp:setProperty name="mailer" property=""/>*
<% mailer.sendMail(); %>
</jsp:useBean>
Data process...

basket.jsp (basket_en.jsp)

Η σελίδα αυτή εμφανίζει τα προϊόντα με δυνατότητα επιλογής προϊόντος, σε αντίθεση με την products.jsp που αναλύσαμε προηγουμένως. Εδώ εισάγεται η βοηθητική τάξη Bask που θα εμφανίσει τον πίνακα με τα προϊόντα.

<%@ page import="bask.Bask" %>

Δημιουργούμε μια φόρμα μέσω της οποίας ο χρήστης θα μπορεί να στέλνει τα προϊόντα στο καλάθι. Για την εμφάνιση των προϊόντων δημιουργούμε αρχικά ένα αντικείμενο τύπου connection για την δημιουργία της σύνδεσης με τη βάση μας και στη συνέχεια με την χρήση της βοηθητικής τάξης Bask τα εμφανίζουμε. Με την υποβολή της φόρμας θα καλεστεί το servlet basket, μέλημα του οποίου είναι η εισαγωγή των προϊόντων που επέλεξε ο χρήστης στο καλάθι αγορών.

```
<form method="post" action='basket'>
<%
Class.forName("org.gjt.mm.mysql.Driver");
Connection
con=DriverManager.getConnection("jdbc:mysql://localhost/satosdb?user=chairman&p
assword=satos&"+"useUnicode=true&characterEncoding=iso-8859-7");
String que="select * from products order by ptype,id;";
Bask basks=new Bask(con,que);
String result=basks.toString();
%>
<%= result %>
<input type="submit" value="Προσθήκη">
</form>
```

status.jsp (status_en.jsp)

Από εδώ ο χρήστης έχει την δυνατότητα να ελέγξει την εξέλιξη των παραγγελιών του. Αφού συνδεθούμε με τη βάση, επιστρέφουμε το όνομα του χρήστη που είναι συνδεδεμένος.

```
<%
Class.forName("org.gjt.mm.mysql.Driver");
Connection
con=DriverManager.getConnection("jdbc:mysql://localhost/satosdb?user=chairman&p
assword=satos&"+"useUnicode=true&characterEncoding=iso-8859-7");
Statement s = con.createStatement();
ResultSet rs;
String name="";
String id="";
String status="";
String sdate="";

rs=s.executeQuery("select * from users where username='"+usr+"';");
while(rs.next())
{
        name = rs.getString("name");
}

%>

Καλώς ήρθες, <%= name %>
```

Στη συνέχεια εμφανίζουμε από τον πίνακα των παραγγελιών (orders) τους κωδικούς, την κατάσταση που βρίσκονται οι παραγγελίες καθώς και την ημερομηνία της αλλαγής της

κατάστασης. Η ταξινόμηση γίνεται κατά φθίνουσα σειρά της ημερομηνίας. Με αυτό τον τρόπο δίνεται η δυνατότητα στον χρήστη να βλέπει την εξέλιξη των παραγγελιών.

Κωδικός - Κατάσταση - Ημερομηνία

```
<%
rs=s.executeQuery("select * from orders where user='"+usr+"' order by sdate desc;");

while(rs.next())
{
        id = rs.getString("id");
        status = rs.getString("status");
        sdate = rs.getString("sdate");
%>

<%= id %> <%= status %> <%= sdate %>

<%
}
%>
```

Ο χρήστης θα μπορεί να δει τις λεπτομέρειες της παραγγελίας πατώντας τον κωδικό της. Αυτό επιτυγχάνεται ως εξής: πατώντας ο χρήστης κάποιον κωδικό παραγγελίας, ο κωδικός αυτός θα αποσταλεί με την μέθοδο GET στην τρέχουσα σελίδα, δηλαδή την status.jsp. Έπειτα θα αποθηκευτεί και θα χρησιμοποιηθεί στην SQL εντολή και έτσι θα εμφανιστούν τα υπόλοιπα στοιχεία της παραγγελίας με τον κωδικό αυτό.

```
<%
String id2="";
Object prods;
Object price;
Object send;
Object details;
Object date;
id2=request.getParameter("id2");
%>
```

Προϊόντα - Τιμή - Αποστολή - Λεπτομέρειες – Καταχωρήθηκε

```
<%
rs=s.executeQuery("select * from orders where id='"+id2+"';");

while(rs.next())
{
        prods = rs.getObject("prods");
        price = rs.getObject("price");
```

```
        send = rs.getObject("send");
        details = rs.getObject("details");
        date = rs.getObject("date");
%>
<%= prods %><%= price %><%= send %><%= details %><%= date %>
<%
}
%>
```

topic.jsp (topic_en.jsp)

Από αυτήν τη σελίδα ο χρήστης μπορεί να εισάγει ένα νέο θέμα στη σελίδα του forum (forum.jsp). Μέσω της παρακάτω φόρμας επιτυγχάνεται η παραπάνω λειτουργία.

```
<form method="post" action='insertopic'>
Νέο θέμα <textarea name="topic" rows="3" cols="40"></textarea>
<input type="submit" value=" Εισαγωγή ">
</form>
```

update.jsp (update_en.jsp)

Μπορεί να παρουσιαστεί η ανάγκη κάποιος χρήστης να θέλει να ανανεώσει τα στοιχεία του. Αυτό θα επιτευχθεί από την σελίδα update.jsp.

Αφού συνδεθεί με την βάση καλεί τα δεδομένα που αντιστοιχούν στον συνδεδεμένο χρήστη. Θα αποθηκευτούν και θα εμφανιστούν δίνοντας την δυνατότητα στο χρήστη να τα τροποποιήσει και να τα αποθηκεύσει καλώντας το servlet update (Λεπτομέρειες σελ. 69).

```
<%
String user1 = (String) session.getAttribute("User");
Class.forName("org.gjt.mm.mysql.Driver");
Connection
con=DriverManager.getConnection("jdbc:mysql://localhost/satosdb?user=chairman&p
assword=satos&"+"useUnicode=true&characterEncoding=iso-8859-7");
Statement s = con.createStatement();
ResultSet rs;
String name="";
String surname="";
String mobile="";
String username="";
String password="";
```

```
String email="";
String address="";
String tk="";
String city="";
String prefecture="";
String country="";
String info="";

rs=s.executeQuery("select * from users where username='"+user1+"';");
while(rs.next())
{
	name = rs.getString("name");
	surname = rs.getString("surname");
	mobile = rs.getString("mobile");
	username = rs.getString("username");
	password = rs.getString("password");
	email = rs.getString("email");
	address = rs.getString("address");
	tk = rs.getString("tk");
	city = rs.getString("city");
	prefecture = rs.getString("prefecture");
	country = rs.getString("country");
	info = rs.getString("info");
}
%>

<form method="post" action='update'>
Αλλαγή Στοιχείων

Προσωπικά στοιχεία
Όνομα<input type="text" name="name" value="<%= name %>">
Επώνυμο<input type="text" name="surname" value="<%= surname %>">
Κινητό<input type="text" name="mobile" value="<%= mobile %>">
Κωδικός χρήστη<input type="password" name="password" value="<%= password %>">
Επανάληψη κωδικού<input type="password" name="password2" value="<%= password %>">
e-mail<input type="text" name="email" value="<%= email %>">

Αποστολή παραγγελίας
Διεύθυνση<input type="text" name="address" value="<%= address %>">
T.K.<input type="text" name="tk" value="<%= tk %>">
Πόλη<input type="text" name="city" value="<%= city %>">
Νομός<input type="text" name="prefecture" value="<%= prefecture %>">
Χώρα<input type="text" name="country" value="<%= country %>">
```

<input type="checkbox" name="info" <% if(info.equals("on")){ %> value="on" checked <% } %>>Επιθυμώ να λαμβάνω πληροφορίες για προσφορές και νέα προιόντα μέσω e-mail.
<input class="button" type="submit" value=" Update ">
</form>

3.2. Περιγραφή των Servlets

Πολλά στοιχεία από τα παρακάτω servlet που θα αναλυθούν είναι κοινά. Για παράδειγμα η εισαγωγή συγκεκριμένων βιβλιοθηκών όπως :

```
import java.sql.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;  ,
```

η χρησιμοποίηση των μεθόδων doGet() και doPost()

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
   .
   .
   .
}

public void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
   .
   .
   .
}
```

Επίσης όταν από ένα servlet κριθεί αναγκαία η σύνδεση με τη βάση, η σύνδεση αυτή επιτυγχάνεται εντός της μεθόδου init() με τη κλήση της μεθόδου getConnection() του αντικειμένου MyConnection της βοηθητικής τάξης MyConnection .

```
public void init()
{
        MyConnection connectdb=new MyConnection();
         con=connectdb.getConnection();
}
```

Τέλος, σε όλα τα servlet καλείται η μέθοδος getSession() του αντικειμένου request το οποίο είναι τύπου HttpServletRequest. Επειδή το HttpSession κατά την πρώτη κλήση δεν υπάρχει, δημιουργείται.

```
HttpSession session = request.getSession();
```

Όπως αναφέραμε στην init() δημιουργείται η σύνδεση με τη βάση. Αντίστοιχα στη destroy() η σύνδεση κλείνει.

```
public void destroy()
{
    if (con!=null)
    {
        try
        {
                con.close();
        }
        catch (SQLException e)
        {
                log("Couldn't close connection with MySQL", e);
        }
    }
}
```

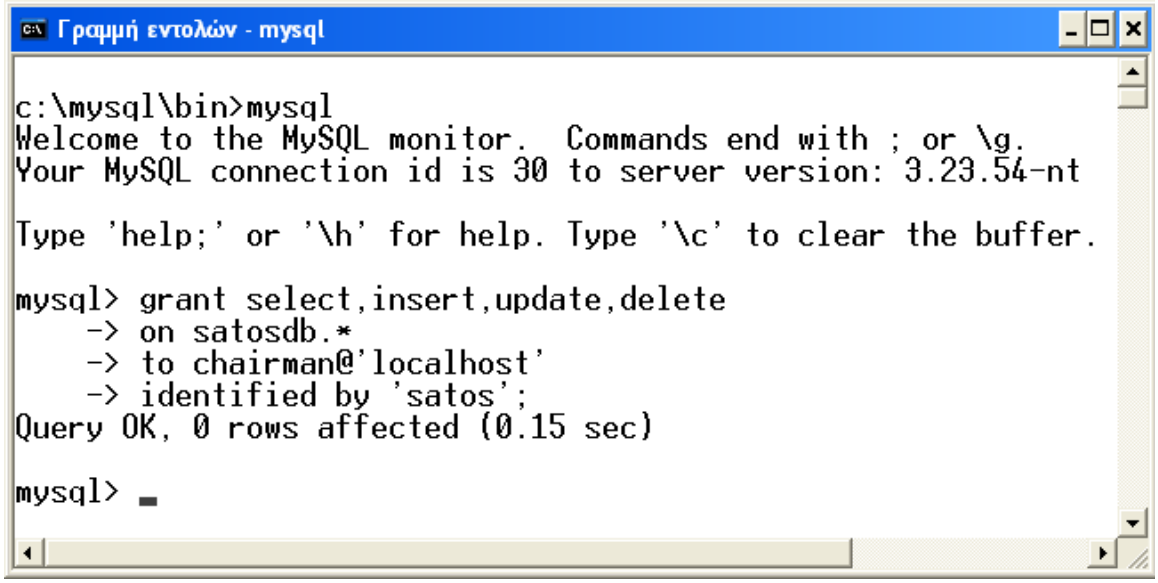
User

Η τάξη User είναι βοηθητική και αποτελείται από ένα κατασκευαστή και δύο μεθόδους. Ο κατασκευαστής παίρνει δύο ορίσματα τύπου String. Με τις μεθόδους getUsername() και getPassword() μας επιστρέφεται η τιμή του ονόματος του χρήστη και του κωδικού του αντίστοιχα.

```
package user;

public class User
{
	private String username;
	private String password;

	public User(String username, String password)
	{
		this.username=username;
		this.password=password;
	}

	public String getUsername()
	{
		return username;
	}
	public String getPassword()
	{
		return password;
	}
}
```

MyConnection

Στο servlet MyConnection δημιουργούμε την σύνδεση με την βάση δεδομένων μας έτσι ώστε σε περίπτωση που γίνει κάποια αλλαγή (αλλαγή του ονόματος της βάσης, αλλαγή των στοιχείων του χρήστη…) να χρειαστεί να παρέμβουμε μόνο σε αυτό και όχι σε όλα.
Με την εντολή *Class.forName("org.gjt.mm.mysql.Driver");* καλούμε την τάξη Driver του connector *mysql-connector-java-3.0.9-stable-bin.jar,* το οποίο είναι ένα jar αρχείο και είναι υπεύθυνο για την επικοινωνία των servlet και των JSP σελίδων με την βάση δεδομένων MySQL. Δημιουργούμε ένα αντικείμενο τύπου Connection και με την εντολή *DriverManager.getConnection("jdbc:mysql://localhost/satosdb?user=chairman&password=satos&"+"useUnicode=true&characterEncoding=iso-8859-7");* επιτυγχάνεται η επικοινωνία. Όπως παρατηρούμε καλούμε την βάση μας με παραμέτρους user=chairman και password=satos. Να επισημάνουμε ότι στον χρήστη έχουμε δώσει δικαιώματα insert, delete, update, select από το command prompt.



Στην εντολή το *chairman@'localhost'* χρησιμοποιούμε το localhost διότι η εφαρμογή μας τρέχει τοπικά. Διαφορετικά στη θέση του θα χρησιμοποιούσαμε το σύμβολο '%'.

Topic

Η βοηθητική τάξη Topic εμφανίζει στη σελίδα του forum τις απαντήσεις των θεμάτων. Αυτό επιτυγχάνεται ως εξής, εντός της σελίδας forum.jsp και με χρήση SQL εντολής καλούνται από τον πίνακα topics της βάσης τα θέματα. Με μία συνθήκη while καλείται κάθε φορά ο κατασκευαστής της τάξης Topic ο οποίος θα παίρνει σαν ορίσματα το αντικείμενο con τύπου Connection και ένα String που κάθε φορά θα κρατά τον κωδικό του κάθε θέματος. Εδώ να τονίσουμε ότι ο κωδικός του κάθε θέματος είναι πρωτεύον κλειδί και άρα είναι μοναδικός. Στην τάξη Topic και εντός της μεθόδου toString() χρησιμοποιείται ο κωδικός του θέματος ως συνθήκη σε SQL εντολή για να μας επιστραφούν οι απαντήσεις του συγκεκριμένου θέματος. Μιας και η μέθοδος αυτή είναι τύπου String θα επιστραφούν σε μορφή String. Το String αυτό θα εμφανιστεί στη forum.jsp και στη συνέχεια η συνθήκη while θα πάει στο επόμενο θέμα.

```
public Topic(Connection con,String id)
{
        this.con=con;
        this.id=id;
}

public String toString()
{
   :
   :
```

```
        return out.toString();
}
```

Από τη σελίδα forum.jsp

```
<%
rs =s.executeQuery("select * from topics order by tdate;");
while (rs.next())
{
Object tdate = rs.getObject("tdate");
Object topic = rs.getObject("topic");
Object tid = rs.getObject("tid");
Object users = rs.getObject("user");
String t=tid.toString();
Topic topics=new Topic(con,t);
String result=topics.toString();
%>
<%= users %> --> <a href="answer?tid=<%= tid %>"><%= topic %></a>
<%= tdate %>

<%= result %>

<%
}
%>
```

answer (answer_en)

Από τη σελίδα forum.jsp ο επισκέπτης είτε είναι συνδεδεμένος είτε όχι, μπορεί να απαντήσει σε κάποιο από τα θέματα του forum πατώντας σε ένα από αυτά. Τότε καλείται το servlet answer το οποίο θα πάρει τον κωδικό του θέματος από την μέθοδο doGet(), ο οποίος έχει αποσταλεί μέσω του url, και θα τον αποθηκεύσει στο session. Έπειτα θα εμφανίσει στον χρήστη την entry.jsp.

```
String tid=request.getParameter("tid");
session.setAttribute("tid",tid);
response.sendRedirect("entry.jsp");
```

send (send_en)

Το servlet send βρίσκει εφαρμογή στο forum. Όταν κάποιος επισκέπτης του forum, όχι απαραίτητα συνδεδεμένος χρήστης, θελήσει να απαντήσει σε κάποιο από τα θέματα, θα το επιλέξει ώστε να εμφανιστεί η σελίδα entry.jsp για να εισάγει την απάντησή του. Το send είναι υπεύθυνο να καταχωρήσει το όνομα και την απάντηση στον πίνακα answers της Β.Δ. Να αναφέρουμε ότι ο κωδικός του θέματος στο οποίο απαντά συγκρατείται στο session. Επίσης δεν μπορεί να καταχωρήσει κενή απάντηση ή χωρίς πρώτα την επιλογή θέματος.

```
String user=request.getParameter("user");
String answer=request.getParameter("answer");
String tid = (String) session.getAttribute("tid");

if(tid==null) {
        session.setAttribute("usermsg","Δεν έχει επιλεγεί θέμα!");
        RequestDispatcher rd =getServletConfig().getServletContext()
        .getRequestDispatcher("/entry.jsp");rd.forward(request, response);
        return;
}
if(answer.equals("")) {
        session.setAttribute("usermsg","Εισάγεται απάντηση!");
        RequestDispatcher rd =getServletConfig().getServletContext(
        ).getRequestDispatcher("/entry.jsp");rd.forward(request, response);
        return;
}
.
.
.
int insert1=stmt.executeUpdate("INSERT INTO answers VALUES(
                ",'"+year+"-"+month+"-"+date+" "+hours+":"+minutes+":"+
                seconds+"','"+answer+"','"+user+"',"+tid+");");

session.removeAttribute("tid");
```

register (register_en)

Το register servlet έχει ως στόχο την αποθήκευση των στοιχείων που εισάγει ο χρήστης, στην βάση δεδομένων. Η εισαγωγή των στοιχείων γίνεται στην φόρμα της σελίδας register.jsp. (Λεπτομέρειες σελ. 22)

Επειδή τα στοιχεία στέλνονται μέσω φόρμας χρησιμοποιούμε την μέθοδο doPost. Με την μέθοδο request.getParameter(“όνομα πεδίου”) παίρνουμε τις τιμές των πεδίων της φόρμας τις οποίες αποθηκεύουμε σε μεταβλητές τύπου String μιας και η συγκεκριμένη μέθοδο επιστρέφει τιμές τύπου String. Για την αποφυγή τυχόν προβλημάτων ελέγχουμε τις τιμές με διαδοχικές ‘if’. Αν το username που εισήγαγε ο χρήστης έχει την τιμή admin,

εκχωρείται στο session το μήνυμα "Παρακαλώ αλλάξτε το username!" και ανακατευθυνόμαστε στη σελίδα register.jsp με τη χρήση της εντολής *RequestDispatcher rd =getServletConfig(). getServletContext(). getRequestDispatcher("/register.jsp"); rd.forward(req, res); return;* όπου και εμφανίζεται το παραπάνω μήνυμα. Αυτό συμβαίνει διότι το username με τιμή admin είναι δεσμευμένο από τον διαχειριστή της εφαρμογής. Την εντολή return την χρησιμοποιούμε για να σταματάει και να μην εκτελείται ο κώδικας που ακολουθεί. Στη συνέχεια γίνεται έλεγχος για κενά πεδία. Στην περίπτωση που κάποιο ή κάποια πεδία είναι κενά, ανακατευθυνόμαστε στη σελίδα register.jsp και με τη χρήση του session εμφανίζεται σχετικό μήνυμα. Για την αποφυγή σφάλματος στον κωδικό που εισάγει ο χρήστης, υπάρχει και δεύτερο πεδίο για την επιβεβαίωση του. Τον έλεγχο αναλαμβάνει το servlet και σε περίπτωση που τα δύο πεδία έχουν διαφορετικές τιμές εμφανίζεται μήνυμα ότι τα δύο πεδία δεν έχουν την ίδια τιμή. Αφού δεν έχουν παρουσιαστεί σφάλματα ξεκινάει η διαδικασία αποθήκευσης των δεδομένων στη βάση μας. Στη συνέχεια δημιουργούμε το αντικείμενο Statement μέσω του οποίου μπορούμε να εκτελέσουμε εντολές στη βάση δεδομένων. Αρχικά καλούμε την μέθοδο executeQuery() ώστε να εκτελέσουμε εντολή select. Η εντολή επιστρέφει ένα αντικείμενο τύπου ResultSet και γι' αυτό το δημιουργούμε. Η μέθοδος παίρνει σαν όρισμα ένα String το οποίο είναι μια εντολή SQL. Στο servlet μας καλούμε την *"select * from users where username='"+username+"';"* η οποία θα επιστρέψει στο ResultSet τις τιμές των πεδίων του πίνακα users της Β.Δ. μας. Με την συνθήκη της εντολής θα επιστραφούν τα δεδομένα όπου η τιμή του πεδίου username είναι ίση με την τιμή που έδωσε ο χρήστης στην φόρμα. Επειδή το πεδίο username είναι primary key δεν θα υπάρχουν περισσότερα αποτελέσματα από ένα. Αν υπάρχει η τιμή στη βάση τότε ο χρήστης θα πρέπει να επιλέξει διαφορετικό username, κάτι που του κοινοποιείται με τον τρόπο που αναφέρθηκε και παραπάνω. Σε διαφορετική περίπτωση καλείται η μέθοδος του Statement executeUpdate με την χρήση της οποίας μπορούμε να εισάγουμε τα δεδομένα του χρήστη στη Β.Δ. Να τονίσουμε ότι η συγκεκριμένη μέθοδος επιστρέφει integer και όχι ResultSet. Αν όλα πάνε καλά και δεν δημιουργηθεί κάποια εξαίρεση, για παράδειγμα μη επικοινωνία με την Β.Δ., η εισαγωγή πραγματοποιείται και ο χρήστης κατευθύνεται στην login.jsp ώστε να συνδεθεί.

login (login_en)

Το login καλείτε μέσω της φόρμας της σελίδας login.jsp, και η χρήση του είναι να μας κατευθύνει είτε στη σελίδα του χρήστη user.jsp είτε στη σελίδα του διαχειριστή admin.jsp. Αυτό επιτυγχάνεται με την ταυτοποίηση των τιμών που έχει εισάγει ο χρήστης στα πεδία 'username' και 'password' της login.jsp με τα δεδομένα που είναι αποθηκευμένα στη βάση μας. Εδώ να αναφέρουμε ότι στη βάση είναι αποθηκευμένα μόνο τα στοιχεία των χρηστών. Οι τιμές του διαχειριστή είναι αποθηκευμένες στο αρχείο web.xml (Λεπτομέρειες σελ. 16) και καλούνται από το servlet μέσω της μεθόδου init(). Η μέθοδος που χρησιμοποιείται είναι η getInitParameter() του αντικειμένου ServletConfig. Στη συνέχεια γίνεται έλεγχος με τις τιμές που έχουν δοθεί και αν είναι ίδιες με τις τιμές του διαχειριστή, τότε εμφανίζεται η σελίδα του διαχειριστή (admin.jsp). Αν δεν είναι, γίνεται έλεγχος με τις τιμές που είναι αποθηκευμένες στη βάση. Αυτό επιτυγχάνεται

εντός της μεθόδου createUser (String, String) που είναι τύπου User. Αν υπάρχει στη βάση χρήστης με αυτές τις τιμές τότε με τη χρήση του constructor της βοηθητικής τάξης User (Λεπτομέρειες σελ. 42) θα δημιουργηθεί ένα αντικείμενο τύπου User όπου θα πάρει ως ορίσματα το όνομα και τον κωδικό του χρήστη. Χρησιμοποιώντας τη μέθοδο getUsername() του αντικειμένου User αποθηκεύουμε το όνομα του χρήστη στο session με όνομα ‘ User ’. Στη συνέχεια θα εμφανιστεί η σελίδα του χρήστη.

```
import user.User;
   :
   :
public void init(ServletConfig sc) throws ServletException
{
        MyConnection connectdb=new MyConnection();
         con=connectdb.getConnection();
        super.init(sc);
        aname = sc.getInitParameter("username");
        apasswd = sc.getInitParameter("password");
}
   :
   :
String name = request.getParameter("username");
String passwd = request.getParameter("password");
if(name.equals(aname) && passwd.equals(apasswd)) {
        session.setAttribute("admin",name);
        RequestDispatcher rd = getServletConfig().
                                getServletContext().getRequestDispatcher("/admin.jsp");
                                rd.forward(request, response);
                return;
}

User User;
   :
   :
        User = createUser(name, passwd);
   :
   :
protected User createUser(String user, String passwd)
throws SQLException, ClassNotFoundException {
        User User;
        Statement s = con.createStatement();
        ResultSet rs =s.executeQuery("select * from users where username='"+user+"'
        and password='"+passwd+"' ;");
        if (rs.next()) {
                User = new User(user, passwd);
                return User;
        }
        return null;
}
```

MailerBean

Το MailerBean είναι το servlet εκείνο που είναι αρμόδιο να αποστέλλει δεδομένα σε ηλεκτρονικό ταχυδρομείο. Στην εφαρμογή μας το χρησιμοποιούμε σε τρεις διαφορετικές περιπτώσεις. Στην πρώτη ο διαχειριστής έχει την δυνατότητα από την σελίδα του να στείλει e-mail σε κάποιον από τους χρήστες. Στην δεύτερη περίπτωση το χρησιμοποιούμε για να αποστείλουμε τον κωδικό στον χρήστη που τον έχει ξεχάσει. Τέλος, όταν ο χρήστης θα στείλει μια παραγγελία, με το MailerBean θα αποσταλεί στον διαχειριστή e-mail που θα τον ενημερώνει. Σαν παράδειγμα ας αναλύσουμε την πρώτη περίπτωση. Στη σελίδα του διαχειριστή υπάρχουν τέσσερα πεδία τα οποία συγκρατούν, την διεύθυνση του αποστολέα, του παραλήπτη, το θέμα και το μήνυμα.

(Από admin.jsp)

```
<input type="hidden" name="from" value="satos@satos.gr">
        To : <input type="text" name="to" >
        Subject : <input type="text" name="subject">
        Message : <textarea rows="10" cols="80" name="message"></textarea>
```

Στη συνέχεια κατευθύνεται στη mailer.jsp εντός της οποίας καλείται η συνάρτηση του MailerBean sendMail(). Αν προκύψει εξαίρεση, θα εμφανιστεί η σελίδα errorPage.jsp. Ας δούμε αναλυτικά το servlet.

```
package com.stardeveloper.bean.test;
import java.io.*;
import java.util.*;
import javax.mail.*;
import javax.mail.event.*;
import javax.mail.internet.*;
```

Όπως βλέπουμε εισάγουμε για πρώτη φορά σε servlet τις τάξεις των πακέτων javax.mail.event και javax.mail.internet που είναι απαραίτητες για να επιτευχθεί η αποστολή του e-mail. Έπειτα δηλώνονται μεταβλητές τύπου String που συγκρατούν τα δεδομένα που αποστέλλονται από τη σελίδα του διαχειριστή. Για την αποθήκευση των δεδομένων χρησιμοποιούνται οι αντίστοιχες μέθοδοι.

```
public final class MailerBean extends Object implements Serializable
{

/* Bean Properties */
private String to = null;
private String from = null;
private String subject = null;
private String message = null;
public static Properties props = null;
public static Session session = null;
```

```
static {
/* Setting Properties for STMP host */
props = System.getProperties();
props.put("mail.smtp.host", "mail.satos.gr");
props.put("mail.smtp.auth", "true");
session = Session.getDefaultInstance(props, null);
}
/* Setter Methods */
public void setTo(String to) {
this.to = to;
}

public void setFrom(String from)
{
this.from = from;
}

public void setSubject(String subject) {
this.subject = subject;
}

public void setMessage(String message) {
this.message = message;
}
```

Εδώ δημιουργείται η συνάρτηση sendMail() που όπως αναφέραμε και προηγουμένως καλείται εντός της mailer.jsp. Δουλεία της είναι η αποστολή του e-mail. Όπως μπορούμε να διακρίνουμε δηλώνουμε το SMTP πρωτόκολλο, το όνομα του διακομιστή και για το e-mail του αποστολέα το όνομα χρήστη και τον κωδικό.

```
/* Sends Email */
public void sendMail() throws Exception {
if(!this.everythingIsSet())
throw new Exception("Could not send email.");
try {
MimeMessage message = new MimeMessage(session);
message.setRecipient(Message.RecipientType.TO,
new InternetAddress(this.to));
message.setFrom(new InternetAddress(this.from));
message.setSubject(this.subject);
message.setText(this.message);
Transport transport = session.getTransport("smtp");
transport.connect("όνομα_διακομιστή", "τιμή_e-mail", "κωδικός_πρόσβασης");
message.saveChanges();
transport.sendMessage(message, message.getAllRecipients());
transport.close();
```

```
} catch (MessagingException e) {
throw new Exception(e.getMessage());
}
}
```

Επίσης εντός του servlet και μία μέθοδος τύπου boolean που ελέγχει αν οι ηλεκτρονικές διευθύνσεις δεν περιλαμβάνουν τα σύμβολα ‘ . ‘ και ‘ @ ‘ καθώς και την περίπτωση που υπάρχει κενό πεδίο.

```
/* Checks whether all properties have been set or not */
private boolean everythingIsSet()
{
if((this.to == null) || (this.from == null) ||
(this.subject == null) || (this.message == null))
return false;

if((this.to.indexOf("@") == -1) ||
(this.to.indexOf(".") == -1))
return false;

if((this.from.indexOf("@") == -1) ||
(this.from.indexOf(".") == -1))
return false;

return true;
}
}
```

UploadTest

Με το servlet UploadTest ο διαχειριστής μπορεί από τη σελίδα του να κάνει upload files. Στην εφαρμογή μας θα χρειαστεί να ‘ανεβάσει’ αρχεία τύπου txt και jpg. Τα περιεχόμενα του text αρχείου το οποίο θα πρέπει να έχει όνομα news.txt (news_en.txt) θα εμφανίζονται στην σελίδα των νέων news.jsp (news_en.jsp).
Οι φωτογραφίες θα πρέπει να αποθηκεύονται με όνομα της μορφής *κωδικός_προϊόντος.JPG*. Με αυτό τον τρόπο θα επιτύχουμε την εμφάνιση της φωτογραφίας, στις σελίδες products.jsp & basket.jsp, δίπλα από το προϊόν που έχει τον ίδιο κωδικό με αυτόν που υπάρχει στο όνομα της φωτογραφίας. Ουσιαστικά εισάγεται η τάξη MultipartRequest, που αναλύεται στη συνέχεια, που είναι υπεύθυνη για το upload. Το παρόν servlet θα μας εμφανίσει το όνομα του αρχείου που ‘ανεβάσαμε’, το τύπο του και το μέγεθός του. Στο τέλος της σελίδας θα υπάρχει link για να μπορεί να επιστρέψει στη σελίδα διαχείρισης ο διαχειριστής.

```
.
.
.
   out.println("name: " + name);
   out.println("filename: " + filename);
   out.println("type: " + type);
   if (f != null) {
     out.println("length: " + f.length());
.
.
.
out.println("<a href='admin.jsp'>Back to Administrator page</a>");
```

MultipartRequest

Η τάξη MultipartRequest εισάγεται στο servlet UploadTest. Με αυτή την τάξη ουσιαστικά επιτυγχάνεται το upload των αρχείων. Ας δούμε τα σημαντικότερα σημεία του servlet.

Ορίζουμε μία μεταβλητή που θα συγκρατεί το μέγιστο μέγεθος του αρχείου που μπορεί να γίνει upload. Εδώ το δηλώνουμε μέχρι 1MB.

```
private static final int DEFAULT_MAX_POST_SIZE = 1024 * 1024;  // 1 Meg
```

Δηλώνουμε το path του φακέλου που θα συγκρατεί τα αρχεία.

```
dir = new File("C:\\Program Files\\Apache Group\\Tomcat 4.1\\webapps\\satos\\files");
```

Κατασκευάζουμε ένα αντικείμενο τύπου File που η τιμή του θα είναι, η διαδρομή του φακέλου που ορίσαμε παραπάνω, η κάθετος και το όνομα του αρχείου.

```
File f = new File(dir + File.separator + filename);
```

Ο κατασκευαστής της τάξης UploadedFile δέχεται ως ορίσματα τη διαδρομή του φακέλου, το όνομα και το είδος του αρχείου. Αυτά θα αποθηκευτούν σε αντικείμενα τύπου String και θα χρησιμοποιηθούν από το servlet.

```
// A class to hold information about an uploaded file.
class UploadedFile {
  private String dir;
  private String filename;
  private String type;

  UploadedFile(String dir, String filename, String type) {
    this.dir = dir;
    this.filename = filename;
    this.type = type;   }
```

Insert

Με τη χρήση του servlet Insert ο διαχειριστής της εφαρμογής εισάγει νέα προϊόντα στη βάση. Οι παράμετροι που δίνει (δηλαδή κωδικό, περιγραφή, τιμή και είδος) θα αποθηκευτούν σε μεταβλητές τύπου String και θα ελεγχθεί η τιμή του κωδικού ώστε να μην είναι κενή. Στη συνέχεια αν δεν παρουσιαστεί εξαίρεση τα δεδομένα θα αποθηκευτούν στη βάση.

```
String id=request.getParameter("id");
String description=request.getParameter("desc");
double price=Double.parseDouble(request.getParameter("price"));
String ptype=request.getParameter("ptype");
   :
   :
Statement stmt = con.createStatement();
int keyword1=stmt.executeUpdate("INSERT INTO products VALUES(
'"+id+"','"+description+"',"+price+",'"+ptype+"');");
```

Upd

Η Upd είναι μία βοηθητική τάξη που χρησιμοποιείται για να εμφανίζονται στη σελίδα του διαχειριστή τα χαρακτηριστικά των προϊόντων, σε πεδία HTML τύπου text, ώστε να μπορεί ο διαχειριστής να τα αλλάξει και συνεπώς να τα ανανεώσει στη βάση. Ουσιαστικά τα πεδία αποτελούν μέρη μιας φόρμας η οποία καλεί το servlet updprod (Λεπτομέρειες σελ. 54) με χρήση του κουμπιού submit. Λόγω του γεγονότος ότι υπάρχει επιπλέον στήλη με πεδία τύπου checkbox, ο διαχειριστής επιλέγοντάς τα έχει την δυνατότητα να ανανεώσει περισσότερα από ένα προϊόντα.

```
out.append("<TABLE><form method='post' action='updprod'>\n");
   :
   :
int j=1;
while(rs.next())
{
        out.append("<TR><form method='post' action='updprod'>");
        Object id="";
        for(int i=1;i<=numcols;i++)
        {
                out.append("<TD>");
                Object obj=rs.getObject(i);
                if(obj!=null)
                {
                        if(i==1)
                        {
                                out.append(obj.toString()+
```

```
                    "<input name='id' type='hidden'
                     value='"+obj.toString()+"'>");
                    }
                if(i==1)
                {
                    id=obj;
                }
                if(i==2)
                {
                    out.append("<input name='desc"+id+"'
                    type='text' value='"+obj.toString()+"'>");
                }
                if(i==3)
                {
                    out.append("<input name='price"+id+"'
                    type='text' value='"+obj.toString()+"'>");
                }
                if(i==4)
                {
                    out.append("<input name='ptype"+id+"'
                    type='text' value='"+obj.toString()+"'>");
                }
        }
                else
                {
                    out.append(" ");
                }
        }
        out.append("<TD>"+"<input type='checkbox'
        name='updprod"+j+"'><input type='hidden'
        name='id"+j+"' value='"+id+"'>");
        j++;
        out.append("</TR>\n");
}
out.append("<TR><TD><input type='hidden'
                name='count' value='"+j+"'></TD></TR>");
out.append("<TR><TD colspan='5'><input class='button' type='submit'
                 value='  Update  '></TD></TR></form></TABLE>\n");
```

updprod

Όπως ήδη έχουμε αναφέρει μία από τις ενέργειες του διαχειριστή είναι και η ανανέωση των προϊόντων. Συγκεκριμένα τα χαρακτηριστικά του κάθε προϊόντος εμπεριέχονται σε πεδία τα οποία είναι μέρος φόρμας. Τα δεδομένα αποστέλλονται μέσω της φόρμας με τη μέθοδο POST και το servlet updprod αναλαμβάνει να τα αποθηκεύσει σε αντικείμενα τύπου String και στη συνέχεια να τα καταχωρήσει στη Β.Δ. Να αναφέρουμε όμως ότι μπορούν να ανανεωθούν η περιγραφή του προϊόντος, η τιμή του και το είδος του. Δεν μπορεί να ανανεωθεί ο κωδικός του προϊόντος. Επίσης δεν επιτρέπεται να αφήσει κενό κάποιο από τα πεδία. Η ανανέωση θα γίνει στα επιλεγμένα προϊόντα. Δηλαδή σε αυτά που ο διαχειριστής επέλεξε το αντίστοιχο checkbox. Το updprod θα ελέγξει ποια είναι επιλεγμένα και θα αποθηκεύσει τον κωδικό του προϊόντος που αντιστοιχούν. Στη συνέχεια θα αποθηκεύσει τα δεδομένα των υπόλοιπων πεδίων και θα τα καταχωρήσει στη βάση. Για παράδειγμα η κλήση της τιμής του πεδίου που συγκρατεί την περιγραφή γίνεται με το παρακάτω κομμάτι κώδικα :

```
String udesc="desc";
udesc=udesc+pid; // όπου pid ο κωδικός του προϊόντος
String desc=req.getParameter(udesc);
```

Όπως γίνεται αντιληπτό το όνομα του πεδίου αποτελείται από την λέξη :

desc + τον κωδικό του προϊόντος

Η ολοκληρωμένη μορφή του servlet είναι η εξής:

```
String count=req.getParameter("count"); // το count συγκρατεί το πλήθος των γραμμών
int counter=Integer.parseInt(count);

if (con!=null)
{
        try
        {
                Statement s = con.createStatement();
                int j=0; // η μεταβλητή j θα συγκρατεί τον αριθμό
                        // των ανανεωμένων προϊόντων
                for(int i=1;i<=counter;i++)
                {
                        String updprod="updprod";
                        String id="id";
                        updprod=updprod+i;
                        id=id+i;
                        String upd=req.getParameter(updprod);
                        String pid=req.getParameter(id);
                        if(upd!=null)
                        {
```

```
                        String udesc="desc";
                        String uprice="price";
                        String uptype="ptype";
                        udesc=udesc+pid;
                        uprice=uprice+pid;
                        uptype=uptype+pid;
                        String desc=req.getParameter(udesc);
                        String price=req.getParameter(uprice);
                        String ptype=req.getParameter(uptype);

                        if(desc.equals("") || price.equals("") || ptype.equals(""))
                        {
                                session.setAttribute("adminmsg",
                                "Παρακαλώ συμπληρώστε όλα τα πεδία!");
                                RequestDispatcher rd =getServletConfig().
                                getServletContext().getRequestDispatcher(
                                "/admin.jsp");rd.forward(req, res);
                                return;
                        }

                        int update =s.executeUpdate("update products
                        set DESCRIPTION='"+desc+"',PRICE="+price+
                        ",PTYPE='"+ptype+"' where ID='"+pid+"';");
                        j++;
                }
        }
        session.setAttribute("adminmsg","Ανανεώθηκαν "+j+" προϊόντα");
        res.sendRedirect("/satos/admin.jsp");
    .
    .
    .
}
```

Del

Η τάξη Del είναι βοηθητική και τη χρησιμοποιούμε για να εμφανίσουμε τα προϊόντα στη σελίδα του διαχειριστή με δυνατότητα διαγραφής των προϊόντων, η οποία επιτυγχάνεται μέσω του servlet Delete (Λεπτομέρειες σελ. 57).

Αρχικά το πακετάρουμε.
package del;

Εντός της τάξης δημιουργούμε δύο μεταβλητές τύπου String και Connection οι οποίες θα περαστούν ως παράμετροι στον κατασκευαστή της τάξης.

private String query;

```
private Connection con;
public Del(Connection con,String query)
{
        this.query=query;
        this.con=con;
}
```

Έπειτα, δημιουργούμε την μέθοδο toString() που είναι τύπου String. Εντός της μεθόδου θα δημιουργήσουμε αρχικά ένα αντικείμενο τύπου StringBuffer. Ο λόγος είναι ότι με τη μέθοδο append() του αντικειμένου αυτού μπορούν νέα δεδομένα να προστίθενται στο τέλος της ήδη υπάρχουσας τιμής. Στη περίπτωση που θα χρησιμοποιούσαμε αντικείμενο τύπου String θα δημιουργούνταν κάθε φορά νέο αντικείμενο κάτι που θα είχε επίπτωση στην απόδοση. Με τη χρήση αυτής της εντολής θα καταφέρουμε να συνδυάσουμε εντολές Java και HTML. Έτσι θα επιτύχουμε τα δεδομένα που θα επιστραφούν από τη βάση να εμφανιστούν μέσα σε πίνακα.

```
StringBuffer out=new StringBuffer();
```

Αφού δημιουργήσουμε τα απαραίτητα αντικείμενα καλούμε τις μεθόδους getColumnCount() και getColumnLabel() του αντικειμένου ResultSetMetaData για να μας επιστραφεί ο αριθμός και τα ονόματα των στηλών του πίνακα products της Β.Δ.

```
int numcols=rsmd.getColumnCount();
for(int i=1;i<=numcols;i++)
{
        rsmd.getColumnLabel(i)
}
```

Θα δημιουργήσουμε μία επιπλέον στήλη με όνομα Delete στην οποία θα υπάρχει checkbox και hidden πεδία που θα συγκρατούν το id του προϊόντος. Τα πεδία αυτά θα αποστέλλονται στο servlet Delete μέσω της φόρμας και με την μέθοδο POST. Έτσι θα μπορεί ο διαχειριστής να διαγράψει ένα ή περισσότερα προϊόντα συγχρόνως από τη βάση.

```
out.append("<TD>"+"<input type='checkbox' name='delprod"+
                j+"'><input type='hidden' name='id"+j+"' value='"+id+"'>");
```

Για την εμφάνιση των δεδομένων αποστέλλεται από τη σελίδα του διαχειριστή η εντολή "select * from products order by ptype,id;"

Όσο θα υπάρχουν δεδομένα θα τα καλούμε με τη μέθοδο getObject() και θα τα αποθηκεύουμε σε αντικείμενα τύπου Object τα οποία και θα εμφανίζονται.

```
while(rs.next())
{
        Object obj=rs.getObject();
}
```

Delete

Όπως αναφέραμε προηγουμένως με την κλήση του servlet Delete αποστέλλονται συγχρόνως και το id του προϊόντος που θέλουμε να διαγράψουμε και οι τιμές των checkbox. Για τις τιμές των checkbox που είναι ίσες με 'on' (δηλαδή διάφορες από 'null') θα διαγραφούν τα αντίστοιχα προϊόντα.

```
String count=request.getParameter("count");
int counter=Integer.parseInt(count);

if (con!=null)
{
        try
        {
                Statement s = con.createStatement();
                int j=0;
                for(int i=1;i<=counter;i++)
                {
                        String delprod="delprod";
                        String pid="id";
                        delprod=delprod+i;
                        pid=pid+i;
                        String prod=request.getParameter(delprod);
                        String id=request.getParameter(pid);
                        if(prod!=null)
                        {
                                int deleteCount =s.executeUpdate("delete
                                from products where id='"+id+"';");
                                j++;
                        }
                }
                session.setAttribute("adminmsg","Διαγράφηκαν "+j+" προϊόντα");
                response.sendRedirect("/satos/admin.jsp");
        }
  .
  .
  .
}
```

Users

Η χρήση του servlet Users είναι να εμφανίζει στη σελίδα του διαχειριστή τα στοιχεία του πίνακα users της βάσης καθώς και μία επιπλέον στήλη που θα περιέχει πεδία checkbox για τη διαγραφή των χρηστών. Εντός του πίνακα που θα δημιουργηθεί θα υπάρχει φόρμα η οποία θα καλεί το servlet DeleteUsr (Λεπτομέρειες σελ. 59). Το όνομα κάθε χρήστη θα αποθηκεύεται σε hidden πεδίο όπως και ο αριθμός των γραμμών και θα αποστέλλονται.

Συγκεκριμένα εντός του servlet υπάρχει ο κατασκευαστής της τάξης ο οποίος παίρνει σαν ορίσματα ένα αντικείμενο τύπου Connection και ένα τύπου String. Επίσης υπάρχει και η μέθοδος toString() που επιστρέφει String το οποίο είναι ένας συνδυασμός των στοιχείων των χρηστών από τη βάση και εντολών HTML. Ως αποτέλεσμα αυτού είναι η προβολή των δεδομένων σε HTML πίνακα.

```
public class Users
{
        private String query;
        private Connection con;

        public Users(Connection con,String query)
        {
                this.query=query;
                this.con=con;
        }

        public String toString()
        {
                StringBuffer out=new StringBuffer();
                   .
                   .
                out.append("<TABLE width='500' BORDER=1>
                                <form method='post' action='DeleteUsr'>\n");
                                   .
                                   .
                                int j=1;
                                while(rs.next())
                                {
                                        out.append("<TR>");
                                        Object user="";
                                        for(int i=1;i<=numcols;i++)
                                        {
                                                        .
                                                        .
                                        }
                                        out.append("<TD>"+"<input type='checkbox'
                                        name='delusr"+j+"'><input type='hidden'
                                         name='user"+j+"' value='"+user+"'>");
                                        j++;
                                out.append("</TR>\n");
                                }
                                out.append("<TR><TD><input type='hidden'
                                         name='count' value='"+j+"'></TD></TR>");

                                out.append("<TR><TD colspan='13'>
                                <input class='button' type='submit'
                                value='  Delete  '></TD></TR></form></TABLE>\n");
```

```
    .
    .
    .
        return out.toString();
        }
}
```

DeleteUsr

Εδώ ο διαχειριστής μπορεί να διαγράψει κάποιον χρήστη με χρήση του ονόματός του, το οποίο είναι και μοναδικό. Με την χρήση των checkbox έχει την δυνατότητα να διαγράψει και περισσότερους από έναν χρήστη.

```
String count=request.getParameter("count");
int counter=Integer.parseInt(count);

if (con!=null)
{
        try
        {
                Statement s = con.createStatement();
                int j=0;
                for(int i=1;i<=counter;i++)
                {
                        String del="delusr";
                        String usr="user";
                        del=del+i;
                        usr=usr+i;
                        String delusr=request.getParameter(del);
                        String user=request.getParameter(usr);
                        if(delusr!=null)
                        {
                                int deleteCount =s.executeUpdate("delete from users
                                where username='"+user+"';");
                                j++;
                        }
                }
                session.setAttribute("adminmsg","Διαγράφηκαν "+j+" χρήστες");
                response.sendRedirect("/satos/admin.jsp");
         }
    .
    .
    .
}
```

Ord

Η βοηθητική τάξη Ord εμφανίζει τον πίνακα των παραγγελιών από τη βάση στη σελίδα του διαχειριστή. Εμφανίζονται επιπλέον δύο στήλες όπου η πρώτη περιέχει ένα πεδίο HTML τύπου select και το κουμπί submit. Η φόρμα αυτή καλεί το servlet status που όπως έχουμε αναφέρει, ανανεώνει την κατάσταση της εξέλιξης μιας παραγγελίας. Η δεύτερη στήλη περιέχει ένα link το οποίο καλεί το servlet DeleteOrd (Λεπτομέρειες σελ. 61) , δουλειά του οποίου είναι να διαγράψει την παραγγελία από την βάση.

```
out.append("<TD>"+"<form method='post' action='status'>
<select name='status' class='form2'>
<option value=''>
<option value='Καταχωρήθηκε'>Καταχωρήθηκε
<option value='Συσκευάστηκε'>Συσκευάστηκε
<option value='Απεστάλη'>Απεστάλη
</select>
<input type='hidden' name='stid' value='"+id+"'>
<input type='hidden' name='datetime' value='"+datetime+"'>
<input type='submit'  value='update'></form></center>");
out.append("<TD>"+"<a href='DeleteOrd?id="+id+"'>delete</a>");
```

status

Πριν αναλύσουμε το servlet status να υπενθυμίσουμε ότι ο χρήστης έχει την δυνατότητα από την σελίδα του να ενημερώνεται για την εξέλιξη των παραγγελιών του. Η εξέλιξη αυτή ενημερώνεται από τον διαχειριστή. Ο διαχειριστής από την σελίδα του, με χρήση της βοηθητικής τάξης Ord, έχει την δυνατότητα να βλέπει όλες τις παραγγελίες που έχουν γίνει. Για κάθε παραγγελία υπάρχει ένα πεδίο που συμπεριλαμβάνει μία φόρμα, δουλειά της οποίας είναι να ανανεώνεται η κατάστασή της. Ο διαχειριστής επιλέγει την καινούργια κατάσταση από ένα πεδίο τύπου select και μαζί με τον κωδικό της παραγγελίας και με την τρέχουσα ημερομηνία και ώρα που είναι αποθηκευμένα σε hidden πεδία θα αποσταλούν στο servlet status μέσω της φόρμας αυτής.
Το servlet status θα αναλάβει την ανανέωση της κατάστασης από την Β.Δ. και θα αποθηκεύσει την ημερομηνία και ώρα που έγινε η αλλαγή από τον διαχειριστή για την παραγγελία με κωδικό, τον κωδικό που απεστάλθηκε από την φόρμα

```
String status = req.getParameter("status");
String stid = req.getParameter("stid");
String datetime = req.getParameter("datetime");

if (con!=null)
{
        Statement s = con.createStatement();
        int update =s.executeUpdate("update orders set
```

```
        status='"+status+"',sdate='"+datetime+"' where id='"+stid+"';");
}
```

DeleteOrd

Με το servlet DeleteOrd ο διαχειριστής μπορεί να διαγράψει μία ή περισσότερες παραγγελίες.

int deleteCount =s.executeUpdate("delete from orders where id='"+id+"';");

όπου id ο κωδικός της παραγγελίας που αποστέλεται από την σελίδα του διαχειριστή.

Topic2

Το servlet Topic2 είναι πανομοιότυπο με το Topic. Οι διαφορές είναι ότι σε αυτό εμφανίζονται τα θέματα και οι απαντήσεις του forum στη σελίδα του διαχειριστή. Η λογική είναι η ίδια με μόνη τροποποίηση την προσθήκη των link 'delete' στη θέση της ημερομηνίας, ώστε ο διαχειριστής να μπορεί να διαγράψει όποιο θεωρήσει απαραίτητο για διαγραφή.

delete

DeleteTop

Εδώ ο διαχειριστής μπορεί να διαγράψει θέματα του forum. Το σημείο που πρέπει να προσέξουμε εδώ είναι ότι εκτός από το θέμα θα διαγραφούν και οι απαντήσεις του, κάτι το οποίο είναι λογικό μιας και δεν μπορούν να υφίστανται απαντήσεις χωρίς θέμα.

<a href="DeleteTop?tid=<%= tid %>">delete // από admin.jsp

String tid=request.getParameter("tid");

int deleteCount =s.executeUpdate("delete from topics where tid='"+tid+"';");
int deleteCount2 =s.executeUpdate("delete from answers where id='"+tid+"';");

Όπου tid ο κωδικός του θέματος που συγκρατείται στη σελίδα του διαχειριστή (admin.jsp).

DeleteAns

Το servlet DeleteAns χρησιμοποιείται για τη διαγραφή των απαντήσεων των θεμάτων του forum και έχει πολλά κοινά στοιχεία με το servlet Delete. Και εδώ αποστέλλεται ο κωδικός της απάντησης με τη μέθοδο GET.

int deleteCount =s.executeUpdate("delete from answers where aid='"+aid+"';");

όπου aid ο κωδικός της απάντησης που θα διαγραφεί.

logoutadmin (logoutadmin_en)

Το servlet logoutadmin αποσυνδέει το διαχειριστή. Στην ουσία αφαιρεί από το session την τιμή της μεταβλητής ‘ admin ’. Έτσι στην περίπτωση που κληθεί ξανά η σελίδα του διαχειριστή δεν θα εμφανιστεί αλλά θα κατευθυνθεί στην login.jsp μιας και η τιμή της μεταβλητής admin του session θα είναι null.

session.removeAttribute("admin");

Prod

Η λειτουργία της βοηθητικής τάξης Prod είναι σχεδόν ίδια με αυτή της Bask. Δηλαδή θα εμφανίσουν τα προϊόντα, η μεν πρώτη στη σελίδα products.jsp ενώ η άλλη στην basket.jsp. Η διαφορά τους είναι ότι η Prod δεν θα εμφανίσει τη στήλη με τα HTML πεδία τύπου text ούτε και το κουμπί ‘προσθήκη’. Επομένως δεν δίνεται η δυνατότητα καταχώρησης προϊόντων στο καλάθι αγορών. Κάτι το οποίο είναι θεμιτό μιας και η σελίδα products.jsp εμφανίζεται και σε μη συνδεδεμένους χρήστες. Ο κώδικας αναλύεται στην Bask.

Bask

Αρχικά στη τάξη Bask με την εντολή package bask; πακετάρουμε την τάξη, δηλαδή θα δημιουργηθεί ένας φάκελος με όνομα bask ο οποίος θα συγκρατεί το αρχείο Bask.class. Η συγκεκριμένη τάξη είναι βοηθητική και χρησιμοποιείται στην basket.jsp. Η εισαγωγή της γίνεται με την εντολή

<%@ page import="bask.Bask" %>.

Ο constructor της τάξης παίρνει σαν ορίσματα ένα αντικείμενο τύπου Connection και ένα τύπου String. Εντός της τάξης δημιουργείται μέθοδος τύπου String με όνομα toString().

Με τη μέθοδο αυτή θα επιτύχουμε την προβολή των προϊόντων, των φωτογραφιών τους και των πεδίων κειμένου στη σελίδα basket.jsp δίνοντας στον χρήστη την δυνατότητα να δει και να παραγγείλει προϊόντα.

Δημιουργούμε αντικείμενα τύπου Connection, Statement με τη μέθοδο createStatement(), ResultSet με τη μέθοδο getResultSet(), ResultSetMetaData με τη μέθοδο getMetaData().

```
Connection con;
Statement stmt=con.createStatement();
ResultSet rs=stmt.getResultSet();
ResultSetMetaData rsmd=rs.getMetaData();
```

Η μέθοδος getColumnCount() του αντικειμένου ResultSetMetaData θα επιστρέψει έναν ακέραιο που θα αντιστοιχεί στον αριθμό των στηλών. Γνωρίζοντας τον αριθμό των στηλών, με την χρήση μίας for εμφανίζουμε τα ονόματα των πεδίων από τη βάση μέσω της μεθόδου getColumnLabel() του αντικειμένου ResultSetMetaData.

```
int numcols=rsmd.getColumnCount();
for(int i=1;i<=numcols;i++)
{
        rsmd.getColumnLabel(i)
}
```

Εδώ να αναφέρουμε ότι μέσω της σελίδας basket.jsp καλείται ο κατασκευαστής της Bask. Όπως αναφέραμε πιο πάνω ο κατασκευαστής παίρνει σαν ορίσματα αντικείμενα τύπου Connection και String. Το δεύτερο όρισμα , δηλαδή το String, είναι μια εντολή sql η οποία καλείται μέσα στο servlet Bask για την επιστροφή αποτελεσμάτων από τη βάση. Η εντολή αυτή που αποστέλλεται από την basket.jsp είναι η "select * from products order by ptype,id;" .

Όσο θα υπάρχουν δεδομένα θα τα καλούμε με τη μέθοδο getObject() και θα τα αποθηκεύουμε σε αντικείμενα τύπου Object.

```
while(rs.next())
{
        Object obj=rs.getObject();
}
```

Επίσης θα εμφανιστούν δύο ακόμη στήλες. Στην πρώτη θα εμφανίζονται οι φωτογραφίες των προϊόντων, οι οποίες είναι αποθηκευμένες στο φάκελο files με όνομα τον κωδικό του προϊόντος και κατάληξη JPG. Στη δεύτερη θα εμφανίζονται πεδία τύπου text με σκοπό να εισάγει ο χρήστης την ποσότητα του προϊόντος που ενδιαφέρεται.

```
out.append("<TD><img src='files/"+obj2.toString()+".JPG' alt='no photo'>");
out.append("<TD><input type='text' name='item"+j3+"' size='5' maxlength='4'>");
```

Αν ένα προϊόν δεν έχει φωτογραφία, θα εμφανιστεί το μήνυμα 'no photo'. Στη συνέχεια εκτελείται η ίδια sql εντολή και για κάθε μία εγγραφή που θα επιστραφεί στο ResulSet θα αποθηκευθούν σε hidden πεδία ο κωδικός και η τιμή αντίστοιχα.

```
int k=1;
rs =stmt.executeQuery("select * from products order by ptype,id;");
while (rs.next())
{
        double pricedb = rs.getDouble("PRICE");
        Double pricedb2=new Double(pricedb);
        String pricedb3=pricedb2.toString();
        String id = rs.getString("ID");
        Integer k2=new Integer(k);
        String k3=k2.toString();
        out.append("<input type='hidden'  name='itempr"+k3+"'
                        value='"+pricedb3+"'>");
        out.append("<input type='hidden' name='id"+k3+"' value='"+id+"'>");
        k++;
}
```

Τα δεδομένα που θα συγκρατούν τα hidden πεδία θα επεξεργαστούν στο servlet basket.

basket (basket_en)

Επειδή το servlet basket μπορεί να καλεστεί πολλές φορές μιας και δίνεται η δυνατότητα στον χρήστη να συνεχίσει τις αγορές του, είναι προτιμότερο να χρησιμοποιήσουμε αντικείμενα PreparedStatement και όχι Statement. Τα PreparedStatements είναι χρήσιμα σε περιπτώσεις που θέλουμε να εκτελέσουμε τις ίδιες εντολές πολλές φορές και αυτό διότι η μεταγλώττιση γίνεται μία φορά και όχι κάθε φορά που εκτελούμε την ερώτηση.
Έτσι τα αποτελέσματα που θα επιστραφούν από την κλήση της ερώτησης "select * from products;" θα αποθηκευτούν μία φορά στο ResultSet. Μέσω της μεθόδου isAfterLast() θα ελέγξουμε την περίπτωση που το ResultSet δεν θα περιέχει δεδομένα. Τότε στη μεταβλητή που συγκρατεί τον αριθμό των γραμμών θα αποθηκευτεί η τιμή μηδέν. Αν θα κληθεί η μέθοδος last() σημαίνει ότι είναι η τελευταία γραμμή των αποτελεσμάτων που υπάρχουν στο ResultSet. Τότε με τη μέθοδο getRow() θα επιστραφεί ένας ακέραιος που θα αντιστοιχεί στον αριθμό των γραμμών και θα αποθηκευτεί στη μεταβλητή numcols.

```
PreparedStatement stmt = con.prepareStatement("select * from products;");
ResultSet rs=stmt.executeQuery();
if(rs.isAfterLast())
{
        numcols=0;
}
```

```
else
{
        rs.last();
        numcols=rs.getRow();
}
```

Μέσω του παρακάτω κώδικα θα ελέγξουμε την περίπτωση που ο χρήστης θα έχει εισάγει για ποσότητα προϊόντος την τιμή μηδέν. Σε αυτή την περίπτωση θα επιστρέψει στην basket.jsp και θα εμφανιστεί μήνυμα που θα τον ενημερώνει για το πρόβλημα.

```
for(int i=1;i<=numcols;i++)
{
        Integer i2=new Integer(i);
        String i3=i2.toString();
        String item2 = "item"+i3;
        String item = req.getParameter(item2);
        if(item.equals("0"))
        {
                session.setAttribute("errorMessage","Έχετε εισάγει '0' ποσότητα");
                RequestDispatcher rd =getServletConfig().getServletContext().
                getRequestDispatcher("/basket.jsp");rd.forward(req, res);
        return;
        }
}
```

Να θυμίσουμε ότι η τάξη Bask δημιουργούσε δύο hidden πεδία που κρατούσαν τον κωδικό και την τιμή των προϊόντων. Οι τιμές αυτές καθώς και οι τιμές των text πεδίων της basket.jsp θα κληθούν και θα αποθηκευτούν μέσω της μεθόδου getParameter() σε αντικείμενα τύπου String. Τα text πεδία αναφέραμε ότι συγκρατούν την ποσότητα των προϊόντων που επιλέγει ο χρήστης. Με τη χρήση μιας if ελέγχουμε την περίπτωση που τα πεδία αυτά δεν περιέχουν τιμή. Τότε στη μεταβλητή item που συγκρατεί την ποσότητα αποθηκεύεται η τιμή μηδέν. Διαφορετικά θα αποθηκευτεί στη μεταβλητή prods η έκφραση ‘ “ποσότητα” x ”κωδικός” ’. Στη συνέχεια η τιμή της ποσότητας θα μετατραπεί σε ακέραιο και θα πολλαπλασιαστεί με την αξία του προϊόντος η οποία έχει μετατραπεί σε πραγματικό. Το αποτέλεσμα είναι η τελική αξία της παραγγελίας και αποθηκεύεται στη μεταβλητή total. Οι τιμές των μεταβλητών prods και total θα αποθηκευτούν στις αντίστοιχες του servlet basket_en. Με αυτό τον τρόπο η παραγγελία θα εμφανίζεται στο καλάθι και στην περίπτωση που ο χρήστης μεταβεί στην αγγλική έκδοση του site. Τέλος, αν ο χρήστης δεν έχει εισάγει ακέραιο αριθμό στην ποσότητα τότε θα καλεστεί η εξαίρεση NumberFormatException, θα τον ανακατευθύνει στη basket.jsp και θα τον ενημερώνει για το πρόβλημα.

```
for(int i=1;i<=numcols;i++)
{
        try
        {
```

```
        Integer i2=new Integer(i);
        String i3=i2.toString();

        String item2 = "item"+i3;
        String itempr2 = "itempr"+i3;
        String id2 = "id"+i3;

        String item = req.getParameter(item2);
        String itempr = req.getParameter(itempr2);

        String id = req.getParameter(id2);

        if(item.equals(""))
        {
                item="0";
        }
        else
        {
                int itemint = Integer.parseInt(item);
                prods = prods+"<br>"+itemint+" x "+id;
        }

        total=total+(Integer.parseInt(item)*Double.parseDouble(itempr));

        basket_en.prods=prods;
        basket_en.total=total;

        }
        catch(NumberFormatException e)
        {
                session.setAttribute("errorMessage","Πρόβλημα με την ποσότητα");
                RequestDispatcher rd =getServletConfig().
                getServletContext().getRequestDispatcher("/basket.jsp");
                rd.forward(req, res);
                return;
        }
}
```

Αφού μετατραπεί η μεταβλητή total σε String θα αποθηκευτεί μαζί με την prods στο session και θα εμφανιστούν στη σελίδα του καλαθιού αγορών (order.jsp).

```
Double total2=new Double(total);
sum=total2.toString();
        try
        {
                session.setAttribute("Message2",prods);
```

```
            session.setAttribute("Message",sum);
            RequestDispatcher rd =getServletConfig().getServletContext().
            getRequestDispatcher("/order.jsp");rd.forward(req, res);
        }
```

Σε περίπτωση που προκύψει κάποια εξαίρεση θα κατευθυνθεί στη login.jsp και θα εμφανιστεί το σχετικό μήνυμα.

```
catch (Exception e)
{
        session.setAttribute("errorMessage","Problem in DB! Try later.");
        RequestDispatcher rd =getServletConfig().getServletContext().
        getRequestDispatcher("/login.jsp");rd.forward(req, res);
        return;
}
```

clear (clear_en)

Το servlet clear καλείται από τη σελίδα του καλαθιού με σκοπό να το αδειάσει. Ουσιαστικά αφαιρεί από το session τις τιμές των μεταβλητών Message2 και Message, που συγκρατούν τα προϊόντα και την τελική τιμή, και αρχικοποιεί τις μεταβλητές prods και total του basket servlet.

```
session.removeAttribute("Message2");
session.removeAttribute("Message");
basket.prods="";
basket.total=0.0;
```

order (order_en)

Από τη σελίδα order.jsp ο χρήστης μπορεί να καταχωρήσει την παραγγελία. Αυτό επιτυγχάνεται πατώντας το κουμπί αποστολή. Τότε καλείται το servlet order. Το servlet αποθηκεύει στον πίνακα orders την παραγγελία. Μια επιπλέον ενέργεια του servlet είναι ότι θα αποθηκεύσει σε μία μεταβλητή τον κωδικό της παραγγελίας και θα το αποθηκεύσει μαζί με το προειδοποιητικό μήνυμα στο session για ενημέρωση του χρήστη. Στη συνέχεια θα αδειάσει το καλάθι αγορών, δηλαδή θα αφαιρέσει από το session το Message2 και το Message που συγκρατούν τα προϊόντα και την τελική τιμή. Επίσης θα αρχικοποιήσει τις μεταβλητές prods και total του servlet basket όπου και αυτές συγκρατούν τα ίδια δεδομένα. Τέλος θα καλέσει την mailer2.jsp την λειτουργία τη οποίας περιγράφουμε αναλυτικά στη σελίδα 34.

```
int update=stmt.executeUpdate("INSERT INTO orders
        VALUES('','"+user+"','"+prods+"',"+price+",'"+send+"','"+details+"','"+
        year+"-"+month+"-"+date+" "+hours+":"+minutes+":"+seconds+
        "','Καταχωρήθηκε','"+datetime+"');");

ResultSet rs = stmt.executeQuery("select id from orders where user='"+user+
                "' and prods='"+prods+"';");

int code=0;
while(rs.next())
{
        code = rs.getInt("id");
}

session.setAttribute("errorMessage",
                "Ευχαριστούμε. Ο κωδικός της παραγγελίας σας είναι: "+code);

session.removeAttribute("Message2");
session.removeAttribute("Message");
basket.prods="";
basket.total=0.0;
basket_en.prods="";
basket_en.total=0.0;
RequestDispatcher rd =getServletConfig().getServletContext().
                        getRequestDispatcher("/mailer2.jsp");rd.forward(req, res);
```

insertopic (insertopic_en)

Το servlet insertopic θα αποθηκεύσει στη βάση και θα εμφανίσει στη σελίδα του forum το θέμα που θα εισάγει ένας εγγεγραμμένος χρήστης. Στη περίπτωση εκείνη κατά την οποία ο χρήστης θα πατήσει το κουμπί εισαγωγής χωρίς όμως να εισάγει κάποιο θέμα, η εισαγωγή δεν θα πραγματοποιηθεί και θα υπάρξει ενημέρωση με σχετικό μήνυμα. Διαφορετικά και αν δεν προκύψει κάποια εξαίρεση η εισαγωγή θα πραγματοποιηθεί. Εκτός από το θέμα στη βάση θα αποθηκευτεί η ημερομηνία και η ώρα εισαγωγής, όπου μαζί με το θέμα θα εμφανιστούν στη σελίδα του forum.

```
import java.util.Date;
import java.text.DateFormat;
   :
   :
Date today=new Date();
int year=today.getYear()+1900;
int month=today.getMonth()+1;
int date=today.getDate();
int hours=today.getHours();
```

```
int minutes=today.getMinutes();
int seconds=today.getSeconds();
   :
   :
Statement stmt = con.createStatement();
int insert1=stmt.executeUpdate("INSERT INTO topics VALUES(
","'+year+"-"+month+"-"+date+""+hours+":"+minutes+":"+
seconds+"','"+topic+"','"+user+"');");
```

update (update_en)

Με το servlet update ο χρήστης έχει δυνατότητα να ανανεώσει τα προσωπικά του στοιχεία. Για να επιτευχθεί η ανανέωση θα πρέπει οι συνθήκες if να αποτύχουν. Οι if θα επιτύχουν στις περιπτώσεις που ο χρήστης θα διαγράψει την τιμή από ένα ή περισσότερα πεδία και όταν η τιμή του κωδικού θα διαφέρει από την τιμή του πεδίου επαλήθευσης.

```
if(name.equals("") || surname.equals("") || mobile.equals("") || username.equals("")
   || password.equals("") || email.equals("") || address.equals("") || tk.equals("")
   || city.equals("") || prefecture.equals("") || country.equals(""))
{
        session.setAttribute("errorMessage2","Παρακαλώ συμπληρώστε όλα τα πεδία!");
        RequestDispatcher rd =getServletConfig().getServletContext().
                getRequestDispatcher("/register.jsp");rd.forward(req, res);
        return;
}

if(!password2.equals(password))
{
        session.setAttribute("errorMessage2","Προσοχή! Τα πεδία Password
                                δεν έχουν την ίδια τιμή");
        RequestDispatcher rd =getServletConfig().getServletContext().
                getRequestDispatcher("/update.jsp");rd.forward(req, res);
        return;
}
```

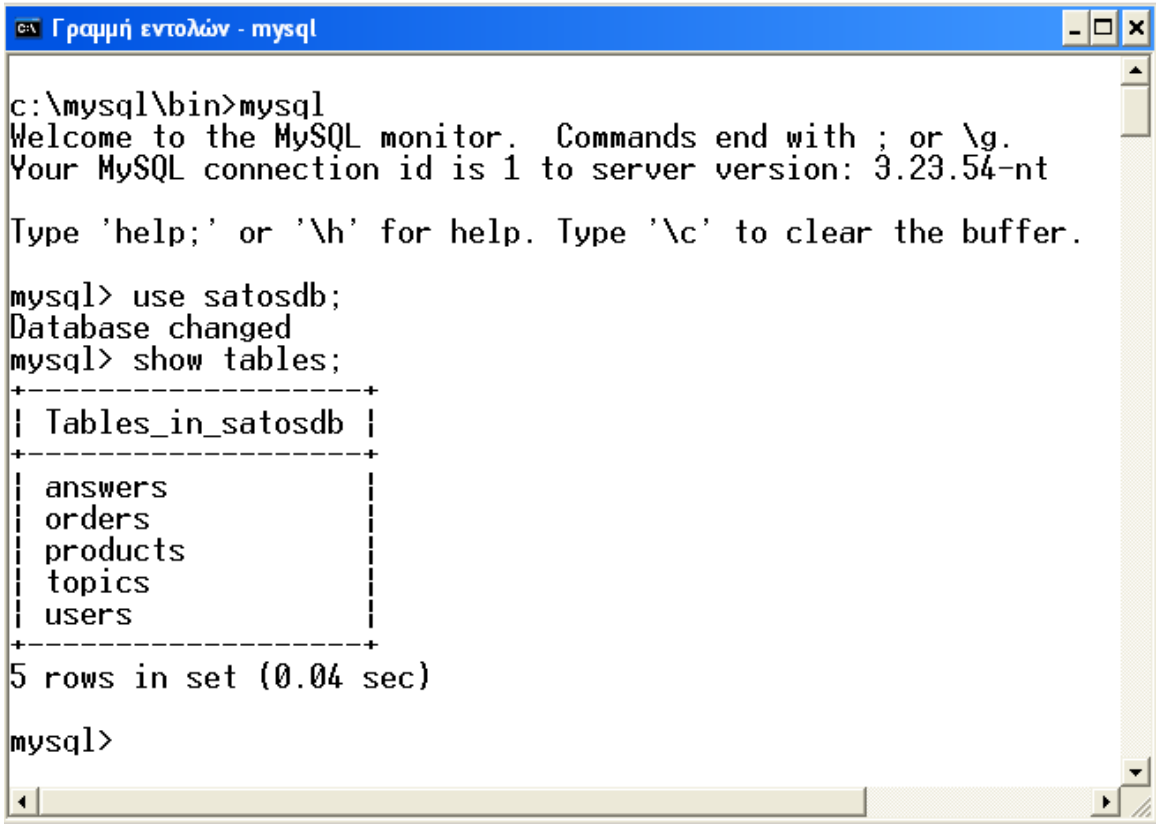
logout (logout_en)

Το servlet logout αποσυνδέει τον συνδεδεμένο χρήστη. Επίσης αδειάζει το καλάθι αγορών, κάτι που είναι λογικό μιας και δεν θα πρέπει να υπάρχουν προϊόντα όταν αποσυνδέεται ο χρήστης. Σε διαφορετική περίπτωση θα υπήρχαν προϊόντα στο καλάθι όταν θα συνδεόταν κάποιος άλλος χρήστης. Είναι προφανές ότι αυτό είναι μη θεμιτό.

```
session.removeAttribute("User");
session.removeAttribute("Message2");
session.removeAttribute("Message");
basket.prods="";
basket.total=0.0;
```

3.3. Περιγραφή της MySQL βάσης δεδομένων

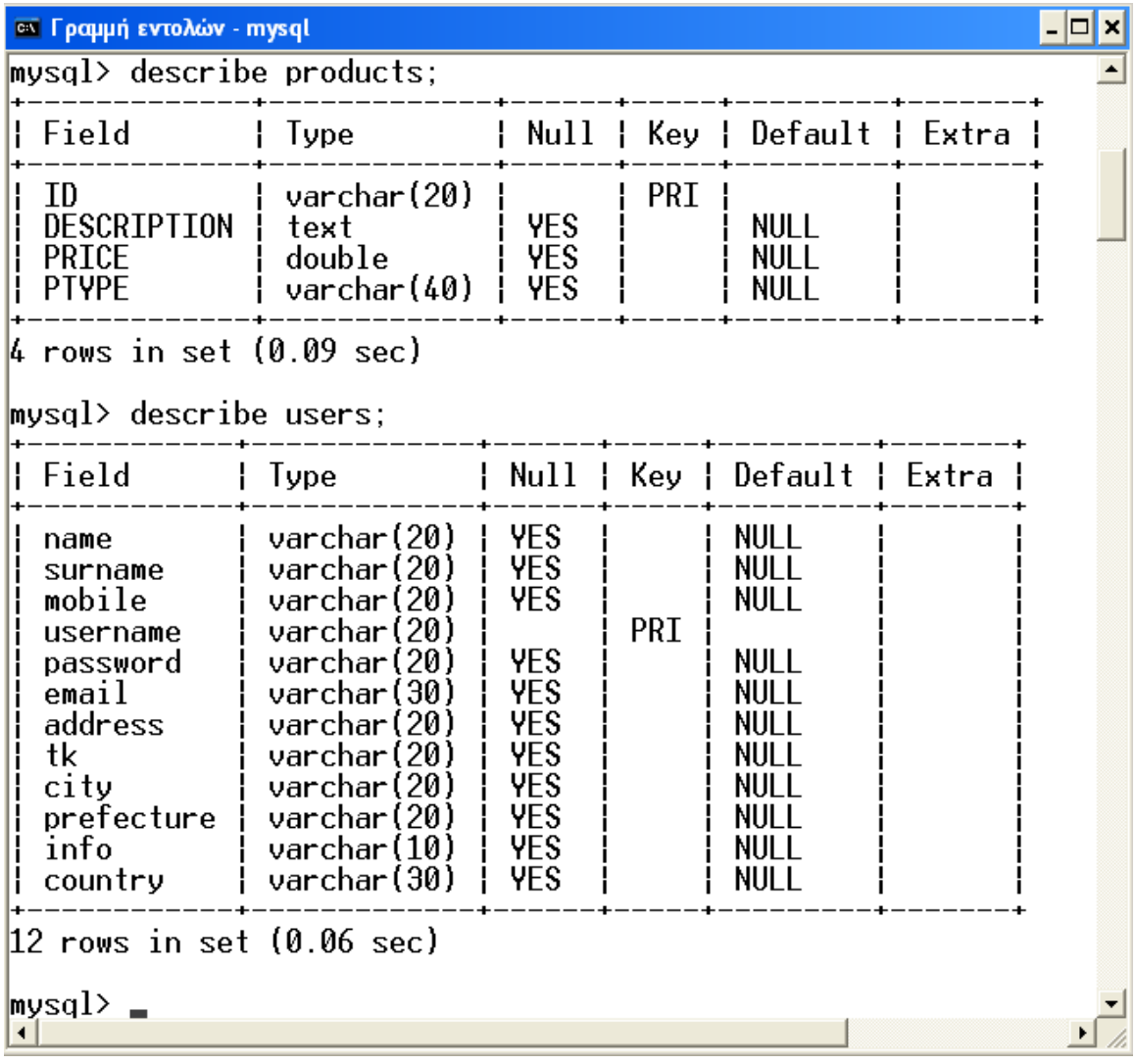
Δημιουργούμε την Βάση Δεδομένων με όνομα satosdb, η οποία αποτελείται από πέντε πίνακες. Τον πίνακα των προϊόντων (products), των χρηστών (users), των παραγγελιών (orders), των θεμάτων (topics) και των απαντήσεων (answers) του forum. Για την δημιουργία τους χρησιμοποιούμε τις παρακάτω SQL εντολές :



create database satosdb;
connect (ή use) satosdb; *// για να συνδεθούμε με την βάση*

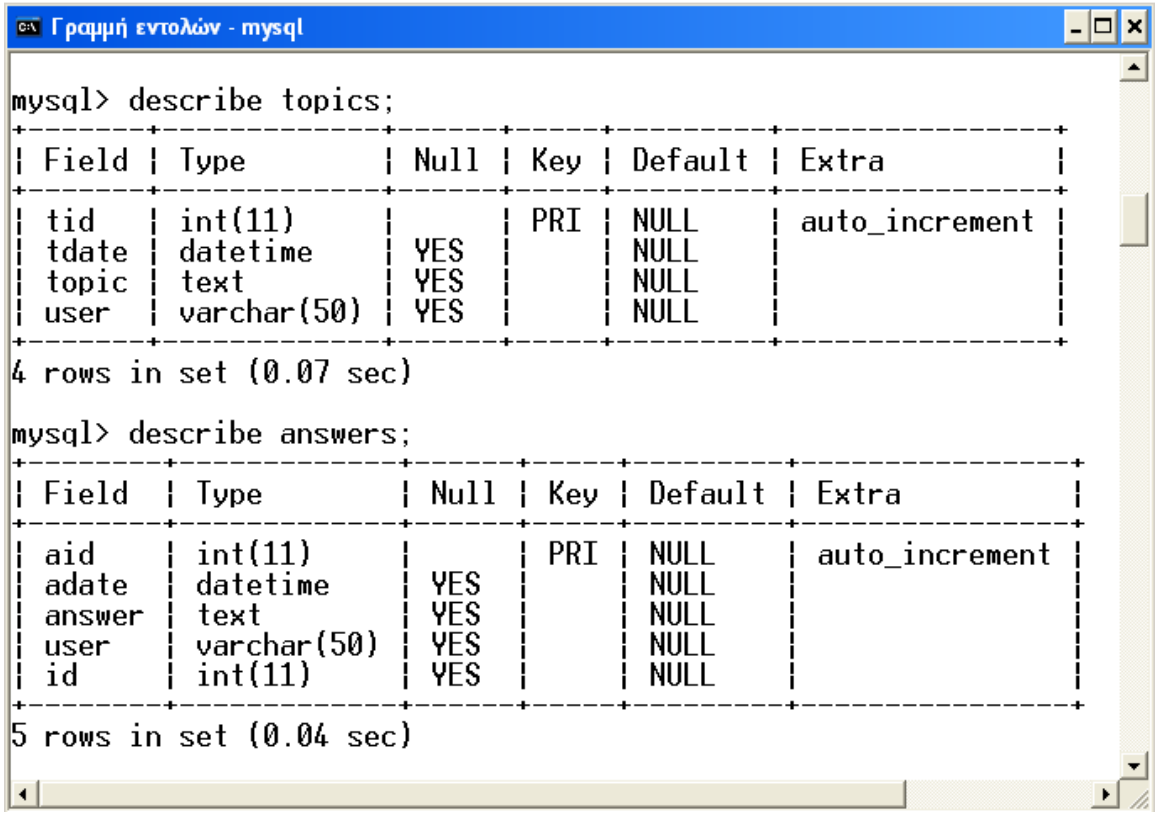
create table products (
-> id varchar(20) primary key, *// κωδικός προϊόντος*
-> description text, *// περιγραφή*
-> price double, *// τιμή*
-> ptype varchar(40)); *// είδος - κατηγορία προϊόντος*

```
create table users (
-> name varchar(20), // όνομα
-> surname varchar(20), // επίθετο
-> mobile varchar(20), // αριθμός κινητού τηλεφώνου
-> username varchar(20) primary key, // όνομα χρήστη
-> password varchar(20), // κωδικός πρόσβασης
-> email varchar(30), // ηλεκτρονική διεύθυνση
-> address varchar(20), // διεύθυνση
-> tk varchar(20), // ταχυδρομικός κώδικας
-> city varchar(20), // πόλη
-> prefecture varchar(20), // νομός
-> info varchar(10), // checkbox για ενημερωτικά e-mail
-> country varchar(30)); // χώρα
```

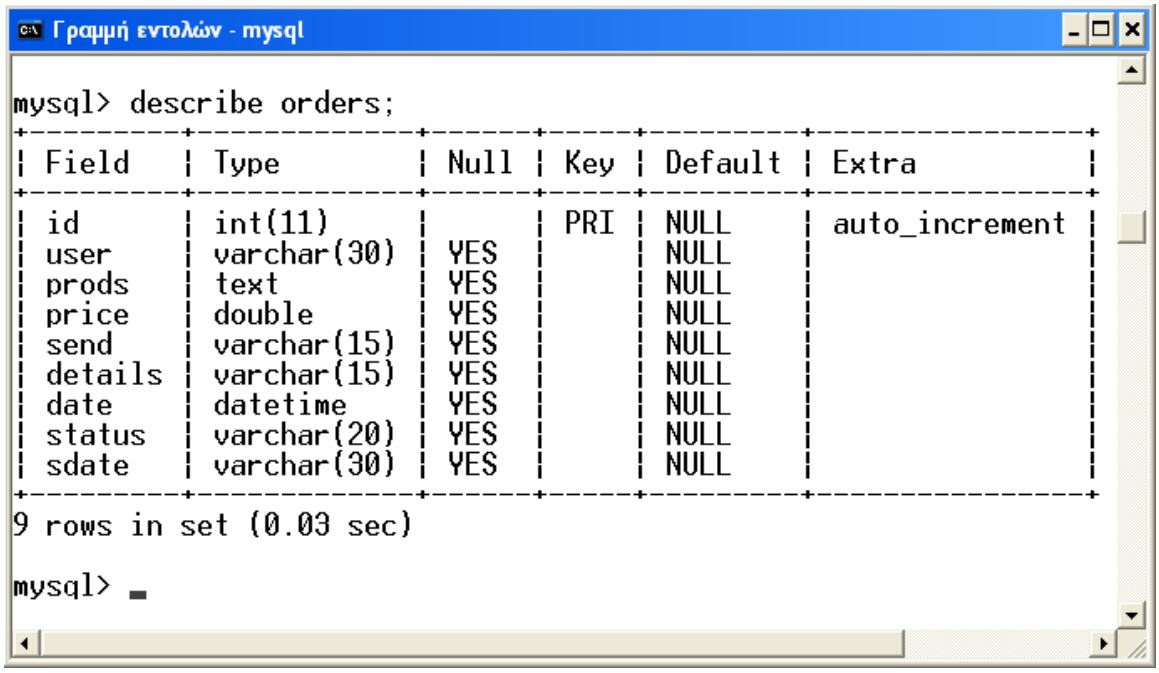


create table topics (
-> tid integer primary key auto_increment, *// κωδικός θέματος (αυτόματη αρίθμηση)*
-> tdate datetime, *// ημερομηνία εισαγωγής*
-> topic text, *// θέμα*
-> user varchar(50)); *// χρήστης που το εισήγαγε*

create table answers (
-> aid integer primary key auto_increment, *// κωδικός απάντησης (αυτόματη αρίθμηση)*
-> adate datetime, *// ημερομηνία καταχώρησης*
-> answer text, *// απάντηση*
-> user varchar(50), *// χρήστης που την εισήγαγε*
-> id integer); *// κωδικός του θέματος στο οποίο απαντά*

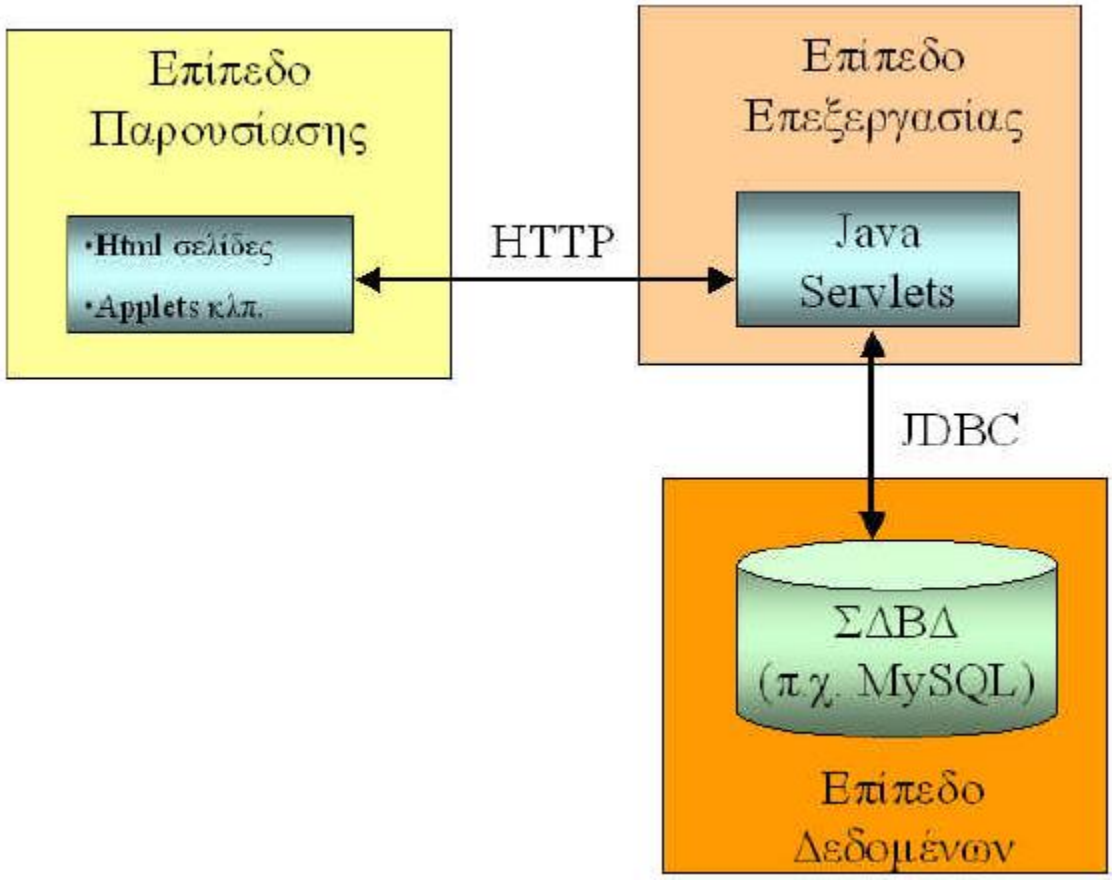


```
create table orders (
-> id integer primary key auto_increment, // κωδικός θέματος (αυτόματη αρίθμηση)
-> user varchar(30), // χρήστης
-> prods text, // προϊόντα
-> price double, // τελική τιμή
-> send varchar(15), // τρόπος αποστολής
-> details varchar(15), // λεπτομέρειες αποστολής
-> date datetime, // ημερομηνία καταχώρησης
-> status varchar(20), // κατάσταση παραγγελίας
-> sdate varchar(30)); // ημερομηνία που έγινε αλλαγή της κατάστασης
```



4. Συμπεράσματα

Μετά τη δεκάμηνη ενασχόλησή μας με την πτυχιακή μας μελέτη, αποκτήσαμε μεγάλο εύρος γνώσεων σχετικά με την κατασκευή δυναμικών web εφαρμογών και συγκεκριμένα με χρήση Java τεχνολογιών. Την λειτουργία – φιλοσοφία του ακόλουθου μοντέλου την κατανοήσαμε πλήρως και πήραμε μια ιδέα από τις δυνατότητες που προσφέρουν τα JavaServlets και οι JSP σελίδες.



πηγή : [2]

Συγκεκριμένα, με τη χρήση του session πετύχαμε την ‘σύνδεση’ των σελίδων οι οποίες, λόγω του γεγονότος ότι το HTTP πρωτόκολλο είναι stateless, συμπεριφέρονται ως ανεξάρτητες. Εκτός από την χρήση του session για την επικοινωνία σελίδων και Servlets χρησιμοποιήσαμε και τα hidden πεδία που αποδείχτηκαν εξίσου λειτουργικά. Εν κατακλείδι, μπορούμε να πούμε ότι οι τεχνολογίες που χρησιμοποιήσαμε ανταποκρίθηκαν στις αρχικές μας απαιτήσεις. Οι υπηρεσίες που προσφέρει η εφαρμογή μας υλοποιήθηκαν χάρη στις πολλές δυνατότητες (κατάλληλα ‘εργαλεία’, ευελιξία) που διαθέτουν. Αντίθετα υστέρησαν σε σχεδιαστική ευκολία αφού θα ήταν προτιμότερη η οπτική σχεδίαση της εφαρμογής με την αποφυγή δηλαδή της συγγραφής HTML κώδικα. Το τελευταίο βέβαια μπορεί να ξεπεραστεί με χρήση επιπλέον Java τεχνολογιών ή και

προγραμμάτων. Πιστεύουμε λοιπόν ότι αξίζει να τις χρησιμοποιήσει κάποιος για να υλοποιήσει web εφαρμογές.

Ως πλεονεκτήματα για τη χρήση τεχνολογιών Java μπορούμε να αναφέρουμε τα κάτωθι :

- ☑ Χρησιμοποιούν μια πολλή δυνατή γλώσσα προγραμματισμού, τη Java
- ☑ Για τη χρήση τους απαιτείται μηδενικό κόστος
- ☑ Μέσω του JDBC μπορούν να επικοινωνούν με Βάσεις Δεδομένων
- ☑ Είναι ανεξάρτητες από Λειτουργικά Συστήματα αφού εκτελούνται από τη Java Virtual Machine

Μειονεκτήματα είναι :

- ☒ Το επίπεδο δυσκολίας εκμάθησης της Java, αφού υπάρχουν πολύ πιο εύκολες γλώσσες που υλοποιούν Διαδικτυακές εφαρμογές
- ☒ Η μη υποστήριξη τους από τους servers των περισσοτέρων εταιρειών web hosting της αγοράς

5. Βιβλιογραφία

1. Γ. Κακαρόντζας, "Προγραμματισμός Internet με την τεχνολογία των JSP."
2. Γ.Κακαρόντζας,"Προγραμματισμός Internet με την τεχνολογία των JavaServlets."
3. M. Hall, "Core Servlets and JavaServer Pages."
4. Ν. Γεωργόπουλος, "Ηλεκτρονικό Επιχειρείν, Προγραμματισμός & Σχεδίαση."
5. J. C. Meloni, "Μάθετε PHP, MySQL και APACHE σε 24 ώρες."
6. M. Fowler, "Εισαγωγή στη uml."
7. Γ. Γκαράνη, "Εργαστήριο - Ειδικά θέματα Βάσεων Δεδομένων."
8. Γ. Κακαρόντζας, "Εργαστήρια τεχνολογίας λογισμικού."
9. E. Kramer, "Οπτικός οδηγός της HTML 4."
10. Χ. Κόπανος, "Εισαγωγή στην HTML και τα CSS."
11. M. Webb, "Πλήρες εγχειρίδιο της JavaScript."
12. P. Kerman, "Εγχειρίδιο του Macromedia Flash MX."
13. D. Franklin, "Macromedia Flash MX ActionScript."
14. http://java.sun.com/j2ee/index.jsp
15. http://www.jspinsider.com/content/jsp/javamail/jspjavamail_6.jsp
16. http://java.ittoolbox.com/code/d.asp?d=2378&a=s
17. http://javaalmanac.com/egs/javax.mail/SendApp.html
18. http://www.unix.org.ua/orelly/java-ent/servlet/ch13_02.htm
19. http://www.unix.org.ua/orelly/java-ent/servlet/ch04_04.htm#ch04-34777
20. http://www.javahellug.org/phpnuke/PHP-Nuke-6.5/html/index.php
21. http://rollerjm.free.fr/pro/Struts11.html