ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΗΠΕΙΡΟΥ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε.

TECHNOLOGICAL EDUCATIONAL INSTITUTE OF EPIRUS

SCHOOL OF APPLIED TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΕΦΑΡΜΟΓΗ ΤΗΣ ΤΕΧΝΟΛΟΓΙΑΣ JAVA SERVER FACES (JSF)

ΣΤΗΝ ΑΝΑΠΤΥΞΗ ΕΝΟΣ ΑΥΤΟΜΑΤΟΠΟΙΗΜΕΝΟΥ ΣΥΣΤΗΜΑΤΟΣ ΕΞΕΤΑΣΗΣ

ΜΑΥΡΟΜΑΤΗΣ ΣΥΜΕΩΝ

ΙΑΝΟΥΑΡΙΟΣ 2015

Πτυχιακή εργασία, μέρος των απαιτήσεων του τμήματος
Μηχανικών Πληροφορικής Τ.Ε.

ΕΦΑΡΜΟΓΗ ΤΗΣ ΤΕΧΝΟΛΟΓΙΑΣ JAVA SERVER FACES (JSF)

ΣΤΗΝ ΑΝΑΠΤΥΞΗ ΕΝΟΣ ΑΥΤΟΜΑΤΟΠΟΙΗΜΕΝΟΥ

ΣΥΣΤΗΜΑΤΟΣ ΕΞΕΤΑΣΗΣ

Μαυρομάτης Συμεών

Ιανουάριος 2015

ΕΥΧΑΡΙΣΤΙΕΣ

Ευχαριστώ τον καθηγητή μου Δρ. Δημήτριο Λιαροκάπη για την εμπιστοσύνη προς το άτομό μου και την καθοδήγησή του.

Ευχαριστώ τους καθηγητές μου που μου μετέδωσαν την αγάπη τους για το προγραμματισμό και μου δίδαξαν τις απεριόριστες δυνατότητες και ευκολίες που προσφέρει.

Επίσης θέλω να ευχαριστήσω τους γονείς μου που είναι πάντα δίπλα μου και στηρίζουν διακριτικά κάθε προσπάθεια μου όλα αυτά τα χρόνια.

ΠΕΡΙΕΧΟΜΕΝΑ

Προς τον αναγνώστη

Αγαπητέ αναγνώστη όπως είναι αυτονόητο δεν μπορείς να χρησιμοποιήσεις ή να αναπαράγεις μέρος ή και ολόκληρη αυτή την εργασία για την αποκομιδή προσωπικού οφέλους χωρίς την έγγραφη άδεια μου.

Ευχαριστώ,
Μαυρομάτης Συμεών

ΠΕΡΙΛΗΨΗ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

ΕΦΑΡΜΟΓΗ ΤΗΣ ΤΕΧΝΟΛΟΓΙΑΣ JAVA SERVER FACES (JSF) ΣΤΗΝ ΑΝΑΠΤΥΞΗ ΕΝΟΣ ΑΥΤΟΜΑΤΟΠΟΙΗΜΕΝΟΥ ΣΥΣΤΗΜΑΤΟΣ ΕΞΕΤΑΣΗΣ

ΙΑΝΟΥΑΡΙΟΣ 2015

ΜΑΥΡΟΜΑΤΗΣ ΣΥΜΕΩΝ

Ανώτατο Τεχνολογικό Εκπαιδευτικό Ίδρυμα Ηπείρου

Επιβλέπων Καθηγητής: **Δρ. ΔΗΜΗΤΡΙΟΣ ΛΙΑΡΟΚΑΠΗΣ**

Η συγκεκριμένη πτυχιακή εργασία αφορά την μελέτη της τεχνολογίας ανάπτυξης διαδικτυακών εφαρμογών Java Server Faces και την εφαρμογή της, στην δημιουργία μιας διαδικτυακής εφαρμογής ενός Αυτοματοποιημένου Συστήματος Εξέτασης. Η εφαρμογή περιλαμβάνει τμήμα διαχείρισης εξεταζόμενων μαθημάτων από καθηγητές και τμήμα πραγματοποίησης εξετάσεων από σπουδαστές.

Η κατασκευή της έγινε με χρήση του ενοποιημένου περιβάλλοντος ανάπτυξης Eclipse IDE το οποίο προσφέρει μια αρκετά βασική υποστήριξη της τεχνολογίας JSF. Η Βάση Δεδομένων που αποτελεί μέρος της εφαρμογής υλοποιήθηκε σε περιβάλλον MySQL Workbench το οποίο προσφέρει ένα προσιτό και εύχρηστο τρόπο κατασκευής πινάκων σε MySQL με βάση το σχεσιακό μοντέλο.

Η εκτέλεσή της βασίζεται καθαρά στον τρόπο λειτουργίας μιας διαδικτυακής εφαρμογής. Το κύριο κομμάτι της εγκαθίστανται σε έναν διακομιστή και οι αντίστοιχοι χρήστες του συστήματος που επιθυμούν πρόσβαση σε αυτό πληκτρολογούν το URL της σε έναν φυλλομετρητή.

Στην ύλη της εργασίας περιλαμβάνονται όλες οι απαραίτητες γνώσεις και προϋποθέσεις που χρησιμοποιήθηκαν για την εκπόνηση της συγκεκριμένης Πτυχιακής Εργασίας.

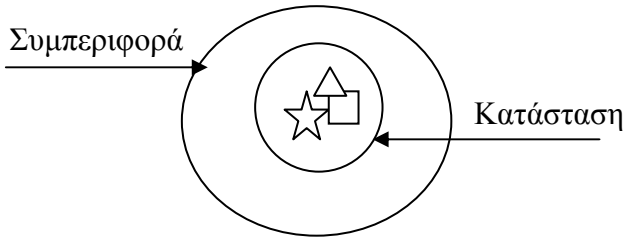
ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

1.1 Έννοιες στο πραγματικό κόσμο

Ας δούμε σε βάθος τι είναι ένα αντικείμενο.
Στον πραγματικό κόσμο που ζούμε, γύρω μας υπάρχουν διάφορα αντικείμενα , Αυτοκίνητα, Ποδήλατα, Σκύλοι, Άνθρωποι κ.τ.λ.π. Όλα αυτά τα αντικείμενα μοιράζονται δυο κοινά σημεία. Την κατάσταση τους και την συμπεριφορά τους.
Για παράδειγμα η κατάσταση του Ποδηλάτου μπορεί να είναι η Τρέχων Ταχύτητα, η Χιλιομετρική Ταχύτητα, η Πίεση του Πεταλιού. Η συμπεριφορά του η αλλαγή της Ταχύτητας, η αλλαγή της Χιλιομετρικής Ταχύτητας και η αλλαγή της Πίεσης του Πεταλιού.
Αν καταφέρουμε να αναγνωρίζουμε τις πιθανές καταστάσεις και συμπεριφορές γύρω μας έχουμε κάνει ένα μεγάλο βήμα στην κατανόηση του αντικειμενοστραφούς προγραμματισμού.

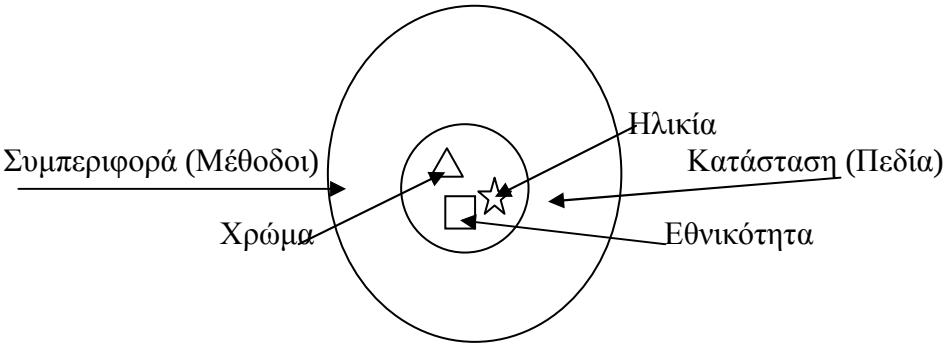
Παρατηρώντας το κάθε αντικείμενο στον χώρο μας μπορούμε να κάνουμε δύο ερωτήσεις στον εαυτό μας. Τι περιγράφει αυτό το αντικείμενο; Τι μπορεί να συμβεί σε αυτό το αντικείμενο;
Αν καταγράψουμε τις παρατηρήσεις μας θα δούμε ότι το κάθε τι γύρω μας διαφέρει σε πολυπλοκότητα. Μπορεί μια λάμπα στο δωμάτιο μας να διαθέτει μόνο δύο καταστάσεις Αναμμένη - Σβηστή και δύο συμπεριφορές Άνοιγμα – Σβήσιμο. Το ραδιόφωνο όμως μπορεί να έχει επιπλέον καταστάσεις όπως Αναμμένο, Σβηστό, Ένταση, Σταθμός και συμπεριφορές όπως Άνοιγμα, Σβήσιμο, Αύξηση της Έντασης, Μείωση της Έντασης, Αναζήτηση Σταθμού, Αποθήκευση Σταθμού κ.τ.λ.π.
'Όλες αυτές οι παρατηρήσεις λοιπόν που κάναμε στον πραγματικό κόσμο μεταφράζονται και στον αντικειμενοστραφή προγραμματισμό.



Εικόνα 1.

Ένα αντικείμενο αποθηκεύει την κατάσταση του σε πεδία (δηλαδή σε μεταβλητές προγραμματιστικά) και εκφράζει την συμπεριφορά του μέσω των μεθόδων (δηλαδή σε συναρτήσεις προγραμματιστικά).
Οι μέθοδοι λειτουργούν σαν ένας κύριος ενδιάμεσος μηχανισμός που επιδρά στο αντικείμενο και συγκεκριμένα στις μεταβλητές του αλλάζοντας έτσι την κατάσταση του.
Το να αποτρέπουμε την άμεση επικοινωνία με το αντικείμενο και το να απαιτούμε όλη αυτή η επικοινωνία να γίνεται πλάγια μέσω των μεθόδων είναι μια βασική αρχή κάθε προγραμματιστή.

Η διαδικασία αυτή ονομάζεται ενθυλάκωση του αντικειμένου.



Εικόνα 2

Με το να χαρακτηρίζουμε ένα αντικείμενο και με το να παρέχουμε μεθόδους για την αλλαγή της κατάστασης του, το αντικείμενο ασφαλίζεται ως προς το τι επιτρέπεται να αλλάξει κάποιος σε αυτό.

Για παράδειγμα είπαμε πως ο άνθρωπος είναι ένα αντικείμενο και η ηλικία όπως και το χρώμα αποτελούν την κατάσταση του. Ο φυσικός περιορισμός που προκύπτει είναι ότι η Ηλικία ενός ανθρώπου δεν θα είναι ποτέ αρνητική (Ηλικία < 0) ούτε και πολύ μεγάλη (Ηλικία > 120), το Χρώμα του δεν θα είναι ποτέ κάτι άλλο από Άσπρο, Μαύρο, Κίτρινο (Χρώμα = Άσπρο ή Μαύρο ή Κίτρινο).

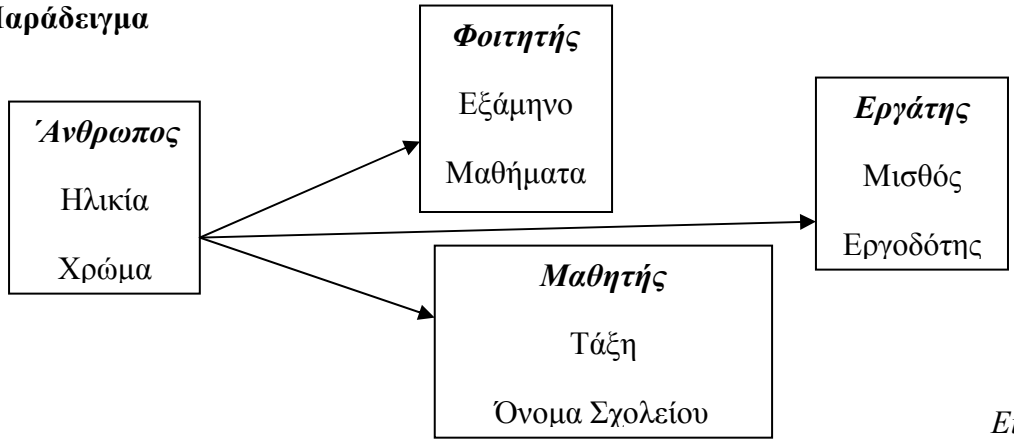
Όλοι αυτοί οι περιορισμοί μεταφράζονται σαν έλεγχοι στα πεδία και γίνονται μέσα στις μεθόδους.

Ένα ακόμα χαρακτηριστικό που μπορούμε εύκολα να παρατηρήσουμε στο κόσμο μας, είναι ότι κάθε αντικείμενο μπορεί ναι μεν να είναι κατασκευασμένο απο τα ίδιο σχέδιο και να έχει κοινά χαρακτηριστικά το ένα με το άλλο, αφετέρου δε σε ορισμένες περιπτώσεις υπάρχουν χαρακτηριστικά τα οποία προστίθενται στο αντικείμενο. Δηλαδή στο σχέδιο του και στις προηγούμενες του καταστάσεις.

Για παράδειγμα ένας άνθρωπος μπορεί να έχει όλες τις προηγούμενες καταστάσεις που έχουμε αναφέρει (Ηλικία, Χρώμα, Εθνικότητα) και να είναι έχει μία ακόμα όπως την ιδιότητα του.

Το αν κάποιος είναι φοιτητής, εργάτης, μαθητής δεν αναιρεί το σχέδιο του ως άνθρωπο αντίθετα, στις καταστάσεις που κουβαλά με την ιδιότητα του προστίθενται και η καταστάσεις του ανθρώπου. Με προγραμματιστικούς όρους κληρονομεί όλες τις καταστάσεις του ανθρώπου.

Παράδειγμα



Εικόνα 3.

1.2 Έννοιες στον προγραμματιστικό κόσμο.

Είπαμε ότι στον πραγματικό κόσμο μπορούμε να βρούμε πολλά αντικείμενα τα οποία ξεχωρίζουν μεταξύ τους όμως αυτά τα αντικείμενα ίσως ανήκουν στην ίδια κατηγορία. Για παράδειγμα στον κόσμο μας υπάρχουν πολύ άνθρωποι και όλοι μοιραζόμαστε το ίδιο σχέδιο που μας περιγράφει. Έχουμε κοινές καταστάσεις αλλά διαφορετικές τιμές, στα πεδία μας.

Με όρους αντικειμενοστραφούς προγραμματισμού θα λέγαμε ότι ο άνθρωπος για παράδειγμα μπορεί να αναπαρασταθεί σαν μία κλάση. Η κλάση είναι στην ουσία το σχέδιο που τον περιγράφει και βάση με την οποία όλοι οι άνθρωποι έχουν δημιουργηθεί.

Παράδειγμα

```
Class Human{
        Int Age = 0;
        String Color =" "
        String Nationality = " "
}
```

Πεδία της κλάσης

Εφόσον έχουμε την κλάση που περιγράφει τον άνθρωπο, τώρα μπορούμε να δημιουργήσουμε αντικείμενα, δηλαδή ανθρώπους. Τα πεδία Ηλικία (Age), Χρώμα (Color), Εθνικότητα (Nationality), αποτελούν την κατάσταση του αντικειμένου.
Αν θέλαμε να τροποποιήσουμε την κατάσταση του ανθρώπου θα έπρεπε να αλλάξουμε τις τιμές των πεδίων του. Καταρχάς για να είναι δυνατή μια τέτοια αλλαγή πρέπει να δημιουργήσουμε το αντικείμενο μας.
Αυτό γίνεται με τον ίδιο τρόπο, που δηλώνουμε μια μεταβλητή.
Η σωστή σύνταξη είναι**:** **Τύπος - Όνομα μεταβλητής** π.χ. Integer my_number**;**

Στην δικής μας περίπτωση **:** **Κλάση - Όνομα αντικειμένου** π.χ. Human Simon

Η δήλωση αυτή μας δημιουργεί το αντικείμενο με όνομα Simon σύμφωνα με το πως περιγράφεται από την κλάση Human.
Σε αντίθεση με την απλή μεταβλητή που στο παράδειγμα μας αποτελεί μια ακέραια μεταβλητή, ο Simon αποτελείται από πολλές μεταβλητές μια Ακέραια (Int Age) και δύο αλφαριθμητικές (String Color, String Nationality).

Ας επιστρέψουμε τώρα στο σημείο που θέλαμε αρχικά να τροποποιήσουμε την κατάσταση του Simon αλλάζοντας τις τιμές των πεδίων του.
Για να αναφερθούμε σε κάποιο πεδίο του Simon χρησιμοποιούμε την τελεία **(.)**.

Η σωστή σύνταξη είναι:
Simon**.Age = 24,**
Simon**.Color = White και**
Simon**.Nationality = Greek**

Η προσέγγιση αυτή αν και είναι ορθή συντακτικά δεν είναι ο πιο σωστός τρόπος για να αναφερόμαστε στα πεδία ενός αντικειμένου, γιατί έχουμε άμεση επικοινωνία με τα πεδία του. Δεν έχει γίνει ακόμα ενθυλάκωση του αντικειμένου.

Ο σωστός τρόπος αναφοράς είναι μόνο μέσο μεθόδων που θα περιλαμβάνουν και τους σχετικούς ελέγχους στα πεδία.

Οι μέθοδοι αυτοί μπορεί να είναι οι εξής : **SetAge( ), GetAge( ),**
SetColor( ), GetColor( ),
SetNationality( ), GetNationality( )

Η κλάση Human με τις νέες προσθήκες πέρνει την μορφή :

```
Class Human{
  Int Age = 0;
  String Color =" ";
  String Nationality = " ";

  SetAge(InAge) {Age =  InAge ;}
  GetAge() {return Age;}
  SetColor(InColor) {Color = InColor; }
  GetColor() {return Color;}
  SetNationality (InNationality) {Nationality = InNationality ;}
  GetNationality() {return Nationality;}

}
```

Όπως παρατηρούμε σε κάθε πεδίο (Age, Color, Nationality) αντιστοιχεί ένα ζευγάρι από μεθόδους που αρχίζουν με τα ονόματα Set και Get.Ας δούμε αναλυτικά τι ακριβώς εξυπηρετεί η κάθε μια από αυτές.

Οι μέθοδοι που αρχίζουν με το όνομα Set έχουν ως κύριο στόχο την εισαγωγή μιας τιμής στο αντίστοιχο πεδίο που αναφέρονται. Για τον λόγο αυτό η μέθοδος αυτή έχεις ως είσοδο μια εξωτερική μεταβλητή που περιέχει την τιμή που θέλουμε να εκχωρήσουμε στο πεδίο.
Αν για παράδειγμα θέλουμε να εισάγουμε στο αντικείμενο Simon που δημιουργήσαμε την Ηλικία (Age) 24,

Η σωστή σύνταξη είναι: Simon**.SetAge(24);**

Αντίστοιχα Simon.SetColor(White), Simon.SetNationality(Greek) εάν θέλαμε να ενημερώσουμε το χρώμα και την εθνικότητα του Simon με τις τιμές White και Greek.

Οι μέθοδοι που αρχίζουν με το όνομα Get έχουν τον αντίθετο στόχο από τις Set μεθόδους. Δηλαδή να μας επιστρέψουν την τιμή του ενός πεδίου. Για τον λόγο αυτό η μέθοδος αυτή δεν έχει καμία μεταβλητή ως είσοδο απλά την λέξη **return** που επιστρέφει την τιμή του πεδίου.
Οπότε αν θέλουμε να "ζητήσουμε" την ηλικία του Simon

Η σωστή σύνταξη είναι: Simon**.GetAge( );**

Αντίστοιχα Simon.GetColor( ), Simon.GetNationality( ) εάν θέλαμε το χρώμα και την εθνικότητα του Simon .

Στο σημείο αυτό πρέπει να σημειώσουμε ότι οι μέθοδοι που έχουμε αναφέρει μέχρι στιγμής έχουν ως στόχο την προσθήκη ή την λήψη των πεδίων μιας κλάσης. Όμως αν και αυτή η χρήση των μεθόδων είναι πολύ βασική και πρέπει πάντα να γίνεται κατά την δημιουργία της, αυτό δεν σημαίνει πως πρέπει να περιοριζόμαστε σε αυτή τη χρήση και μόνο. Μπορούμε πολύ εύκολα να δημιουργήσουμε και επιπλέον μεθόδους που θα πραγματοποιούν άλλες λειτουργίες που ίσως φανούν χρήσιμες στην εφαρμογή που κατασκευάζουμε.

Για παράδειγμα στην κλάση που έχουμε υλοποίηση μέχρι στιγμής μια μέθοδος που φαίνεται αρκετά χρήσιμη είναι αυτή που αυξάνει την ηλικία ενός αντικειμένου κατά έναν χρόνο. Μπορούμε να επιλέξουμε ένα σχετικό όνομα για την μέθοδο μας σύμφωνα με την λειτουργία που εκτελεί. Για παράδειγμα να την ονομάσουμε AgeProgression.

Ας δούμε λίγο ποιο αναλυτικά πως θα μπορούσε να υλοποιηθεί μια τέτοια μέθοδος.

```
AgeProgression( ){
        Age = Age + 1 ;
}
```

Όπως φαίνεται και στην υλοποίηση μας αυτό που συμβαίνει είναι ότι προσθέτουμε στην μεταβλητή **Age** την προηγούμενη τιμή της συν ένα. Πρέπει να δοθεί προσοχή στο ότι η μεταβλητή **Age** αποτελεί ένα πεδίο της κλάσης μας και η τιμή του είναι ξεχωριστή για κάθε ένα αντικείμενο που δημιουργούμε απο αυτή την κλάση.

Για παράδειγμα εάν θέλουμε να αυξήσουμε την ηλικία του Simon που έχουμε δημιουργήσει

Η σωστή σύνταξη είναι: Simon**.AgeProgression( ) ;**

Με τον τρόπο αυτό έχουμε αυξήσει την ηλικία του Simon από εικοσιτέσσερα που την ορίσαμε αρχικά σε εικοσιπέντε.

Το τελευταίο κομμάτι που μένει να αναλύσουμε για να ολοκληρώσουμε την κλάση μας (Human) είναι αυτό της κληρονομικότητας. Το σενάριο που είχαμε αναφέρει ήταν ότι ένας άνθρωπος πέρα από τις καταστάσεις που τον χαρακτηρίζουν άνθρωπο (Ηλικία, Χρώμα, Εθνικότητα) πολύ πιθανό να έχει και μια ιδιότητα στην κοινωνία. Ενδεικτικά είχαμε αναφέρει ότι μπορεί να είναι Φοιτητής, Εργάτης ή Μαθητής.

Στο παράδειγμα που ακολουθεί βλέπουμε πως μπορούμε να εφαρμόσουμε την κληρονομικότητα.

Η κλάση που παρουσιάσαμε στην αρχή.

.

```
Class Human{
        Int Age = 0;
        String Color =" "
        String Nationality = " ";

        SetAge(InAge) {Age =  InAge ;}
        GetAge( ) {return Age;}
        SetColor(InColor) {Color = InColor; }
        GetColor(  ) {return Color;}
```

```
        SetNationality (InNationality) {Nationality = InNationality ;}
        GetNationality( ) {return Nationality;}
        AgeProgression( ){
            Age =  Age + 1;
        }
}
```

Στη συνέχεια δημιουργούμε μια νέα κλάση ούτως ώστε να περιγράψουμε την ιδιότητα του Φοιτητή.

```
Class Student{
        Int Semester = 0;
        Int Courses=" " ;

        SetSemester (In Semester) { Semester =  In Semester ;}
        GetSemester () {return Semester;}
        SetCourses (In Courses) { Courses = In Courses; }
        GetCourses () {return Courses;}
}
```

Η υλοποίηση των νέων κλάσεων, είναι παρόμοια με αυτή που είχαμε παρουσιάσει στην κλάση Human. Δηλώσαμε σαν δύο ακέραιες μεταβλητές το εξάμηνο ενός φοιτητή και το σύνολο των μαθημάτων που έχει περάσει. Αυτά είναι και τα πεδία που περιγράφουν την κατάσταση του. Τέλος προσθέσαμε τις σχετικές μεθόδους που είναι απαραίτητες για την προσπέλαση των πεδίων αυτών.

Αντίστοιχα και για την κλάση που περιγράφει τον Εργάτη

```
Class Worker{
        Float Salary = 0;
        String Employer = " ";

        Set Salary (In Salary) { Salary  = InSalary;}
        GetSalary () {return Salary;}
        SetEmployer (InEmployer) { Employer = InEmployer; }
        GetEmployer () {return Employer;}
}
```

Και τον Μαθητή

```
Class Learner{
        Float  Grade  =  0;
        String  School_Name  =  " " ;

        Set Salary (In Grade) { Grade =  In Grade;}
        Get Grade () {return Grade;}
        SetSchool_Name  (InSchool_Name  ) {School_Name = School_Name; }
        Get School_Name  () {return School_Name  ;}
}
```

Μέχρι στιγμής έχουμε δημιουργήσει τις τρεις νέες κλάσεις που θα χρειαστούμε την Student, Worker και Learner. Σε αυτό το σημείο είναι να δούμε πως όλα αυτά συνδυάζονται μεταξύ τους μέσω της κληρονομικότητας.

Αρχικά πρέπει να σημειώσουμε ότι η κλάση Human που δημιουργήσαμε στην αρχή στον αντικειμενοστραφή προγραμματισμό αναφέρεται και ως κλάση **Γονέας** (**Parent Class**).Όλες οι υπόλοιπες κλάσεις που προέρχονται από αυτή όπως είναι και το προφανές αναφέρονται ως κλάσεις Παιδιά (Child Classes). Για να κληρονομήσει λοιπόν μια από αυτές τις κλάσεις (Student, Worker και Learner) τις καταστάσεις ενός ανθρώπου δεν έχουμε παρά να προσθέσουμε σε συνέχεια του ονόματος της κλάσης που μας ενδιαφέρει την λέξη Inherits και μετά το όνομα της κλάσης που θέλουμε να κληροδοτήσουμε τις καταστάσεις.
Στο παράδειγμα μας θα χρησιμοποιήσουμε την κλάση Learner καθώς η υλοποίηση είναι η ίδια και για τις υπόλοιπες κλάσεις μας.

```
Class Learner Inherits Human{
            Float  Grade  =  0;
            String  School_Name  =  ” ”;

            Set Grade (In Grade) { Grade =  In Grade;}
            Get Grade () {return Grade;}
            SetSchool_Name  (InSchool_Name  ) {School_Name = School_Name ;}
            Get School_Name  () {return School_Name  ;}
      }
```

Με τον τρόπο αυτό η κλάση Learner πέρα από τις καταστάσεις που περιείχε όταν εμείς αρχικά την δημιουργήσαμε τώρα πλέον διαθέτει και επιπλέον τις καταστάσεις της κλάσης Human. Αυτό μας προσφέρει την δυνατότητα να δημιουργούμε αντικείμενα της κλάσης Learner και να αλληλεπιδρούμε με τα πεδία της όπως ακριβώς θα κάναμε και αν η ίδια η κλάση Learner περιλάμβανε αυτά τα πεδία.
Στο σημείο αυτό ίσως δημιουργούνται απορίες όπως για παράδειγμα.
Πως ακριβώς θα αλληλεπιδράω με αυτά τα πεδία εφόσον στην κλάση Human κατασκευάσαμε ολόκληρες μεθόδους, για αυτό το έργο;
Τι γίνεται λοιπόν με αυτές τις μεθόδους;
Η απάντηση στο ερώτημα αυτό είναι απλή. Με την κληρονομικότητα μεταξύ των κλάσεων δεν κληρονομούνται μόνο τα πεδία τους αλλά και όλες οι μέθοδοι που περιλαμβάνει η κλάση Γονέας. Σε συνέπεια αυτού όλες οι μέθοδοι που είχαμε κατασκευάσει στην κλάση Γονέα για να αλληλεπιδρούμε με τα πεδίας της, είναι στην διάθεση μας για αυτό το σκοπό.

Προσοχή, διότι αυτό δεν αφορά μόνο τις μεθόδους Set και Get που είναι άκρως απαραίτητες αλλά και όλες τις υπόλοιπες που δεν έχουν άμεση σχέση με την εισαγωγή ή εξαγωγή των τιμών ενός πεδίου μιας κλάσης. Στην περίπτωση που αναλύουμε αυτή τη στιγμή έχουμε μια τέτοια μέθοδο στην κλάση Human. Αυτή είναι η μέθοδος AgeProgression( ) που αυξάνει την ηλικίας ενός ανθρώπου κατά ένα.
Ας δούμε τώρα ένα παράδειγμα που αφορά την κληρονομικότητα με το να δημιουργήσουμε ένα αντικείμενο.

Η σωστή σύνταξη είναι: **Learner Simon ή**
Student Simon ή
Worker Simon

Μια υλοποίηση ανάλογα με το τι ιδιότητα θέλουμε να δώσουμε στον Simon. Όπως έχουμε πει και για τις τρεις ο Simon παραμένει άνθρωπος. Αυτό που προσθέσαμε είναι μόνο τις καταστάσεις που περιέχει κάθε ιδιότητα που του αναθέσαμε. Αν για παράδειγμα θέλουμε τώρα να αυξήσουμε την ηλικία του Simon μπορούμε να το κάνουμε όπως είχαμε παρουσιάσει και στην περιγραφή της AgeProgression( ).

Η σωστή σύνταξη είναι: Simon.AgeProgression( )

Η ηλικία του Simon θα έχει αυξηθεί κατά ένα χρόνο. Μπορούμε επίσης να αλλάξουμε την ηλικία αυτή.

Η σωστή σύνταξη είναι: Simon.SetAge(17)

Και φυσικά να ενημερώσουμε τα πεδία της κλάσης από την οποία για δεύτερη φορά δημιουργήσαμε τον Simon.

Η σωστή σύνταξη είναι: Simon.SetGrade(6) και
Simon.SetShool_Name(2ο Λύκειο Αρτας)

Στο σημείο αυτό έχουμε ολοκληρώσει την περιγραφή των διαφόρων εννοιών που αφορούν τον αντικειμενοστραφή προγραμματισμό και είμαστε σε θέση να κατανοήσουμε πως λειτουργεί και από που πηγάζει την οργάνωση της αυτή η πλευρά του προγραμματισμού.

Στο κεφάλαιο αυτό δε σταθήκαμε μόνο σε έννοιες αλλα παρουσιάσαμε και μερικά παραδείγματα κάνοντας χρήση μιας υποθετικής γλώσσας που βασίζεται στις έννοιες που αναλύσαμε αρχικά.

ΚΕΦΑΛΑΙΟ 2

JAVA

Οι περισσότεροι άνθρωποι στον κόσμο που ασχολούνται με τους υπολογιστές και το Internet γνωρίζουν τι εστί JAVA. Είναι μια γλώσσα προγραμματισμού η οποία διαφέρει από τις άλλες γλώσσες όπως την Pascal την C ακόμα και από την C++ καθώς είναι καθαρά αντικειμενοστραφής (Object Oriented). Ορισμένα από τα βασικά χαρακτηριστικά που την κάνουν να ξεχωρίζει είναι:

- Δεν περιλαμβάνει δείκτες (Pointers) ή δομές (Structures) σε σχέση με την C, C++.
- Προσφέρει τη δυνατότητα σε έναν προγραμματιστή να δημιουργήσει προγράμματα τα οποία "τρέχουν" μέσο ενός φυλλομετρητή (Browser).Ενδεικτικά ορισμένοι φυλλομετρητές είναι ο Internet Explorer, Mozilla Firefox, Google Chrome κ.τ.λ.π.
- Χρησιμοποιεί κλάσεις (Classes) για να οργανώσει τον κώδικα σε μικρότερα κομμάτια.
- Δεν υποστηρίζει πολλαπλή κληρονομικότητα.
- Παρέχει βιβλιοθήκες με έτοιμο κώδικα για διάφορες χρήσεις , όπως δημιουργία γραφικών, χειρισμό αλφαριθμητικών στοιχείων, χειρισμό σχεσιακών βάσεων δεδομένων, δημιουργία εφαρμογών πελάτη – εξυπηρετητή (Client – Server), μαθηματικές πράξεις κ.τ.λ.π.

Στο σημείο αυτό είναι εμφανές οτι η JAVA είναι μια γλώσσα προγραμματισμού η οποία μπορεί να χρησιμοποιηθεί σε ένα ευρύ φάσμα εργασιών δημιουργία προγραμμάτων που εκτελούνται σε έναν φυλλομετρητή είναι μόνο ένας τομέας που έχει εφαρμογή αυτή η γλώσσα.

2.1 Εκδόσεις της JAVA

Η εταιρεία Oracle προσφέρει την JAVA σε τρεις μορφές - πλατφόρμες (platforms), κάθε μια από τις οποίες στοχεύει σε διαφορετικό τομέα τεχνολογίας. Οι πλατφόρμες αυτές είναι:

- **JAVA SE (Java Standard Edition).** Η ονομασία αυτή καθιερώθηκε από την έκδοση 1.6 της JAVA.Είναι η στάνταρ έκδοση της η οποία χρησιμοποιείται τόσο για εφαρμογές Παγκόσμιου Ιστού (Web Applications) αλλά και για απλές προγραμματιστικές εφαρμογές.
- **JAVA ME (Java Micro Edition).** Στις προηγούμενες εκδόσεις ονομάζονταν J2ME(Java 2 platform Micro Edition). Η έκδοση αυτή προσφέρει ένα ευέλικτο περιβάλλον για εφαρμογές που μπορούν να "τρέχουν" σε φορητές συσκευές όπως κινητά τηλέφωνα , προσωπικούς ψηφιακού βοηθούς (PDA), συστήματα πλοήγησης αυτοκινήτων (GPS) και άλλα.

- **JAVA EE (Java Enterprise Edition).** Η πρώην ονομασία της ήταν J2EE (Java 2 platform Enterprise Edition). Η έκδοση αυτή περιλαμβάνει μια σειρά από

τεχνολογίες και αρχιτεκτονικές και στοχεύει στην δημιουργία επιχειρηματικών εφαρμογών (όπως ηλεκτρονικού εμπορίου) καθώς και άλλων υπηρεσιών.

Μερικές απο τις τεχνολογίες που περιλαμβάνονται στην έκδοση JAVA EE είναι οι EJB (Enterprise Java Beans), JSF (Java Server Faces), JSP (Java Server Pages) και άλλες τεχνολογιες βασισμένες σε XML και JAVA.

.Με αυτή την έκδοση θα ασχοληθούμε στην υλοποίηση της συγκεκριμένης πτυχιακής εργασίας.

2.2 Η ιστορία της JAVA

Όλα άρχισαν το 1990 όταν μια ομάδα προγραμματιστών της εταιρείας Sun Microsystems, στην οποία συμμετείχαν οι James Gosling, Patrick Naughton και Mike Sheridan, ξεκίνησαν ένα πρόγραμμα με κωδικό όνομα “Green Project”. Ο σκοπός του έργου ήταν να δημιουργήσουν προγράμματα τα οποία θα έδιναν μια διαφορετική νότα στη χρήση και τη λειτουργία των οικιακών ηλεκτρονικών συσκευών. Η ομάδα ξεκίνησε να δημιουργεί τα προγράμματα με χρήση της C++ λόγω της ταχύτητας και του αντικειμενοστραφούς χαρακτήρα της.

Οι προγραμματιστές όμως διαπίστωσαν ότι η C++ δεν ήταν η κατάλληλη γλώσσα για τα προγράμματα που σκόπευαν να δημιουργήσουν. Η απόδοση σε ένα πρόγραμμα είναι πολύ σημαντική και ορισμένα χαρακτηριστικά της όπως η λεπτομερή διαχείριση της μνήμης, καθώς και η πολυπλοκότητα της τους δυσκόλευαν.

Έτσι λοιπόν έπρεπε να δημιουργηθεί μια καινούργια γλώσσα στα πρότυπα της C++ που να κρατά τη βασική της σύνταξη και επιπλέον να διαθέτει δυνατότητες που δεν υπήρχαν στην C++. Τον Αύγουστο του 1991 όπου ο Gosling με την βοήθεια του Patrick Naughton δημιούργησε μια νέα γλώσσα την οποία έδωσε το όνομα OAK. Ο πρώτος καρπός του Green Project ήταν η δημιουργία ενός υπολογιστή χειρός του Star Seven. Το εγχείρημα αυτό παρά το γεγονός κατασκευής του, δεν παρουσίασε ιδιαίτερη εμπορική επιτυχία.

Φτάνουμε λοιπόν τον Αύγουστο του 1994 όπου ο Billy Joy ένας από τους υπαλλήλους της Sun εκείνη την περίοδο παρατήρησε ότι η OAK θα μπορούσε να χρησιμοποιηθεί στον Παγκόσμιο Ιστό. Είναι μια εποχή την οποία το Internet ξεφεύγει από την κατάσταση απλού κειμένου HTML και αρχίζουν να χρησιμοποιούνται γραφικά μέσω του Παγκόσμιου Ιστού. Η ομάδα αντιλαμβάνεται ότι η νέα γλώσσα μπορεί να φέρει επανάσταση στην κατάσταση που διαμορφώνεται και η OAK μπαίνει σε νέα τροχιά. Το πρώτο βήμα που ακολούθησε είναι η μετονομασία της OAK σε JAVA το 1995 και λίγο αργότερα παρουσιάζεται επίσημα από την εταιρεία Sun ως εργαλείο ανάπτυξες εφαρμογών για το Internet.Δημιουργείται μάλιστα και ένα πρόγραμμα πλοήγησης αποκλειστικά γραμμένο σε JAVA το HotJava.

Η εταιρεία Netscape Communications κυκλοφορεί την έκδοση 2.0 του προγράμματος περιήγησης Netscape Navigator με υποστήριξη JAVA.Το ίδιο κάνει και η εταιρεία Microsoft για τον δικό της περιηγητή Internet Explorer 3.0.Μετά από πέντε χρόνια που πρώτο εμφανίστηκε στον χώρο, η JAVA πήρε την μορφή που έχει και σήμερα.

2.3 Έννοιες στον κόσμο της JAVA

Σε προηγούμενη ενότητα εξηγήσαμε έννοιες και παρουσιάσαμε παραδείγματα σε μια υποθετική αντικειμενοστραφής γλώσσα. Πλέον, με εφόδιο τις γνώσεις αυτές μπορούμε να επεκταθούμε στην JAVA.Θα μπορούσε να πει κάποιος τι αλλάζει; Για ποιο λόγο να μάθω ξανά τα ίδια; Ξέρω ήδη αντικειμενοστραφή. Η απάντηση θα μπορούσε να είναι απλή. Ότι δεν αλλάζει τίποτα. Όμως αυτό είναι εν μέρει και σωστό αλλά και λάθος.

Το σωστό μέρος περιλαμβάνει τις γνώσεις που έχουμε όσο αναφορά τον αντικειμενοστραφή αλλά και την αντίληψη μας. Τον τρόπο με τον οποίο έχουμε μάθει να οργανώνουμε και να αντιμετωπίζουμε τα διάφορα προβλήματα ούτως ώστε να καταφέρουμε να τα επιλύσουμε. Σημαντικό ρόλο έχει η εμπειρία που έχει αποκτήσει κανείς προγραμματίζοντας και ο όγκος των προβλημάτων που έχει επιλύσει. Αναφέρω τελευταία την εμπειρία γιατί είναι η γέφυρα που συνδέει το σωστό με το λάθος που αναφέραμε αρχικά. Πως συμβαίνει αυτό; Η γνώση, η αντίληψη είναι χαρακτηριστικά που μπορούν από μόνα τους να μας βοηθήσουν σε προγραμματιστικά προβλήματα σε όποια γλώσσα και αν προγραμματίζουμε. Η εμπειρία είναι ένα χαρακτηριστικό, που μπορεί να υπάρχει γενικά στον προγραμματισμό αλλά και ειδικά σε μια συγκεκριμένη γλώσσα. Αυτό συμβαίνει γιατί κάθε αντικειμενοστραφής γλώσσα έχει και μια δική της υλοποίηση για κάθε λειτουργία της με μικρές αλλά και μεγάλες διαφορές. Άρα λοιπόν από το γνωρίζω αντικειμενοστραφή προγραμματισμό μέχρι το γνωρίζω να προγραμματίζω σε μια γλώσσα έχει μια μικρή απόσταση και εξηγήσαμε το γιατί.

2.4 Ενοποιημένα Περιβάλλοντα Ανάπτυξης (IDE)

.Καταρχάς για να προγραμματίσει κάποιος σε JAVA αλλά και σε οποιαδήποτε τεχνολογία βασισμένη στην γλώσσα αυτή, είναι καλό να κάνει χρήση κάποιου περιβάλλοντος που έχει κατασκευαστεί για αυτή την δουλειά. .Για τον λόγο αυτό υπάρχουν προϊόντα από τρίτους κατασκευαστές ακόμα και από την ίδια την Oracle τα οποία λέγονται ενοποιημένα περιβάλλοντα ανάπτυξης (Integrated Development Enviroments, IDE) που μας βοηθούν να γράφουμε πιο εύκολα διάφορες εφαρμογές. Τα περιβάλλοντα αυτά περιλαμβάνουν συντάκτες κειμένου ακόμα και διορθωτές που βοηθούν στην πρόληψη των λαθών με αντίστοιχες ειδοποιήσεις αλλά και συμβουλές ως προς το τι έχουμε την δυνατότητα να κάνουμε. Επίσης επικοινωνούν άμεσα με έναν μεταγλωττιστή και διερμηνευτή πατώντας απλά ένα κουμπί. Μερικά από αυτά είναι τόσο εξελιγμένα που περιλαμβάνουν ακόμα και έναν εσωτερικό περιηγητή.
Τέτοια IDE είναι το NetBeans, το Jbuilder, το BlueJ, το Eclipse καθώς και πληθώρα άλλων.

Στην συγκεκριμένη πτυχιακή εργασία θα γίνει χρήση του Eclipse καθώς το συγκεκριμένο εργαλείο προσφέρει μια βασική υποστήριξη του JSF. Πρέπει να σημειώσουμε ότι το Eclipse είναι ένα πολυεργαλείο. Αυτό σημαίνει ότι εγκαθιστώντας κάποιος τα αντίστοιχα εργαλεία μπορεί να προγραμματίσει σε διάφορες τεχνολογίες. Αυτό δεν προσφέρει ευελιξία στον προγραμματιστή καθώς δεν αφιερώνεται σε μια μόνο τεχνολογία. Υπάρχουν περιβάλλοντα ανάπτυξης τα οποία ειδικεύονται αποκλειστικά και μόνο στο JSF όπως το JDeloper της Oracle.

Κλείνοντας την αναφορά στα IDE και την χρήση τους μπορούμε να συνεχίσουμε στην παρουσίαση κώδικα. Πολλά από αυτά που θα παρουσιάσουμε

αφορούν την παραδοσιακή και αρχική μορφή της γλώσσας. Η χρήση τους ήταν απαραίτητη στην υλοποίηση της συγκεκριμένης πτυχιακής

2.5 Κλάσεις

Για να ορίσουμε μια κλάση στην JAVA ο τρόπος είναι όμοιος με αυτόν που γίνεται σε όλες τις αντικειμενοστραφής γλώσσες. Με τη λέξη Class ακολουθούμενη από το όνομα που εμείς επιλέγουμε για την κλάση μας. Η πιο απλή μορφή μιας κλάσης που μπορούμε να συναντήσουμε είναι η εξής:

```
Class XXX{
}
```

Το παράδειγμα είναι μια ολοκληρωμένη κλάση της JAVA.Μπορούμε να την αποθηκεύσουμε και να την εκτελέσουμε (μεταγλωττίσουμε). Η κλάση αυτή φυσικά δεν κάνει τίποτα για δεν έχουμε καθορίσει τι να κάνει.
Ας δούμε ένα παράδειγμα στο οποίο θα ορίσουμε την κλάση Circle η οποία θα αποτελεί το σχέδιο όπως έχουμε πει ενός κύκλου.

```
Class Circle{
        static double p = 3.14; //Ο Αριθμός “π”
        static int count = 0;
        double radius;
        double x;
        double y;
}
```

Όπως φαίνεται στην κλάση μας ορίζονται πέντε μεταβλητές: Pi, count, radius, x, y. Οι μεταβλητές αυτές ανήκουν σε δύο κατηγορίες:

Στην πρώτη κατηγορία βρίσκονται οι μεταβλητές “p” και “count”.Κάθε μία από τις οποίες θεωρείται ως μεταβλητή κλάσης (class variable) και δηλώνεται με την λέξη static. Μία μεταβλητή κλάσης όπως η “p” και “count” υπάρχει έστω και αν δεν δημιουργήσουμε κανένα αντικείμενο από την κλάση μας. Για να αναφερθούμε σε ένα τέτοιο πεδίο γράφουμε το όνομα της κλάσης ακολουθούμενο από την τελεία και το όνομα της μεταβλητής που μας ενδιαφέρει.

Η σωστή σύνταξη είναι: Circle.p = 3.14 και
Circle.count = 3

Κάθε αντικείμενο της κλάσης όταν δημιουργηθεί, περιέχει την τιμή της “p” και “count”. Αν κάποια στιγμή αλλάξουμε τις τιμές αυτές τότε οι νέες τιμές θα τοποθετηθούν σε όλα τα αντικείμενα που έχουμε δημιουργήσει. Επίσης είναι δυνατό να δηλώσουμε και μεθόδους κλάσης (class methods) με την λέξη static. Αυτές οι μέθοδοι μπορούν να εκτελεστούν ακόμα και αν δεν έχουμε δημιουργήσει κανένα αντικείμενο.

Στην δεύτερη κατηγορία βρίσκονται οι μεταβλητές x, y και radius.Οι μεταβλητές αυτές λέγονται μεταβλητές στιγμιότυπου (instance variables).
Σε κάθε αντικείμενο της κλάσης Circle που δημιουργείται οι μεταβλητές αυτές υπάρχουν, αλλά έχουν διαφορετική τιμή από αντικείμενο σε αντικείμενο. Μια τέτοια

μεταβλητή δηλώνεται με τον συνήθη τρόπο. Με το είδος του τύπου της και στη συνέχεια το όνομα της.¨ Έχουμε ήδη δώσει μία βάση στο τι είναι μια μέθοδος όμως στο σημείο αυτό θα προσθέσουμε κάτι πολύ σημαντικό ως προς την συμπεριφορά τους και την χρήση τους.
Ας θυμηθούμε λίγο την δομή μια μεθόδου σε δύο γραμμές.

Τύπος Όνομα Μεθόδου (όρισμα1, όρισμα2, όρισμα3, ή και κανένα)
{
Κώδικας της μεθόδου μας
}

Ο τύπος αναφέρεται στην τιμή που επιστρέφει μια μέθοδος και μπορεί να είναι οποιοσδήποτε. Αν η μέθοδος δεν επιστρέφει καμία τιμή τότε στη θέση του τύπου βάζουμε την λέξη **void.** Τα υπόλοιπα είναι γνωστά. Το όνομα της μεθόδου το καθορίζουμε εμείς όπως επίσης και τα ορίσματα της, Αν δεν υπάρχουν ορίσματα τοποθετούμε απλώς τις παρενθέσεις. Για να επιστρέψουμε μια τιμή από μια μέθοδο κάνουμε χρήση της λέξης **return**. Ας ορίσουμε μια μέθοδο για την κλάση με το όνομα area( ), η οποία υπολογίζει το εμβαδόν κάθε κύκλου – αντικειμένου που έχουμε δημιουργήσει.

Η σωστή σύνταξη είναι:

```
double area( )
{  return p*radius*radius; }
```

Όλος και όλος μέχρι στιγμής ο κώδικας μας για την κλάση έχει ως εξής:

```
Class Circle{
        static double p = 3.14;
        static int count = 0;
        double radius;
        double x;
        double y;
        double area( ) { return p*radius*radius; } (In line τρόπος γραφής)
}
```

2.6 Κατασκευαστές

Στο σημείο αυτό θα αναφερθούμε σε μια ειδική μέθοδος που καλό θα ήταν να υπάρχει πάντα σε μια κλάση, αν και η απουσία της δεν δημιουργεί κάποιο συγκεκριμένο πρόβλημα στη μεταγλώττιση. Ένας κατασκευαστής έχει πάντα το ίδιο όνομα με την κλάση και έχει σκοπό να δίνει αρχικές τιμές στις μεταβλητές της. Δηλαδή στις μεταβλητές ενός αντικειμένου. Όπως είπαμε αρχικά ο κατασκευαστής είναι μια μέθοδος μόνο που ξεχωρίζει σε ορισμένα κομμάτια από τις υπόλοιπες.

- Έχει πάντα το ίδιο όνομα με την κλάση, διαφορετικά δεν ορίζουμε ένα κατασκευαστή αλλά μια νέα μέθοδο.

- Δεν βάζουμε μπροστά του τύπο επιστροφής ούτε καν το void.

Ένας κατασκευαστής αν και δεν επιστρέφει ποτέ τίποτα δεν περιορίζεται από πουθενά στο πόσες παραμέτρους θα έχει ως είσοδο. Μπορεί να έχει όσες θέλουμε ακόμα και καμία. Η χρήση του γίνεται κατά την δημιουργία του αντικειμένου καθώς αποτελεί μέρος της δήλωσης του. Για να δημιουργήσουμε ένα αντικείμενο στην JAVA πρέπει να ορίσουμε μια μεταβλητή που θα είναι τύπου της κλάσης μας ώστε να αναφερόμαστε σε αυτό. Στη συνέχεια κάνουμε χρήση του τελεστή "**new**" με τον κατασκευαστή που έχουμε ορίσει η όχι. Πλέον είμαστε σε θέση να ορίσουμε έναν ή και δύο κατασκευαστές για την κλάση Circle.

1ος Κατασκευαστής χωρίς παραμέτρους

```
Circle ( ){
        Radius  = 0;
        X= 0;
        Y=0;
 }
```

2ος Κατασκευαστής με παραμέτρους

```
Circle (double rin, double xin, double yin ){
        Radius  = rin;
        X= xin;
        Y=yin;
 }
```

Τώρα που έχουμε στα χέρια μας και τους κατασκευαστές για να δημιουργήσουμε ένα αντικείμενο.

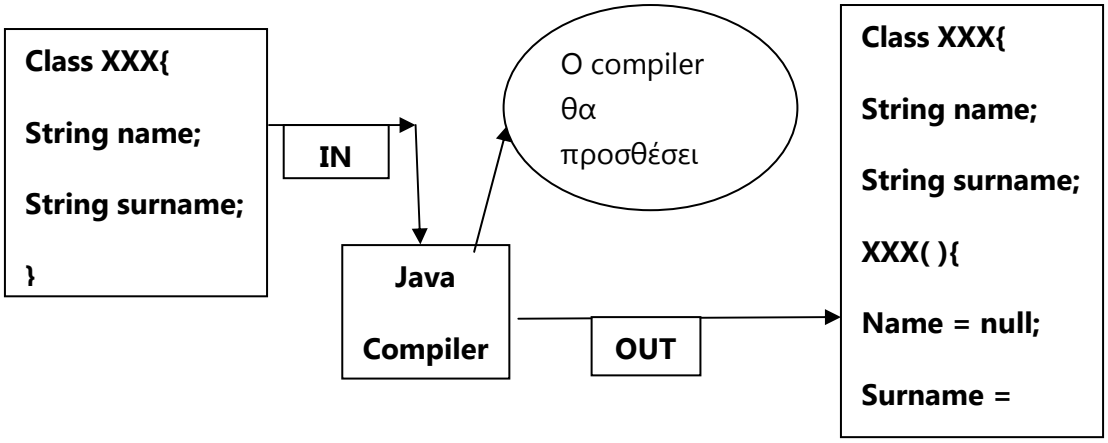
Η σωστή σύνταξη είναι:

```
Circle wheel;
wheel = new Circle( ); ή wheel( 3,5,8 );
                ή
Circle wheel = new Circle( ); ή wheel( 3,5,8 );
```

Στην πρώτη περίπτωση κάναμε χρήση του κατασκευαστή χωρίς ορίσματα άρα κατά την δημιουργία του αντικειμένου οι τιμές των radius, x, y θα είναι 0.
Στην δεύτερη κάναμε χρήση εκείνου με τα ορίσματα δίνοντας δικές μας αρχικές στο αντικείμενο. Αυτές των 3, 5 και 8.

Είπαμε στην αρχή πως αν δεν υπάρχει κατασκευαστής ένας μεταγλωττιστής δεν θα το θεωρεί λάθος. Αυτό όμως δεν είναι τελείως σωστό γιατί λαμβάνουν θέση τυχαίοι μηχανισμοί δίνοντας δικές τους τιμές. Συγκεκριμένα ο ίδιος μεταγλωττιστής προσθέτει ένα κατασκευαστή στην κλάση αν αυτή δεν περιλαμβάνει έναν. Ακόμα και στην τεχνολογία JSF λαμβάνει θέση αυτός ο μηχανισμός καθώς είναι βασισμένη σε JAVA.

Κατά την εκτέλεση των κλάσεων της συγκεκριμένης πτυχιακής εργασίας επειδή δεν ορίστηκαν κατασκευαστές η ροή που ακολούθησε ο μεταγλωττιστής περιγράφεται από το παρακάτω σχήμα:



Εικόνα 4

Όταν κατασκευάζουμε ένα πρόγραμμα δεν είναι καλό να αφήνουμε τιμές μεταβλητών στο έλεος τέτοιων μηχανισμών αλλά να έχουμε τον πλήρη έλεγχο δίνοντας πάντα δικές μας τιμές για αρχικοποίηση.

2.7 Τα πακέτα και η λειτουργία τους (Packages)

Πριν από χρόνια όταν ο προγραμματισμός βρισκόταν σε αρχικά στάδια ένας προγραμματιστής έγραφε κώδικα για να δημιουργήσει κάτι νέο. Αυτό που οραματιζόταν. Στην διάρκεια της πορείας του ήταν βέβαιο ότι θα παρατηρήσει το εξής φαινόμενο. Ένα μεγάλο κομμάτι του κώδικα του επαναλαμβανόταν σε όλα τα προγράμματα που έγραφε ακόμα και αν ένα προϊόν ήταν τελείως διαφορετικό από το άλλο .

Το συμπέρασμα που προκύπτει σε αυτό το σημείο είναι το εξής. Ο κώδικας πρέπει να οργανωθεί και να τοποθετηθεί κάπου και να αναφερόμαστε σε αυτόν όταν τον χρειαζόμαστε. Έτσι και έγινε. Τουλάχιστον για κάποιο διάστημα οι προγραμματιστές δημιουργούσαν τα δικά τους πακέτα με κώδικα τα οποία στην ουσία αποτελούσαν και τα εργαλεία τους για την κατασκευή νέων εφαρμογών. Στη συνέχεια ολόκληρες εταιρείες λάνσαραν πακέτα κώδικα τα οποία ήταν σε θέση να καλύψουν τις περισσότερες ανάγκες ενός προγραμματιστή. Αυτό έφερε νέες δυνατότητες καθώς δεν υπήρχε πια η ανάγκη να δημιουργούμε κάθε φορά τον τροχό αλλά να επενδύσουμε τον χρόνο σε κάτι καινούργιο. Τις σύγχρονες αυτές ημέρες αυτά τα πακέτα είναι από τα πιο σημαντικά εργαλεία.. Το JSF αποτελεί μέρος ενός πακέτου και προσφέρει μία σειρά από βιβλιοθήκες με έτοιμο κώδικα.

Ένα πακέτο (package) είναι μια συλλογή από κλάσεις, οι οποίες μπορούν να προσθέσουν δυνατότητες στον κώδικα. Ο κώδικας που βρίσκεται σε ένα πακέτο μπορεί να ξαναχρησιμοποιηθεί σε διαφορετικά προγράμματα. Είναι σαν μια βιβλιοθήκη. Κάθε κλάση στην JAVA πρέπει να περιέχεται σε ένα πακέτο και πολλά IDE κάνουν αυτόματα αυτή τη δουλειά. Για να χρησιμοποιήσουμε μια μέθοδο ή μια κλάση ενός πακέτου υπάρχουν δύο τρόποι.

- Αρχικά μπορούμε να κάνουμε χρήση του πλήρες ονόματος του πακέτου στη δημιουργία ενός αντικειμένου.
  Η σωστή σύνταξη π.χ. είναι **: Java.util.Date mdate = new javautil.Date( )**

  Εδώ χρησιμοποιήσαμε την κλάση Date του πακέτου Util στην δημιουργία του αντικειμένου mdate.
- Ο δεύτερος και πιο συνηθισμένος τρόπος είναι να εισαγάγουμε την κλάση στην αρχή ενός προγράμματος με την λέξη import και στη συνέχεια να δημιουργήσουμε ένα αντικείμενο τύπου της κλάσης που θέλουμε.

Η σωστή σύνταξη π.χ. είναι :

```
import java.util.date;
. . .
Date  mdate = new Date( )
```

Πολλές φορές όμως αντί να εισάγουμε μια κλάση είναι καλύτερα να εισάγουμε ολόκληρο το πακέτο κάνοντας χρήση του αστερίσκου “*”.Αυτό το χαρακτηριστικό είναι πολύ σημαντικό γιατί μπορούμε να χρησιμοποιήσουμε όλες τις κλάσεις που υπάρχουν μέσα στο πακέτο αυτό.

Η σωστή σύνταξη είναι : import java.util.*;

Στο σημείο αυτό πρέπει να δοθεί προσοχή ώστε να το όνομα της κλάσης μας να μην είναι ίδιο με κάποια από την κλάση του πακέτου.

Με απλά λόγια η JAVA οφείλει την δύναμη της και την λειτουργικότητας της στα πακέτα που διαθέτει τα οποία προσφέρουν πολύ μεγάλη ευελιξία και ανανεώνονται με κάθε νέα έκδοση της γλώσσας. Είπαμε αρχικά ότι ένα πακέτο αποτελείται από κλάσεις και μεθόδους τις οποίες μπορούμε να χρησιμοποιήσουμε σα να ήταν μέρος του κώδικα μας.
Τα πακέτα αυτά των κλάσεων ονομάζονται API (Application Programming Interface) και αυτή είναι η σωστή ορολογία αναφοράς προς αυτά.

Μερικά από τα πιο γνωστά API περιλαμβάνονται στον πίνακα που ακολουθεί.

Πακέτα της JAVA SE

Πακέτο	Περιγραφή
Java.sql	**Περιέχει κλάσεις που αφορούν την διεπαφή με βάσεις δεδομένων**
Java.lang	**Το πακέτο αυτό περιλαμβάνει βασικές κλάσεις της Java.Για την συγκεκριμένη κλάση δεν είναι απαραίτητη η χρήση του import**
Java.io	Περιέχει κλάσεις για την είσοδο και έξοδο δεδομένων
Java.net	Περιέχει κλάσεις που αφορούν δίκτυα
Java.swing	Περιέχει κλάσεις για την κατασκευή ενός GUI.Πολύ σημαντική κλάση για την υλοποίηση desktop εφαρμογών.

Ενδεικτικά από το πακέτο **Java.lang** η χρήση της κλάσης System που περιλαμβάνει είναι αρκετά χρήσιμη. Η λειτουργία της είναι να τυπώνει μηνύματα στην κονσόλα. Στην περίπτωση της διαδυκτιακής εφαρμογής που κατασκευάσαμε χρησιμοποιήθηκε σε διάφορα μέρη του κώδικα για την καλύτερη δυνατή εξφαλμάτωση, (debugging) τυπώνοντας μηνύματα ώστε να εντοπιστεί κάποιο λάθος.

Ακόμα ένα πακέτο που περιλαμβάνει πολύ βασικές κλάσεις ειδικά εάν η εφαρμογή μας περιλαμβάνει την χρήση μιας βάσης δεδομένων ανεξάρτητα απο το αν είναι web ή desktop εφαρμογή είναι το πακέτο **Java.sql**. Μερικές από τις πιο τις πιο σημαντικές κλάσεις του είναι η Statement και Connection.

Πακέτα της JAVA EE

Πακέτο	Περιγραφή
Javax.annotation	**Περιλαμβάνει τα βασικά Annotations της JAVA**
Javax.faces	**Περιλαμβάνει για το framwork JavaServerFaces**

Ενδεικτικά από το πακέτο **Java.annotation** η χρήση όλων των Annotation είναι πολύ σημαντική όταν κρίνουμε ότι είναι απαραίτητα. Τα συγκεκριμένα μας βοηθούν κάνοντας ελέγχους η παρέχοντας διάφορες πληροφορίες στον κώδικα μας. Το πιο σημαντικό από αυτά είναι το @SessionScoped.

2.8 Οι προσδιοριστές public, private και protected (Access Modifiers)

Στην JAVA πέραν του τύπου μιας μεταβλητής (Ο τύπος μπορεί να είναι ένας γνωστός τύπος δεδομένων ή ακόμα να προέρχεται και από μια κλάση) έχουμε την δυνατότητα να καθορίσουμε με την χρήση ορισμένων δεσμευμένων λέξεων το επίπεδο πρόσβασης που θέλουμε για μια μεταβλητή. Εννοώντας επίπεδο πρόσβασης αναφερόμαστε στο αν έχουμε την δυνατότητα να προσπελάσουμε μια μεταβλητή από διάφορα σημεία του κώδικα μας, από άλλες κλάσεις ή πακέτα. Οι δεσμευμένες αυτές λέξεις είναι η public, private και protected. Η χρήση τους είναι δυνατό να γίνει και για μεθόδους μιας κλάσης, πέραν των μεταβλητών. Αναλόγα αν επιθυμούμε και για αυτές μια συγκεκριμένη πρόσβαση.

- Οι μεταβλητές, οι μέθοδοι και οι κλάσεις που δηλώνονται ως public μπορούν να προσπελαστούν από οποιαδήποτε άλλη κλάση, όπου και αν βρίσκεται αυτή.
- Όσες μεταβλητές και μέθοδοι χαρακτηρίζονται ως private μπορούν να προσπελαστούν και να χρησιμοποιηθούν μόνο από το ίδιο το αντικείμενο. Δεν μπορούν να προσπελαστούν από πουθενά έξω από αυτό.
- Τέλος οι μεταβλητές και μέθοδοι που έχουν δηλωθεί ως protected μέσα σε μια κλάση μπορούν να προσπελαστούν από όλες τις κλάσεις που βρίσκονται στο ίδιο πακέτο.
- Εάν δεν κάνουμε χρήση κανενός εκ των προσδιοριστών για μια μεταβλητή ή μέθοδο τότε αυτή μπορεί να χρησιμοποιηθεί μόνο από τις κλάσεις που βρίσκονται στο ίδιο πακέτο.

Στον παρακάτω πίνακα που ακολουθεί φαίνεται η δυνατότητα προσπέλασης για τους τύπους των προσδιοριστών.

Δυνατότητα Προσπέλασης	**PUBLIC**	**PROTECTED**	**PRIVATE**	**Χωρίς Προσδιοριστή**
Από την ίδια κλάση.	ΝΑΙ	ΝΑΙ	**ΝΑΙ**	ΝΑΙ
Από τις κλάσεις του ίδιου πακέτου	**ΝΑΙ**	ΝΑΙ	**ΟΧΙ**	ΝΑΙ
Από οποιαδήποτε κλάση εκτός του πακέτου στο οποίο ανήκει η κλάση	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ
Από μια υποκλάση του ίδιου πακέτου	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ
Από μια υποκλάση έξω από το πακέτο στο οποίο ανήκει η κλάση	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ

Η στρατηγική που ακολουθείται στις περισσότερες των περιπτώσεων αλλά και αυτή που εφαρμόστηκε στην υλοποίηση της συγκεκριμένης πτυχιακής είναι οι μεταβλητές των κλάσεων να δηλώνονται ως private και οι μέθοδοι αυτών να δηλώνονται ως public.

Η σωστή σύνταξη είναι:

```
Class Circle{
        private static double p = 3.14;
        private static int count = 0;
        private double radius;
        private double x;
        private double y;
        public double area( ) { return p*radius*radius; }
        public void setradius( double radius ) { this.radius = radius;}
```

```
        public void setx( double x ) { this.x = x ;}
        public void sety( double y ) { this. y = y ;}
        public double getradius (  ) { return radius;}
        public double getx(  ) { return x;}
        public double gety(  ) { return y;}
}
```

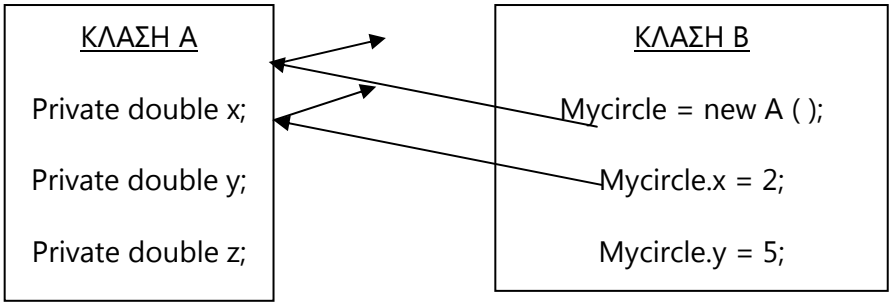
2.9 Ενθυλάκωση – Μέθοδοι Ιδιωτικής Πρόσβασης Set και Get

Με την χρήση των προσδιοριστών που αναφέραμε στο προηγούμενο κεφάλαιο έχουμε ενθυλακώσει την κλάση μας Circle. Αυτό συνέβη διότι πλέον έχουμε ορίσει την πρόσβαση που θέλουμε για αυτές και έχουμε αποτρέψει την απευθείας αναφορά σε αυτές. Υπενθυμίζω τι σημαίνει απευθείας αναφορά .

Η σωστή σύνταξη πριν την ενθυλάκωση είναι: Mycircle.radius = 2 ;

Μετά την ενθυλάκωση η συγκεκριμένη υλοποίηση είναι λάθος εφόσον έχουμε ορίσει ότι η μεταβλητή radius είναι πλέον μια private μεταβλητή. Κάποιος παρατηρητικός ίσως πει. Γιατί είναι λάθος; Αν το αντικείμενο έχει δημιουργηθεί μέσα στην ίδια την κλάση Circle δεν υπάρχει κανένα πρόβλημα και η σύνταξη είναι ίδια με το να ήταν public.Ο άνθρωπος αυτός θα έχει δίκιο.

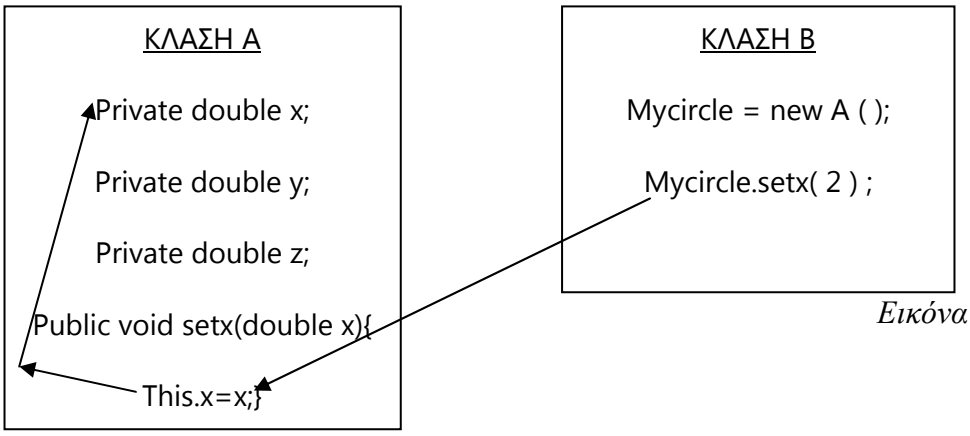
Στο σημείο αυτό πρέπει να αναφέρουμε ένα πολύ συγκεκριμένο πρόβλημα που παρουσιάζεται στο 99.9% των εφαρμογών που έχουν σχέση με τον αντικειμενοστραφή προγραμματισμό. Ελάχιστες φορές θα δημιουργούμε ένα αντικείμενο μέσα στην ίδια κλάση σε μία πραγματική εφαρμογή. Αυτό που κάνουμε συνήθως είναι να δημιουργήσουμε μια κλάση ώστε να περιγράψουμε κάτι, όμως τα αντικείμενα που προέρχονται από αυτή την κλάση θα χρησιμοποιούνται κατά βάση σε άλλες κλάσεις. Το πρόβλημα σιγά σιγά γίνεται όλο και μεγαλύτερο. Στην προηγούμενη ενότητα είπαμε για τις μεταβλητές που δηλώνονται ως private δεν έχουμε πρόσβαση από άλλες κλάσεις. Πως θα έχω επικοινωνία με αυτές αν το αντικείμενο βρίσκεται σε άλλη κλάση;



Εικόνα 5

Την απάντηση σε αυτό το ερώτημα έρχονται να δώσουν οι Μέθοδοι Ιδιωτικής Πρόσβασης (Set και.Get).Το ζευγάρι αυτών των μεθόδων πρέπει πάντα να δηλώνετε για κάθε private μεταβλητή που έχουμε ορίσει και η πρόσβαση που θα ορίζουμε πάντα για αυτές πρέπει να είναι αυστηρά public.

Με τον ορισμός τους ως public έχουμε την δυνατότητα να τις προσπελάσουμε ακόμα και αν ένα αντικείμενο είναι εκτός της κλάσης που τις περιλαμβάνει. Στην συνέχεια εφόσον αναφερθούμε σε μια μέθοδο, η ίδια η μέθοδος θα προσπελάσει την private μεταβλητή του αντικειμένου μας.



Εικόνα 6

Ας δούμε τώρα πως διαμορφώνεται η κλάση μας Circle με την προσθήκη των

Μεθόδων Ιδιωτικής Πρόσβασης Set και Get.

Η σωστή σύνταξη είναι:

```
Class Circle{
        private static double p = 3.14; //Ο Αριθμός "π"
        private static int count = 0;
        private double radius;
        private double x;
        private double y;

        public double area( ) { return p*radius*radius; }
        public void setradius( double radius ) { this.radius = radius;}
        public void setx( double x ) { this.x = x ;}
        public void sety( double y ) { this. y = y ;}
        public double getradius (  ) { return radius;}
        public double getx(  ) { return x;}
        public double gety(  ) { return y;}}
```

2.10 Η μεταβλητή "this"

Στην συγκεκριμένη πτυχιακή θα γίνει εκτεταμένη χρήση της λέξης αυτής κυρίως σε μεθόδους. Ας δούμε τι ακριβώς κάνει αυτή η λέξη. Κάθε μέθοδος στιγμιότυπου διαθέτει μια αναφορά με το όνομα **this** η οποία αναφέρεται στο τρέχον αντικείμενο για το οποίο έχει κληθεί η μέθοδος.

Στο παράδειγμα μας όταν η μέθοδος area( ) αναφερθεί στη μεταβλητή radius ο μεταγλωττιστής θα εισάγει το αντικείμενο this έτσι ώστε η παραπάνω κατάσταση να είναι ισοδύναμη με **this.radius**. Παρότι μπορεί να έχουμε πολλά αντικείμενα που προέρχονται από μια κλάση κάθε φορά στη μνήμη υπάρχει μόνο ένα αντίγραφο της

μεθόδου στην μνήμη. Κάνοντας χρήση της λέξης this μπορούμε να κάνουμε τη μέθοδο μας να εργάζεται κάθε φορά για διαφορετικό αντικείμενο. Για το αντικείμενο που βρίσκεται στη μνήμη εκείνη την στιγμή και το οποίο κάλεσε την μέθοδο.

Στο σημείο αυτό πρέπει να σημειώσουμε ότι το this αναφέρεται μόνο στο τρέχον αντικείμενο μιας κλάσης άρα μπορούμε να το χρησιμοποιήσουμε μόνο για τις μεθόδους στιγμιότυπου που προϋποθέτουν την δημιουργία ενός αντικειμένου και όχι με τις μεθόδους κλάσης για τις οποίες δεν χρειάζεται η ύπαρξη ενός αντικειμένου.

Μια υλοποίηση είναι για το παράδειγμα μας είναι η εξής:

```
Class Circle{
        . . .
        double area( ) {
         return p*radius*radius;  }
}
```

Με την προσθήκη του this το return θα διαμορφωθεί ως εξής:

return p*__this__.radius*__this__.radius;

Η προσθήκη του this, δεν είναι κάτι που αφορά μόνο τον μεταγλωττιστή. Βοηθά και τον προγραμματιστή να αποσαφηνίσει το τι ακριβώς θέλει να κάνει. ώστε να αποφύγει διάφορα προβλήματα που προκαλούν για παράδειγμα τα ονόματα μεταβλητών.

Στις μεθόδους set περνάμε συνήθως ως όρισμα μια μεταβλητή που έχει το ίδιο όνομα με το πεδίο της κλάσης στο οποίο θέλουμε να αναφερθούμε ώστε να εισάγουμε μια τιμή. Γι' αυτό στο προηγούμενο παράδειγμα που προσθέσαμε τις αντίστοιχες μεθόδους γράψαμε:

```
public void setx( double x ) { this.x = x ;}
```

Με την σύνταξη αυτή είναι ξεκάθαρο ότι αναθέτουμε την τιμή της μεταβλητής "**x**" στην μεταβλητή στιγμιότυπου "**x**" που ανήκει στο αντικείμενο.

2.11 Πίνακες και αντικείμενα

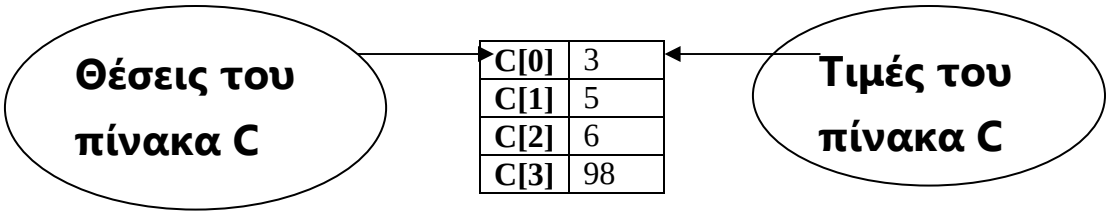
Σε αυτό το σημείο θα γίνει μια αναφορά στους πίνακες και πως ένας πίνακας μπορεί να φιλοξενήσει αντικείμενα. Ας δούμε αρχικά πως υλοποιείται ένας πίνακας για απλά δεδομένα και εν συνεχεία θα το επεκτείνουμε και για αντικείμενα.

Σε σύνθετες εφαρμογές αρκετές φορές θα έχουμε την ανάγκη να ομαδοποιήσουμε τιμές μεταβλητών που έχουν τον ίδιο τύπο. Αν δεν είχαμε στην διάθεση μας τους πίνακες, θα αναγκαζόμασταν να δηλώνουμε κάθε φορά μια ξεχωριστή μεταβλητή για κάθε μια από αυτές τις τιμές που έχουν τον ίδιο τύπο. Για να μην καταφύγουμε σε αυτή την λύση θα κάνουμε χρήση ενός πίνακα.

Οι θέσεις ενός πίνακα έχουν πάντα ένα κοινό όνομα και στην συνέχεια ακολουθεί ένας δείκτης ώστε να ξεχωρίζουμε σε ποια θέση αναφερόμαστε. Ο δείκτης θέσης είναι ένας ακέραιος αριθμός καθώς μια θέση δεν μπορεί να προσδιοριστεί με δεκαδικούς αριθμούς.

Στο σχήμα που ακολουθεί μπορούμε να δούμε έναν πίνακα.

Στο συγκεκριμένο παράδειγμα βλέπουμε ένα πίνακα



Εικόνα 7

του οποίου το όνομα είναι C και περιέχει τέσσερεις τιμές στις αντίστοιχες θέσεις C[0], C[1], C[2], C[3].Όπως φαίνεται και στο σχήμα η πρώτη θέση του πίνακα είναι πάντα αυτή που δείχνει στη θέση 0 στην περίπτωση η C[0].

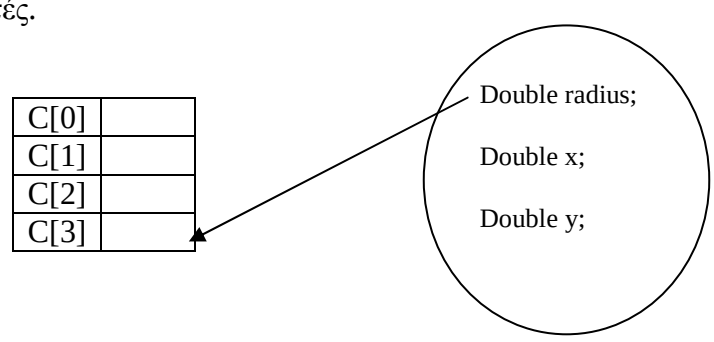
Όπως κάθε μεταβλητή έχει ένα συγκεκριμένο τύπο int, String, double έτσι πρέπει να έχει και ο πίνακας. Πέραν των τύπων που αναφέραμε μπορούμε να ορίσουμε και δικό μας τύπο ο οποίος να προέρχεται από μια κλάση. Για να δηλώσουμε έναν πίνακα ο τρόπος είναι ίδιος με αυτών των μεταβλητών.

Η σωστή σύνταξη είναι :
Int C[ ]; ή int [ ]C;

Και οι δύο δηλώσεις είναι ισοδύναμες.
Αν θέλαμε να δηλώσουμε έναν πίνακα αντικειμένων για την κλάση Circle

Η σωστή σύνταξη είναι :
Circle C[ ]; ή Circle [ ]C;

Εάν έχουμε υλοποιήσει, σε κάποιο σημείο του κώδικα μας την κλάση Circle δεν υπάρχει κανένα πρόβλημα να την χρησιμοποιήσουμε σαν τύπο για τον πίνακα μας. Η διαφορά που υπάρχει σε σχέση με τον προηγούμενο πίνακα C[ ] είναι ότι τώρα κάθε θέση του πίνακα δεν περιέχει μόνο μια τιμή αλλά πολλές τιμές. Κάτι που δεν είναι καθόλου περίεργο εφόσον εκ φύσεως τα αντικείμενα αποτελούνται από πολλές τιμές - μεταβλητές.



Εικόνα 8

Πέραν των μεταβλητών κάθε θέση περιέχει και τις μεθόδους (Set, Get) τους κατασκευαστές και γενικά ότι περιέχει η κλάση Circle και αφορά τα αντικείμενα.

Στο σημείο αυτό έχουμε απλώς δηλώσει με ποια μεταβλητή θα αναφερόμαστε στον πίνακα μας. Ακόμα όμως δε τον έχουμε δημιουργήσει, δηλαδή δεν έχουμε δεσμεύσει χώρο για αυτόν στη μνήμη. Για να το κάνουμε αυτό

Η σωστή σύνταξη είναι :

Circle [ ]C = new [4]Circle( );

Με τον τρόπο αυτό έχουμε στην ουσία δημιουργήσει τέσσερα διαφορετικά αντικείμενα τύπου Circle τα οποία είναι αρχικοποιημένα με 0 καθώς καλέσαμε τον κατασκευαστή χωρίς εισόδους που έχουμε υλοποιήσει σε προηγούμενη ενότητα.

Η έννοια του πίνακα, επεκτείνεται και σε άλλες μορφές όπως οι λίστες. Κυρίως αυτές χρησιμοποιήθηκαν για την παραμετροποίηση διαφόρων εργαλείων που ανήκουν στην τεχνολογία JSF.

ΚΕΦΑΛΑΙΟ 3

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

3.1 Έννοιες Βάσεων Δεδομένων

Σιγά- σιγά έχουμε αναφέρει όλες τις βασικές έννοιες και είμαστε σε θέση να αναγνωρίζουμε πως λειτουργούν αρκετές εργασίες στην JAVA και σε οποιοδήποτε τεχνολογία προέρχεται βασισμένη σε αυτή την γλώσσα όπως το JSF. Στο κεφάλαιο αυτό θα επεκτείνουμε τις γνώσεις μας στις βάσεις δεδομένων (Databases) επισημαίνοντας ορισμένα θέματα για αυτές, αλλά μας ενδιαφέρει κυρίως ο τρόπος με τον οποίο θα συνδέουμε μια εφαρμογή με μια βάση δεδομένων ώστε να εισάγουμε δεδομένα και να εξάγουμε πληροφορίες.

Με τον όρο βάση δεδομένων εννοούμε οργανωμένα δεδομένα τα οποία αντιστοιχούν σε μια συγκεκριμένη ενότητα (π.χ. ένα άτομο, ένα μάθημα, ή μια εξέταση). Η οργάνωση αυτή μας επιτρέπει να καταχωρούμε εύκολα δεδομένα και να εξάγουμε πληροφορίες.

Υπάρχουν πολλά μοντέλα βάσεων όμως εμείς θα κάνουμε χρήση του σχεσιακού μοντέλου καθώς είναι αρκετά εύχρηστο αλλά και το πιο δημοφιλές. Στο μοντέλο αυτό τα δεδομένα μας είναι οργανωμένα σε πίνακες. Ο κάθε πίνακας έχει γραμμές τις οποίες ονομάζουμε εγγραφές (records) και στήλες τις οποίες ονομάζουμε πεδία (fields).Οι εγγραφές του πίνακα αντιπροσωπεύουν κάθε ένα ξεχωριστό μάθημα, φοιτητή, εξέταση π.χ. ,ανάλογα πάντα με την θεματολογία που έχουμε για την βάση μας. Ο πίνακας που ακολουθεί ονομάζεται Users και αντιπροσωπεύει τους χρήστες του συστήματος διαδικτυακής εξέτασης.. Κάθε γραμμή του δηλαδή είναι και ένας διαφορετικός χρήστης με διαφορετικές τιμές στα πεδία του.
Τα πεδία αυτά είναι το username, password και level.

Πίνακας Users

username	**password**	**level**
10840	mypass	1
9854	Paswort	1
P1421	logon	2

Για παράδειγμα η πρώτη εγγραφή που βλέπουμε στον πίνακα είναι ένας χρήστης που έχει username “10840”, password “mypass” και το επίπεδο του ως χρήστης είναι 1.Με την ίδια λογική ακολουθούν και οι υπόλοιποι.

Στο σημείο αυτό πρέπει να αναφέρουμε κάτι πολύ βασικό για τις βάσεις δεδομένων. Το **πρωτεύον κλειδί**. Ένα από τα πεδία κάθε εγγραφής η και ο συνδυασμός κάποιον πεδίων πρέπει να είναι σε θέση να ξεχωρίζει κάθε μια εγγραφή ώστε να μην υπάρχει δεύτερη με τα ίδια στοιχεία. Το πεδίο αυτό ορίζεται ως πρωτεύον κλειδί.

Στις περισσότερες εφαρμογές συνήθως θα έχουμε πολλούς πίνακες και με βάση το σχεσιακό μοντέλο που ακολουθούμε θα είμαστε σε θέση να ενώνουμε τους πίνακες μεταξύ τους. Μια τέτοια σύνδεση μπορεί να γίνει με ένα κοινό πεδίο το οποίο θα υπάρχει και στους δύο πίνακες. Η διαφορά τους είναι ότι στον ένα πίνακα το πεδίο αυτό θα είναι πάντα το πρωτεύον κλειδί και στον άλλο θα είναι ένα **ξένο κλειδί** (foreign key)..

Στους πίνακες που ακολουθούν βλέπουμε ένα παράδειγμα σύνδεσης
Πίνακας Courses

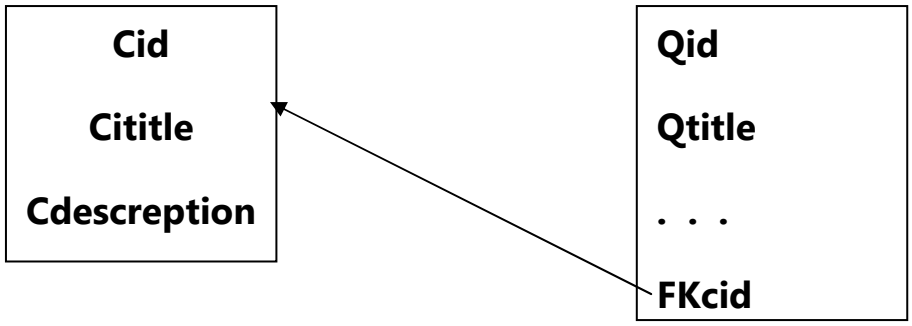
Cid	Ctitle	Cdescreption
1	JAVA	Classes,jsf,J2EE
2	NETWORKS	Tcp/Ip
3	DATA BASES	Basics

Πίνακας Questions

Qid	Qtitle	Qanswer1	Qanswer2	Qcorrectanswer	FKcid
1	Who was..			2	1
2	How can..			1	2
3	In case of.			1	1
4	What if..			2	3

Ο πρώτος πίνακας αντιπροσωπεύει διάφορα μαθήματα και αποτελείται από τρία πεδία το cid που αποτελεί και κλειδί για τον πίνακα το ctitle που είναι ο τίτλος του μαθήματος και το cdescreption για μια σύντομη περιγραφή του μαθήματος.

Ο δεύτερος πίνακας αντιπροσωπεύει ερωτήσεις που αντιστοιχούν σε κάποιο μάθημα με έξη πεδία. Το qid αποτελεί το κλειδί του δεύτερου πίνακα, το qtitle είναι ο τίτλος της ερώτησης και στη συνέχεια ακολουθούν δύο πεδία για τις πιθανές απαντήσεις, ένα πεδίο για να προσδιορίσουμε ποια είναι η σωστή απάντηση εκ των δύο και τέλος ένα πεδίο που ονομάζεται fkcid. Αυτό το πεδίο φιλοξενεί τα ξένα κλειδιά και δείχνει σε ποιο μάθημα αντιστοιχεί κάθε ερώτηση.



Εικόνα 9

3.2 Εντολές SQL

Η γλώσσα SQL είναι μια γλώσσα που δημιουργήθηκε για να αλληλεπιδρά με τις βάσεις δεδομένων. Επί του έργου στον κώδικα που θα γράφουμε ο οποίος θα είναι σε JAVA θα περνάμε εντολές SQL που θα απευθύνονται σε μία βάση δεδομένων.

Οι εντολές αυτές είναι λίγες και απλές στην σύνταξη και υπέρ αρκετές για να πραγματοποιήσουμε την όποια λειτουργία επιθυμούμε.

Ας δούμε λίγο ποιο αναλυτικά τις εντολές αυτές.

- **CREATE DATABASE** όνομα_βάσης_δεδομένων
- **CREATE TABLE** όνομα_πίνακα (όνομα_στήλης τύπος_δεδομένου,.. όνομα_στήλης τύπος_δεδομένου)

Τύποι δεδομένων SQL	**Περιγραφή**
VARCHAR	Ομάδα χαρακτήρων
DATE	Ημερομηνία
INTEGER	Ακέραιες τιμές

Παράδειγμα

```
CREATE TABLE Users (
        username varchar (30),
        password varchar (30),
        level int (10)
        PRIMARY KEY (username,password));

CREATE TABLE Questions (
        qid int (20) unsigned NOT NULL auto_increment,
        qtitle varchar (100),
        qanswer1 varchar (100),
        qanswer2 varchar (100),
        qcorrectanswer int (20),
        FKcid varchar (20),
        PRIMARY KEY (qid),
        FOREIGN KEY (FKcid) REFERENCES Courses (cid)) ;
```

- **ALTER TABLE** όνομα_πίνακα **ADD** (όνομα_στήλης τύπος_δεδομένου,.. όνομα_στήλης τύπος_δεδομένου)
- **DROP TABLE** όνομα_πίνακα
- **INSERT INTO** όνομα_πίνακα **VALUES** (είσοδος1, είσοδος 2,..)
- **DELETE FROM** όνομα_πίνακα **WHERE** συνθήκη
- **UPDATE** όνομα_πίνακα **SET** όνομα_στήλης=τιμή **WHERE συνθήκη**
- **SELECT** όνομα_στήλης **FROM** όνομα_πίνακα **WHERE** συνθήκη

  Παράδειγμα

  SELECT * FROM Users

Αν και όλες οι εντολές είναι σημαντικές η SELECT είναι αυτή που κατά βάση θα χρησιμοποιούμε περισσότερο καθώς η χρήση της αφορά την αναζήτηση και ανάκτηση πληροφοριών από τους πίνακες.

3.3 Σύνδεση με τον JDBC

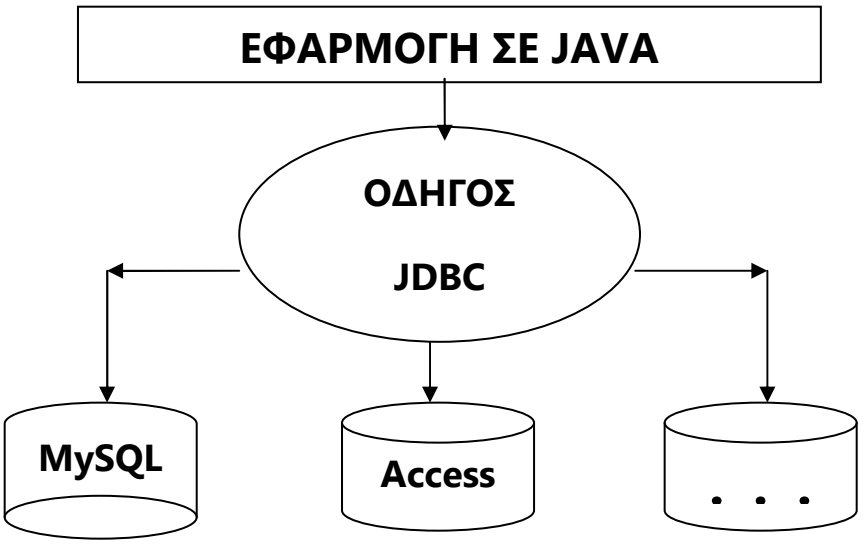
Στο σημείο αυτό γνωρίζουμε τι εστί πίνακας σε μια βάση δεδομένων και έχουμε εις γνώση της εντολές που δομούν την γλώσσα SQL.Πλέον ,είμαστε σε θέση να δούμε πως μπορούμε να συνδέσουμε μια εφαρμογή γραμμένη σε JAVA με μια βάση δεδομένων.Ο τρόπος είναι σχετικά εύκολος καθώς τα βήματα που πρέπει να γίνουν είναι πολύ συγκεκριμένα κάθε φορά πλην ελάχιστων εξαιρέσεων κυρίως στο πέρασμα μεταβλητών σε εντολές SQL.

Για να καταλήξουμε στην ουσία το πρόβλημα της επικοινωνίας της JAVA με μία βάση δεδομένων λύνει ο οδηγός **JDBC**. Με την χρήση του συγκεκριμένου οδηγού η γλώσσα έχει την δυνατότητα να υποστηρίζει σχεσιακές βάσεις δεδομένων από διάφορες εταιρείες όπως η IBM, Mircrosoft, Sybase και Oracle.

Αναλυτικά το JDBC δίνει στον προγραμματιστή τις εξής δυνατότητες:

- Να συνδέει μια εφαρμογή σε μία βάση δεδομένων
- Να δημιουργεί εντολές SQL και να τις εισάγει σε μια βάση δεδομένων ούτως ώστε να εκτελέσουμε διάφορες λειτουργίες
- Να λάβει αποτελέσματα της επεξεργασίας των εντολών SQL από μία βάση δεδομένων
- Να κάνει επεξεργασία των αποτελεσμάτων που έλαβε

Με την χρήση αυτού του ενδιάμεσου επιπέδου μια εφαρμογή αποδεσμεύεται από την εκάστοτε βάση δεδομένων και από την πλευρά του προγραμματιστή η εφαρμογή παρουσιάζεται ως εξής.



Εικόνα 10

Αναλυτικά το JDBC περιλαμβάνει

- **Τον Driver Manager**.
  Είναι υπεύθυνος για να συνδέει τον σωστό οδηγό JDBC ανάλογα με την βάση δεδομένων στην οποία θέλουμε να συνδέσουμε την εφαρμογή μας. Μόλις υπάρξει σύνδεση ο Driver Manager πραγματοποιεί την επικοινωνία μεταξύ του JDBC και της εφαρμογής

- **Connection**
  Αποτελεί την σύνδεση με την βάση δεδομένων. Η επικοινωνία με την Β.Δ. γίνεται μέσω των μεθόδων που περιλαμβάνει η Connection.

- **Statement**
  Είναι μια εντολή SQL.Η εκτέλεση των εντολών SQL που έχουμε αναφέρει πραγματοποιούνται μέσω των μεθόδων που περιλαμβάνει η Statement.
- **ResultSet**
  Περιέχει τα αποτελέσματα που λαμβάνουμε από την εκτέλεση των εντολών SQL.Με τις μεθόδους της έχουμε την δυνατότητα να διαχειριστούμε τα αποτελέσματα αυτά.

Στην αρχή του κεφαλαίου είπαμε ότι η διαδικασία σύνδεσης με την βάση είναι εύκολη γιατί τα βήματα που πρέπει να γίνουν είναι συγκεκριμένα. Ένα πρόγραμμα λοιπόν ακολουθεί τα εξής βήματα για να συνδεθεί..

1. Εισάγει κλάσεις που χρειάζεται
2. Φορτώνει τον κατάλληλο οδηγό
3. Προσδιορίζει την πηγή των δεδομένων
4. Δημιουργεί ένα αντικείμενο της κλάσης Connection
5. Δημιουργεί ένα αντικείμενο της κλάσης Statement
6. Προσδιορίζει ένα ερώτημα με την βοήθεια του Statement αντικειμένου
7. Παίρνει δεδομένα (αν υπάρχουν) με το αντικείμενο ResultSet
8. Κλείνει την σύνδεση του αντικειμένου Connection

Στο παράδειγμα που ακολουθεί έχει γραφτεί μία μέθοδος getcoursess( ) για την σύνδεση με τη βάση δεδομένων “myapp” της εφαρμογής μας ώστε να πάρουμε τους τίτλους όλων των μαθημάτων που περιλαμβάνει.

```
import java.sql.*
public List<SelectItem> getcoursess()
        {
```

```
        Connection c = null;
        java.sql.Statement smt = null;
        List<SelectItem> retVal = new ArrayList<SelectItem>();

        try {

            Class.forName("com.mysql.jdbc.Driver");
c=DriverManager.getConnection("jdbc:mysql://localhost:3306/myapp","root",   "root");

            String s = "SELECT ctitle FROM courses";
            smt= c.createStatement();
            ResultSet rs = smt.executeQuery(s);
            while (rs.next( ))
               { retVal.add(new SelectItem(rs.getString("ctitle")));}
            }
        catch(Exception e)
            { System.out.println("Error Connecting To DataBase" );  }

        finally {
            try
               {c.close();}
              catch(Exception e)
               { System.out.println("Error On Close up");}
               }
            return retVal; }
```

Το πρόγραμμα. λειτουργεί ως εξής. Αρχικά γίνεται η εισαγωγή των απαραίτητων κλάσεων. Όπως έχουμε ήδη αναφέρει οι κλάσεις που αφορούν τις βάσεις δεδομένων είναι μέρος του πακέτου java.sql.*

Η σωστή σύνταξη είναι: import java.sql.*

Στη συνέχεια φορτώνουμε τους κατάλληλους οδηγούς με την μέθοδο forName() της κλάσης Class.

Η σωστή σύνταξη είναι: Class.*forName*("com.mysql.jdbc.Driver");

Επειδή για την συγκεκριμένη εφαρμογή έχει γίνει χρήση της βάσης MySQL ο οδηγός που χρειαζόμαστε είναι ο "**com.mysql.jdbc.Driver**".Σε περίπτωση που η βάση έχει κατασκευαστεί διαφορετικά ο οδηγός θα είναι άλλος και ο συγκεκριμένος δεν θα ισχύει.

Στη συνέχεια έχουμε δηλώσει ένα αντικείμενο Connection με όνομα "c".Το αντικείμενο αυτό πραγματοποιεί τη σύνδεση με την πηγή των δεδομένων και κατασκευάζει το περιβάλλον που θα κάνουμε χρήση για να εκτελούμε διάφορες εντολές. Το αντικείμενο ουσιαστικά θα δημιουργηθεί με την μέθοδο getConnection( ) της κλάσης DriverManager.

Η σωστή σύνταξη είναι:

```
Connection c = null;
c = DriverManager.getConnection("jdbc:mysql://localhost:3306/myapp","root", "root");
```

Στο σημείο αυτό πρέπει να αναφέρουμε ορισμένες πληροφορίες όσο αναφορά το όρισμα που περνάμε στην μέθοδο getConnection( ). Το όρισμα αυτό είναι ένα URL, καθώς δηλώνει ποιοι είναι και που βρίσκονται οι πόροι που χρειαζόμαστε για να συνδεθούμε. Η αρχική του μορφή είναι η εξής:

Jdbc:<υποπρωτόκολλο>://host:port/όνομα_βάσης,username,κωδικός

Το υποπρωτόκολλο έχει σχέση με το που έχει υλοποιηθεί η βάση δεδομένων. Στην περίπτωση μας που είναι σε MySQL γράφουμε mysql. Αν η βάση είναι σε MS ACCESS για παράδειγμα θα γράφαμε odbc.

Στην συνέχεια ορίζεται που είναι εγκατεστημένη αυτή η βάση ώστε να την προσπελάσουμε. Το host δηλαδή μπορεί να είναι μια IP ενός εξυπηρετητή. Στην περίπτωση που εξυπηρετητής είναι ο υπολογιστής μας γράφουμε απλά localhost ακολουθούμενο από την θύρα στην οποία ακούει η βάση δεδομένων. Η εργοστασιακή ρύθμιση της MySQL προβλέπει την θύρα **3306**.

Τέλος ορίζουμε τα στοιχεία που έχουν σχέση με την βάση δεδομένων. Πρώτα γράφουμε το όνομα της βάσης μετά το username της και τέλος το password της. Για την περίπτωση μας η βάση ονομάζεται myapp και έχει username και password root.

Για να ολοκληρώσουμε την περιγραφή της getcoursess( ) μετά τον DriverManager βλέπουμε την δημιουργία ενός αλφαριθμητικού "**s**". Η μέθοδος CreateStatement( ) δημιουργεί το smt που είναι αντικείμενο της κλάσης Statement. Το αντικείμενο αυτό θα χρησιμοποιηθεί για το ερώτημα στη βάση δεδομένων. Το SQL ερώτημα **SELECT ctitle FROM courses** εκχωρείται στο αλφαριθμητικό "s" και στην συνέχεια το περνάμε ως όρισμα στη μέθοδο executeQuery( ) του αντικειμένου smt.

Μόλις εκτελεστεί το ερώτημα προς τη βάση myapp τα αποτελέσματα θα επιστραφούν σαν ένα αντικείμενο της ResulSet με το όνομα "**rs**". Το αντικείμενο αυτό περιέχει τα αποτελέσματα με την μορφή ενός πίνακα γραμμών και στηλών. Για να τα διαβάσουμε κάνουμε χρήση μιας μεθόδου της κλάσης ResultSet, την **next( )**. Η μέθοδος αυτή επιστρέφει **true** εάν υπάρχει επόμενη γραμμή στον πίνακα και **false** εάν δεν υπάρχει. Η συγκεκριμένη υλοποίηση της μεθόδου αναγκάζει την ύπαρξη της σχεδόν πάντα μέσα σε μία επανάληψη ούτως ώστε να διαβάσουμε όλα τα στοιχεία του πίνακα αν αυτά υπάρχουν.

Τέλος κλείνουμε την σύνδεση μας με χρήση της μεθόδου **close( )** της κλάσης Connection

3.4 Try και Catch

Όπως φαίνεται στην συνάρτηση που παρουσιάστηκε παραπάνω ο κώδικας που γράψαμε για την επικοινωνία με την βάση, δεν είναι αυθαίρετα γραμμένος αλλά περιέχεται μέσα σε μπλοκ που δημιουργούν οι λέξεις **try** και **catch**. Οι λέξεις αυτές δεν

έχουν σχέση με τις βάσεις δεδομένων αλλά αποτελούν κομμάτι της JAVA. Όμως επειδή χρησιμοποιούνται κατά κόρον σε κώδικα που έχει σχέση με τις βάσεις για τον λόγω αυτό θα τις αναφέρουμε εδώ. Πρέπει να σημειωθεί πως η λειτουργία τους μπορεί να ενσωματωθεί και σε άλλα κομμάτια κώδικα πέραν των βάσεων αν κρίνουμε ότι είναι απαραίτητο.

Όταν συμβαίνει ένα πρόβλημα στον κώδικα μας, για παράδειγμα εάν προσπαθήσουμε να συνδεθούμε σε μια βάση με λάθος στοιχεία ή όταν γράψουμε εκτός των ορίων ενός πίνακα τότε γεννάται μια εξαίρεση. Μια εξαίρεση μπορεί να υπάρξει και σε μια απροσδόκητη κατάσταση την οποία δεν έχουμε σκεφτεί άρα αυτό σημαίνει ότι δε θα αντιμετωπισθεί και αυτόματα εφόσον δεν την είχαμε υπολογίσει εξ αρχής. Με την εισαγωγή λοιπόν του try και catch δημιουργούμε ένα τρόπο αντιμετώπισης για όλες αυτές τις απροσδόκητες καταστάσεις καθώς κάθε μια θα αντιμετωπιστεί από τον κώδικα που υπάρχει μέσα στο catch.

Η φιλοσοφία είναι απλή. Ο κώδικας αποτελείται από τρία μέρη. Το τμήμα try, το τμήμα catch και το προαιρετικό τμήμα finally. Όταν σε κάποιο σημείο του κώδικα που έχουμε μέσα στο τμήμα try δημιουργηθεί μια εξαίρεση τότε το πρόγραμμα μας θα διακοπεί και θα αναζητήσει το τμήμα catch που θα την αντιμετωπίσει. Το ενδιαφέρον χαρακτηριστικό που παρέχει η εξής υλοποίηση είναι ότι η εκτέλεση του υπόλοιπου προγράμματος συνεχίζεται αν το έχουμε διαχειριστεί κατά αυτόν τον τρόπο στο τμήμα catch.

Να σημειώσουμε ότι για ένα τμήμα try έχουμε την δυνατότητα να κλείνουμε με παραπάνω από ένα τμήμα catch, εάν θέλουμε να διαχειριστούμε πολλές εξαιρέσεις. Το τελευταίο κομμάτι που θα αναφέρουμε είναι προαιρετικό και μπαίνει πάντα στο τέλος.

Ό κώδικας που περιλαμβάνεται μέσα στο finally θα εκτελεστεί οπωσδήποτε μετά τα try/catch.

Στην περίπτωση του παραδείγματος μας τυπώνουμε μηνύματα κειμένου κατά την σύνδεση μας στην βάση δεδομένων myapp αλλά και στο κλείσιμο της. Με την χρήση του finally εξασφαλίζουμε ότι η βάση θα επιχειρήσει να κλείσει ανεξάρτητα με το αν δημιουργηθεί κάποια εξαίρεση κατά την σύνδεση.

ΚΕΦΑΛΑΙΟ 4

ΠΑΡΟΥΣΙΑΣΗ ΤΟΥ ΔΙΑΔΙΚΤΥΑΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ΕΞΕΤΑΣΗΣ

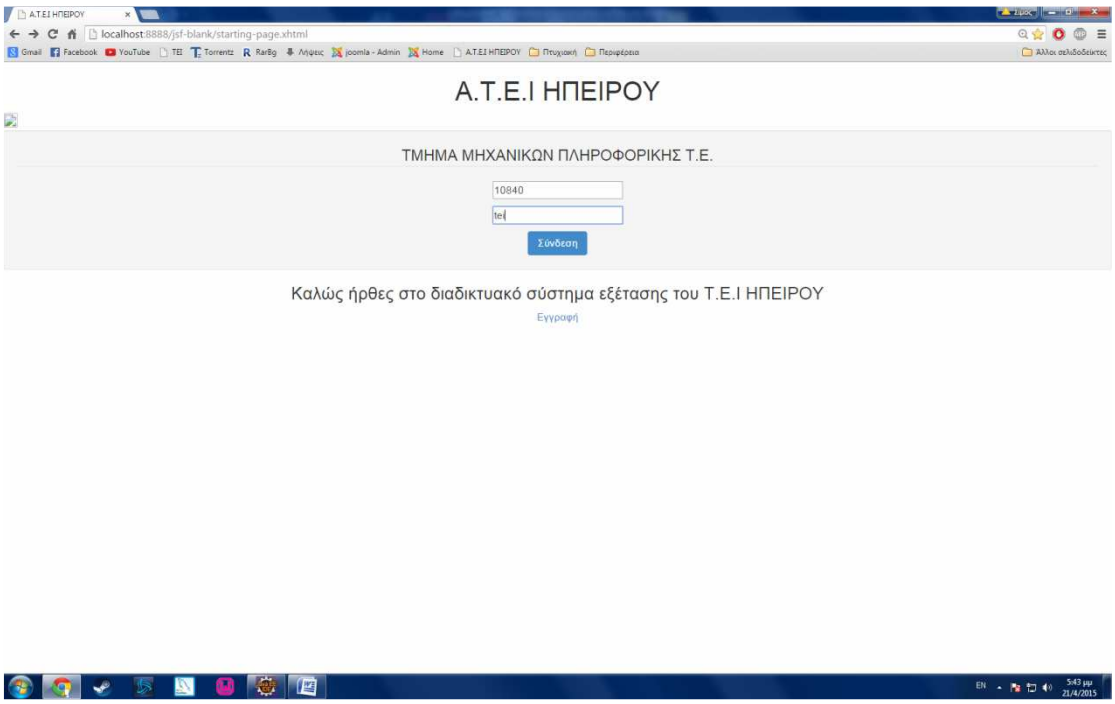
Η κατασκευή μιας διαδυκτιακής εφαρμογής προϋποθέτει μια αιτία – λόγο. Ο λόγος ύπαρξης της δικής μας διαδυκτιακής εφαρμογής είναι να βελτιώσει την εξεταστική διαδικασία σε σχέση με τους σπουδαστές και τους καθηγητές.

Για την υλοποίηση του διαδικτυακού συστήματος εξέτασης πρέπει να ορίσουμε τους αντίστοιχους χρήστες

Οι χρήστες του συστήματος είναι τρεις:

1. Οι σπουδαστές (επίπεδο 3)
2. Οι καθηγητές (επίπεδο 2)
3. Ο διαχειριστής (επίπεδο 1)

Ο κάθε ένας από αυτούς έχει την δυνατότητα εκτέλεσης διαφορετικών λειτουργιών ανάλογα με το επίπεδο τους. Με βάση αυτό το κριτήριο γίνεται εμφάνιση των αντίστοιχων οθονών για τον κάθε έναν. Ανεξάρτητα το επίπεδο του χρήστη η αρχική σελίδα παραμένει κοινή.



.Εικόνα 11 Αρχική Σελίδα

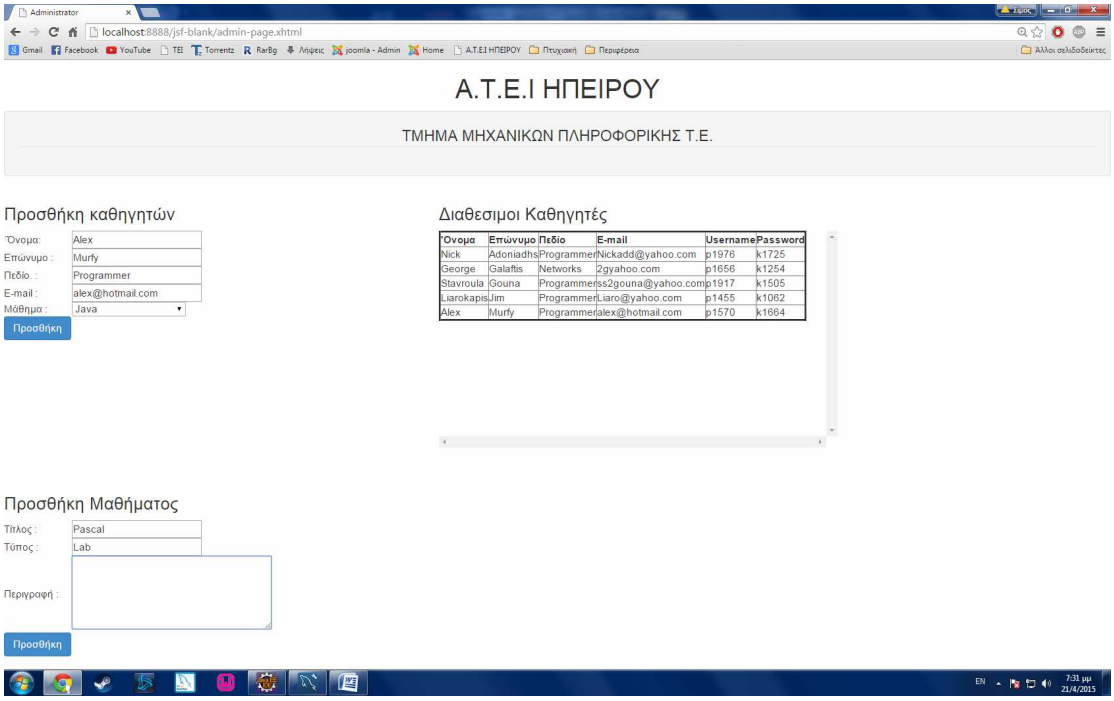
4.1 Διαχειριστής

Η λειτουργία του συστήματος ξεκινά από τον διαχειριστή ο οποίος είναι μοναδικός και έχει οριστεί εσωτερικά από τον κατασκευαστή, χωρίς την δυνατότητα αλλαγής από την εφαρμογή.

Οι ενέργειες του περιλαμβάνουν την προσθήκη μαθημάτων και καθηγητών. Κατά την προσθήκη ενός μαθήματος ανατίθεται σε αυτό ένα ID της μορφής **c-αριθμός.**

Το έργο της προσθήκης ενός καθηγητή ανατέθηκε στον διαχειριστή διότι ο προαναφερθέντας δεν μπορεί να εγγραφεί στο σύστημα μέσω της αρχικής σελίδας που εγγράφονται οι σπουδαστές λόγω ασφάλειας. Με την ολοκλήρωση της εγγραφής τους ο διαχειριστής τους παρέχει ένα ζευγάρι από στοιχεία σύνδεσης (username – password) τα οποία δημιουργεί μόνο του το σύστημα.

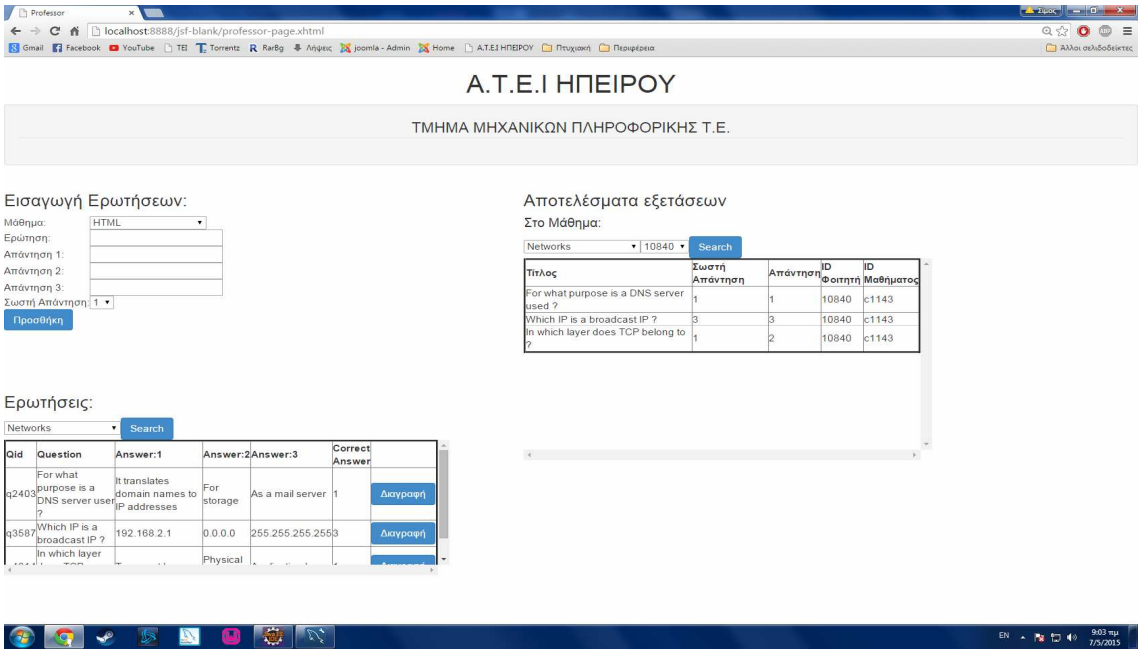
Με χρήση αυτών των στοιχείων μπορούν να συνδεθούν στο σύστημα



Εικόνα 12 Σελίδα Διαχειριστή

4.2 Καθηγητής

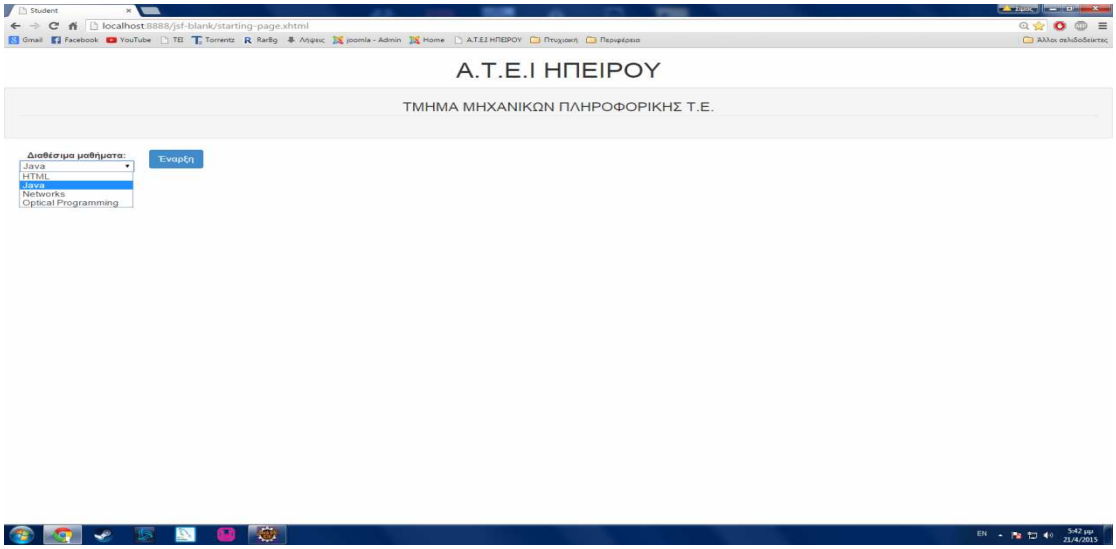
Ένας καθηγητής μπορεί να ξεκινήσει την δημιουργία ενός διαγωνίσματος. Έχει την δυνατότητα να διαγράψει ερωτήσεις σε περίπτωση κάποιου λάθους, να εισάγει νέες και να παρακολουθεί τα αποτελέσματα των διαγωνισμάτων από τις εξετάσεις των σπουδαστών



Εικόνα 13 Σελίδα Καθηγητή

4.3 Ο Σπουδαστής

Ο σπουδαστής μπορεί να εγγραφή στο σύστημα μέσω της εφαρμογής χρησιμοποιώντας την σελίδα εγγραφής. Στη συνέχεια κάνοντας χρήση του Username (που αποτελείται από τον αριθμό μητρώου) και του Password (το οποίο ορίζει ο ίδιος), έχει την δυνατότητα να συνδεθεί στο σύστημα και να επιλέξει ένα εκ των διαθέσιμων διαγωνισμάτων για να εξεταστεί.

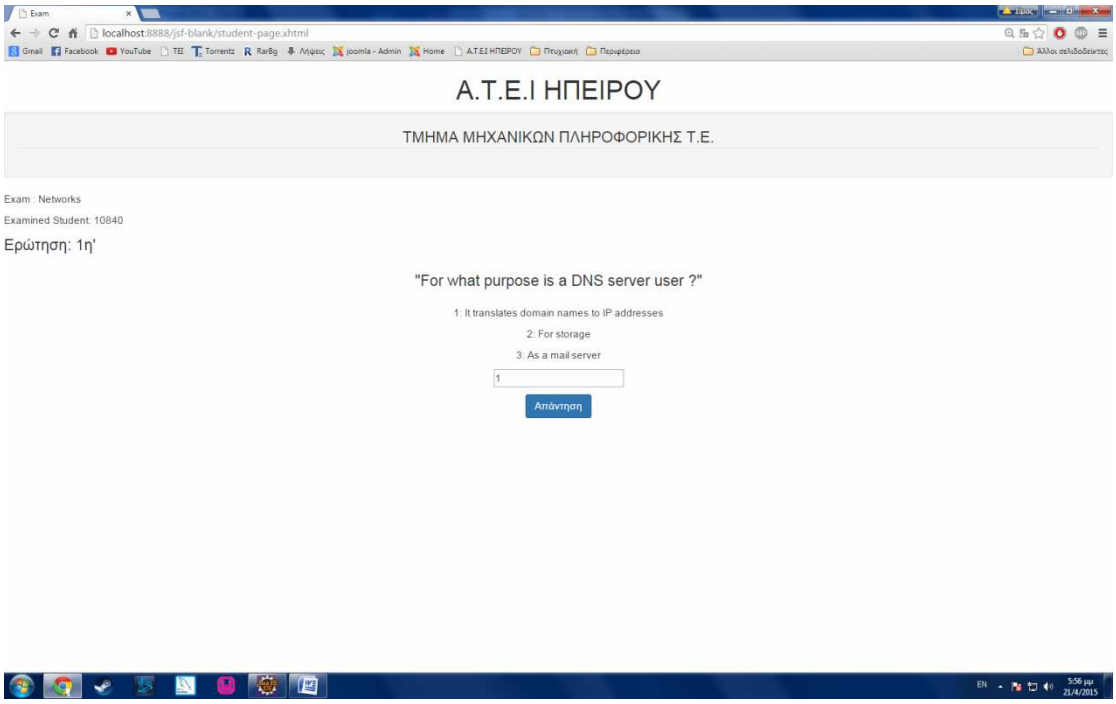


Εικόνα 14 Σελίδα Σπουδαστή

4.4 Οθόνη Εξέτασης

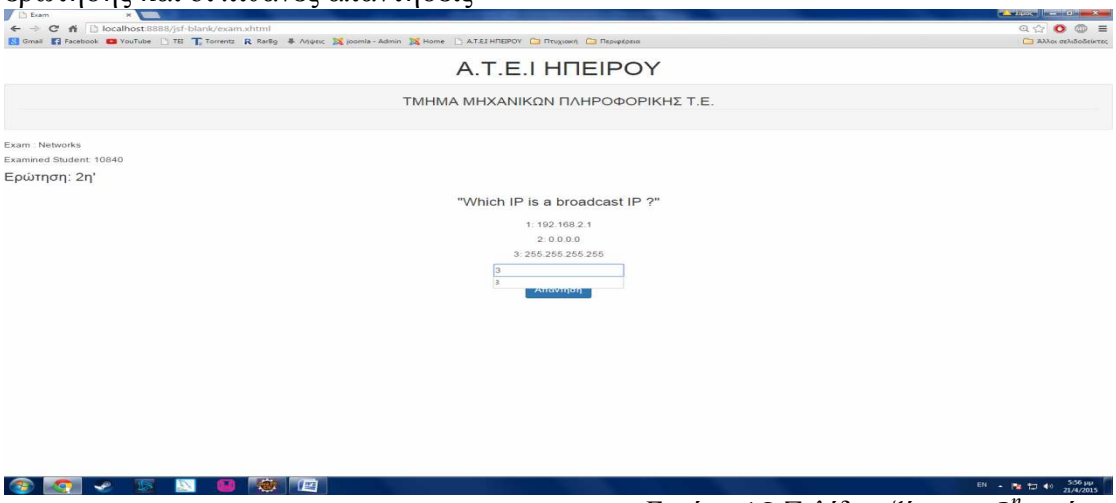
Η οθόνη της εξέτασης αποτελείται από τον τίτλο της εκάστοτε ερώτησης και τις τρεις πιθανές απαντήσεις της.

Η απάντηση του εξεταζόμενου δίνεται μέσω ενός πλαισίου κειμένου. Πληκτρολογώντας τον αριθμό της αντίστοιχης απάντησης και πατώντας το κουμπί **Απάντηση** το αποτέλεσμα φιλτράρεται από το σύστημα και αποθηκεύεται στην βάση δεδομένων.



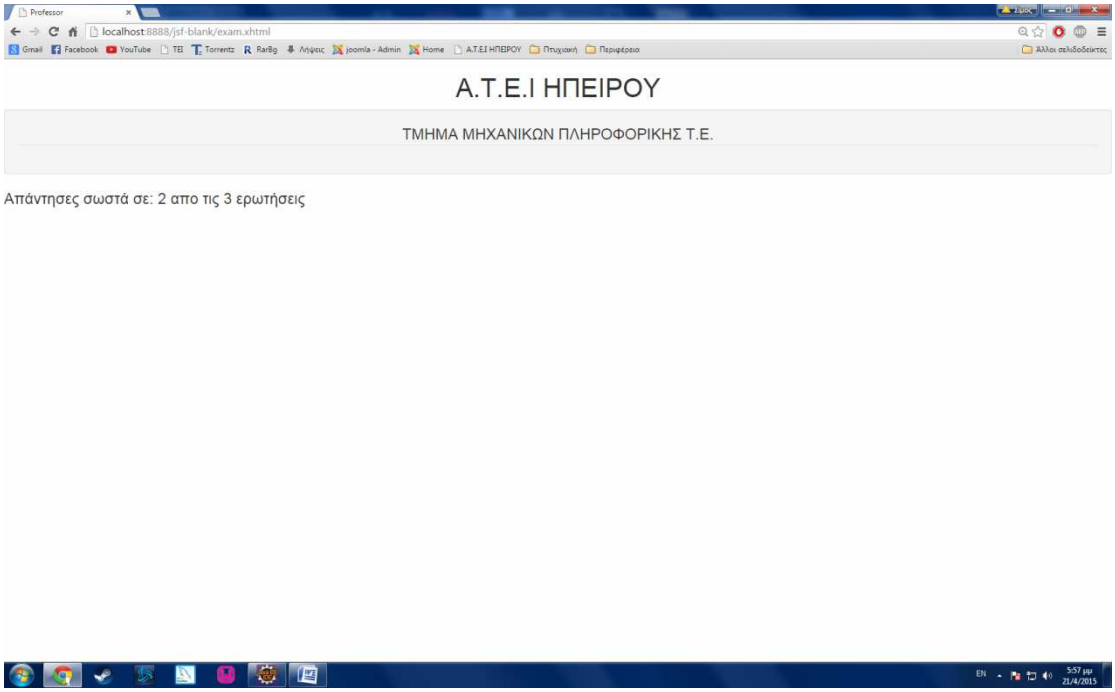
Εικόνα 15 Σελίδα εξέτασης 1η ερώτηση

Στην πορεία της εξέτασης εμφανίζονται αντίστοιχες οθόνες αλλάζοντας ο τίτλος της ερώτησης και οι πιθανές απαντήσεις



Εικόνα 16 Σελίδα εξέτασης 2η ερώτηση

Με την ολοκλήρωση της εξέτασης εμφανίζεται μία οθόνη που έχει καταγράψει τον αριθμό των σωστών απαντήσεων σε σχέση με τις συνολικές ερωτήσεις που έδωσε ο σπουδαστής. Στη συνέχεια τα αποτελέσματα της εξέτασης αυτής θα είναι διαθέσιμα για παρουσίαση σε ένα καθηγητή.



Εικόνα 17 Σελίδα Αποτελέσματος

ΚΕΦΑΛΑΙΟ 5

ΣΧΕΔΙΑΣΜΟΣ ΤΟΥ ΔΙΑΔΙΚΤΥΑΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ΕΞΕΤΑΣΗΣ

Έχοντας πραγματοποιήσει έναν αρχικό σχεδιασμό που αφορά το περιβάλλον που αλληλεπιδρά ένας χρήστης και τις λειτουργίες που θέλουμε να υλοποιήσουμε σειρά έχει να σχεδιάσουμε την εφαρμογή προγραμματιστικά. Δηλαδή να σκεφτούμε πως θα υλοποιηθεί η βάση δεδομένων, τι σελίδες θα χρειαστούμε και τι κλάσεις, μεθόδους θα περιλαμβάνει η εφαρμογή.

5.1 Βάση Δεδομένων

Το Διαδικτυακό Σύστημα Εξέτασης περιλαμβάνει πίνακες προκειμένου να αποθηκεύουμε πληροφορίες που αφορούν την εφαρμογή.

Πίνακας Users περιλαμβάνει : Username, Password, Level

Username	**Password**	**Level**
10045	simos	3

Πέραν αυτών των στοιχείων ένας χρήστης διαθέτει και άλλες πληροφορίες τις οποίες χρειάζεται να αποθηκεύσουμε σε διαφορετικούς πίνακες.

Πίνακας Students περιλαμβάνει : Sid, Sfname, Slname, Ssemester, Semail

Sid	**Sfname**	**Slname**	**Ssmester**	**Semail**
10045	Stavroula	Gounaropoulou	3	Roula12@gmail.com

Πίνακας Professors περιλαμβάνει : Pid, Pfname, Plname, Parea, Pemail

Pid	**Pfname**	**Slname**	**Parea**	**Pemail**
P1976	Alexis	Orfanos	Networks	Alex.or@gmail.com

Για τον **Διαχειριστή** δεν χρειάζεται να αποθηκεύσουμε περεταίρω πληροφορίες σε κάποιο πίνακα αρκεί να υπάρχει μια μοναδική εγγραφή επιπέδου 1(Level 1) στον πίνακα Users.

Πίνακας Courses περιλαμβάνει : Cid, Ctitle, Ctheory, Cdesscription

Cid	**Ctitle**	**Ctheory**	**Cdesscription**
C1143	Networks	Theory	TCP/IP

Πίνακας Couprof περιλαμβάνει : Cid, Pid

Cid	**Pid**

C1143	P1976

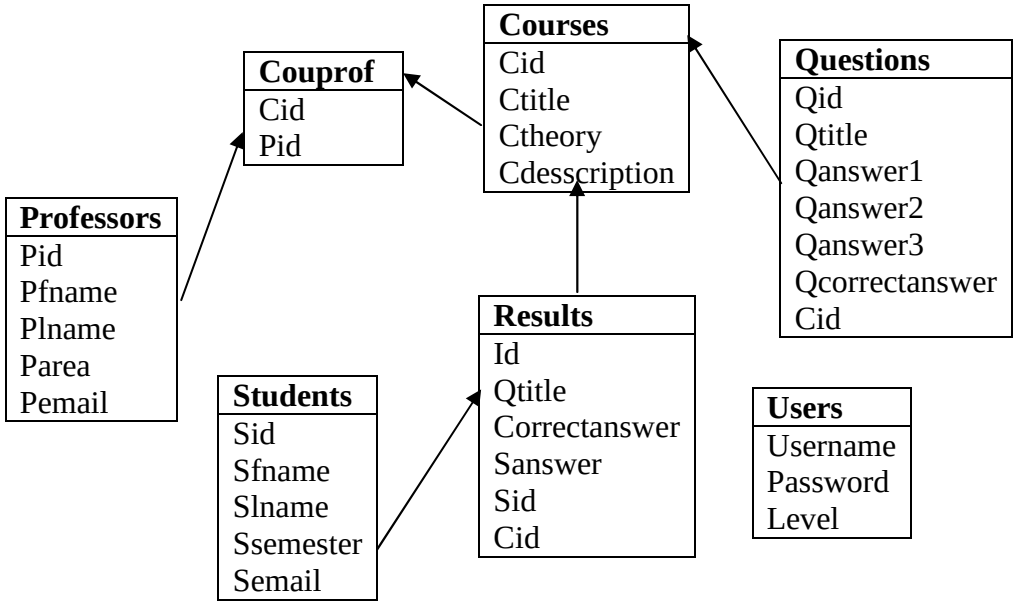
Πίνακας Questions περιλαμβάνει : Qid, Qtitle, Qanswer1, Qanswer2, Qanswer3, Qcorrectanswer, Cid

Qid	**Qtitle**	**Qanswer1**	**Qanswer2**	**Qanswer3**	**Qcorrectanswer**	**Cid**
Q2403	For what purpose is a DNS server used?	It translates domain names to IP addresses	For strorage	As a mail Server	1	C1143

Πίνακας Results περιλαμβάνει : Id, Qtitle, Correctanswer, Sanswer, Sid, Cid

Id	Qtitle	Correctanswer	Sanswer	Sid	Cid
15	For what purpose is a DNS server used?	1	3	10045	C1143

Σχεδιάγραμμα της βάσης δεδομένων.



5.2 Επίπεδο εφαρμογής.

Στο επίπεδο αυτό αναφέρομαι στις κλάσεις και μεθόδους της εφαρμογής. Το πρώτο βήμα που έχει γίνει είναι να κατασκευαστούν οι κλάσεις που αντιπροσωπεύουν τους πίνακες που υπάρχουν στην βάση δεδομένων. Αυτές μας βοηθούν να μεταφέρουμε στοιχεία από τους πίνακες στην εφαρμογή και εντέλει στις τελικές σελίδες του χρήστη. Η σχέση αυτή είναι αμφίδρομη. Κατά δεύτερον προστέθηκαν οι μέθοδοι που προσφέρουν την λειτουργικότητα της εφαρμογής.

Οι κλάσεις που ακολουθούν περιλαμβάνουν πεδία και μεθόδους και αντιπροσωπεύουν ένα αντίστοιχο πίνακα στην βάση δεδομένων.

- **Κλάση Question (*Question.java*).** Αντιπροσωπεύει τον πίνακα **Questions**. Περιλαμβάνει τις μεθόδους **addquestion( )** ,**deletequestion( )** και **isCorrect( ).** Οι δύο πρώτες αφορούν την προσθήκη και την διαγραφή των ερωτήσεων και η τελευταία ελέγχει αν είναι σωστή μια απάντηση ενός σπουδαστή.

- **Κλάση Result (*Result.java*).** Αντιπροσωπεύει τον πίνακα **Results.**

- **Κλάση Professor (*Professor.java*).** Αντιπροσωπεύει τον πίνακα **Professors.** Περιλαμβάνει την μέθοδο **addprof( )** για την εισαγωγή ενός καθηγητή.

- **Κλάση User (*User.java*).** Αντιπροσωπεύει τον πίνακα. **Students.** Περιλαμβάνει την μέθοδο **addstudent( )** για την εισαγωγή ενός σπουδαστή.

- **Κλάση Login (*Login.java*).** Αντιπροσωπεύει τον πίνακα. **Users.** Περιλαμβάνει την μέθοδο **userlogin( )** που ελέγχει την ύπαρξη ενός χρήστη στο σύστημα.

- **Κλάση Course (*Course.java*).** Αντιπροσωπεύει τον πίνακα Courses. Περιλαμβάνει την μέθοδο **addcourse( )** που εισάγει τα μαθήματα στο σύστημα.

Οι κλάσεις που ακολουθούν περιλαμβάνουν πεδία και μεθόδους χωρίς να αντιπροσωπεύουν κάποιο πίνακα στην βάση δεδομένων. Η ύπαρξη αυτών των κλάσεων προσφέρει μια περαιτέρω λειτουργικότητα.

- **Κλάση Exam (*Exam.java*).** Περιλαμβάνει τις μεθόδους **getQuestion( ), initQuestions( ) ,answerAction( )** και **nextQuestion( ).** Οι κλάσεις αυτές ελέγχουν την ροή μιας εξέτασης με το να φορτώνουν τις ερωτήσεις, να τις αλλάζουν, να αποθηκεύουν την απάντηση ενός σπουδαστή κ.τ.λ.π.

- **Κλάση Coursemenu (*Coursemenu.java*).** Περιλαμβάνει την μέθοδο **getCourses( )** η οποία φορτώνει τις απαραίτητες τιμές ενός σύνθετου πλαισίου.

- **Κλάση Proflist (*Proflist.java*).**

Περιλαμβάνει την μέθοδο **getProfessorList( )** η οποία φορτώνει τις απαραίτητες τιμές για τον πίνακα με τους διαθέσιμους καθηγητές.

5.3 Επίπεδο Παρουσίασης.

Από την στιγμή που έχει ολοκληρωθεί ο σχεδιασμός της βάσης και το επίπεδο εφαρμογής, σειρά έχει να σκεφτούμε το επίπεδο παρουσίασης. Αναφέρομαι στις σελίδες xhtml που σίγουρα θα χρειαστούμε για να δημιουργήσουμε το περιβάλλον διεπαφής του χρήστη.

- **Σελίδα υποδοχής (*starting-page.xhtml*)**.Είναι η πρώτη που εμφανίζεται στον χρήστη και σκοπό έχει να τον κατευθύνει στις υπόλοιπες σελίδες που αντιστοιχούν στους χρήστες (σπουδαστής, καθηγητής, διαχειριστής). Τα δομικά συστατικά της είναι δύο πλαίσια ένα για το *username* και ένα για το *password* τα οποία εξυπηρετούν στην εισαγωγή των στοιχείων σύνδεσης, ένα κουμπί (*Σύνδεση*) προκειμένου να συνδεθούμε και ένα link το οποίο οδηγεί σπουδαστή στην σελίδα εγγραφής.

- **Σελίδα εγγραφής (*register-page.xhtml*)**. Στην σελίδα αυτή οδηγείται ένας σπουδαστής ο οποίος δεν διαθέτει στοιχεία σύνδεσης προκειμένου να συνδεθεί στο σύστημα. Στο σημείο αυτό του ζητάτε να παρέχει ορισμένες πληροφορίες που αφορούν το προφίλ του (Αρ. Μητρώου , όνομα , επώνυμο, εξάμηνο, e-mail, username και password) σε αντίστοιχα πλαίσια. Επίσης διαθέτει ένα κουμπί (*Εγγραφή*) προκειμένου να καταχωρηθούν τα στοιχεία.

- **Σελίδα του σπουδαστή (*student-page.xhtml*)**.Στην σελίδα αυτή οδηγείται ένας εγγεγραμμένος σπουδαστής έχει δημιουργηθεί ένα σύνθετο πλαίσιο το οποίο παρέχει τα ονόματα των μαθημάτων που μπορεί να εξεταστεί. Επίσης, υπάρχει ένα κουμπί (*Έναρξη)* ώστε να ξεκινήσει η εξέταση.

- **Σελίδα του καθηγητή (*professor-page.xhtml*)**.Στην σελίδα αυτή υπάρχουν αρκετά συστατικά. Ένας πίνακας ο οποίος παρουσιάζει τις ερωτήσεις που έχουν εκχωρηθεί στο σύστημα μαζί με ένα σύνθετο πλαίσιο το οποίο εξυπηρετεί ως φίλτρο για τον ίδιο (για πιο μάθημα να εμφανίζονται οι ερωτήσεις). Η ανανέωση των στοιχείων γίνεται με το πάτημα ενός κουμπιού (*Search*). Ένας δεύτερος πίνακας για να παρουσιάζει τις απαντήσεις των σπουδαστών σε διάφορες εξετάσεις μαζί με ένα πακέτο σύνθετων πλαισίων που εξυπηρετούν ως φίλτρα (για πιο μάθημα και για ποιον σπουδαστή να εμφανίζονται τα αποτελέσματα). Η ανανέωση των στοιχείων γίνεται με το πάτημα ενός κουμπιού (*Search*).

Πλαίσια για να πραγματοποιεί την εκχώρηση ερωτήσεων ενός διαγωνίσματος. Η εισαγωγή ολοκληρώνεται με το πάτημα ενός κουμπιού (*Προσθήκη*).

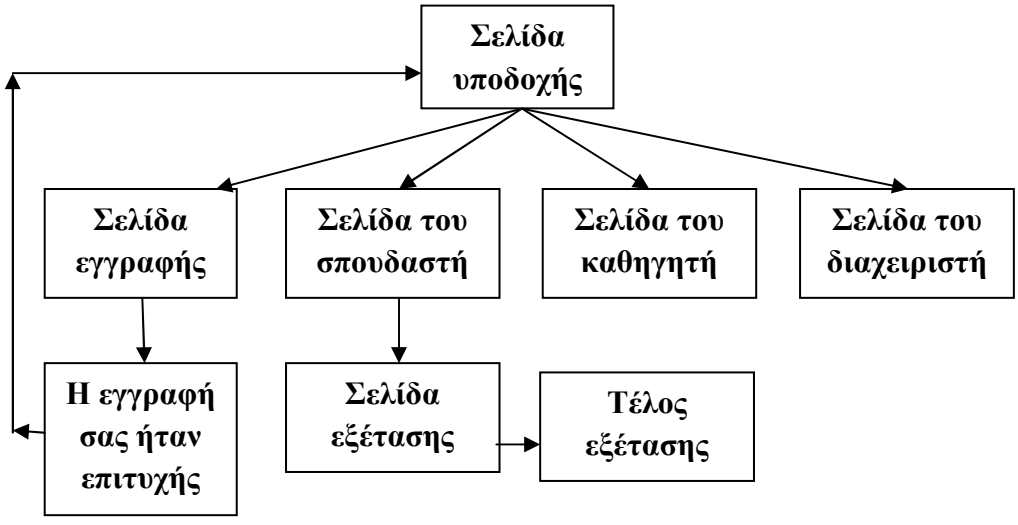
- **Σελίδα του διαχειριστή (*administrator-page.xhtml*)**.Η σελίδα αυτή διαθέτει έναν πίνακα ο οποίος παρουσιάζει τους καθηγητές που υπάρχουν στο σύστημα μαζί με το username και password που δημιουργείται κατά την εγγραφή τους. Περιέχει πλαίσια για την εισαγωγή στοιχείων που αφορούν την εγγραφή ενός καθηγητή και ενός μαθήματος. Η εισαγωγή των στοιχείων ολοκληρώνεται με το πάτημα του κουμπιού (*Προσθήκη*).

- **Σελίδα εξέτασης (*exam.xhtml*)**.Η σελίδα αυτή διαθέτει ένα πλαίσιο για να εισάγει ο σπουδαστής την απάντησή του , μία σειρά από ετικέτες οι οποίες αλλάζουν το κείμενο τους ανάλογα με την τρέχουσα ερώτηση και τις απαντήσεις της και ένα κουμπί (*Απάντηση)* για να καταχωρείται η πληροφορία στο σύστημα.

Οι σελίδες που αναφέρθηκαν είναι οι βασικές σελίδες της εφαρμογής αλλά όχι και οι μοναδικές. Έχουν δημιουργηθεί ορισμένες δεύτερης σημασίας που απλά ενημερώνουν τον χρήστη για την εκπλήρωση ορισμένων λειτουργιών.

Οι σελίδες αυτές είναι : **Η εγγραφή σας ήταν επιτυχής**
Τέλος εξέτασης.

Με την δημιουργία των σελίδων αυτών προκύπτουν ορισμένες μεταβάσεις μεταξύ τους οι οποίες έχουν υλοποιηθεί με χρήση ενός εργαλείου του **Framework** οι μεταβάσεις αυτές είναι:



ΚΕΦΑΛΑΙΟ 6

ΔΟΜΙΚΑ ΣΥΣΤΑΤΙΚΑ ΤΟΥ ΔΙΑΔΙΚΤΥΑΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ΕΞΕΤΑΣΗΣ

6.1 Εισαγωγή στις Εφαρμογές Διαδικτύου

Στο σημείο αυτό έχουμε περιγράψει τις περισσότερες βασικές έννοιες του προγραμματισμού, της JAVA και των βάσεων δεδομένων. Με γνώσεις σαν αυτές έχουμε την δυνατότητα να μεταβούμε σε μια άλλη πλευρά του προγραμματισμού, αυτή των εφαρμογών διαδικτύου. Αρχικά πρέπει να κατανοήσουμε τι είναι εφαρμογή.

Επί της ουσίας εφαρμογή είναι ένα πρόγραμμα-λογισμικό το οποίο γράφουμε σε μια γλώσσα προγραμματισμού και το εγκαθιστούμε σε έναν υπολογιστή. Το πρόγραμμα αυτό είναι σχεδιασμένο να πραγματοποιεί συγκεκριμένες λειτουργίες και στόχους ώστε να παρέχει στον χρήστη αποτελέσματα και πληροφορίες από διάφορες εργασίες. Μια εφαρμογή διαδικτύου είναι κάτι αντίστοιχο, με μόνη διαφορά ότι η εφαρμογή, δεν θα εγκατασταθεί μόνο σε έναν υπολογιστή ή και σε περισσότερους μεμονωμένα προκειμένου να εκτελεστεί. Είναι προσβάσιμη με την χρήση ενός φυλλομετρητή μέσω του διαδικτύου ή και ενός δικτύου.

Κύριος σκοπός μας είναι να δημιουργήσουμε ένα ολοκληρωμένο περιβάλλον για τον χρήστη ώστε να του παρέχουμε συγκεκριμένες πληροφορίες αλλά και να καλύψουμε την ανάγκη που μας οδήγησε στην υλοποίηση της συγκεκριμένης εφαρμογής.

6.2 JavaServer Faces

Στις μέρες μας μπορούμε να επιλέξουμε ανάμεσα σε πολλές τεχνολογίες-framewoks για να δημιουργήσουμε μια εφαρμογή διαδικτύου. Μια από αυτές είναι και η τεχνολογία JavaServer Faces (JSF) η οποία βασίζεται σε εργαλεία. Για παράδειγμα εάν θέλουμε να εμφανίσουμε έναν πίνακα δεν θα το κάνουμε δημιουργώντας το αντίστοιχο HTML μέσα σε μια επανάληψη αλλά θα εισάγουμε το αντίστοιχο εργαλείο που έχει κατασκευαστεί για αυτή την δουλειά. Με τον τρόπο αυτό εξασφαλίζουμε χρόνο και κόπο ώστε να εμπλουτίσουμε και να κάνουμε πιο προσιτό και εύχρηστο το περιβάλλον που βλέπει ο χρήστης αλλά και πιο ξεκάθαρο τον κώδικα που γράφουμε ως προγραμματιστές.

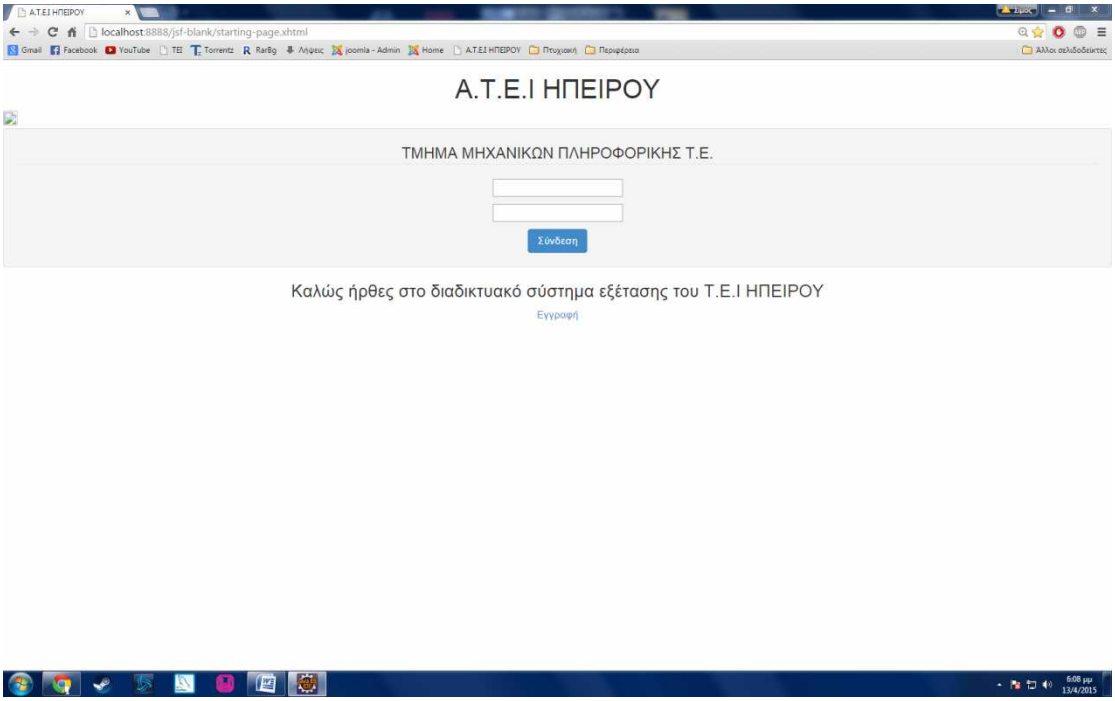
Το JSF περιλαμβάνει:

- Ένα σετ από προκατασκευασμένα εργαλεία για την προσθήκη τους σε αντίστοιχες σελίδες.
- Ένα μοντέλο λειτουργίας με βάση την εκκίνηση ενός γεγονότος.(π.χ το πάτημα ενός κουμπιού).
- Τα εργαλεία που περιλαμβάνει είναι κατασκευασμένα με τέτοιο τρόπο ώστε να μπορούμε να εισάγουμε ιδιότητες για αυτά τα εργαλεία με κώδικα τρίτων κατασκευαστών (π.χ. προσθήκη αρχείων CSS).

Κάποια εργαλεία του JSF είναι αρκετά απλά όπως τα κουμπιά και τα πλαίσια κειμένου. Άλλα πάλι είναι πολύπλοκα όπως για παράδειγμα οι πίνακες.

6.3 Μια πρώτη ματιά

Ας ρίξουμε μια ματιά σε ένα κομμάτι του Συστήματος Εξέτασης που είναι βασισμένο στην τεχνολογία JSF. Το παράδειγμα μας ξεκινά με την σελίδα υποδοχής.



Εικόνα18

Παρακάτω βλέπουμε το αρχείο που δημιουργεί την σελίδα υποδοχής, το οποίο είναι επί της ουσίας ένα αρχείο HTML με μερικές διαφορές. Οπτικά μπορεί να βελτιωθεί από κάποιον χωρίς απαραίτητα να είναι προγραμματιστής.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">

<h:head>
<title>A.T.E.I ΗΠΕΙΡΟΥ</title>

<link href="./bootstrap/css/bootstrap.css"
      rel="stylesheet" type="text/css"/>

</h:head>
<h:body>

<h1 align="center">A.T.E.I ΗΠΕΙΡΟΥ</h1>
```

```
<div id="container">
 <img src="logo.jpg"/>
 <div id="log-in" align="center">
 <h:form class="well" >
 <legend>ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε.</legend>
 <table>
      <tr>
      <td><h:inputText value="#{login.username}"/><br/><p></p></td>
      </tr>
      <tr>
       <td><h:inputText value="#{login.lpassword}"/><br/><p></p></td>
      </tr>
 </table>
      <h:commandButton class="btn btn-primary"
                  value="Σύνδεση"
                  action="#{login.userlogin}"/>

 </h:form>
</div>

<div id="welcome-message" align="center">
    <h3>Καλώς ήρθες στο διαδικτυακό σύστημα εξέτασης του Τ.Ε.Ι ΗΠΕΙΡΟΥ </h3>
    <h:outputLink value="register-page.xhtml">Εγγραφή</h:outputLink>
</div>

</div>
<script src="./bootstrap/js/bootstrap.js"></script>
</h:body>
         </html>
```

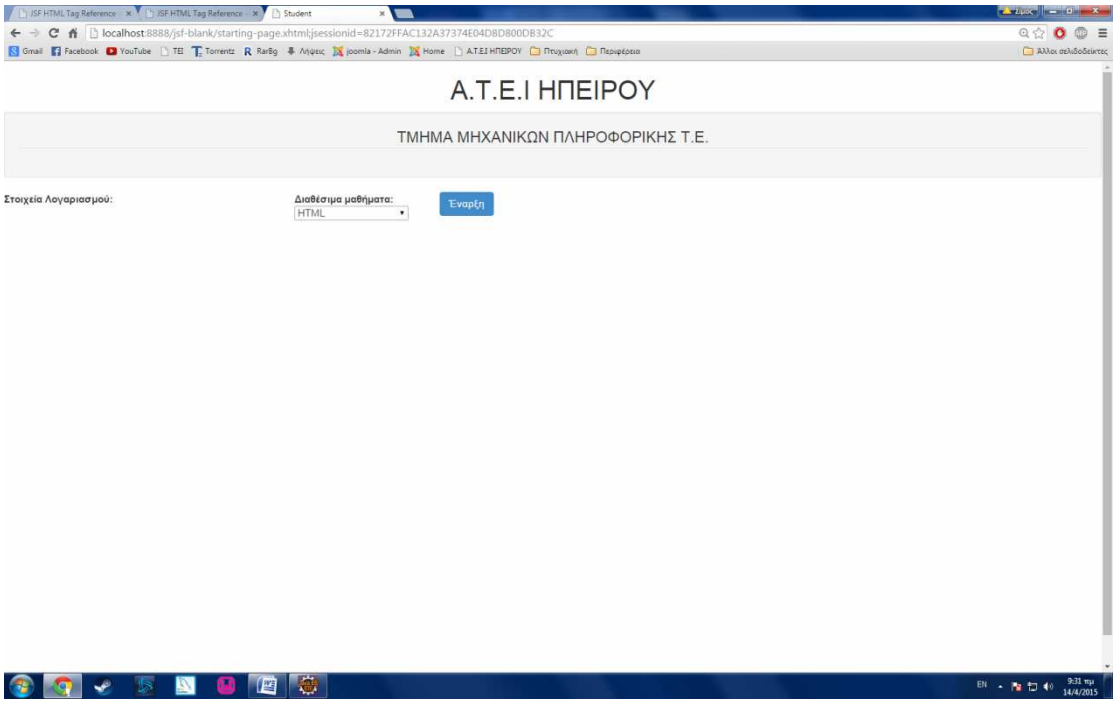
Ας περιγράψουμε με λίγα λόγια τι ακριβώς βλέπουμε μέχρι εδώ.

- Μια σειρά από ετικέτες που ανήκουν στο HTML όπως η h1 και κάποιες άλλες νέες ετικέτες όπως η h:form.

- Ορισμένες ετικέτες περιλαμβάνουν προθέματα όπως είναι η h:head και η h:inputText. Αυτές οι ετικέτες ανήκουν στο JSF. Η προσθήκη του χαρακτηριστικού XMLNS στην αρχή του αρχείου, μας εξασφαλίζει την χρήση αυτών των ετικετών.
- Το h:inputText, h:outputLink, h:commandButton είναι ετικέτες που συνδέονται με τα πλαίσια κειμένου της σελίδας υποδοχής και το τελευταίο δημιουργεί το αντίστοιχο κουμπί..

- Τα h:inputText δεν υπάρχουν αυθαίρετα μέσα στο αρχείο αλλά συνδέονται με τα πεδία μιας κλάσης. Για παράδειγμα το χαρακτηριστικό value="#{login.username}" λέει στο JSF να συνδέσει το συγκεκριμένο h:inputText με το πεδίο username που ανήκει στην κλάση login.

Όταν ο χρήστης εισάγει το username, το password και πατήσει το κουμπί “Σύνδεση”, τότε θα εμφανιστεί το αρχείο student-page.xhtml. Η σελίδα αυτή ανήκει στον σπουδαστή.
Την συγκεκριμένη λειτουργία έχουμε καθορίσει στην υλοποίηση του κουμπιού.



Εικόνα 19

Όπως βλέπουμε και στην εικόνα η δεύτερη σελίδα είναι πολύ πιο απλή από την πρώτη σελίδα που παρουσιάσαμε.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">


<h:head>
<title>Student</title>
<link href="./bootstrap/css/bootstrap.css"
      rel="stylesheet" type="text/css"/>
</h:head>
<h:body>
<h1 align="center">Α.Τ.Ε.Ι ΗΠΕΙΡΟΥ</h1>
<div id="container">
<div id="subtitle" align="center">
  <h:form class="well">
    <legend>ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε.</legend>
  </h:form>
```

```
</div>
<div id="Account" style="height:700px;width:400px;float:left;">
<b>Στοιχεία Λογαριασμού:</b>
</div>
<h:form>
<div id="Availabe" style="height:200px;width:200px;float:left;">
<b>Διαθέσιμα μαθήματα:</b><br/>
        <h:selectOneMenu value="#{exam.selectedcourse}">
      <f:selectItems value="#{exam.coursess}"/>
        </h:selectOneMenu>
</div>

<div id="startbutton" style="height:100px;width:100px;float:left;">
  <h:commandButton class="btn btn-primary" value="Εναρξη" action="exam"/>
</div>
</h:form>
</div>
</h:body></html>
```

Συστατικά τις εφαρμογής όπως έχει παρουσιαστεί μέχρι εδώ:

- Σελίδες που αντιπροσωπεύουν την σελίδα υποδοχής και την σελίδα του σπουδαστή.

- Κώδικα που διαχειρίζεται τα δεδομένα που εισάγει ο χρήστης(για την περίπτωση μας το username και το password).Ο κώδικας αυτός ονομάζεται bean και επί της ουσίας είναι μια κλάση γραμμένη σε JAVA με μεθόδους set και get.Ο κώδικας βρίσκεται σε ένα αρχείο που ονομάζεται login.java.

- Έναν τρόπο χαρτογράφησης που αφορά την πλοήγηση στην εφαρμογή.

Ακόμα και οι πιο πολύπλοκες εφαρμογές που είναι γραμμένες με την τεχνολογία JSF έχουν την ίδια υλοποίηση. Περιλαμβάνουν σίγουρα κάποιες επιπλέον κλάσεις για ελέγχους, για την επεξεργασία των δεδομένων εισόδου, για την αναπαράσταση πινάκων από βάσεις δεδομένων κ.τ.λ.π.

Το bean που ακολουθεί είναι υπεύθυνο για την διαχείριση των δεδομένων εισόδου (username, password) που εισάγει ο χρήστης στην σελίδα υποδοχής (*βλ. Εικόνα 19*) προκειμένου να συνδεθεί.

```
package com.dev.user.model;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.faces.bean.SessionScoped;
```

```
@ManagedBean(name="login")
@SessionScoped
public class login
{
        private String username;
        private String lpassword;
        private int authLevel = 0;

        public String getUsername() { return username; }
        public String getusername() { return username; }

        public void setusername(String username) { this.username = username; }

        public String getlpassword() { return username; }
        public void setlpassword(String lpassword) { this.lpassword = lpassword; }

        public int getAuthLevel() {return authLevel;}

         public String userlogin()
         {
           Connection conn = null;
           PreparedStatement ps2 = null;

           try {
                  Class.forName("com.mysql.jdbc.Driver");
                 conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/myap
                 p",    "root","root");
                 String ssql = "SELECT username,password,level FROM users WHERE
                 username = ? and password = ?";

                   ps2= conn.prepareStatement(ssql);
                   ps2.setString(1, username);
                   ps2.setString(2, lpassword);
                   ResultSet rs = ps2.executeQuery();

               int i;
               if (rs.next()) // found
               {   i = rs.getInt("level");
                   authLevel = i;
                   if (i==1)
                   return "admin";
                   else if (i==2)
                   return "prof";
                   else
                   return "student";}
               else
               {return "notlogin";}
                }
```

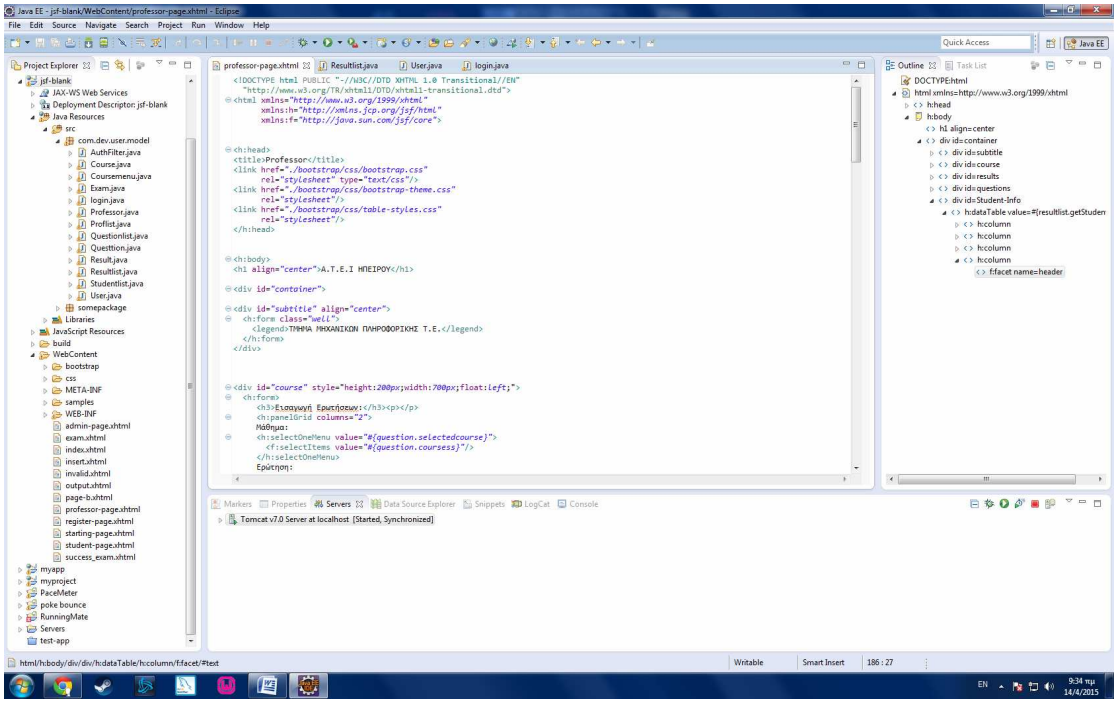
```
                catch(Exception e)
               { System.out.println("Error in login" + e.getMessage());
                return "notlogin";}

                finally
               {
                try
                {conn.close();}
                catch(Exception e)
                { e.printStackTrace(); }
            }
        }
}//class end
```

6.4 Δομή μιας εφαρμογής σε JSF

Μία εφαρμογή στο JSF έχει μια συγκεκριμένη δομή φακέλων όταν δημιουργείται. Η δομή αυτή εφαρμόζεται και από το περιβάλλον ανάπτυξης το οποίο χρησιμοποιούμε.

Τα beans, δηλαδή τα αρχεία JAVA που είναι υπεύθυνα για την συμπεριφορά της εφαρμογής είναι μέσα σε ένα πακέτο του φακέλου SRC και ότι αφορά την εμφάνιση της εφαρμογής, δηλαδή οι σελίδες XHTML είναι μέσα στον φάκελο WEBCONTENT. Στην εικόνα που ακολουθεί βλέπουμε ένα παράδειγμα εφαρμογής οργανωμένο σε φακέλους στο Eclipse IDE.



Εικόνα 20

Έχουμε ήδη εξηγήσει πως λειτουργεί ένα μέρος του κώδικα μας οπότε τώρα μπορούμε να έχουμε μια πιο διεξοδική ματιά στο πως δομείται όλος αυτός ο κώδικας. Οι εφαρμογές διαδικτύου αποτελούνται ουσιαστικά από δύο κομμάτια. Το ένα είναι το κομμάτι της παρουσίασης και άλλο είναι η επιχειρηματική λογική.

Το κομμάτι της παρουσίασης αναλαμβάνει την εμφάνιση της εφαρμογής. Το πως θα υλοποιηθεί το περιβάλλον διεπαφής του χρήστη. Το κομμάτι αυτό το οποίο τρέχει σε έναν φυλλομετρητή καθορίζει την μορφή του μέσο ετικετών του HTML με την δημιουργία κουμπιών, ετικετών κ.τ.λ.π.

Το κομμάτι της επιχειρηματικής λογικής είναι ο κώδικας που γράφουμε σε JAVA ώστε να καθορίσουμε την συμπεριφορά της εφαρμογής.

Ορισμένες τεχνολογίες που διατίθενται για την δημιουργία εφαρμογών διαδικτύου μπερδεύουν το κομμάτι της παρουσίασης με αυτό της επιχειρηματικής λογικής. Με λίγα λόγια γράφουν σε ένα αρχείο της ετικέτες του HTML με τον κώδικα JAVA. Ίσως για μια πάρα πολύ απλή εφαρμογή ο τρόπος αυτός να είναι σχετικά ευέλικτος καθώς όλος ο κώδικας θα είναι σε ένα αρχείο. Αλλά για μεγαλύτερες εφαρμογές αυτή η μίξη σίγουρα θα δημιουργήσει πολλά προβλήματα λόγω της πολυπλοκότητας.

Ένας άλλος λόγος διαχωρισμού είναι ότι το επίπεδο παρουσίασης έχει εξελιχθεί τόσο πολύ στις μέρες μας που υπάρχουν άνθρωποι που ασχολούνται μόνο με αυτό. Με το πως θα δείχνει μια εφαρμογή διαδικτύου και το κάνουν χωρίς οι ίδιοι να είναι προγραμματιστές. Σίγουρα δεν θα ήθελαν να μπλέκουν τα δικά τους εργαλεία με τα αντίστοιχα προγραμματιστικά. Από την άλλη οι προγραμματιστές έχουν να αντιμετωπίσουν μια διαφορετική πολυπλοκότητα και όσο αναφορά την μορφοποίηση μιας σελίδας οι γνώσεις τους ίσως είναι απλά βασικές.

Με απλά λόγια είναι απαραίτητο να ξεχωρίζουμε τα δύο κομμάτια που αναφέραμε στην αρχή ώστε να επιτρέψουμε και στους προγραμματιστές αλλά και στους σχεδιαστές να κάνουν ο κάθε ένας το έργο του.

Το JSF φυσικά και προσφέρει αυτόν τον διαχωρισμό και αυτό φαίνεται και μέσο της δομής που έχει η εφαρμογή στα αριστερά της εικόνας που παρουσιάστηκε στην αρχή (*βλ. Εικόνα. 20)*.

6.5 Beans

Ένα JAVA bean είναι μία κλάση που φιλοξενεί ιδιότητες και χειρίζεται διάφορα γεγονότα. Αυτή είναι η θέση του σαν κομμάτι της τεχνολογίας JSF . Μια ιδιότητα έχει ένα συγκεκριμένο όνομα και ένα τύπο επιστροφής ανάλογα με το πεδίο το οποίο συνδέεται ώστε να μας παρέχει έναν τρόπο να διαβάζουμε και να γράφουμε αυτό το πεδίο. Ο πιο εύκολος τρόπος να ορίσουμε μια ιδιότητα είναι να ακολουθήσουμε τον κλασικό τρόπο που έχουμε μάθει για τις μεθόδους ιδιωτικής πρόσβασης τα γνωστά set και get.

Στις ιδιότητες που θα ορίσουμε στο JSF το πρώτο γράμμα που ακολουθεί μετά από ένα set ή get θα πρέπει πάντα να είναι κεφαλαίο. Αυτές τις ιδιότητες χρησιμοποιεί το JSF σε αρκετά σημεία που αφορούν άλλες λειτουργίες και περιμένει να τις βρει γραμμένες με τον τρόπο που περιγράψαμε. Διαφορετικά ο κώδικας που γράφουμε δε θα λειτουργεί σωστά και ίσως δεν θα είναι εύκολο να βρούμε το λάθος.

Για παράδειγμα το bean login έχει τρεις ιδιότητες, το username το password και το authlevel.Τα δυο πρώτα είναι τύπου String και το τρίτο τύπου int.

```
public String getUsername() { return username; }
public void setUsername(String username) { this.username = username; }

public String getLpassword() { return username; }

public void setLpassword(String lpassword) { this.lpassword = lpassword; }

public int getAuthLevel() {return authLevel;}
```

Στο JSF οι μέθοδοι set και get δεν χρησιμοποιούνται μόνο για να ενθυλακώσουν ένα πεδίο μιας κλάσης. Στις περισσότερες των περιπτώσεων απλά θα αλληλεπιδρούν με ένα πεδίο κειμένου το οποίο ανήκει σε κάποια σελίδα. Σε άλλες περιπτώσεις πάλι θα τις χρειαστούμε για ορισμένους υπολογισμούς ή ακόμα και για να εξάγουμε δεδομένα από μια βάση δεδομένων.

Στο σημείο αυτό θα επεκτείνουμε την έννοια των beans με την προσθήκη του managed bean. Ένα managed bean είναι ένα JAVA bean το οποίο θέλουμε να έχει πρόσβαση σε αυτό μια σελίδα JSF. Ένα managed bean πρέπει να έχει ένα όνομα και ένα scope (Το scope θα το αναλύσουμε αργότερα). Με τον τρόπο αυτό ένα αντικείμενο αυτού του bean θα είναι διαθέσιμο για έναν χρήστη σε πολλές σελίδες JSF της εφαρμογής. Οι διαφορετικοί χρήστες που θα κάνουν χρήση της εφαρμογής, έχουν διαφορετικά ας πούμε αντικείμενα ο κάθε ένας. Όπως γίνεται δηλαδή με βάση την αντικειμενοστραφή υλοποίηση.

Σε ένα bean έχουμε πρόσβαση με την εξής έννοια. Όταν το όνομα ενός bean εμφανιστεί σε μία σελίδα JSF τότε το ίδιο το JSF θα εντοπίσει αυτό το αντικείμενο με το συγκεκριμένο όνομα ή αν δε το βρει θα το δημιουργήσει και θα το βάλει μέσα σε ένα scope. Στη συνέχεια για παράδειγμα, εάν συνδεθεί ένας νέος χρήστης στην εφαρμογή μας τότε θα δημιουργηθεί ένα νέο αντικείμενο της κλάσης login. Ο ποιο εύκολος τρόπος για να υλοποιήσουμε ένα τέτοιο managed bean είναι αυτός που ακολουθεί.

Σε σχέση με το bean:

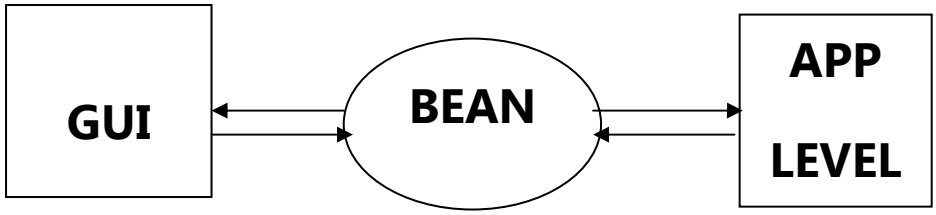
```
@ManagedBean(name="login")  ή  @Named(“login”)
@SessionScoped
public class login
        { . . .}
```

Σε σχέση με την σελίδα:

```
<h:inputText value="#{login.username}"/>
```

Στις εφαρμογές που είναι βασισμένες στο JSF χρησιμοποιούμε πάντα τα managed beans για όλα τα δεδομένα τα οποία πρέπει να είναι προσβάσιμα από μία σελίδα.

Τα beans είναι επί της ουσίας ο ενδιάμεσος ανάμεσα στο περιβάλλον του χρήστη και της εφαρμογής.



Εικόνα 21

6.6 Σελίδες JSF

Για κάθε διαφορετική σελίδα που θέλουμε να παρουσιάσουμε σε έναν χρήστη πρέπει να δημιουργήσουμε αντίστοιχες σελίδες JSF. Η διαδικασία αυτή ονομάζεται templating. Για ιστορικούς λόγους να αναφέρουμε ότι υπάρχουν αρκετοί μηχανισμοί για την δημιουργία σελίδων JSF. Το JSF v.1 χρησιμοποιούσε σελίδες JSP για αυτή τη διαδικασία.

Όμως, επειδή το JSP είναι ουσιαστικά μια άλλη αυτόνομη τεχνολογία, η συνύπαρξη τους δεν ήταν πολύ επιτυχής τελικά καθώς δημιουργούσε πολλά προβλήματα. Για τον λόγο αυτό στο JSF v.2 το JSP δεν αποτελεί πλέον επιλογή, καθώς ο διάδοχος τους τα Facelets, είναι πολύ πιο εύχρηστα από την πλευρά της κατασκευής, της προσθήκης εργαλείων από τρίτους και γενικά του όλου μηχανισμού τους.

Όταν δημιουργούμε ένα Facelet προσθέτουμε ετικέτες που ανήκουν στο JSF σε μία σελίδα XHTML. Μία σελίδα XHTML είναι στην ουσία μια HTML σελίδα σε συνδυασμό με XML. Για τα Facelets χρησιμοποιούμε την κατάληξη .xhtml. Αν ρίξουμε μια ματιά στις σελίδες που έχουμε παρουσιάσει ως τώρα θα παρατηρήσουμε ότι στην αρχή τους υπάρχει μια δήλωση.

```
<html   xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html">
```

Η δεύτερη γραμμή είναι αυτή που εισάγει το πρόθεμα h για τις JSF - HTML ετικέτες. Το JSF σαν τεχνολογία παρέχει από μόνο του και ορισμένες ετικέτες που δεν ανήκουν στο HTML. Αν έχουμε ανάγκη από μια τέτοια ετικέτα στην σελίδα μας θα πρέπει να προσθέσουμε ακόμα μια δήλωση.

```
xmlns:f=http://java,sun.com/jsf/core
```

Αν θέλουμε να κάνουμε χρήση ετικετών από άλλους παρόχους τότε θα πρέπει να κάνουμε αντίστοιχες δηλώσεις για να τις χρησιμοποιήσουμε. Όσο αναφορά την υλοποίηση μιας σελίδας JSF είναι παρόμοια με αυτή του HTML. Οι βασικές τους διαφορές είναι η εξής:

- Η σελίδα XHTML που κατασκευάζουμε πέρα από την μορφοποίηση που παρέχει δεν αναλαμβάνει μόνο να παρουσιάσει δεδομένα όπως κάνει το HTML αλλά αναλαμβάνει να τα μεταφέρει κιόλας. Δεν αποτελεί σε καμία περίπτωση αντικαταστάτη για το HTML.

- Κάνουμε χρήση των ετικετών h:head, h:body και h:form αντί των head, body και form.

- Αντί να χρησιμοποιήσουμε τις ετικέτες εισόδου HTML κάνουμε χρήση των h:inputText και h:commandButton.

Στην ουσία με τις ετικέτες που μας παρέχει το JSF σαν τεχνολογία γράφουμε κλασσικό HTML απλά είμαστε ένα επίπεδο παραπάνω. Αυτό συμβαίνει γιατί μια ετικέτα του JSF αντιπροσωπεύει ένα αρκετά πιο μεγάλο κομμάτι HTML.Για παράδειγμα.

Για το HTML στην ετικέτα form γράφαμε :

<form id="XXX" name="XXX" method="POST" action="..."
. . .
</form>

Βλέπουμε ότι επιλέγαμε και την μέθοδο στο χαρακτηριστικό method..(POST, GET)

Ενώ στο JSF γράφουμε:

<h:form>
. . .
</h:form>

Η μέθοδος χρειάζεται και για τις δύο περιπτώσεις μόνο που στην δεύτερη περίπτωση εισάγετε από μόνη της γιατί εμπεριέχετε μέσα στην ετικέτα h:form.

6.7 Το αρχείο web.xml

Προκειμένου να εκτελέσουμε μια εφαρμογή διαδικτύου πρέπει να αναθέσουμε αυτό το έργο σε έναν εξυπηρετητή. Στην περίπτωση μας είναι ο Tomcat. Για να ολοκληρώσουμε αυτή την ενέργεια εισάγουμε στην εφαρμογή μας ένα αρχείο με το όνομα web.xml. Εάν θέλουμε, η λειτουργία που παρέχει αυτό το αρχείο μπορεί να εφαρμοστεί σε όλες τις εφαρμογές που δημιουργούμε με βάση το JSF καθώς απευθύνεται σε γενικές ρυθμίσεις και όχι σε θέματα υλοποίησης.

Εδώ παρουσιάζουμε ένα αρχείο web.xml .

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
<display-name>jsfJdbcInsert</display-name>
<welcome-file-list>
```

```
<welcome-file>faces/starting-page.xhtml</welcome-file>
</welcome-file-list>
<servlet>
<servlet-name>Faces Servlet</servlet-name>
<servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
<servlet-name>Faces Servlet</servlet-name>
<url-pattern>/faces/*</url-pattern>
<url-pattern>*.jsf</url-pattern>
<url-pattern>*.xhtml</url-pattern>
</servlet-mapping>
<context-param>
<description>State saving method: 'client' or 'server' (=default). See JSF Specification 2.5.2</description>
<param-name>javax.faces.STATE_SAVING_METHOD</param-name>
<param-value>client</param-value>
</context-param>
<context-param>
<param-name>javax.servlet.jsp.jstl.fmt.localizationContext</param-name>
<param-value>resources.application</param-value>
</context-param>
<listener>
<listener-class>com.sun.faces.config.ConfigureListener</listener-class>
</listener>
<servlet-mapping>
<servlet-name>Faces Servlet</servlet-name>
<url-pattern>*.faces</url-pattern>
</servlet-mapping>
</web-app>
```

Όλες οι σελίδες JSF που δημιουργούμε μπορούν να ενταχθούν σε ένα τέτοιο αρχείο αρκεί να είναι σωστά διαμορφωμένες XHTML σελίδες .Οι ρυθμίσεις που εισάγουμε εδώ έχουν σχέση με το URL το οποίο γράφουμε στον φυλλομετρητή μας καθώς θέλουμε να εξασφαλίσουμε ότι θα φορτώσουμε την σωστή σελίδα. Τα URL στο JSF έχουν μια συγκεκριμένη μορφή και περιλαμβάνουν το /faces. Το servlet-mapping που γράφουμε στο συγκεκριμένο αρχείο παραμετροποιεί το URL μας.

Για παράδειγμα δεν μπορούμε να γράψουμε απλά http://localhost:8080/myapp/starting-page.jsf για να τρέξουμε την σελίδα starting-page.xhtml αλλά **http://localhost:8080/myapp/faces/starting-page.jsf** για να μην εμφανίζεται το /faces/ στο URL εισάγουμε μέσα στο servlet-mapping την γραμμή <url-pattern>/faces/*</url-pattern> ώστε να το αφαιρέσει.

Τέλος το web.xml ορίζει και μια αρχική σελίδα μέσω του χαρακτηριστικού <welcome-file> </welcome-file>.Στην περίπτωση μας έχουμε ορίσει σαν αρχική σελίδα την starting-page.xhtml.

```
<welcome-file-list>
        <welcome-file>faces/starting-page.xhtml</welcome-file>
</welcome-file-list>
```

Εάν γράψουμε το URL http:localhost:8080/myapp τότε θα μας εμφανιστεί η σελίδα starting-page αυτόματα.

6.8 Το αρχείο faces-config.xml

Το αρχείο αυτό είναι επίσης ένα κομμάτι της τεχνολογίας JSF το οποίο παρέχει αρκετές λειτουργίες όπως το να εκτελέσει ορισμένους ελέγχους και να ορίσει την πλοήγηση ανάμεσα στις διαφορετικές σελίδες μιας εφαρμογής. Θα αναλύσουμε ένα παράδειγμα πλοήγησης για να καταλάβουμε ακριβώς την χρησιμότητα του αλλά και την λειτουργία του.

Ο τρόπος με τον οποίο ελέγχει την πλοήγηση το συγκεκριμένο αρχείο είναι μέσω της δημιουργίας διάφορων κανόνων και σεναρίων. Στα σενάρια αυτά, δεχόμαστε μια είσοδο και ανάλογα με την τιμή της κατευθυνόμαστε σε μία νέα σελίδα. Η βασική μορφή ενός σεναρίου είναι η ακόλουθη:

```
<navigation-case>
<from-action          </from-action>
<from-outcome>        </from-outcome>
<to-view-id>          </to-view-id>
</navigation-case>
```

- Αρχικά περικλείουμε όλα τα υπόλοιπα στοιχεία που δομούν το σενάριο μας μέσα σε στο block <navigation-case> </navigation-case>

- Στην συνέχεια εισάγουμε το <from-action> </from-action>. Μπορούμε να καθορίσουμε από ποιο bean και από ποια μέθοδος του περιμένουμε την έξοδο.

- Στο <from-outcome> </form-outcome> εισάγουμε τις διαφορετικές τιμές που μπορεί να επιστρέψει η αντίστοιχη μέθοδος που ορίσαμε στην αρχή και ανάλογα με αυτές τις τιμές θέλουμε να περιηγηθούμε. Ανάλογα με το πόσες είναι αυτές οι διαφορετικές τιμές τόσα σενάρια θα πρέπει να δημιουργήσουμε.

- Τελικά ανάμεσα στο μπλοκ <to-view-id> </to-view-id> εισάγουμε το όνομα της σελίδα που θέλουμε να περιηγηθούμε.

Στη συνέχεια παρουσιάζεται ένα μέρος του αρχείου faces.config.xml με σενάρια που υλοποιήσαμε για το σύστημα εξέτασης..

```
<?xml version="1.0" encoding="UTF-8"?>

<faces-config
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-facesconfig_2_0.xsd"
version="2.0">

<managed-bean>
<managed-bean-name>user</managed-bean-name>
<managed-bean-class>com.dev.user.model.User</managed-bean-class>
<managed-bean-scope>request</managed-bean-scope>
</managed-bean>

• • •
<navigation-case>
<from-action>#{login.userlogin}</from-action>
<from-outcome>student</from-outcome>
<to-view-id>/student-page.xhtml</to-view-id>
</navigation-case>

<navigation-case>
<from-action>#{login.userlogin}</from-action>
<from-outcome>prof</from-outcome>
<to-view-id>/professor-page.xhtml</to-view-id>
</navigation-case>
<navigation-case>
<from-action>#{login.userlogin}</from-action>
<from-outcome>admin</from-outcome>
<to-view-id>/admin-page.xhtml</to-view-id>
</navigation-case>

<navigation-case>
<from-action>#{login.userlogin}</from-action>
<from-outcome>notlogin</from-outcome>
<to-view-id>/invalid.xhtml</to-view-id>
</navigation-case>
</navigation-rule>

 • • •
</faces-config>
```

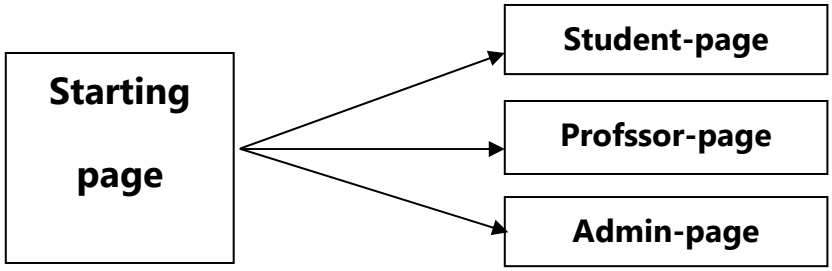
Ας έρθουμε σιγά - σιγά στο παράδειγμα μας για να δούμε την χρησιμότητα αυτού του αρχείου και των κανόνων που το δομούν. Όταν παρουσιάσαμε σε προηγούμενη ενότητα την αρχική σελίδα starting-page.xhtml, είδαμε πως όταν ο χρήστης ήθελε να συνδεθεί πληκτρολογούσε τα στοιχεία του και πατούσε το κουμπί Σύνδεση ώστε να βρεθεί στην σελίδα student-page.xhtml. Θα μπορούσε κανείς να σκεφτεί πως η πλοήγηση γίνεται μέσω του κουμπιού σύνδεση περιλαμβάνοντας στην ιδιότητα action του κουμπιού το όνομα της σελίδα student-page.xhtml.

```
<h:commandButton class="btn btn-primary"
                 value="Σύνδεση"
                 action="student-page.xhtml"/>
```

Αυτό είναι σχεδόν σωστό γιατί :

- Πρώτον, εάν είναι να βρεθείς έτσι απλά στην σελίδα student-page.xhtml τι νόημα έχει να ζητάς στοιχεία εισόδου. Αφού με το πάτημα του κουμπιού μπορείς να πλοηγηθείς απευθείας στη σελίδα. Πρέπει να μεσολαβήσουν κάποιοι έλεγχοι αρχικά.

- Δεύτερον, στην εφαρμογή μας υπάρχουν και άλλες σελίδες πέραν της student-page. Εάν στην ιδιότητα action εισάγουμε απευθείας το όνομα μιας συγκεκριμένης σελίδας τότε πως θα μεταβούμε στις υπόλοιπες σελίδες;



Εικόνα 22

Τελικά η πλοήγηση δεν γίνεται απευθείας από το κουμπί σύνδεση αλλά με έναν διαφορετικό τρόπο. Μέσω του αρχείου faces-config.xml. Προσοχή στο ότι αυτό δεν αναιρεί την λειτουργικότητα της ιδιότητας action που υπάρχει στο κουμπί. Ας δούμε την αρχή λοιπόν. Εάν και η πλοήγηση δεν γίνεται απευθείας από το κουμπί η όλη διαδικασία ξεκινά από εκεί.

Στην ιδιότητα action του κουμπιού γράφουμε:

```
<h:commandButton  class="btn btn-primary"
                  value="Σύνδεση"
                  action="#{login.userlogin}"/>
```

Αυτό έχει ως συνέπεια να καλέσουμε μια μέθοδο που ονομάζεται userlogin και ανήκει στην κλάση login. Στην μέθοδο αυτή πραγματοποιούνται οι έλεγχοι που είπαμε αρχικά.

- Ελέγχει εάν ο χρήστης αυτός υπάρχει στην βάση δεδομένων.

- Αν υπάρχει, ανάλογα με το τι χρήστης είναι επιστρέφει μία τιμή. Στην εφαρμογή μας κάθε χρήστης ανάλογα με το επίπεδο του έχει και έναν αριθμό που τον αντιπροσωπεύει. Διαφορετικό επίπεδο άρα και διαφορετικός χρήστης.
  Αν δεν υπάρχει τότε επιστρέφει διαφορετική τιμή

```
if (rs.next( )) // found
{
        i = rs.getInt("level");
        authLevel = i;
```

```
                    if (i==1)
                        return "admin";
                    else if (i==2)
                        return "prof";
                    else
                        return "student";
                }
               else
                     { return "notlogin"; }
```

Στο κομμάτι της μεθόδου userlogin βλέπουμε τις διαφορετικές τιμές που επιστρέφει.

- Αν η μέθοδος rs.next( ) επιστρέψει true τότε ο χρήστης υπάρχει.
- Εφόσον υπάρχει αντλούμε από τον αντίστοιχο πίνακα που έχουμε στην βάση δεδομένων το επίπεδο του και ανάλογα με αυτό η μέθοδος επιστρέφει τις τιμές **admin, prof , student**
- Αν η μέθοδος rs.next( ) επιστρέψει false τότε ο χρήστης δεν υπάρχει και η μέθοδος επιστρέφει **notlogin.**

Με βάση αυτές τις τιμές λοιπόν διαμορφώνουμε και ένα διαφορετικό σενάριο στο αρχείο faces- config.xml

```
<navigation-case>
<from-action>#{login.userlogin}</from-action>
<from-outcome>student</from-outcome>
<to-view-id>/student-page.xhtml</to-view-id>
</navigation-case>
```

Αν η τιμή που επιστρέφει η μέθοδος είναι student τότε μεταβαίνουμε στην student-page.xhtml. Αντίστοιχα λειτουργούν και οι υπόλοιποι κανόνες για τις τιμές που επιστρέφει η userlogin.

6.9 Τα Session και Request Scopes

Μπορούμε να φανταστούμε τα Scope σαν κάποιον που κρατά δεδομένα για μας. Στην περίπτωση μιας εφαρμογής διαδικτύου τα δεδομένα αυτά μπορεί να είναι πληροφορίες από διαφορετικά beans τις οποίες θέλουμε να παρέχουμε σε άλλα beans ή σελίδες που περιλαμβάνουν ορισμένα εργαλεία και χρειάζονται τις πληροφορίες αυτές. Προς διευκόλυνση μας η τεχνολογία JSF παρέχει για τις εφαρμογές μας αρκετά Scope το κάθε ένα με διαφορετικές λειτουργίες. Είχαμε αναφέρει στο κεφάλαιο των beans ότι με την δημιουργία ενός bean πρέπει να εισάγουμε στην αρχή του και ένα Scope.

Τα Scope που έχουμε στην διάθεση μας είναι το:

- Session Scope

- Request Scope
- Application Scope

Στην τεχνολογία JSF v.2 έχουμε την δυνατότητα να κάνουμε χρήση των annotations, για ακόμα μεγαλύτερη ευελιξία. Μπορούμε δηλαδή να γράψουμε στην αρχή ενός bean:

- @SessionScoped
- @RequestScoped
- @ApplicationScoped

Στο σημείο αυτό να αναφέρουμε πως τα annotations αυτά είναι μέρος του πακέτου javax.faces.bean και περιλαμβάνεται στην JAVA EE.Ας δούμε αναλυτικά τα δύο πρώτα scope ξεκινώντας από το Session Scope και καταλήγοντας στο Request Scope.

6.9.1 Session Scope

Οι περισσότεροι από μας γνωρίζουμε το πρωτόκολλο HTTP και ξέρουμε ότι είναι ένα πρωτόκολλο μεταφοράς. Ένας φυλλομετρητής στέλνει ένα αίτημα σε έναν εξυπηρετητή, αυτός αντίστοιχα στέλνει μια απάντηση και κανένας εκ των δύο δεν έχει την πρόθεση να κρατήσει τις πληροφορίες που αφορούν αυτή την συναλλαγή. Αυτή η υλοποίηση δουλεύει αρκετά καλά για ανταλλαγή βασικών πληροφοριών αλλά σε ορισμένες περιπτώσεις απλά δεν επαρκή.

Το πιο κλασικό παράδειγμα είναι αυτό με το καλάθι αγοράς. Όταν προσθέτουμε προϊόντα στο καλάθι θέλουμε αυτό να τα θυμάται. Όταν προσθέσουμε προϊόντα σαν επισκέπτης και συνδεθούμε σαν χρήστης θέλουμε πάλι τα προϊόντα μας να είναι στο καλάθι .Όλες αυτές οι πληροφορίες λοιπόν διατηρούνται μέσα σε ένα Session μια σύνοδο. Για να κατανοήσουμε καλύτερα την συγκεκριμένη υλοποίηση ας δούμε ένα παράδειγμα.

Όταν είχαμε περιγράψει το bean με το όνομα login είδαμε πως περιλάμβανε δύο πεδία στο εσωτερικό του. Είχαμε πει επίσης ότι όταν ένας χρήστης συνδεθεί θα δημιουργηθεί ένα νέο αντικείμενο της κλάσης αυτής. Το αντικείμενο αυτό θα υπάρχει μέσα σε ένα scope και συγκεκριμένα Session Scope λόγω του @SessionScoped που έχουμε εισάγει στην αρχή της κλάσης. Αυτό μας παρέχει αρκετές ευκολίες κυρίως όταν θέλουμε να κρατηθούν αυτές οι πληροφορίες ενός bean για να τις εισάγουμε σε ένα άλλο bean.

```
@ManagedBean(name="login")
@SessionScoped
public class login
{
        private String username;
        private String lpassword;
        . . . }
```

Εισάγοντας το annotation @ManagedProperty(**value="#{login.username}")** πάνω από το πεδίο μιας άλλης κλάσης τότε έχουμε μεταφέρει την τιμή ενός πεδίου της κλάσης

login που έχει διατηρηθεί σε ένα Session στο πεδίο αυτής της κλάσης. Στην περίπτωση μας είναι το πεδίο examinedStudent της κλάσης Exam.

```
@ManagedBean(name="login")
@SessionScoped
public class Exam implements Serializable
{
 @ManagedProperty(value="#{login.username}")
 private String examinedStudent;

 public String getExaminedStudent()
      { return examinedStudent; }


 public void setExaminedStudent(String examinedStudent)
      { this.examinedStudent=examinedStudent; }
```

Πρέπει να δοθεί ιδιαίτερη προσοχή όταν εκτελούμε τέτοιες εργασίες όπως για παράδειγμα:

1. Και οι δύο κλάσεις να περιλαμβάνουν το @SessionScoped στην αρχή τους.
2. Το πεδίο μιας κλάσης που θέλουμε να μεταφέρουμε πρέπει να διαθέτει τουλάχιστον την μέθοδο get με το πρώτο γράμμα μετά αυτό να είναι κεφαλαίο (αλλιώς δεν λειτουργεί).
3. Ένα bean που υπάρχει μέσα σε ένα scope στην μνήμη μετά από ένα ορισμένο χρονικό διάστημα καταστρέφετε. Άρα ότι πληροφορίες περιλάμβανε χάνονται.

6.9.2 Request Scope

Το Request Scope είναι παρόμοιο με το Session με την διαφορά ότι η διάρκεια ζωής του είναι κατά πολύ μικρότερη. Ξεκινά όταν υπάρξει ένα αίτημα HTTP για κάποιον εξυπηρετητή και μόλις έρθει η απάντηση αυτό καταστρέφετε. Αξίζει να κάνουμε χρήση του συγκεκριμένου scope όταν έχουμε πρόβλημα με τον χώρο που διαθέτουμε για να αποθηκεύουμε διάφορες πληροφορίες όπως κάναμε με το Session Scope. Παρόλα αυτά μπορούμε να σκεφτούμε ότι δεν θα θέλουμε να αποθηκεύουμε πάντα κάτι από ένα bean. Αυτά τα bean μπορούν να έχουν το @RequestedScope.

6.10 Ετικέτες επιλογής : h:selectOneMenu

Είχαμε αναφέρει και σε προηγούμενες ενότητες το JSF είναι μια τεχνολογία που βασίζεται στα εργαλεία. Αποτελούν βασικά συστατικά μιας σελίδας JSF για όλες τις εφαρμογές. Η βασική ιδέα πίσω από αυτά είναι φυσικά να υπάρχουν και να καλύπτουν ορισμένες λειτουργίες χωρίς να μας ενδιαφέρει πως το κάνουν. Μας ενδιαφέρει να τα παραμετροποιήσουμε και να επεκτείνουμε την λειτουργία τους στο σύνολο μιας μεγαλύτερης λειτουργίας.

Έχουμε ήδη δει κάποια εργαλεία όπως το h:commandbutton και το h:outputlink τα οποία είναι από τα πιο απλά. Το πρώτο και πιο συνηθισμένο εργαλείο που θα

συναντήσει κανείς μετά από αυτά είναι το h:selectOneMenu το οποίο έχει λίγο μεγαλύτερη πολυπλοκότητα από τα προηγούμενα.
Το εργαλείο αυτό αποτελεί μια ετικέτα που παρέχει το JSF και ξέρουμε ότι στην ουσία δημιουργεί από πίσω HTML.

Ετικέτα	**HTML**	
h:selectOneMenu	<select size=”1”> <option value=News=> News Music . . . </option> </select>	

Ένα μέρος του ονόματος του περιλαμβάνει τις λέξεις selectOne από εκεί καταλάβουμε ότι μας επιτρέπει να διαλέξουμε μόνο μια τιμή από το αναδυόμενο Menu που δημιουργείται. Γενικά αυτό το κομμάτι υπάρχει και σε άλλες ετικέτες όπως είναι η h:selectOneRadio και η h:selectOneListbox και εννοεί ακριβώς το ίδιο.

Υπάρχουν δύο τρόποι για να εισάγουμε τιμές στο h:selectOneMenu:

1. Εισαγωγή τιμών στατικά (Οι τιμές δεν αλλάζουν πάντα είναι πάντα οι ίδιες).
2. Εισαγωγή τιμών δυναμικά (Οι τιμές προέρχονται από μια βάση δεδομένων.

Η γενική μορφή που ισχύει για το h:selectOnemenu και για τους δύο τρόπους είναι η εξής:

<h:selectOneMenu value=" ">
<f:selectItem value=" "/> ή <f:selectItems value=" "/>
</h:selectOneMenu>

Όπως βλέπουμε το εργαλείο αυτό ανοίγει και κλείνει ένα μπλοκ <h:selectOneMenu> </h:selectOneMenu>. Στα δεξιά της ετικέτας που το ανοίγει υπάρχει μια ιδιότητα value. Με αυτή την ιδιότητα συνδέουμε το εργαλείο αυτό με ένα bean καθώς μπορούμε να περάσουμε την τιμή που επέλεξε ένας χρήστης από το μενού στο πεδίο ενός bean ώστε να το διαχειριστούμε ανάλογα στον κώδικα μας.

Για παράδειγμα μπορούμε να γράψουμε:

<h:selectOneMenu value = “#{exam.selectedcourse}”>

Με την υλοποίηση αυτή έχουμε μεταφέρει μια επιλογή από το μενού στην κλάση exam και συγκεκριμένα στο πεδίο selectedcourse.

6.10.1 Στατική υλοποίηση

Στη συνέχεια βλέπουμε το <f:selectItem /> το οποίο και αυτό περιλαμβάνει την ιδιότητα value μόνο που έχει διαφορετική λειτουργία από το προηγούμενο.

Σε αυτό το value εισάγουμε μια και μόνο τιμή που θέλουμε να εμφανίζεται στο μενού μας. Για κάθε μια ορίζουμε ένα ξεχωριστό selectItem στην ιδιότητα value.

```
<h:selectOneMenu value =”#{exam.selectedcourse}”>
        <f:selectItem value=”Java”/>
        <f:selectItem value=”HTML”/>
        <f:selectItem value=”DataBases”/>
</h:selectOneMenu>
```

6.10.2 Δυναμική υλοποίηση

Σε ορισμένες περιπτώσεις που οι τιμές ενός h:selectOneMenu δεν θέλουμε να είναι πάντα οι ίδιες αυτή η υλοποίηση δεν μας καλύπτει. Για να το καταφέρουμε αυτό πρέπει να βρούμε έναν τρόπο να εισάγουμε δυναμικά τις τιμές που θέλουμε, από μια βάση δεδομένων. Αυτό θα γίνει με την δημιουργία ενός bean που περιλαμβάνει μια μέθοδο και θα επιστρέφει τις τιμές που θέλουμε στην ιδιότητα value αλλά όχι της ετικέτας f:selectItem αλλά της f:selectItems.

```
<h:selectOneMenu  value="#{exam.selectedcourse}">
        <f:selectItems value="#{exam.coursess}"/>
</h:selectOneMenu>
```

Εδώ βλέπουμε το αντίστοιχο τμήμα της υλοποίησης αυτής που ανήκει σε μια σελίδα JSF. Στο πεδίο value βλέπουμε ότι πλέον δεν υπάρχουν ονόματα μαθημάτων αλλά αναφερόμαστε στην μέθοδο getcoursess του bean exam.

Κομμάτι της μεθόδου getcoursess( ) που επιστρέφει τους τίτλους μαθημάτων

```
public List<SelectItem> getcoursess()
{
   • • •
  try
  {
       • • •
       String s = "SELECT ctitle FROM courses";
       smt= c.createStatement();
       ResultSet rs = smt.executeQuery(s);
       while (rs.next())
         { retVal.add(newSelectItem(rs.getString("ctitle"))); }
  }
   • • •
  return retVal;}
```

6.11 Ετικέτα πίνακα δεδομένων: h:dataTable

Σχεδόν σε όλες τις εφαρμογές ανεξάρτητα αν είναι διαδικτυακές ή όχι γίνεται χρήση ενός πίνακα για να εμφανίσουμε δεδομένα που προέρχονται από μία βάση δεδομένων. Είναι από τα πιο πολύπλοκα εργαλεία που μπορεί να συναντήσει κάποιος και το τελευταίο που χρησιμοποιήθηκε για την κατασκευή του συστήματος εξέτασης. Το JSF παρέχει ένα, μέσω της ετικέτας h:dataTable.
Η γενική του μορφή είναι η εξής:

Παράδειγμα από το αρχείο professor-page.xhtml

```
<h:dataTable value="#{questionlist.getQuestionList()} "var="q" border="3" headerClass="heading">
  <h:column>
  <f:facet name="header">
  Qid
  </f:facet>
  #{q.qid}
  </h:column>
</h:dataTable>
```

Για να ορίσουμε έναν πίνακα πρέπει να περικλύσουμε τον κώδικα που τον αποτελεί, μέσα στο μπλοκ <h:dataTable> </h:dataTable>. Στην ιδιότητα value, φορτώνουμε τις αντίστοιχες τιμές που θέλουμε να περιέχει ο πίνακας. Οι τιμές αυτές προέρχονται από μια βάση δεδομένων και πρέπει να έχει οριστεί ένα bean ώστε να τις αντλήσουμε. Εφόσον ξεπεράσουμε αυτό το στάδιο τις έχουμε μεταφέρει στην σελίδα JSF και συγκεκριμένα μέσα στην ιδιότητα var.

Παράδειγμα από το αρχείο QuestionList.java

```
public List<Questtion> getQuestionList()
{  • • •
        while (r1.next())
                {
                Questtion qst = new Questtion();
                qst.setqid(r1.getString("qid"));
                qst.setqtitle(r1.getString("qtitle"));
                qst.setqanswer1(r1.getString("qanswer1"));
                qst.setqanswer2(r1.getString("qanswer2"));
                qst.setqanswer3(r1.getString("qanswer3"));
                qst.setqcorrectanswer(r1.getString("qcorrectanswer"));
                list.add(qst);
                }

    • • •
return list;
}
```

Για να δημιουργήσουμε τις στήλες που επιθυμούμε για τον πίνακα μας, ανοίγουμε ένα νέο μπλοκ εντός του αρχικού με τις ετικέτες <h:column> </h:column>. Στην συνέχεια ο πίνακας διαθέτει επικεφαλίδες για να δηλώσει τον τύπο των δεδομένων. Για να ορίσουμε μια επικεφαλίδα περικλείουμε το όνομα της μέσα στο <f:facet name=*"header"*> </f:facet>.

΄Έχουμε την δυνατότητα αν το επιθυμούμαι πέραν της εμφάνισης δεδομένων να δημιουργήσουμε στήλες που θα περιέχουν άλλα εργαλεία του JSF. Όπως για παράδειγμα κουμπιά που εκτελούν μια διαγραφή από τα δεδομένα του πίνακα.

Παράδειγμα από το αρχείο professor-page.xhtml

```
<h:column>
<h:commandButton class="btnbtn-primary" value="Διαγραφή"
        action="#{question.deletequestion(q.qid)}" />
</h:column>
```

ΚΕΦΑΛΑΙΟ 7

ΣΥΜΠΕΡΑΣΜΑΤΑ

Η Πτυχιακή εργασία αποτελεί σίγουρα ένα από τα πιο βασικά κομμάτια στην πορεία που έχει κάποιος ως σπουδαστής. Είναι στην ουσία ένα πρώτο έργο για εμάς και μας αναγκάζει να σκεφτούμε με έναν διαφορετικό τρόπο από αυτόν που είχαμε συνηθίσει ως σπουδαστές. Με την εκπόνησή της βλέπουμε, ότι πλέων δεν υπάρχει κάποιο έτοιμο σχέδιο- χάρτης που σου προσφέρει ένας καθηγητής και εσύ τον ακολουθείς αλλά, πρέπει μόνος σου να γράψεις το δικό σου χάρτη. Βγαίνοντας από αυτή την διαδικασία σίγουρα θα είσαι πιο έμπειρος και πιο σίγουρος για τις προτάσεις σου και τις γνώσεις σου.

Με την υλοποίηση της δικής μου Πτυχιακής εργασίας κατάλαβα ότι, το μεγαλύτερο μέρος μιας εφαρμογής πρέπει να αποτελεί ο σχεδιασμός και η εύρεση των σωστών εργαλείων.
Στον σχεδιασμό συνάντησα το πρώτο μεγάλο μου εμπόδιο. Όταν δεν έχεις δημιουργήσει ποτέ κάτι ολοκληρωμένο και σκέφτεσαι μόνο ως προγραμματιστής και όχι σαν χρήστης σου είναι πολύ δύσκολο να οραματιστείς πως θα είναι αυτό που θέλεις να φτιάξεις (π.χ. τι λειτουργίες θα καλύψω; τι σελίδες θα κατασκευάσω; πως θα τις οργανώσω;).

Από την στιγμή που καταλήξεις σε ένα σχέδιο σειρά έχει να επιλέξεις και το κατάλληλο εργαλείο. Το εργαλείο που επέλεξα ήταν το Eclipse-IDE. Δυστυχώς δεν έδωσα ιδιαίτερη προσοχή στην επιλογή κάποιου άλλου καθώς το συγκεκριμένο χρησιμοποίησα στη σχολή και εκείνη την περίοδο φαινόταν αρκετά πλήρης. Στην πορεία ανακάλυψα πως με την επιλογή μου ίσως δυσκόλεψα τον εαυτό μου άδικα γιατί μετά από ορισμένες αναποδιές ανακάλυψα πολύ πιο εύχρηστα περιβάλλοντα.

Ενδεικτικά να αναφέρω ότι στο Eclipse-IDE για να εισάγω τιμές σε ένα h:dataTable ; έπρεπε να γράψω μια σειρά από κλάσεις προκειμένου να συνδεθώ στην βάση και να φορτώσω τα δεδομένα που χρειαζόμουν. Η υλοποίηση των κλάσεων παρουσίασε ένα κομμάτι δυσκολίας. Σε άλλα περιβάλλοντα ανάπτυξης όπως το Jdeveloper οι ενέργειες αυτές γίνονται με ορισμένα κλικ. Επίσης στο Eclipse-IDE δεν υπήρχε βοήθεια στο να γλυτώσω χρόνο από το templating αντίθετα με το Jdeveloper που προσφέρει ένα μηχανισμό.

Όσο αναφορά τον κώδικα που έγραψα πιστεύω ότι λόγο της μεγάλης απειρίας μου δεν τον οργάνωσα σωστά προκειμένου να βοηθήσω τον εαυτό μου. Κατέληξα σε αρκετά σημεία της εφαρμογής να επαναλαμβάνω κώδικα τον οποίο με καλύτερη οργάνωση θα είχα αποφύγει. Πιστεύω ότι στο πρόβλημα αυτό συνέβαλε και η ύπαρξη πολλών περισσοτέρων κλάσεων από αυτές που είχα δει σε κάποιο εγχείρημα μου. Σε συνέχεια αυτού παρατηρήθηκαν δυσκολίες στην σύνδεση όλων αυτών των κλάσεων μεταξύ τους. Αλλά γενικά όλα αυτά τα προβλήματα με την απόκτηση εμπειρίας σίγουρα διορθώνονται.

Τα πιο δύσκολα προβλήματα είναι αυτά για τα οποία δεν έχεις ένδειξη ότι είναι προβλήματα. Επειδή η Java χρησιμοποιεί μια διαδικασία η οποία ονομάζεται mirroring και πρέπει ορισμένα κομμάτια να είναι γραμμένα με συγκεκριμένο τρόπο προκειμένου να λειτουργήσει ο συγκεκριμένος μηχανισμός. Εάν δεν τα έχουμε συντάξει σωστά τότε η εφαρμογή δεν λειτουργεί - δεν υπάρχει ένδειξη ως προς το λάθος και απλά αναρωτιέσαι τι συμβαίνει (ίσως είναι κάποιο κεφαλαίο γράμμα).

Όσο αφορά το Διαδικτυακό Σύστημα Εξέτασης πιστεύω προκύπτουν θετικές αλλά και αρνητικές προοπτικές.
Το μεγαλύτερο θετικό που προσφέρει έχει σχέση με τον χρόνο. Η διεξαγωγή μιας ηλεκτρονικής εξέτασης που αποτελείται από την κατασκευή ενός διαγωνίσματος, την εξέταση από τον σπουδαστή και την διόρθωση, γίνεται πολύ πιο γρήγορα σε σχέση με τον αντίστοιχο γραπτό τρόπο. Σε συνέχεια αυτού, προκύπτουν και άλλα θετικά όπως, η πιο συχνή πραγματοποίηση εξετάσεων, και δραστηριοποίηση των σπουδαστών λόγω της συχνότερης εξέτασης
Ένα αρνητικό το οποίο μπορεί να θεωρήσει κάποιος είναι πως σε μια ηλεκτρονική εξέταση, δεν φαίνεται η συνολική εικόνα των σκέψεων ενός σπουδαστή για μια απάντηση, διότι αυτή θα είναι τυποποιημένη. Όμως, σαν συμπλήρωμα της εκπαιδευτικής διαδικασίας θα μπορούσε να προηγηθεί της γραπτής εξέτασης.

Θεωρώ ότι σαν εφαρμογή επιδέχεται εξέλιξη με την προσθήκη νέων λειτουργιών στο σύστημα (π.χ. διαφορετικό τύπο ασκήσεων), αλλά και καλύτερη μορφοποίηση του περιβάλλοντος των χρηστών.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Βιβλία:

- Γεώργιος, Λ..(2008). «*Εισαγωγή στην Java*» Αθήνα , Εκδόσεις Κλειδάριθμος Τζιόλα,2008.ISBN:978-960-461-169-0

- David, G. , Cay, H..(2012).«*Core JavaServer Faces*» (3η εκδ.), United States Εκδόσεις Prentice Hall,2012.ISBN:978-0-13-701289-3
  - Ένα πολύ καλό βιβλίο για να κατανόηση κάποιος όλες τις αρχικές έννοιες που χρειάζεται για να ξεκινήσει. Δεν θα επιμείνω σε λεπτομέρειες για να αποδείξω γιατί είναι καλό αυτό το βιβλίο αλλά θα επισημάνω το εξής. Ήταν γραμμένο στα Αγγλικά και αυτό μου πρόσφερε στο να κατανοήσω καλύτερα την ορολογία που χρειάζομαι χωρίς τις άστοχες μεταφράσεις.

Ιστότοποι:

- http://www.oracle.com/index.html

- https://en.wikipedia.org/wiki/Main_Page

- http://stackoverflow.com/
  - Οι περισσότερες απαντήσεις που ψάχνει κάποιος για ένα πρόβλημα ανεξάρτητα σε ποια τεχνολογία προγραμματίζει βρίσκονται εδώ. Πιθανότατα, η μανία των χρηστών να απαντήσουν στο ερώτημα κάποιου προκειμένου να συλλέξουν τα περισσότερα αστεράκια έχει ως αποτέλεσμα πολύ σωστές απαντήσεις. Με βοήθησε σε πολλά κομμάτια της Πτυχιακής μου εργασίας.

- http://www.coreservlets.com/
  - Ένας πολύ χρήσιμος Ιστότοπος, κυρίως γιατί περιλαμβάνει πολύ αναλυτικές οδηγίες κυρίως όσο αναφορά την εγκατάσταση του JSF στο Eclipse- IDE. Προσφέρει ένα τρόπο υλοποίησης για διάφορα θέματα, απλά μετά από ένα σημείο είναι λίγο δύσκολο να το ακολουθήσεις.

ΠΑΡΑΡΤΗΜΑ

Παρουσιάζεται ένα μέρος των Bean και των σελίδων της εφαρμογής

```
package com.dev.user.model;

import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.faces.bean.SessionScoped;
import java.util.Random;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;
import javax.faces.model.SelectItem;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.faces.bean.SessionScoped;
import java.sql.*;
import com.mysql.jdbc.Statement;

@ManagedBean(name="resultlist")
@RequestScoped
public class Resultlist
{
	List<Result> list;
	PreparedStatement p7 = null;
	Connection c7 = null;
	ResultSet r7 = null;

	List<User> studentlist;
	PreparedStatement p1 = null;


	private String selectedcourse;
	private String selectedstudent;

	public String getselectedcourse()
	{ return selectedcourse; }
```

```
        public void setselectedcourse(String selectedcourse)
        { this.selectedcourse = selectedcourse; }

        public String getselectedstudent()
        { return selectedstudent; }

        public void setselectedstudent(String selectedstudent)
        { this.selectedstudent = selectedstudent; }

        public List<Result> getList()
          { return list; }

        public List<Result> getResultList()
        {
                list = new ArrayList<Result>();

                String tmpcid="";
                java.sql.Statement smt1 = null;
                Connection cone = null;

                try
                        {

                                Class.forName("com.mysql.jdbc.Driver");
                                cone =
DriverManager.getConnection("jdbc:mysql://localhost:3306/myapp", "root", "root");
                                System.out.println(selectedcourse);
                                String a1 = "select cid from courses where ctitle ='" +
selectedcourse + "'";
                        smt1= cone.createStatement();
                           ResultSet rs1 = smt1.executeQuery(a1);
                           if (rs1.next() )
                             { System.out.println(rs1.getString("cid"));
                               tmpcid = rs1.getString("cid"); }
                           else
                             { System.out.println("den to vrika"); }
                           }catch(Exception e)
                           { e.printStackTrace(); }

                try
                {
                        Class.forName("com.mysql.jdbc.Driver");
                c7 =
DriverManager.getConnection("jdbc:mysql://localhost:3306/myapp", "root", "root");
```

```
                String sql = "select * from results where sid='" + selectedstudent + "'" +
" and " + "cid='" + tmpcid + "'";
                p7 = c7.prepareStatement(sql);
                r7= p7.executeQuery();

                while (r7.next())
                {
                        Result res = new Result();
                        res.setqtitle(r7.getString("qtitle"));
                        res.setqcorrectanswer(r7.getString("correctanswer"));
                        res.setsanswer(r7.getString("sanswer"));
                        res.setsid(r7.getString("sid"));
                        res.setcid(r7.getString("cid"));
                        list.add(res);
                }
                }
                catch(Exception e)
                {
                        e.printStackTrace();
                }

                return list;
        }

        public List<SelectItem> getcoursess()
        {
                Connection c = null;
           java.sql.Statement smt = null;
           List<SelectItem> retVal = new ArrayList<SelectItem>();
           try {

                Class.forName("com.mysql.jdbc.Driver");
                c = DriverManager.getConnection("jdbc:mysql://localhost:3306/myapp",
"root", "root");

                String s = "SELECT ctitle FROM courses";
                smt= c.createStatement();
                ResultSet rs = smt.executeQuery(s);
                while (rs.next()) {
                                retVal.add(new SelectItem(rs.getString("ctitle")));
                }
           }

           catch(Exception e)
           { System.out.println("Error");  }
```

```
            finally
      {
         try
         {c.close();}
         catch(Exception e)
          { e.printStackTrace(); }
      }
                  return retVal;
         }

         public List<SelectItem> getstudents()
         {
                  Connection c = null;
            java.sql.Statement smt = null;
            List<SelectItem> retstu = new ArrayList<SelectItem>();
            try {

                  Class.forName("com.mysql.jdbc.Driver");
                  c = DriverManager.getConnection("jdbc:mysql://localhost:3306/myapp",
"root", "root");

                  String s = "SELECT username FROM users where level=3";
                  smt= c.createStatement();
                  ResultSet rs = smt.executeQuery(s);
                  while (rs.next()) {
                           retstu.add(new SelectItem(rs.getString("username")));
                  }
            }

            catch(Exception e)
            { System.out.println("Error");  }
            finally
      {
         try
         {c.close();}
         catch(Exception e)
          { e.printStackTrace(); }
      }
                  return retstu;
         }



         public List<User> getSList()
          { return studentlist; }
```

```
	public List<User> getStudentList()
	{
		studentlist = new ArrayList<User>();
		java.sql.Statement smt = null;
		Connection c1 = null;
		try
		{
			Class.forName("com.mysql.jdbc.Driver");
			c1 =
DriverManager.getConnection("jdbc:mysql://localhost:3306/myapp", "root", "root");

			String s = "SELECT * FROM students WHERE sid='" +
selectedstudent + "'";
		smt= c1.createStatement();
		ResultSet rs = smt.executeQuery(s);
		while (rs.next())
		{
			User usr = new User();
			usr.setsfname(rs.getString("sfname"));
			usr.setslname(rs.getString("slname"));
			usr.setssemester(rs.getString("ssemester"));
			usr.setsemail(rs.getString("semail"));
			studentlist.add(usr);
		}
		}
		catch(Exception e)
	   { System.out.println("Error");  }


		return studentlist;
	}


}
```

```
package com.dev.user.model;

import java.io.IOException;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebFilter;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


/**
 * Servlet Filter implementation class AuthFilter
 */
@WebFilter("/*")
public class AuthFilter implements Filter {

    /**
     * Default constructor.
     */
    public AuthFilter() {
        // TODO Auto-generated constructor stub
    }

	/**
	 * @see Filter#destroy()
	 */
	public void destroy() {
		// TODO Auto-generated method stub
	}

	/**
	 * @see Filter#doFilter(ServletRequest, ServletResponse, FilterChain)
	 */
	public void doFilter(ServletRequest request, ServletResponse response,
FilterChain chain) throws IOException, ServletException {
		HttpServletRequest req = (HttpServletRequest) request;
		login auth = (login) req.getSession().getAttribute("login");

		String reqURI = req.getRequestURI();
		int neededAuth = 0;
		if (reqURI.indexOf("/student-page.xhtml")>=0 ||
reqURI.indexOf("/exam.xhtml")>=0 ) {
			neededAuth = 3;
		} else if (reqURI.indexOf("/professor-page.xhtml")>=0) {
			neededAuth = 2;
		} else if (reqURI.indexOf("/admin-page.xhtml")>=0) {
			neededAuth = 1;
		}
	    if ( reqURI.indexOf("/starting-page.xhtml") >= 0 ||
	        (auth != null && auth.getAuthLevel() == neededAuth)) {
```

```
                // User is logged in, so just continue request.
                chain.doFilter(request, response);
            } else {
                // User is not logged in, so redirect to index.
                HttpServletResponse res = (HttpServletResponse) response;
                res.sendRedirect(req.getContextPath() + "/starting-page.xhtml");
            }
        }

        /**
         * @see Filter#init(FilterConfig)
         */
        public void init(FilterConfig fConfig) throws ServletException {
                // TODO Auto-generated method stub
        }

}
```

```
package com.dev.user.model;
import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;

import java.util.Random;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

import javax.faces.model.SelectItem;

import java.sql.*;

import com.mysql.jdbc.Statement;

@ManagedBean(name="course")
@RequestScoped
public class Course
{
        private String ctitle;
        private String cdescription;
        private String ctheory;

        public String getctitle()
        { return ctitle; }
```

```
        public void setctitle(String ctitle)
        { this.ctitle = ctitle; }

        public String getcdescription()
        { return cdescription; }

        public void setcdescription(String cdescription)
        { this.cdescription = cdescription; }

        public String getctheory()
        { return ctheory; }

        public void setctheory(String ctheory)
        { this.ctheory = ctheory; }

        public String generator()
        {
                String number;
                int start = 1000;
           int end = 2000;
           Random random = new Random();

           long range = (long)end - (long)start + 1;
           long fraction = (long)(range * random.nextDouble());
           int randomnumber =  (int)(fraction + start);
           number = String.valueOf(randomnumber);
                return "c" + number;
        }

        public String addcourse()
        {
                Connection co = null;
           PreparedStatement ps3 = null;
           int i;
           try {

                Class.forName("com.mysql.jdbc.Driver");
                co =
DriverManager.getConnection("jdbc:mysql://localhost:3306/myapp", "root", "root");
                String csql = "INSERT INTO courses(cid, ctitle, ctheory, cdescription)
VALUES(?,?,?,?)";
                ps3= co.prepareStatement(csql);
                ps3.setString(1, generator());
              ps3.setString(2, ctitle);
```

```
                ps3.setString(3, ctheory);
                ps3.setString(4, cdescription);
                i = ps3.executeUpdate();

            }
            catch(Exception e)
              {
                        System.out.println("Error" + e.getMessage());
                return "notlogin";
              }

                return "";
        }
}//class end
```

```
package com.dev.user.model;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;

import java.util.Random;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

import javax.faces.model.SelectItem;

import java.sql.*;

import com.mysql.jdbc.Statement;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
```

```
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.faces.bean.SessionScoped;

import java.sql.*;

import com.mysql.jdbc.Statement;

import javax.faces.bean.ManagedProperty;
@ManagedBean(name="exam")
@SessionScoped

public class Exam implements Serializable
{   String tmp ="";

        @ManagedProperty(value="#{login.username}")//12/1/2015
        private String examinedStudent;//12/1/2015

        private String selectedcourse;
        private int currentquestion;
        private int score;
        private String response = "";
        private ArrayList<Questtion> questions;
                        Connection c1 = null;
                PreparedStatement p1 = null;
                        ResultSet r1 = null;

                        Connection c2 = null;
                PreparedStatement p2 = null;
                        ResultSet r2 = null;

        public String getExaminedStudent()//12/1/2015
        {       return examinedStudent;}

        public void setExaminedStudent(String examinedStudent)//12/1/2015
        {       this.examinedStudent=examinedStudent;}

        public int getcurrentquestion()
        {       return  currentquestion+1;}
```

```
public String getqanswer1()
{       return questions.get(currentquestion).getqanswer1(); }

public String getqanswer2()
{       return questions.get(currentquestion).getqanswer2(); }

public String getqanswer3()
{       return questions.get(currentquestion).getqanswer3(); }


public String getQuestion()
{
        if (questions==null) {
          initQuestions();
        }

        return questions.get(currentquestion).getqtitle();
}

public int getScore()
{       return score; }

public String getResponse()
{       return response;}

public void setResponse(String newValue)
{       response = newValue;}

public String getselectedcourse()
{ return selectedcourse; }

public void setselectedcourse(String selectedcourse)
{ this.selectedcourse = selectedcourse; }


public Exam () {

}

private void initQuestions() {
        String tmpcid="";
        questions = new ArrayList<Questtion>();
        java.sql.Statement smt1 = null;
        Connection cone = null;
```

```
                try
                {

                        Class.forName("com.mysql.jdbc.Driver");
                        cone =
DriverManager.getConnection("jdbc:mysql://localhost:3306/myapp", "root", "root");
                        System.out.println(selectedcourse);
                        String a1 = "select cid from courses where ctitle ='" +
selectedcourse + "'";
             smt1= cone.createStatement();
                ResultSet rs1 = smt1.executeQuery(a1);
                if (rs1.next() )
                 { System.out.println(rs1.getString("cid"));
                   tmpcid = rs1.getString("cid"); }
                else
                 { System.out.println("den to vrika"); }
                }catch(Exception e)
                { e.printStackTrace(); }

           try
           {
                   Class.forName("com.mysql.jdbc.Driver");
                                c1 =
DriverManager.getConnection("jdbc:mysql://localhost:3306/myapp", "root", "root");


                        String sql = "select * from questions where cid='" + tmpcid + "'";
                p1 = c1.prepareStatement(sql);
                r1= p1.executeQuery();

                while (r1.next())
                {
                        Questtion qst = new Questtion();
                        qst.setqid(r1.getString("qid"));
                        qst.setqtitle(r1.getString("qtitle"));
                        qst.setqanswer1(r1.getString("qanswer1"));
                        qst.setqanswer2(r1.getString("qanswer2"));
                        qst.setqanswer3(r1.getString("qanswer3"));
                        qst.setqcorrectanswer(r1.getString("qcorrectanswer"));
                        questions.add(qst);
                }
                }
                catch(Exception e)
                {
                        e.printStackTrace();
```

```
                }

                //return questions;
        }

        public String answerAction()
        {

              PreparedStatement ps = null;
              Connection con = null;
              int i = 0;

              java.sql.Statement smt1 = null;
                    Connection cone = null;
                    String tmpcid="";

              try
              {
                     Class.forName("com.mysql.jdbc.Driver");
                     con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/myapp", "root", "root");

                     Class.forName("com.mysql.jdbc.Driver");
                           cone =
DriverManager.getConnection("jdbc:mysql://localhost:3306/myapp", "root", "root");
                           System.out.println(selectedcourse);
                           String a1 = "select cid from courses where ctitle ='" +
selectedcourse + "'";
               smt1= cone.createStatement();
                  ResultSet rs1 = smt1.executeQuery(a1);
                  if (rs1.next() )
                    { System.out.println(rs1.getString("cid"));
                      tmpcid = rs1.getString("cid"); }
                  else
                    { System.out.println("den to vrika"); }

                     String ssql = "INSERT INTO RESULTS(qtitle, correctanswer, sanswer, sid,
cid) VALUES(?,?,?,?,?)";
                     ps= con.prepareStatement(ssql);
                     ps.setString(1, questions.get(currentquestion).getqtitle());
                     ps.setString(2, questions.get(currentquestion).getqcorrectanswer());
                     ps.setString(3, response);
                     ps.setString(4, examinedStudent);
                     ps.setString(5, tmpcid);
```

```
                    i = ps.executeUpdate();
                }
                    catch(Exception e)
                    { System.out.println(e); }

                if (questions.get(currentquestion).isCorrect(response))
                {
                        score++;System.out.println(score);
                        nextQuestion();System.out.println("hi");

                        if (currentquestion == questions.size())
                        {System.out.println("ops");
                        return "done";}
                }
                else
                {
                        nextQuestion();
                        if (currentquestion == questions.size())
                                {System.out.println("ops"); return "done";}
        }
            return "";
          }

          private void nextQuestion()
          {
                currentquestion++;
                response = "";
          }

          public List<SelectItem> getcoursess()
          {
                Connection c = null;
             java.sql.Statement smt = null;
             List<SelectItem> retVal = new ArrayList<SelectItem>();
             try {

                Class.forName("com.mysql.jdbc.Driver");
                c = DriverManager.getConnection("jdbc:mysql://localhost:3306/myapp",
"root", "root");

                String s = "SELECT ctitle FROM courses";
                smt= c.createStatement();
                ResultSet rs = smt.executeQuery(s);
                while (rs.next()) {
                                retVal.add(new SelectItem(rs.getString("ctitle")));
```

```
                    }
                }

                catch(Exception e)
                { System.out.println("Error");  }
                finally
        {
            try
            {c.close();}
            catch(Exception e)
              { e.printStackTrace(); }
        }
                    return retVal;
            }
}//class end
```

```
package com.dev.user.model;

import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import java.util.Random;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;
import javax.faces.model.SelectItem;
import java.sql.*;
import com.mysql.jdbc.Statement;
import java.io.Serializable;


@ManagedBean(name="question")
@RequestScoped
public class Questtion implements Serializable
{
        private String qid;
        private String qtitle;
        private String qanswer1;
        private String qanswer2;
        private String qanswer3;
        private String qcorrectanswer;
        private String selectedcourse;
```

```
public String getqid()
{ return qid; }

public void setqid(String qid)
{ this.qid = qid; }

public String getqtitle()
{ return qtitle; }

public void setqtitle(String qtitle)
{ this.qtitle = qtitle; }

public String getqanswer1()
{ return qanswer1; }

public void setqanswer1(String qanswer1)
{ this.qanswer1 = qanswer1; }

public String getqanswer2()
{ return qanswer2; }

public void setqanswer2(String qanswer2)
{ this.qanswer2 = qanswer2; }

public String getqanswer3()
{ return qanswer3; }

public void setqanswer3(String qanswer3)
{ this.qanswer3 = qanswer3; }

public String getqcorrectanswer()
{ return qcorrectanswer; }

public void setqcorrectanswer(String qcorrectanswer)
{ this.qcorrectanswer = qcorrectanswer; }

public String getselectedcourse()
{ return selectedcourse; }

public void setselectedcourse(String selectedcourse)
{ this.selectedcourse = selectedcourse; }

public String generator(int instart,int inend,String inchar)
{
        String number;
```

```
            Random random = new Random();

            long range = (long)inend - (long)instart + 1;
            long fraction = (long)(range * random.nextDouble());
            int randomnumber =  (int)(fraction + instart);
            number = String.valueOf(randomnumber);
                  return inchar + number;
         }

         public List<SelectItem> getcoursess()
         {
                  Connection c = null;
            java.sql.Statement smt = null;
            List<SelectItem> retVal = new ArrayList<SelectItem>();
            try {

                  Class.forName("com.mysql.jdbc.Driver");
                  c = DriverManager.getConnection("jdbc:mysql://localhost:3306/myapp",
"root", "root");

                  String s = "SELECT ctitle FROM courses";
                  smt= c.createStatement();
                  ResultSet rs = smt.executeQuery(s);
                  while (rs.next()) {
                              retVal.add(new SelectItem(rs.getString("ctitle")));
                  }
            }

            catch(Exception e)
            { System.out.println("Error");  }
            finally
      {
            try
            {c.close();}
            catch(Exception e)
              { e.printStackTrace(); }
      }
                  return retVal;
            }

            public String addquestion()
            {
                  Connection cone = null;
                  PreparedStatement s = null;
                  java.sql.Statement smt1 = null;
```

```
                int i;


            try
            {
                String qid = generator(1000,10000,"q");

                Class.forName("com.mysql.jdbc.Driver");
                cone =
DriverManager.getConnection("jdbc:mysql://localhost:3306/myapp", "root", "root");

                String a1 = "SELECT cid FROM courses WHERE ctitle ='" +
selectedcourse + "'";
              smt1= cone.createStatement();
                ResultSet rs1 = smt1.executeQuery(a1);
                if (rs1.next() )
                 { System.out.println(rs1.getString("cid")); }

                String tmpcid = rs1.getString("cid");

                String ssql = "INSERT INTO questions(qid, qtitle, qanswer1, qanswer2,
qanswer3, qcorrectanswer, cid) VALUES(?,?,?,?,?,?,?)";
              s= cone.prepareStatement(ssql);
                   s.setString(1, qid);
                   s.setString(2, qtitle);
                   s.setString(3, qanswer1);
                   s.setString(4, qanswer2);
                   s.setString(5, qanswer3);
                   s.setString(6, qcorrectanswer);
                   s.setString(7, tmpcid);

                   i = s.executeUpdate();

            }
            catch(Exception e)
            { System.out.println("Error");  }

            return "";
        }

        public void deletequestion(String qid)
        {
                PreparedStatement p = null;
                Connection c = null;
                int y;
```

```
                    try
                    {
                            Class.forName("com.mysql.jdbc.Driver");
                    c = DriverManager.getConnection("jdbc:mysql://localhost:3306/myapp",
"root", "root");
                    String dsql = "DELETE FROM questions WHERE qid='" + qid + "'" ;
                    p= c.prepareStatement(dsql);

                y = p.executeUpdate();
                if(y >0)
                {
                System.out.println("Row deleted successfully");
                System.out.println(qid);
                }
                else
                {
                System.out.println("Row not deleted successfully");
                }
                    }
                    catch(Exception e)
                    {
                            e.printStackTrace();
                    }
          }

          public boolean isCorrect(String response)
          {
            return response.trim().equalsIgnoreCase(qcorrectanswer);
    }
}//class end
```

```
package com.dev.user.model;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.faces.bean.SessionScoped;

import java.util.Random;
```

```
import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

import javax.faces.model.SelectItem;
import com.mysql.jdbc.Statement;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.faces.bean.SessionScoped;
import java.sql.*;
@ManagedBean(name="questionlist")
@SessionScoped
public class Questionlist
{
	List<Questtion > list;
	PreparedStatement p1 = null;
	Connection c1 = null; Connection c2 = null;
	ResultSet r1 = null;
	PreparedStatement s = null;
	java.sql.Statement smt1 = null;

	private String selectedcourse;

	public String getselectedcourse()
	{ return selectedcourse; }

	public void setselectedcourse(String selectedcourse)
	{ this.selectedcourse = selectedcourse; }

	public List<SelectItem> getcoursess()
	{
		Connection c = null;
	   java.sql.Statement smt = null;
	   List<SelectItem> retVal = new ArrayList<SelectItem>();
	   try {

		Class.forName("com.mysql.jdbc.Driver");
		c = DriverManager.getConnection("jdbc:mysql://localhost:3306/myapp",
"root", "root");

		String s = "SELECT ctitle FROM courses";
```

```
                smt= c.createStatement();
                ResultSet rs = smt.executeQuery(s);
                while (rs.next()) {
                                retVal.add(new SelectItem(rs.getString("ctitle")));
                }
          }

          catch(Exception e)
          { System.out.println("Error");  }
          finally
      {
         try
         {c.close();}
         catch(Exception e)
          { e.printStackTrace(); }
      }
                return retVal;
         }

         public List<Questtion> getList()
          { return list; }

         public List<Questtion> getQuestionList()
         {
                list = new ArrayList<Questtion>();
                String tmpcid="";
                java.sql.Statement smt1 = null;
                Connection cone = null;

                try
                        {

                                Class.forName("com.mysql.jdbc.Driver");
                                cone =
DriverManager.getConnection("jdbc:mysql://localhost:3306/myapp", "root", "root");
                                System.out.println(selectedcourse);
                                String a1 = "select cid from courses where ctitle ='" +
selectedcourse + "'";
                        smt1= cone.createStatement();
                           ResultSet rs1 = smt1.executeQuery(a1);
                           if (rs1.next() )
                             { System.out.println(rs1.getString("cid"));
                               tmpcid = rs1.getString("cid"); }
                           else
                             { System.out.println("den to vrika"); }
```

```
                        }catch(Exception e)
                        { e.printStackTrace(); }

                try
                {
                        Class.forName("com.mysql.jdbc.Driver");
                c1 =
DriverManager.getConnection("jdbc:mysql://localhost:3306/myapp", "root", "root");

                //Questtion q=new Questtion();
                /*String a1 = "SELECT cid FROM courses WHERE ctitle='" +
selectedcourse + "'";
              smt1= c2.createStatement();
                ResultSet rs1 = smt1.executeQuery(a1);
                if (rs1.next() )
                  { System.out.println(rs1.getString("cid")); }
                    String tmpcid = rs1.getString("cid");*/



                String sql = "select * from questions where cid='" + tmpcid + "'";
                p1 = c1.prepareStatement(sql);
                r1= p1.executeQuery();

                while (r1.next())
                {
                        Questtion qst = new Questtion();
                        qst.setqid(r1.getString("qid"));
                        qst.setqtitle(r1.getString("qtitle"));
                        qst.setqanswer1(r1.getString("qanswer1"));
                        qst.setqanswer2(r1.getString("qanswer2"));
                        qst.setqanswer3(r1.getString("qanswer3"));
                        qst.setqcorrectanswer(r1.getString("qcorrectanswer"));
                        list.add(qst);
                        Map<String, Boolean> checked = new HashMap<String,
Boolean>();
                        List<Questtion> checkedItems = new ArrayList<Questtion>();
                        for (Questtion item : list)
                        {
                        if (checked.get(item.getqid())!= null)
                           {
                                checkedItems.add(item);
                                qst.deletequestion(qst.getqid());
                              }
                        }
```

```
                    }
                    }
                    catch(Exception e)
                    {
                            e.printStackTrace();
                    }

                    return list;
            }
}//class end
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">


<h:head>
<title>Administrator</title>
<link href="./bootstrap/css/bootstrap.css"
      rel="stylesheet" type="text/css"/>
<link href="./bootstrap/css/table-styles.css"
      rel="stylesheet"/>
</h:head>


<h:body>
<h1 align="center">Α.Τ.Ε.Ι ΗΠΕΙΡΟΥ</h1>

<div id="container">

<div id="subtitle" align="center">
  <h:form class="well">
    <legend>ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε.</legend>
  </h:form>
</div>

<div id="content" style="height:400px;width:600px;float:left;">
  <h:form>

    <h3>Προσθήκη καθηγητών</h3>
    <table>
    <tr>
    <td> <h:messages showDetail="true" /></td>
          </tr>
    <tr>
    <td><h:outputText value="΄Ονομα: "/></td>
    <td><h:inputText value="#{professor.pfname}"/></td>
    </tr>
    <tr>
    <td><h:outputText value="Επώνυμο : "/></td>
    <td><h:inputText value="#{professor.plname}"/></td>
```

```
    </tr>
    <tr>
    <td><h:outputText value="Πεδίο. : "/></td>
    <td><h:inputText value="#{professor.parea}"/></td>
    </tr>
    <tr>
    <td><h:outputText value="E-mail : "/></td>
    <td><h:inputText value="#{professor.pemail}"/></td>
    </tr>
    <tr>
    <td><h:outputText value="Μάθημα : "/></td>
    <td><h:selectOneMenu value="#{professor.selectedcourse}">
    <f:selectItems value="#{professor.coursess}"/>

            </h:selectOneMenu></td>
            </tr>
    <tr>
    <td><h:commandButton class="btn btn-primary"
                    value="Προσθήκη"
                    action="#{professor.addprof}"/></td>
    </tr>
    </table>
  </h:form>
</div>

<div id="professors" style="height:400px;width:850px;float:left;">
  <h:form>
  <h3>Διαθεσιμοι Καθηγητές</h3>
  <div style="overflow:scroll; width:550px; height:300px;">
  <h:dataTable value="#{proflist.getProfessorList()}" var="p" border="3"
headerClass="heading">
  <h:column>
  <f:facet name="header">
  'Ονομα
  </f:facet>
  #{p.pfname}
  </h:column>
  <h:column>
  <f:facet name="header">
  Επώνυμο
  </f:facet>
  #{p.plname}
  </h:column>
  <h:column>
  <f:facet name="header">
  Πεδίο
  </f:facet>
  #{p.parea}
  </h:column>
  <h:column>
  <f:facet name="header">
  E-mail
  </f:facet>
  #{p.pemail}
  </h:column>
  <h:column>
  <f:facet name="header">
  Username
```

```
  </f:facet>
  #{p.pid}
  </h:column>
  <h:column>
  <f:facet name="header">
  Password
  </f:facet>
  #{p.password}
  </h:column>

  </h:dataTable></div>

  </h:form>
 </div>

<div id="courses" style="width:600px;float:left;">
    <h:form>
    <h3>Προσθήκη Μαθήματος</h3>
    <table>
    <tr>
    <td> <h:messages showDetail="true" /></td>
    </tr>
    <tr>
    <td><h:outputText value="Τίτλος : "/></td>
    <td><h:inputText value="#{course.ctitle}"/></td>
    </tr>
    <tr>
    <td><h:outputText value="Τύπος : "/></td>
    <td><h:inputText value="#{course.ctheory}"/></td>
    </tr>
    <tr>
    <td><h:outputText value="Περιγραφή : "/></td>
    <td><h:inputTextarea value="#{course.cdescription}" rows="5" cols="35"/></td>
    </tr>
    <tr>
    <td><h:commandButton class="btn btn-primary"
                value="Προσθήκη"
                action="#{course.addcourse}"/></td>
    </tr>
    </table>
    </h:form>

 </div>


</div>
</h:body></html>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">


<h:head>
<title>Professor</title>
```

```
<link href="./bootstrap/css/bootstrap.css"
      rel="stylesheet" type="text/css"/>
<link href="./bootstrap/css/bootstrap-theme.css"
      rel="stylesheet"/>
<link href="./bootstrap/css/table-styles.css"
      rel="stylesheet"/>
</h:head>


<h:body>
<h1 align="center">Α.Τ.Ε.Ι ΗΠΕΙΡΟΥ</h1>

<div id="container">

<div id="subtitle" align="center">
  <h:form class="well">
    <legend>ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε.</legend>
  </h:form>
</div>



<div id="course" style="height:200px;width:700px;float:left;">
  <h:form>
     <h3>Εισαγωγή Ερωτήσεων:</h3><p></p>
     <h:panelGrid columns="2">
     Μάθημα:
     <h:selectOneMenu value="#{question.selectedcourse}">
       <f:selectItems value="#{question.coursess}"/>

     </h:selectOneMenu>
     Ερώτηση:
     <h:inputText value="#{question.qtitle}"/>
     Απάντηση 1:
     <h:inputText value="#{question.qanswer1}"/>
     Απάντηση 2:
     <h:inputText value="#{question.qanswer2}"/>
     Απάντηση 3:
     <h:inputText value="#{question.qanswer3}"/>
     Σωστή Απάντηση:
     <h:selectOneMenu value="#{question.qcorrectanswer}">
       <f:selectItem itemValue="1" itemLabel="1"/>
       <f:selectItem itemValue="2" itemLabel="2"/>
       <f:selectItem itemValue="3" itemLabel="3"/>
     </h:selectOneMenu>
     </h:panelGrid>
     <h:commandButton class="btn btn-primary"
                      value="Προσθήκη"
                      action="#{question.addquestion}"/>
  </h:form>
</div>

<div id="results" style="height:300px;width:550px;float:left;">
   <h:form>
   <h3>Αποτελέσματα εξετάσεων</h3><p></p>
   <h4>Στο Μάθημα:</h4>
   <h:selectOneMenu value="#{resultlist.selectedcourse}">
```

```
    <f:selectItems value="#{resultlist.coursess}"/>

    </h:selectOneMenu>
    <h:selectOneMenu value="#{resultlist.selectedstudent}">
    <f:selectItems value="#{resultlist.students}"/>

    </h:selectOneMenu>
    <h:commandButton class="btn btn-primary" value="Search"
  action="#{resultlist.getResultList()}"  />
    <div style="overflow:scroll; width:550px; height:300px;">
    <h:dataTable value="#{resultlist.getResultList()}" var="r" border="3" headerClass="heading">
    <h:column>
    <f:facet name="header">
    Τίτλος
    </f:facet>
    #{r.qtitle}
    </h:column>
    <h:column>
    <f:facet name="header">
    Σωστή Απάντηση
    </f:facet>
    #{r.qcorrectanswer}
    </h:column>
    <h:column>
    <f:facet name="header">
    Απάντηση
    </f:facet>
    #{r.sanswer}
    </h:column>
    <h:column>
    <f:facet name="header">
    ID Φοιτητή
    </f:facet>
    #{r.sid}
    </h:column>
    <h:column>
    <f:facet name="header">
    ID Μαθήματος
    </f:facet>
    #{r.cid}
    </h:column>
    </h:dataTable></div>
    </h:form>
  </div>

  <div id="questions" style="height:100px;width:500px;float:left;">
   <h:form>
    <h3>Ερωτήσεις:</h3><p></p>
    <h:selectOneMenu value="#{questionlist.selectedcourse}">
        <f:selectItems value="#{questionlist.coursess}"/>

    </h:selectOneMenu>
    <h:commandButton class="btn btn-primary" value="Search"
  action="#{questionlist.getQuestionList()}" />
    <div style="overflow:scroll; width:600px; height:200px;">
    <h:dataTable value="#{questionlist.getQuestionList()}" var="q" border="3"
  headerClass="heading">
    <h:column>
```

```
   <f:facet name="header">
   Qid
   </f:facet>
   #{q.qid}
   </h:column>
   <h:column>
   <f:facet name="header">
   Question
   </f:facet>
   #{q.qtitle}
   </h:column>
   <h:column>
   <f:facet name="header">
   Answer:1
   </f:facet>
   #{q.qanswer1}
   </h:column>
   <h:column>
   <f:facet name="header">
   Answer:2
   </f:facet>
   #{q.qanswer2}
   </h:column>
   <h:column>
   <f:facet name="header">
   Answer:3
   </f:facet>
   #{q.qanswer3}
   </h:column>
   <h:column>
   <f:facet name="header">
   Correct Answer
   </f:facet>
   #{q.qcorrectanswer}
   </h:column>
   <h:column>
   <h:commandButton class="btn btn-primary" value="Διαγραφή"
action="#{question.deletequestion(q.qid)}" />
   </h:column>


  </h:dataTable></div>

 </h:form>
</div>

</div>
</h:body></html>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">


<h:head>
```

```
<title>Student</title>
<link href="./bootstrap/css/bootstrap.css"
      rel="stylesheet" type="text/css"/>
</h:head>


<h:body>
<h1 align="center">A.T.E.I ΗΠΕΙΡΟΥ</h1>

<div id="container">

<div id="subtitle" align="center">
  <h:form class="well">
    <legend>ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε.</legend>
  </h:form>
</div>

<h:form>

<div id="Availabe" align="center" style="height:200px;width:200px;float:left;">
<b>Διαθέσιμα μαθήματα:</b><br/>
            <h:selectOneMenu value="#{exam.selectedcourse}">
       <f:selectItems value="#{exam.coursess}"/>

            </h:selectOneMenu>
</div>

<div id="startbutton" style="height:500px;width:500px;float:left;">
    <h:commandButton class="btn btn-primary" value="Έναρξη" action="exam" />
</div>
</h:form>


</div>
</h:body></html>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">

<h:head>
<title>A.T.E.I ΗΠΕΙΡΟΥ</title>

<link href="./bootstrap/css/bootstrap.css"
      rel="stylesheet" type="text/css"/>


</h:head>
<h:body>

<h1 align="center">A.T.E.I ΗΠΕΙΡΟΥ</h1>


<div id="container">
 <img src="logo.jpg"/>
 <div id="log-in" align="center">
```

```
 <h:form class="well"     >
 <legend>ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε.</legend>
 <table>
      <tr>
      <td><h:inputText value="#{login.username}"/><br/><p></p></td>
      </tr>
      <tr>
       <td><h:inputText value="#{login.lpassword}"/><br/><p></p></td>
      </tr>
 </table>
      <h:commandButton class="btn btn-primary"
                  value="Σύνδεση"
                  action="#{login.userlogin}"/>

 </h:form>
</div>

<div id="welcome-message" align="center">
    <h3>Καλώς ήρθες στο διαδικτυακό σύστημα εξέτασης του Τ.Ε.Ι ΗΠΕΙΡΟΥ </h3>
    <h:outputLink value="register-page.xhtml">Εγγραφή</h:outputLink>
</div>


</div>
<script src="./bootstrap/js/bootstrap.js"></script>
</h:body>
</html>
```