Τελικό Project προγράμματος Skills4Jobs

ΓΙΑΝΝΗΜΑΡΑΣ ΔΗΜΗΤΡΙΟΣ

Εισαγωγή

Το project είναι ένα full-stack webapp υλοποιημένο με αρχιτεκτονική MVC (Model-View-Controller).

Οι τεχνολογίες που χρησιμοποιήθηκαν είναι:

- Βάση Δεδομένω MySQL
- JavaEE με JSP και Servlets
- Html , Css , Bootstrap

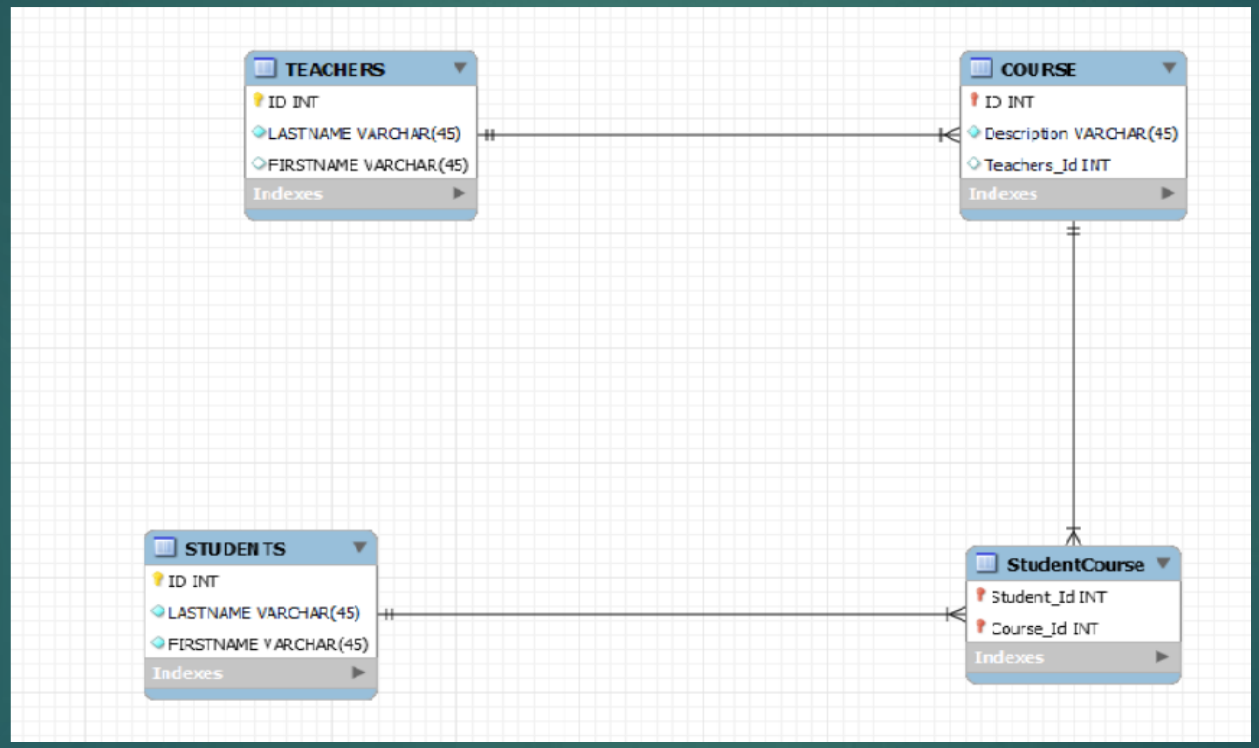
Σε επόμενες διαφάνειες θα δούμε πιο αναλυτικά το κάθε Layer και την υλοποίηση του.

Σκοπός της Εφαρμογής

Η ιδέα είναι πως το app απευθύνεται σε ένα χρήστη με δικαιώματα διαχειριστή στην γραμματεία του Οικονομικού Πανεπιστημίου Αθηνών.Αφού κάνει login θα μπορεί να κάνει CRUD ενέργειες

(Create-Read-Update-Delete) πάνω στους πίνακες και τα πεδία της ΒΔ μέσα από το webapp.Οι υπηρεσίες που προσφέρουμε απευθύνονται για τους πίνακες Καθηγητών,Σπουδαστών και Μαθημάτων.

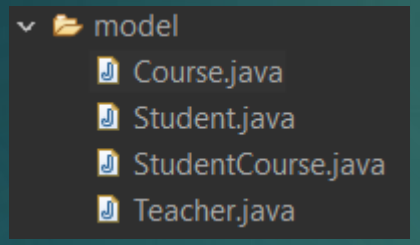
Βαση Δεδομένων
ER Diagram



Το διάγραμμα πάνω στο οποίο βασίστηκε η ΒΔ

Model

Το model είναι μία abstract αναπαράσταση της ΒΔ με κλάσεις.



Το package σε περιβάλλον Eclipse

Model

```java
package gr.aueb.cf.teachersapp.model;

public class Student {

    public int Id ;
    public String Firstname;
    public String Lastname ;

    public Student() {}


    public Student(int id, String firstname, String lastname) {
        super();
        Id = id;
        Firstname = firstname;
        Lastname = lastname;
    }
    public int getId() {
        return Id;
    }
    public void setId(int id) {
        Id = id;
    }
    public String getFirstname() {
        return Firstname;
    }
    public void setFirstname(String firstname) {
        Firstname = firstname;
    }
    public String getLastname() {
        return Lastname;
    }
    public void setLastname(String lastname) {
        Lastname = lastname;
    }


}
```

Η κλάση Student με πεδία ID,Firstame και Lastname, όπως και στη ΒΔ

DAO

Το DAO(Data Access Object) είναι το layer μεταξύ της ΒΔ και του Model.Ως παραμέτρους παίρνει όπου χρειάζεται model objects.Εδώ ορίζουμε public API και τις CRUD υπηρεσίες που παρέχουμε.

```java
package gr.aueb.cf.teachersapp.dao;

import java.sql.SQLException;


public interface ICourseDAO {

    void insert(Course course) throws SQLException;
    void update(Course oldCourse,Course newCourse) throws SQLException;
    void delete(Course course) throws SQLException;
    Course getCourse(String description) throws SQLException;
    List <Course> getAll() throws SQLException;

}
```

Πάνω βλέπουμε το Interface του ICourseDAO

DAO

Στην συνέχεια πρέπει να κάνουμε implement τις μεθόδους του ICourseDAO.Αυτό γίνεται στο CourseDAOImpl.Ας δούμε για παράδειγμα το Implementation της insert

```java
package gr.aueb.cf.teachersapp.dao;

import static gr.aueb.cf.teachersapp.dao.dbutil.DBUtil.closeConnection;

public class CourseDAOImpl implements ICourseDAO {

    @Override
    public void insert(Course course) throws SQLException {

        PreparedStatement pst = null;

        try {

            String sql = "INSERT INTO COURSES (Description, Teacher_Id) VALUES (?, ?)";

            pst = openConnection().prepareStatement(sql);
            pst.setString(1,  course.getDescription());
            pst.setInt(2,  course.getTeacherId());

             pst.executeUpdate();

        } catch (SQLException e) {
            e.printStackTrace();
            throw e;
        } finally {
            if (pst != null) pst.close();
            if (openConnection() != null) closeConnection();
        }

    }
```

Παρατηρούμε την επικοινωνία μεταξύ το DAO και της ΒΔ.Για να γίνει αυτό έχουμε την κλάση DBUtil την οποία θα δούμε παρακάτω

Java Database Connectivity

```java
package gr.aueb.cf.teachersapp.dao.dbutil;

import java.sql.Connection;

public class DBUtil {
    private static Connection conn;

    /**
     * No instances will be available
     */
    private DBUtil() {}

    public static Connection openConnection() throws SQLException {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");

            String url = "jdbc:mysql://localhost:3306/teachersDB?serverTime
            String username = "Dimitris2";
            String password = "1234";

            conn = DriverManager.getConnection(url, username, password);
            return conn;
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
            return null;

        }
    }


    public static void closeConnection() throws SQLException {
        conn.close();
    }
}
```

Εδώ καλούμε μη αυτόματα τον απαραίτητο Driver για JSBC καλώντας την Class.forName().
Η επικοινωνία γίνεται στη θύρα TCP/3306.
Παρατηρούμε την υλοποίηση επικοινωνίας με την teacherDb που είναι η ΒΔ στην MySQL καθώς και τα στοιχεία (username,password) του User που εμείς έχουμε φτιάξει στην MySQL

DTO

Τα DTO(Data Transfer Objects) είναι κλάσεις που χρησιμεύουν στη δημιουργία αντικειμένων μεταφοράς πληροφορίας από τον client στα services .

```java
public class CourseDTO {

    private int id;
    private String description;
    private int teacherId;

    public CourseDTO() {}

    public CourseDTO(int id, String description, int teacherId) {

        this.id = id;
        this.description = description;
        this.teacherId = teacherId;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public int getTeacherId() {
        return teacherId;
    }

    public void setTeacherId(int teacherId) {
        this.teacherId = teacherId;
```

Εδώ βλέπουμε για παράδειγμα την κλάση CourseDTO,την οποία θα χρησιμοποιήσουμε για μεταφορά πληροφορίας στο Service Layer παρακάτω

Service Layer

Το SOA (Service-Oriented Architecture) είναι ανεξάρτητο και αποτελείται από επιχειρησιακές υπηρεσίες που παρέχει η εφαρμογή μας στους clients.

Εδώ ορίζουμε το Public API. Το services API λαμβάνει DTOs ως παραμέτρους των μεθόδων. Ο controller θα διαβάζει τα πεδία από το input , θα φτιάχνει DTO και μετά θα καλεί το αντίστοιχο service περνώντας DTOs ως παραμέτρους.

Επίσης εδώ δημιουργούμε και το package των Exceptions.

Service Layer

```java
package gr.aueb.cf.teachersapp.service;

import java.sql.SQLException;

public interface ITeacherService {

    /**
     * Inserts a {@link Teacher} based on the data carried by the {@link TeacherDTO}.
     *
     * @param teacherDTO
     *          DTO object that contains the data
     *
     * @throws TeacherIdAlreadyExistsException
     *          if any teacher, identified by their id, needed to be inserted
     *          has already been inserted
     * @throws SQLException
     *          if any error happens during SQL insert
     */
    void insertTeacher(TeacherDTO teacherDTO)
            throws SQLException;



    /**
     * Removes a {@link Teacher} based on the data carried by the {@link TeacherDTO}.
     *
     * @param teacherDTO
     *          DTO object that contains the data (mainly the id)
     * @throws TeacherNotFoundException
     *          if any teacher, identified by their id, needed to be removed
     *          not found
     * @throws SQLException
     *          if any error happens during SQL delete
     */
    void deleteTeacher(TeacherDTO teacherDTO)
            throws TeacherNotFoundException, SQLException;
```

```java
/**
 * Updates a {@link Teacher} based on the data carried by the {@link TeacherDTO}.
 *
 * @param oldTeacherDTO
 *          DTO object hat contains the data -mainly the id- of the teacher
 *          that will be updated.
 * @param newTeacherDTO
 *          DTO object that contains the data of the new teacher.
 * @throws TeacherNotFoundException
 *          if any teacher, identified by their id, needed to be updated
 *          not found
 * @throws SQLException
 *          if any error happens during SQL update
 */
void updateTeacher(TeacherDTO oldTeacherDTO, TeacherDTO newTeacherDTO)
        throws SQLException;  // TeacherNotFoundException


/**
 * Gets back to the caller a list of the {@link Teacher} objects identified
 * by their lastname or lastaname's initial characters
 *
 * @param lastname
 *          a string object that contains the lastnane or the initial letters
 *          that lastname starts with.
 * @return
 *          a list that contains the results of the search, or null if no
 *          results are found
 * @throws SQLException
 *          if any error happens during SQL search
 */
List<Teacher> getTeacherByLastname(String lastname) throws SQLException;
```

Το interface του ITeacherService.
Παρατηρούμε δύο πράγματα.Όπως είπαμε παραπάνω οι μέθοδοι παίρνουν σαν παραμέτρους DTO objects .Επίσης,χρησιμοποιούμε doc commments καθώς εδώ ορίζουμε το Public API

Service Layer

Αφού λοιπόν ορίσαμε το Public API ,σειρά έχει η υλοποίηση του.

Ας δούμε για παράδειγμα την υλοποίηση της insertTeacher

```java
package gr.aueb.cf.teachersapp.service;

import java.sql.SQLException;

public class TeacherServiceImpl implements ITeacherService {

    private final ITeacherDAO teacherDAO;

    public TeacherServiceImpl(ITeacherDAO teacherDAO) {
        this.teacherDAO = teacherDAO;
    }

    @Override
    public void insertTeacher(TeacherDTO teacherDTO) throws SQLException {


        Teacher teacher = new Teacher();
        teacher.setSname(teacherDTO.getSname());
        teacher.setFname(teacherDTO.getFname());

        try {
            teacherDAO.insert(teacher);
        } catch (SQLException e) {
            throw e;
        }

    }
```

Βλέπουμε πως στον constructor κάνουμε Dependency Injection στο ITeacherDAO interface.
Πρόκειται για Composition and Forwarding Design Pattern.

Controller

Ο controller συνεργάζεται τόσο με το view όσο με το model. Πρόκειται για ενδιάμεση οντότητα που δρομολογεί (routing) τις αιτήσεις του χρήστη στην αντίστοιχη υπηρεσία.Για τον ρόλο του Controller χρησιμοποιήθηκαν Servlets, κλάσεις της Java που κάνουν extend το HttpServlet abstract class της Java EE και χειρίζονται την επικοινωνία με τον Web Browser.Ας δούμε στην επόμενη διαφάνεια ένα παράδειγμα.

Controller

```
3  import java.io.IOException;
24
25
26 @WebServlet("/DeleteCourseController")
27 public class DeleteCourseController extends HttpServlet {
28     private static final long serialVersionUID = 1L;
29
30     ICourseDAO courseDAO = new CourseDAOImpl();
31     ICourseService courseServ = new CourseServiceImpl(courseDAO);
32
33     protected void doGet(HttpServletRequest request, HttpServletResponse response)
34             throws ServletException, IOException {
35
36         response.setContentType("text/html");
37
38         int id = Integer.parseInt(request.getParameter("id").trim());
39         String description = request.getParameter("description").trim();
40         int teacherId = Integer.parseInt(request.getParameter("teacherid").trim());
41
42
43         CourseDTO courseDTO  = new CourseDTO ();
44         courseDTO .setId(id);
45         courseDTO .setDescription(description);
46         courseDTO .setTeacherId(teacherId);
47
48
49         try {
50             courseServ.deleteCourse(courseDTO);
51
52             request.setAttribute("course", courseDTO);
53             request.getRequestDispatcher("/jsps/coursedeleted.jsp").forward(request, response);
54         } catch (CourseNotFoundException e1) {
55             request.setAttribute("deleteAPIError", true);
56             request.getRequestDispatcher("/jsps/courses.jsp").forward(request, response);
57         } catch (SQLException e2) {
58             request.setAttribute("deleteAPIError", true);
59             request.getRequestDispatcher("/jsps/courses.jsp").forward(request, response);
```

Στην γραμμή 26 βλέπουμε το annotation που χρησιμοποιήσαμε για το mapping.Στο συγκεκριμένο παράδειγμα είναι ίδιο με το όνομα της κλάσης

Με setAttribute και getRequestDispatcher μεταβιβάζουμε στο coursedeleted.jsp το αντικείμενο course με τιμή courseDTO

View

Το view μας είναι html σε αρχεία jsp και με χρήση Expression Language.

Όπως είπαμε και πριν υπάρχει επικοινωνία view και controller,κάτι που ίσως γίνεται πιο σαφές στις παρακάτω εικόνες.

```
    href="${pageContext.request.contextPath}/DeleteCourseController?id=${course.id}
&description=${course.description}&teacherid=${course.teacherId}">Delete</a></td>
```

```
localhost:8080/teachers-webapp-jsp-mysql/DeleteCourseController?id=9&description=papapap&teacherid=10
```

View και Front-end

```jsp
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ page pageEncoding="UTF-8" contentType="text/html; charset=UTF-8"%>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet"
    href="${pageContext.request.contextPath}/static/css/afterLogin.css">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css
    <title>Welcome!</title>
</head>
<body >



<nav class="navbar navbar-inverse">
  <div class="container-fluid">
    <div class="navbar-header">
      <img
                    src="${pageContext.request.contextPath}/static/img/aueb100b.png"
                    alt="AUEB 100 years">
    </div>
    <ul class="nav navbar-nav">
      <li class="active"><a href="${pageContext.request.contextPath}/jsps/index.jsp">Αρχική</a></li>
      <li><a href="${pageContext.request.contextPath}/jsps/teachersmenu.jsp">Teachers Menu</a></li>
      <li><a href="${pageContext.request.contextPath}/jsps/studentsmenu.jsp">Students Menu</a></li>
      <li><a href="${pageContext.request.contextPath}/jsps/coursesmenu.jsp">Courses Menu</a></li>
    </ul>
  </div>
</nav>
```

Το afterLogin.jsp είναι η σελίδα που εμφανίζεται στον χρήστη μετά το login.Είναι ένα απλό page με σκοπό να επιλέξει ο χρήστης το menu που τον ενδιαφέρει στο navbar.Το navbar είναι προϊόν bootstrap.

Afterlogin Page

Γιατί MVC;

Προτιμήθηκε η συγκεκριμένη αρχιτεκτονική από μία μονολιθική καθώς έτσι ο κώδικας μας είναι πιο εύκολο να τεσταριστεί και να επεκταθεί,

Βιβλιογραφία-Πηγές

- Αθανάσιος Ανδρούτσος και οι αναλυτικές σημειώσεις του