# Додатки

# додаток а

Код Програми:

package beans;

import org.apache.log4j.Logger;

import javax.persistence.EntityManager;

import javax.persistence.PersistenceContext;

/\*\*

\* Created by win10 on 21.11.2016.

\*/

public class Bean<T> {

protected final static Logger logger = Logger.getLogger(Bean.class);

@PersistenceContext(unitName = "electronicJournal")

protected EntityManager entityManager;

public T add(T entity) {

return entityManager.merge(entity);

}

public void update(T entity) {

try {

entityManager.merge(entity);

} catch (Exception e) {

logger.error("An error occurred while updating an entity(" + entity + "). Error: " + e.getMessage());

throw e;

}

logger.info("Entity(" + entity + ") was updated.");

}

public void delete(T entity) {

entityManager.remove(entityManager.merge(entity));

logger.info("Entity(" + entity + ") was deleted.");

}

}

package beans;

import entities.ClassName;

import entities.School;

import javax.ejb.Stateless;

import javax.ejb.TransactionManagement;

import javax.ejb.TransactionManagementType;

import javax.persistence.NoResultException;

import javax.persistence.TypedQuery;

import java.util.List;

/\*\*

\* Created by win10 on 29.11.2016.

\*/

@Stateless

@TransactionManagement(value = TransactionManagementType.CONTAINER)

public class ClassBean extends Bean<ClassName> {

public List<ClassName> getAllBySchoolId(School school) {

TypedQuery<ClassName> query =

entityManager.createQuery("SELECT c FROM ClassName c WHERE c.school = :school", ClassName.class);

logger.info(getClass().getName() + ":: all classes with school (" + school + ") were received.");

return query.setParameter("school", school).getResultList();

}

public ClassName getByName(String className) {

TypedQuery<ClassName> query = entityManager.

createQuery("SELECT c FROM ClassName c WHERE c.name = :className", ClassName.class);

ClassName clazz = null;

try {

clazz = query.setParameter("className", className).getSingleResult();

logger.info(getClass().getName() + ":: class (name=" + className + ") was received. Entity: " + clazz);

} catch (NoResultException e) {

logger.warn("Class with name '" + className + "' does not exist!");

}

return clazz;

}

}

package beans;

import entities.ClassTeacher;

import entities.School;

import javax.ejb.Stateless;

import javax.ejb.TransactionManagement;

import javax.ejb.TransactionManagementType;

import javax.persistence.TypedQuery;

import java.util.List;

/\*\*

\* Created by win10 on 28.11.2016.

\*/

@Stateless

@TransactionManagement(value = TransactionManagementType.CONTAINER)

public class ClassTeacherBean extends Bean<ClassTeacher> {

public List<ClassTeacher> getAllBySchoolId(School school) {

TypedQuery<ClassTeacher> query =

entityManager.createQuery("SELECT ct FROM ClassTeacher ct WHERE ct.school = :school", ClassTeacher.class);

logger.info(getClass().getName() + ":: all class teachers with school (" + school + ") were received.");

return query.setParameter("school", school).getResultList();

}

public ClassTeacher getByLogin(String classTeacherLogin) {

TypedQuery<ClassTeacher> query = entityManager.

createQuery("SELECT ct FROM ClassTeacher ct WHERE ct.login = :login", ClassTeacher.class);

ClassTeacher classTeacher = query.setParameter("login", classTeacherLogin).getSingleResult();

logger.info(getClass().getName() + ":: class teacher (login=" + classTeacherLogin + ") was received. Entity: "

+ classTeacher);

return classTeacher;

}

}

package beans;

import entities.Director;

import javax.ejb.Stateless;

import javax.ejb.TransactionManagement;

import javax.ejb.TransactionManagementType;

import javax.persistence.TypedQuery;

/\*\*

\* Created by win10 on 23.11.2016.

\*/

@Stateless

@TransactionManagement(value = TransactionManagementType.CONTAINER)

public class DirectorBean extends Bean<Director> {

public Director getByLogin(String director) {

TypedQuery<Director> query = entityManager.

createQuery("SELECT d FROM Director d WHERE d.login = :person", Director.class);

Director director1 = query.setParameter("person", director).getSingleResult();

logger.info(getClass().getName() + ":: director (login=" + director + ") was received. Entity: " + director1);

return director1;

}

}

package beans;

import entities.Discipline;

import javax.ejb.Stateless;

import javax.ejb.TransactionManagement;

import javax.ejb.TransactionManagementType;

import javax.persistence.NoResultException;

import javax.persistence.TypedQuery;

/\*\*

\* Created by win10 on 28.11.2016.

\*/

@Stateless

@TransactionManagement(value = TransactionManagementType.CONTAINER)

public class DisciplineBean extends Bean<Discipline> {

public Discipline getByName(String discipline) {

TypedQuery<Discipline> query = entityManager.

createQuery("SELECT d FROM Discipline d WHERE d.name = :discipline", Discipline.class);

Discipline d = null;

try {

d = query.setParameter("discipline", discipline).getSingleResult();

logger.info(getClass().getName() + ":: discipline (name=" + discipline + ") was received. Entity: " + d);

} catch (NoResultException e) {

logger.warn("Discipline with name '" + discipline + "' does not exist!");

}

return d;

}

}

package beans;

import entities.ClassName;

import entities.DisciplineForClass;

import javax.ejb.Stateless;

import javax.ejb.TransactionManagement;

import javax.ejb.TransactionManagementType;

import javax.persistence.TypedQuery;

import java.util.List;

/\*\*

\* Created by win10 on 03.12.2016.

\*/

@Stateless

@TransactionManagement(value = TransactionManagementType.CONTAINER)

public class DisciplineForClassBean extends Bean<DisciplineForClass> {

public List<DisciplineForClass> getAllDisciplinesForClass(ClassName className) {

TypedQuery<DisciplineForClass> query = entityManager.

createQuery("SELECT dfc FROM DisciplineForClass dfc WHERE dfc.className = :className", DisciplineForClass.class);

logger.info(getClass().getName() + ":: all disciplines for class (" + className + ") were received.");

return query.setParameter("className", className).getResultList();

}

}

package beans;

import entities.ClassName;

import entities.Discipline;

import entities.Lesson;

import javax.ejb.Stateless;

import javax.ejb.TransactionManagement;

import javax.ejb.TransactionManagementType;

import javax.persistence.TypedQuery;

import java.util.List;

/\*\*

\* Created by win10 on 04.12.2016.

\*/

@Stateless

@TransactionManagement(value = TransactionManagementType.CONTAINER)

public class LessonBean extends Bean<Lesson> {

public List<Lesson> getAllByClassAndDiscipline(ClassName className, Discipline discipline) {

TypedQuery<Lesson> query =

entityManager.createQuery(

"SELECT l FROM Lesson l WHERE l.className = :className AND l.discipline = :discipline",

Lesson.class);

logger.info(getClass().getName() + ":: all lesson (" + className + ", " + discipline + ") was received.");

return query.setParameter("className", className).setParameter("discipline", discipline).getResultList();

}

}

package beans;

import entities.Lesson;

import entities.LessonResult;

import entities.Student;

import javax.ejb.Stateless;

import javax.ejb.TransactionManagement;

import javax.ejb.TransactionManagementType;

import javax.persistence.NoResultException;

import javax.persistence.TypedQuery;

/\*\*

\* Created by win10 on 04.12.2016.

\*/

@Stateless

@TransactionManagement(value = TransactionManagementType.CONTAINER)

public class LessonResultBean extends Bean<LessonResult> {

public LessonResult getByStudentAndLesson(Student student, Lesson lesson) {

TypedQuery<LessonResult> query =

entityManager.createQuery(

"SELECT lr FROM LessonResult lr WHERE lr.student = :student AND lr.lesson = :lesson",

LessonResult.class);

LessonResult lessonResult = null;

try {

lessonResult = query.setParameter("student", student).setParameter("lesson", lesson).getSingleResult();

logger.info(getClass().getName() + ":: lesson result(" + student + ", " + lesson +

") was received. Entity: " + lessonResult);

} catch (NoResultException e) {

logger.warn("Lesson result for " + student + " in lesson " + lesson + " does not exist!");

}

if (lessonResult == null) {

lessonResult = new LessonResult();

lessonResult.setStudent(student);

lessonResult.setLesson(lesson);

lessonResult.setActivity("");

lessonResult.setAdditional("");

lessonResult = add(lessonResult);

}

return lessonResult;

}

}

package beans;

import entities.Person;

import javax.ejb.Stateless;

import javax.ejb.TransactionManagement;

import javax.ejb.TransactionManagementType;

import javax.persistence.NoResultException;

import javax.persistence.TypedQuery;

/\*\*

\* Created by win10 on 06.12.2016.

\*/

@Stateless

@TransactionManagement(value = TransactionManagementType.CONTAINER)

public class PersonBean extends Bean<Person> {

public Person getByMail(String mail) {

TypedQuery<Person> query = entityManager.

createQuery("SELECT p FROM Person p WHERE p.mail = :mail", Person.class);

Person person = null;

try {

person = query.setParameter("mail", mail).getSingleResult();

logger.info(getClass().getName() + ":: person (e-mail=" + mail + ") was received. Entity: " + person);

} catch (NoResultException e) {

logger.warn("Person with e-mail '" + mail + "' does not exist!");

}

return person;

}

}

package beans;

import entities.School;

import javax.ejb.Stateless;

import javax.ejb.TransactionManagement;

import javax.ejb.TransactionManagementType;

/\*\*

\* Created by win10 on 27.11.2016.

\*/

@Stateless

@TransactionManagement(value = TransactionManagementType.CONTAINER)

public class SchoolBean extends Bean<School> {

}

package beans;

import entities.ClassName;

import entities.Student;

import javax.ejb.Stateless;

import javax.ejb.TransactionManagement;

import javax.ejb.TransactionManagementType;

import javax.persistence.TypedQuery;

import java.util.List;

/\*\*

\* Created by win10 on 29.11.2016.

\*/

@Stateless

@TransactionManagement(value = TransactionManagementType.CONTAINER)

public class StudentBean extends Bean<Student> {

public Student getByLogin(String student) {

TypedQuery<Student> query = entityManager.

createQuery("SELECT s FROM Student s WHERE s.login = :student", Student.class);

Student s = query.setParameter("student", student).getSingleResult();

logger.info(getClass().getName() + ":: student (login=" + student + ") was received. Entity: " + s);

return s;

}

public List<Student> getAllByClass(ClassName className) {

TypedQuery<Student> query =

entityManager.createQuery("SELECT s FROM Student s WHERE s.className = :className", Student.class);

logger.info(getClass().getName() + ":: all students (" + className + ") were received.");

return query.setParameter("className", className).getResultList();

}

}

package beans;

import entities.School;

import entities.Teacher;

import javax.ejb.Stateless;

import javax.ejb.TransactionManagement;

import javax.ejb.TransactionManagementType;

import javax.persistence.NoResultException;

import javax.persistence.TypedQuery;

import java.util.List;

/\*\*

\* Created by win10 on 28.11.2016.

\*/

@Stateless

@TransactionManagement(value = TransactionManagementType.CONTAINER)

public class TeacherBean extends Bean<Teacher> {

public List<Teacher> getAllBySchoolId(School school) {

TypedQuery<Teacher> query =

entityManager.createQuery("SELECT t FROM Teacher t WHERE t.school = :school", Teacher.class);

logger.info(getClass().getName() + ":: all teachers (" + school + ") were received.");

return query.setParameter("school", school).getResultList();

}

public Teacher getByLogin(String teacherLogin) {

TypedQuery<Teacher> query = entityManager.

createQuery("SELECT t FROM Teacher t WHERE t.login = :login", Teacher.class);

Teacher teacher = null;

try {

teacher = query.setParameter("login", teacherLogin).getSingleResult();

logger.info(getClass().getName() + ":: teacher (login=" + teacherLogin + ") was received. Entity: " + teacher);

} catch (NoResultException e) {

logger.warn("Teacher with login '" + teacherLogin + "' does not exist!");

}

return teacher;

}

}

package entities;

import javax.persistence.\*;

import java.io.Serializable;

import java.util.Set;

/\*\*

\* Created by win10 on 21.11.2016.

\*/

@Entity

@Table(name = "class")

public class ClassName implements Serializable {

@Id

@Column(name = "class\_id", unique = true, nullable = false)

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int classId;

@Column(name = "name", unique = true, nullable = false)

private String name;

@OneToOne(mappedBy = "className")

private ClassTeacher classTeacher;

@OneToMany(mappedBy = "className", fetch = FetchType.EAGER)

private Set<Student> students;

@OneToMany(mappedBy = "className", fetch = FetchType.EAGER)

private Set<Lesson> lessons;

@ManyToOne(cascade = CascadeType.MERGE)

@JoinColumn(name = "school\_id", nullable = false)

private School school;

public ClassName() {

}

public ClassName(String name, School school) {

this.name = name;

this.school = school;

}

public int getClassId() {

return classId;

}

public void setClassId(int classId) {

this.classId = classId;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public ClassTeacher getClassTeacher() {

return classTeacher;

}

public void setClassTeacher(ClassTeacher classTeacher) {

this.classTeacher = classTeacher;

}

public Set<Student> getStudents() {

return students;

}

public void setStudents(Set<Student> students) {

this.students = students;

}

public School getSchool() {

return school;

}

public void setSchool(School school) {

this.school = school;

}

public Set<Lesson> getLessons() {

return lessons;

}

public void setLessons(Set<Lesson> lessons) {

this.lessons = lessons;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

ClassName className = (ClassName) o;

if (getClassId() != className.getClassId()) return false;

return getName() != null ? getName().equals(className.getName()) : className.getName() == null;

}

@Override

public int hashCode() {

int result = getClassId();

result = 31 \* result + (getName() != null ? getName().hashCode() : 0);

return result;

}

@Override

public String toString() {

return "ClassName{" +

"classId=" + classId +

", name='" + name + '\'' +

'}';

}

}

package entities;

import javax.persistence.DiscriminatorValue;

import javax.persistence.Entity;

import javax.persistence.JoinColumn;

import javax.persistence.OneToOne;

import java.io.Serializable;

import java.security.NoSuchAlgorithmException;

import java.util.Date;

/\*\*

\* Created by win10 on 21.11.2016.

\*/

@Entity

@DiscriminatorValue("CLASS\_TEACHER")

public class ClassTeacher extends TeachingStaff implements Serializable {

@OneToOne()

@JoinColumn(name = "class\_id")

private ClassName className;

public ClassName getClassName() {

return className;

}

public void setClassName(ClassName className) {

this.className = className;

}

public ClassTeacher() {

}

public ClassTeacher(String name, String patronymic, String surname, Date birth, String phoneNumber, String address,

String login, String password, String mail, String additional, School school, String qualification,

ClassName className) throws NoSuchAlgorithmException {

super(name, patronymic, surname, birth, phoneNumber, address, login, password, mail, additional, school, qualification);

this.className = className;

}

public ClassTeacher(String name, String patronymic, String surname, Date birth, String phoneNumber, String address,

String login, String password, String mail, String additional,

String qualification, ClassName className) throws NoSuchAlgorithmException{

super(name, patronymic, surname, birth, phoneNumber, address, login, password, mail, additional, qualification);

this.className = className;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

if (!super.equals(o)) return false;

ClassTeacher that = (ClassTeacher) o;

return getClassName() != null ? getClassName().equals(that.getClassName()) : that.getClassName() == null;

}

@Override

public int hashCode() {

int result = super.hashCode();

result = 31 \* result + (getClassName() != null ? getClassName().hashCode() : 0);

return result;

}

@Override

public String toString() {

return "ClassTeacher{" +

"personId=" + getPersonId() +

", name='" + getName() + '\'' +

", patronymic='" + getPatronymic() + '\'' +

", surname='" + getSurname() + '\'' +

", birth=" + getBirth() +

", phoneNumber='" + getPhoneNumber() + '\'' +

", address='" + getAddress() + '\'' +

", login='" + getLogin() + '\'' +

", school=" + getSchool() +

", className=" + (className != null ? className.getName() : "none") +

", qualification='" + getQualification() + '\'' +

'}';

}

}

package entities;

import javax.persistence.DiscriminatorValue;

import javax.persistence.Entity;

import java.io.Serializable;

import java.security.NoSuchAlgorithmException;

import java.util.Date;

/\*\*

\* Created by win10 on 21.11.2016.

\*/

@Entity

@DiscriminatorValue("DIRECTOR")

public class Director extends TeachingStaff implements Serializable {

public Director() {

}

public Director(String name, String patronymic, String surname, Date birth, String phoneNumber, String address,

String login, String password, String mail, String additional,

School school, String qualification) throws NoSuchAlgorithmException {

super(name, patronymic, surname, birth, phoneNumber, address, login, password, mail, additional, school, qualification);

}

public Director(String name, String patronymic, String surname, Date birth, String phoneNumber, String address,

String login, String password, String mail,

String additional, String qualification) throws NoSuchAlgorithmException{

super(name, patronymic, surname, birth, phoneNumber, address, login, password, mail, additional, qualification);

}

@Override

public String toString() {

return "Director{" +

"personId=" + getPersonId() +

", name='" + getName() + '\'' +

", patronymic='" + getPatronymic() + '\'' +

", surname='" + getSurname() + '\'' +

", birth=" + getBirth() +

", phoneNumber='" + getPhoneNumber() + '\'' +

", address='" + getAddress() + '\'' +

", qualification=" + (getQualification() != null ? getQualification() : "null") +

", login='" + getLogin() + '\'' +

", school=" + getSchool() +

"}";

}

}

package entities;

import javax.persistence.\*;

import java.io.Serializable;

/\*\*

\* Created by win10 on 21.11.2016.

\*/

@Entity

@Table(name = "disciplines")

public class Discipline implements Serializable {

@Id

@Column(name = "discipline\_id", unique = true, nullable = false)

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int disciplineId;

@Column(name = "name", unique = true, nullable = false)

private String name;

public Discipline() {

}

public Discipline(String name) {

this.name = name;

}

public int getDisciplineId() {

return disciplineId;

}

public void setDisciplineId(int disciplineId) {

this.disciplineId = disciplineId;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Discipline that = (Discipline) o;

if (getDisciplineId() != that.getDisciplineId()) return false;

return getName() != null ? getName().equals(that.getName()) : that.getName() == null;

}

@Override

public int hashCode() {

int result = getDisciplineId();

result = 31 \* result + (getName() != null ? getName().hashCode() : 0);

return result;

}

@Override

public String toString() {

return "Discipline{" +

"disciplineId=" + disciplineId +

", name='" + name + '\'' +

'}';

}

}

package entities;

import javax.persistence.\*;

import java.io.Serializable;

import java.util.Date;

import java.util.List;

/\*\*

\* Created by win10 on 01.12.2016.

\*/

@Entity

@Table(name = "lesson")

public class Lesson implements Serializable {

@Id

@Column(name = "lesson\_id", unique = true, nullable = false)

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int lessonId;

@Column(name = "theme", nullable = false)

private String theme;

@Column(name = "date", nullable = false)

private Date date;

@ManyToOne(cascade = CascadeType.MERGE)

@JoinColumn(name = "discipline\_id", nullable = false)

private Discipline discipline;

@ManyToOne(cascade = CascadeType.MERGE)

@JoinColumn(name = "person\_id", nullable = false)

private Person teacher;

@ManyToOne(cascade = CascadeType.MERGE)

@JoinColumn(name = "class\_id", nullable = false)

private ClassName className;

@OneToMany(mappedBy = "lesson")

private List<LessonResult> lessonResults;

public int getLessonId() {

return lessonId;

}

public void setLessonId(int lessonId) {

this.lessonId = lessonId;

}

public String getTheme() {

return theme;

}

public void setTheme(String theme) {

this.theme = theme;

}

public Date getDate() {

return date;

}

public void setDate(Date date) {

this.date = date;

}

public Discipline getDiscipline() {

return discipline;

}

public void setDiscipline(Discipline discipline) {

this.discipline = discipline;

}

public Person getTeacher() {

return teacher;

}

public void setTeacher(Person teacher) {

this.teacher = teacher;

}

public ClassName getClassName() {

return className;

}

public void setClassName(ClassName className) {

this.className = className;

}

public List<LessonResult> getLessonResults() {

return lessonResults;

}

public void setLessonResults(List<LessonResult> lessonResults) {

this.lessonResults = lessonResults;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Lesson lesson = (Lesson) o;

if (getLessonId() != lesson.getLessonId()) return false;

if (getTheme() != null ? !getTheme().equals(lesson.getTheme()) : lesson.getTheme() != null) return false;

return getDate() != null ? getDate().equals(lesson.getDate()) : lesson.getDate() == null;

}

@Override

public int hashCode() {

int result = getLessonId();

result = 31 \* result + (getTheme() != null ? getTheme().hashCode() : 0);

result = 31 \* result + (getDate() != null ? getDate().hashCode() : 0);

return result;

}

@Override

public String toString() {

return "Lesson{" +

"lessonId=" + lessonId +

", theme='" + theme + '\'' +

", date=" + date +

", discipline=" + (discipline != null ? discipline.getName() : "-") +

", teacher=" + (teacher != null ? teacher.getName() + " " + teacher.getSurname() : "-") +

'}';

}

}

package entities;

import javax.persistence.\*;

import java.io.Serializable;

/\*\*

\* Created by win10 on 02.12.2016.

\*/

@Entity

@Table(name = "lesson\_result")

public class LessonResult implements Serializable {

@Id

@Column(name = "lr\_id", unique = true, nullable = false)

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int lrId;

@ManyToOne(cascade = CascadeType.MERGE)

@JoinColumn(name = "person\_id", nullable = false)

private Student student;

@Column(name = "additional")

private String additional;

@Column(name = "activity")

private String activity;

@ManyToOne(cascade = CascadeType.MERGE)

@JoinColumn(name = "lesson\_id")

private Lesson lesson;

public int getLrId() {

return lrId;

}

public void setLrId(int lrId) {

this.lrId = lrId;

}

public Student getStudent() {

return student;

}

public void setStudent(Student student) {

this.student = student;

}

public String getAdditional() {

return additional;

}

public void setAdditional(String additional) {

this.additional = additional;

}

public String getActivity() {

return activity;

}

public void setActivity(String activity) {

this.activity = activity;

}

public Lesson getLesson() {

return lesson;

}

public void setLesson(Lesson lesson) {

this.lesson = lesson;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

LessonResult that = (LessonResult) o;

if (getLrId() != that.getLrId()) return false;

if (getAdditional() != null ? !getAdditional().equals(that.getAdditional()) : that.getAdditional() != null)

return false;

return getActivity() != null ? getActivity().equals(that.getActivity()) : that.getActivity() == null;

}

@Override

public int hashCode() {

int result = getLrId();

result = 31 \* result + (getAdditional() != null ? getAdditional().hashCode() : 0);

result = 31 \* result + (getActivity() != null ? getActivity().hashCode() : 0);

return result;

}

@Override

public String toString() {

return "LessonResult{" +

"lrId=" + lrId +

", student=" + student.getName() + " " + student.getSurname() +

", additional='" + additional + '\'' +

", activity='" + activity + '\'' +

", lesson=" + lesson +

'}';

}

}

package entities;

import javax.persistence.\*;

import java.io.Serializable;

import java.util.Set;

/\*\*

\* Created by win10 on 21.11.2016.

\*/

@Entity

@Table(name = "school")

public class School implements Serializable {

@Id

@Column(name = "school\_id", unique = true, nullable = false)

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int schoolId;

@Column(name = "name")

private String name;

@Column(name = "phone")

private String phoneNumber;

@Column(name = "address")

private String address;

@Column(name = "information")

private String information;

@OneToOne(cascade = CascadeType.MERGE)

@JoinColumn(name = "director\_id", unique = true, nullable = false)

private Director director;

@OneToMany(mappedBy = "school", fetch = FetchType.EAGER)

private Set<ClassName> classNames;

@OneToMany(mappedBy = "school")

private Set<Person> persons;

public School() {

}

public School(String name, String phoneNumber, String address, String information, Director director) {

this.name = name;

this.phoneNumber = phoneNumber;

this.address = address;

this.information = information;

this.director = director;

}

public int getSchoolId() {

return schoolId;

}

public void setSchoolId(int schoolId) {

this.schoolId = schoolId;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPhoneNumber() {

return phoneNumber;

}

public void setPhoneNumber(String phoneNumber) {

this.phoneNumber = phoneNumber;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String getInformation() {

return information;

}

public void setInformation(String information) {

this.information = information;

}

public Director getDirector() {

return director;

}

public void setDirector(Director director) {

this.director = director;

}

public Set<ClassName> getClassNames() {

return classNames;

}

public void setClassNames(Set<ClassName> classNames) {

this.classNames = classNames;

}

public Set<Person> getPersons() {

return persons;

}

public void setPersons(Set<Person> persons) {

this.persons = persons;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

School school = (School) o;

if (getSchoolId() != school.getSchoolId()) return false;

if (getName() != null ? !getName().equals(school.getName()) : school.getName() != null) return false;

if (getPhoneNumber() != null ? !getPhoneNumber().equals(school.getPhoneNumber()) : school.getPhoneNumber() != null)

return false;

if (getAddress() != null ? !getAddress().equals(school.getAddress()) : school.getAddress() != null)

return false;

return getInformation() != null ? getInformation().equals(school.getInformation()) : school.getInformation() == null;

}

@Override

public int hashCode() {

int result = getSchoolId();

result = 31 \* result + (getName() != null ? getName().hashCode() : 0);

result = 31 \* result + (getPhoneNumber() != null ? getPhoneNumber().hashCode() : 0);

result = 31 \* result + (getAddress() != null ? getAddress().hashCode() : 0);

result = 31 \* result + (getInformation() != null ? getInformation().hashCode() : 0);

return result;

}

@Override

public String toString() {

return "School{" +

"schoolId=" + schoolId +

", name='" + name + '\'' +

", phoneNumber='" + phoneNumber + '\'' +

", address='" + address + '\'' +

", information='" + information + '\'' +

", director=" + director.getName() + ' ' + director.getSurname() +

'}';

}

}

package entities;

import javax.persistence.DiscriminatorValue;

import javax.persistence.Entity;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import java.io.Serializable;

import java.security.NoSuchAlgorithmException;

import java.util.Date;

/\*\*

\* Created by win10 on 21.11.2016.

\*/

@Entity

@DiscriminatorValue("STUDENT")

public class Student extends Person implements Serializable {

@ManyToOne()

@JoinColumn(name = "class\_id")

private ClassName className;

public Student() {

}

public Student(String name, String patronymic, String surname, Date birth, String phoneNumber, String address,

String login, String password, String mail, String additional, School school, ClassName className)

throws NoSuchAlgorithmException {

super(name, patronymic, surname, birth, phoneNumber, address, login, password, mail, additional, school);

this.className = className;

}

public ClassName getClassName() {

return className;

}

public void setClassName(ClassName className) {

this.className = className;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

if (!super.equals(o)) return false;

Student student = (Student) o;

return getClassName() != null ? getClassName().equals(student.getClassName()) : student.getClassName() == null;

}

@Override

public int hashCode() {

int result = super.hashCode();

result = 31 \* result + (getClassName() != null ? getClassName().hashCode() : 0);

return result;

}

@Override

public String toString() {

return "Student{" +

"personId=" + getPersonId() +

", name='" + getName() + '\'' +

", patronymic='" + getPatronymic() + '\'' +

", surname='" + getSurname() + '\'' +

", birth=" + getBirth() +

", phoneNumber='" + getPhoneNumber() + '\'' +

", address='" + getAddress() + '\'' +

", mail='" + getMail() + '\'' +

", additional='" + (getAdditional() != null ? getAdditional() : "-") + '\'' +

", login='" + getLogin() + '\'' +

", school=" + (getSchool() != null ? getSchool() : "none") +

", className=" + (className != null ? className.getName() : "none") +

'}';

}

}

package entities;

import javax.persistence.DiscriminatorValue;

import javax.persistence.Entity;

import java.security.NoSuchAlgorithmException;

import java.util.Date;

/\*\*

\* Created by win10 on 27.11.2016.

\*/

@Entity

@DiscriminatorValue("TEACHER")

public class Teacher extends TeachingStaff {

public Teacher() {

}

public Teacher(String name, String patronymic, String surname, Date birth, String phoneNumber, String address,

String login, String password, String mail, String additional, School school, String qualification) throws NoSuchAlgorithmException {

super(name, patronymic, surname, birth, phoneNumber, address, login, password, mail, additional, school, qualification);

}

public Teacher(String name, String patronymic, String surname, Date birth, String phoneNumber, String address,

String login, String password, String mail, String additional, String qualification) throws NoSuchAlgorithmException {

super(name, patronymic, surname, birth, phoneNumber, address, login, password, mail, additional, qualification);

}

@Override

public String toString() {

return "Teacher{" +

"personId=" + getPersonId() +

", name='" + getName() + '\'' +

", patronymic='" + getPatronymic() + '\'' +

", surname='" + getSurname() + '\'' +

", birth=" + getBirth() +

", phoneNumber='" + getPhoneNumber() + '\'' +

", address='" + getAddress() + '\'' +

", login='" + getLogin() + '\'' +

", qualification=" + getQualification() +

", school=" + getSchool() +

"}";

}

}

package entities;

import javax.persistence.\*;

import java.io.Serializable;

import java.security.NoSuchAlgorithmException;

import java.util.Date;

import java.util.Set;

/\*\*

\* Created by win10 on 21.11.2016.

\*/

@MappedSuperclass

public abstract class TeachingStaff extends Person implements Serializable {

@Column(name = "qualification")

private String qualification;

@ManyToMany(fetch = FetchType.EAGER)

@JoinTable(name = "teacher\_discipline",

joinColumns = {@JoinColumn(name = "person\_id", referencedColumnName = "person\_id")},

inverseJoinColumns = {@JoinColumn(name = "discipline\_id", referencedColumnName = "discipline\_id")})

private Set<Discipline> disciplines;

@OneToMany(mappedBy = "teacher", fetch = FetchType.EAGER)

private Set<DisciplineForClass> classes;

public TeachingStaff() {}

public TeachingStaff(String name, String patronymic, String surname, Date birth, String phoneNumber, String address,

String login, String password, String mail, String additional, School school, String qualification) throws NoSuchAlgorithmException {

super(name, patronymic, surname, birth, phoneNumber, address, login, password, mail, additional, school);

this.qualification = qualification;

}

public TeachingStaff(String name, String patronymic, String surname, Date birth, String phoneNumber, String address,

String login, String password, String mail, String additional, String qualification) throws NoSuchAlgorithmException {

super(name, patronymic, surname, birth, phoneNumber, address, login, password, mail, additional);

this.qualification = qualification;

}

public String getQualification() {

return qualification;

}

public void setQualification(String qualification) {

this.qualification = qualification;

}

public Set<Discipline> getDisciplines() {

return disciplines;

}

public void setDisciplines(Set<Discipline> disciplines) {

this.disciplines = disciplines;

}

public Set<DisciplineForClass> getClasses() {

return classes;

}

public void setClasses(Set<DisciplineForClass> classes) {

this.classes = classes;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (!(o instanceof TeachingStaff)) return false;

if (!super.equals(o)) return false;

TeachingStaff teacher = (TeachingStaff) o;

return getQualification() != null ? getQualification().equals(teacher.getQualification()) : teacher.getQualification() == null;

}

@Override

public int hashCode() {

int result = super.hashCode();

result = 31 \* result + (getQualification() != null ? getQualification().hashCode() : 0);

return result;

}

}

package managedBeans.classTeacher;

import beans.ClassBean;

import beans.ClassTeacherBean;

import beans.PersonBean;

import beans.StudentBean;

import entities.\*;

import org.apache.log4j.Logger;

import utils.Helper;

import javax.ejb.EJB;

import javax.faces.application.FacesMessage;

import javax.faces.bean.ManagedBean;

import javax.faces.bean.SessionScoped;

import javax.faces.component.UIComponent;

import javax.faces.component.UIInput;

import javax.faces.context.FacesContext;

import javax.faces.event.ComponentSystemEvent;

import java.security.NoSuchAlgorithmException;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import java.util.Date;

import java.util.Locale;

/\*\*

\* Created by win10 on 06.12.2016.

\*/

@ManagedBean(name = "classTeacherController", eager = true)

@SessionScoped

public class ClassTeacherController {

private final static Logger logger = Logger.getLogger(ClassTeacherController.class);

@EJB

private ClassTeacherBean classTeacherBean;

@EJB

private StudentBean studentBean;

@EJB

private ClassBean classBean;

@EJB

private PersonBean personBean;

private ClassTeacher classTeacher;

private School school;

private Director director;

private ClassName className;

private String birthStr;

private boolean addStudent;

private boolean editStudent;

private boolean dissolveClass;

private String name;

private String patronymic;

private String surname;

private Date birth;

private String phone;

private String mail;

private String address;

private String additionalInformation;

private Student student;

private void init() {

addStudent = false;

editStudent = false;

}

public String toClassPage() {

init();

return "/classTeacher/class?faces-redirect=true";

}

public String toHomePage() {

init();

return "/classTeacher/classTeacherIndex?faces-redirect=true";

}

public String cancelAddStudent() {

addStudent = false;

return "/classTeacher/class?faces-redirect=true";

}

public String addStudent() throws NoSuchAlgorithmException {

Student student = new Student(name, patronymic, surname, birth, phone, address, mail, "password", mail,

additionalInformation, school, className);

Student addedStudent = studentBean.add(student);

getClassName().getStudents().add(addedStudent);

classBean.update(getClassName());

addStudent = false;

return "/classTeacher/class?faces-redirect=true";

}

public void validateForm(ComponentSystemEvent event) {

FacesContext facesContext = FacesContext.getCurrentInstance();

UIComponent components = event.getComponent();

String patronymic = Helper.getInformation("patronymic", components);

String name = Helper.getInformation("name", components);

String surname = Helper.getInformation("surname", components);

String phone = Helper.getInformation("phone", components);

String mail = Helper.getInformation("mail", components);

String address = Helper.getInformation("address", components);

FacesMessage message = null;

if (name.isEmpty() || patronymic.isEmpty() || surname.isEmpty() || phone.isEmpty()

|| mail.isEmpty() || address.isEmpty()) {

message = new FacesMessage("Fill in all required fields!");

logger.warn("All required fields must be filled!");

} else {

boolean checkPhone = Helper.checkPhone(phone);

boolean checkMail = Helper.checkMail(mail);

if (!checkPhone || !checkMail) {

if (!checkPhone && !checkMail) {

message = new FacesMessage("Wrong phone number and e-mail format!");

logger.warn("Wrong phone number(" + phone + ") and e-mail(" + mail + ") format!");

((UIInput) components.findComponent("phone")).setValue("");

((UIInput) components.findComponent("mail")).setValue("");

} else if (!checkMail) {

message = new FacesMessage("Wrong e-mail format!");

logger.warn("Wrong e-mail(" + mail + ") format!");

((UIInput) components.findComponent("mail")).setValue("");

} else {

message = new FacesMessage("Wrong phone number format!");

logger.warn("Wrong phone number(" + phone + ") format!");

((UIInput) components.findComponent("phone")).setValue("");

}

}

}

Person person = personBean.getByMail(mail);

if (person != null) {

message = new FacesMessage("This email address is already taken!");

logger.warn("Email address (" + mail + ") is already taken!");

((UIInput) components.findComponent("mail")).setValue("");

}

if (message != null) {

message.setSeverity(FacesMessage.SEVERITY\_ERROR);

facesContext.addMessage(components.getClientId(), message);

facesContext.renderResponse();

}

}

public ClassTeacher getClassTeacher() {

classTeacher = classTeacher == null ? classTeacherBean.getByLogin(Helper.getCurrentUser()) : classTeacher;

return classTeacher;

}

public String getClassTeacherBirth() {

Calendar birth = Calendar.getInstance(Locale.ROOT);

birth.setTime(classTeacher.getBirth());

return birth.get(Calendar.DATE) + "-" + (birth.get(Calendar.MONTH) + 1) + "-" + birth.get(Calendar.YEAR);

}

public boolean isDisciplinesExist() {

return classTeacher.getDisciplines() != null && !classTeacher.getDisciplines().isEmpty();

}

public boolean isLessonsExist() {

return classTeacher.getClasses() != null && !classTeacher.getClasses().isEmpty();

}

public School getSchool() {

return school == null ? classTeacher.getSchool() : school;

}

public Director getDirector() {

return director == null ? getSchool().getDirector() : director;

}

public ClassName getClassName() {

return className == null ? classTeacher.getClassName() : className;

}

public String getDirectorBirth() {

Calendar date = Calendar.getInstance(Locale.ROOT);

date.setTime(getDirector().getBirth());

return date.get(Calendar.DATE) + "-" + (date.get(Calendar.MONTH) + 1) + "-" + date.get(Calendar.YEAR);

}

public String toSchoolPage() {

init();

return "/classTeacher/school?faces-redirect=true";

}

public String toAddStudent() {

init();

addStudent = true;

name = "";

patronymic = "";

surname = "";

birth = null;

phone = "";

mail = "";

address = "";

additionalInformation = "";

return "/classTeacher/addStudent";

}

public String studentDetails(Student student) {

init();

setStudent(student);

return "/classTeacher/studentDetails";

}

public String cancelSaveStudent() {

editStudent = false;

return "/classTeacher/studentDetails";

}

public String saveStudent() throws ParseException {

SimpleDateFormat simpleDateFormat = new SimpleDateFormat("dd-MM-yyyy");

student.setBirth(simpleDateFormat.parse(birthStr));

studentBean.update(student);

editStudent = false;

return "/classTeacher/studentDetails";

}

public String toEditStudent() {

init();

editStudent = true;

return "/classTeacher/studentDetails";

}

public boolean isDissolveClass() {

return getClassName().getStudents().isEmpty();

}

public String getBirthStr() {

Calendar birth = Calendar.getInstance(Locale.ROOT);

birth.setTime(student.getBirth());

return birth.get(Calendar.DATE) + "-" + (birth.get(Calendar.MONTH) + 1) + "-" + birth.get(Calendar.YEAR);

}

public Student getStudent() {

return student;

}

public void setStudent(Student student) {

this.student = student;

}

public String getAdditionalInformation() {

return additionalInformation;

}

public void setAdditionalInformation(String additionalInformation) {

this.additionalInformation = additionalInformation;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String getMail() {

return mail;

}

public void setMail(String mail) {

this.mail = mail;

}

public String getPhone() {

return phone;

}

public void setPhone(String phone) {

this.phone = phone;

}

public Date getBirth() {

return birth;

}

public void setBirth(Date birth) {

this.birth = birth;

}

public String getSurname() {

return surname;

}

public void setSurname(String surname) {

this.surname = surname;

}

public String getPatronymic() {

return patronymic;

}

public void setPatronymic(String patronymic) {

this.patronymic = patronymic;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public void setDissolveClass(boolean dissolveClass) {

this.dissolveClass = dissolveClass;

}

public boolean isEditStudent() {

return editStudent;

}

public void setEditStudent(boolean editStudent) {

this.editStudent = editStudent;

}

public boolean isAddStudent() {

return addStudent;

}

public void setAddStudent(boolean addStudent) {

this.addStudent = addStudent;

}

public void setBirthStr(String birthStr) {

this.birthStr = birthStr;

}

public void setClassName(ClassName className) {

this.className = className;

}

public void setDirector(Director director) {

this.director = director;

}

public void setSchool(School school) {

this.school = school;

}

public void setClassTeacher(ClassTeacher classTeacher) {

this.classTeacher = classTeacher;

}

}

package managedBeans.director;

import beans.\*;

import entities.\*;

import org.apache.log4j.Logger;

import utils.Helper;

import javax.ejb.EJB;

import javax.faces.application.FacesMessage;

import javax.faces.bean.ManagedBean;

import javax.faces.bean.ManagedProperty;

import javax.faces.bean.SessionScoped;

import javax.faces.component.UIComponent;

import javax.faces.component.UIInput;

import javax.faces.context.FacesContext;

import javax.faces.event.ComponentSystemEvent;

import java.security.NoSuchAlgorithmException;

import java.util.Date;

import java.util.List;

import java.util.Set;

/\*\*

\* Created by win10 on 29.11.2016.

\*/

@ManagedBean(name = "classBean", eager = true)

@SessionScoped

public class ClassesController {

private final static Logger logger = Logger.getLogger(ClassesController.class);

@EJB

private ClassBean classBean;

@EJB

private PersonBean personBean;

@EJB

private SchoolBean schoolBean;

@EJB

private TeacherBean teacherBean;

@EJB

private ClassTeacherBean classTeacherBean;

@ManagedProperty(value = "#{directorBean}")

private DirectorController directorController;

private boolean addNewClassTeacher;

private boolean addClassTeacher;

private boolean addExistingTeacher;

private boolean addClass;

private boolean editClass;

private boolean dissolveClass;

private String teacherLogin;

private String name;

private String patronymic;

private String surname;

private Date birth;

private String phone;

private String address;

private String mail;

private String additionalInformation;

private String qualification;

private ClassName className;

private School school;

private void init() {

addNewClassTeacher = false;

addClassTeacher = false;

addExistingTeacher = false;

addClass = false;

editClass = false;

}

public String toSendMessage() {

init();

return "/director/sendMessage?faces-redirect=true";

}

public String toTeachersPage() {

init();

return "/director/teachers/teachers?faces-redirect=true";

}

public String toClassesPage() {

init();

return "/director/classes/classes?faces-redirect=true";

}

public String toSchoolPage() {

init();

return "/director/school?faces-redirect=true";

}

public String toHomePage() {

init();

return "/director/directorIndex?faces-redirect=true";

}

public String toAddExistingTeacher() {

init();

addClassTeacher = true;

addExistingTeacher = true;

teacherLogin = "";

return "/director/classes/classDetails?faces-redirect=true";

}

public String addNewClassTeacher() throws NoSuchAlgorithmException {

ClassTeacher classTeacher = new ClassTeacher(name, patronymic, surname, birth, phone, address, mail,

"password", mail, additionalInformation, getSchool(), qualification, className);

className.setClassTeacher(classTeacherBean.add(classTeacher));

classBean.update(className);

addNewClassTeacher = false;

addClassTeacher = false;

return "/director/classes/classDetails?faces-redirect=true";

}

public String cancelAddNewClassTeacher() {

addNewClassTeacher = false;

addClassTeacher = false;

return "/director/classes/classDetails?faces-redirect=true";

}

public String cancelAddExistingTeacher() {

addExistingTeacher = false;

addClassTeacher = false;

return "/director/classes/classDetails?faces-redirect=true";

}

public String addExistingTeacher() throws NoSuchAlgorithmException {

Teacher teacher = teacherBean.getByLogin(teacherLogin);

ClassTeacher classTeacher = new ClassTeacher(teacher.getName(), teacher.getPatronymic(), teacher.getSurname(),

teacher.getBirth(), teacher.getPhoneNumber(), teacher.getAddress(), teacher.getLogin(), "password",

teacher.getMail(), teacher.getAdditional(), teacher.getSchool(), teacher.getQualification(), className );

classTeacher.setDisciplines(teacher.getDisciplines());

classTeacher.setClasses(teacher.getClasses());

teacherBean.delete(teacher);

className.setClassTeacher(classTeacher);

classTeacherBean.add(classTeacher);

addExistingTeacher = false;

addClassTeacher = false;

return "/director/classes/classDetails?faces-redirect=true";

}

public String toAddClassTeacher() {

init();

addClassTeacher = true;

return "/director/classes/classDetails?faces-redirect=true";

}

public boolean isClassTeacherExist() {

return getClassName().getClassTeacher() != null;

}

public boolean isDissolveClass() {

return getClassName().getStudents().isEmpty();

}

public Set<Student> getAllStudent() {

return getClassName().getStudents();

}

public String toEditClass() {

init();

editClass = true;

return "/director/classes/classDetails";

}

public String toAddNewClassTeacher() {

init();

addNewClassTeacher = true;

name = "";

patronymic = "";

surname = "";

birth = null;

phone = "";

address = "";

mail = "";

additionalInformation = "";

qualification = "";

return "/director/classes/addNewClassTeacher?faces-redirect=true";

}

public String cancelSaveClass() {

editClass = false;

return "/director/classes/classDetails";

}

public String saveClass() {

classBean.update(className);

editClass = false;

return "/director/classes/classDetails";

}

public String dissolveClass() throws NoSuchAlgorithmException {

ClassTeacher classTeacher = className.getClassTeacher();

if (classTeacher != null) {

Teacher teacher = new Teacher(classTeacher.getName(), classTeacher.getPatronymic(),

classTeacher.getSurname(), classTeacher.getBirth(), classTeacher.getPhoneNumber(),

classTeacher.getAddress(), classTeacher.getLogin(), "password", classTeacher.getMail(),

classTeacher.getAdditional(), classTeacher.getSchool(), classTeacher.getQualification());

teacher.setDisciplines(classTeacher.getDisciplines());

teacher.setClasses(classTeacher.getClasses());

classTeacherBean.delete(classTeacher);

teacherBean.add(teacher);

}

classBean.delete(className);

return "/director/classes/classes";

}

public String cancelAddClass() {

addClass = false;

return "/director/classes/classes";

}

public String addClass() {

ClassName className = new ClassName(name, getSchool());

classBean.add(className);

addClass = false;

return "/director/classes/classes";

}

public void validateForm(ComponentSystemEvent event) {

FacesContext facesContext = FacesContext.getCurrentInstance();

UIComponent components = event.getComponent();

String name = Helper.getInformation("name", components);

if (name.isEmpty()) {

FacesMessage message = new FacesMessage("Fill in 'name' fields!");

message.setSeverity(FacesMessage.SEVERITY\_ERROR);

logger.warn("Required field 'name' must be filled!");

facesContext.addMessage(components.getClientId(), message);

facesContext.renderResponse();

}

}

public void validateTeacherLogin(ComponentSystemEvent event) {

FacesContext facesContext = FacesContext.getCurrentInstance();

UIComponent components = event.getComponent();

String teacherLogin = Helper.getInformation("teacherLogin", components);

FacesMessage message = null;

if (teacherLogin.isEmpty()) {

message = new FacesMessage("Fill in 'login' fields!");

logger.warn("Required field 'login' must be filled!");

} else if (teacherBean.getByLogin(teacherLogin) == null) {

message = new FacesMessage("Teacher with login '" + teacherLogin + "' does not exist!");

logger.warn("Teacher with login '" + teacherLogin + "' does not exist!");

}

if (message != null) {

message.setSeverity(FacesMessage.SEVERITY\_ERROR);

facesContext.addMessage(components.getClientId(), message);

facesContext.renderResponse();

}

}

public void validateClassTeacher(ComponentSystemEvent event) {

FacesContext facesContext = FacesContext.getCurrentInstance();

UIComponent components = event.getComponent();

String name = Helper.getInformation("name", components);

String patronymic = Helper.getInformation("patronymic", components);

String surname = Helper.getInformation("surname", components);

String phone = Helper.getInformation("phone", components);

String mail = Helper.getInformation("mail", components);

String address = Helper.getInformation("address", components);

String qualification = Helper.getInformation("qualification", components);

FacesMessage message = null;

if (name.isEmpty() || patronymic.isEmpty() || surname.isEmpty() || phone.isEmpty()

|| mail.isEmpty() || address.isEmpty() || qualification.isEmpty()) {

message = new FacesMessage("Fill in all required fields!");

logger.warn("All required fields must be filled!");

} else {

boolean checkPhone = Helper.checkPhone(phone);

boolean checkMail = Helper.checkMail(mail);

if (!checkPhone || !checkMail) {

if (!checkPhone && !checkMail) {

message = new FacesMessage("Wrong phone number and e-mail format!");

logger.warn("Wrong phone number(" + phone + ") and e-mail(" + mail + ") format!");

((UIInput) components.findComponent("phone")).setValue("");

((UIInput) components.findComponent("mail")).setValue("");

} else if (!checkPhone) {

message = new FacesMessage("Wrong phone number format!");

logger.warn("Wrong phone number(" + phone + ") format!");

((UIInput) components.findComponent("phone")).setValue("");

} else {

message = new FacesMessage("Wrong e-mail format!");

logger.warn("Wrong e-mail(" + mail + ") format!");

((UIInput) components.findComponent("mail")).setValue("");

}

}

}

Person person = personBean.getByMail(mail);

if (person != null) {

message = new FacesMessage("This email address is already taken!");

logger.warn("Email address (" + mail + ") is already taken!");

((UIInput) components.findComponent("mail")).setValue("");

}

if (message != null) {

message.setSeverity(FacesMessage.SEVERITY\_ERROR);

facesContext.addMessage(components.getClientId(), message);

facesContext.renderResponse();

}

}

public String toAddClass() {

init();

addClass = true;

name = "";

return "/director/classes/classes";

}

public String classDetails(ClassName className) {

this.className = className;

return "/director/classes/classDetails?faces-redirect=true";

}

public List<ClassName> getAllClasses() {

return classBean.getAllBySchoolId(getSchool());

}

public boolean isClassesExist() {

List<ClassName> classNameList = classBean.getAllBySchoolId(getSchool());

return classNameList != null && !classNameList.isEmpty();

}

public School getSchool() {

school = school == null ? directorController.getDirector().getSchool() : school;

return school;

}

public DirectorController getDirectorController() {

return directorController;

}

public void setDirectorController(DirectorController directorController) {

this.directorController = directorController;

}

public boolean isAddNewClassTeacher() {

return addNewClassTeacher;

}

public void setAddNewClassTeacher(boolean addNewClassTeacher) {

this.addNewClassTeacher = addNewClassTeacher;

}

public boolean isAddClassTeacher() {

return addClassTeacher;

}

public void setAddClassTeacher(boolean addClassTeacher) {

this.addClassTeacher = addClassTeacher;

}

public boolean isAddExistingTeacher() {

return addExistingTeacher;

}

public void setAddExistingTeacher(boolean addExistingTeacher) {

this.addExistingTeacher = addExistingTeacher;

}

public boolean isAddClass() {

return addClass;

}

public void setAddClass(boolean addClass) {

this.addClass = addClass;

}

public boolean isEditClass() {

return editClass;

}

public void setEditClass(boolean editClass) {

this.editClass = editClass;

}

public void setDissolveClass(boolean dissolveClass) {

this.dissolveClass = dissolveClass;

}

public String getTeacherLogin() {

return teacherLogin;

}

public void setTeacherLogin(String teacherLogin) {

this.teacherLogin = teacherLogin;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPatronymic() {

return patronymic;

}

public void setPatronymic(String patronymic) {

this.patronymic = patronymic;

}

public String getSurname() {

return surname;

}

public void setSurname(String surname) {

this.surname = surname;

}

public Date getBirth() {

return birth;

}

public void setBirth(Date birth) {

this.birth = birth;

}

public String getPhone() {

return phone;

}

public void setPhone(String phone) {

this.phone = phone;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String getMail() {

return mail;

}

public void setMail(String mail) {

this.mail = mail;

}

public String getAdditionalInformation() {

return additionalInformation;

}

public void setAdditionalInformation(String additionalInformation) {

this.additionalInformation = additionalInformation;

}

public String getQualification() {

return qualification;

}

public void setQualification(String qualification) {

this.qualification = qualification;

}

public ClassName getClassName() {

return className;

}

public void setClassName(ClassName className) {

this.className = className;

}

public void setSchool(School school) {

this.school = school;

}

}

package managedBeans.director;

import beans.ClassBean;

import beans.DirectorBean;

import beans.DisciplineBean;

import beans.DisciplineForClassBean;

import entities.ClassName;

import entities.Director;

import entities.Discipline;

import entities.DisciplineForClass;

import org.apache.log4j.Logger;

import utils.Helper;

import javax.ejb.EJB;

import javax.faces.application.FacesMessage;

import javax.faces.bean.ManagedBean;

import javax.faces.bean.SessionScoped;

import javax.faces.component.UIComponent;

import javax.faces.context.FacesContext;

import javax.faces.event.ComponentSystemEvent;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import java.util.Locale;

/\*\*

\* Created by win10 on 25.11.2016.

\*/

@ManagedBean(name = "directorBean", eager = true)

@SessionScoped

public class DirectorController {

private final static Logger logger = Logger.getLogger(DirectorController.class);

@EJB

private DirectorBean directorBean;

@EJB

private DisciplineBean disciplineBean;

@EJB

private ClassBean classBean;

@EJB

private DisciplineForClassBean disciplineForClassBean;

private Director director;

private boolean editPersonalInformation;

private boolean editTeacherOpportunities;

private boolean addQualification;

private boolean addDiscipline;

private boolean addLesson;

private String birth;

private String qualification;

private String discipline;

private String className;

private void init() {

addLesson = false;

editPersonalInformation = false;

editTeacherOpportunities = false;

addQualification = false;

addDiscipline = false;

}

public String toSendMessage() {

init();

return "/director/sendMessage?faces-redirect=true";

}

public String toSchoolPage() {

init();

return "/director/school?faces-redirect=true";

}

public String toHomePage() {

init();

return "/director/directorIndex?faces-redirect=true";

}

public String toTeachersPage() {

init();

return "/director/teachers/teachers?faces-redirect=true";

}

public String toClassesPage() {

init();

return "/director/classes/classes?faces-redirect=true";

}

public String toAddDiscipline() {

init();

addDiscipline = true;

discipline = "";

return "/director/directorIndex";

}

public String cancelAddDiscipline() {

addDiscipline = false;

return "/director/directorIndex";

}

public String addDiscipline() {

Discipline disciplineForAdd = disciplineBean.getByName(discipline);

if (disciplineForAdd == null) {

disciplineForAdd = new Discipline(discipline);

disciplineForAdd = disciplineBean.add(disciplineForAdd);

}

director.getDisciplines().add(disciplineForAdd);

directorBean.update(director);

addDiscipline = false;

return "/director/directorIndex";

}

public String deleteDiscipline(Discipline discipline) {

director.getDisciplines().remove(discipline);

directorBean.update(director);

return "/director/directorIndex";

}

public void validateDiscipline(ComponentSystemEvent event) {

FacesContext facesContext = FacesContext.getCurrentInstance();

UIComponent components = event.getComponent();

String discipline = Helper.getInformation("discipline", components);

if (discipline.isEmpty()) {

FacesMessage message = new FacesMessage("Fill in 'discipline' field!");

logger.warn("Required field 'discipline' must be filled!");

message.setSeverity(FacesMessage.SEVERITY\_ERROR);

facesContext.addMessage(components.getClientId(), message);

facesContext.renderResponse();

}

}

public String toAddQualification() {

init();

addQualification = true;

qualification = "";

return "/director/directorIndex";

}

public String toAddLesson() {

init();

addLesson = true;

className = "";

discipline = "";

return "/director/directorIndex";

}

public void validateQualification(ComponentSystemEvent event) {

FacesContext facesContext = FacesContext.getCurrentInstance();

UIComponent components = event.getComponent();

String qualification = Helper.getInformation("qualification", components);

if (qualification.isEmpty()) {

FacesMessage message = new FacesMessage("Fill in 'qualification' field!");

logger.warn("Required field 'qualification' must be filled!");

message.setSeverity(FacesMessage.SEVERITY\_ERROR);

facesContext.addMessage(components.getClientId(), message);

facesContext.renderResponse();

}

}

public void validateLesson(ComponentSystemEvent event) {

FacesContext facesContext = FacesContext.getCurrentInstance();

UIComponent components = event.getComponent();

String discipline = Helper.getInformation("addDiscipline", components);

String className = Helper.getInformation("class", components);

FacesMessage message = null;

if (discipline.isEmpty() || className.isEmpty()) {

message = new FacesMessage("Fill in all required fields!");

logger.warn("All required fields must be filled!");

}

ClassName clazz = classBean.getByName(className);

if (clazz == null) {

message = new FacesMessage("Class with name '" + className + "' does not exist!");

logger.warn("Class with name '" + className + "' does not exist!");

}

if (message != null) {

message.setSeverity(FacesMessage.SEVERITY\_ERROR);

facesContext.addMessage(components.getClientId(), message);

facesContext.renderResponse();

}

}

public String addQualification() {

director.setQualification(qualification);

directorBean.update(director);

addQualification = false;

return "/director/directorIndex";

}

public String cancelAddQualification() {

addQualification = false;

return "/director/directorIndex";

}

public String cancelAddLesson() {

addLesson = false;

return "/director/directorIndex";

}

public String addLesson() {

Discipline disciplineForAdd = disciplineBean.getByName(discipline);

if (disciplineForAdd == null) {

disciplineForAdd = new Discipline(discipline);

Discipline addedDiscipline = disciplineBean.add(disciplineForAdd);

director.getDisciplines().add(addedDiscipline);

directorBean.update(director);

}

DisciplineForClass disciplineForClass = new DisciplineForClass();

disciplineForClass.setClassName(classBean.getByName(className));

disciplineForClass.setTeacher(director);

disciplineForClass.setDiscipline(disciplineBean.getByName(discipline));

DisciplineForClass forClass = disciplineForClassBean.add(disciplineForClass);

director.getClasses().add(forClass);

addLesson = false;

return "/director/directorIndex";

}

public String toChangeTeacherOpportunities() {

editTeacherOpportunities = true;

return "/director/directorIndex";

}

public String cancelSaveTeacherOpportunities() {

editTeacherOpportunities = false;

return "/director/directorIndex";

}

public String saveTeacherOpportunities() {

directorBean.update(director);

editTeacherOpportunities = false;

return "/director/directorIndex";

}

public Director getDirector() {

if (director == null) {

director = directorBean.getByLogin(Helper.getCurrentUser());

}

return director;

}

public String getBirth() {

Calendar birth = Calendar.getInstance(Locale.ROOT);

birth.setTime(director.getBirth());

return birth.get(Calendar.DATE) + "-" + (birth.get(Calendar.MONTH) + 1) + "-" + birth.get(Calendar.YEAR);

}

public String toChangePersonalInformation() {

editPersonalInformation = true;

return "/director/directorIndex";

}

public String cancelSavePersonalInformation() {

editPersonalInformation = false;

return "/director/directorIndex";

}

public String savePersonalInformation() throws ParseException {

SimpleDateFormat simpleDateFormat = new SimpleDateFormat("dd-MM-yyyy");

director.setBirth(simpleDateFormat.parse(birth));

directorBean.update(director);

editPersonalInformation = false;

return "/director/directorIndex";

}

public boolean isSchoolExist() {

return getDirector().getSchool() != null;

}

public boolean isQualificationExist() {

return director.getQualification() != null && !director.getQualification().isEmpty();

}

public boolean isLessonsExist() {

return director.getClasses() != null && !director.getClasses().isEmpty();

}

public boolean isDisciplinesExist() {

return director.getDisciplines() != null && !director.getDisciplines().isEmpty();

}

public void setDirector(Director director) {

this.director = director;

}

public boolean isEditPersonalInformation() {

return editPersonalInformation;

}

public void setEditPersonalInformation(boolean editPersonalInformation) {

this.editPersonalInformation = editPersonalInformation;

}

public boolean isEditTeacherOpportunities() {

return editTeacherOpportunities;

}

public void setEditTeacherOpportunities(boolean editTeacherOpportunities) {

this.editTeacherOpportunities = editTeacherOpportunities;

}

public boolean isAddQualification() {

return addQualification;

}

public void setAddQualification(boolean addQualification) {

this.addQualification = addQualification;

}

public boolean isAddDiscipline() {

return addDiscipline;

}

public void setAddDiscipline(boolean addDiscipline) {

this.addDiscipline = addDiscipline;

}

public boolean isAddLesson() {

return addLesson;

}

public void setAddLesson(boolean addLesson) {

this.addLesson = addLesson;

}

public void setBirth(String birth) {

this.birth = birth;

}

public String getQualification() {

return qualification;

}

public void setQualification(String qualification) {

this.qualification = qualification;

}

public String getDiscipline() {

return discipline;

}

public void setDiscipline(String discipline) {

this.discipline = discipline;

}

public String getClassName() {

return className;

}

public void setClassName(String className) {

this.className = className;

}

}

package managedBeans.director;

import beans.ClassBean;

import beans.LessonBean;

import beans.LessonResultBean;

import beans.StudentBean;

import entities.\*;

import org.apache.log4j.Logger;

import utils.Helper;

import javax.ejb.EJB;

import javax.faces.application.FacesMessage;

import javax.faces.bean.ManagedBean;

import javax.faces.bean.SessionScoped;

import javax.faces.component.UIComponent;

import javax.faces.context.FacesContext;

import javax.faces.event.ComponentSystemEvent;

import javax.servlet.http.HttpServletRequest;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.\*;

/\*\*

\* Created by win10 on 04.12.2016.

\*/

@ManagedBean(name = "disciplineBean", eager = true)

@SessionScoped

public class DisciplineController {

private final static Logger logger = Logger.getLogger(DisciplineController.class);

@EJB

private StudentBean studentBean;

@EJB

private ClassBean classBean;

@EJB

private LessonBean lessonBean;

@EJB

private LessonResultBean lessonResultBean;

private DisciplineForClass disciplineForClass;

private boolean addLesson;

private boolean editLessonResult;

private boolean editLesson;

private Date date;

private String theme;

private List<LessonResult> lessonResults;

private Student student;

private Lesson lesson;

private LessonResult lessonResult;

private List<LessonResult> lessonResultsForEdit = new ArrayList<>();

private String lessonDate;

private List<Lesson> lessons;

private void init() {

addLesson = false;

editLessonResult = false;

editLesson = false;

}

public String toSendMessage() {

init();

return "/director/sendMessage?faces-redirect=true";

}

public String toSchoolPage() {

init();

return "/director/school?faces-redirect=true";

}

public String toTeacherSchoolPage() {

init();

return "/teacher/school?faces-redirect=true";

}

public String toTeachersPage() {

init();

return "/director/teachers/teachers?faces-redirect=true";

}

public String toClassesPage() {

init();

return "/director/classes/classes?faces-redirect=true";

}

public String toHomePage() {

init();

return "/director/directorIndex?faces-redirect=true";

}

public String toTeacherHomePage() {

init();

return "/teacher/teacherIndex?faces-redirect=true";

}

public String toEditLesson() {

init();

setLessonResultsForEdit(new ArrayList<LessonResult>());

editLesson = true;

return "/discipline/lessonDetails";

}

public String toLessonDetails(Lesson lesson) {

init();

setLesson(lesson);

return "/discipline/lessonDetails";

}

public LessonResult receiveLessonResult(Lesson lesson) {

setLessonResult(lessonResultBean.getByStudentAndLesson(student, lesson));

return getLessonResult();

}

public LessonResult receiveLessonResult(Student student) {

setLessonResult(lessonResultBean.getByStudentAndLesson(student, lesson));

lessonResultsForEdit.add(getLessonResult());

return getLessonResult();

}

public String toEditLessonResult(Lesson lesson) {

init();

setLesson(lesson);

receiveLessonResult(lesson);

editLessonResult = true;

return "/discipline/studentDetails";

}

public String cancelEditLessonResult() {

editLessonResult = false;

return "/discipline/studentDetails";

}

public String editLessonResult() {

lessonResultBean.update(getLessonResult());

editLessonResult = false;

return "/discipline/studentDetails";

}

public String toStudentDetails(Student student) {

init();

setStudent(student);

return "/discipline/studentDetails";

}

public String toAddLesson() {

addLesson = true;

date = null;

theme = "";

lessonResults = new ArrayList<>();

List<Student> students = studentBean.getAllByClass(disciplineForClass.getClassName());

for (Student student : students) {

LessonResult lessonResult = new LessonResult();

lessonResult.setStudent(student);

lessonResults.add(lessonResult);

}

return "/discipline/addLesson?faces-redirect=true";

}

public String cancelEditLesson() {

editLesson = false;

return "/discipline/disciplineDetails?faces-redirect=true";

}

public String editLesson() throws ParseException {

for (LessonResult lessonResult : lessonResultsForEdit) {

lessonResultBean.update(lessonResult);

}

SimpleDateFormat simpleDateFormat = new SimpleDateFormat("dd-MM-yyyy");

lesson.setDate(simpleDateFormat.parse(lessonDate));

lessonBean.update(lesson);

editLesson = false;

return "/discipline/disciplineDetails?faces-redirect=true";

}

public String getLessonDate() {

Calendar birth = Calendar.getInstance(Locale.ROOT);

birth.setTime(lesson.getDate());

return birth.get(Calendar.DATE) + "-" + (birth.get(Calendar.MONTH) + 1) + "-" + birth.get(Calendar.YEAR);

}

public String addLesson() {

Lesson lesson = new Lesson();

lesson.setDiscipline(disciplineForClass.getDiscipline());

lesson.setTeacher(disciplineForClass.getTeacher());

lesson.setClassName(disciplineForClass.getClassName());

lesson.setDate(date);

lesson.setTheme(theme);

lesson.setLessonResults(lessonResults);

Lesson lesson1 = lessonBean.add(lesson);

for (LessonResult lessonResult : lessonResults) {

lessonResult.setLesson(lesson1);

lessonResultBean.add(lessonResult);

}

ClassName className = disciplineForClass.getClassName();

className.getLessons().add(lesson1);

classBean.update(className);

addLesson = false;

return "/discipline/disciplineDetails?faces-redirect=true";

}

public String cancelAddLesson() {

addLesson = false;

return "/discipline/disciplineDetails?faces-redirect=true";

}

public String disciplineForClassDetails(DisciplineForClass disciplineForClass) {

setDisciplineForClass(disciplineForClass);

return "/discipline/disciplineDetails?faces-redirect=true";

}

public List<Student> getAllStudents() {

return studentBean.getAllByClass(getDisciplineForClass().getClassName());

}

public List<Lesson> getAllLessons() {

lessons = lessonBean.getAllByClassAndDiscipline(disciplineForClass.getClassName(),

disciplineForClass.getDiscipline());

return lessons;

}

public String dateOfLesson(Lesson lesson) {

Calendar birth = Calendar.getInstance(Locale.ROOT);

birth.setTime(lesson.getDate());

return birth.get(Calendar.DATE) + "-" + (birth.get(Calendar.MONTH) + 1) + "-" + birth.get(Calendar.YEAR);

}

// public boolean isLessonExist() {

// Set<Lesson> lessons = getDisciplineForClass().getClassName().getLessons();

// return lessons != null && !lessons.isEmpty();

// }

public boolean isStudentExist() {

Set<Student> students = getDisciplineForClass().getClassName().getStudents();

return students != null && !students.isEmpty();

}

public void validateForm(ComponentSystemEvent event) {

FacesContext facesContext = FacesContext.getCurrentInstance();

UIComponent components = event.getComponent();

String theme = Helper.getInformation("theme", components);

if (theme.isEmpty()) {

FacesMessage message = new FacesMessage("Fill in 'theme' field!");

logger.warn("Required field 'theme' must be filled!");

message.setSeverity(FacesMessage.SEVERITY\_ERROR);

facesContext.addMessage(components.getClientId(), message);

facesContext.renderResponse();

}

}

public boolean isDirectorRole() {

return ((HttpServletRequest) FacesContext.getCurrentInstance().getExternalContext().getRequest())

.isUserInRole("DIRECTOR");

}

public boolean isTeacherRole() {

return ((HttpServletRequest) FacesContext.getCurrentInstance().getExternalContext().getRequest())

.isUserInRole("TEACHER");

}

public boolean isClassTeacherRole() {

return ((HttpServletRequest) FacesContext.getCurrentInstance().getExternalContext().getRequest())

.isUserInRole("CLASS\_TEACHER");

}

public DisciplineForClass getDisciplineForClass() {

return disciplineForClass;

}

public void setDisciplineForClass(DisciplineForClass disciplineForClass) {

this.disciplineForClass = disciplineForClass;

}

public boolean isAddLesson() {

return addLesson;

}

public void setAddLesson(boolean addLesson) {

this.addLesson = addLesson;

}

public boolean isEditLessonResult() {

return editLessonResult;

}

public void setEditLessonResult(boolean editLessonResult) {

this.editLessonResult = editLessonResult;

}

public boolean isEditLesson() {

return editLesson;

}

public void setEditLesson(boolean editLesson) {

this.editLesson = editLesson;

}

public Date getDate() {

return date;

}

public void setDate(Date date) {

this.date = date;

}

public String getTheme() {

return theme;

}

public void setTheme(String theme) {

this.theme = theme;

}

public List<LessonResult> getLessonResults() {

return lessonResults;

}

public void setLessonResults(List<LessonResult> lessonResults) {

this.lessonResults = lessonResults;

}

public Student getStudent() {

return student;

}

public void setStudent(Student student) {

this.student = student;

}

public Lesson getLesson() {

return lesson;

}

public void setLesson(Lesson lesson) {

this.lesson = lesson;

}

public LessonResult getLessonResult() {

return lessonResult;

}

public void setLessonResult(LessonResult lessonResult) {

this.lessonResult = lessonResult;

}

public List<LessonResult> getLessonResultsForEdit() {

return lessonResultsForEdit;

}

public void setLessonResultsForEdit(List<LessonResult> lessonResultsForEdit) {

this.lessonResultsForEdit = lessonResultsForEdit;

}

public void setLessonDate(String lessonDate) {

this.lessonDate = lessonDate;

}

public List<Lesson> getLessons() {

return lessons;

}

public void setLessons(List<Lesson> lessons) {

this.lessons = lessons;

}

}

package managedBeans.director;

import beans.DirectorBean;

import beans.SchoolBean;

import entities.Director;

import entities.School;

import org.apache.log4j.Logger;

import utils.Helper;

import javax.ejb.EJB;

import javax.faces.application.FacesMessage;

import javax.faces.bean.ManagedBean;

import javax.faces.bean.ManagedProperty;

import javax.faces.bean.SessionScoped;

import javax.faces.component.UIComponent;

import javax.faces.component.UIInput;

import javax.faces.context.FacesContext;

import javax.faces.event.ComponentSystemEvent;

/\*\*

\* Created by win10 on 27.11.2016.

\*/

@ManagedBean(name = "schoolBean", eager = true)

@SessionScoped

public class SchoolController {

private final static Logger logger = Logger.getLogger(SchoolController.class);

@EJB

private SchoolBean schoolBean;

@EJB

private DirectorBean directorBean;

@ManagedProperty(value = "#{directorBean}")

private DirectorController directorController;

private String name;

private String address;

private String phone;

private String information;

private School school;

private boolean addSchool;

private boolean editSchool;

private void init() {

addSchool = false;

editSchool = false;

}

public String toSendMessage() {

init();

return "/director/sendMessage?faces-redirect=true";

}

public String toTeachersPage() {

init();

return "/director/teachers/teachers?faces-redirect=true";

}

public String toClassesPage() {

init();

return "/director/classes/classes?faces-redirect=true";

}

public String toHomePage() {

init();

return "/director/directorIndex?faces-redirect=true";

}

public String toAddSchool() {

addSchool = true;

name = "";

address = "";

phone = "";

information = "";

return "/director/school?faces-redirect=true";

}

public String cancelCreateSchool() {

addSchool = false;

return "/director/school?faces-redirect=true";

}

public String cancelEditSchool() {

editSchool = false;

return "/director/school?faces-redirect=true";

}

public String createSchool() {

Director director = directorController.getDirector();

School school = new School(name, phone, address, information, director);

School createdSchool = schoolBean.add(school);

director.setSchool(createdSchool);

directorBean.update(director);

return "/director/school?faces-redirect=true";

}

public String toEditSchool() {

editSchool = true;

return "/director/school?faces-redirect=true";

}

public String editSchoolInformation() {

schoolBean.update(school);

editSchool = false;

return "/director/school?faces-redirect=true";

}

public void validateForm(ComponentSystemEvent event) {

FacesContext facesContext = FacesContext.getCurrentInstance();

UIComponent components = event.getComponent();

String name = Helper.getInformation("name", components);

String address = Helper.getInformation("address", components);

String phone = Helper.getInformation("phone", components);

String information = Helper.getInformation("information", components);

FacesMessage message = null;

if (name.isEmpty() || address.isEmpty() || phone.isEmpty() || information.isEmpty()) {

message = new FacesMessage("Fill in all required fields!");

logger.warn("All required fields must be filled!");

} else if (!Helper.checkPhone(phone)) {

message = new FacesMessage("Wrong phone number format!");

logger.warn("Wrong phone number(" + phone + ") format!");

((UIInput) components.findComponent("phone")).setValue("");

}

if (message != null) {

message.setSeverity(FacesMessage.SEVERITY\_ERROR);

facesContext.addMessage(components.getClientId(), message);

facesContext.renderResponse();

}

}

public School getSchool() {

school = school == null ? directorController.getDirector().getSchool() : school;

return school;

}

public DirectorController getDirectorController() {

return directorController;

}

public void setDirectorController(DirectorController directorController) {

this.directorController = directorController;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String getPhone() {

return phone;

}

public void setPhone(String phone) {

this.phone = phone;

}

public String getInformation() {

return information;

}

public void setInformation(String information) {

this.information = information;

}

public void setSchool(School school) {

this.school = school;

}

public boolean isAddSchool() {

return addSchool;

}

public void setAddSchool(boolean addSchool) {

this.addSchool = addSchool;

}

public boolean isEditSchool() {

return editSchool;

}

public void setEditSchool(boolean editSchool) {

this.editSchool = editSchool;

}

}

package managedBeans.director;

import beans.ClassBean;

import beans.PersonBean;

import beans.StudentBean;

import entities.ClassName;

import entities.Person;

import entities.School;

import entities.Student;

import org.apache.log4j.Logger;

import utils.Helper;

import javax.ejb.EJB;

import javax.faces.application.FacesMessage;

import javax.faces.bean.ManagedBean;

import javax.faces.bean.ManagedProperty;

import javax.faces.bean.SessionScoped;

import javax.faces.component.UIComponent;

import javax.faces.component.UIInput;

import javax.faces.context.FacesContext;

import javax.faces.event.ComponentSystemEvent;

import java.security.NoSuchAlgorithmException;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import java.util.Date;

import java.util.Locale;

/\*\*

\* Created by win10 on 29.11.2016.

\*/

@ManagedBean(name = "studentBean", eager = true)

@SessionScoped

public class StudentController {

private final static Logger logger = Logger.getLogger(StudentController.class);

@EJB

private StudentBean studentBean;

@EJB

private ClassBean classBean;

@EJB

private PersonBean personBean;

@ManagedProperty(value = "#{directorBean}")

private DirectorController directorController;

private String name;

private String patronymic;

private String surname;

private Date birth;

private String phone;

private String address;

private String mail;

private String additionalInformation;

private ClassName className;

private String birthStr;

private boolean addStudent;

private boolean editStudent;

private Student student;

private School school;

private void init() {

addStudent = false;

editStudent = false;

}

public String toTeachersPage() {

init();

return "/director/teachers/teachers?faces-redirect=true";

}

public String toClassesPage() {

init();

return "/director/classes/classes?faces-redirect=true";

}

public String toSchoolPage() {

init();

return "/director/school?faces-redirect=true";

}

public String toHomePage() {

init();

return "/director/directorIndex?faces-redirect=true";

}

public String toTransferStudent() {

init();

//// TODO: transfer student

return "/director/classes/studentDetails";

}

public String excludeStudent() {

ClassName tmp = student.getClassName();

student.setClassName(null);

student.setSchool(null);

tmp.getStudents().remove(student);

studentBean.update(student);

return "/director/classes/classDetails?faces-redirect=true";

}

public String toEditStudent() {

init();

editStudent = true;

return "/director/classes/studentDetails";

}

public String cancelSaveStudent() {

editStudent = false;

return "/director/classes/studentDetails";

}

public String saveStudent() throws ParseException {

SimpleDateFormat simpleDateFormat = new SimpleDateFormat("dd-MM-yyyy");

student.setBirth(simpleDateFormat.parse(birthStr));

studentBean.update(student);

editStudent = false;

return "/director/classes/studentDetails";

}

public String getBirthStr() {

Calendar birth = Calendar.getInstance(Locale.ROOT);

birth.setTime(getStudent().getBirth());

return birth.get(Calendar.DATE) + "-" + (birth.get(Calendar.MONTH) + 1) + "-" + birth.get(Calendar.YEAR);

}

public String studentDetails(Student student) {

init();

setStudent(student);

return "/director/classes/studentDetails";

}

public String toAddStudent(ClassName className) {

init();

addStudent = true;

setClassName(className);

name = "";

patronymic = "";

surname = "";

birth = null;

phone = "";

mail = "";

address = "";

additionalInformation = "";

return "/director/classes/addStudent";

}

public String cancelAddStudent() {

addStudent = false;

return "/director/classes/classDetails";

}

public String addStudent() throws NoSuchAlgorithmException {

Student student = new Student(name, patronymic, surname, birth, phone, address, mail, "password", mail,

additionalInformation, school, className);

className.getStudents().add(studentBean.add(student));

classBean.update(className);

addStudent = false;

return "/director/classes/classDetails";

}

public School getSchool() {

school = school == null ? directorController.getDirector().getSchool() : school;

return school;

}

public void validateForm(ComponentSystemEvent event) {

FacesContext facesContext = FacesContext.getCurrentInstance();

UIComponent components = event.getComponent();

String name = Helper.getInformation("name", components);

String patronymic = Helper.getInformation("patronymic", components);

String surname = Helper.getInformation("surname", components);

String phone = Helper.getInformation("phone", components);

String mail = Helper.getInformation("mail", components);

String address = Helper.getInformation("address", components);

FacesMessage message = null;

if (name.isEmpty() || patronymic.isEmpty() || surname.isEmpty() || phone.isEmpty()

|| mail.isEmpty() || address.isEmpty()) {

message = new FacesMessage("Fill in all required fields!");

logger.warn("All required fields must be filled!");

} else {

boolean checkPhone = Helper.checkPhone(phone);

boolean checkMail = Helper.checkMail(mail);

if (!checkPhone || !checkMail) {

if (!checkPhone && !checkMail) {

message = new FacesMessage("Wrong phone number and e-mail format!");

logger.warn("Wrong phone number(" + phone + ") and e-mail(" + mail + ") format!");

((UIInput) components.findComponent("phone")).setValue("");

((UIInput) components.findComponent("mail")).setValue("");

} else if (!checkPhone) {

message = new FacesMessage("Wrong phone number!");

logger.warn("Wrong phone number(" + phone + ") format!");

((UIInput) components.findComponent("phone")).setValue("");

} else {

message = new FacesMessage("Wrong e-mail format!");

logger.warn("Wrong e-mail(" + mail + ") format!");

((UIInput) components.findComponent("mail")).setValue("");

}

}

}

Person person = personBean.getByMail(mail);

if (person != null) {

message = new FacesMessage("This email address is already taken!");

logger.warn("Email address (" + mail + ") is already taken!");

((UIInput) components.findComponent("mail")).setValue("");

}

if (message != null) {

message.setSeverity(FacesMessage.SEVERITY\_ERROR);

facesContext.addMessage(components.getClientId(), message);

facesContext.renderResponse();

}

}

public DirectorController getDirectorController() {

return directorController;

}

public void setDirectorController(DirectorController directorController) {

this.directorController = directorController;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPatronymic() {

return patronymic;

}

public void setPatronymic(String patronymic) {

this.patronymic = patronymic;

}

public String getSurname() {

return surname;

}

public void setSurname(String surname) {

this.surname = surname;

}

public Date getBirth() {

return birth;

}

public void setBirth(Date birth) {

this.birth = birth;

}

public String getPhone() {

return phone;

}

public void setPhone(String phone) {

this.phone = phone;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String getMail() {

return mail;

}

public void setMail(String mail) {

this.mail = mail;

}

public String getAdditionalInformation() {

return additionalInformation;

}

public void setAdditionalInformation(String additionalInformation) {

this.additionalInformation = additionalInformation;

}

public ClassName getClassName() {

return className;

}

public void setClassName(ClassName className) {

this.className = className;

}

public void setBirthStr(String birthStr) {

this.birthStr = birthStr;

}

public boolean isAddStudent() {

return addStudent;

}

public void setAddStudent(boolean addStudent) {

this.addStudent = addStudent;

}

public boolean isEditStudent() {

return editStudent;

}

public void setEditStudent(boolean editStudent) {

this.editStudent = editStudent;

}

public Student getStudent() {

return student;

}

public void setStudent(Student student) {

this.student = student;

}

public void setSchool(School school) {

this.school = school;

}

}

package managedBeans.director;

import beans.\*;

import entities.\*;

import org.apache.log4j.Logger;

import utils.Helper;

import javax.ejb.EJB;

import javax.faces.application.FacesMessage;

import javax.faces.bean.ManagedBean;

import javax.faces.bean.ManagedProperty;

import javax.faces.bean.SessionScoped;

import javax.faces.component.UIComponent;

import javax.faces.component.UIInput;

import javax.faces.context.FacesContext;

import javax.faces.event.ComponentSystemEvent;

import java.security.NoSuchAlgorithmException;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import java.util.Date;

import java.util.List;

import java.util.Locale;

/\*\*

\* Created by win10 on 28.11.2016.

\*/

@ManagedBean(name = "teachersBean", eager = true)

@SessionScoped

public class TeachersController {

private final static Logger logger = Logger.getLogger(TeachersController.class);

@EJB

private TeacherBean teacherBean;

@EJB

private DisciplineForClassBean disciplineForClassBean;

@EJB

private DisciplineBean disciplineBean;

@EJB

private PersonBean personBean;

@EJB

private ClassTeacherBean classTeacherBean;

@EJB

private ClassBean classBean;

@ManagedProperty(value = "#{directorBean}")

private DirectorController directorController;

private String name;

private String patronymic;

private String surname;

private Date birth;

private String phone;

private String address;

private String mail;

private String additionalInformation;

private String qualification;

private String className;

private String discipline;

private String teacherBirthStr;

private String classTeacherBirthStr;

private Teacher teacher;

private ClassTeacher classTeacher;

private boolean editTeacher;

private boolean editClassTeacher;

private boolean addTeacher;

private boolean addClassTeacher;

private boolean addDisciplineForTeacher;

private boolean addDisciplineForClassTeacher;

private boolean addLessonForTeacher;

private boolean addLessonForClassTeacher;

private School school;

private void init() {

addTeacher = false;

addClassTeacher = false;

editTeacher = false;

editClassTeacher = false;

addDisciplineForTeacher = false;

addDisciplineForClassTeacher = false;

addLessonForTeacher = false;

addLessonForClassTeacher = false;

}

public String toTeachersPage() {

init();

return "/director/teachers/teachers?faces-redirect=true";

}

public String toClassesPage() {

init();

return "/director/classes/classes?faces-redirect=true";

}

public String toSchoolPage() {

init();

return "/director/school?faces-redirect=true";

}

public String toHomePage() {

init();

return "/director/directorIndex?faces-redirect=true";

}

public String toAddTeacher() {

init();

addTeacher = true;

reset();

return "/director/teachers/addTeachers";

}

public String toAddClassTeacher() {

init();

addClassTeacher = true;

reset();

className = "";

return "/director/teachers/addTeachers";

}

private void reset() {

name = "";

patronymic = "";

surname = "";

birth = null;

phone = "";

mail = "";

address = "";

additionalInformation = "";

qualification = "";

}

public String toEditTeacher() {

init();

editTeacher = true;

return "/director/teachers/teacherDetails";

}

public String toEditClassTeacher() {

init();

editClassTeacher = true;

return "/director/teachers/classTeacherDetails";

}

public boolean isTeacherExist() {

List<Teacher> teachers = teacherBean.getAllBySchoolId(getSchool());

return teachers != null && !teachers.isEmpty();

}

public boolean isClassTeacherExist() {

List<ClassTeacher> teachers = classTeacherBean.getAllBySchoolId(getSchool());

return teachers != null && !teachers.isEmpty();

}

public void validateForm(ComponentSystemEvent event) {

FacesContext facesContext = FacesContext.getCurrentInstance();

UIComponent components = event.getComponent();

String name = Helper.getInformation("teacherName", components);

String patronymic = Helper.getInformation("teacherPatronymic", components);

String surname = Helper.getInformation("teacherSurname", components);

String phone = Helper.getInformation("teacherPhone", components);

String mail = Helper.getInformation("teacherMail", components);

String address = Helper.getInformation("teacherAddress", components);

String qualification = Helper.getInformation("teacherQualification", components);

FacesMessage message = null;

if (name.isEmpty() || patronymic.isEmpty() || surname.isEmpty() || phone.isEmpty()

|| mail.isEmpty() || address.isEmpty() || qualification.isEmpty()) {

message = new FacesMessage("Fill in all required fields!");

logger.warn("All required fields must be filled!");

} else {

boolean checkPhone = Helper.checkPhone(phone);

boolean checkMail = Helper.checkMail(mail);

if (!checkPhone || !checkMail) {

if (!checkPhone && !checkMail) {

message = new FacesMessage("Wrong phone number and e-mail format!");

logger.warn("Wrong phone number(" + phone + ") and e-mail(" + mail + ") format!");

((UIInput) components.findComponent("teacherPhone")).setValue("");

((UIInput) components.findComponent("teacherMail")).setValue("");

} else if (!checkPhone) {

message = new FacesMessage("Wrong phone number format!");

logger.warn("Wrong phone number(" + phone + ") format!");

((UIInput) components.findComponent("teacherPhone")).setValue("");

} else {

message = new FacesMessage("Wrong e-mail format!");

logger.warn("Wrong e-mail(" + mail + ") format!");

((UIInput) components.findComponent("teacherMail")).setValue("");

}

}

}

Person person = personBean.getByMail(mail);

if (person != null) {

message = new FacesMessage("This email address is already taken!");

logger.warn("Email address (" + mail + ") is already taken!");

((UIInput) components.findComponent("teacherMail")).setValue("");

}

if (addClassTeacher) {

String className = Helper.getInformation("className", components);

ClassName clazz = classBean.getByName(className);

if (clazz == null) {

message = new FacesMessage("Class with name '" + className + "' does not exist!");

logger.warn("Class with name '" + className + "' does not exist!");

}

}

if (message != null) {

message.setSeverity(FacesMessage.SEVERITY\_ERROR);

facesContext.addMessage(components.getClientId(), message);

facesContext.renderResponse();

}

}

public String cancelAddTeacher() {

init();

return "/director/teachers/teachers";

}

public String addTeacher() throws NoSuchAlgorithmException {

Teacher teacher = new Teacher();

teacher.setName(name);

teacher.setPatronymic(patronymic);

teacher.setSurname(surname);

teacher.setBirth(birth);

teacher.setPhoneNumber(phone);

teacher.setAddress(address);

teacher.setMail(mail);

teacher.setQualification(qualification);

teacher.setSchool(getSchool());

teacher.setLogin(mail);

teacher.setPassword("password");

teacher.setAdditional(additionalInformation);

teacherBean.add(teacher);

addTeacher = false;

return "/director/teachers/teachers";

}

public String addClassTeacher() throws NoSuchAlgorithmException {

ClassTeacher teacher = new ClassTeacher();

teacher.setName(name);

teacher.setPatronymic(patronymic);

teacher.setSurname(surname);

teacher.setBirth(birth);

teacher.setPhoneNumber(phone);

teacher.setAddress(address);

teacher.setMail(mail);

teacher.setQualification(qualification);

teacher.setSchool(getSchool());

teacher.setLogin(mail);

teacher.setPassword("password");

teacher.setClassName(classBean.getByName(className));

teacher.setAdditional(additionalInformation);

classTeacherBean.add(teacher);

addClassTeacher = false;

return "/director/teachers/teachers";

}

public String teacherDetails(Teacher teacher) {

init();

setTeacher(teacher);

return "/director/teachers/teacherDetails";

}

public String classTeacherDetails(ClassTeacher teacher) {

init();

setClassTeacher(teacher);

return "/director/teachers/classTeacherDetails";

}

public List<Teacher> getAllTeachers() {

return teacherBean.getAllBySchoolId(getSchool());

}

public List<ClassTeacher> getAllClassTeachers() {

return classTeacherBean.getAllBySchoolId(getSchool());

}

public String getTeacherBirthStr() {

Calendar birth = Calendar.getInstance(Locale.ROOT);

birth.setTime(teacher.getBirth());

return birth.get(Calendar.DATE) + "-" + (birth.get(Calendar.MONTH) + 1) + "-" + birth.get(Calendar.YEAR);

}

public String getClassTeacherBirthStr() {

Calendar birth = Calendar.getInstance(Locale.ROOT);

birth.setTime(classTeacher.getBirth());

return birth.get(Calendar.DATE) + "-" + (birth.get(Calendar.MONTH) + 1) + "-" + birth.get(Calendar.YEAR);

}

public void setClassTeacherBirthStr(String birthStr) {

this.classTeacherBirthStr = birthStr;

}

public String cancelSaveTeacher() {

editTeacher = false;

return "/director/teachers/teacherDetails";

}

public String cancelSaveClassTeacher() {

editClassTeacher = false;

return "/director/teachers/classTeacherDetails";

}

public String saveTeacher() throws ParseException {

SimpleDateFormat simpleDateFormat = new SimpleDateFormat("dd-MM-yyyy");

teacher.setBirth(simpleDateFormat.parse(teacherBirthStr));

teacherBean.update(teacher);

editTeacher = false;

return "/director/teachers/teacherDetails";

}

public String saveClassTeacher() throws ParseException {

SimpleDateFormat simpleDateFormat = new SimpleDateFormat("dd-MM-yyyy");

classTeacher.setBirth(simpleDateFormat.parse(classTeacherBirthStr));

classTeacherBean.update(classTeacher);

editClassTeacher = false;

return "/director/teachers/classTeacherDetails";

}

public School getSchool() {

school = school == null ? directorController.getDirector().getSchool() : school;

return school;

}

public String dismissalTeacher() {

teacher.setSchool(null);

teacherBean.update(teacher);

return "/director/teachers/teachers";

}

public String dismissalClassTeacher() {

classTeacher.setSchool(null);

classTeacher.setClassName(null);

classTeacherBean.update(classTeacher);

return "/director/teachers/teachers";

}

public boolean isDisciplinesInTeacherExist() {

return teacher.getDisciplines() != null && !teacher.getDisciplines().isEmpty();

}

public boolean isDisciplinesInClassTeacherExist() {

return classTeacher.getDisciplines() != null && !classTeacher.getDisciplines().isEmpty();

}

public String deleteDisciplineForTeacher(Discipline discipline) {

teacher.getDisciplines().remove(discipline);

teacherBean.update(teacher);

return "director/teachers/teacherDetails";

}

public String deleteDisciplineForClassTeacher(Discipline discipline) {

classTeacher.getDisciplines().remove(discipline);

classTeacherBean.update(classTeacher);

return "director/teachers/classTeacherDetails";

}

public void validateDiscipline(ComponentSystemEvent event) {

FacesContext facesContext = FacesContext.getCurrentInstance();

UIComponent components = event.getComponent();

String discipline = Helper.getInformation("discipline", components);

if (discipline.isEmpty()) {

FacesMessage message = new FacesMessage("Fill in required 'theme' field!");

logger.warn("Required field 'theme' must be filled!");

message.setSeverity(FacesMessage.SEVERITY\_ERROR);

facesContext.addMessage(components.getClientId(), message);

facesContext.renderResponse();

}

}

public String getDiscipline() {

return discipline;

}

public void setDiscipline(String discipline) {

this.discipline = discipline;

}

public String toAddDisciplineForTeacher() {

init();

addDisciplineForTeacher = true;

discipline = "";

return "director/teachers/teacherDetails";

}

public String toAddDisciplineForClassTeacher() {

init();

addDisciplineForClassTeacher = true;

discipline = "";

return "director/teachers/classTeacherDetails";

}

public String addDisciplineForTeacher() {

Discipline disciplineForAdd = disciplineBean.getByName(discipline);

if (disciplineForAdd == null) {

disciplineForAdd = new Discipline(discipline);

disciplineForAdd = disciplineBean.add(disciplineForAdd);

}

teacher.getDisciplines().add(disciplineForAdd);

teacherBean.update(teacher);

addDisciplineForTeacher = false;

return "director/teachers/teacherDetails";

}

public String addDisciplineForClassTeacher() {

Discipline disciplineForAdd = disciplineBean.getByName(discipline);

if (disciplineForAdd == null) {

disciplineForAdd = new Discipline(discipline);

disciplineForAdd = disciplineBean.add(disciplineForAdd);

}

classTeacher.getDisciplines().add(disciplineForAdd);

classTeacherBean.update(classTeacher);

addDisciplineForClassTeacher = false;

return "director/teachers/classTeacherDetails";

}

public String cancelAddDisciplineForTeacher() {

addDisciplineForTeacher = false;

return "director/teachers/teacherDetails";

}

public String cancelAddDisciplineForClassTeacher() {

addDisciplineForClassTeacher = false;

return "director/teachers/classTeacherDetails";

}

public boolean isLessonInTeacherExist() {

return teacher.getClasses() != null && !teacher.getClasses().isEmpty();

}

public boolean isLessonInClassTeacherExist() {

return classTeacher.getClasses() != null && !classTeacher.getClasses().isEmpty();

}

public String toAddLessonForTeacher() {

init();

addLessonForTeacher = true;

className = "";

discipline = "";

return "director/teachers/teacherDetails";

}

public String toAddLessonForClassTeacher() {

init();

addLessonForClassTeacher = true;

className = "";

discipline = "";

return "director/teachers/classTeacherDetails";

}

public void validateLesson(ComponentSystemEvent event) {

FacesContext facesContext = FacesContext.getCurrentInstance();

UIComponent components = event.getComponent();

String discipline = Helper.getInformation("addDiscipline", components);

String className = Helper.getInformation("class", components);

FacesMessage message = null;

if (discipline.isEmpty() || className.isEmpty()) {

message = new FacesMessage("Fill in all required fields!");

logger.warn("All required fields must be filled!");

} else {

ClassName clazz = classBean.getByName(className);

if (clazz == null) {

message = new FacesMessage("Class with name '" + className + "' does not exist!");

logger.warn("Class with name '" + className + "' does not exist!");

}

}

if (message != null) {

message.setSeverity(FacesMessage.SEVERITY\_ERROR);

facesContext.addMessage(components.getClientId(), message);

facesContext.renderResponse();

}

}

public String addLessonForTeacher() {

Discipline disciplineForAdd = disciplineBean.getByName(discipline);

if (disciplineForAdd == null) {

disciplineForAdd = new Discipline(discipline);

teacher.getDisciplines().add(disciplineBean.add(disciplineForAdd));

teacherBean.update(teacher);

}

DisciplineForClass disciplineForClass = new DisciplineForClass();

disciplineForClass.setClassName(classBean.getByName(className));

disciplineForClass.setTeacher(teacher);

disciplineForClass.setDiscipline(disciplineBean.getByName(discipline));

DisciplineForClass forClass = disciplineForClassBean.add(disciplineForClass);

teacher.getClasses().add(forClass);

addLessonForTeacher = false;

return "director/teachers/teacherDetails";

}

public String addLessonForClassTeacher() {

Discipline disciplineForAdd = disciplineBean.getByName(discipline);

if (disciplineForAdd == null) {

disciplineForAdd = new Discipline(discipline);

classTeacher.getDisciplines().add(disciplineBean.add(disciplineForAdd));

classTeacherBean.update(classTeacher);

}

DisciplineForClass disciplineForClass = new DisciplineForClass();

disciplineForClass.setClassName(classBean.getByName(className));

disciplineForClass.setTeacher(classTeacher);

disciplineForClass.setDiscipline(disciplineBean.getByName(discipline));

DisciplineForClass forClass = disciplineForClassBean.add(disciplineForClass);

classTeacher.getClasses().add(forClass);

addLessonForClassTeacher = false;

return "director/teachers/classTeacherDetails";

}

public String cancelAddLessonForTeacher() {

addLessonForTeacher = false;

return "director/teachers/teacherDetails";

}

public String cancelAddLessonForClassTeacher() {

addLessonForClassTeacher = false;

return "director/teachers/classTeacherDetails";

}

public DirectorController getDirectorController() {

return directorController;

}

public void setDirectorController(DirectorController directorController) {

this.directorController = directorController;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPatronymic() {

return patronymic;

}

public void setPatronymic(String patronymic) {

this.patronymic = patronymic;

}

public String getSurname() {

return surname;

}

public void setSurname(String surname) {

this.surname = surname;

}

public Date getBirth() {

return birth;

}

public void setBirth(Date birth) {

this.birth = birth;

}

public String getPhone() {

return phone;

}

public void setPhone(String phone) {

this.phone = phone;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String getMail() {

return mail;

}

public void setMail(String mail) {

this.mail = mail;

}

public String getAdditionalInformation() {

return additionalInformation;

}

public void setAdditionalInformation(String additionalInformation) {

this.additionalInformation = additionalInformation;

}

public String getQualification() {

return qualification;

}

public void setQualification(String qualification) {

this.qualification = qualification;

}

public String getClassName() {

return className;

}

public void setClassName(String className) {

this.className = className;

}

public void setTeacherBirthStr(String teacherBirthStr) {

this.teacherBirthStr = teacherBirthStr;

}

public Teacher getTeacher() {

return teacher;

}

public void setTeacher(Teacher teacher) {

this.teacher = teacher;

}

public ClassTeacher getClassTeacher() {

return classTeacher;

}

public void setClassTeacher(ClassTeacher classTeacher) {

this.classTeacher = classTeacher;

}

public boolean isEditTeacher() {

return editTeacher;

}

public void setEditTeacher(boolean editTeacher) {

this.editTeacher = editTeacher;

}

public boolean isEditClassTeacher() {

return editClassTeacher;

}

public void setEditClassTeacher(boolean editClassTeacher) {

this.editClassTeacher = editClassTeacher;

}

public boolean isAddTeacher() {

return addTeacher;

}

public void setAddTeacher(boolean addTeacher) {

this.addTeacher = addTeacher;

}

public boolean isAddClassTeacher() {

return addClassTeacher;

}

public void setAddClassTeacher(boolean addClassTeacher) {

this.addClassTeacher = addClassTeacher;

}

public boolean isAddDisciplineForTeacher() {

return addDisciplineForTeacher;

}

public void setAddDisciplineForTeacher(boolean addDisciplineForTeacher) {

this.addDisciplineForTeacher = addDisciplineForTeacher;

}

public boolean isAddDisciplineForClassTeacher() {

return addDisciplineForClassTeacher;

}

public void setAddDisciplineForClassTeacher(boolean addDisciplineForClassTeacher) {

this.addDisciplineForClassTeacher = addDisciplineForClassTeacher;

}

public boolean isAddLessonForTeacher() {

return addLessonForTeacher;

}

public void setAddLessonForTeacher(boolean addLessonForTeacher) {

this.addLessonForTeacher = addLessonForTeacher;

}

public boolean isAddLessonForClassTeacher() {

return addLessonForClassTeacher;

}

public void setAddLessonForClassTeacher(boolean addLessonForClassTeacher) {

this.addLessonForClassTeacher = addLessonForClassTeacher;

}

public void setSchool(School school) {

this.school = school;

}

}

package managedBeans.student;

import beans.DisciplineForClassBean;

import beans.LessonBean;

import beans.LessonResultBean;

import beans.StudentBean;

import entities.\*;

import org.apache.log4j.Logger;

import utils.Helper;

import javax.ejb.EJB;

import javax.faces.bean.ManagedBean;

import javax.faces.bean.SessionScoped;

import java.util.\*;

/\*\*

\* Created by win10 on 06.12.2016.

\*/

@ManagedBean(name = "studentController", eager = true)

@SessionScoped

public class StudentController {

private final static Logger logger = Logger.getLogger(StudentController.class);

@EJB

private StudentBean studentBean;

@EJB

private DisciplineForClassBean disciplineForClassBean;

@EJB

private LessonBean lessonBean;

@EJB

private LessonResultBean lessonResultBean;

private Student student;

private Discipline discipline;

private LessonResult lessonResult;

private TeachingStaff teacher;

private List<Lesson> lessons;

private ClassTeacher classTeacher;

private School school;

private Director director;

private boolean teachersPage;

public boolean isClassTeacherExist() {

return getClassTeacher() != null;

}

private void init() {

teachersPage = false;

}

public String toClassTeacher() {

init();

return "/student/classTeacherPage?faces-redirect=true";

}

public String toTeacherDetails() {

init();

return "/student/teacherDetails?faces-redirect=true";

}

public String toTeacherDetails(TeachingStaff teacher) {

setTeacher(teacher);

teachersPage = true;

return "/student/teacherDetails?faces-redirect=true";

}

public String toDetailsOfLesson(Discipline discipline, Person teacher) {

setDiscipline(discipline);

setTeacher((TeachingStaff) teacher);

setLessons(lessonBean.getAllByClassAndDiscipline(getStudent().getClassName(), discipline));

return "/student/disciplineDetails?faces-redirect=true";

}

public String dateOfLesson(Lesson lesson) {

Calendar date = Calendar.getInstance(Locale.ROOT);

date.setTime(lesson.getDate());

return date.get(Calendar.DATE) + "-" + (date.get(Calendar.MONTH) + 1) + "-" + date.get(Calendar.YEAR);

}

public String getTeacherBirth() {

Calendar date = Calendar.getInstance(Locale.ROOT);

date.setTime(getTeacher().getBirth());

return date.get(Calendar.DATE) + "-" + (date.get(Calendar.MONTH) + 1) + "-" + date.get(Calendar.YEAR);

}

public String getClassTeacherBirth() {

Calendar date = Calendar.getInstance(Locale.ROOT);

date.setTime(getClassTeacher().getBirth());

return date.get(Calendar.DATE) + "-" + (date.get(Calendar.MONTH) + 1) + "-" + date.get(Calendar.YEAR);

}

public String getDirectorBirth() {

Calendar date = Calendar.getInstance(Locale.ROOT);

date.setTime(getDirector().getBirth());

return date.get(Calendar.DATE) + "-" + (date.get(Calendar.MONTH) + 1) + "-" + date.get(Calendar.YEAR);

}

public LessonResult receiveLessonResult(Lesson lesson) {

setLessonResult(lessonResultBean.getByStudentAndLesson(getStudent(), lesson));

return getLessonResult();

}

public String toSchoolPage() {

init();

return "/student/school?faces-redirect=true";

}

public String toHomePage() {

init();

return "/student/studentIndex?faces-redirect=true";

}

public String toTeachersPage() {

init();

return "/student/teachers?faces-redirect=true";

}

public List<DisciplineForClass> getAllDisciplinesForClass() {

return disciplineForClassBean.getAllDisciplinesForClass(getStudent().getClassName());

}

public Set<TeachingStaff> getAllTeachers() {

List<DisciplineForClass> forClasses = getAllDisciplinesForClass();

Set<TeachingStaff> result = new HashSet<>();

for (DisciplineForClass forClass : forClasses) {

result.add((TeachingStaff) forClass.getTeacher());

}

return result;

}

public Student getStudent() {

student = student == null ? studentBean.getByLogin(Helper.getCurrentUser()) : student;

return student;

}

public void setStudent(Student student) {

this.student = student;

}

public String getBirth() {

Calendar birth = Calendar.getInstance(Locale.ROOT);

birth.setTime(getStudent().getBirth());

return birth.get(Calendar.DATE) + "-" + (birth.get(Calendar.MONTH) + 1) + "-" + birth.get(Calendar.YEAR);

}

public ClassTeacher getClassTeacher() {

return classTeacher == null ? student.getClassName().getClassTeacher() : classTeacher;

}

public School getSchool() {

return school == null ? student.getSchool() : school;

}

public Director getDirector() {

return director == null ? getSchool().getDirector() : director;

}

public Discipline getDiscipline() {

return discipline;

}

public void setDiscipline(Discipline discipline) {

this.discipline = discipline;

}

public LessonResult getLessonResult() {

return lessonResult;

}

public void setLessonResult(LessonResult lessonResult) {

this.lessonResult = lessonResult;

}

public TeachingStaff getTeacher() {

return teacher;

}

public void setTeacher(TeachingStaff teacher) {

this.teacher = teacher;

}

public List<Lesson> getLessons() {

return lessons;

}

public void setLessons(List<Lesson> lessons) {

this.lessons = lessons;

}

public void setClassTeacher(ClassTeacher classTeacher) {

this.classTeacher = classTeacher;

}

public void setSchool(School school) {

this.school = school;

}

public void setDirector(Director director) {

this.director = director;

}

public boolean isTeachersPage() {

return teachersPage;

}

public void setTeachersPage(boolean teachersPage) {

this.teachersPage = teachersPage;

}

}

package managedBeans.teacher;

import beans.TeacherBean;

import entities.Director;

import entities.School;

import entities.Teacher;

import org.apache.log4j.Logger;

import javax.ejb.EJB;

import javax.faces.bean.ManagedBean;

import javax.faces.bean.SessionScoped;

import javax.faces.context.FacesContext;

import java.util.Calendar;

import java.util.Locale;

/\*\*

\* Created by win10 on 06.12.2016.

\*/

@ManagedBean(name = "teacherController", eager = true)

@SessionScoped

public class TeacherController {

private final static Logger logger = Logger.getLogger(TeacherController.class);

@EJB

private TeacherBean teacherBean;

private Teacher teacher;

private School school;

private Director director;

private String getCurrentUser() {

return FacesContext.getCurrentInstance().getExternalContext().getRemoteUser();

}

public Teacher getTeacher() {

teacher = teacher == null ? teacherBean.getByLogin(getCurrentUser()) : teacher;

return teacher;

}

public void setTeacher(Teacher teacher) {

this.teacher = teacher;

}

public String getBirth() {

Calendar birth = Calendar.getInstance(Locale.ROOT);

birth.setTime(teacher.getBirth());

return birth.get(Calendar.DATE) + "-" + (birth.get(Calendar.MONTH) + 1) + "-" + birth.get(Calendar.YEAR);

}

public boolean isDisciplinesExist() {

return teacher.getDisciplines() != null && !teacher.getDisciplines().isEmpty();

}

public boolean isLessonsExist() {

return teacher.getClasses() != null && !teacher.getClasses().isEmpty();

}

public School getSchool() {

return school == null ? teacher.getSchool() : school;

}

public void setSchool(School school) {

this.school = school;

}

public Director getDirector() {

return director == null ? getSchool().getDirector() : director;

}

public void setDirector(Director director) {

this.director = director;

}

public String getDirectorBirth() {

Calendar date = Calendar.getInstance(Locale.ROOT);

date.setTime(getDirector().getBirth());

return date.get(Calendar.DATE) + "-" + (date.get(Calendar.MONTH) + 1) + "-" + date.get(Calendar.YEAR);

}

public String toSchoolPage() {

return "/teacher/school?faces-redirect=true";

}

}

package managedBeans;

import beans.DirectorBean;

import beans.PersonBean;

import entities.Director;

import entities.Person;

import org.apache.log4j.Logger;

import utils.Helper;

import javax.ejb.EJB;

import javax.faces.application.FacesMessage;

import javax.faces.bean.ManagedBean;

import javax.faces.bean.SessionScoped;

import javax.faces.component.UIComponent;

import javax.faces.component.UIInput;

import javax.faces.context.FacesContext;

import javax.faces.event.ComponentSystemEvent;

import java.security.NoSuchAlgorithmException;

import java.util.Date;

/\*\*

\* Created by win10 on 23.11.2016.

\*/

@ManagedBean(name = "accountBean", eager = true)

@SessionScoped

public class AccountController {

private final static Logger logger = Logger.getLogger(AccountController.class);

@EJB

private DirectorBean directorBean;

@EJB

private PersonBean personBean;

private String login;

private String password;

private String name;

private String patronymic;

private String surname;

private Date birth;

private String phone;

private String address;

private String mail;

private String additionalInformation;

public String signUp() throws NoSuchAlgorithmException {

Director director = new Director();

director.setName(name);

director.setPatronymic(patronymic);

director.setSurname(surname);

director.setBirth(birth);

director.setPhoneNumber(phone);

director.setAddress(address);

director.setMail(mail);

director.setLogin(login);

director.setPassword(password);

if (!additionalInformation.isEmpty()) {

director.setAdditional(additionalInformation);

}

directorBean.add(director);

return "/index?faces-redirect=true";

}

public String toRegistration() {

login = "";

password = "";

name = "";

patronymic = "";

surname = "";

birth = null;

phone = "";

mail = "";

address = "";

additionalInformation = "";

return "registration";

}

public void validateForm(ComponentSystemEvent event) {

FacesContext facesContext = FacesContext.getCurrentInstance();

UIComponent components = event.getComponent();

String name = Helper.getInformation("name", components);

String patronymic = Helper.getInformation("patronymic", components);

String surname = Helper.getInformation("surname", components);

String phone = Helper.getInformation("phone", components);

String mail = Helper.getInformation("mail", components);

String address = Helper.getInformation("address", components);

String login = Helper.getInformation("login", components);

String password = Helper.getInformation("password", components);

String confirmPassword = Helper.getInformation("confirmPassword", components);

FacesMessage message = null;

if (name.isEmpty() || patronymic.isEmpty() || surname.isEmpty() || phone.isEmpty() || mail.isEmpty()

|| address.isEmpty() || login.isEmpty() || password.isEmpty() || confirmPassword.isEmpty()) {

message = new FacesMessage("Fill in all required fields!");

logger.warn("All required fields must be filled!");

} else {

boolean checkPhone = Helper.checkPhone(phone);

boolean checkMail = Helper.checkMail(mail);

if (!checkPhone || !checkMail) {

if (!checkPhone && !checkMail) {

message = new FacesMessage("Wrong phone number and e-mail format!");

logger.warn("Wrong phone number(" + phone + ") and e-mail(" + mail + ") format!");

((UIInput) components.findComponent("mail")).setValue("");

((UIInput) components.findComponent("phone")).setValue("");

} else if (!checkPhone) {

message = new FacesMessage("Wrong phone format!");

logger.warn("Wrong phone number(" + phone + ") format!");

((UIInput) components.findComponent("phone")).setValue("");

} else {

message = new FacesMessage("Wrong e-mail format!");

logger.warn("Wrong e-mail(" + mail + ") format!");

((UIInput) components.findComponent("mail")).setValue("");

}

} else if (!password.isEmpty() && !confirmPassword.isEmpty() && !password.equals(confirmPassword)) {

message = new FacesMessage("Passwords must be the same!");

logger.warn("Passwords must be the same!");

}

}

Person person = personBean.getByMail(mail);

if (person != null) {

message = new FacesMessage("This email address is already taken!");

logger.warn("Email address (" + mail + ") is already taken!");

((UIInput) components.findComponent("mail")).setValue("");

}

if (message != null) {

message.setSeverity(FacesMessage.SEVERITY\_ERROR);

facesContext.addMessage(components.getClientId(), message);

facesContext.renderResponse();

}

}

public String getLogin() {

return login;

}

public void setLogin(String login) {

this.login = login;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPatronymic() {

return patronymic;

}

public void setPatronymic(String patronymic) {

this.patronymic = patronymic;

}

public String getSurname() {

return surname;

}

public void setSurname(String surname) {

this.surname = surname;

}

public Date getBirth() {

return birth;

}

public void setBirth(Date birth) {

this.birth = birth;

}

public String getPhone() {

return phone;

}

public void setPhone(String phone) {

this.phone = phone;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String getMail() {

return mail;

}

public void setMail(String mail) {

this.mail = mail;

}

public String getAdditionalInformation() {

return additionalInformation;

}

public void setAdditionalInformation(String additionalInformation) {

this.additionalInformation = additionalInformation;

}

}

package managedBeans;

import org.apache.log4j.Logger;

import utils.Helper;

import javax.faces.application.FacesMessage;

import javax.faces.bean.ManagedBean;

import javax.faces.bean.SessionScoped;

import javax.faces.component.UIComponent;

import javax.faces.context.FacesContext;

import javax.faces.event.ComponentSystemEvent;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;

/\*\*

\* Created by win10 on 23.11.2016.

\*/

@ManagedBean(name = "userSession", eager = true)

@SessionScoped

public class UserSession {

private static final Logger logger = Logger.getLogger(UserSession.class);

private String username;

private String password;

private boolean isLogin;

public String login() {

if (isLogin) {

logout();

}

HttpServletRequest request =

(HttpServletRequest) FacesContext.getCurrentInstance().getExternalContext().getRequest();

try {

request.login(username, password);

isLogin = true;

if (request.isUserInRole("DIRECTOR")) {

return "/director/directorIndex?faces-redirect=true";

} else if (request.isUserInRole("TEACHER")) {

return "/teacher/teacherIndex?faces-redirect=true";

} else if (request.isUserInRole("STUDENT")) {

return "/student/studentIndex?faces-redirect=true";

} else if (request.isUserInRole("CLASS\_TEACHER")) {

return "/classTeacher/classTeacherIndex?faces-redirect=true";

}

} catch (ServletException e) {

logger.error("Error authorization user with login=" + username);

return "/error";

} finally {

username = "";

password = "";

}

logger.warn("Unknown user role!");

logout();

return "/error";

}

public String logout() {

HttpSession session = (HttpSession) FacesContext.getCurrentInstance().getExternalContext().getSession(true);

logger.info("Invalidate session for user: " + Helper.getCurrentUser());

session.invalidate();

isLogin = false;

return "/index?faces-redirect=true";

}

public String toUserPage() {

HttpServletRequest request =

(HttpServletRequest) FacesContext.getCurrentInstance().getExternalContext().getRequest();

if (request.isUserInRole("DIRECTOR")) {

return "/director/directorIndex?faces-redirect=true";

} else if (request.isUserInRole("TEACHER")) {

return "/teacher/teacherIndex?faces-redirect=true";

} else if (request.isUserInRole("CLASS\_TEACHER")) {

return "/classTeacher/classTeacherIndex?faces-redirect=true";

} else if (request.isUserInRole("STUDENT")) {

return "/student/studentIndex?faces-redirect=true";

}

logger.warn("Unknown user role!");

logout();

return "/error";

}

public void validateForm(ComponentSystemEvent event) {

FacesContext facesContext = FacesContext.getCurrentInstance();

UIComponent components = event.getComponent();

String login = Helper.getInformation("login", components);

String password = Helper.getInformation("password", components);

if (login.isEmpty() || password.isEmpty()) {

FacesMessage message = new FacesMessage("Fill in 'login' and 'password' required fields!");

logger.warn("All required fields must be filled!");

message.setSeverity(FacesMessage.SEVERITY\_ERROR);

facesContext.addMessage(components.getClientId(), message);

facesContext.renderResponse();

}

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public boolean isLogin() {

return isLogin;

}

}

package utils;

import javax.faces.component.UIComponent;

import javax.faces.component.UIInput;

import javax.faces.context.FacesContext;

/\*\*

\* Created by win10 on 07.12.2016.

\*/

public class Helper {

public static String getInformation(String component, UIComponent components) {

Object input = ((UIInput) components.findComponent(component)).getLocalValue();

return input != null ? input.toString() : "";

}

public static boolean checkPhone(String phone) {

return ((phone.matches("^\\+[\\(\\-]?(\\d[\\(\\)\\-]?){11}\\d$") || phone.matches("^\\(?(\\d[\\-\\(\\)]?){9}\\d$"))

&& phone.matches("[\\+]?\\d\*(\\(\\d{3}\\))?\\d\*\\-?\\d\*\\-?\\d\*\\d$"));

}

public static boolean checkMail(String mail) {

return mail.matches("^[\_A-Za-z0-9-\\+]+(\\.[\_A-Za-z0-9-]+)\*@[A-Za-z0-9-]+(\\.[A-Za-z0-9]+)\*(\\.[A-Za-z]{2,})$");

}

public static String getCurrentUser() {

return FacesContext.getCurrentInstance().getExternalContext().getRemoteUser();

}

}

# 