# ДОДАТКИ

**SQL код створення таблиць бази даних**

CREATE SCHEMA `trainer\_online` DEFAULT CHARACTER SET utf8 ;

CREATE TABLE `trainer\_online`.`accounts` (

`id` INT NOT NULL AUTO\_INCREMENT,

`login` VARCHAR(100) NOT NULL,

`password` VARCHAR(100) NOT NULL,

`role` VARCHAR(100) NOT NULL,

`name` VARCHAR(100) NOT NULL,

`surname` VARCHAR(100) NOT NULL,

`dob` DATE NULL,

`trainerID` INT NULL,

`rating` DOUBLE NULL,

`description` LONGTEXT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `id\_UNIQUE` (`id` ASC),

UNIQUE INDEX `login\_UNIQUE` (`login` ASC));

CREATE TABLE `trainer\_online`.`tranings` (

`id` INT NOT NULL AUTO\_INCREMENT,

`date` DATE NULL,

`description` LONGTEXT NULL,

`clientID` INT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `id\_UNIQUE` (`id` ASC),

INDEX `clientID\_idx` (`clientID` ASC),

CONSTRAINT `clientID`

FOREIGN KEY (`clientID`)

REFERENCES `trainer\_online`.`accounts` (`id`)

ON DELETE CASCADE

ON UPDATE CASCADE);

CREATE TABLE `trainer\_online`.`exercises` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(1024) NOT NULL,

`duration` DOUBLE NULL,

`description` LONGTEXT NULL,

`trainingID` INT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `id\_UNIQUE` (`id` ASC),

INDEX `trainingID\_idx` (`trainingID` ASC),

CONSTRAINT `trainingID`

FOREIGN KEY (`trainingID`)

REFERENCES `trainer\_online`.`tranings` (`id`)

ON DELETE CASCADE

ON UPDATE CASCADE);

CREATE TABLE `trainer\_online`.`schedule\_unit` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(1000) NOT NULL,

`description` LONGTEXT NULL,

`weekDay` INT NOT NULL,

`clientID` INT NOT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `id\_UNIQUE` (`id` ASC),

INDEX `clientID\_idx` (`clientID` ASC),

CONSTRAINT `schedule\_clientID`

FOREIGN KEY (`clientID`)

REFERENCES `trainer\_online`.`accounts` (`id`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

CREATE TABLE `trainer\_online`.`messages` (

`id` INT NOT NULL AUTO\_INCREMENT,

`isRead` TINYINT NOT NULL,

`message` LONGTEXT NOT NULL,

`date` DATE NOT NULL,

`fromAccount` INT NOT NULL,

`toAccount` INT NOT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `id\_UNIQUE` (`id` ASC),

INDEX `fromFK\_idx` (`from` ASC),

INDEX `toFK\_idx` (`to` ASC),

CONSTRAINT `fromFK`

FOREIGN KEY (`from`)

REFERENCES `trainer\_online`.`accounts` (`id`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `toFK`

FOREIGN KEY (`to`)

REFERENCES `trainer\_online`.`accounts` (`id`)

ON DELETE CASCADE

ON UPDATE CASCADE);

## Код програми

**Account.java**

package entity;

import javax.persistence.\*;

import java.io.Serializable;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.util.Base64;

import java.util.Calendar;

import java.util.Date;

import java.util.Locale;

/\*\*

\* Users accounts

\* @author Bogdan Kaftanatiy

\*/

@Entity

@Inheritance(strategy = InheritanceType.SINGLE\_TABLE)

@DiscriminatorColumn(name = "role")

@Table(name = "accounts")

public abstract class Account implements Serializable {

@Id

@Column(name = "id")

@GeneratedValue(strategy = GenerationType.IDENTITY)

protected int id;

@Column(name = "login")

protected String login;

@Column(name = "password")

protected String password;

@Column(name = "name")

protected String name;

@Column(name = "surname")

protected String surname;

@Column(name = "dob")

@Temporal(TemporalType.TIMESTAMP)

protected Date dob;

public Account() {

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getLogin() {

return login;

}

public void setLogin(String login) {

this.login = login;

}

public String getPassword() {

return password;

}

public void setPassword(String password) throws NoSuchAlgorithmException {

this.password = getEncodedPassword(password);

}

private static String getEncodedPassword(String password) throws NoSuchAlgorithmException {

return new String(Base64.getEncoder().encode(encryption(password)));

}

private static byte[] encryption(String password) throws NoSuchAlgorithmException {

MessageDigest md = MessageDigest.getInstance("SHA-256");

md.update(password.getBytes());

return md.digest();

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getSurname() {

return surname;

}

public void setSurname(String surname) {

this.surname = surname;

}

public Date getDob() {

return dob;

}

public void setDob(Date dob) {

this.dob = dob;

}

public int getAge() {

Date currentDate = new Date();

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

current.setTime(currentDate);

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

birth.setTime(dob);

int diff = current.get(Calendar.YEAR) - birth.get(Calendar.YEAR);

if(birth.get(Calendar.DAY\_OF\_YEAR) > birth.get(Calendar.DAY\_OF\_YEAR)) {

diff--;

}

return diff;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

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

Account account = (Account) o;

if (id != account.id) return false;

if (!login.equals(account.login)) return false;

if (!password.equals(account.password)) return false;

if (!name.equals(account.name)) return false;

if (!surname.equals(account.surname)) return false;

return dob != null ? dob.equals(account.dob) : account.dob == null;

}

@Override

public int hashCode() {

int result = (int) (id ^ (id >>> 32));

result = 31 \* result + login.hashCode();

result = 31 \* result + password.hashCode();

result = 31 \* result + name.hashCode();

result = 31 \* result + surname.hashCode();

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

return result;

}

}

**Admin.java**

package entity;

import javax.persistence.DiscriminatorValue;

import javax.persistence.Entity;

import java.util.Date;

/\*\*

\* @author Bogdan Kaftanatiy

\*/

@Entity

@DiscriminatorValue("ADMIN")

public class Admin extends Account {

public Admin() {}

public Admin(String login, String password, String name, String surname, Date dob) {

this.login = login;

this.password = password;

this.name = name;

this.surname = surname;

this.dob = dob;

}

@Override

public String toString() {

return "Admin{" +

"id=" + id +

", login='" + login + '\'' +

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

", surname='" + surname + '\'' +

", dob=" + dob +

'}';

}

}

**Client.java**

package entity;

import javax.persistence.\*;

import java.util.Date;

/\*\*

\* @author Bogdan Kaftanatiy

\*/

@Entity

@DiscriminatorValue("CLIENT")

@NamedQuery(name = "Client.getAll", query = "SELECT c FROM Client c")

public class Client extends Account {

@ManyToOne(fetch = FetchType.EAGER, cascade = CascadeType.MERGE)

@JoinColumn(name = "trainerID")

private Trainer trainer;

public Client() {

}

public Client(String login, String password, String name, String surname, Date dob) {

this.login = login;

this.password = password;

this.name = name;

this.surname = surname;

this.dob = dob;

}

public Trainer getTrainer() {

return trainer;

}

public void setTrainer(Trainer trainer) {

if (this.trainer != null) {

this.trainer.removeClient(this);

}

if (trainer == null) {

this.trainer = null;

} else {

this.trainer = trainer;

this.trainer.addClient(this);

}

}

@Override

public String toString() {

return "Client{" +

"id=" + id +

", login='" + login + '\'' +

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

", surname='" + surname + '\'' +

", dob=" + dob +

'}';

}

}

**Exercise.java**

package entity;

import javax.persistence.\*;

import java.io.Serializable;

/\*\*

\* @author Bogdan Kaftanatiy

\*/

@Entity

@Table(name = "exercises")

public class Exercise implements Serializable {

@Id

@Column(name = "id")

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@Column(name = "name")

private String name;

@Column(name = "duration")

private double duration;

@Column(name = "description")

private String description;

@ManyToOne(fetch = FetchType.EAGER, cascade = CascadeType.MERGE)

@JoinColumn(name = "trainingID")

private Training training;

public Exercise() {

}

public Exercise(String name, double duration, String description, Training training) {

this.name = name;

this.duration = duration;

this.description = description;

this.training = training;

}

public Exercise(int id, String name, double duration, String description, Training training) {

this.id = id;

this.name = name;

this.duration = duration;

this.description = description;

this.training = training;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public double getDuration() {

return duration;

}

public void setDuration(double duration) {

this.duration = duration;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

public Training getTraining() {

return training;

}

public void setTraining(Training training) {

this.training = training;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

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

Exercise exercise = (Exercise) o;

if (id != exercise.id) return false;

if (Double.compare(exercise.duration, duration) != 0) return false;

if (name != null ? !name.equals(exercise.name) : exercise.name != null) return false;

if (description != null ? !description.equals(exercise.description) : exercise.description != null)

return false;

return training != null ? training.equals(exercise.training) : exercise.training == null;

}

@Override

public int hashCode() {

int result;

long temp;

result = id;

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

temp = Double.doubleToLongBits(duration);

result = 31 \* result + (int) (temp ^ (temp >>> 32));

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

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

return result;

}

@Override

public String toString() {

return "Exercise{" +

"id=" + id +

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

", duration=" + duration +

", description='" + description + '\'' +

", training=" + training +

'}';

}

}

**Message.java**

package entity;

import javax.persistence.\*;

import java.io.Serializable;

import java.util.Date;

/\*\*

\* @author Bogdan Kaftanatiy

\*/

@Entity

@Table(name = "messages")

public class Message implements Serializable {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id")

private int id;

@Column(name = "isRead")

private boolean isRead;

@Column(name = "message")

private String message;

@Column(name = "date")

@Temporal(TemporalType.TIMESTAMP)

private Date date;

@ManyToOne(fetch = FetchType.EAGER)

@JoinColumn(name = "fromAccount")

private Account fromAccount;

@ManyToOne(fetch = FetchType.EAGER)

@JoinColumn(name = "toAccount")

private Account toAccount;

public Message() {

this.isRead = false;

}

public Message(String message, Date date, Account from, Account to) {

this.message = message;

this.date = date;

this.fromAccount = from;

this.toAccount = to;

this.isRead = false;

}

public Message(String message) {

this.message = message;

this.isRead = false;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public boolean isRead() {

return isRead;

}

public void setRead(boolean read) {

isRead = read;

}

public String getMessage() {

return message;

}

public void setMessage(String message) {

this.message = message;

}

public Date getDate() {

return date;

}

public void setDate(Date date) {

this.date = date;

}

public Account getFromAccount() {

return fromAccount;

}

public void setFromAccount(Account fromAccount) {

this.fromAccount = fromAccount;

}

public Account getToAccount() {

return toAccount;

}

public void setToAccount(Account to) {

this.toAccount = to;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

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

Message message1 = (Message) o;

if (id != message1.id) return false;

if (isRead != message1.isRead) return false;

if (message != null ? !message.equals(message1.message) : message1.message != null) return false;

if (date != null ? !date.equals(message1.date) : message1.date != null) return false;

if (fromAccount != null ? !fromAccount.equals(message1.fromAccount) : message1.fromAccount != null) return false;

return toAccount != null ? toAccount.equals(message1.toAccount) : message1.toAccount == null;

}

@Override

public int hashCode() {

int result = id;

result = 31 \* result + (isRead ? 1 : 0);

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

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

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

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

return result;

}

@Override

public String toString() {

return "Message{" +

"id=" + id +

", isRead=" + isRead +

", message='" + message + '\'' +

", date=" + date +

", from=" + fromAccount +

", to=" + toAccount +

'}';

}

}

**SceduleUnit.java**

package entity;

import javax.persistence.\*;

import java.io.Serializable;

/\*\*

\* @author Bogdan Kaftanatiy

\*/

@Entity

@Table(name = "schedule\_unit")

public class ScheduleUnit implements Serializable {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id")

private int id;

@Column(name = "name")

private String name;

@Column(name = "description")

private String description;

@Column(name = "weekDay")

private int weekDay;

@ManyToOne(fetch = FetchType.EAGER, cascade = CascadeType.MERGE)

@JoinColumn(name = "clientID", nullable = false)

private Client client;

public ScheduleUnit() {

}

public ScheduleUnit(String name, String description, int weekDay, Client client) {

this.name = name;

this.description = description;

this.weekDay = weekDay;

this.client = client;

}

public ScheduleUnit(int id, String name, String description, int weekDay, Client client) {

this.id = id;

this.name = name;

this.description = description;

this.weekDay = weekDay;

this.client = client;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

public int getWeekDay() {

return weekDay;

}

public void setWeekDay(int weekDay) {

this.weekDay = weekDay;

}

public Client getClient() {

return client;

}

public void setClient(Client client) {

this.client = client;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

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

ScheduleUnit that = (ScheduleUnit) o;

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

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

if (name != null ? !name.equals(that.name) : that.name != null) return false;

if (description != null ? !description.equals(that.description) : that.description != null) return false;

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

}

@Override

public int hashCode() {

int result = id;

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

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

result = 31 \* result + weekDay;

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

return result;

}

@Override

public String toString() {

return "ScheduleUnit{" +

"id=" + id +

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

", description='" + description + '\'' +

", weekDay=" + weekDay +

", clientID=" + client.getId() +

'}';

}

}

**Trainer.java**

package entity;

import javax.persistence.\*;

import java.util.\*;

/\*\*

\* @author Bogdan Kaftanatiy

\*/

@Entity

@DiscriminatorValue("TRAINER")

@NamedQuery(query = "SELECT t FROM Trainer t", name = "Trainer.getAll")

public class Trainer extends Account {

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

private List<Client> clients;

@Column(name = "rating")

private double rating;

@Column(name = "description")

private String description;

public Trainer() {

clients = new ArrayList<>();

}

public Trainer(String login, String password, String name, String surname, Date dob) {

this.login = login;

this.password = password;

this.name = name;

this.surname = surname;

this.dob = dob;

}

public Trainer(String login, String password, String name, String surname, Date dob, double rating) {

this.login = login;

this.password = password;

this.name = name;

this.surname = surname;

this.dob = dob;

this.rating = rating;

}

public Trainer(String login, String password, String name, String surname, Date dob, double rating, String description) {

this.login = login;

this.password = password;

this.name = name;

this.surname = surname;

this.dob = dob;

this.rating = rating;

this.description = description;

}

public double getRating() {

return rating;

}

public void setRating(double rating) {

this.rating = rating;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

public void addClient(Client client) {

clients.add(client);

}

public void removeClient(Client client) {

clients.remove(client);

}

@Override

public String toString() {

return "Trainer{" +

"id=" + id +

", login='" + login + '\'' +

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

", surname='" + surname + '\'' +

", dob=" + dob +

'}';

}

}

**Training.java**

package entity;

import javax.persistence.\*;

import java.io.Serializable;

import java.util.\*;

/\*\*

\* @author Bogdan Kaftanatiy

\*/

@Entity

@Table(name = "tranings")

public class Training implements Serializable {

@Id

@Column(name = "id")

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@Column(name = "date")

@Temporal(TemporalType.TIMESTAMP)

private Date date;

@Column(name = "description")

private String description;

@ManyToOne(fetch = FetchType.EAGER, cascade = CascadeType.MERGE)

@JoinColumn(name = "clientID")

private Client client;

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

private Set<Exercise> exercises;

public Training() {

exercises = new HashSet<>();

}

public Training(Date date, String description, Client client) {

this.date = date;

this.description = description;

this.client = client;

this.exercises = new HashSet<>();

}

public Training(int id, Date date, String description, Client client) {

this.id = id;

this.date = date;

this.description = description;

this.client = client;

this.exercises = new HashSet<>();

}

public void addExercise(Exercise exercise) {

exercises.add(exercise);

exercise.setTraining(this);

}

public void removeExercise(Exercise exercise) {

exercises.remove(exercise);

exercise.setTraining(null);

}

public String getFormattedDate() {

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

calendarDate.setTime(date);

String result = calendarDate.get(Calendar.DAY\_OF\_MONTH) + "." + (calendarDate.get(Calendar.MONTH) + 1) + "."

+ calendarDate.get(Calendar.YEAR);

return result;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public Date getDate() {

return date;

}

public void setDate(Date date) {

this.date = date;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

public Client getClient() {

return client;

}

public void setClient(Client client) {

this.client = client;

}

public Set<Exercise> getExercises() {

return exercises;

}

public void setExercises(Set<Exercise> exercises) {

this.exercises = exercises;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

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

Training training = (Training) o;

if (id != training.id) return false;

if (date != null ? !date.equals(training.date) : training.date != null) return false;

if (description != null ? !description.equals(training.description) : training.description != null)

return false;

if (client != null ? !client.equals(training.client) : training.client != null) return false;

return exercises != null ? exercises.equals(training.exercises) : training.exercises == null;

}

@Override

public int hashCode() {

int result = id;

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

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

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

return result;

}

@Override

public String toString() {

return "Training{" +

"id=" + id +

", date=" + date +

", description='" + description + '\'' +

", client=" + client +

'}';

}

}

**AbstractDAO.java**

package bean;

import org.apache.log4j.Logger;

import javax.ejb.TransactionAttribute;

import javax.ejb.TransactionAttributeType;

import javax.persistence.EntityManager;

import javax.persistence.PersistenceContext;

/\*\*

\* @author Bogdan Kaftanatiy

\*/

public abstract class AbstractDAO<DataT> {

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

@PersistenceContext(unitName = "program\_pu")

protected EntityManager em;

public abstract DataT getObject(int id);

@TransactionAttribute(TransactionAttributeType.REQUIRES\_NEW)

public DataT updateObject(DataT object) {

DataT result = em.merge(object);

logger.info("Updated object: " + object);

return result;

}

public void deleteObject(int id) {

em.remove(getObject(id));

logger.info("Deleted object with id=" + id);

}

}

**AdminBean.java**

package bean;

import entity.Admin;

import javax.ejb.Stateless;

import javax.ejb.TransactionManagement;

import javax.ejb.TransactionManagementType;

/\*\*

\* @author Bogdan Kaftanatiy

\*/

@Stateless

@TransactionManagement(value = TransactionManagementType.CONTAINER)

public class AdminBean extends AbstractDAO<Admin> {

@Override

public Admin getObject(int id) {

Admin result = em.find(Admin.class, id);

if(result != null) {

logger.info("Find admin with id = " + id);

} else {

logger.info("Admin with id = " + id + ", not found");

}

return result;

}

}

**ClientBean.java**

package bean;

import entity.Client;

import entity.Trainer;

import javax.ejb.Stateless;

import javax.ejb.TransactionManagement;

import javax.ejb.TransactionManagementType;

import javax.persistence.Query;

import javax.persistence.TypedQuery;

import java.util.List;

/\*\*

\* @author Bogdan Kaftanatiy

\*/

@Stateless

@TransactionManagement(value = TransactionManagementType.CONTAINER)

public class ClientBean extends AbstractDAO<Client> {

@Override

public Client getObject(int id) {

Client result = em.find(Client.class, id);

if(result != null) {

logger.info("Find client with id = " + id);

} else {

logger.info("Client with id = " + id + ", not found");

}

return result;

}

public Client searchByLogin(String login) {

TypedQuery<Client> query = em.createQuery("SELECT c FROM Client c WHERE c.login=:loginParam", Client.class);

query.setParameter("loginParam", login);

List<Client> array = query.getResultList();

if(array==null || array.isEmpty()){

logger.warn("Client with login='" + login + "' not exist");

return null;

} else {

logger.info("Find client with login=" + login);

return array.get(0);

}

}

public List<Client> trainerClients(Trainer trainer) {

TypedQuery<Client> query =

em.createQuery("SELECT c FROM Client c WHERE c.trainer=:trainerParam",

Client.class);

query.setParameter("trainerParam", trainer);

List<Client> result = query.getResultList();

logger.info("Find " + result.size() + " clients for trainer: " + trainer);

return result;

}

public List<Client> getAllClients() {

TypedQuery<Client> query = em.createNamedQuery("Client.getAll", Client.class);

List<Client> result = query.getResultList();

logger.info("Get all client");

return result;

}

}

**ExerciseBean.java**

package bean;

import entity.Exercise;

import javax.ejb.Stateless;

import javax.ejb.TransactionManagement;

import javax.ejb.TransactionManagementType;

/\*\*

\* @author Bogdan Kaftanatiy

\*/

@Stateless

@TransactionManagement(value = TransactionManagementType.CONTAINER)

public class ExerciseBean extends AbstractDAO<Exercise> {

@Override

public Exercise getObject(int id) {

Exercise result = em.find(Exercise.class, id);

if(result != null) {

logger.info("Find exercise with id = " + id);

} else {

logger.info("Exercise with id = " + id + ", not found");

}

return result;

}

}

**MessageBean.java**

package bean;

import entity.Account;

import entity.Message;

import javax.ejb.Stateless;

import javax.ejb.TransactionManagement;

import javax.ejb.TransactionManagementType;

import javax.persistence.TypedQuery;

import java.util.List;

/\*\*

\* @author Bogdan Kaftanatiy

\*/

@Stateless

@TransactionManagement(value = TransactionManagementType.CONTAINER)

public class MessageBean extends AbstractDAO<Message> {

@Override

public Message getObject(int id) {

Message result = em.find(Message.class, id);

if(result != null) {

logger.info("Find message with id = " + id);

} else {

logger.info("Message with id = " + id + ", not found");

}

return result;

}

public List<Message> getAccountMessages(Account fromAccount, Account toAccount) {

TypedQuery<Message> query =

em.createQuery( "SELECT m FROM Message m " +

"WHERE (m.fromAccount=:fromParam AND m.toAccount=:toParam) " +

"OR (m.fromAccount=:toParam AND m.toAccount=:fromParam)",

Message.class);

query.setParameter("fromParam", fromAccount);

query.setParameter("toParam", toAccount);

List<Message> result = query.getResultList();

logger.info("Find " + result.size() + " messages for dialog: " + fromAccount + " and " + toAccount);

return result;

}

}

**ScheduleUnit.java**

package bean;

import entity.Client;

import entity.ScheduleUnit;

import javax.ejb.Stateless;

import javax.ejb.TransactionManagement;

import javax.ejb.TransactionManagementType;

import javax.persistence.TypedQuery;

import java.util.List;

/\*\*

\* @author Bogdan Kaftanatiy

\*/

@Stateless

@TransactionManagement(value = TransactionManagementType.CONTAINER)

public class ScheduleUnitBean extends AbstractDAO<ScheduleUnit> {

@Override

public ScheduleUnit getObject(int id) {

ScheduleUnit result = em.find(ScheduleUnit.class, id);

if(result != null) {

logger.info("Find schedule unit with id = " + id);

} else {

logger.info("Schedule unit with id = " + id + ", not found");

}

return result;

}

public List<ScheduleUnit> getClientSchedule(Client client, int weekDay) {

TypedQuery<ScheduleUnit> query =

em.createQuery("SELECT s FROM ScheduleUnit s WHERE s.client=:clientParam AND s.weekDay=:weekParam",

ScheduleUnit.class);

query.setParameter("clientParam", client);

query.setParameter("weekParam", weekDay);

List<ScheduleUnit> result = query.getResultList();

logger.info("Find " + result.size() + " schedule units for client: " + client + ", in day: " + weekDay);

return result;

}

}

**TrainerBean.java**

package bean;

import entity.Trainer;

import javax.ejb.Stateless;

import javax.ejb.TransactionManagement;

import javax.ejb.TransactionManagementType;

import javax.persistence.TypedQuery;

import java.util.ArrayList;

import java.util.List;

/\*\*

\* @author Bogdan Kaftanatiy

\*/

@Stateless

@TransactionManagement(value = TransactionManagementType.CONTAINER)

public class TrainerBean extends AbstractDAO<Trainer> {

@Override

public Trainer getObject(int id) {

Trainer result = em.find(Trainer.class, id);

if(result != null) {

logger.info("Find trainer with id = " + id);

} else {

logger.info("Trainer with id = " + id + ", not found");

}

return result;

}

public List<Trainer> getAll() {

TypedQuery<Trainer> query = em.createNamedQuery("Trainer.getAll", Trainer.class);

List<Trainer> result = query.getResultList();

logger.info("Get all trainer");

return result;

}

public List<Trainer> getAppropriateTrainers(double rating) {

List<Trainer> allTrainer = getAll();

List<Trainer> result = new ArrayList<>();

for (Trainer t : allTrainer) {

if(t.getRating() >= rating)

result.add(t);

}

return result;

}

public Trainer searchByLogin(String login) {

TypedQuery<Trainer> query = em.createQuery("SELECT t FROM Trainer t WHERE t.login=:loginParam", Trainer.class);

query.setParameter("loginParam", login);

List<Trainer> array = query.getResultList();

if(array==null || array.isEmpty()){

logger.warn("Trainer with login='" + login + "' not exist");

return null;

} else {

logger.info("Find trainer with login=" + login);

return array.get(0);

}

}

}

**TrainingBean.java**

package bean;

import entity.Client;

import entity.Training;

import javax.ejb.Stateless;

import javax.ejb.TransactionManagement;

import javax.ejb.TransactionManagementType;

import javax.persistence.TypedQuery;

import java.util.List;

/\*\*

\* @author Bogdan Kaftanatiy

\*/

@Stateless

@TransactionManagement(value = TransactionManagementType.CONTAINER)

public class TrainingBean extends AbstractDAO<Training> {

@Override

public Training getObject(int id) {

Training result = em.find(Training.class, id);

if(result != null) {

logger.info("Find training with id = " + id);

} else {

logger.info("Training with id = " + id + ", not found");

}

return result;

}

public List<Training> getClientTrainings(Client client) {

TypedQuery<Training> query =

em.createQuery("SELECT t FROM Training t WHERE t.client=:clientParam", Training.class);

query.setParameter("clientParam", client);

List<Training> result = query.getResultList();

logger.info("Find " + result.size() + " trainings for client: " + client);

return result;

}

}

**UserSession.java**

package managedBean;

import org.apache.log4j.Logger;

import javax.faces.bean.ManagedBean;

import javax.faces.bean.SessionScoped;

import javax.faces.context.FacesContext;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;

/\*\*

\* @author Bogdan Kaftanatiy

\*/

@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 = false;

public String login() {

if(isLogin)

logout();

HttpServletRequest request =

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

try {

request.login(username, password);

isLogin = true;

if(request.isUserInRole("ADMIN"))

return "/admin/admin\_index?faces-redirect=true";

if(request.isUserInRole("TRAINER"))

return "/trainer/trainer\_index?faces-redirect=true";

if(request.isUserInRole("CLIENT"))

return "/client/client\_index?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: " + getCurrentUser());

session.invalidate();

isLogin = false;

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

}

public boolean isLogin() {

return isLogin;

}

public String getCurrentUser() {

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

}

public String toUserPage() {

HttpServletRequest request =

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

if(request.isUserInRole("ADMIN"))

return "/admin/admin\_index?faces-redirect=true";

if(request.isUserInRole("TRAINER"))

return "/trainer/trainer\_index?faces-redirect=true";

if(request.isUserInRole("CLIENT"))

return "/client/client\_index?faces-redirect=true";

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

logout();

return "/error";

}

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;

}

}

**AccountController.java**

package managedBean;

import bean.AdminBean;

import bean.ClientBean;

import bean.TrainerBean;

import entity.Admin;

import entity.Client;

import entity.Trainer;

import org.apache.log4j.Logger;

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;

/\*\*

\* @author Bogdan Kaftanatiy

\*/

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

@SessionScoped

public class AccountController {

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

@EJB

private ClientBean clientDao;

@EJB

private TrainerBean trainerDao;

@EJB

private AdminBean adminDao;

private String login;

private String password;

private String name;

private String surname;

private Date dob;

private String role;

public String insert() throws NoSuchAlgorithmException {

switch (role) {

case "client":

Client client = new Client(login, password, name, surname, dob);

client.setPassword(password);

clientDao.updateObject(client);

break;

case "trainer":

Trainer trainer = new Trainer(login, password, name, surname, dob);

trainer.setPassword(password);

trainerDao.updateObject(trainer);

break;

case "admin":

Admin admin = new Admin(login, password, name, surname, dob);

admin.setPassword(password);

adminDao.updateObject(admin);

break;

default:

logger.error("Illegal account role!");

break;

}

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

}

public String toRegistration() {

login = "";

password = "";

name = "";

surname = "";

dob = null;

return "registration";

}

public void validateFrom(ComponentSystemEvent event) {

FacesContext fc = FacesContext.getCurrentInstance();

UIComponent components = event.getComponent();

String groupID = components.getClientId();

UIInput uiInputLogin = (UIInput) components.findComponent("login");

String login = uiInputLogin.getLocalValue() == null ? "" : uiInputLogin.getLocalValue().toString();

UIInput uiInputPass = (UIInput) components.findComponent("pass");

String pass = uiInputPass.getLocalValue() == null ? "" : uiInputPass.getLocalValue().toString();

UIInput uiInputName = (UIInput) components.findComponent("name");

String name = uiInputName.getLocalValue() == null ? "" : uiInputName.getLocalValue().toString();

UIInput uiInputSurname = (UIInput) components.findComponent("surname");

String surname = uiInputSurname.getLocalValue() == null ? "" : uiInputSurname.getLocalValue().toString();

if(login.isEmpty() || pass.isEmpty() || name.isEmpty() || surname.isEmpty()) {

FacesMessage msg = new FacesMessage("Fill all input fields");

msg.setSeverity(FacesMessage.SEVERITY\_ERROR);

fc.addMessage(groupID, msg);

fc.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 getSurname() {

return surname;

}

public void setSurname(String surname) {

this.surname = surname;

}

public Date getDob() {

return dob;

}

public void setDob(Date dob) {

this.dob = dob;

}

public String getRole() {

return role;

}

public void setRole(String role) {

this.role = role;

}

}

**AdminController.java**

package managedBean.admin;

import bean.ClientBean;

import bean.TrainerBean;

import entity.Client;

import entity.Trainer;

import org.apache.log4j.Logger;

import javax.ejb.EJB;

import javax.faces.bean.ManagedBean;

import javax.faces.bean.SessionScoped;

import java.util.List;

/\*\*

\* @author Bogdan Kaftanatiy

\*/

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

@SessionScoped

public class AdminController {

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

@EJB

private ClientBean clientDao;

@EJB

private TrainerBean trainerDao;

public String deleteClient(Client client) {

clientDao.deleteObject((int) client.getId());

return toClientsList();

}

public String deleteTrainer(Trainer trainer) {

trainerDao.deleteObject((int) trainer.getId());

return toTrainersList();

}

public List<Client> getClientsList() {

return clientDao.getAllClients();

}

public List<Trainer> getTrainerList() {

return trainerDao.getAll();

}

public String toClientsList() {

return "clients\_list";

}

public String toTrainersList() {

return "trainers\_list";

}

public String toRegistration() {

return "registration";

}

}

**ClientController.java**

package managedBean.client;

import bean.ClientBean;

import bean.MessageBean;

import bean.TrainerBean;

import entity.Client;

import entity.Message;

import entity.Trainer;

import managedBean.UserSession;

import org.apache.log4j.Logger;

import javax.ejb.EJB;

import javax.faces.bean.ManagedBean;

import javax.faces.bean.ManagedProperty;

import javax.faces.bean.SessionScoped;

import java.util.Calendar;

import java.util.Date;

import java.util.List;

import java.util.Locale;

/\*\*

\* @author Bogdan Kaftanatiy

\*/

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

@SessionScoped

public class ClientController {

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

@EJB

private ClientBean clientDao;

@EJB

private TrainerBean trainerDao;

@EJB

private MessageBean messageDao;

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

private UserSession userSession;

private Client currentClient;

private double minRating;

private double inputRating;

private boolean isRatingUpdate = false;

private String inputMessage;

public String sendMessage() {

Message message = new Message();

message.setMessage(inputMessage);

message.setDate(new Date());

message.setRead(false);

message.setFromAccount(getCurrentClient());

message.setToAccount(getCurrentClient().getTrainer());

messageDao.updateObject(message);

inputMessage = "";

return "message\_page";

}

public List<Message> getUserMessages() {

List<Message> result = messageDao.getAccountMessages(getCurrentClient(), getCurrentClient().getTrainer());

for (Message m : result) {

if (m.getToAccount().equals(this.getCurrentClient())) {

m.setRead(true);

messageDao.updateObject(m);

}

}

return result;

}

public String edit() {

clientDao.updateObject(currentClient);

return "client\_profile?faces-redirect=true";

}

public String getSender(Message message) {

if(message.getFromAccount().equals(getCurrentClient()))

return "You";

else

return message.getFromAccount().getName();

}

private void setClient() {

String currentLogin = userSession.getCurrentUser();

currentClient = clientDao.searchByLogin(currentLogin);

logger.info("Update current client");

}

public Client getCurrentClient(){

if(currentClient == null){

setClient();

}

return currentClient;

}

public void setCurrentClient(Client currentClient) {

this.currentClient = currentClient;

}

public String getCurrentTrainer() {

if(currentClient.getTrainer() == null) {

return "trainer is not selected";

} else {

return currentClient.getTrainer().getName() + " " + currentClient.getTrainer().getSurname();

}

}

public String filter() {

return "trainerList";

}

public String chooseTrainer(Trainer trainer) {

currentClient.setTrainer(trainer);

clientDao.updateObject(currentClient);

return "trainerPage?faces-redirect=true";

}

public String estimateTrainer() {

if(currentClient.getTrainer().getRating() == 0)

currentClient.getTrainer().setRating(inputRating);

else {

double newRating = (currentClient.getTrainer().getRating() + inputRating)/2;

currentClient.getTrainer().setRating(newRating);

}

trainerDao.updateObject(currentClient.getTrainer());

isRatingUpdate = false;

return "trainerPage?faces-redirect=true";

}

public String getDOB() {

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

birth.setTime(currentClient.getDob());

String result = birth.get(Calendar.DATE) + "." + birth.get(Calendar.MONTH) +

"." + birth.get(Calendar.YEAR);

return result;

}

public List<Trainer> getTrainerList() {

if(minRating == 0)

return trainerDao.getAll();

else

return trainerDao.getAppropriateTrainers(minRating);

}

public UserSession getUserSession() {

return userSession;

}

public void setUserSession(UserSession userSession) {

this.userSession = userSession;

}

public double getMinRating() {

return minRating;

}

public void setMinRating(double minRating) {

this.minRating = minRating;

}

public double getInputRating() {

return inputRating;

}

public void setInputRating(double inputRating) {

this.inputRating = inputRating;

}

public String getInputMessage() {

return inputMessage;

}

public void setInputMessage(String inputMessage) {

this.inputMessage = inputMessage;

}

public boolean isRatingUpdate() {

return isRatingUpdate;

}

public void updateRating() {

isRatingUpdate = true;

}

public String toProfile() {

return "client\_profile";

}

public String toEdit() {

return "editProfile";

}

public String toTrainerPage() {

return "trainerPage";

}

public String toTrainerList() {

return "trainerList";

}

public String toMessagePage() {

return "message\_page";

}

}

**ScheduleController.java**

package managedBean.client;

import bean.ScheduleUnitBean;

import entity.Client;

import entity.ScheduleUnit;

import org.apache.log4j.Logger;

import javax.annotation.PostConstruct;

import javax.ejb.EJB;

import javax.faces.bean.ManagedBean;

import javax.faces.bean.ManagedProperty;

import javax.faces.bean.SessionScoped;

import java.util.ArrayList;

import java.util.List;

/\*\*

\* @author Bogdan Kaftanatiy

\*/

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

@SessionScoped

public class ScheduleController {

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

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

private ClientController clientBean;

@EJB

private ScheduleUnitBean scheduleDao;

private List<String> weekList;

private List<ScheduleUnit> scheduleUnits;

@PostConstruct

public void init() {

weekList = new ArrayList<>();

weekList.add("Monday");

weekList.add("Tuesday");

weekList.add("Wednesday");

weekList.add("Thursday");

weekList.add("Friday");

weekList.add("Saturday");

weekList.add("Sunday");

scheduleUnits = new ArrayList<>();

for (int i = 0; i < 7; i++)

scheduleUnits.add(new ScheduleUnit());

}

public String addScheduleUnit(int index) {

ScheduleUnit currentUnit = scheduleUnits.get(index);

currentUnit.setClient(clientBean.getCurrentClient());

currentUnit.setWeekDay(index);

scheduleDao.updateObject(currentUnit);

return "schedulePage?faces-redirect=true";

}

public String deleteScheduleUnit(ScheduleUnit scheduleUnit) {

scheduleDao.deleteObject(scheduleUnit.getId());

return "schedulePage?faces-redirect=true";

}

public List<List<ScheduleUnit>> getDaySchedule() {

List<List<ScheduleUnit>> result = new ArrayList<>();

Client currentClient = clientBean.getCurrentClient();

for (int i = 0; i < 7; i++)

result.add(scheduleDao.getClientSchedule(currentClient, i));

return result;

}

public String getDayName(int index) {

return weekList.get(index);

}

public String toSchedule() {

scheduleUnits = new ArrayList<>();

for (int i = 0; i < 7; i++)

scheduleUnits.add(new ScheduleUnit());

return "schedulePage";

}

public ClientController getClientBean() {

return clientBean;

}

public void setClientBean(ClientController clientBean) {

this.clientBean = clientBean;

}

public List<ScheduleUnit> getScheduleUnits() {

return scheduleUnits;

}

public void setScheduleUnits(List<ScheduleUnit> scheduleUnits) {

this.scheduleUnits = scheduleUnits;

}

}

**TrainingController.java**

package managedBean.client;

import bean.ExerciseBean;

import bean.TrainingBean;

import entity.Exercise;

import entity.Training;

import org.apache.log4j.Logger;

import javax.ejb.EJB;

import javax.faces.bean.ManagedBean;

import javax.faces.bean.ManagedProperty;

import javax.faces.bean.SessionScoped;

import java.util.List;

/\*\*

\* @author Bogdan Kaftanatiy

\*/

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

@SessionScoped

public class TrainingController {

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

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

private ClientController clientBean;

@EJB

private TrainingBean trainingDao;

@EJB

private ExerciseBean exerciseDao;

private Exercise currentExercise;

private Training currentTraining;

public String addExercise() {

currentTraining = trainingDao.updateObject(currentTraining);

currentExercise = exerciseDao.updateObject(currentExercise);

currentTraining.addExercise(currentExercise);

exerciseDao.updateObject(currentExercise);

currentExercise = new Exercise();

return "addTrainings";

}

public String addTraining() {

currentTraining.setClient(clientBean.getCurrentClient());

trainingDao.updateObject(currentTraining);

return "trainingsList?faces-redirect=true";

}

public String deleteTraining(Training training) {

currentTraining.getExercises().remove(training);

trainingDao.deleteObject(training.getId());

currentExercise = new Exercise();

return "trainingsList?faces-redirect=true";

}

public String deleteExercise(Exercise exercise) {

exerciseDao.deleteObject(exercise.getId());

return "addTrainings";

}

public String toTrainingsList() {

return "trainingsList";

}

public String toAddTraining() {

currentTraining = new Training();

currentExercise = new Exercise();

return "addTrainings";

}

public String toUpdateTraining(Training training) {

currentTraining = training;

currentExercise = new Exercise();

return "addTrainings";

}

public List<Training> getTrainingsList() {

return trainingDao.getClientTrainings(clientBean.getCurrentClient());

}

public Exercise getCurrentExercise() {

return currentExercise;

}

public void setCurrentExercise(Exercise currentExercise) {

this.currentExercise = currentExercise;

}

public Training getCurrentTraining() {

return currentTraining;

}

public void setCurrentTraining(Training currentTraining) {

this.currentTraining = currentTraining;

}

public ClientController getClientBean() {

return clientBean;

}

public void setClientBean(ClientController clientBean) {

this.clientBean = clientBean;

}

}

**TrainerController.java**

package managedBean.trainer;

import bean.ClientBean;

import bean.MessageBean;

import bean.TrainerBean;

import bean.TrainingBean;

import entity.\*;

import managedBean.UserSession;

import org.apache.log4j.Logger;

import javax.ejb.EJB;

import javax.faces.bean.ManagedBean;

import javax.faces.bean.ManagedProperty;

import javax.faces.bean.SessionScoped;

import java.util.Calendar;

import java.util.Date;

import java.util.List;

import java.util.Locale;

/\*\*

\* @author Bogdan Kaftanatiy

\*/

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

@SessionScoped

public class TrainerController {

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

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

private UserSession userSession;

@EJB

private TrainerBean trainerDao;

@EJB

private ClientBean clientDao;

@EJB

private TrainingBean trainingDao;

@EJB

private MessageBean messageDao;

private Trainer currentTrainer;

private Client currentClient = new Client();

private String inputMessage;

public String sendMessage() {

Message message = new Message();

message.setMessage(inputMessage);

message.setDate(new Date());

message.setRead(false);

message.setFromAccount(currentTrainer);

message.setToAccount(currentClient);

messageDao.updateObject(message);

inputMessage = "";

return "messages\_page";

}

public List<Message> getUserMessages() {

List<Message> result = messageDao.getAccountMessages(getCurrentTrainer(), getCurrentClient());

for (Message m : result) {

if (m.getToAccount().equals(this.getCurrentTrainer())) {

m.setRead(true);

messageDao.updateObject(m);

}

}

return result;

}

public String getSender(Message message) {

if(message.getFromAccount().equals(this.getCurrentTrainer()))

return "You";

else

return message.getFromAccount().getName();

}

public String edit() {

trainerDao.updateObject(currentTrainer);

return "trainer\_profile?faces-redirect=true";

}

public List<Client> getClientList() {

List<Client> result = clientDao.trainerClients(getCurrentTrainer());

return result;

}

public Trainer getCurrentTrainer() {

if(currentTrainer == null)

setTrainer();

return currentTrainer;

}

public int getCurrentClientTrainingsCount() {

return trainingDao.getClientTrainings(getCurrentClient()).size();

}

public void setCurrentTrainer(Trainer currentTrainer) {

this.currentTrainer = currentTrainer;

}

private void setTrainer() {

String currentLogin = userSession.getCurrentUser();

currentTrainer = trainerDao.searchByLogin(currentLogin);

logger.info("Update current trainer");

}

public UserSession getUserSession() {

return userSession;

}

public void setUserSession(UserSession userSession) {

this.userSession = userSession;

}

public String convertDateToString(Date date) {

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

calendar.setTime(date);

String result = calendar.get(Calendar.DATE) + "." + (calendar.get(Calendar.MONTH) + 1) +

"." + calendar.get(Calendar.YEAR);

return result;

}

public List<Training> getTrainingsList() {

return trainingDao.getClientTrainings(currentClient);

}

public Client getCurrentClient() {

return currentClient;

}

public void setCurrentClient(Client currentClient) {

this.currentClient = currentClient;

}

public String getInputMessage() {

return inputMessage;

}

public void setInputMessage(String inputMessage) {

this.inputMessage = inputMessage;

}

public String toClientProfile(Client client) {

currentClient = client;

return "client\_profile";

}

public String toProfile() {

return "trainer\_profile";

}

public String toClientList() {

return "client\_list?faces-redirect=true";

}

public String toEditProfile() {

return "edit\_profile";

}

public String toMessagePage(Client client) {

currentClient = client;

inputMessage = "";

return "messages\_page";

}

public String toTrainingsPage(Client client) {

currentClient = client;

return "trainings\_page";

}

}

**persistence.xml**

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

<persistence xmlns="http://java.sun.com/xml/ns/persistence" version="2.0">

<persistence-unit name="program\_pu" transaction-type="JTA">

<provider>org.hibernate.ejb.HibernatePersistence</provider>

<jta-data-source>java:jboss/datasources/trainer\_online</jta-data-source>

<class>entity.Client</class>

<class>entity.Trainer</class>

<class>entity.Admin</class>

<properties>

<property name="hibernate.transaction.jta.platform" value="org.hibernate.service.jta.platform.internal.JBossAppServerJtaPlatform"/>

<property name="dialect" value="org.hibernate.dialect.MySQL5Dialect" />

<property name="hbm2ddl.auto" value="create"/>

</properties>

</persistence-unit>

</persistence>

**log4j.properties**

log4j.rootLogger = info, stdout

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.target=System.out

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.conversionPattern=%d{ABSOLUTE} %5p %t %c{1}:%M:%L - %m%n

**index.xhml**

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://java.sun.com/jsf/html">

<h:head>

<title>Index page</title>

<h:outputStylesheet library="css" name="index\_style.css" />

</h:head>

<h:body>

<div class="indexBlock">

<h:form>

<h:commandLink rendered="#{userSession.isLogin()}" value="To my page" action="#{userSession.toUserPage()}" />

</h:form>

<br />

<h:link rendered="#{!userSession.isLogin()}" outcome="/login" value="Login"/>

<br />

<h:link rendered="#{!userSession.isLogin()}" styleClass="register"

outcome="#{accountBean.toRegistration()}" value="Registration new account"/>

</div>

</h:body>

</html>

**login.xhtml**

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

<!DOCTYPE html

PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://java.sun.com/jsf/html"

xmlns:a="http://xmlns.jcp.org/jsf/passthrough">

<h:head>

<title>Login</title>

<h:outputStylesheet library="css" name="login\_style.css" />

</h:head>

<h:body>

<h:form styleClass="loginForm">

<h:inputText id="username" a:placeholder="Login" value="#{userSession.username}" />

<h:message for="username" />

<br />

<h:inputSecret id="password" a:placeholder="Password" value="#{userSession.password}" />

<h:message for="password" />

<br />

<h:commandButton action="#{userSession.login()}" value="Login" />

<p class="createAccount">Not registered? <h:link rendered="#{!userSession.isLogin()}"

outcome="#{accountBean.toRegistration()}" value="Create new account"/></p>

</h:form>

</h:body>

</html>

**error.html**

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

<!DOCTYPE html

PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://java.sun.com/jsf/html">

<h:head>

<title>Error</title>

<h:outputStylesheet library="css" name="error\_style.css" />

</h:head>

<h:body>

<div class="contentBlock">

<h3 style="color:red;font-weight: bold;">Sorry, Access Denied</h3>

<p>Your credentials could not be verified or you are not authorized to

view the content you desire.</p>

<p><h:link outcome="/index" value="To Home Page" /></p>

</div>

</h:body>

</html>

**registration.xtml**

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

<!DOCTYPE html

PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://java.sun.com/jsf/html"

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

xmlns:a="http://xmlns.jcp.org/jsf/passthrough">

<h:head>

<title>Registration new account</title>

<h:outputStylesheet library="css" name="registration\_style.css" />

</h:head>

<h:body>

<h:form styleClass="registerForm">

<h:panelGrid columns="1" id="RegisterGroupPanel" styleClass="registerGroup">

<f:event listener="#{accountBean.validateFrom}" type="postValidate" />

<h:panelGroup>

<h:inputText id="login" a:placeholder="Login" value="#{accountBean.login}"/>\*

</h:panelGroup>

<h:panelGroup>

<h:inputSecret id="pass" a:placeholder="Password" value="#{accountBean.password}"/>\*

</h:panelGroup>

<h:panelGroup>

<h:inputText id="name" a:placeholder="Name" value="#{accountBean.name}"/>\*

</h:panelGroup>

<h:panelGroup>

<h:inputText id="surname" a:placeholder="Surname" value="#{accountBean.surname}"/>\*

</h:panelGroup>

<h:panelGroup>

<h:inputText a:placeholder="Date of born" id="dateInput" value="#{accountBean.dob}" label="Date">

<f:convertDateTime pattern="dd-MM-yyyy" />

</h:inputText>

<br />

<h:message for="dateInput" style="color:red" />

</h:panelGroup>

<h:panelGroup>

<h:selectOneRadio id="radio" required="true" styleClass="radioBlock" value="#{accountBean.role}">

<f:selectItem itemValue="client" itemLabel="Client" />

<f:selectItem itemValue="trainer" itemLabel="Trainer" />

</h:selectOneRadio>

<br />

<h:message for="radio" style="color: red;" />

</h:panelGroup>

</h:panelGrid>

<h:message for="RegisterGroupPanel" style="color: red;" />

<h:commandButton value="Registration" action="#{accountBean.insert()}" />

</h:form>

</h:body>

</html>

**Client\_index.xhtml**

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

<!DOCTYPE html

PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://java.sun.com/jsf/html">

<h:head>

<title>Index client page</title>

<h:outputStylesheet library="css" name="client\_index.css" />

</h:head>

<h:body>

<div class="contentBlock">

<div class="leftMenu">

<p><h:link value="Profile" outcome="#{clientBean.toProfile()}" /></p>

<p><h:link value="Trainer" outcome="#{clientBean.toTrainerPage()}" /></p>

<p><h:link value="Training list" outcome="#{trainingBean.toTrainingsList()}" /></p>

<p><h:link value="Schedule" outcome="#{scheduleBean.toSchedule()}" /></p>

<p><h:link value="Important information" outcome="importantPage" /></p>

<p><h:link outcome="/index" value="To Home Page" /></p>

<h:form>

<h:commandLink value="Logout" action="#{userSession.logout()}" />

</h:form>

</div>

<div class="dataBlock">

<h1>Hi, #{clientBean.currentClient.name}!</h1>

Current user: #{userSession.currentUser}

</div>

</div>

</h:body>

</html>

**Client\_profile.xhtml**

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

<!DOCTYPE html

PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://java.sun.com/jsf/html">

<h:head>

<title>Client profile</title>

<h:outputStylesheet library="css" name="client\_index.css" />

</h:head>

<h:body>

<div class="contentBlock">

<div class="leftMenu">

<p><h:link value="Profile" outcome="#{clientBean.toProfile()}" /></p>

<p><h:link value="Trainer" outcome="#{clientBean.toTrainerPage()}" /></p>

<p><h:link value="Training list" outcome="#{trainingBean.toTrainingsList()}" /></p>

<p><h:link value="Schedule" outcome="#{scheduleBean.toSchedule()}" /></p>

<p><h:link value="Important information" outcome="importantPage" /></p>

<p><h:link outcome="/index" value="To Home Page" /></p>

<h:form>

<h:commandLink value="Logout" action="#{userSession.logout()}" />

</h:form>

</div>

<div class="dataBlock">

<h2>Your profile</h2>

<table>

<tr>

<td>Login: </td>

<td>#{clientBean.currentClient.login}</td>

</tr>

<tr>

<td>Name: </td>

<td>#{clientBean.currentClient.name}</td>

</tr>

<tr>

<td>Surname: </td>

<td>#{clientBean.currentClient.surname}</td>

</tr>

<tr>

<td>Date of birth: </td>

<td>#{clientBean.DOB}</td>

</tr>

</table>

<h:form>

<h:commandLink value="Edit" action="#{clientBean.toEdit()}" />

</h:form>

</div>

</div>

</h:body>

</html>

**trainingsList.xhtml**

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

<!DOCTYPE html

PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://java.sun.com/jsf/html"

xmlns:ui="http://xmlns.jcp.org/jsf/facelets">

<h:head>

<title>List of trainings</title>

<h:outputStylesheet library="css" name="client\_index.css" />

</h:head>

<h:body>

<div class="contentBlock">

<div class="leftMenu">

<p><h:link value="Profile" outcome="#{clientBean.toProfile()}" /></p>

<p><h:link value="Trainer" outcome="#{clientBean.toTrainerPage()}" /></p>

<p><h:link value="Training list" outcome="#{trainingBean.toTrainingsList()}" /></p>

<p><h:link value="Schedule" outcome="#{scheduleBean.toSchedule()}" /></p>

<p><h:link value="Important information" outcome="importantPage" /></p>

<p><h:link outcome="/index" value="To Home Page" /></p>

<h:form>

<h:commandLink value="Logout" action="#{userSession.logout()}" />

</h:form>

</div>

<div class="dataBlock">

<h2>Trainings list</h2>

<h:form>

<ui:repeat value="#{trainingBean.trainingsList}" var="training">

#{training.formattedDate} - #{training.description} |

<h:commandLink value="Update" action="#{trainingBean.toUpdateTraining(training)}" /> |

<h:commandLink value="Delete" action="#{trainingBean.deleteTraining(training)}" />

<table>

<thead>

<tr>

<th>Name</th>

<th>Duration</th>

<th>Description</th>

</tr>

</thead>

<ui:repeat value="#{training.exercises.toArray()}" var="exercise">

<tr>

<td>#{exercise.name}</td>

<td>#{exercise.duration}</td>

<td>#{exercise.description}</td>

</tr>

</ui:repeat>

</table>

<br />

</ui:repeat>

<h:link outcome="#{trainingBean.toAddTraining()}" value="Add new training" />

</h:form>

</div>

</div>

</h:body>

</html>

**addTranings.xhtml**

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

<!DOCTYPE html

PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://java.sun.com/jsf/html"

xmlns:ui="http://java.sun.com/jsf/facelets"

xmlns:f="http://java.sun.com/jsf/core">

<h:head>

<title>Add new training</title>

<h:outputStylesheet library="css" name="client\_index.css" />

</h:head>

<h:body>

<div class="contentBlock">

<div class="leftMenu">

<p><h:link value="Profile" outcome="#{clientBean.toProfile()}" /></p>

<p><h:link value="Trainer" outcome="#{clientBean.toTrainerPage()}" /></p>

<p><h:link value="Training list" outcome="#{trainingBean.toTrainingsList()}" /></p>

<p><h:link value="Schedule" outcome="#{scheduleBean.toSchedule()}" /></p>

<p><h:link value="Important information" outcome="importantPage" /></p>

<p><h:link outcome="/index" value="To Home Page" /></p>

<h:form>

<h:commandLink value="Logout" action="#{userSession.logout()}" />

</h:form>

</div>

<div class="dataBlock">

<h:form>

Date:

<h:inputText id="date" value="#{trainingBean.currentTraining.date}">

<f:convertDateTime pattern="dd-MM-yyyy" />

</h:inputText>

<h:message for="date" style="color: red;"/>

<br />

Description: <h:inputTextarea value="#{trainingBean.currentTraining.description}" />

<table>

<thead>

<tr>

<th>Name</th>

<th>Duration</th>

<th>Description</th>

<th>Action</th>

</tr>

</thead>

<ui:repeat value="#{trainingBean.currentTraining.exercises.toArray()}" var="exercise">

<tr>

<td>#{exercise.name}</td>

<td>#{exercise.duration}</td>

<td>#{exercise.description}</td>

<td><h:commandLink value="Delete" action="#{trainingBean.deleteExercise(exercise)}"/></td>

</tr>

</ui:repeat>

<tr>

<td><h:inputText value="#{trainingBean.currentExercise.name}" /></td>

<td>

<h:inputText id="duration" value="#{trainingBean.currentExercise.duration}" />

<h:message for="duration" style="color: red;"/>

</td>

<td><h:inputText value="#{trainingBean.currentExercise.description}" /></td>

<td><h:commandLink value="Add" action="#{trainingBean.addExercise()}"/></td>

</tr>

</table>

<h:commandLink value="Add training" action="#{trainingBean.addTraining()}" />

</h:form>

</div>

</div>

</h:body>

</html>

**trainierPage.xhtml**

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

<!DOCTYPE html

PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://java.sun.com/jsf/html">

<h:head>

<title>Trainer page</title>

<h:outputStylesheet library="css" name="client\_index.css" />

</h:head>

<h:body>

<div class="contentBlock">

<div class="leftMenu">

<p><h:link value="Profile" outcome="#{clientBean.toProfile()}" /></p>

<p><h:link value="Trainer" outcome="#{clientBean.toTrainerPage()}" /></p>

<p><h:link value="Training list" outcome="#{trainingBean.toTrainingsList()}" /></p>

<p><h:link value="Schedule" outcome="#{scheduleBean.toSchedule()}" /></p>

<p><h:link value="Important information" outcome="importantPage" /></p>

<p><h:link outcome="/index" value="To Home Page" /></p>

<h:form>

<h:commandLink value="Logout" action="#{userSession.logout()}" />

</h:form>

</div>

<div class="dataBlock">

<div>

Current trainer: #{clientBean.currentTrainer}

<h:link rendered="#{clientBean.currentClient.trainer != null}" value="Messages" outcome="#{clientBean.toMessagePage()}" />

<h:form>

<h:panelGroup rendered="#{clientBean.isRatingUpdate()}">

<h:outputLabel value="Rating: " />

<h:inputText id="rating" value="#{clientBean.inputRating}" />

<br />

<h:message for="rating" style="color: red;" />

<br />

<h:commandLink value="Update" action="#{clientBean.estimateTrainer()}" />

</h:panelGroup>

<h:commandLink

rendered="#{!clientBean.isRatingUpdate() and clientBean.currentClient.trainer != null}"

value="Estimate tainer" action="#{clientBean.updateRating()}" />

<br />

<h:commandLink rendered="#{!clientBean.isRatingUpdate()}" value="Change trainer" action="#{clientBean.toTrainerList()}" />

</h:form>

</div>

</div>

</div>

</h:body>

</html>

**messagePage.xhtml**

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

<!DOCTYPE html

PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://java.sun.com/jsf/html"

xmlns:ui="http://java.sun.com/jsf/facelets">

<h:head>

<title>Messages page</title>

<h:outputStylesheet library="css" name="client\_index.css" />

</h:head>

<h:body>

<div class="contentBlock">

<div class="leftMenu">

<p><h:link value="Profile" outcome="#{clientBean.toProfile()}" /></p>

<p><h:link value="Trainer" outcome="#{clientBean.toTrainerPage()}" /></p>

<p><h:link value="Training list" outcome="#{trainingBean.toTrainingsList()}" /></p>

<p><h:link value="Schedule" outcome="#{scheduleBean.toSchedule()}" /></p>

<p><h:link value="Important information" outcome="importantPage" /></p>

<p><h:link outcome="/index" value="To Home Page" /></p>

<h:form>

<h:commandLink value="Logout" action="#{userSession.logout()}" />

</h:form>

</div>

<div class="dataBlock">

<h2>Dialog with #{clientBean.currentClient.trainer.name}</h2>

<ui:repeat value="#{clientBean.userMessages}" var="message" varStatus="status">

<div>

#{clientBean.getSender(message)}: #{message.message} -

#{message.date.date}.#{message.date.month}.#{message.date.year}

</div>

</ui:repeat>

<h:form>

<h:inputTextarea value="#{clientBean.inputMessage}" />

<h:commandButton value="Send" action="#{clientBean.sendMessage()}" />

</h:form>

</div>

</div>

</h:body>

</html>

**Trainier\_index.xhtml**

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

<!DOCTYPE html

PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://java.sun.com/jsf/html"

xmlns:f="http://java.sun.com/jsf/core">

<h:head>

<title>Index trainer page</title>

<h:outputStylesheet library="css" name="trainer\_index.css" />

</h:head>

<h:body>

<div class="contentBlock">

<div class="leftMenu">

<p><h:link value="Profile" outcome="#{trainerBean.toProfile()}" /></p>

<p><h:link value="Client list" outcome="#{trainerBean.toClientList()}" /></p>

<p><h:link outcome="/index" value="To Home Page" /></p>

<h:form>

<h:commandLink value="Logout" action="#{userSession.logout()}" />

</h:form>

</div>

<div class="dataBlock">

<h1>Hi, #{trainerBean.currentTrainer.name}!</h1>

Current user: #{userSession.currentUser}

</div>

</div>

</h:body>

</html>

**clientList.xhtml**

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

<!DOCTYPE html

PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://java.sun.com/jsf/html"

xmlns:ui="http://java.sun.com/jsf/facelets">

<h:head>

<title>Client list</title>

<h:outputStylesheet library="css" name="trainer\_index.css" />

</h:head>

<h:body>

<div class="contentBlock">

<div class="leftMenu">

<p><h:link value="Profile" outcome="#{trainerBean.toProfile()}" /></p>

<p><h:link value="Client list" outcome="#{trainerBean.toClientList()}" /></p>

<p><h:link outcome="/index" value="To Home Page" /></p>

<h:form>

<h:commandLink value="Logout" action="#{userSession.logout()}" />

</h:form>

</div>

<div class="dataBlock">

<h1>Client list</h1>

<ui:repeat value="#{trainerBean.clientList}" var="client" varStatus="status">

<div>

#{status.index + 1}. #{client.name} #{client.surname} - #{client.age} years old

<h:form>

<h:commandLink value="Profile"

action="#{trainerBean.toClientProfile(trainerBean.clientList.get(status.index))}" /> |

<h:commandLink value="Messages"

action="#{trainerBean.toMessagePage(trainerBean.clientList.get(status.index))}" /> |

<h:commandLink value="Trainings"

action="#{trainerBean.toTrainingsPage(trainerBean.clientList.get(status.index))}" />

</h:form>

</div>

<br />

</ui:repeat>

</div>

</div>

</h:body>

</html>

**trainingsPage.xhtml**

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

<!DOCTYPE html

PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://java.sun.com/jsf/html"

xmlns:ui="http://java.sun.com/jsf/facelets">

<h:head>

<title>Client trainings page</title>

<h:outputStylesheet library="css" name="trainer\_index.css" />

</h:head>

<h:body>

<div class="contentBlock">

<div class="leftMenu">

<p><h:link value="Profile" outcome="#{trainerBean.toProfile()}" /></p>

<p><h:link value="Client list" outcome="#{trainerBean.toClientList()}" /></p>

<p><h:link outcome="/index" value="To Home Page" /></p>

<h:form>

<h:commandLink value="Logout" action="#{userSession.logout()}" />

</h:form>

</div>

<div class="dataBlock">

<h2>#{trainerBean.currentClient.name} trainings list</h2>

<h:form>

<ui:repeat value="#{trainerBean.trainingsList}" var="training">

#{training.formattedDate} - #{training.description}

<table>

<thead>

<tr>

<th>Name</th>

<th>Duration</th>

<th>Description</th>

</tr>

</thead>

<ui:repeat value="#{training.exercises.toArray()}" var="exercise">

<tr>

<td>#{exercise.name}</td>

<td>#{exercise.duration}</td>

<td>#{exercise.description}</td>

</tr>

</ui:repeat>

</table>

<br />

</ui:repeat>

<h:link outcome="#{trainerBean.toClientList()}" value="Client list" />

</h:form>

</div>

</div>

</h:body>

</html>

**Admin\_index.xhtml**

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

<!DOCTYPE html

PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://java.sun.com/jsf/html">

<h:head>

<title>Index admin page</title>

<h:outputStylesheet library="css" name="admin\_index.css" />

</h:head>

<h:body>

<div class="contentBlock">

<div class="leftMenu">

<p><h:link value="Clients list" outcome="#{adminBean.toClientsList()}" /></p>

<p><h:link value="Trainers list" outcome="#{adminBean.toTrainersList()}" /></p>

<p><h:link value="Register new account" outcome="#{adminBean.toRegistration()}" /></p>

<h:form>

<h:commandLink value="Send important information" action="#{jmsService.sendMessage()}" />

<p><h:link outcome="/index" value="To Home Page" /></p>

<h:commandLink value="Logout" action="#{userSession.logout()}" />

</h:form>

</div>

<div class="dataBlock">

<h1>Hi admin!</h1>

Current user: #{userSession.currentUser}

</div>

</div>

</h:body>

</html>

**web.xml**

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

<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"

version="3.1">

<servlet>

<servlet-name>Faces Servlet</servlet-name>

<servlet-class>javax.faces.webapp.FacesServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>Faces Servlet</servlet-name>

<url-pattern>\*.xhtml</url-pattern>

</servlet-mapping>

<welcome-file-list>

<welcome-file>faces/index.xhtml</welcome-file>

</welcome-file-list>

<error-page>

<error-code>403</error-code>

<location>/error.xhtml</location>

</error-page>

<session-config>

<session-timeout>30</session-timeout>

</session-config>

<security-constraint>

<web-resource-collection>

<web-resource-name>admin</web-resource-name>

<url-pattern>/admin/\*</url-pattern>

<http-method>GET</http-method>

<http-method>POST</http-method>

</web-resource-collection>

<auth-constraint>

<role-name>ADMIN</role-name>

</auth-constraint>

</security-constraint>

<security-constraint>

<web-resource-collection>

<web-resource-name>trainer</web-resource-name>

<url-pattern>/trainer/\*</url-pattern>

<http-method>GET</http-method>

<http-method>POST</http-method>

</web-resource-collection>

<auth-constraint>

<role-name>TRAINER</role-name>

</auth-constraint>

</security-constraint>

<security-constraint>

<web-resource-collection>

<web-resource-name>client</web-resource-name>

<url-pattern>/client/\*</url-pattern>

<http-method>GET</http-method>

<http-method>POST</http-method>

</web-resource-collection>

<auth-constraint>

<role-name>CLIENT</role-name>

</auth-constraint>

</security-constraint>

<security-role>

<role-name>ADMIN</role-name>

</security-role>

<security-role>

<role-name>CLIENT</role-name>

</security-role>

<security-role>

<role-name>TRAINER</role-name>

</security-role>

<login-config>

<auth-method>FORM</auth-method>

<form-login-config>

<form-login-page>/login.xhtml</form-login-page>

<form-error-page>/error.xhtml</form-error-page>

</form-login-config>

</login-config>

<security-constraint>

<web-resource-collection>

<web-resource-name>Exclude from Security</web-resource-name>

<url-pattern>/resources/css/\*</url-pattern>

</web-resource-collection>

</security-constraint>

</web-app>

**pom.xml**

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

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.kaftanatiy.apps</groupId>

<artifactId>OnlineTrainerWebApp</artifactId>

<version>1.0-SNAPSHOT</version>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<source>1.7</source>

<target>1.7</target>

</configuration>

</plugin>

</plugins>

</build>

<dependencies>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-entitymanager</artifactId>

<version>4.2.1.Final</version>

</dependency>

<dependency>

<groupId>org.hibernate.javax.persistence</groupId>

<artifactId>hibernate-jpa-2.0-api</artifactId>

<version>1.0.1.Final</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.25</version>

</dependency>

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.12</version>

</dependency>

<dependency>

<groupId>javax.faces</groupId>

<artifactId>javax.faces-api</artifactId>

<version>2.1</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

<dependency>

<groupId>org.jboss.spec.javax.jms</groupId>

<artifactId>jboss-jms-api\_2.0\_spec</artifactId>

<version>1.0.0.Final</version>

</dependency>

<dependency>

<groupId>javax.inject</groupId>

<artifactId>javax.inject</artifactId>

<version>1</version>

</dependency>

</dependencies>

</project>