Application

1)Application

package com.taekwondoacademy.taekwondoacademy;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class TaekwondoacademyApplication {

public static void main(String[] args) {

SpringApplication.run(TaekwondoacademyApplication.class, args);

}

}

Controller

1)Taekwondo Controller

package com.taekwondoacademy.taekwondoacademy.Controller;

import com.taekwondoacademy.taekwondoacademy.Mails.MailSenderService;

import com.taekwondoacademy.taekwondoacademy.Models.CurrentLoginUser;

import com.taekwondoacademy.taekwondoacademy.Models.PaymentDetails;

import com.taekwondoacademy.taekwondoacademy.Models.UserRegister;

import com.taekwondoacademy.taekwondoacademy.Services.TaekWondService;

import lombok.AllArgsConstructor;

import org.apache.commons.lang.RandomStringUtils;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.servlet.ModelAndView;

@Controller

@AllArgsConstructor

public class TaekController {

private final TaekWondService service;

private final MailSenderService mail;

@RequestMapping(path = "/register")

public ModelAndView register() {

var model = new ModelAndView("Register");

model.addObject("reg", new UserRegister());

return model;

}

@RequestMapping(path = "/saveRegister")

public String saveRegister(@ModelAttribute("reg") UserRegister user) {

if (user != null) {

user.setUserId(generateUserId());

mail.sendEmail(user.getEmail(), "Congratulation", userBody(user));

service.savePayment(new PaymentDetails(user.getUserId(),500,"Not Paid",null));

service.saveUserRegister(user);

return "thanks";

}

return "error";

}

@RequestMapping(path = "/all\_user")

public ModelAndView all\_user() {

var model = new ModelAndView("All\_member");

model.addObject("allUser", service.listOfMember());

model.addObject("user", service.currentLoginUser());

return model;

}

@RequestMapping(path = "/contact")

public ModelAndView contact(){

var model=new ModelAndView("contactUs");

model.addObject("user", service.currentLoginUser());

return model;

}

@RequestMapping(path ="/about")

public ModelAndView about(){

var model=new ModelAndView("about");

model.addObject("user", service.currentLoginUser());

return model;

}

private String generateUserId() {

return RandomStringUtils.randomAlphanumeric(5);

}

private String userBody(UserRegister user) {

var sb = new StringBuilder();

return sb

.append("\t\t\tRegister Successfully")

.append("\nHai...\t")

.append(user.getSName())

.append("\nUser id:\t")

.append(user.getUserId())

.append("\npassword\t")

.append(user.getPassword())

.append("\n\t\t\tThanks").toString();

}

@RequestMapping(path = "/payment")

public ModelAndView payment() {

var model = new ModelAndView("paymentList");

model.addObject("notPay", service.getNotPaidStudent());

model.addObject("user", service.currentLoginUser());

return model;

}

@RequestMapping(path = "/pay/{id}")

public ModelAndView pay(@PathVariable("id") Long id) {

if (id != null) {

service.updatePayment(id,generateTransactionId());

mail.sendEmail(service.getUser(service.getPaymentDetails(id).getUserId()).getEmail(),"Payment Successfully Done",paymentBody(service.getPaymentDetails(id)));

var model=new ModelAndView("paymentList");

model.addObject("user",service.currentLoginUser());

model.addObject("notPay",service.getNotPaidStudent());

return model;

}

return new ModelAndView("error");

}

@RequestMapping(path = "/login")

public ModelAndView login() {

var login = new ModelAndView("Login");

login.addObject("log", new UserRegister());

return login;

}

@RequestMapping(path = "/")

public String home() {

return "fragements/UserHome";

}

@RequestMapping(path = "/checkLog")

public ModelAndView redirectLog(@ModelAttribute("log") UserRegister user) {

if (user != null) {

if (service.isUser(user)) {

service.deleteCurrentUser();

service.saveCurrentUser(service.getUser(user));

if (service.isAdmin(user)) {

var admin = new ModelAndView("adminHome");

admin.addObject("admin", service.getUser(user));

admin.addObject("user", service.currentLoginUser());

return admin;

}

var useHome = new ModelAndView("fragements/secondnavigation");

useHome.addObject("home", service.getUser(user));

useHome.addObject("user", service.currentLoginUser());

return useHome;

}

}

return new ModelAndView("error");

}

@RequestMapping(path = "/mypayment")

public ModelAndView myPayment() {

var model = new ModelAndView("mypayment");

CurrentLoginUser currentLoginUser = service.currentLoginUser();

model.addObject("myPay", service.yourPayment(currentLoginUser.getUserId()));

model.addObject("user", currentLoginUser);

return model;

}

@RequestMapping(path = "/reset")

public String reset() {

service.resetPayment();

return "redirect:/payment";

}

private String generateTransactionId(){

return RandomStringUtils.randomAlphanumeric(10);

}

private String paymentBody(PaymentDetails pay){

var sb=new StringBuilder();

return sb

.append("\t\t\tCongratulation")

.append("\nHaii...\t")

.append(service.getUser(pay.getUserId()).getSName())

.append("\nUser Transaction Id is:\t")

.append(pay.getTransactionId())

.append("\n\t\t\tThanks").toString();

}

@RequestMapping(path ="/userhome")

public ModelAndView userHome(){

var model = new ModelAndView("fragements/secondnavigation");

model.addObject("user", service.currentLoginUser());

return model;

}

@RequestMapping(path ="/adminhome")

public ModelAndView adminHome(){

var model = new ModelAndView("adminHome");

model.addObject("user", service.currentLoginUser());

return model;

}

}

Mail Service

1)MailService

package com.taekwondoacademy.taekwondoacademy.Mails;

import lombok.AllArgsConstructor;

import org.springframework.mail.SimpleMailMessage;

import org.springframework.mail.javamail.JavaMailSender;

import org.springframework.stereotype.Service;

@Service

@AllArgsConstructor

public class MailSenderService {

private final JavaMailSender javaMailSender;

public void sendEmail(String toEmail, String subject, String body) {

SimpleMailMessage message = new SimpleMailMessage();

message.setFrom("onlinevehicleinvestigation@gmail.com");

message.setText(body);

message.setSubject(subject);

message.setTo(toEmail);

javaMailSender.send(message);

}

}

Models

1)UserRegister

package com.taekwondoacademy.taekwondoacademy.Models;

import lombok.\*;

import org.springframework.format.annotation.DateTimeFormat;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import java.time.LocalDate;

@Setter

@Getter

@AllArgsConstructor

@NoArgsConstructor

@ToString

@Entity

public class UserRegister {

@Id

@GeneratedValue(strategy= GenerationType.IDENTITY)

private Long id;

private String sName;

private String gender;

private Integer age;

private String email;

private String phone;

private String password;

@DateTimeFormat(pattern = "yyyy-MM-dd")

private LocalDate dob;

private String address;

private String userId;

public UserRegister(String sName, String userId) {

this.sName = sName;

this.userId = userId;

}

}

2)User Role

package com.taekwondoacademy.taekwondoacademy.Models;

import lombok.\*;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Getter

@Setter

@AllArgsConstructor

@NoArgsConstructor

@ToString

@Entity

public class UserRole {

@Id

@GeneratedValue(strategy= GenerationType.IDENTITY)

private Long id;

private String userId;

private String role;

}

3)Payment Details

package com.taekwondoacademy.taekwondoacademy.Models;

import lombok.\*;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Setter

@Getter

@AllArgsConstructor

@NoArgsConstructor

@ToString

@Entity

public class PaymentDetails {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String userId;

private Integer amount;

private String payStatus;

private String transactionId;

public PaymentDetails(String userId, Integer amount, String payStatus, String transactionId) {

this.userId = userId;

this.amount = amount;

this.payStatus = payStatus;

this.transactionId = transactionId;

}

}

3)CurrentLoginUser

package com.taekwondoacademy.taekwondoacademy.Models;

import lombok.\*;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Setter

@Getter

@AllArgsConstructor

@NoArgsConstructor

@ToString

@Entity

public class CurrentLoginUser {

@Id

@GeneratedValue(strategy= GenerationType.IDENTITY)

private Long id;

private String UserId;

private String name;

public CurrentLoginUser(String userId, String name) {

UserId = userId;

this.name = name;

}

}

4)CompetitonModels

package com.taekwondoacademy.taekwondoacademy.Models;

import lombok.\*;

import org.springframework.format.annotation.DateTimeFormat;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import java.time.LocalDate;

@Getter

@Setter

@AllArgsConstructor

@NoArgsConstructor

@ToString

@Entity

public class Competiton {

@Id

@GeneratedValue(strategy= GenerationType.IDENTITY)

private Long id;

private String comName;

private String comPlace;

@DateTimeFormat(pattern = "yyyy-MM-dd")

private LocalDate date;

}

5)achievement

package com.taekwondoacademy.taekwondoacademy.Models;

import lombok.\*;

import javax.persistence.Entity;

import javax.persistence.Id;

@Getter

@Setter

@AllArgsConstructor

@NoArgsConstructor

@ToString

@Entity

public class AchievementsList {

@Id

private Long id;

private String sName;

private String tComp;

private String typeBelt;

}

Repository

1) UserRoleRepository

package com.taekwondoacademy.taekwondoacademy.Repository;

import com.taekwondoacademy.taekwondoacademy.Models.UserRole;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

@Repository

public interface UserRoleRepository extends JpaRepository<UserRole,Long> {

}

2)User Register Repository

package com.taekwondoacademy.taekwondoacademy.Repository;

import com.taekwondoacademy.taekwondoacademy.Models.UserRegister;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

@Repository

public interface UserRegisterRepository extends JpaRepository<UserRegister,Long> {

}

3) PaymentDetails Repository

package com.taekwondoacademy.taekwondoacademy.Repository;

import com.taekwondoacademy.taekwondoacademy.Models.PaymentDetails;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

@Repository

public interface PaymentDetailsRepository extends JpaRepository<PaymentDetails,Long> {

}

4)Current Login User Repository

package com.taekwondoacademy.taekwondoacademy.Repository;

import com.taekwondoacademy.taekwondoacademy.Models.CurrentLoginUser;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

@Repository

public interface CurrentLoginUserRepository extends JpaRepository<CurrentLoginUser,Long> {

}

5)Competiton Repository

package com.taekwondoacademy.taekwondoacademy.Repository;

import com.taekwondoacademy.taekwondoacademy.Models.Competiton;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

@Repository

public interface CompetitionRepository extends JpaRepository<Competiton,Long> {

}

6)Achievement Repository

package com.taekwondoacademy.taekwondoacademy.Repository;

import com.taekwondoacademy.taekwondoacademy.Models.AchievementsList;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

@Repository

public interface AchievementsListRepository extends JpaRepository<AchievementsList,Long> {

}

Services

1)Services

package com.taekwondoacademy.taekwondoacademy.Services;

import com.taekwondoacademy.taekwondoacademy.Models.\*;

import java.util.List;

public interface TaekWondService {

void saveUserRegister(UserRegister user);

void savePayment(PaymentDetails pay);

void upcomingCompetition(Competiton upCompetition);

void saveAchievement(AchievementsList list);

List<UserRegister> listOfMember();

List<AchievementsList> listOfAchiever();

void saveUserRole(UserRole role);

List<PaymentDetails> getNotPaidStudent();

CurrentLoginUser currentLoginUser();

boolean deleteCurrentUser();

void saveCurrentUser(UserRegister user);

boolean isUser(UserRegister user);

boolean isAdmin(UserRegister user);

boolean resetPayment();

boolean deleteById(Long id);

UserRegister getUser(UserRegister user);

PaymentDetails yourPayment(String id);

PaymentDetails getPaymentDetails(Long id);

boolean updatePayment(Long id,String t\_Id);

UserRegister getUser(String userId);

}2)Service Implements

package com.taekwondoacademy.taekwondoacademy.Services;

import com.taekwondoacademy.taekwondoacademy.Models.\*;

import com.taekwondoacademy.taekwondoacademy.Repository.\*;

import lombok.AllArgsConstructor;

import org.springframework.stereotype.Service;

import java.util.ArrayList;

import java.util.List;

import java.util.Optional;

import java.util.stream.Collectors;

@Service

@AllArgsConstructor

public class TaekWondServiceImp implements TaekWondService {

private final UserRegisterRepository userRegisterRepository;

private final PaymentDetailsRepository paymentDetails;

private final AchievementsListRepository achievementsListRepository;

private final UserRoleRepository userRoleRepository;

private final CompetitionRepository competitionRepository;

private final CurrentLoginUserRepository currentLoginUserRepository;

@Override

public void saveUserRegister(UserRegister user) {

userRegisterRepository.save(user);

}

@Override

public List<UserRegister> listOfMember() {

return userRegisterRepository.findAll();

}

@Override

public void savePayment(PaymentDetails pay) {

paymentDetails.save(pay);

}

@Override

public void upcomingCompetition(Competiton upCompetition) {

competitionRepository.save(upCompetition);

}

@Override

public void saveAchievement(AchievementsList list) {

achievementsListRepository.save(list);

}

@Override

public List<AchievementsList> listOfAchiever() {

return achievementsListRepository.findAll();

}

@Override

public void saveUserRole(UserRole role) {

userRoleRepository.save(role);

}

@Override

public List<PaymentDetails> getNotPaidStudent() {

return paymentDetails.findAll().stream().filter(p -> p.getPayStatus().equals("Not Paid")).collect(Collectors.toList());

}

@Override

public CurrentLoginUser currentLoginUser() {

return currentLoginUserRepository.findAll().get(0);

}

@Override

public boolean deleteCurrentUser() {

currentLoginUserRepository.deleteAll();

return true;

}

@Override

public void saveCurrentUser(UserRegister user) {

currentLoginUserRepository.save(new CurrentLoginUser(user.getUserId(),user.getSName()));

}

@Override

public boolean isUser(UserRegister user) {

Optional<UserRegister> isUser = userRegisterRepository

.findAll()

.stream()

.filter(p -> p.getUserId().equals(user.getUserId()))

.findAny();

return isUser.isPresent();

}

@Override

public boolean isAdmin(UserRegister user) {

Optional<UserRole> any = userRoleRepository

.findAll()

.stream()

.filter(p -> p.getUserId().equals(user.getUserId()))

.findAny();

return any.map(userRole -> userRole.getRole().equals("Admin")).orElse(false);

}

@Override

public boolean resetPayment() {

paymentDetails.deleteAll();

List<PaymentDetails> paymentDetails1=new ArrayList<PaymentDetails>();

List<UserRegister> all = userRegisterRepository.findAll();

for(UserRegister user:all){

paymentDetails1.add(new PaymentDetails(user.getUserId(),500,"Not Paid",null));

}

paymentDetails.saveAll(paymentDetails1);

return true;

}

@Override

public boolean deleteById(Long id) {

Optional<PaymentDetails> delete = paymentDetails.findAll().stream().filter(p -> p.getId().equals(id)).findAny();

delete.ifPresent(details -> paymentDetails.deleteById(details.getId()));

return true;

}

@Override

public UserRegister getUser(UserRegister user) {

Optional<UserRegister> any = userRegisterRepository.findAll().stream().filter(p -> p.getUserId().equals(user.getUserId())).findAny();

return any.orElseGet(UserRegister::new);

}

@Override

public PaymentDetails yourPayment(String id) {

Optional<PaymentDetails> yourPayment=paymentDetails.findAll().stream().filter(p->p.getUserId().equals(id)).findAny();

return yourPayment.orElseGet(PaymentDetails::new);

}

@Override

public PaymentDetails getPaymentDetails(Long id) {

Optional<PaymentDetails> getPaymentDetails=paymentDetails.findById(id);

return getPaymentDetails.orElseGet(PaymentDetails::new);

}

@Override

public boolean updatePayment(Long id,String transactionId) {

Optional<PaymentDetails> byId = paymentDetails.findById(id);

if(byId.isPresent())

{

byId.get().setTransactionId(transactionId);

byId.get().setPayStatus("Paid");

savePayment(byId.get());

}

return false;

}

@Override

public UserRegister getUser(String userId) {

Optional<UserRegister> any = userRegisterRepository.findAll().stream().filter(p -> p.getUserId().equals(userId)).findAny();

return any.orElseGet(UserRegister::new);

}

}