**Ketan Malik**

**NUID – 001378783**

**Web Development Tools & Methods**

**“BookAnEvent – Book your next event through us”**

**Overview**

The project aims at providing an online marketplace, *BookAnEvent*, for finding and booking events such as movies, plays, and musicals using this web application. It would help customers to browse events happening near them and then book tickets for the same. Furthermore, it also provides a platform for admins to add and manage events, venues and shows for the customers to book. Some of the key features of the project are:

1. Events that can be booked are – movies, plays, and musical shows
2. Customers can search for events based on their city
3. Admins would perform all the CRUD operations on the events being offered
4. Admins would perform all the CRUD operations on the venues being offered
5. Admins would create shows for different events at different times for a particular venue

**Supported Roles & Functionalities**

**Role 1 – Customer**

1. Ability to Register and Login
2. Browse events based on the city and event name
3. Book a particular event
4. View past and upcoming bookings
5. Ability to cancel the booking (only if the show date is atleast after three days from current date)
6. Ability to download a PDF document for the booking

**Role 2 – Admin**

1. Create new events in the system
2. Modify or Delete existing events
3. Create new venues in the system
4. Modify or Delete existing venues
5. Create new shows for a particular event and venue in the system

**Managing session integrity** – If suppose there are 5 seats left for a particular show and two different customers are trying to **book those 5 seats at the same time** then the customer to book first will get a confirmed booking and the second customer will get an error saying the seats were filled up.

**Technology**

1. Frontend – HTML, CSS, JavaScript, Ajax, JSP, Bootstrap
2. Backend – Spring MVC, Hibernate, Spring Forms & Validators, HQL
3. Database – MySQL

**Home Page**

**Graphical user interface, text, application

Description automatically generated**

**User Sign Up Form**

Graphical user interface, website

Description automatically generated

**User Flow – Choose city and event**

Graphical user interface, website

Description automatically generated

**User Flow - Select show time at a particular venue**

Graphical user interface, website

Description automatically generated

**User Flow – Enter total seats and phone number for booking**

Graphical user interface

Description automatically generated

**User Flow – Booking Confirmation**

Graphical user interface

Description automatically generated

**User Flow – Booking Confirmation PDF**

Graphical user interface, text, application

Description automatically generated

**User Flow – View and cancel bookings**

Graphical user interface, website

Description automatically generated

**Admin Flow – View all the events**

Graphical user interface, website

Description automatically generated

**Admin Flow – Updating an existing event**

Graphical user interface, website

Description automatically generated

**Admin Flow – Adding a new event**

Graphical user interface, website

Description automatically generated

**Admin Flow – View all the venues**

Graphical user interface

Description automatically generated

**Admin Flow – Updating an existing venue**

Graphical user interface, website

Description automatically generated

**Admin Flow – Adding a new venue**

Graphical user interface, website

Description automatically generated

**Admin Flow – Adding a new show at a particular venue for a particular event**

Graphical user interface, website

Description automatically generated

**Validations – While updating an event**

Graphical user interface, website

Description automatically generated

**Validations – While adding a new show**

Graphical user interface, website

Description automatically generated

**Validations – While signing up as a new user**

Graphical user interface, website

Description automatically generated

**Validations – While trying to book more than available number of seats**

Graphical user interface, website

Description automatically generated

**Validations – Trying to cancel a booking which is within next 3 days**

Graphical user interface, website

Description automatically generated

**Validations – While two customers trying to book the same show**

Graphical user interface, application

Description automatically generated

**AdminEventController**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package com.mycompany.controller;

import com.mycompany.dao.EventDao;

import com.mycompany.pojo.Event;

import com.mycompany.validator.UpdateEventValidator;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.WebDataBinder;

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

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

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

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

import org.springframework.web.servlet.ModelAndView;

/\*\*

\*

\* @author ketanmalik

\*/

@Controller

public class AdminEventController {

@Autowired

UpdateEventValidator updateEventValidator;

@InitBinder("event")

protected void initBinder(WebDataBinder binder) {

binder.setValidator(updateEventValidator);

}

@PostMapping("/add-event.htm")

public ModelAndView addEvent(HttpServletRequest request, HttpServletResponse response, HttpSession session) {

if (invalidSessionObj(session, "user")) {

return sessionTimedOut(request);

}

return new ModelAndView("add-event-view");

}

@PostMapping("/delete-event.htm")

public ModelAndView deleteEvent(HttpServletRequest request, HttpServletResponse response, HttpSession session, EventDao eventDao) {

if (invalidSessionObj(session, "user")) {

return sessionTimedOut(request);

}

int res = eventDao.deleteEvent(Integer.parseInt(request.getParameter("delete-event")));

if (res > 0) {

List<Event> eventList = eventDao.getEventList();

session.setAttribute("eventList", eventList);

}

return new ModelAndView("manage-events-view");

}

@PostMapping("event-added.htm")

public ModelAndView saveEvent(HttpServletRequest request, HttpServletResponse response, HttpSession session, EventDao eventDao) throws ParseException {

if (invalidSessionObj(session, "user")) {

return sessionTimedOut(request);

}

String event\_name = request.getParameter("event-name");

String event\_type = request.getParameter("event-type");

String event\_cast = request.getParameter("event-cast");

String event\_rating = request.getParameter("event-rating");

String event\_genre = request.getParameter("event-genre");

String event\_language = request.getParameter("event-language");

String event\_summary = request.getParameter("event-summary");

String event\_duration = request.getParameter("event-duration");

String event\_date = request.getParameter("event-date");

if (event\_name.trim().equals("") || event\_type.trim().equals("") || event\_cast.trim().equals("") || event\_rating.trim().equals("") || event\_genre.trim().equals("")

|| event\_language.trim().equals("") || event\_summary.trim().equals("") || event\_summary.length() > 2000 || event\_duration.trim().equals("") || event\_date.trim().equals("")) {

request.setAttribute("errorMsg1", "New event could not be added.");

request.setAttribute("errorMsg2", "Some of the fields you entered were invalid. Please try again.");

return new ModelAndView("error-view");

}

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

int res = eventDao.saveEvent(event\_name, event\_type, event\_cast, event\_rating, event\_genre,

event\_language, event\_summary, event\_duration, formatter.parse(event\_date));

if (res > 0) {

List<Event> eventList = eventDao.getEventList();

session.setAttribute("eventList", eventList);

request.setAttribute("successMsg1", "New event added successfully.");

request.setAttribute("successMsg2", "Please go back to dashboard to manage this newly added event.");

return new ModelAndView("success-view");

}

return serverError(request, "New event could not be added.");

}

@PostMapping("/updated-event.htm")

public ModelAndView saveUpdatedEvent(@ModelAttribute("updateEventForm") Event event, BindingResult result, Model model, HttpServletRequest request, HttpSession session, EventDao eventDao) {

updateEventValidator.validate(event, result);

if (result.hasErrors()) {

return new ModelAndView("update-event-view");

} else {

Event sessionEvent = (Event) session.getAttribute("event");

int res = eventDao.updateEvent(sessionEvent.getEvent\_id(), event);

if (res == 1) {

event.setEvent\_id(sessionEvent.getEvent\_id());

session.removeAttribute("event");

session.setAttribute("event", event);

Event obj = (Event) session.getAttribute("event");

request.setAttribute("successMsg1", "Event updated successfully.");

request.setAttribute("successMsg2", "Please go back to dashboard to manage events, users, and venues.");

return new ModelAndView("success-view");

} else {

return serverError(request, "The event could not be updated.");

}

}

}

@GetMapping("/manage-events.htm")

public ModelAndView signUpPage(HttpServletRequest request, HttpSession session) {

if (invalidSessionObj(session, "user")) {

return sessionTimedOut(request);

}

return new ModelAndView("manage-events-view");

}

@PostMapping("/manage-events.htm")

public ModelAndView manageEvents(HttpServletRequest request, HttpServletResponse response, HttpSession session) {

if (invalidSessionObj(session, "user")) {

return sessionTimedOut(request);

}

return new ModelAndView("manage-events-view");

}

@PostMapping("/update-event.htm")

public ModelAndView updateEvent(HttpServletRequest request, HttpServletResponse response, HttpSession session, EventDao eventDao, Model model) {

if (invalidSessionObj(session, "user")) {

return sessionTimedOut(request);

}

int event\_id = Integer.parseInt(request.getParameter("update-event"));

Event event = eventDao.getEvent(event\_id);

if (event == null) {

if (session.getAttribute("event") != null) {

session.removeAttribute("event");

}

return serverError(request, "The event could not be updated.");

}

session.setAttribute("event", event);

Event e = new Event();

e.setEvent\_id(event\_id);

e.setEvent\_name(event.getEvent\_name());

e.setEvent\_type(event.getEvent\_type());

e.setEvent\_cast(event.getEvent\_cast());

e.setEvent\_rating(event.getEvent\_rating());

e.setEvent\_genre(event.getEvent\_genre());

e.setEvent\_language(event.getEvent\_language());

e.setEvent\_summary(event.getEvent\_summary());

e.setEvent\_duration(event.getEvent\_duration());

e.setEvent\_date(event.getEvent\_date());

model.addAttribute("updateEventForm", e);

return new ModelAndView("update-event-view");

}

public boolean invalidSessionObj(HttpSession session, String obj) {

return session.getAttribute(obj) == null;

}

public ModelAndView sessionTimedOut(HttpServletRequest request) {

request.setAttribute("errorMsg1", "Your session has been timed out.");

request.setAttribute("errorMsg2", "Please log in again to start a new session.");

return new ModelAndView("error-view");

}

public ModelAndView serverError(HttpServletRequest request, String msg1) {

request.setAttribute("errorMsg1", msg1);

request.setAttribute("errorMsg2", "There was a problem in reaching out to our servers. Please try again later.");

return new ModelAndView("error-view");

}

}

**AdminShowController**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package com.mycompany.controller;

import com.mycompany.dao.EventDao;

import com.mycompany.dao.ShowDao;

import com.mycompany.dao.VenueDao;

import com.mycompany.pojo.Event;

import com.mycompany.pojo.Show;

import com.mycompany.pojo.Venue;

import com.mycompany.validator.ShowValidator;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.WebDataBinder;

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

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

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

import org.springframework.web.servlet.ModelAndView;

/\*\*

\*

\* @author ketanmalik

\*/

@Controller

public class AdminShowController {

@Autowired

ShowValidator showValidator;

@InitBinder("show")

protected void initBinder(WebDataBinder binder) {

binder.setValidator(showValidator);

}

@PostMapping("/add-show.htm")

public ModelAndView addShow(HttpServletRequest request, HttpSession session, Model model) {

if (invalidSessionObj(session, "user")) {

return sessionTimedOut(request);

}

Show show = new Show();

model.addAttribute("addShowForm", show);

session.setAttribute("venue-id", request.getParameter("add-show"));

return new ModelAndView("add-show-view");

}

@PostMapping("/show-added.htm")

public ModelAndView saveShow(@ModelAttribute("addShowForm") Show show, BindingResult result, HttpServletRequest request, HttpSession session, ShowDao showDao, EventDao eventDao, VenueDao venueDao) {

showValidator.validate(show, result);

if (result.hasErrors()) {

return new ModelAndView("add-show-view");

} else {

int event\_id = Integer.parseInt(request.getParameter("event-name"));

int venue\_id = Integer.parseInt(session.getAttribute("venue-id") + "");

Event event = eventDao.getEvent(event\_id);

Venue venue = venueDao.getVenue(venue\_id);

showValidator.validateCustom(result, event, show);

if (result.hasErrors()) {

return new ModelAndView("add-show-view");

}

show.setEvent(event);

show.setVenue(venue);

int total\_rows = show.getTotal\_rows();

int seats\_per\_row = show.getSeats\_per\_row();

int seats\_left = total\_rows \* seats\_per\_row;

show.setSeats\_left(seats\_left);

session.removeAttribute("venue-id");

int res = showDao.addShow(show);

if (res == 1) {

request.setAttribute("successMsg1", "Show added successfully.");

request.setAttribute("successMsg2", "Please go back to dashboard to manage events, users, and venues.");

return new ModelAndView("success-view");

} else {

return serverError(request, "The show could not be added.");

}

}

}

public boolean invalidSessionObj(HttpSession session, String obj) {

return session.getAttribute(obj) == null;

}

public ModelAndView sessionTimedOut(HttpServletRequest request) {

request.setAttribute("errorMsg1", "Your session has been timed out.");

request.setAttribute("errorMsg2", "Please log in again to start a new session.");

return new ModelAndView("error-view");

}

public ModelAndView serverError(HttpServletRequest request, String msg1) {

request.setAttribute("errorMsg1", msg1);

request.setAttribute("errorMsg2", "There was a problem in reaching out to our servers. Please try again later.");

return new ModelAndView("error-view");

}

}

**AdminVenueController**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package com.mycompany.controller;

import com.mycompany.dao.VenueDao;

import com.mycompany.pojo.Venue;

import com.mycompany.validator.VenueValidator;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.WebDataBinder;

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

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

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

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

import org.springframework.web.servlet.ModelAndView;

/\*\*

\*

\* @author ketanmalik

\*/

@Controller

public class AdminVenueController {

@Autowired

VenueValidator venueValidator;

@InitBinder("venue")

protected void initBinder(WebDataBinder binder) {

binder.setValidator(venueValidator);

}

@PostMapping("/add-venue.htm")

public ModelAndView addVenue(HttpServletRequest request, HttpServletResponse response, HttpSession session, Model model) {

if (invalidSessionObj(session, "user")) {

return sessionTimedOut(request);

}

Venue venue = new Venue();

model.addAttribute("addVenueForm", venue);

return new ModelAndView("add-venue-view");

}

@PostMapping("/delete-venue.htm")

public ModelAndView deleteVenue(HttpServletRequest request, HttpServletResponse response, HttpSession session, VenueDao venueDao) {

if (invalidSessionObj(session, "user")) {

return sessionTimedOut(request);

}

int res = venueDao.deleteVenue(Integer.parseInt(request.getParameter("delete-venue")));

if (res > 0) {

List<Venue> venueList = venueDao.getVenueList();

session.setAttribute("venueList", venueList);

return new ModelAndView("manage-venues-view");

} else {

return serverError(request, "The venue could not be deleted.");

}

}

@GetMapping("/manage-venues.htm")

public ModelAndView updateViewsInSession(HttpServletRequest request, HttpSession session, VenueDao venueDao) {

if (invalidSessionObj(session, "user")) {

return sessionTimedOut(request);

}

List<Venue> venueList = venueDao.getVenueList();

session.setAttribute("venueList", venueList);

return new ModelAndView("manage-venues-view");

}

@PostMapping("/manage-venues.htm")

public ModelAndView manageVenues(HttpServletRequest request, HttpServletResponse response, HttpSession session, VenueDao venueDao) {

if (invalidSessionObj(session, "user")) {

return sessionTimedOut(request);

}

List<Venue> venueList = venueDao.getVenueList();

session.setAttribute("venueList", venueList);

return new ModelAndView("manage-venues-view");

}

@PostMapping("/updated-venue.htm")

public ModelAndView saveUpdatedVenue(@ModelAttribute("updateVenueForm") Venue venue, BindingResult result, Model model, HttpServletRequest request, HttpSession session, VenueDao venueDao) {

venueValidator.validate(venue, result);

if (result.hasErrors()) {

return new ModelAndView("update-venue-view");

} else {

Venue sessionVenue = (Venue) session.getAttribute("venue");

int res = venueDao.updateVenue(sessionVenue.getVenue\_id(), venue);

if (res == 1) {

venue.setVenue\_id(sessionVenue.getVenue\_id());

session.removeAttribute("venue");

session.setAttribute("venue", venue);

Venue obj = (Venue) session.getAttribute("venue");

request.setAttribute("successMsg1", "Venue updated successfully.");

request.setAttribute("successMsg2", "Please go back to dashboard to manage events, users, and venues.");

return new ModelAndView("success-view");

} else {

return serverError(request, "The venue could not be updated.");

}

}

}

@PostMapping("/venue-added.htm")

public ModelAndView saveVenue(@ModelAttribute("addVenueForm") Venue venue, BindingResult result, HttpServletRequest request, HttpSession session, VenueDao venueDao) {

venueValidator.validate(venue, result);

if (result.hasErrors()) {

return new ModelAndView("add-venue-view");

} else {

int res = venueDao.saveVenue(venue);

if (res == 1) {

List<Venue> venueList = venueDao.getVenueList();

session.setAttribute("eventList", venueList);

request.setAttribute("successMsg1", "Venue added successfully.");

request.setAttribute("successMsg2", "Please go back to dashboard to manage events, users, and venues.");

return new ModelAndView("success-view");

} else {

return serverError(request, "The venue could not be added.");

}

}

}

@PostMapping("/update-venue.htm")

public ModelAndView updateVenue(HttpServletRequest request, HttpServletResponse response, HttpSession session, VenueDao venueDao, Model model) {

if (invalidSessionObj(session, "user")) {

return sessionTimedOut(request);

}

int venue\_id = Integer.parseInt(request.getParameter("update-venue"));

Venue venue = venueDao.getVenue(venue\_id);

// System.out.println("Venue id: " + venue.getVenue\_id());

// System.out.println("Venue name: " + venue.getVenue\_name());

// System.out.println("Venue city: " + venue.getVenue\_city());

// System.out.println("Venue state: " + venue.getVenue\_state());

// System.out.println("Venue country: " + venue.getVenue\_country());

// System.out.println("Venue rows: " + venue.getVenue\_rows());

// System.out.println("Venue seat: " + venue.getSeat\_per\_row());

// System.out.println("Venue seat price: " + venue.getSeat\_price());

if (venue == null) {

if (session.getAttribute("venue") != null) {

session.removeAttribute("venue");

}

return serverError(request, "The venue could not be updated.");

}

session.setAttribute("venue", venue);

model.addAttribute("updateVenueForm", venue);

return new ModelAndView("update-venue-view");

}

public boolean invalidSessionObj(HttpSession session, String obj) {

return session.getAttribute(obj) == null;

}

public ModelAndView sessionTimedOut(HttpServletRequest request) {

request.setAttribute("errorMsg1", "Your session has been timed out.");

request.setAttribute("errorMsg2", "Please log in again to start a new session.");

return new ModelAndView("error-view");

}

public ModelAndView serverError(HttpServletRequest request, String msg1) {

request.setAttribute("errorMsg1", msg1);

request.setAttribute("errorMsg2", "There was a problem in reaching out to our servers. Please try again later.");

return new ModelAndView("error-view");

}

}

**BookingController**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package com.mycompany.controller;

import com.mycompany.dao.BookingDao;

import com.mycompany.dao.ShowDao;

import com.mycompany.dao.VenueDao;

import com.mycompany.pojo.Booking;

import com.mycompany.pojo.Event;

import com.mycompany.pojo.Show;

import com.mycompany.pojo.User;

import com.mycompany.pojo.Venue;

import com.mycompany.utils.Util;

import com.mycompany.validator.BookingValidator;

import com.mycompany.view.BookingView;

import java.util.Date;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.WebDataBinder;

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

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

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

import org.springframework.web.servlet.ModelAndView;

/\*\*

\*

\* @author ketanmalik

\*/

@Controller

public class BookingController {

@Autowired

BookingValidator bookingValidator;

@InitBinder("show")

protected void initBinder(WebDataBinder binder) {

binder.setValidator(bookingValidator);

}

@PostMapping("/step-3.htm")

public ModelAndView verifyBookingDetails(HttpServletRequest request, HttpSession session, ShowDao showDao, Model model, VenueDao venueDao) {

if (invalidSessionObj(session, "user")) {

return sessionTimedOut(request);

}

int show\_id = Integer.parseInt(request.getParameter("show-id"));

try {

int venue\_id = Integer.parseInt(session.getAttribute("venue-id") + "");

Venue requestedVenue = venueDao.getVenue(venue\_id);

if (requestedVenue == null) {

System.out.println("Venue could not be found in verifyBookingDetails (Booking Controller)");

return serverError(request, "We could not proceed with your booking.");

}

session.setAttribute("requestedVenue", requestedVenue);

session.removeAttribute("venue-id");

} catch (Exception e) {

System.out.println("Exception in verifyBookingDetails (Booking Controller: )");

e.printStackTrace();

return serverError(request, "We could not proceed with your booking.");

}

Show selectedShow = showDao.getShowObj(show\_id);

session.setAttribute("selectedShow", selectedShow);

Booking booking = new Booking();

model.addAttribute("bookingForm", booking);

return new ModelAndView("customer-details-view");

}

@PostMapping("/step-4.htm")

public ModelAndView confirmBooking(@ModelAttribute("bookingForm") Booking booking, BindingResult result, HttpServletRequest request, HttpSession session, BookingDao bookingDao, ShowDao showDao) {

if (invalidSessionObj(session, "user")) {

return sessionTimedOut(request);

}

bookingValidator.validate(booking, result);

if (result.hasErrors()) {

return new ModelAndView("customer-details-view");

}

Show selectedShow = (Show) session.getAttribute("selectedShow");

Show mostRecentShow = showDao.getShowObj(selectedShow.getShow\_id());

if (mostRecentShow.getSeats\_left() == 0) {

return seatsExhausted(request, bookingDao, booking);

}

bookingValidator.validateCustom(result, booking, mostRecentShow);

if (result.hasErrors()) {

return new ModelAndView("customer-details-view");

}

try {

Date booking\_date = new Date();

String phone = booking.getPhone();

int seats = booking.getSeats();

int price = seats \* selectedShow.getSeat\_price();

User user = (User) session.getAttribute("user");

Event event = (Event) session.getAttribute("requestedEvent");

Venue venue = (Venue) session.getAttribute("requestedVenue");

Booking savedBooking = bookingDao.saveBooking(booking\_date, phone, seats, price, user, event, venue, selectedShow);

if (savedBooking != null) {

Show updatedShow = showDao.updateSeatsLeft(selectedShow.getShow\_id(), seats);

if (updatedShow == null) {

bookingDao.deleteBooking(savedBooking.getBooking\_id());

return seatsExhausted(request, bookingDao, savedBooking);

}

saveBookingPdf(session, savedBooking, venue, selectedShow);

return new ModelAndView("customer-confirm-view");

} else {

return serverError(request, "Your booking could not be confirmed");

}

} catch (Exception e) {

System.out.println("Exception in confirmBooking (BookingController): ");

return serverError(request, "Your booking could not be confirmed");

}

}

public boolean invalidSessionObj(HttpSession session, String obj) {

return session.getAttribute(obj) == null;

}

public ModelAndView sessionTimedOut(HttpServletRequest request) {

request.setAttribute("errorMsg1", "Your session has been timed out.");

request.setAttribute("errorMsg2", "Please log in again to start a new session.");

return new ModelAndView("error-view");

}

public ModelAndView serverError(HttpServletRequest request, String msg1) {

request.setAttribute("errorMsg1", msg1);

request.setAttribute("errorMsg2", "There was a problem in reaching out to our servers. Please try again later.");

return new ModelAndView("error-view");

}

public void saveBookingPdf(HttpSession session, Booking booking, Venue venue, Show show) {

User user = booking.getUser();

String booking\_id = booking.getBooking\_id() + "";

String event\_name = (String) session.getAttribute("selectedEvent");

String pdf\_venue = venue.getVenue\_name() + ", " + venue.getVenue\_city();

String pdf\_event\_date\_time = show.getShow\_time() + ", " + Util.dateToString(show.getShow\_date(), "date");

String screen = show.getScreen();

String seats = booking.getSeats() + "";

String price = "$" + booking.getPrice();

String booking\_date = Util.dateToString(booking.getBooking\_date(), "dateTime");

String name = user.getfName() + " " + user.getlName();

String contact = user.getEmail() + " " + booking.getPhone();

BookingView bookingPdf = new BookingView(booking\_id, event\_name, pdf\_venue, pdf\_event\_date\_time, screen, seats, price, booking\_date, name, contact);

removeTempSessionAttributes(session);

session.setAttribute("bookingPdf", bookingPdf);

}

private void removeTempSessionAttributes(HttpSession session) {

session.removeAttribute("selectedCity");

session.removeAttribute("requestedShows");

session.removeAttribute("requestedVenue");

session.removeAttribute("requestedEvent");

session.removeAttribute("selectedShow");

session.removeAttribute("selectedEvent");

session.removeAttribute("eventsInCity");

}

private ModelAndView seatsExhausted(HttpServletRequest request, BookingDao bookingDao, Booking savedBooking) {

request.setAttribute("errorMsg1", "Your booking could not be confirmed");

request.setAttribute("errorMsg2", "The show you selected ran out of seats. Please select a different show to book.");

return new ModelAndView("error-view");

}

}

**CustomerViewController**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package com.mycompany.controller;

import com.mycompany.dao.EventDao;

import com.mycompany.dao.ShowDao;

import com.mycompany.pojo.Event;

import com.mycompany.pojo.Show;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import org.springframework.stereotype.Controller;

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

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

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

import org.springframework.web.servlet.ModelAndView;

import org.springframework.web.servlet.view.RedirectView;

/\*\*

\*

\* @author ketanmalik

\*/

@Controller

public class CustomerViewController {

@GetMapping("/get-events.htm")

@ResponseBody

public String getEventsInCity(HttpServletRequest request, HttpServletResponse response, HttpSession session, ShowDao showDao) {

List<Event> eventsList = showDao.getEventsInCity(request.getParameter("city"));

String eventsInCity = "";

for (int i = 0; i < eventsList.size(); i++) {

eventsInCity += eventsList.get(i).getEvent\_name() + ",";

}

if (eventsInCity.length() > 0) {

eventsInCity = eventsInCity.substring(0, eventsInCity.length() - 1);

}

session.setAttribute("eventsInCity", eventsList);

return eventsInCity;

}

@PostMapping("/step-2.htm")

public ModelAndView selectShow(HttpServletRequest request, HttpServletResponse response, HttpSession session, EventDao eventDao, ShowDao showDao) {

session.removeAttribute("cityError");

session.removeAttribute("eventError");

if (invalidSessionObj(session, "user")) {

return sessionTimedOut(request);

}

String city = request.getParameter("city");

String event = request.getParameter("event");

boolean errors = false;

if (city == null || (city != null && city.equals(""))) {

errors = true;

session.setAttribute("cityError", "Please select a city");

}

if (event == null || (event != null && event.equals(""))) {

errors = true;

session.setAttribute("eventError", "Please select an event");

}

if (errors) {

return new ModelAndView(new RedirectView("/log-in-user.htm", true));

}

session.setAttribute("selectedCity", city);

session.setAttribute("selectedEvent", event);

Event requestedEvent = eventDao.getEventFromName(event);

List<Show> shows = showDao.getShow(city, event);

session.setAttribute("requestedEvent", requestedEvent);

session.setAttribute("requestedShows", shows);

if (shows == null) {

return serverError(request, "We could not retrieve any shows at the moment.");

}

return new ModelAndView("customer-select-view");

}

public boolean invalidSessionObj(HttpSession session, String obj) {

return session.getAttribute(obj) == null;

}

public ModelAndView sessionTimedOut(HttpServletRequest request) {

request.setAttribute("errorMsg1", "Your session has been timed out.");

request.setAttribute("errorMsg2", "Please log in again to start a new session.");

return new ModelAndView("error-view");

}

public ModelAndView serverError(HttpServletRequest request, String msg1) {

request.setAttribute("errorMsg1", msg1);

request.setAttribute("errorMsg2", "There was a problem in reaching out to our servers. Please try again later.");

return new ModelAndView("error-view");

}

}

**PdfController**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package com.mycompany.controller;

import com.mycompany.pojo.Booking;

import com.mycompany.pojo.Show;

import com.mycompany.pojo.Venue;

import com.mycompany.view.BookingView;

import java.util.HashMap;

import java.util.Map;

import javax.servlet.http.HttpSession;

import org.springframework.stereotype.Controller;

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

import org.springframework.web.servlet.ModelAndView;

/\*\*

\*

\* @author ketanmalik

\*/

@Controller

public class PdfController {

@GetMapping("/booking.pdf")

public ModelAndView generateBookingPdf(HttpSession session, Booking booking, Venue venue, Show show) {

BookingView bookingPdf = (BookingView) session.getAttribute("bookingPdf");

return new ModelAndView("booking-pdf", "bookingPdf", bookingPdf);

}

private void removeTempSessionAttributes(HttpSession session) {

session.removeAttribute("selectedCity");

session.removeAttribute("requestedShows");

session.removeAttribute("requestedVenue");

session.removeAttribute("requestedEvent");

session.removeAttribute("selectedShow");

session.removeAttribute("selectedEvent");

session.removeAttribute("eventsInCity");

}

}

**UserBookingsController**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package com.mycompany.controller;

import com.mycompany.dao.BookingDao;

import com.mycompany.dao.ShowDao;

import com.mycompany.pojo.Booking;

import com.mycompany.pojo.User;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.List;

import java.util.concurrent.TimeUnit;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;

import org.springframework.stereotype.Controller;

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

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

import org.springframework.web.servlet.ModelAndView;

/\*\*

\*

\* @author ketanmalik

\*/

@Controller

public class UserBookingsController {

@GetMapping("/user-bookings.htm")

public ModelAndView userBookings(HttpServletRequest request, HttpSession session, BookingDao bookingDao) {

if (invalidSessionObj(session, "user")) {

return sessionTimedOut(request);

}

User user = (User) session.getAttribute("user");

List<Booking> bookings = bookingDao.getBookings(user.getUser\_id());

session.setAttribute("bookings", bookings);

return new ModelAndView("user-bookings-view");

}

@PostMapping("/cancel-booking.htm")

public ModelAndView cancelBooking(HttpServletRequest request, HttpSession session, BookingDao bookingDao, ShowDao showDao) throws ParseException {

if (invalidSessionObj(session, "user")) {

return sessionTimedOut(request);

}

int booking\_id = 0;

try {

booking\_id = Integer.parseInt(request.getParameter("cancel-booking") + "");

} catch (Exception e) {

System.out.println("Exception in cancelBooking userbookingcontroller");

e.printStackTrace();

return serverError(request, "We could not retrieve your booking at the moment");

}

Booking booking = bookingDao.getBooking(booking\_id);

if (booking == null) {

return serverError(request, "We could not retrieve your booking at the moment");

}

Date bookingDate = booking.getShow().getShow\_date();

Date todaysDate = new Date();

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

todaysDate = sdformat.parse(sdformat.format(todaysDate));

bookingDate = sdformat.parse(sdformat.format(bookingDate));

long diffInMillies = Math.abs(bookingDate.getTime() - todaysDate.getTime());

long diff = TimeUnit.DAYS.convert(diffInMillies, TimeUnit.MILLISECONDS);

if (diff < 3) {

return new ModelAndView("user-bookings-view", "errorMsg", "Your booking cannot be cancelled because it is past the cancellation period.");

}

showDao.addSeats(booking.getShow().getShow\_id(), booking.getSeats());

bookingDao.deleteBooking(booking\_id);

User user = (User) session.getAttribute("user");

List<Booking> bookings = bookingDao.getBookings(user.getUser\_id());

session.setAttribute("bookings", bookings);

return new ModelAndView("user-bookings-view", "successMsg", "Your booking has been cancelled successfully!");

}

public boolean invalidSessionObj(HttpSession session, String obj) {

return session.getAttribute(obj) == null;

}

public ModelAndView sessionTimedOut(HttpServletRequest request) {

request.setAttribute("errorMsg1", "Your session has been timed out.");

request.setAttribute("errorMsg2", "Please log in again to start a new session.");

return new ModelAndView("error-view");

}

public ModelAndView serverError(HttpServletRequest request, String msg1) {

request.setAttribute("errorMsg1", msg1);

request.setAttribute("errorMsg2", "There was a problem in reaching out to our servers. Please try again later.");

return new ModelAndView("error-view");

}

}

**UserController**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package com.mycompany.controller;

import com.mycompany.dao.BookingDao;

import com.mycompany.dao.EventDao;

import com.mycompany.dao.ShowDao;

import com.mycompany.dao.UserDao;

import com.mycompany.dao.VenueDao;

import com.mycompany.pojo.Event;

import com.mycompany.pojo.Show;

import com.mycompany.pojo.User;

import com.mycompany.pojo.Venue;

import com.mycompany.validator.UserValidator;

import java.io.Serializable;

import java.util.Enumeration;

import java.util.List;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.WebDataBinder;

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

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

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

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

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

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

import org.springframework.web.servlet.ModelAndView;

/\*\*

\*

\* @author ketanmalik

\*/

@Controller

public class UserController implements Serializable {

@Autowired

UserValidator userValidator;

@InitBinder("show")

protected void initBinder(WebDataBinder binder) {

binder.setValidator(userValidator);

}

@PostMapping("/manage-users.htm")

public ModelAndView manageEvents(HttpServletRequest request, HttpServletResponse response, HttpSession session) {

if (invalidSessionObj(session, "user")) {

return sessionTimedOut(request);

}

User user = (User) session.getAttribute("user");

System.out.println("usr: " + user.getEmail());

return new ModelAndView("manage-users-view");

}

@PostMapping("/sign-up-success.htm")

public ModelAndView signUpUser(@ModelAttribute("signupForm") User user, BindingResult result, HttpServletRequest request, HttpServletResponse response, UserDao userDao) {

userValidator.validate(user, result);

if (result.hasErrors()) {

return new ModelAndView("sign-up-view");

}

String fName = request.getParameter("fName");

String lName = request.getParameter("lName");

String email = request.getParameter("email");

String password = request.getParameter("password");

String city = request.getParameter("city");

String state = request.getParameter("state");

String country = request.getParameter("country");

int res = userDao.addUser(fName, lName, email, password, city, state, country, "customer");

if (res == -1) {

request.setAttribute("errorMsg1", "You could not be signed up.");

request.setAttribute("errorMsg2", "A user already exists with this email. Please try to log in instead.");

return new ModelAndView("error-view");

}

if (res == 1) {

request.setAttribute("successMsg1", "You have signed up successfully.");

request.setAttribute("successMsg2", "Please log in from home page to enjoy our services.");

return new ModelAndView("success-view");

}

request.setAttribute("errorMsg1", "You could not be signed up.");

request.setAttribute("errorMsg2", "There was a problem in reaching out to our servers. Please try again later.");

return new ModelAndView("error-view");

}

@PostMapping("/log-in-user.htm")

public ModelAndView logInUser(HttpServletRequest request, HttpServletResponse response, UserDao userDao, HttpSession session, EventDao eventDao, VenueDao venueDao, ShowDao showDao, BookingDao bookingDao) {

String email = request.getParameter("email");

User user = userDao.getUser(email);

if (user == null) {

request.setAttribute("errorMsg1", "You could not be logged in.");

request.setAttribute("errorMsg2", "Either your browser doesn't allow Javascript or our network is not responding.");

if (session.getAttribute("user") != null) {

session.removeAttribute("user");

};

return new ModelAndView("error-view");

}

List<Event> eventList = eventDao.getEventList();

List<Venue> venueList = venueDao.getVenueList();

List<Show> showList = showDao.getShowList();

session.setAttribute("user", user);

session.setAttribute("eventList", eventList);

session.setAttribute("venueList", venueList);

session.setAttribute("showList", showList);

if (user.getUser\_type().equals("admin")) {

return new ModelAndView("manage-events-view");

} else {

List<String> venueCityList = venueDao.getUniqueVenueCityList();

session.setAttribute("venueCityList", venueCityList);

return new ModelAndView("customer-view");

}

}

@GetMapping("/log-in-user.htm")

public ModelAndView redirectUser(HttpServletRequest request, HttpSession session, EventDao eventDao) {

User loggedInUser = (User) session.getAttribute("user");

if (loggedInUser != null) {

if (loggedInUser.getUser\_type().equals("admin")) {

session.removeAttribute("eventList");

List<Event> eventList = eventDao.getEventList();

session.setAttribute("eventList", eventList);

return new ModelAndView("manage-events-view");

} else {

return new ModelAndView("customer-view");

}

} else {

return new ModelAndView("index");

}

}

@PostMapping("/sign-out.htm")

public ModelAndView signOutUser(HttpSession session) {

Enumeration<String> attributes = session.getAttributeNames();

while (attributes.hasMoreElements()) {

String attribute = (String) attributes.nextElement();

session.removeAttribute(attribute);

}

return new ModelAndView("index");

}

@PostMapping("/validate-user.htm")

@ResponseBody

public String validateUser(HttpServletRequest request, HttpServletResponse response, HttpSession session, UserDao userDao) {

String email = request.getParameter("email");

String password = request.getParameter("password");

if (email.trim().equals("") || password.trim().equals("")) {

return "invalidUser";

}

boolean validUser = userDao.checkUser(email, password, "log-in");

if (validUser) {

return "validUser";

}

return "invalidUser";

}

@GetMapping("/log-in.htm")

public ModelAndView logInPage(HttpServletRequest request, HttpServletResponse response) {

return new ModelAndView("log-in-view");

}

@GetMapping("/sign-up.htm")

public ModelAndView signUpPage(HttpServletRequest request, HttpServletResponse response, Model model) {

User user = new User();

model.addAttribute("signupForm", user);

return new ModelAndView("sign-up-view");

}

public boolean invalidSessionObj(HttpSession session, String obj) {

return session.getAttribute(obj) == null;

}

public ModelAndView sessionTimedOut(HttpServletRequest request) {

request.setAttribute("errorMsg1", "Your session has been timed out.");

request.setAttribute("errorMsg2", "Please log in again to start a new session.");

return new ModelAndView("error-view");

}

public ModelAndView serverError(HttpServletRequest request, String msg1) {

request.setAttribute("errorMsg1", msg1);

request.setAttribute("errorMsg2", "There was a problem in reaching out to our servers. Please try again later.");

return new ModelAndView("error-view");

}

}

**BookingDao**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package com.mycompany.dao;

import com.mycompany.pojo.Booking;

import com.mycompany.pojo.Event;

import com.mycompany.pojo.Show;

import com.mycompany.pojo.User;

import com.mycompany.pojo.Venue;

import java.util.Date;

import java.util.List;

import org.hibernate.query.Query;

/\*\*

\*

\* @author ketanmalik

\*/

public class BookingDao extends DAO {

public Booking saveBooking(Date date, String phone, int seats, int price, User user, Event event, Venue venue, Show show) {

try {

Show selectedShow = new Show(show.getShow\_id(), show.getSeat\_price(), show.getTotal\_rows(), show.getSeats\_per\_row(), show.getShow\_time(), show.getSeats\_left(), show.getShow\_date(), show.getScreen(), event, venue);

Booking booking = new Booking(date, phone, seats, price, user, selectedShow);

beginTransaction();

getSession().save(booking);

commit();

return booking;

} catch (Exception e) {

rollback();

System.out.println("Exception in saveBooking BookingDAO: ");

e.printStackTrace();

return null;

}

}

public void deleteBooking(int booking\_id) {

try {

String hql = "DELETE FROM Booking WHERE booking\_id = :booking\_id";

Query query = getSession().createQuery(hql);

query.setParameter("booking\_id", booking\_id);

beginTransaction();

query.executeUpdate();

commit();

} catch (Exception e) {

rollback();

System.out.println("Exception in deleteBooking BookingDAO: ");

e.printStackTrace();

}

}

public List<Booking> getBookings(int user\_id) {

try {

String hql = "FROM Booking WHERE user.user\_id = :user\_id";

Query query = getSession().createQuery(hql);

query.setParameter("user\_id", user\_id);

beginTransaction();

List<Booking> bookings = query.list();

if (bookings.isEmpty()) {

rollback();

return null;

}

commit();

return bookings;

} catch (Exception e) {

rollback();

System.out.println("Exception in getBookings BookingDAO: ");

e.printStackTrace();

return null;

}

}

public Booking getBooking(int booking\_id) {

try {

String hql = "FROM Booking WHERE booking\_id = :booking\_id";

Query query = getSession().createQuery(hql);

query.setParameter("booking\_id", booking\_id);

beginTransaction();

List<Booking> bookings = query.list();

if (bookings.isEmpty()) {

rollback();

return null;

}

commit();

System.out.println("committed2");

return bookings.get(0);

} catch (Exception e) {

rollback();

System.out.println("Exception in getBooking BookingDAO: ");

e.printStackTrace();

return null;

}

}

}

**EventDao**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package com.mycompany.dao;

import com.mycompany.pojo.Event;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

import org.hibernate.query.Query;

/\*\*

\*

\* @author ketanmalik

\*/

public class EventDao extends DAO {

public int deleteEvent(int event\_id) {

String hql = "DELETE FROM Event WHERE event\_id=:event\_id";

Query query = getSession().createQuery(hql);

query.setParameter("event\_id", event\_id);

beginTransaction();

int res = query.executeUpdate();

commit();

return res;

}

public Event getEvent(int event\_id) {

try {

String hql = "FROM Event WHERE event\_id=:event\_id";

Query query = getSession().createQuery(hql);

query.setParameter("event\_id", event\_id);

List<Event> eventList = new ArrayList<>();

beginTransaction();

eventList = query.list();

if (eventList.isEmpty()) {

rollback();

return null;

}

commit();

return eventList.get(0);

} catch (Exception e) {

rollback();

System.out.println("Exception in getEvent: " + e);

return null;

}

}

public Event getEventFromName(String event\_name) {

try {

String hql = "FROM Event WHERE event\_name = :event\_name";

Query query = getSession().createQuery(hql);

query.setParameter("event\_name", event\_name);

beginTransaction();

Event event = (Event) query.list().get(0);

commit();

return event;

} catch (Exception e) {

rollback();

System.out.println("Exception in getEventFromName Dao: " + e);

return null;

}

}

public List<Event> getEventList() {

try {

String hql = "FROM Event";

Query query = getSession().createQuery(hql);

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

beginTransaction();

result = query.list();

commit();

return result;

} catch (Exception e) {

rollback();

System.out.println("Exception in getEventList: " + e);

return null;

}

}

public int saveEvent(String event\_name, String event\_type, String event\_cast, String event\_rating, String event\_genre,

String event\_language, String event\_summary, String event\_duration, Date event\_date) {

Event event = new Event();

event.setEvent\_name(event\_name);

event.setEvent\_type(event\_type);

event.setEvent\_cast(event\_cast);

event.setEvent\_rating(event\_rating);

event.setEvent\_genre(event\_genre);

event.setEvent\_language(event\_language);

event.setEvent\_summary(event\_summary);

event.setEvent\_duration(event\_duration);

event.setEvent\_date(event\_date);

beginTransaction();

getSession().save(event);

commit();

return 1;

}

public int updateEvent(int event\_id, Event event) {

String event\_name = event.getEvent\_name();

String event\_cast = event.getEvent\_cast();

String event\_type = event.getEvent\_type();

String event\_rating = event.getEvent\_rating();

String event\_genre = event.getEvent\_genre();

String event\_language = event.getEvent\_language();

String event\_summary = event.getEvent\_summary();

Date event\_date = event.getEvent\_date();

String event\_duration = event.getEvent\_duration();

String hql = "UPDATE Event SET event\_name=:event\_name, event\_cast=:event\_cast, event\_type=:event\_type, "

+ "event\_rating=:event\_rating, event\_genre=:event\_genre, event\_language=:event\_language, "

+ "event\_summary=:event\_summary, event\_date=:event\_date, event\_duration=:event\_duration WHERE event\_id=:event\_id";

Query query = getSession().createQuery(hql);

query.setParameter("event\_id", event\_id);

query.setParameter("event\_name", event\_name);

query.setParameter("event\_cast", event\_cast);

query.setParameter("event\_type", event\_type);

query.setParameter("event\_rating", event\_rating);

query.setParameter("event\_genre", event\_genre);

query.setParameter("event\_language", event\_language);

query.setParameter("event\_summary", event\_summary);

query.setParameter("event\_date", event\_date);

query.setParameter("event\_duration", event\_duration);

beginTransaction();

int res = query.executeUpdate();

commit();

return res;

}

}

**ShowDao**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package com.mycompany.dao;

import com.mycompany.pojo.Event;

import com.mycompany.pojo.Show;

import java.util.ArrayList;

import java.util.List;

import org.hibernate.query.Query;

/\*\*

\*

\* @author ketanmalik

\*/

public class ShowDao extends DAO {

public int addShow(Show show) {

try {

beginTransaction();

getSession().save(show);

commit();

return 1;

} catch (Exception e) {

rollback();

System.out.println("Exception in addShow DAO: ");

e.printStackTrace();

return 0;

}

}

public List<Show> getShowList() {

try {

String hql = "FROM Show";

Query query = getSession().createQuery(hql);

List result = new ArrayList<>();

beginTransaction();

result = query.list();

commit();

return result;

} catch (Exception e) {

rollback();

System.out.println("Exception in getShowList Dao: ");

e.printStackTrace();

return null;

}

}

public List<Event> getEventsInCity(String city) {

try {

String hql = "SELECT DISTINCT event FROM Show WHERE venue\_id in (SELECT venue\_id FROM Venue WHERE venue\_city = :city)";

Query query = getSession().createQuery(hql);

query.setParameter("city", city);

beginTransaction();

List<Event> eventsList = query.list();

commit();

return eventsList;

} catch (Exception e) {

rollback();

System.out.println("Exception in getEventsFromCity Dao: ");

e.printStackTrace();

return null;

}

}

public List<Show> getShow(String city, String event) {

try {

beginTransaction();

String event\_hql = "SELECT event\_id FROM Event WHERE event\_name = :event\_name";

Query event\_query = getSession().createQuery(event\_hql);

event\_query.setParameter("event\_name", event);

List<Integer> event\_id\_list = event\_query.list();

if (event\_id\_list.isEmpty()) {

rollback();

return null;

}

int event\_id = (Integer) event\_query.list().get(0);

String venue\_hql = "SELECT venue\_id FROM Venue WHERE venue\_city = :venue\_city";

Query venue\_query = getSession().createQuery(venue\_hql);

venue\_query.setParameter("venue\_city", city);

List<Integer> venue\_id\_list = venue\_query.list();

if (venue\_id\_list.isEmpty()) {

rollback();

return null;

}

String shows\_hql = "FROM Show WHERE event.event\_id = :event\_id AND venue.venue\_id IN (:venue\_ids)";

Query shows\_query = getSession().createQuery(shows\_hql);

shows\_query.setParameter("event\_id", event\_id);

shows\_query.setParameterList("venue\_ids", venue\_id\_list);

List<Show> shows = shows\_query.list();

if (shows.isEmpty()) {

rollback();

return null;

}

commit();

return shows;

} catch (Exception e) {

rollback();

System.out.println("Exception in getShowList Dao: ");

e.printStackTrace();

return null;

}

}

public Show getShowObj(int show\_id) {

try {

String hql = "FROM Show WHERE show\_id = :show\_id";

Query query = getSession().createQuery(hql);

query.setParameter("show\_id", show\_id);

beginTransaction();

List<Show> shows = query.list();

if (shows.isEmpty()) {

rollback();

return null;

}

commit();

return shows.get(0);

} catch (Exception e) {

rollback();

System.out.println("Exception in getShowList Dao: ");

e.printStackTrace();

return null;

}

}

public Show updateSeatsLeft(int show\_id, int seats\_booked) {

try {

String hql = "FROM Show WHERE show\_id = :show\_id";

Query query = getSession().createQuery(hql);

query.setParameter("show\_id", show\_id);

beginTransaction();

List<Show> shows = query.list();

if (shows.isEmpty()) {

rollback();

return null;

}

Show show\_to\_update = shows.get(0);

int seats\_left = show\_to\_update.getSeats\_left() - seats\_booked;

if (seats\_left < 0) {

rollback();

return null;

}

hql = "UPDATE Show SET seats\_left = :seats\_left where show\_id = :show\_id";

query = getSession().createQuery(hql);

query.setParameter("seats\_left", seats\_left);

query.setParameter("show\_id", show\_id);

query.executeUpdate();

commit();

show\_to\_update.setSeats\_left(seats\_left);

return show\_to\_update;

} catch (Exception e) {

rollback();

System.out.println("Exception in updateSeatsLeft ShowDao: ");

e.printStackTrace();

return null;

}

}

public void addSeats(int show\_id, int seatsToAdd) {

try {

String hql = "FROM Show WHERE show\_id = :show\_id";

Query query = getSession().createQuery(hql);

query.setParameter("show\_id", show\_id);

beginTransaction();

List<Show> shows = query.list();

if (shows.isEmpty()) {

rollback();

}

Show show = shows.get(0);

show.setSeats\_left(show.getSeats\_left() + seatsToAdd);

getSession().update(show);

commit();

} catch (Exception e) {

rollback();

System.out.println("Exception in getShowList Dao: ");

e.printStackTrace();

}

}

}

**UserDao**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package com.mycompany.dao;

import com.mycompany.pojo.User;

import java.util.List;

import org.hibernate.query.Query;

/\*\*

\*

\* @author ketanmalik

\*/

public class UserDao extends DAO {

public int addUser(String fName, String lName, String email, String password, String city, String state, String country, String user\_type) {

boolean userAlreadyExists = checkUser(email, "", "sign-up");

if (userAlreadyExists) {

return -1;

}

try {

User user = new User(fName, lName, email, password, city, state, country, user\_type);

beginTransaction();

getSession().save(user);

commit();

return 1;

} catch (Exception e) {

rollback();

System.out.println("Exception in addUser UserDao: ");

e.printStackTrace();

return -1;

}

}

public boolean checkUser(String email, String password, String mode) {

String hql;

if (mode.equalsIgnoreCase("sign-up")) {

hql = "FROM User where email=:email";

} else {

hql = "FROM User where email=:email and password =:password";

}

try {

Query query = getSession().createQuery(hql);

query.setParameter("email", email);

if (mode.equalsIgnoreCase("log-in")) {

query.setParameter("password", password);

}

beginTransaction();

List users = query.list();

if (users.isEmpty()) {

rollback();

return false;

}

commit();

return true;

} catch (Exception e) {

rollback();

System.out.println("Exception in checkUser UserDao: ");

e.printStackTrace();

return false;

}

}

public User getUser(String email) {

try {

String hql = "From User Where email = :email";

Query query = getSession().createQuery(hql);

query.setParameter("email", email);

beginTransaction();

List<User> users = query.list();

if (users.isEmpty()) {

rollback();

return null;

}

commit();

return users.get(0);

} catch (Exception e) {

rollback();

System.out.println("Exception in getUser UserDao: ");

e.printStackTrace();

return null;

}

}

}

**VenueDao**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package com.mycompany.dao;

import java.util.List;

import org.hibernate.query.Query;

import com.mycompany.pojo.Venue;

import java.util.ArrayList;

/\*\*

\*

\* @author ketanmalik

\*/

public class VenueDao extends DAO {

public int deleteVenue(int venue\_id) {

String hql = "DELETE FROM Venue WHERE venue\_id=:venue\_id";

Query query = getSession().createQuery(hql);

query.setParameter("venue\_id", venue\_id);

beginTransaction();

int res = query.executeUpdate();

commit();

return res;

}

public List<String> getUniqueVenueCityList() {

try {

String hql = "SELECT DISTINCT venue\_city FROM Venue";

Query query = getSession().createQuery(hql);

List<String> venueList = new ArrayList<>();

beginTransaction();

venueList = query.list();

commit();

return venueList;

} catch (Exception e) {

rollback();

System.out.println("Exception in getUniqueVenueList Dao: ");

e.printStackTrace();

return null;

}

}

public Venue getVenue(int venue\_id) {

try {

String hql = "FROM Venue WHERE venue\_id=:venue\_id";

Query query = getSession().createQuery(hql);

query.setParameter("venue\_id", venue\_id);

List<Venue> venueList = new ArrayList<>();

beginTransaction();

venueList = query.list();

if (venueList.isEmpty()) {

rollback();

return null;

}

commit();

return venueList.get(0);

} catch (Exception e) {

rollback();

System.out.println("Exception in getVenue: " + e);

return null;

}

}

public List<Venue> getVenueList() {

try {

String hql = "FROM Venue";

Query query = getSession().createQuery(hql);

List result = new ArrayList<>();

beginTransaction();

result = query.list();

commit();

return result;

} catch (Exception e) {

rollback();

System.out.println("Exception in getVenueList: " + e);

return null;

}

}

public int saveVenue(Venue venue) {

try {

beginTransaction();

getSession().save(venue);

commit();

return 1;

} catch (Exception e) {

rollback();

System.out.println("Exception in saveVenue: ");

e.printStackTrace();

return 0;

}

}

public int updateVenue(int venue\_id, Venue venue) {

try {

String venue\_name = venue.getVenue\_name();

String venue\_city = venue.getVenue\_city();

String venue\_state = venue.getVenue\_state();

String venue\_country = venue.getVenue\_country();

String hql = "UPDATE Venue SET venue\_name=:venue\_name, venue\_city=:venue\_city, venue\_state=:venue\_state, "

+ "venue\_country=:venue\_country WHERE venue\_id=:venue\_id";

Query query = getSession().createQuery(hql);

query.setParameter("venue\_id", venue\_id);

query.setParameter("venue\_name", venue\_name);

query.setParameter("venue\_city", venue\_city);

query.setParameter("venue\_state", venue\_state);

query.setParameter("venue\_country", venue\_country);

beginTransaction();

int res = query.executeUpdate();

commit();

return res;

} catch (Exception e) {

rollback();

System.out.println("Exception in updateVenue: " + e);

return 0;

}

}

}