IrctcController.java

package com.cts.irctc.controller;

import java.time.LocalDate;

import java.util.ArrayList;

import java.util.List;

import javax.validation.Valid;

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

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.ui.ModelMap;

import org.springframework.validation.BindingResult;

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

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

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

import com.cts.irctc.exception.ApplicationException;

import com.cts.irctc.model.TicketBooking;

import com.cts.irctc.model.TrainInfo;

import com.cts.irctc.service.IrctcService;

@Controller

public class IrctcController {

@Autowired

private IrctcService service;

// @Autowired

// public IrctcController(IrctcService service) {

// super();

// this.service = service;

// }

@Autowired

TicketBooking ticketBooking;

@RequestMapping(value="/showTicketBookingForm",method=RequestMethod.GET)

public String showTicketBookingForm(Model m) {

//Add code here..

m.addAttribute("ticketBooking",ticketBooking);

return "ticketBooking"; // TODO, modify this value

}

@RequestMapping(value="getTicketBookingResultPage",method=RequestMethod.POST)

public String getCarSearchResultForm(@Valid @ModelAttribute("ticketBooking") TicketBooking ticketBooking,BindingResult result,ModelMap map) throws ApplicationException {

//Add code here..

if(result.hasErrors())

{

return "ticketBooking";

}

else

{

try

{

TrainInfo ti = service.getTicketBookingResult(ticketBooking);

System.out.println("controller");

String pnr = service.getPNRNumber(ticketBooking);

System.out.println("controller2");

System.out.println(ti.getTrainNumber());

String pnr\_number = ti.getTrainNumber().substring(0,2);

map.addAttribute("pnr",pnr\_number+pnr);

map.addAttribute("tno",ti.getTrainNumber());

map.addAttribute("tname",ti.getTrainName());

map.addAttribute("dot",ticketBooking.getDateOfTravel());

map.addAttribute("dept",ti.getDeparture());

map.addAttribute("duration",ti.getDuration());

map.addAttribute("a",ti.getArrival());

map.addAttribute("price",ti.getFarePerPassenger() \* ticketBooking.getNoOfTickets());

//return "ticketBookingResult";

}catch(ApplicationException e)

{

}

return "ticketBookingResult"; // TODO, modify this value

}

}

@ModelAttribute("fromCities")

public List<String> populateFromCities() {

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

fromCities.add("Chennai");

fromCities.add("Delhi");

fromCities.add("Bangalore");

fromCities.add("Pune");

return fromCities;

}

@ModelAttribute("toCities")

public List<String> populateToCities() {

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

toCities.add("Chennai");

toCities.add("Delhi");

toCities.add("Bangalore");

toCities.add("Pune");

return toCities;

}

@ModelAttribute("class")

public List<String> populateClassType() {

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

classTypes.add("AC First Class (1A)");

classTypes.add("AC 2 Tier (2A)");

classTypes.add("AC 3 Tier (3A)");

classTypes.add("Sleeper (SL)");

classTypes.add("Second Sitting (2S)");

return classTypes;

}

}

ApplicationException.java

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* This class ApplicationException is a user defined exception for the proposed system

\*

\* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS, EXCEPTION CLAUSES, RETURN TYPES

\* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE

\* DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS

\* CHANGE THE RETURN TYPE FROM NULL OF THE METHODS ONCE YOU BUILT THE LOGIC

\* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF NEED BE,

\* YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER EXCEPTION CLAUSE

\* ADD CONSTRUCTORS AS NEEDED

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

package com.cts.irctc.exception;

public class ApplicationException extends Exception {

private static final long serialVersionUID = -9079454849611061074L;

}

ExceptionHandlerControllerAdvice.java

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* This class ExceptionHandlerControllerAdvice is used to handle different exceptions raised by Controller

\*

\* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS, EXCEPTION CLAUSES, RETURN TYPES

\* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE

\* DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS

\* CHANGE THE RETURN TYPE FROM NULL OF THE METHODS ONCE YOU BUILT THE LOGIC

\* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF NEED BE,

\* YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER EXCEPTION CLAUSE

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

package com.cts.irctc.exception;

import org.springframework.ui.Model;

import org.springframework.web.servlet.ModelAndView;

public class ExceptionHandlerControllerAdvice {

public ModelAndView handleResourceNotFound(final ApplicationException exception, final Model model) {

//Add code here..

return null; //TODO, modify return value

}

}

TicketBooking.java

package com.cts.irctc.model;

import java.time.LocalDate;

import java.time.LocalDateTime;

import javax.validation.constraints.Max;

import javax.validation.constraints.Min;

import javax.validation.constraints.NotBlank;

import javax.validation.constraints.Pattern;

import org.springframework.stereotype.Component;

@Component

public class TicketBooking {

// Use validation annotations as per the requirement

@NotBlank(message="Customer name is required")

private String customerName;

@NotBlank(message="Mobile Number is required")

@Pattern(regexp = "^[789][0-9]{9}$")

private String mobileNumber;

private String fromCity;

private String toCity;

private String travelClass;

@Min(value = 1,message="No. of Tickets must be more than 0")

@Max(value= 4)

private int noOfTickets;

private LocalDate dateOfTravel;

private String pnrNumber;

private double totalfareAmount;

//public TicketBooking() {

// TODO Auto-generated constructor stub

//}

/\*

\* public TicketBooking(String customerName,String mobileNumber,String fromCity,

\* String toCity, String travelClass, int noOfTickets,LocalDate dateOfTravel,

\* String pnrNumber, double totalfareAmount) { super(); this.customerName =

\* customerName; this.mobileNumber = mobileNumber; this.fromCity = fromCity;

\* this.toCity = toCity; this.travelClass = travelClass; this.noOfTickets =

\* noOfTickets; this.dateOfTravel = dateOfTravel; this.pnrNumber = pnrNumber;

\* this.totalfareAmount = totalfareAmount; }

\*/

public String getCustomerName() {

return customerName;

}

public void setCustomerName(String customerName) {

this.customerName = customerName;

}

public String getMobileNumber() {

return mobileNumber;

}

public void setMobileNumber(String mobileNumber) {

this.mobileNumber = mobileNumber;

}

public String getFromCity() {

return fromCity;

}

public void setFromCity(String fromCity) {

this.fromCity = fromCity;

}

public String getToCity() {

return toCity;

}

public void setToCity(String toCity) {

this.toCity = toCity;

}

public String getTravelClass() {

return travelClass;

}

public void setTravelClass(String travelClass) {

this.travelClass = travelClass;

}

public int getNoOfTickets() {

return noOfTickets;

}

public void setNoOfTickets(int noOfTickets) {

this.noOfTickets = noOfTickets;

}

public LocalDate getDateOfTravel() {

return dateOfTravel;

}

public void setDateOfTravel(LocalDate dateOfTravel) {

this.dateOfTravel = dateOfTravel;

}

public String getPnrNumber() {

return pnrNumber;

}

public void setPnrNumber(String pnrNumber) {

this.pnrNumber = pnrNumber;

}

public double getTotalfareAmount() {

return totalfareAmount;

}

public void setTotalfareAmount(double totalfareAmount) {

this.totalfareAmount = totalfareAmount;

}

}

TrainInfo.java

package com.cts.irctc.model;

public class TrainInfo {

private String trainNumber;

private String trainName;

private String from;

private String to;

private String tavelClass;

private String departure;

private String duration;

private String arrival;

private double farePerPassenger;

/\*

\* public TrainInfo() { // TODO Auto-generated constructor stub }

\*/

public TrainInfo(String trainNumber, String trainName, String from, String to, String tavelClass, String departure,

String duration, String arrival, double farePerPassenger) {

super();

this.trainNumber = trainNumber;

this.trainName = trainName;

this.from = from;

this.to = to;

this.tavelClass = tavelClass;

this.departure = departure;

this.duration = duration;

this.arrival = arrival;

this.farePerPassenger = farePerPassenger;

}

public String getTrainNumber() {

return trainNumber;

}

public void setTrainNumber(String trainNumber) {

this.trainNumber = trainNumber;

}

public String getTrainName() {

return trainName;

}

public void setTrainName(String trainName) {

this.trainName = trainName;

}

public String getFrom() {

return from;

}

public void setFrom(String from) {

this.from = from;

}

public String getTo() {

return to;

}

public void setTo(String to) {

this.to = to;

}

public String getTavelClass() {

return tavelClass;

}

public void setTavelClass(String tavelClass) {

this.tavelClass = tavelClass;

}

public String getDeparture() {

return departure;

}

public void setDeparture(String departure) {

this.departure = departure;

}

public String getDuration() {

return duration;

}

public void setDuration(String duration) {

this.duration = duration;

}

public String getArrival() {

return arrival;

}

public void setArrival(String arrival) {

this.arrival = arrival;

}

public double getFarePerPassenger() {

return farePerPassenger;

}

public void setFarePerPassenger(double farePerPassenger) {

this.farePerPassenger = farePerPassenger;

}

}

IrctcService.java

package com.cts.irctc.service;

import java.util.ArrayList;

import java.util.List;

import org.springframework.stereotype.Service;

import com.cts.irctc.exception.ApplicationException;

import com.cts.irctc.model.TicketBooking;

import com.cts.irctc.model.TrainInfo;

@Service

public class IrctcService {

public TrainInfo getTicketBookingResult(TicketBooking ticketBooking) throws ApplicationException {

//Add code here..

//IrctcService is = new IrctcService ();

System.out.println("gettbr");

List<TrainInfo> list1 =buildTrains();

if(ticketBooking.getFromCity() == ticketBooking.getToCity())

{

throw new ApplicationException();

}

else

{

for(TrainInfo ti:list1)

{

System.out.println("class :"+ticketBooking.getTravelClass());

System.out.println("trainsclass :"+ti.getTavelClass());

if(ti.getTavelClass() == ticketBooking.getTravelClass() && ti.getFrom() == ticketBooking.getFromCity() && ti.getTo() == ticketBooking.getToCity())

{

return ti;

}

}

}//TODO, modify return value

return null;

}

public String getPNRNumber(TicketBooking ticketBooking) throws ApplicationException {

//Add code here..

if(ticketBooking.getFromCity() == ticketBooking.getToCity())

{

throw new ApplicationException();

}

else

{

return ticketBooking.getMobileNumber().substring(0,6);

}//TODO, modify return value

}

// DO NOT MODIFY THIS METHOD

//DO NOT MODIFY CODE WITHIN METHOD

private List<TrainInfo> buildTrains() {

System.out.println("trains");

List<TrainInfo> trains = new ArrayList<TrainInfo>();

//pune to bangalore

TrainInfo trainInfo1=new TrainInfo("01013", "LTT CBE SPL", "Pune", "Bangalore", "AC First Class (1A)", "1:55 AM", "20 hr 30 min", "10:25 PM", 3080);

TrainInfo trainInfo2=new TrainInfo("01013", "LTT CBE SPL", "Pune", "Bangalore", "2A", "1:55 AM", "20 hr 30 min", "10:25 PM", 1825);

TrainInfo trainInfo3=new TrainInfo("01013", "LTT CBE SPL", "Pune", "Bangalore", "3A", "1:55 AM", "20 hr 30 min", "10:25 PM", 1150);

TrainInfo trainInfo4=new TrainInfo("01013", "LTT CBE SPL", "Pune", "Bangalore", "SL", "1:55 AM", "20 hr 30 min", "10:25 PM", 475);

TrainInfo trainInfo5=new TrainInfo("01013", "LTT CBE SPL", "Pune", "Bangalore", "2S", "1:55 AM", "20 hr 30 min", "10:25 PM", 285);

//bagalore to pune

TrainInfo trainInfo6=new TrainInfo("01302", "UDYAN EXP", "Bangalore", "Pune","1A", "8:20 PM", "19 hr 40 min", "4:00 PM", 3080);

TrainInfo trainInfo7=new TrainInfo("01302", "UDYAN EXP", "Bangalore", "Pune", "2A", "8:20 PM", "19 hr 40 min", "4:00 PM", 1825);

TrainInfo trainInfo8=new TrainInfo("01302", "UDYAN EXP", "Bangalore","Pune", "3A", "8:20 PM", "19 hr 40 min", "4:00 PM", 1275);

TrainInfo trainInfo9=new TrainInfo("01302", "UDYAN EXP", "Bangalore","Pune", "SL", "8:20 PM", "19 hr 40 min", "4:00 PM", 475);

TrainInfo trainInfo10=new TrainInfo("01302", "UDYAN EXP", "Bangalore","Pune", "2S", "8:20 PM", "19 hr 40 min", "4:00 PM", 285);

//chennai to pune

TrainInfo trainInfo11=new TrainInfo("02164", "MAS LTT EXPRESS", "Chennai", "Pune", "2A", "6:25 PM", "16 hr 45 min", "11:10 AM", 2440);

TrainInfo trainInfo12=new TrainInfo("02164", "MAS LTT EXPRESS", "Chennai","Pune", "3A", "6:25 PM", "16 hr 45 min", "11:10 AM", 1785);

TrainInfo trainInfo13=new TrainInfo("02164", "MAS LTT EXPRESS", "Chennai","Pune", "SL", "6:25 PM", "16 hr 45 min", "11:10 AM", 685);

TrainInfo trainInfo14=new TrainInfo("02164", "MAS LTT EXPRESS", "Chennai","Pune", "2S", "6:25 PM", "16 hr 45 min", "11:10 AM", 335);

//bangalore to chennai

TrainInfo trainInfo15=new TrainInfo("02295", "SANGHAMITRA EXP", "Bangalore","Chennai", "2A", "9:00 AM", "6 hr 15 min", "3:15 PM", 945);

TrainInfo trainInfo16=new TrainInfo("02295", "SANGHAMITRA EXP", "Bangalore", "Chennai", "3A", "9:00 AM", "6 hr 15 min", "3:15 PM", 625);

TrainInfo trainInfo17=new TrainInfo("02295", "SANGHAMITRA EXP", "Bangalore","Chennai", "SL", "9:00 AM", "6 hr 15 min", "3:15 PM", 265);

TrainInfo trainInfo18=new TrainInfo("02295", "SANGHAMITRA EXP", "Bangalore", "Chennai", "2S", "9:00 AM", "6 hr 15 min", "3:15 PM", 160);

//chennai to bangalore

TrainInfo trainInfo19=new TrainInfo("02296", "DNR SBC SPL", "Chennai","Bangalore", "2A", "1:55 PM", "6 hr 25 min", "8:20 PM", 945);

TrainInfo trainInfo20=new TrainInfo("02296", "DNR SBC SPL", "Chennai","Bangalore", "3A", "1:55 PM", "6 hr 25 min", "8:20 PM", 680);

TrainInfo trainInfo21=new TrainInfo("02296", "DNR SBC SPL", "Chennai","Bangalore", "SL", "1:55 PM", "6 hr 25 min", "8:20 PM", 265);

TrainInfo trainInfo22=new TrainInfo("02296", "DNR SBC SPL", "Chennai","Bangalore", "2S", "1:55 PM", "6 hr 25 min", "8:20 PM", 160);

trains.add(trainInfo1);

trains.add(trainInfo2);

trains.add(trainInfo3);

trains.add(trainInfo4);

trains.add(trainInfo5);

trains.add(trainInfo6);

trains.add(trainInfo7);

trains.add(trainInfo8);

trains.add(trainInfo9);

trains.add(trainInfo10);

trains.add(trainInfo11);

trains.add(trainInfo12);

trains.add(trainInfo13);

trains.add(trainInfo14);

trains.add(trainInfo15);

trains.add(trainInfo16);

trains.add(trainInfo17);

trains.add(trainInfo18);

trains.add(trainInfo19);

trains.add(trainInfo20);

trains.add(trainInfo21);

trains.add(trainInfo22);

System.out.println("trains2");

return trains;

}

}

SkeletonValidator.java

package com.cts.irctc.skeletonvalidator;

import java.lang.reflect.Method;

import java.util.Set;

import java.util.logging.Level;

import java.util.logging.Logger;

public class SkeletonValidator {

public SkeletonValidator() {

validateClassName("com.cts.irctc.model.TicketBooking");

validateClassName("com.cts.irctc.model.TrainInfo");

validateClassName("com.cts.irctc.service.IrctcService");

validateClassName("com.cts.irctc.controller.IrctcController");

validateClassName("com.cts.irctc.exception.ApplicationException");

validateClassName("com.cts.irctc.exception.ExceptionHandlerControllerAdvice");

validateMethodSignature("getTicketBookingResult:com.cts.irctc.model.TrainInfo,getPNRNumber:java.lang.String",

"com.cts.irctc.service.IrctcService");

validateMethodSignature("showTicketBookingForm:java.lang.String,getCarSearchResultForm:java.lang.String,populateFromCities:java.util.List,populateToCities:java.util.List,populateClassType:java.util.List",

"com.cts.irctc.controller.IrctcController");

}

private static final Logger LOG = Logger.getLogger("SkeletonValidator");

protected final boolean validateClassName(String className) {

boolean iscorrect = false;

try {

Class.forName(className);

iscorrect = true;

LOG.info("Class Name " + className + " is correct");

} catch (ClassNotFoundException e) {

LOG.log(Level.SEVERE, "You have changed either the " + "class name/package. Use the correct package "

+ "and class name as provided in the skeleton");

} catch (Exception e) {

LOG.log(Level.SEVERE,

"There is an error in validating the " + "Class Name. Please manually verify that the "

+ "Class name is same as skeleton before uploading");

}

return iscorrect;

}

protected final void validateMethodSignature(String methodWithExcptn, String className) {

Class cls = null;

try {

String[] actualmethods = methodWithExcptn.split(",");

boolean errorFlag = false;

String[] methodSignature;

String methodName = null;

String returnType = null;

for (String singleMethod : actualmethods) {

boolean foundMethod = false;

methodSignature = singleMethod.split(":");

methodName = methodSignature[0];

returnType = methodSignature[1];

cls = Class.forName(className);

Method[] methods = cls.getMethods();

for (Method findMethod : methods) {

if (methodName.equals(findMethod.getName())) {

foundMethod = true;

if (!(findMethod.getReturnType().getName().equals(returnType))) {

errorFlag = true;

LOG.log(Level.SEVERE, " You have changed the " + "return type in '" + methodName

+ "' method. Please stick to the " + "skeleton provided");

} else {

LOG.info("Method signature of " + methodName + " is valid");

}

}

}

if (!foundMethod) {

errorFlag = true;

LOG.log(Level.SEVERE, " Unable to find the given public method " + methodName

+ ". Do not change the " + "given public method name. " + "Verify it with the skeleton");

}

}

if (!errorFlag) {

LOG.info("Method signature is valid");

}

} catch (Exception e) {

LOG.log(Level.SEVERE,

" There is an error in validating the " + "method structure. Please manually verify that the "

+ "Method signature is same as the skeleton before uploading");

}

}

}