Grade settings: Maximum grade: 100
Based on: Travel Outstation Cost Estimator - V1
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum execution time**: 240 s **Maximum memory used**: 1.50 GiB **Maximum execution file size**: 1.50 GiB

Travel Outstation Cost Estimator

Create a Spring MVC Spring Boot Application for developing a Outstation Travel cost Estimation application. Design a cost Estimation Page to choose an appropriate Service (Train or Car or Flight) and Distance of the travel. On clicking Estimate Cost, the application should calculate the cost estimation depending on the transportation type chosen and the Distance of the travel.The user has to then be redirected to billdesk.jsp page that displays the message "**Thank you for choosing our Service. Your estimated travel cost is Rs. << estimatedCost >>"**

Application Work Flow:

BillController is the Controller class.

TravelBean is the model class with three attributes transportationType, transportationCost, distance and getters and setters for all the attributes mentioned.

TravelService which is the Service class has a method called estimateTransportationCost that takes TravelBean as its argument and returns double**.**

- This method needs to set the transportationCost instance variable based on the cost for distance (in Rs) value given in the below table.
- Calculate the transportationCost depending on the transportationType chosen and Distance of the travel.
- Calculate the estimatedCost depending on the transportationCost +1750(for *maintenance and others*).
- estimatedCost should be returned as double.

Transportation Type	**Distance of the travel (in Kms)** **Less than or equal to 250**	**Distance of the travel (in Kms)** **Greater than 250 and less than or equal to 500**	**Distance of the travel (in Kms)** **Greater than 500**
Train	Rs. 10 (per Kilometer)	Rs. 9 (per Kilometer)	Rs. 8 (per Kilometer)
Car	Rs. 15 (per Kilometer)	Rs. 14 (per Kilometer)	Rs. 13 (per Kilometer)
Flight	Rs. 30 (per Kilometer)	Rs. 25 (per Kilometer)	Rs. 20 (per Kilometer)

For example: If the transportation Type is Car and Distance of the travel is 300 then the transportationCost should be set as 4200.0 (300 * 14) and the estimated cost returned should be 4200.0+1750 = 5950.0

TravelService should be autowired inside the BillController.

Initially, the user should be routed via the BillController's showPage method to showpage.jsp that allows user to choose the transportation type and enter the Distance of the travel.

[Note: showPage method has to be written inside the BillController]

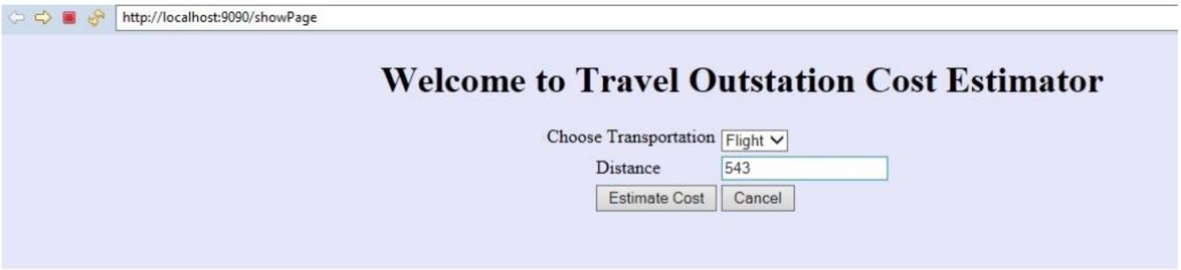
A method in the BillController known as buildState should have a ModelAttribute "transportationList". This method should populate the transportations(Train, Car, Flight) in the Map as a Key=Value pair. (Both key and value be the same ServiceName.Eg: Key- Train, Value- Train, Key-Car,Value- Car) and then return the Map. This should be then used to autopopulate the transportations in the showpage.jsp.

[Note:buildState method should be written inside the BillController]

On clicking the Estimate Cost button, the BillController's estimateTransportationCost method should be called. This method takes two arguments - model attribute named "travel" which holds the form populated TravelBean Object and the ModelMap.

- This method should calculate the estimatedCost by invoking the estimateTransportationCost method of the TravelService.

Screen designs and Expected Output:



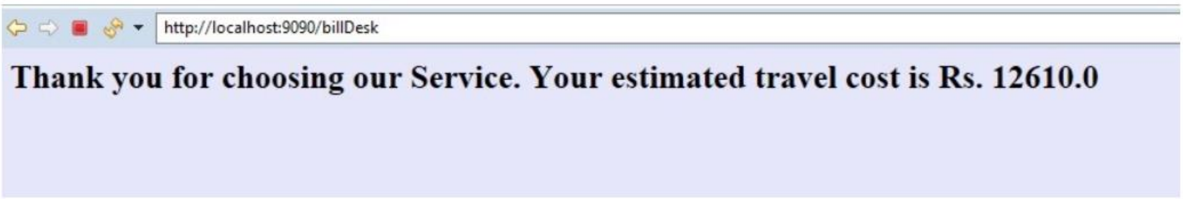
http://localhost:9090/showPage

Welcome to Travel Outstation Cost Estimator

Choose Transportation: Flight

Distance: 543

Â· Redirect the user to billdesk.jsp page with a message "**Thank you for choosing our Service. Your estimated travel cost is Rs. << estimatedCost >>**".



http://localhost:9090/billDesk

Thank you for choosing our Service. Your estimated travel cost is Rs. 12610.0

Design Constraints:

UI Design Constraints:

showpage.jsp		
Component	**ID**	**Constraints**
Select	transportationType	**Should be auto populated using the model attribute written above the buildState method, inside the BillController. Do not hard code**

		the values
Textbox	distance	-
submit	submit (Name)	-

billdesk.jsp

***The Result has to be rendered in the <h2> tag**

Component Specification

Controller

BillController			
AttributeName	**AttributeType**	**Access Specifier**	**Constraints**
travelService	TravelService	private	Use annotation to Autowire

BillController			
Method Name	**Method Argument name:type**	**Return type**	**RequestMapping URL & Request Method**
showPage	modelAttribute "travel":TravelBean	String	/showPage & GET
estimateTransportationCost	modelAttribute "travel":TravelBean, model:ModelMap	String	/billDesk & POST
buildState		Map <String,String>	Should be annotated with ModelAttribute with name "transportationList"

Service

TravelService		
Method Name	**Method Argument name:type**	**Return type**
estimateTransportationCost	travelBean:TravelBean	double

Model

TravelBean	
AttributeName	**AttributeType**
transportationType	String
transportationCost	double
distance	int

Overall Design Constraints:

- BillController should be inside the package com.controller
- TravelService should be inside the package com.service
- TravelBean should be in the package com.model
- Use appropriate annotation to configure TravelService as a Service
- Use appropriate annotation to configure BillController as a Controller
- TravelService should be autowired inside the BillController.
- Do not change the property name given in the application.properties files, you can change the value and you can include additional property if needed.
- In the pom.xml you are provided with all the dependencies needed for developing the application.
- You will not be evaluated based on the UI design (layout, color, formatting, etc.). You are free to have a basic UI with all the required UI components (input fields, buttons, labels, etc.). Expected components with the id alone should be designed as per the requirement.
- Adhere to the design specifications mentioned in the case study.

- **Do not change or delete the class/method names or return types which are provided to you as a part of the base code skeleton.**
- Please make sure that your code does not have any compilation errors while submitting your case study solution.

Automatic evaluation[+]

TravelCostEstimator/pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>2.1.8.RELEASE</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>com.controller</groupId>
	<artifactId>HolidayParty_new</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>HolidayParty_new</name>
	<description>Demo project for Spring Boot</description>

	<properties>
		<java.version>1.8</java.version>
	</properties>

	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
		</dependency>

		 <dependency>
     <groupId>org.apache.tomcat.embed</groupId>
     <artifactId>tomcat-embed-jasper</artifactId>
     <scope>provided</scope>
   </dependency>
   <dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>servlet-api</artifactId>
  <version>2.5</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>javax.servlet.jsp</groupId>
  <artifactId>jsp-api</artifactId>
  <version>2.1</version>
  <scope>provided</scope>
</dependency>

<dependency>
  <groupId>taglibs</groupId>
  <artifactId>standard</artifactId>
  <version>1.1.2</version>
</dependency>
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>jstl</artifactId>
  <version>1.2</version>
</dependency>
<!-- Spring test dependencies -->
	<dependency>
			<groupId>org.mockito</groupId>
			<artifactId>mockito-all</artifactId>
			<version>1.10.19</version>
```

```
            <scope>test</scope>
        </dependency>

        <!-- <dependency>
            <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-java</artifactId>
            <version>2.53.0</version>
        </dependency>-->

        <dependency>
  <groupId>org.seleniumhq.selenium</groupId>
  <artifactId>htmlunit-driver</artifactId>
  <version>2.26</version>
</dependency>

    <!-- https://mvnrepository.com/artifact/org.w3c.css/sac -->
<dependency>
  <groupId>org.w3c.css</groupId>
  <artifactId>sac</artifactId>
  <version>1.3</version>
</dependency>


<dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-test</artifactId>
             <version>5.1.7.RELEASE</version>  <!-- 4.0.5 -->
            <scope>test</scope>
        </dependency>
<!-- End of spring test dependencies -->



        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
            <exclusions>
                <exclusion>
                    <groupId>org.junit.vintage</groupId>
                    <artifactId>junit-vintage-engine</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>


```

TravelCostEstimator/src/main/java/com/controller/BillController.java

```java
package com.controller;

import java.util.HashMap;
import java.util.Map;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import com.model.TravelBean;
import com.service.TravelService;

//use appropriate annotation to configure BillController as Controller
@Controller
public class BillController {

    @Autowired
    private TravelService travelService;

    //invoke the service class - estimateTransportationCost method.
    @RequestMapping(value = "/showPage", method = RequestMethod.GET)
    public String showPage(@ModelAttribute("travel") TravelBean travelBean) {

        return "showpage";
    }

    @ModelAttribute("transportationList")
    public  Map<String, String> buildState(){

        Map<String, String> serviceMap = new HashMap<String, String>();
        serviceMap.put("Train", "Train");
        serviceMap.put("Car", "Car");
        serviceMap.put("Flight", "Flight");
        return serviceMap;

    }

    @RequestMapping(value = "/billDesk", method = RequestMethod.POST)
    public String estimateTransportationCost(@ModelAttribute("travel") TravelBean
travelBean,ModelMap model) {

        double cost=travelService.estimateTransportationCost(travelBean);


        model.addAttribute("cost", cost);
        return "billdesk";
    }



}


```

TravelCostEstimator/src/main/java/com/example/demo/TravelCostEstimatorApplication.java

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication
@ComponentScan({"com.controller","com.model","com.service"})

public class TravelCostEstimatorApplication {

    public static void main(String[] args) {
        SpringApplication.run(TravelCostEstimatorApplication.class, args);
    }

}

```

TravelCostEstimator/src/main/java/com/model/TravelBean.java

```
package com.model;

import org.springframework.stereotype.Component;

//pojo class with required attributes,getters and setters

@Component
public class TravelBean {

    private String transportationType;
    private double transportationCost;
    private int distance;
    public String getTransportationType() {
        return transportationType;
    }
    public void setTransportationType(String transportationType) {
        this.transportationType = transportationType;
    }
    public double getTransportationCost() {
        return transportationCost;
    }
    public void setTransportationCost(double transportationCost) {
        this.transportationCost = transportationCost;
    }
    public int getDistance() {
        return distance;
    }
    public void setDistance(int distance) {
        this.distance = distance;
    }



}


```

TravelCostEstimator/src/main/java/com/service/TravelService.java

```
package com.service;

import org.springframework.stereotype.Service;

import com.model.TravelBean;

//use appropriate annotation to configure ParcelTransportation as a Transportation
@Service
public class TravelService {

    // calculate the estimatedCost and return the cost
    public double estimateTransportationCost(TravelBean travelBean) {
        String transportationType = travelBean.getTransportationType();
        int distance = travelBean.getDistance();
//      System.out.println("distance"+distance);
        double cost = 0;


        switch (transportationType) {
        case "Train": {
            if (distance <= 250) {
                cost = distance * 10;
            }
            if (distance > 250 && distance <= 500) {
                cost = distance * 9;
            }
            if (distance > 500) {
                cost = distance * 8;
//              System.out.println("cost:"+cost);
            }
            break;
        }
        case "Car": {
            if (distance <= 250) {
                cost = distance * 15;
            }
            if (distance > 250 && distance <= 500) {
                cost = distance * 14;
            }
            if (distance > 500) {
                cost = distance * 13;
            }
            break;
        }
        case "Flight": {
            if (distance <= 250) {
                cost = distance * 30;
            }
            if (distance > 250 && distance <= 500) {
                cost = distance * 25;
            }
            if (distance > 500) {
                cost = distance * 20;
            }
            break;
        }
        default:
     {

     }
        }
        double estimatedCost=cost+1750;
//      System.out.println("estimatedCost"+estimatedCost);
        travelBean.setTransportationCost(estimatedCost);

        return estimatedCost;

    }

}


```

TravelCostEstimator/src/main/resources/application.properties

```
#Do not change the key server.port
#Its value port number can be changed

#Include other required configurations here

server.port=9090
spring.mvc.view.prefix = /WEB-INF/views/
spring.mvc.view.suffix = .jsp
spring.mvc.static-path-pattern=/resources/**

```

TravelCostEstimator/src/main/webapp/WEB-INF/views/billdesk.jsp

```
<%@page isELIgnored="false" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<body bgcolor="lavender">
<h2>Thank you for choosing our Service. Your estimated travel cost is Rs. ${cost}</h2>
</body>
</html>
```

TravelCostEstimator/src/main/webapp/WEB-INF/views/showpage.jsp

```
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>
<%@ taglib prefix="sf" uri="http://www.springframework.org/tags/form" %>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" isELIgnored="false"%>


<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<body style="background-color:lavender">
<h1><center> Welcome to Travel Outstation Cost Estimator</center></h1>
<form:form method="post" action="billDesk" modelAttribute="travel">
<table style="margin: 0px auto; margin-left: auto; margin-right:auto">


            <tr>
                <td>Choose Service:</td>
                <td>
                 <form:select path="transportationType" id="transportationType">
                        <form:options items="${transportationList}"/>
                    </form:select>

                </td>
            </tr>

            <tr>
            <td>Distance</td><td><form:input path="distance" id="distance"/></td>
            <tr>
                <td><input type="submit" value="Estimate Cost" name="submit" /></td>
                <td><input type="reset" value="Cancel"/></td>
            </tr>

        </table>
</form:form>


</body>
</html>

```

Grade

Reviewed on Friday, 18 February 2022, 12:28 AM by Automatic grade

Grade 100 / 100

Assessment report

Assessment Completed Successfully

[-]Grading and Feedback

```
Feature Test - 7.50 / 7.50(Success)
          * check whether the Controller / Service class with the required annotation is implemented - 2.50 / 2.50
          * check whether the Controller class with the required annotation is implemented - 1.50 / 1.50
```

```
         * check whether the Service class with the required annotation is implemented -
1.50 / 1.50
         * check whether the model class with the required annotation is implemented -
2.00 / 2.00

Functional testing - 45.00 / 45.00(Success)
         * check whether the travel cost  for Train is calculated - 15.00 / 15.00
         * check whether the travel cost for the Car is calculated - 15.00 / 15.00
         * check whether the travel cost for the Flight is calculated - 15.00 / 15.00

Spring Testing in the Controller - 25.00 / 25.00(Success)
         * check whether the request is mapped for showpage and redirected to billdesk jsp
page - 10.00 / 10.00
         * check whether the model attribute is specified with the name above the
buildState method that holds the Map containing the transportation types - 2.50 / 2.50
         * check whether the model attribute with the name transportationList is specified
above the buildState method that holds the Map containing the Travel Services - 2.50 / 2.50
         * check whether layering is followed that is whether travel cost is calculated by
invoking the service method inside the Controller - 10.00 / 10.00

UI Testing - 17.50 / 17.50(Success)
         * check whether UI from showpage  to billdesk page is redirected after
calculating the travelcost for Car - 5.00 / 5.00
         * check whether UI from showpage  to billdesk page is redirected after
calculating the travelcost for Train - 5.00 / 5.00
         * check whether UI from showpage  to billdesk page is redirected after
calculating the travelcost for Flight - 5.00 / 5.00
         * check whether transportationType component is rendered in showpage as expected
- 1.50 / 1.50
         * check whether distance component is rendered in showpage as expected - 1.00 /
1.00

Comments and best practices/standards - 5.0 / 5.0(Success)
```