Grade settings: Maximum grade: 100
Based on: Sai Home Builders - 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

Sai Home Builders

Sai Home Builders is one of the leading builders in Tamilnadu, they approaches you to create a separate online application for calculating building coverage ratio based on property type (Commercial/Residential) and inform the customer whether the plan is approved or not.

Create a Spring MVC Spring Boot Application for developing a Building Coverage Ratio Application. Design a Building Coverage Ration Estimator Page to choose an appropriate building Type **("Commercial", "Residential")** and enter the building area and site area.

On clicking Calculate button, the application should calculate the BCR based on the building type chosen and the entered building area and site area. The customer has to then be redirected to **result.jsp** page that displays the message "**Welcome to Sai Home Builders. Building Coverage Ratio for your property is <<ratio>>, Your plan is <<approvalStatus>>"**

Application Work Flow:

- **BuildingController** is the Controller class.
- **Building** is the model class with four attributes buildingType, buildingArea, siteArea and approvalStatus along with its getters and setters.
- **BuildingService** is the Service class which has a method called **calculateBCR** that takes Building as its argument and returns double.

1. Calculate the BCR depending on the buildingArea and siteArea for the buildingType chosen by the customer.
2. This method should set the approvalStatus instance variable as "**Approved or NotApproved"** based on BCR value given in the below table.
3. It should be returned as the double.

Example:

Consider building type is ="**Commercial"** with building area is **500.0** and site area is **1000.0**

BCR = (buildingArea / siteArea) * 100.0 = (500.0/1000.0) *100.0 = 50.0

If buildingType is "commercial" and BCR is greater than 85.5 set,

approvalStatus = Not Approved else set,

approvalStatus = Approved

Note: siteArea should be always greater than buildingArea

buildingType	BCR	approvalStatus
Commercial	>85.5	NotApproved
Residential	>75.5	NotApproved

Initially, the customer should be routed via the BuildingControllerâ€™s **calculateBCRPage** method to **BCRpage.jsp** that allows customer to choose the buildingType and enter the buildingArea and siteArea.

[Note: calculateBCRPage method has to be written inside the BuildingController]

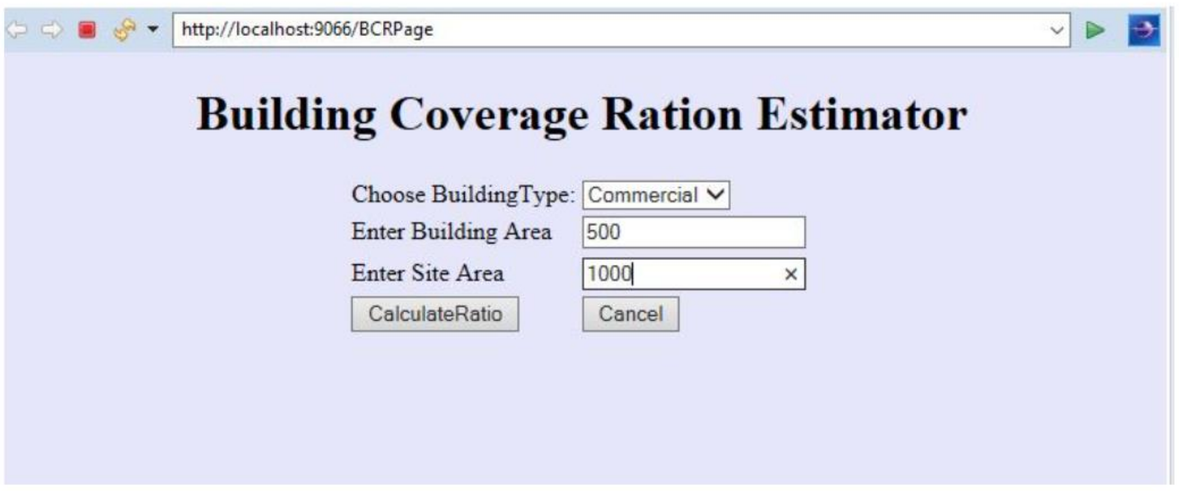
A method in the **BuildingController** known as buildState should be annotated with the ModelAttribute "**buildingList"**. This method should populate the building types (Commercial, Residential) the Map as key-value pair. Both key and value be the same buildingType. (Eg: Key-Residential, Value- Residential, Key- Commercial, Value- Commercial) and then return the Map. This should be then used to autopopulate the buildingType in the **BCRpage.jsp**

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

On clicking the Calculate button, the BuildingControllerâ€™s **calculateBuildingCoverageRatio** method should be called. This method takes three arguments - model attribute named "**building"** which holds the form populated Building Object, **BindingResult** and the **ModelMap**.

- This method should calculate the **BCR** by invoking the **calculateBuildingCoverageRatio** method of the **BuildingService**.

BCRpage.jsp


http://localhost:9066/BCRPage

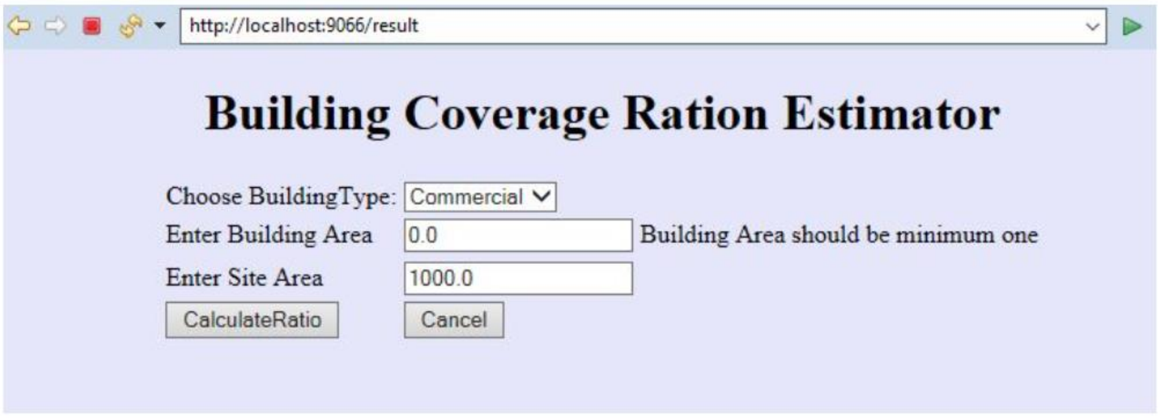
Building Coverage Ration Estimator

Choose BuildingType: Commercial
Enter Building Area 500
Enter Site Area 1000

- If the buildingArea entered by the customer is less than 1 then an error messages "Building Area should be minimum one" has to be displayed.

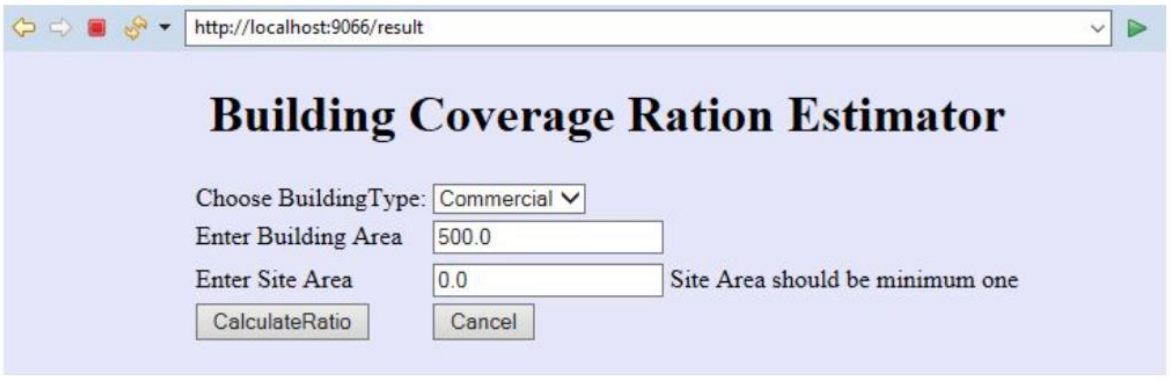
- If the siteArea entered by the customer is less than 1 then an error messages "Site Area should be minimum one" has to be displayed.

BCRpage.jsp


http://localhost:9066/result

Building Coverage Ration Estimator

Choose BuildingType: Commercial
Enter Building Area 0.0 Building Area should be minimum one
Enter Site Area 1000.0

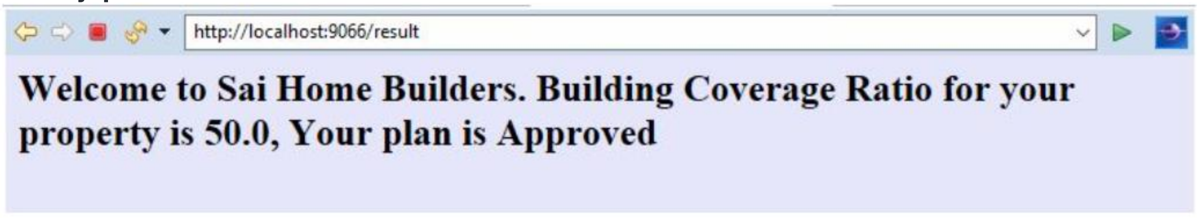

http://localhost:9066/result

Building Coverage Ration Estimator

Choose BuildingType: Commercial
Enter Building Area 500.0
Enter Site Area 0.0 Site Area should be minimum one

Redirect the customer to result.jsp page with a message "**Welcome to Sai Home Builders. Building Coverage Ratio for your property is <<ratio>>, Your plan is <<approvalStatus>>"**

result.jsp


http://localhost:9066/result

Welcome to Sai Home Builders. Building Coverage Ratio for your property is 50.0, Your plan is Approved

Design Constraints:

UI Design Constraints:

BCRpage.jsp		
Component	**ID**	**Constraints**
Select	buildingType	Should be auto populated using the model attribute written above the buildState method inside the BuildingController. Do not hard code the values
Textbox	buildingArea	Minimum value should be 1
Textbox	siteArea	Minimum value should be 1
submit	submit	-

Note:

- **In result.jsp, the Result has to be rendered in the <h2> tag**

Component Specification:

Controller

BuildingController			
AttributeName	**AttributeType**	**Access Specifier**	**Constraints**
service	BuildingService	private	Use annotation to Autowire

BuildingController			
Method Name	**Method Argument name: type**	**Return type**	**RequestMapping URL & Request Method**
calculateBCRPage	modelAttribute "building":Building	String	/BCRPage & GET
calculateBuildingCoverageRatio	modelAttribute "building":Building, result:BindingResult, map:ModelMap	String	/result & POST
buildState		Map <String,String>	Should be annotated with ModelAttribute with name "buildingList"

Service

BuildingService		
Method Name	**Method Argument name: type**	**Return type**
calculateBuildingCoverageRatio	building:Building	double

Model

Building	
AttributeName	**AttributeType**
buildingType	String
buildingArea	double
siteArea	double
approvalStatus	String

Overall Design Constraints:

- **BuildingController** should be inside the package **com.controller**
- **BuildingService** should be inside the package **com.service**
- **Building** should be in the package **com.model**
- Use appropriate annotation to configure **BuildingService** as a **Service**
- Use appropriate annotation to configure **BuildingController** as a **Controller**
- **BuildingService** should be **autowired** inside the **BuildingController**.
- Use annotations to implement the business Validation as specified in the screen shot **[That is, when the buildingArea or siteArea is less than 1 then error message "building area / site area should be minimum one" should be rendered in the UI]**
- **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[+]

SaiHomeBuilders/pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 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>SaiHomeBuilders</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>SaiHomeBuilders</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>
```

```xml
			<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>


```

SaiHomeBuilders/src/main/java/com/controller/BuildingController.java

```java
package com.controller;

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

import javax.validation.Valid;

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 org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;

import com.model.Building;
import com.service.BuildingService;

@Controller
public class BuildingController {

	@Autowired
	private BuildingService service;

	@RequestMapping(value = "/BCRPage", method = RequestMethod.GET)
	public String calculateBCRPage(@ModelAttribute("building") Building building)
	{
		return "BCRpage";
	}

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

		Map<String, String> serviceMap = new HashMap<String, String>();
		serviceMap.put("Commercial", "Commercial");
		serviceMap.put("Residential", "Residential");
		return serviceMap;
	}

	@RequestMapping(value = "/result", method = RequestMethod.POST)
	public String calculateBuildingCoverageRatio(@Valid @ModelAttribute("building") Building building,
					BindingResult result,ModelMap map)
	{
			if(result.hasErrors()) {
					return "BCRpage";
	}
			else
			{
			double cost=service.calculateBCR(building);
			String status =building.getApprovalStatus();
			map.addAttribute("status",status);
			map.addAttribute("cost",cost);
			return "result";

			}
	}

}

```

SaiHomeBuilders/src/main/java/com/example/demo/SaiHomeBuildersApplication.java

```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 SaiHomeBuildersApplication {

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

}

```

SaiHomeBuilders/src/main/java/com/model/Building.java

```java
package com.model;

import javax.validation.constraints.Min;

import org.springframework.stereotype.Component;

@Component
public class Building {

	private String buildingType;
	@Min(value=1, message="Building Area should be minimum one")
	private double buildingArea;
	@Min(value=1, message="Site Area should be minimum one")
	private double siteArea;
	private String approvalStatus;

	public String getBuildingType() {
			return buildingType;
	}
	public void setBuildingType(String buildingType) {
			this.buildingType = buildingType;
	}
	public double getBuildingArea() {
			return buildingArea;
	}
	public void setBuildingArea(double buildingArea) {
		this.buildingArea = buildingArea;
	}
	public double getSiteArea() {
		return siteArea;
	}
	public void setSiteArea(double siteArea) {
		this.siteArea = siteArea;
	}
	public String getApprovalStatus() {
		return approvalStatus;
	}
	public void setApprovalStatus(String approvalStatus) {
		this.approvalStatus = approvalStatus;
	}



}

```

SaiHomeBuilders/src/main/java/com/service/BuildingService.java

```java
package com.service;

import org.springframework.stereotype.Service;

import com.model.Building;

@Service
public class BuildingService {

	public double calculateBCR(Building obj)
	{
		double bcr=0.0;
		bcr = (obj.getBuildingArea()/obj.getSiteArea())*100.0;
		if(obj.getBuildingType().equals("Commercial"))
		{
			if(bcr>85.5) {
				obj.setApprovalStatus("Not Approved");
			}
			else {
				obj.setApprovalStatus("Approved");
			}
		}
		else if(obj.getBuildingType().equals("Residential"))
		{
			if(bcr>75.5) {
				obj.setApprovalStatus("Not Approved");
			}
			else {
				obj.setApprovalStatus("Approved");
			}
		}
		return bcr;
	}

}


```

SaiHomeBuilders/src/main/resources/application.properties

```
server.port=9066
spring.mvc.view.prefix = /WEB-INF/views/
spring.mvc.view.suffix = .jsp
spring.mvc.static-path-pattern=/resources/**
```

SaiHomeBuilders/src/main/webapp/WEB-INF/views/BCRpage.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> Building Coverage Ration Estimator </center></h1>
<form:form method="post" action="result" modelAttribute="building">

<table style="margin: 0px auto; margin-left: auto; margin-right:auto">

                        <tr>
                                <td>Choose BuildingType:</td>
                                <td>
                                 <form:select path="buildingType" id="buildingType">
                                                <form:options items="${buildingList}"/>
                                        </form:select>

                                </td>
                        </tr>

                        <tr>
                        <td>Enter Building Area</td><td><form:input path="buildingArea" id="buildingArea"/></td><td><form:errors path="buildingArea"/></tr>
                        <tr>
                        <tr>
                        <td>Enter Site Area</td><td><form:input path="siteArea" id="siteArea"/></td><td><form:errors path="siteArea"/></tr>
                        <tr>
                                <td><input type="submit" value="CalculateRatio" id="submit" /></td>
                                <td><input type="reset" value="Cancel"/></td>
                        </tr>

                </table>
</form:form>


</body>
</html>

```

SaiHomeBuilders/src/main/webapp/WEB-INF/views/result.jsp

```
<%@page isELIgnored="false" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<body bgcolor="lavender">
<h2>Welcome to Sai Home Builders. Building Coverage Ratio for your property is ${cost}, Your plan is ${status}</h2>
</body>
</html>
```

Grade

Reviewed on Friday, 18 February 2022, 12:30 AM by Automatic grade
Grade 98.75 / 100

Assessment report

[-]SOURCE CODE ANALYZER REPORT

Error Msg: A method should have only one exit point, and that should be the last statement in the method
Variable Name:
Class Name: BuildingController

Assessment Completed Successfully

[-]Grading and Feedback

```
Feature Test - 15.00 / 15.00(Success)
          * check whether Service is configured in Controller class  with the required Annotation - 4.50 / 4.50
          * check whether the Controller class with the required annotation is implemented - 3.50 / 3.50
          * check whether the Service class with the required annotation is implemented - 3.50 / 3.50
          * check whether the model class with the required annotation is implemented - 3.50 / 3.50

Functional testing - 15.00 / 15.00(Success)
          * check whether the building coverage ratio for the commercial building is calculated as per the requirement - 7.50 / 7.50
          * check whether the building coverage ratio for the Residential building is calculated as per the requirement - 7.50 / 7.50

Spring Testing in the Controller - 35.00 / 35.00(Success)
          * check whether the request is mapped for BCRpage and redirected to BCR jsp page - 7.50 / 7.50
          * check whether the model attribute is specified above the buildState method that holds the Map containing the buildingTypes - 7.50 / 7.50
          * check whether the model attribute with the name buildingList is specified above the buildState method that holds the Map containing the buildingTypes - 7.50 / 7.50
          * check whether layering is followed that is whether BCR is calculated by invoking the service method inside the Controller - 7.50 / 7.50
          * check whether the validations are working correctly - 5.00 / 5.00

UI Testing - 30.00 / 30.00(Success)
          * check whether UI from BCRpage  to result page is redirected after calculating the BCR for Residential building - 10.00 / 10.00
          * check whether UI from BCRpage  to result page is redirected after calculating the BCR for Commercial building - 10.00 / 10.00
          * check whether buildingType is rendered in BCR page as expected - 4.00 / 4.00
          * check whether the BCR jsp page is rendered - 3.00 / 3.00
          * check whether siteArea is rendered in BCR page as expected - 3.00 / 3.00

Comments and best practices/standards - 3.75 / 5.0(Partial)
```