

# Construction Cost Estimator - V1

**Grade settings:** Maximum grade: 100

**Based on:** [Construction 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

## Construction Cost Estimator

Smart Builders is one of the leading builders in Tamilnadu, they approaches you to create a separate online application for calculating construction cost based on bricks type (UnburntClayBricks/BurntClayBricks/ConcreteBricks).

Create a Spring MVC Spring Boot Application for developing a Construction Cost Estimator Application. Design a Construction Cost Estimator Page to choose an appropriate bricks Type (**UnburntClayBricks/BurntClayBricks/ConcreteBricks**) and enter the buildup area.

On clicking CalculateCost button, the application should calculate the Construction cost based on the bricks type chosen and the entered buildup area. The customer has to then be redirected to result.jsp page that displays the message "**Approximate construction cost for given parameters is Rs.<<constructionCost>>**"

### Application Work Flow:

- **ConstructionController** is the Controller class.
  - **Construction** is the model class with three attributes **bricksType**, **costPerSqFt** and **buildupArea** along with its getters and setters.
  - **ConstructionService** is the Service class which has a method called **calculateConstructionCost** that takes Invoice as its argument and returns double.
1. Calculate the construction cost depending on the rate per square feet based on the bricksType selected and the buildupArea given by the customer.
  2. This method should set the **costPerSqFt** instance variable based on bricks type value given in the below table.
  3. It should be returned as the double.

### Example:

- If buildupArea is 1200 and bricksType is ConcreteBricks then set costPerSqFt = 300.00
- totalCost = buildupArea \* costPerSqFt = 1200 \* 300.00 = 360000.00

bricksType	costPerSqFt
UnburntClayBricks	340.00
BurntClayBricks	390.00
ConcreteBricks	300.00

Initially, the customer should be routed via the ConstructionController's **estimatorPage** method to **estimatorpage.jsp** that allows customer to choose the bricksType and enter the buildupArea.

[Note: estimatorPage method has to be written inside the ConstructionController]

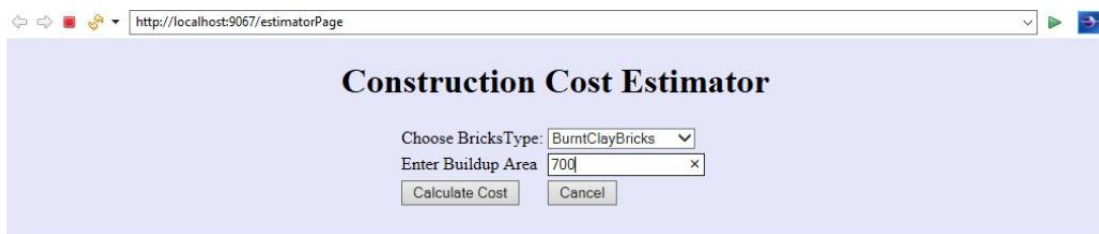
A method in the **ConstructionController** known as buildState should be annotated with the ModelAttribute "**bricksList**". This method should populate the bricks type (**UnburntClayBricks/BurntClayBricks/ConcreteBricks**) the Map as key-value pair. Both key and value be the same buildingType. (Eg: Key- **ConcreteBricks**, Value- **ConcreteBricks**) and then return the Map. This should be then used to autopopulate the bricksType in the **estimatorpage.jsp**

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

On clicking the Calculate Cost button, the ConstructionController's **calculateConstructionCost** method should be called. This method takes three arguments - model attribute named "**construction**" which holds the form populated **Construction** Object, BindingResult and the ModelMap.

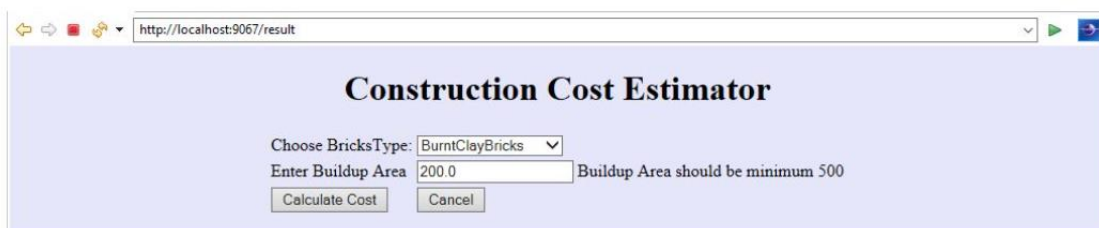
- This method should calculate the **constructionCost** by invoking the **calculateConstructionCost** method of the **ConstructionService**.

estimatorpage.jsp



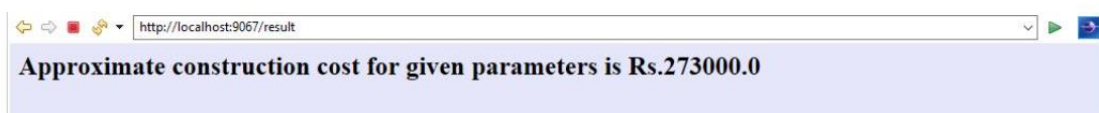
- If the buildupArea entered by the customer is less than 500 then an error message "Buildup Area should be minimum 500" has to be displayed.

estimatorpage.jsp



Redirect the customer to result.jsp page with a message "Approximate construction cost for given parameters is Rs. <<constructionCost>>"

result.jsp



### Design Constraints:

### UI Design Constraints:

estimatorpage.jsp		
Component	ID	Constraints
Select	bricksType	Should be auto populated using the model attribute written above the <b>buildState</b> method inside the <b>ConstructionController</b> . Do not hard code the values
Textbox	buildupArea	Minimum value should be 500
submit	submit	-

### Note:

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

### Component Specification:

#### Controller

ConstructionController			
AttributeName	AttributeType	Access Specifier	Constraints
service	ConstructionService	private	Use annotation to Autowire

ConstructionController			
Method Name	Method Argument name: type	Return type	RequestMapping URL & Request Method
estimatorPage	ModelAttribute "construction": Construction	String	/ estimatorPage & GET
calculateConstructionCost	ModelAttribute "construction": Construction, result:BindingResult, map:ModelMap	String	/result & POST
buildState		Map <String,String>	Should be annotated with ModelAttribute with name " <b>bricksList</b> "

#### Service

ConstructionService		
Method Name	Method Argument name: type	Return type
calculateConstructionCost	construction: Construction	double

## Model

Construction	
AttributeName	AttributeType
bricksType	String
buildupArea	double
costPerSqFt	double

### Overall Design Constraints:

- **ConstructionController** should be inside the package **com.controller**
- **ConstructionService** should be inside the package **com.service**
- **Construction** should be in the package **com.model**
- Use appropriate annotation to configure ConstructionService as a Service
- Use appropriate annotation to configure ConstructionController as a Controller
- ConstructionService should be autowired inside the BuildingController.
- Use annotations to implement the business Validation as specified in the screen shot **[That is, when the buildupArea is less than 500 then error message "Buildup Area should be minimum 500" 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.

Submitted by sathya .

## Automatic evaluation[+]

### ConstructionCostEstimator/pom.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
3     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-
4.0.0.xsd">
4     <modelVersion>4.0.0</modelVersion>
5     <parent>
6         <groupId>org.springframework.boot</groupId>
7         <artifactId>spring-boot-starter-parent</artifactId>
8         <version>2.1.8.RELEASE</version>
```

```

9             <relativePath/> <!-- lookup parent from repository -->
10         </parent>
11         <groupId>com.controller</groupId>
12         <artifactId>ConstructionCostEstimator</artifactId>
13         <version>0.0.1-SNAPSHOT</version>
14         <name>ConstructionCostEstimator</name>
15         <description>Demo project for Spring Boot</description>
16
17         <properties>
18             <java.version>1.8</java.version>
19         </properties>
20
21         <dependencies>
22             <dependency>
23                 <groupId>org.springframework.boot</groupId>
24                 <artifactId>spring-boot-starter-web</artifactId>
25             </dependency>
26
27             <dependency>
28                 <groupId>org.apache.tomcat.embed</groupId>
29                 <artifactId>tomcat-embed-jasper</artifactId>
30                 <scope>provided</scope>
31             </dependency>
32             <dependency>
33                 <groupId>javax.servlet</groupId>
34                 <artifactId>servlet-api</artifactId>
35                 <version>2.5</version>
36                 <scope>provided</scope>
37             </dependency>
38             <dependency>
39                 <groupId>javax.servlet.jsp</groupId>
40                 <artifactId>jsp-api</artifactId>
41                 <version>2.1</version>
42                 <scope>provided</scope>
43             </dependency>
44
45             <dependency>
46                 <groupId>taglibs</groupId>
47                 <artifactId>standard</artifactId>
48                 <version>1.1.2</version>
49             </dependency>
50             <dependency>
51                 <groupId>javax.servlet</groupId>
52                 <artifactId>jstl</artifactId>
53                 <version>1.2</version>
54             </dependency>
55         <!-- Spring test dependencies -->
56             <dependency>
57                 <groupId>org.mockito</groupId>
58                 <artifactId>mockito-all</artifactId>
59                 <version>1.10.19</version>
60                 <scope>test</scope>
61             </dependency>
62
63             <!-- <dependency>
64                 <groupId>org.seleniumhq.selenium</groupId>
65                 <artifactId>selenium-java</artifactId>
66                 <version>2.53.0</version>
67             </dependency>-->
68
69             <dependency>
70                 <groupId>org.seleniumhq.selenium</groupId>
71                 <artifactId>htmlunit-driver</artifactId>
72                 <version>2.26</version>
73             </dependency>
74
75         <!-- https://mvnrepository.com/artifact/org.w3c.css/sac -->

```

```

76 <dependency>
77   <groupId>org.w3c.css</groupId>
78   <artifactId>sac</artifactId>
79   <version>1.3</version>
80 </dependency>
81
82
83 <dependency>
84   <groupId>org.springframework</groupId>
85   <artifactId>spring-test</artifactId>
86   <version>5.1.7.RELEASE</version> <!-- 4.0.5 -->
87   <scope>test</scope>
88 </dependency>
89 <!-- End of spring test dependencies -->
90
91
92
93   <dependency>
94     <groupId>org.springframework.boot</groupId>
95     <artifactId>spring-boot-starter-test</artifactId>
96     <scope>test</scope>
97     <exclusions>
98       <exclusion>
99         <groupId>org.junit.vintage</groupId>
100        <artifactId>junit-vintage-engine</artifactId>
101      </exclusion>
102    </exclusions>
103  </dependency>
104 </dependencies>
105
106 <build>
107   <plugins>
108     <plugin>
109       <groupId>org.springframework.boot</groupId>
110       <artifactId>spring-boot-maven-plugin</artifactId>
111     </plugin>
112   </plugins>
113 </build>
114
115 </project>
116
117

```

## ConstructionCostEstimator/src/main/java/com/controller/ConstructionController.java

```

1  package com.controller;
2
3  import java.util.HashMap;
4  import java.util.Map;
5
6  import javax.validation.Valid;
7
8  import org.springframework.beans.factory.annotation.Autowired;
9  import org.springframework.stereotype.Controller;
10 import org.springframework.ui.ModelMap;
11 import org.springframework.validation.BindingResult;
12 import org.springframework.web.bind.annotation.ModelAttribute;
13 import org.springframework.web.bind.annotation.RequestMapping;
14 import org.springframework.web.bind.annotation.RequestMethod;
15
16 import com.model.Construction;
17 import com.service.ConstructionService;
18
19 @Controller
20 public class ConstructionController {
21

```

```

22     @Autowired
23     private ConstructionService service;
24
25     @RequestMapping(value = "/estimatorPage", method = RequestMethod.GET)
26     public String estimatorPage(@ModelAttribute("construction") Construction construction)
27     {
28         return "estimatorpage";
29     }
30
31     @ModelAttribute("bricksList")
32     public Map<String, String> buildState(){
33
34         Map<String, String> serviceMap = new HashMap<String, String>();
35         serviceMap.put("UnburntClayBricks", "UnburntClayBricks");
36         serviceMap.put("BurntClayBricks", "BurntClayBricks");
37         serviceMap.put("ConcreteBricks", "ConcreteBricks");
38         return serviceMap;
39     }
40
41     @RequestMapping(value = "/result", method = RequestMethod.POST)
42     public String calculateConstructionCost(@Valid @ModelAttribute("construction") Construction
construction,
43         BindingResult result, ModelMap map)
44     {
45         if(result.hasErrors()) {
46             return "estimatorpage";
47         }
48         else
49         {
50             double cost=service.calculateConstructionCost(construction);
51             map.addAttribute("cost",cost);
52             return "result";
53         }
54     }
55 }
56
57
58
59 }
60

```

## ConstructionCostEstimator/src/main/java/com/example/demo/ConstructionCostEstimatorApplication.java

```

1  package com.example.demo;
2
3  import org.springframework.boot.SpringApplication;
4  import org.springframework.boot.autoconfigure.SpringBootApplication;
5  import org.springframework.context.annotation.ComponentScan;
6
7  @SpringBootApplication
8  @ComponentScan({"com.controller","com.model","com.service"})
9  public class ConstructionCostEstimatorApplication {
10
11     public static void main(String[] args) {
12         SpringApplication.run(ConstructionCostEstimatorApplication.class, args);
13     }
14
15 }
16

```

## ConstructionCostEstimator/src/main/java/com/model/Construction.java

```

1  package com.model;
2
3  import javax.validation.constraints.Min;
4
5  import org.springframework.stereotype.Component;
6

```

```

7 @Component
8 public class Construction {
9
10     private String bricksType;
11     @Min(value=500, message="Buildup Area should be minimum 500")
12     private double buildupArea;
13     private double costPerSqFt;
14
15     public String getBricksType() {
16         return bricksType;
17     }
18     public void setBricksType(String bricksType) {
19         this.bricksType = bricksType;
20     }
21     public double getBuildupArea() {
22         return buildupArea;
23     }
24     public void setBuildupArea(double buildupArea) {
25         this.buildupArea = buildupArea;
26     }
27     public double getCostPerSqFt() {
28         return costPerSqFt;
29     }
30     public void setCostPerSqFt(double costPerSqFt) {
31         this.costPerSqFt = costPerSqFt;
32     }
33
34
35 }
36 }
37
38

```

*ConstructionCostEstimator/src/main/java/com/service/ConstructionService.java*

```

1 package com.service;
2
3 import org.springframework.stereotype.Service;
4
5 import com.model.Construction;
6
7 @Service
8 public class ConstructionService {
9
10     public double calculateConstructionCost(Construction obj)
11     {
12         double totalCost=0.0;
13         if(obj.getBricksType().equalsIgnoreCase("UnburntClayBricks"))
14         {
15             obj.setCostPerSqFt(340.00);
16             totalCost=obj.getBuildupArea() * obj.getCostPerSqFt();
17         }
18         else if(obj.getBricksType().equalsIgnoreCase("BurntClayBricks"))
19         {
20             obj.setCostPerSqFt(390.00);
21             totalCost=obj.getBuildupArea() * obj.getCostPerSqFt();
22         }
23         else
24         {
25             obj.setCostPerSqFt(300.00);
26             totalCost=obj.getBuildupArea() * obj.getCostPerSqFt();
27         }
28         return totalCost;
29     }
30
31 }

```



32  
33

### *ConstructionCostEstimator/src/main/resources/application.properties*

```
1 server.port=9067
2 spring.mvc.view.prefix = /WEB-INF/views/
3 spring.mvc.view.suffix = .jsp
4 spring.mvc.static-path-pattern=/resources/**
```

### *ConstructionCostEstimator/src/main/webapp/WEB-INF/views/estimatorpage.jsp*

```
1 <%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>
2 <%@ taglib prefix="sf" uri="http://www.springframework.org/tags/form" %>
3 <%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
4 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
5 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
6   pageEncoding="ISO-8859-1" isELIgnored="false"%>
7
8
9 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
10 <html>
11 <body style="background-color:lavender">
12 <h1><center> Construction Cost Estimator </center></h1>
13 <form:form method="post" action="result" modelAttribute="construction">
14
15 <table style="margin: 0px auto; margin-left: auto; margin-right:auto">
16
17             <tr>
18
19                 <td>Choose BricksType:</td>
20                 <td>
21                     <form:select path="bricksType" id="bricksType">
22                         <form:options items="{bricksList}"/>
23                     </form:select>
24                 </td>
25             </tr>
26
27             <tr>
28                 <td>Enter Buildup Area</td><td><form:input path="buildupArea"
id="buildupArea"/></td><td><form:errors path="buildupArea"/></td>
29             <tr>
30
31                 <td><input type="submit" value="Calculate Cost" id="submit"
/></td>
32                 <td><input type="reset" value="Cancel"/></td>
33             </tr>
34
35         </table>
36 </form:form>
37
38
39 </body>
40 </html>
41
```

### *ConstructionCostEstimator/src/main/webapp/WEB-INF/views/result.jsp*

```
1 <%@page isELIgnored="false" %>
2 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
3 <html>
4 <body bgcolor="lavender">
5 <h2>Approximate construction cost for given parameters is Rs.${cost}</h2>
6 </body>
7 </html>
```

## Grade

Reviewed on Friday, 18 February 2022, 12:31 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: ConstructionController

## 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 - 20.00 / 20.00(Success)

- \* check whether the Construction cost is calculated as per the requirement for ConcreteBricks - 7.50 / 7.50

- \* check whether the Construction cost is calculated as per the requirement for UnburntClayBricks - 5.00 / 5.00

- \* check whether the Construction cost is calculated as per the requirement for BurntClayBricks - 7.50 / 7.50

Spring Testing in the Controller - 30.00 / 30.00(Success)

- \* check whether the request is mapped for estimatorpage and redirected to estimatorpage.jsp - 5.00 / 5.00

- \* check whether the model attribute is specified above the buildState method that holds the Map containing the bricksTypes - 5.00 / 5.00

- \* check whether the model attribute with the name bricksList is specified above the buildState method that holds the Map containing the bricksTypes - 7.50 / 7.50

- \* check whether layering is followed that is whether construction cost 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 estimatorpage to result page is redirected after calculating the construction cost for BurntClayBricks - 5.00 / 5.00

- \* check whether UI from estimatorpage to result page is redirected after calculating the construction cost for ConcreteBricks - 5.00 / 5.00

- \* check whether UI from estimatorpage to result page is redirected after calculating the construction cost for UnburntClayBricks - 10.00 / 10.00

- \* check whether the buildUpArea is rendered in estimatorpage as per the requirement - 5.00 / 5.00

- \* check whether the bricksType is rendered in estimatorpage as per the requirement - 5.00 / 5.00

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