Sakshi Fokane TE-A-42

Assignment No:-10

- Aim :- web application using EJB

- Objective :
1) Understand about basic concept of java beans
2) Understand the basic funcationalit ies of JSP HTML
3) Having the knowledge of JBOSS Server to deploye web application.

- Problem Statement :
Design & develop & deploy web application using EJB.

- Outcomes :-
Student will be able to :
1) Develop a dynamic webpages using Java Beans.
2) To understand the concept & method of web application development process using EJB
3) create a simple EJB 3 Stateless Session bean & local Java application client which will call / invoke the bean for addition of two numbers.

- Software Needed.

1) Ubuntu 2) JDK 3) EJB

- Theory :- Java Beans
J2EE application container container the components that can be used by the clients for executing the business logic. These components are known as Enterprise Java Bean (EJB). EJB 3.0 is being a large Shift from EJB 2.0 & makes development of EJB based application relatives easy.

- Features OF EJB's :
client communication :-
• The client which is often a user interface must be able to call the methods
• Static management : yallu recall Our discussion on this topic
• Transaction management
• Database connection management
• User Authentication & Role base Authorization.
• Asynchronous messaging
• Application Server Administration

- Type of Interprise Java Beans
1) session Beans :

- Session beans are intended to allow the application author to easily implement portion of application code in middle.ware to simplify access to this code.
  - It not Shared.
  - It not prerisitent

2) Stateful session Bean :-
- The client invoke the create method.
- The EJB Container instantiates the bean
- The bean is ready.
- while in the ready state.
- client may invoke a business method.

EJB container may activate a bean, moving it back to that stage & then calls the bean's EJB Activate method.

Enterprise Java Bean (EJB) Architecture :
- The client is working on a web browser.
- there is a database Server that hosts a database like, MySQL / oracle.

- The J2EE server machine is running on an application server.
- The client interface provided web JSP / servlet.

* There are three types :
  1) Dirty read
  2) Non-Repeatable read.
  3) Phantom read.

* Conclusion :
  Hence we have created a simple EJB 3 Stateless Session bean & a local Java application client which will call / invoke the develop for performing addition of two Numbers.

Name :

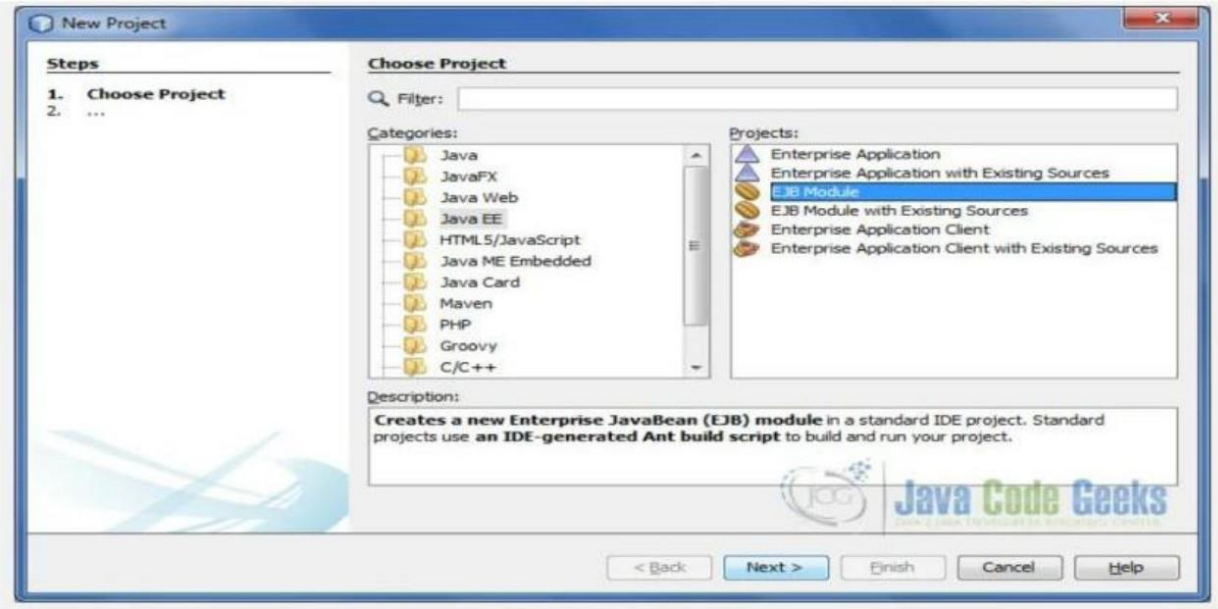
Fokane Sakshi Anil

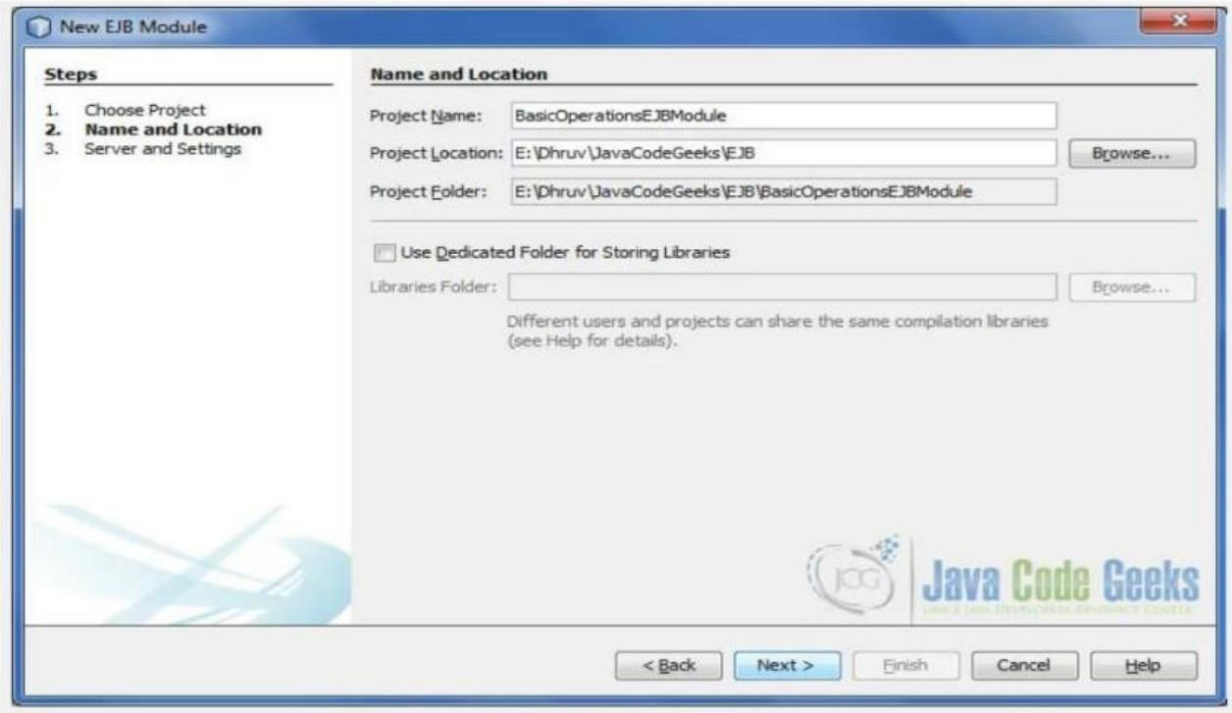
Roll no : 42

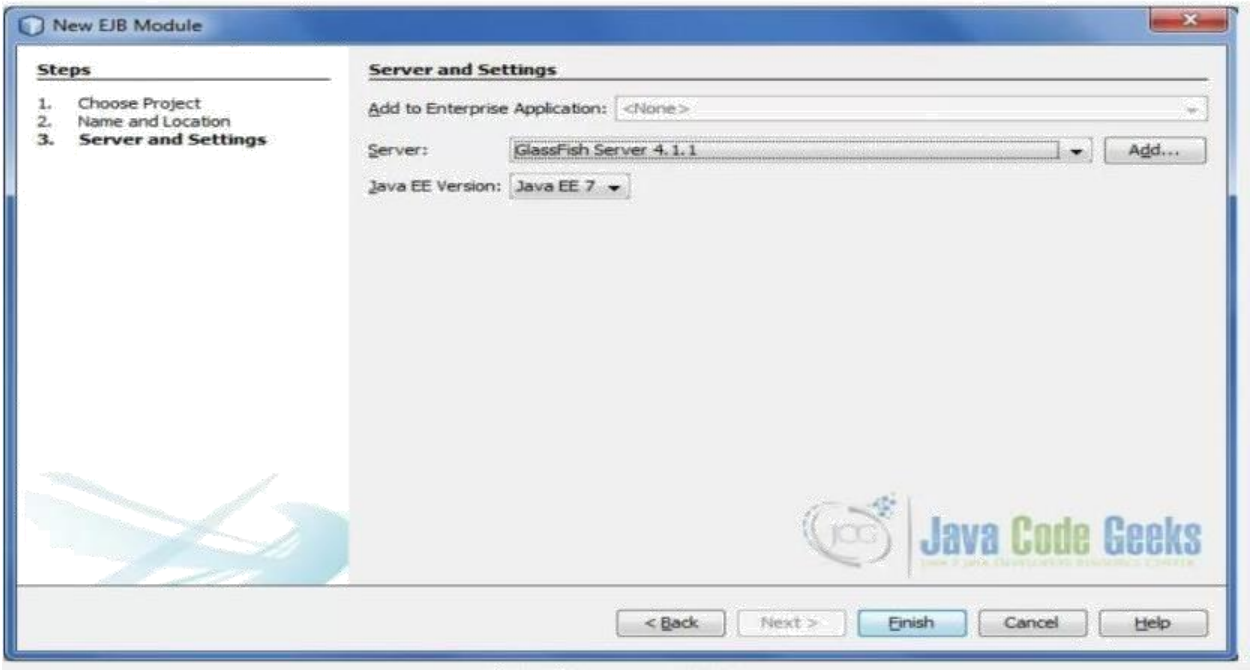
Class : TE-A

Assignment 10 : Design , develop, and deploy web application using EJB.

Create EJB Module

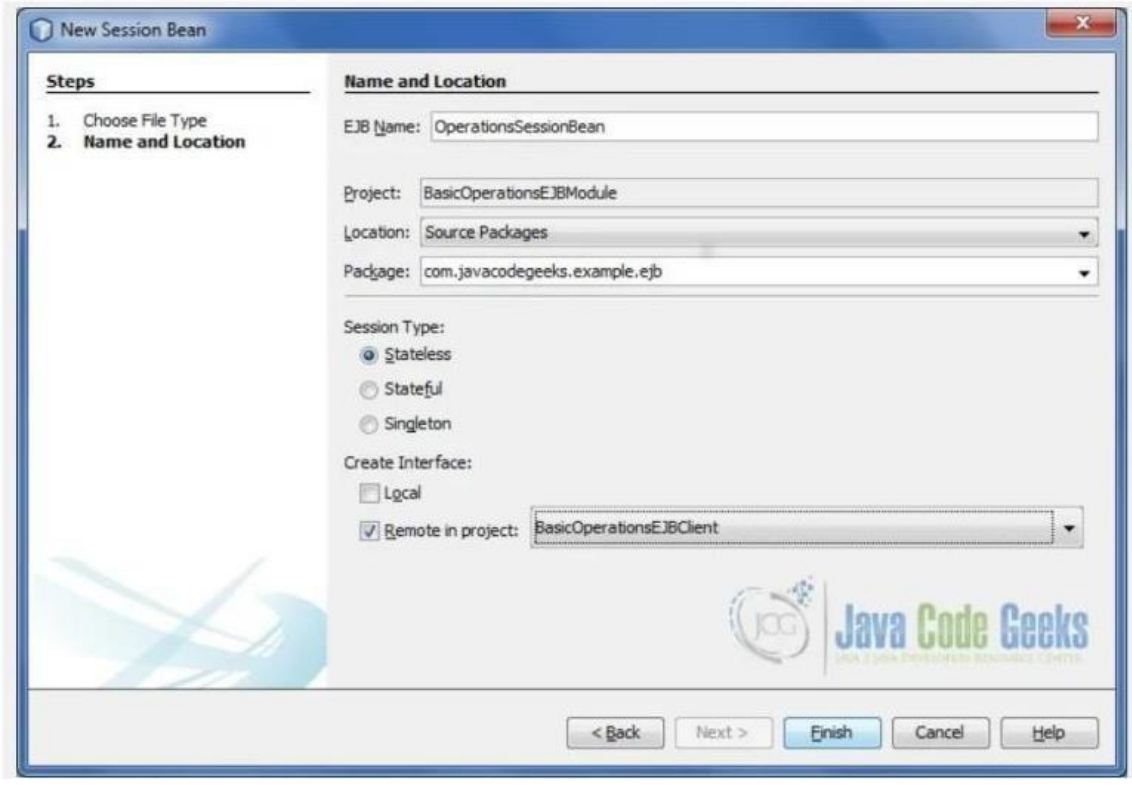




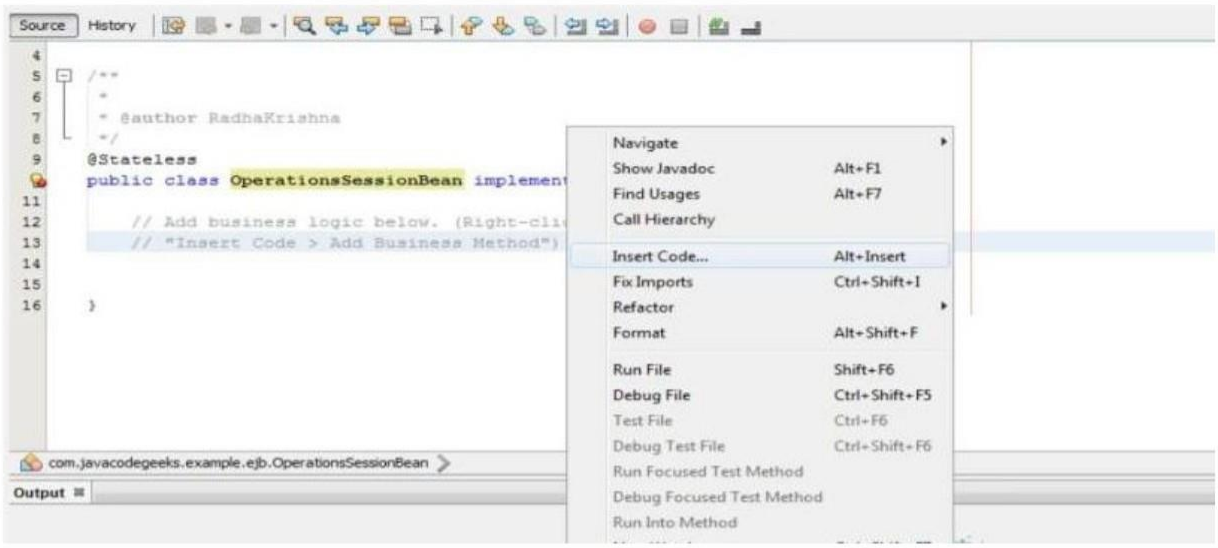


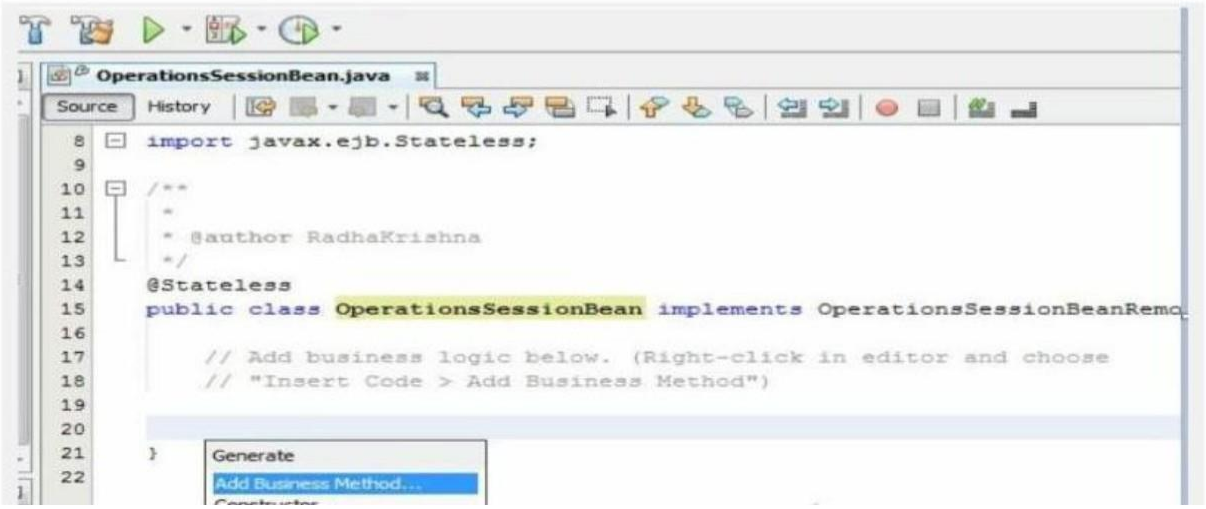
Create a new Application Class Project

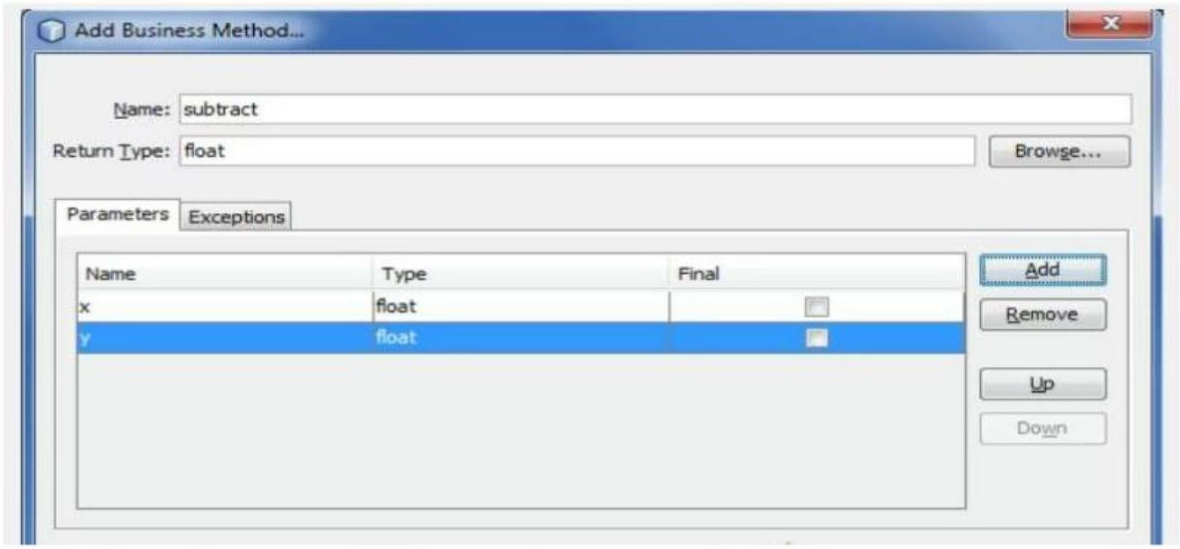
Create Session Bean

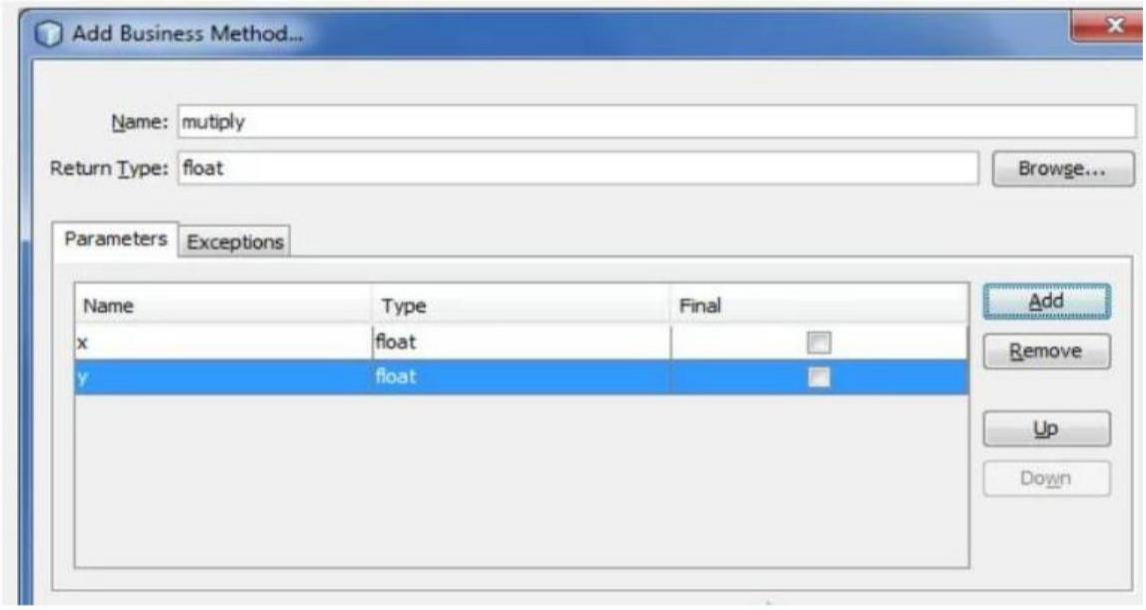


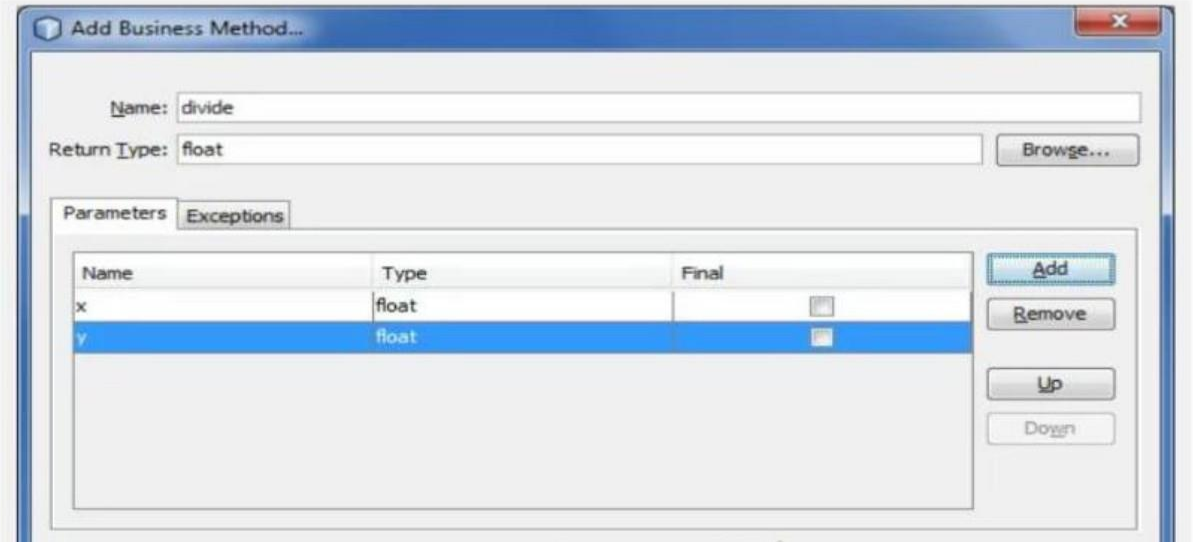
Adding a Business Method






Add Business Method...
Name: subtract
Return Type: float
Browse...
Parameters
Exceptions
Name
Type
Final
x
float
y
float
Add
Remove
Up
Down


Add Business Method...
Name: mutiply
Return Type: float
Browse...
Parameters
Exceptions
Name
Type
Final
x
float
y
float
Add
Remove
Up
Down


Add Business Method...
Name: divide
Return Type: float
Browse...
Parameters
Exceptions
Name
Type
Final
x
float
y
float
Add
Remove
Up
Down

OperationsSessionBean.java

```
package com.javacodegeeks.example.ejb;

import javax.ejb.Stateless;

/**
 *
 * @author
RadhaKrishna */
@Stateless
public class OperationsSessionBean
implements OperationsSessionBeanRemote {

    /  Add business logic below. (Right-click in editor and choose
    /  "Insert Code > Add Business Method")

    @Override
    public float add(float x, float y) {
        return x + y;
    }

    @Override
    public float subtract(float x, float y) {
        return x - y;
    }

   @Override
    public float mutliply(float x, float y) {
        return x * y;
    }

    @Override
    public float divide(float x, float y) {
        return x / y;
    }
}
```

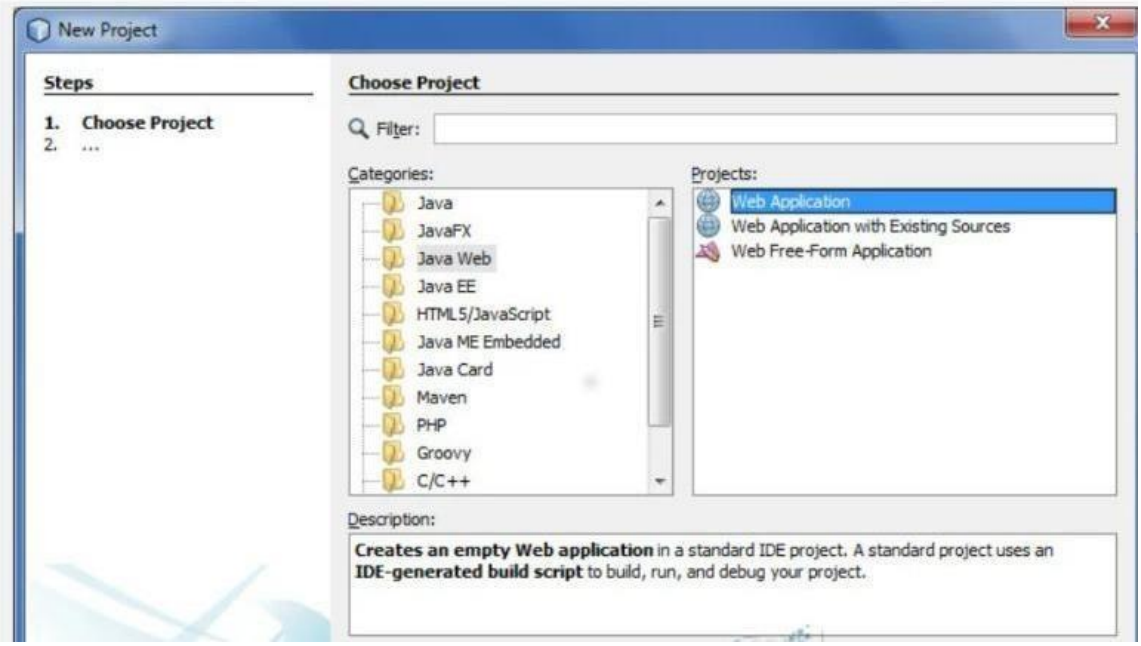
Deploy the EJB Module

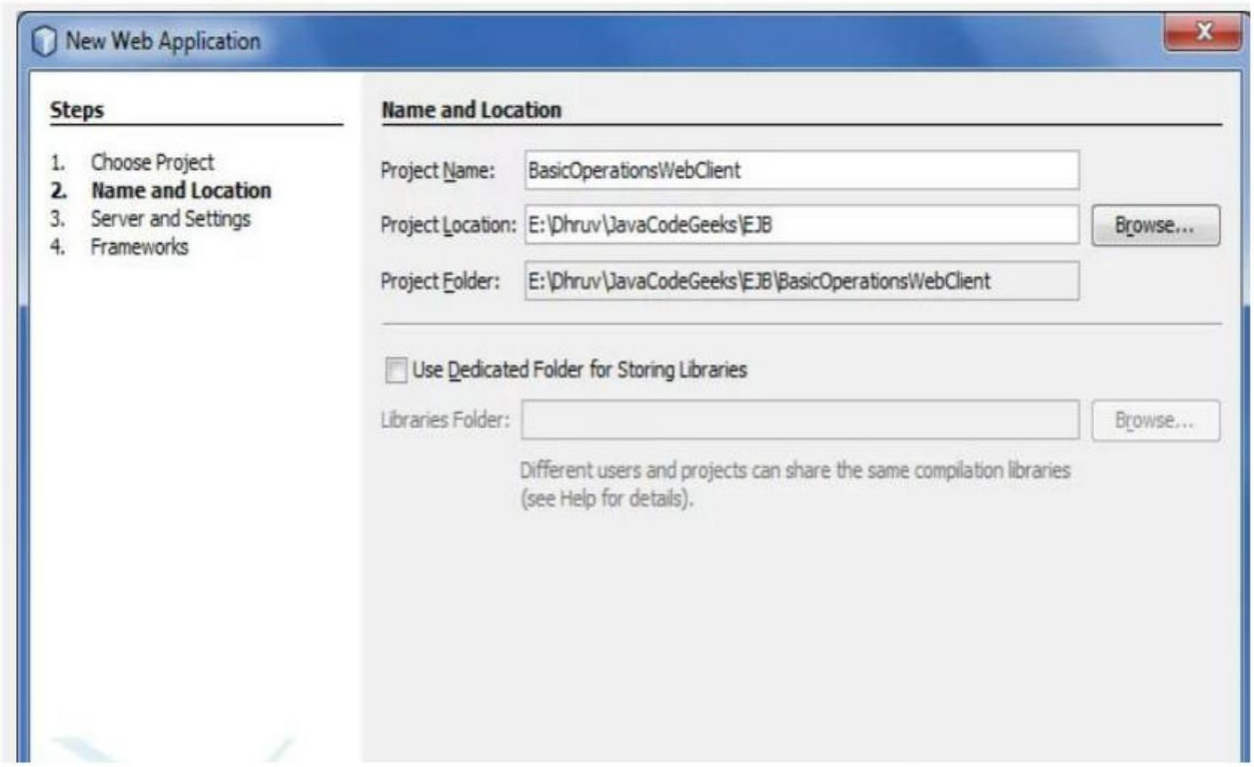
You can now build and deploy the EJB module. Right-click

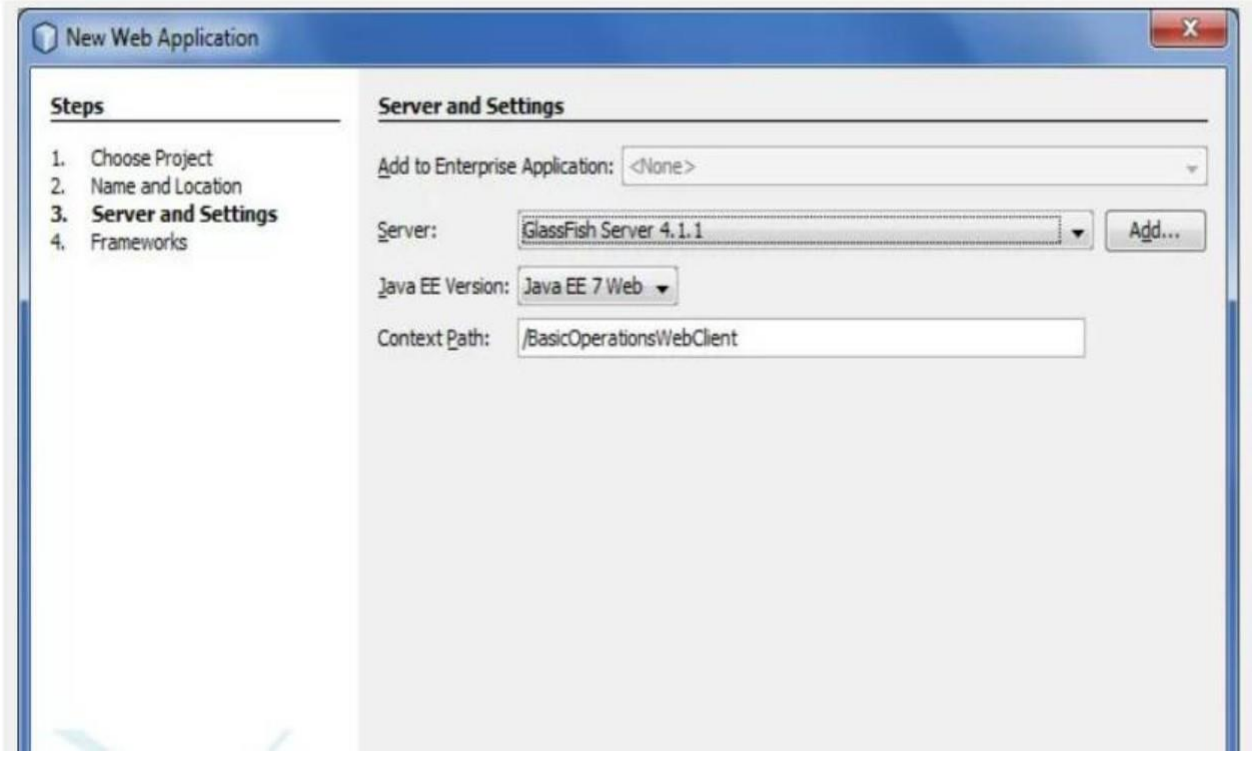
the BasicOperationsEJBModule module and choose Deploy. When you click Deploy, the IDE builds the ejb module and deploys the JAR archive to the server.

In the Services window, if you expand the Applications node of GlassFish Server you can see that BasicOperationsEJBModule was deployed.

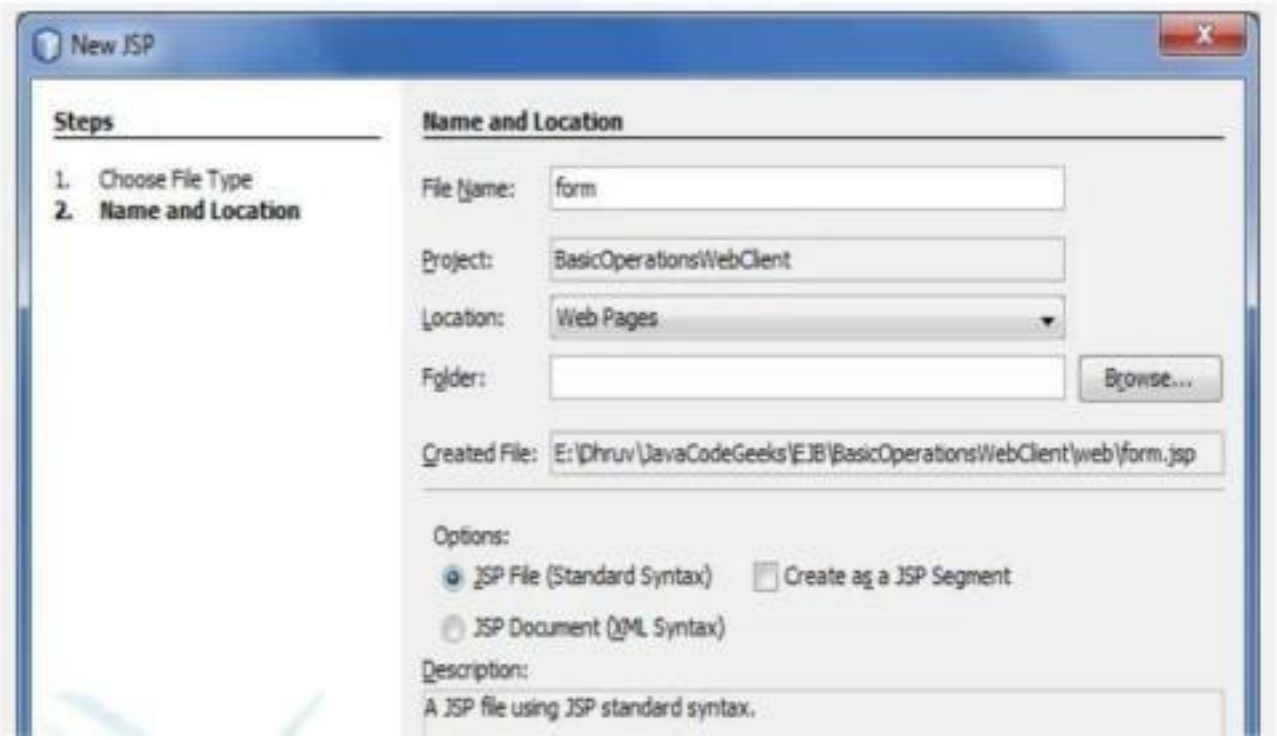
Create a new Web Module to test EJB




New Web Application
Steps
1. Choose Project
2. Name and Location
3. Server and Settings
4. Frameworks
Name and Location
Project Name: BasicOperationsWebClient
Project Location: E:\Dhruv\JavaCodeGeeks\EJB
Browse...
Project Folder: E:\Dhruv\JavaCodeGeeks\EJB\BasicOperationsWebClient
Use Dedicated Folder for Storing Libraries
Libraries Folder:
Browse...
Different users and projects can share the same compilation libraries
(see Help for details).


New Web Application
Steps
1. Choose Project
2. Name and Location
3. Server and Settings
4. Frameworks
Server and Settings
Add to Enterprise Application: <None>
Server: GlassFish Server 4.1.1
Add...
Java EE Version: Java EE 7 Web
Context Path: /BasicOperationsWebClient

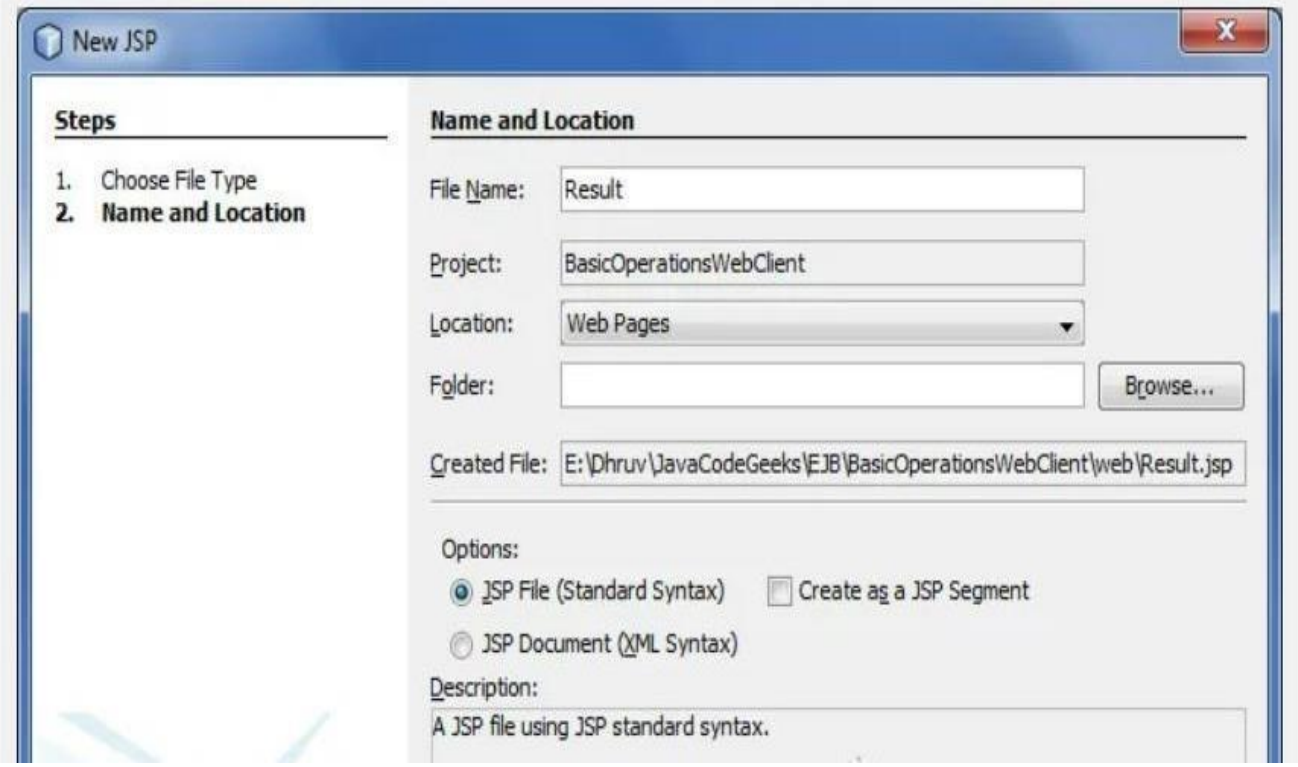
Create JSP Files to test EJB



form.jsp

```
<html>
   <head>
      <title>Calculator</title>
   </head>

   <body bgcolor="lightgreen">
      <h1>Basic Operations</h1>
      <hr>

      <form action="Result.jsp"
         method="POST"> <p>Enter first value:
            <input type="text" name="num1"
         size="25"></p> <br>
         <p>Enter second value:
            <input type="text" name="num2"
         size="25"></p> <br>
```

```
        <b>Select your choice:</b><br>
        <input type="radio" name="group1" value ="add">Addition<br>
        <input type="radio" name="group1" value ="sub">Subtraction<br>
        <input type="radio" name="group1" value
        ="multi">Multiplication<br> <input type="radio" name="group1"
        value ="div">Division<br> <p>
           <input type="submit" value="Submit">
           <input type="reset" value="Reset">
        </p>
      </form>
   </body>
</html>
</form>
```



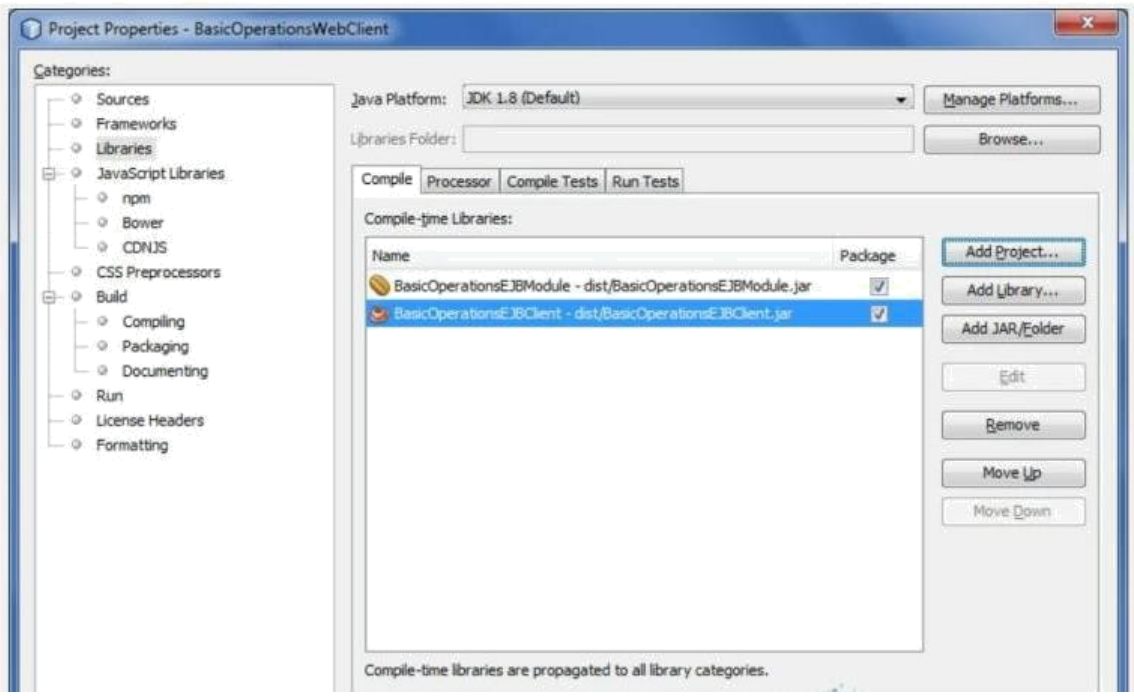
Result.jsp

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ page import="com.javacodegeeks.example.ejb.*, javax.naming.*"%>

<%!
   private OperationsSessionBeanRemote ops = null;
```

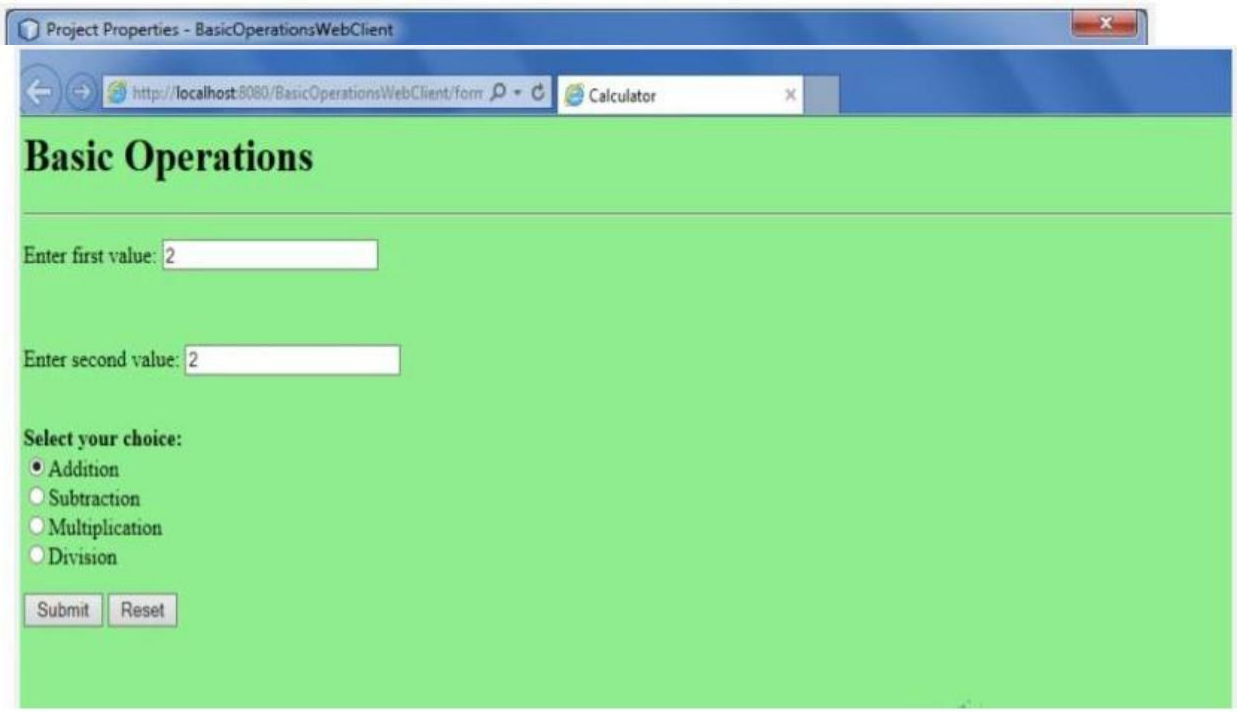
```
   float result = 0;

   public void jspInit() {
      try {

         InitialContext ic = new InitialContext();
         ops =
(OperationsSessionBeanRemote)ic.lookup(OperationsSessionBeanRemote.clas
s.getName());


         System.out.println("Loaded Calculator Bean");


      } catch (Exception ex)
         { System.out.println("Error:
         "
               + ex.getMessage());
      }
   }

   public void jspDestroy() {
      ops = null;
   }
%>


<%

   try {
      String s1 = request.getParameter("num1");
      String s2 = request.getParameter("num2");
      String s3 = request.getParameter("group1");

      System.out.println(s3);

      if (s1 != null && s2 != null) {
         Float num1 = new Float(s1);
         Float num2 = new Float(s2);
```

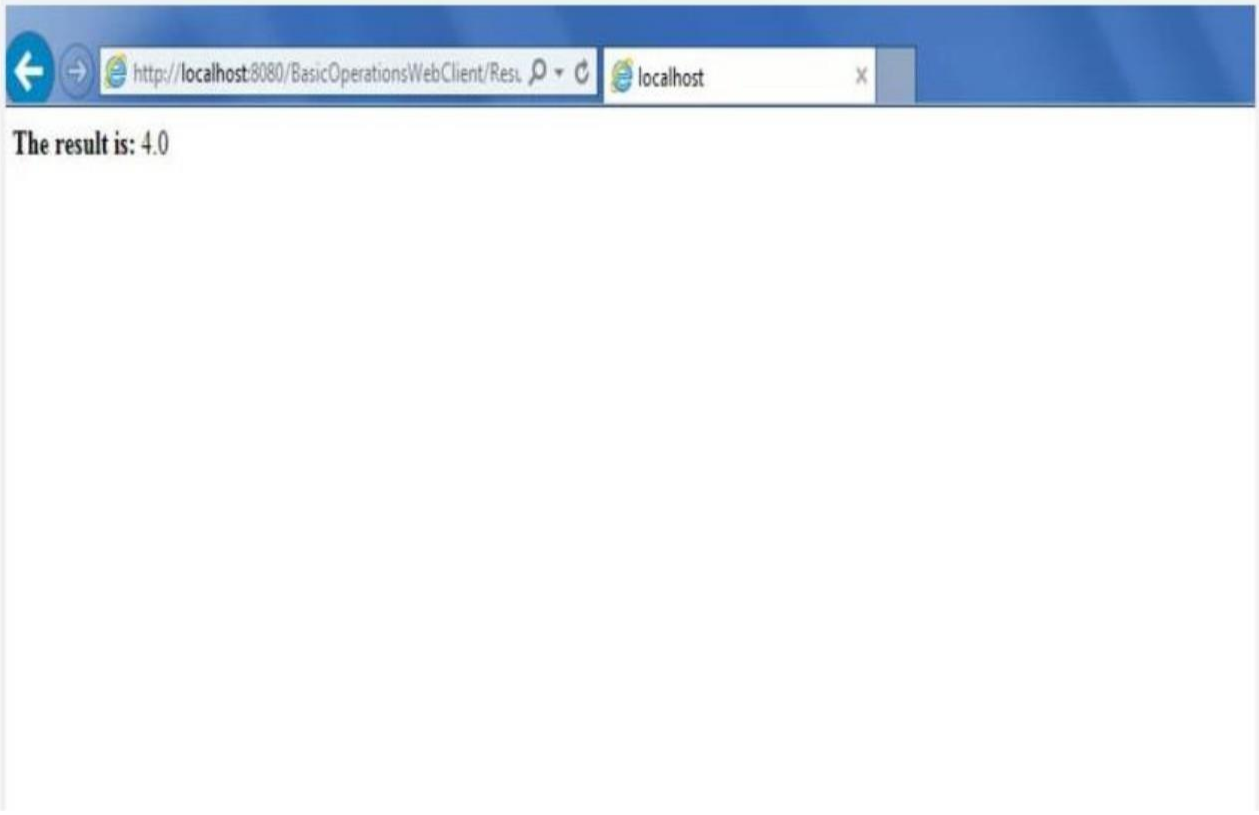
```
if (s3.equals("add")) {
```

```
            result = ops.add(num1.floatValue(),
          num2.floatValue()); } else if (s3.equals("sub")) {
            result = ops.subtract(num1.floatValue(), num2.floatValue());
          } else if (s3.equals("multi")) {
            result = ops.mutliply(num1.floatValue(),
          num2.floatValue()); } else {
            result = ops.divide(num1.floatValue(), num2.floatValue());
          }

%>
<p>
   <b>The result is:</b> <%= result%>
<p>

   <%
          }
       }// end of try
       catch (Exception e) {
          e.printStackTrace();
          //result = "Not valid";
       }
   %>
```



Run the Project


Project Properties - BasicOperationsWebClient
http://localhost:8080/BasicOperationsWebClient/form
Calculator
Basic Operations
Enter first value: 2
Enter second value: 2
Select your choice:
Addition
Subtraction
Multiplication
Division
Submit
Reset


http://localhost:8080/BasicOperationsWebClient/Resu
localhost
The result is: 4.0