Assignment No. 10

AIM:

Design and implement a business interface with necessary business logic for any web application using EJB. e.g., Design and implement the web application logic for deposit and withdraw amounttransactions using EJB.

LEARNING OBJECTIVES:

1. Understand about basic concepts of java beans.
2. Understand the basic functionalities of JSP, HTML.
3. Having the knowledge of JBOSS server to deploy web application.

SOFTWARE REQUIREMENTS:

1. Ubuntu 64 bit / Windows 7.
2. JDK 7 (Java SE 7)
3. EJB 3.0 (stateless session bean)
4. Eclipse luna
5. JBoss Application Server (AS) 7.1.1

THEORY:

Java Beans:

J2EE application container contains the components that can be used by the clients for executing the business logic .These components are known as Enterprise Java Beans (EJB) . J2EE platform has component based architecture to provide multi-tiered, distributed and highly transactional features to enterprise level applications. EJB mainly contains the business logic & business data. EJB component is an EJB class. It is a java class written by EJB developer & this class implements business logic. It is used for developing very much scalable and robust enterprise level applications to be deployed Application Server such as JBOSS, Web Logic etc. EJB 3.0 is being a large shift from EJB 2.0 and makes development of EJB based applications relatively easy.

Features of EJBs:

Some of the features of an application server include the following:

• **Client Communication:** The client, which is often a user interface, must be able to call the methods of objects on the application server via agreed-upon protocols.

• **State Management:** You'll recall our discussions on this topic in the context of JSP (JavaServer Pages) and servlet development back in Chapter 6.

• **Transaction Management:** Some operations, for example, when updating data, must occur as a unit of work. If one update fails, they all should fail.

• **Database Connection Management:** An application server must connect to a database, often using pools of database connections for optimizing resources.

• **User Authentication and Role-Based Authorization:** Users of an application must often log in for security purposes. The functionality of an application to which a user is allowed access is often based on the role associated with a user ID.

• **Asynchronous Messaging:** Applications often need to communicate with other systems in an asynchronous manner; that is, without waiting for the other system to respond. This requires an underlying messaging system that provides guaranteed delivery of these asynchronous messages.

• **Application Server Administration:** Application servers must be administered. For example, they need to be monitored and tuned.

Types of Enterprise Java Beans (EJB):

There are three types of Enterprise Java Beans namely:

1. Session Beans
2. Entity Beans
3. Message driven beans

Session Beans:

• Session beans are intended to allow the application author to easily implement portions of application code in middleware and to simplify access to this code.
• Represents a single client inside the server
• The client calls the session bean to invoke methods of an application on the server
• Perform works for its client, hiding the complexity of interaction with other objects in the server
• Is not shared
• Is not persistent

When the client stops the session, the bean can be assigned to another client from the server Session beans are divided into two types:

1. Stateless Session Bean:

Stateless Session Bean is intended to be simple and "light weight" components. The client, thereby making the server highly scalable, if required, maintains any state. Since no state is maintained in this bean type, stateless session beans are not tied to any specific client, hence any available instance of a stateless session bean can be used to service a client.

• Values only for the duration of the single invocation
• Except during method invocation, all instances of stateless session bean are equivalent

Stateless Session Bean's Life Cycle:

• The client invoke the create method
• The EJB container:

Instantiates the bean

Invokes the setSessionContext Invokes ejbCreate

• The bean is ready
• While in the ready state

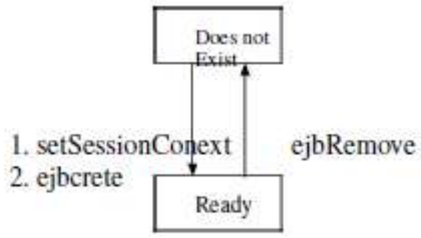


Fig. 1. Stateless Session Bean's Life Cycle

client may invoke a business method :

A client may invoke the remove method and the container calls the bean's ejbRemove method

It's never passivate .

2. Stateful Session Bean:

State ful Session Bean provides easy and transparent state management on the server side. Because state is maintained in this bean type, the application server manages client/bean pairs. Stateful session beans can access persistent resources on behalf of the client, but unlike entity beans, they do not actually represent the data.

Stateful Session Beans Life Cycle:

• The client invoke the create method

• The EJB container :

Instantiates the bean

Invokes the setSessionContext Invokes ejbCreate

• The bean is ready

• While in the ready state

- container may *passivate* the bean moving it from memory to secondary storage
- A client may invoke a business method

EJB container may activate a bean, moving it back to the ready stage, and then calls the bean's ejbActivate method.
A client may invoke the remove method and the container calls the bean's ejbRemove method.

Difference between Stateless and State Full EJB Are as Follows

Stateless:

1. Normally data members are not put in stateless session bean
2. Stateless beans are pooled
3. No effort for keeping client specific data
4. No Activation/Passivation in stateless session bean

Stateful:

1. Data members that represent state are present in stateleful session bean
2. Stateful beans are cached
3. Setting the tag idle-timeout-seconds determines how long data is maintained in stateful session bean
4. Activation – Passivation used

An Entity Bean:

• An entity bean is an object representation of persistent data maintained in a permanent data store such as a database. A primary key identifies each instance of an entity bean. Entity beans are transactional and are recoverable in the event of a system crash.

• Entity beans are representations of explicit data or collections of data, such as a row in a relational database. Entity bean methods provide procedures for acting on the data representation of the bean. An entity bean is persistent and survives if its data remains in the database.

• An entity bean can implement either bean-managed or container-managed persistence. In the case of bean-managed persistence, the implementer of an entity bean stores and retrieves the information managed by the bean through direct database calls. The bean may utilize either Java Database Connectivity (JDBC) or SQL-Java (SQLJ) for this method.

• In the case of container-managed persistence, the container provider may implement access to the database using standard APIs. The container provider can offer tools to map instance variables of an entity bean to calls to an underlying database. The container saves the data. There is no code in the bean for access the database. The container handles all database access required for the bean which create links between beans are created using a structure called abstract schema.

Enterprise Java Beans (EJB) Architecture:

The EJB architecture is an extension of Web architecture. It has an additional tier. The clients of an enterprise bean can be a traditional java application lie, applet, JSP or Servlet. Like in a web application, client browser has to go all the way to web container to use a servlet or JSP, the communication between beans and clients is performed by the EJB container. The following are the flows of the 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 is provided with JSP/Servlet. The enterprise beans reside in the business tier providing to the client tier.
- The Application Server manages the relationships between the client and database machines.

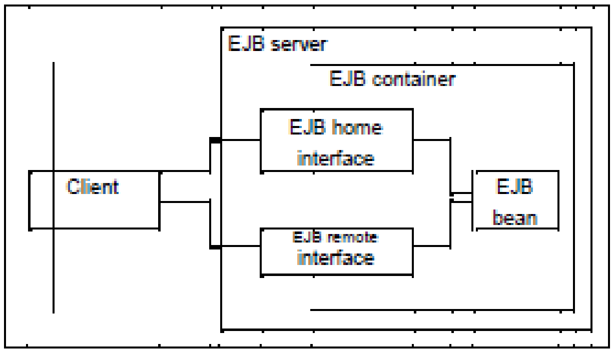


Fig. 2. EJB Architecture

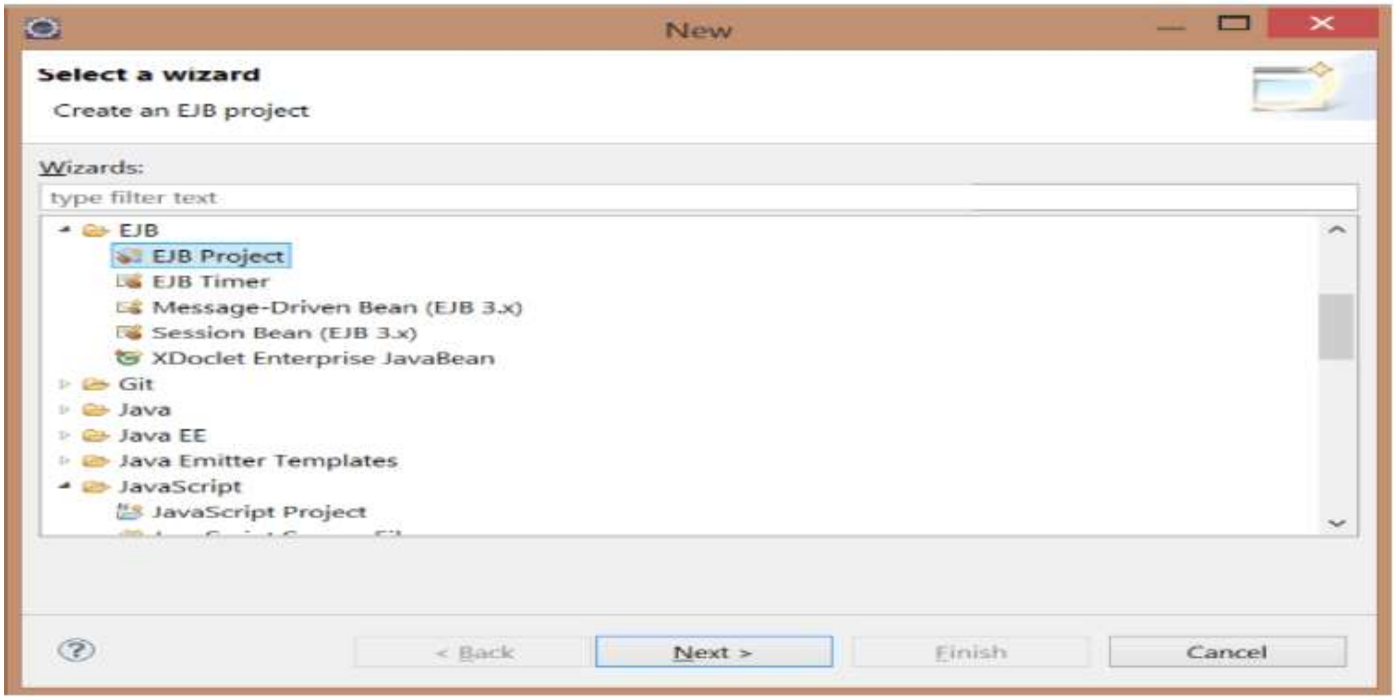
In a diagram, an enterprise bean is a non-visual component of a distributed, transaction oriented enterprise application. Enterprise beans are typically deployed in EJB containers and run on EJB servers.

There are three types through which two or more activities may interfere:
1. Dirty read
2. Non-Repeatable read
3. Phantom read

- **Clustering and Load-Balancing:** Clustering is the process of combining the multiple peripherals, computers and other resources into a single unit.
- A clustered system then works as load balanced system. In a distributed system when a request is send to the server, an algorithm running on the server decides which server has fewer loads and sends the request to that server. EJB container encapsulates these features to provide smooth and efficient service.
- **Deployment Descriptor:** A deployment descriptor is an XML file packaged with the enterprise beans in an EJB JAR file or an EAR file. It contains metadata describing the contents and structure of the enterprise beans, and runtime transaction and security information for the EJB container.
- **EJB Server:** An EJB server is a high-level process or application that provides a runtime environment to support the execution of server applications that use enterprise beans. An EJB server provides a JNDI-accessible naming service. It manages and coordinates the allocation of resources to client applications, provides access to system resources and provides a transaction service.

PROGRAM CODE: INPUT & OUTPUT:
Create a new EJB Project :
Open Eclipse IDE and create a new EJB project which can be done by clicking on, File menu -> New ->

EJB Project
Step 1:
- Create EJB project addition
- Click File –> New –> Other –> EJB –> EJB Project –> Next


New

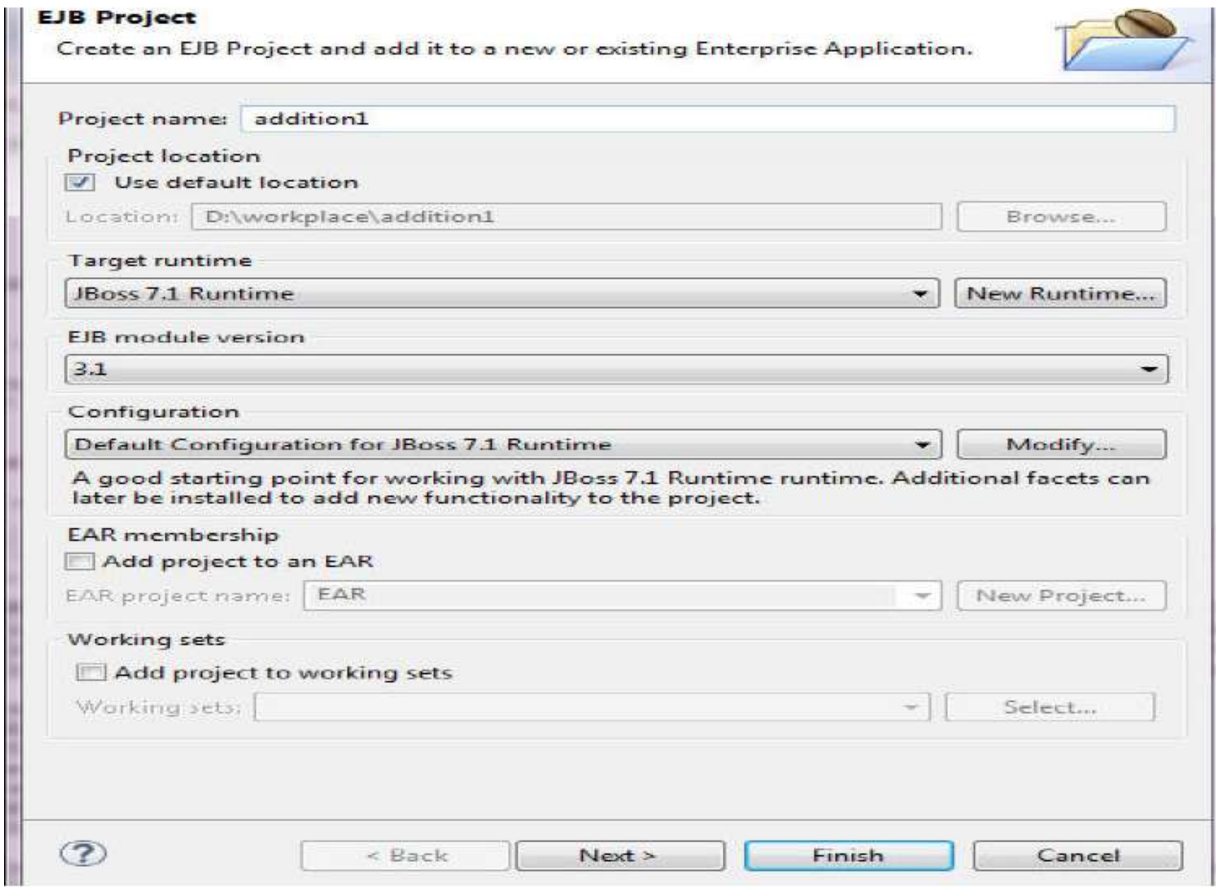
Select a wizard

Create an EJB project

Wizards:

type filter text

- EJB
  - EJB Project
  - EJB Timer
  - Message-Driven Bean (EJB 3.x)
  - Session Bean (EJB 3.x)
  - XDoclet Enterprise JavaBean
- Git
- Java
- Java EE
- Java Emitter Templates
- JavaScript
  - JavaScript Project


EJB Project

Create an EJB Project and add it to a new or existing Enterprise Application.

Project name: addition1

Project location

☑ Use default location

Location: D:\workplace\addition1

Target runtime

JBoss 7.1 Runtime

EJB module version

3.1

Configuration

Default Configuration for JBoss 7.1 Runtime

A good starting point for working with JBoss 7.1 Runtime runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership

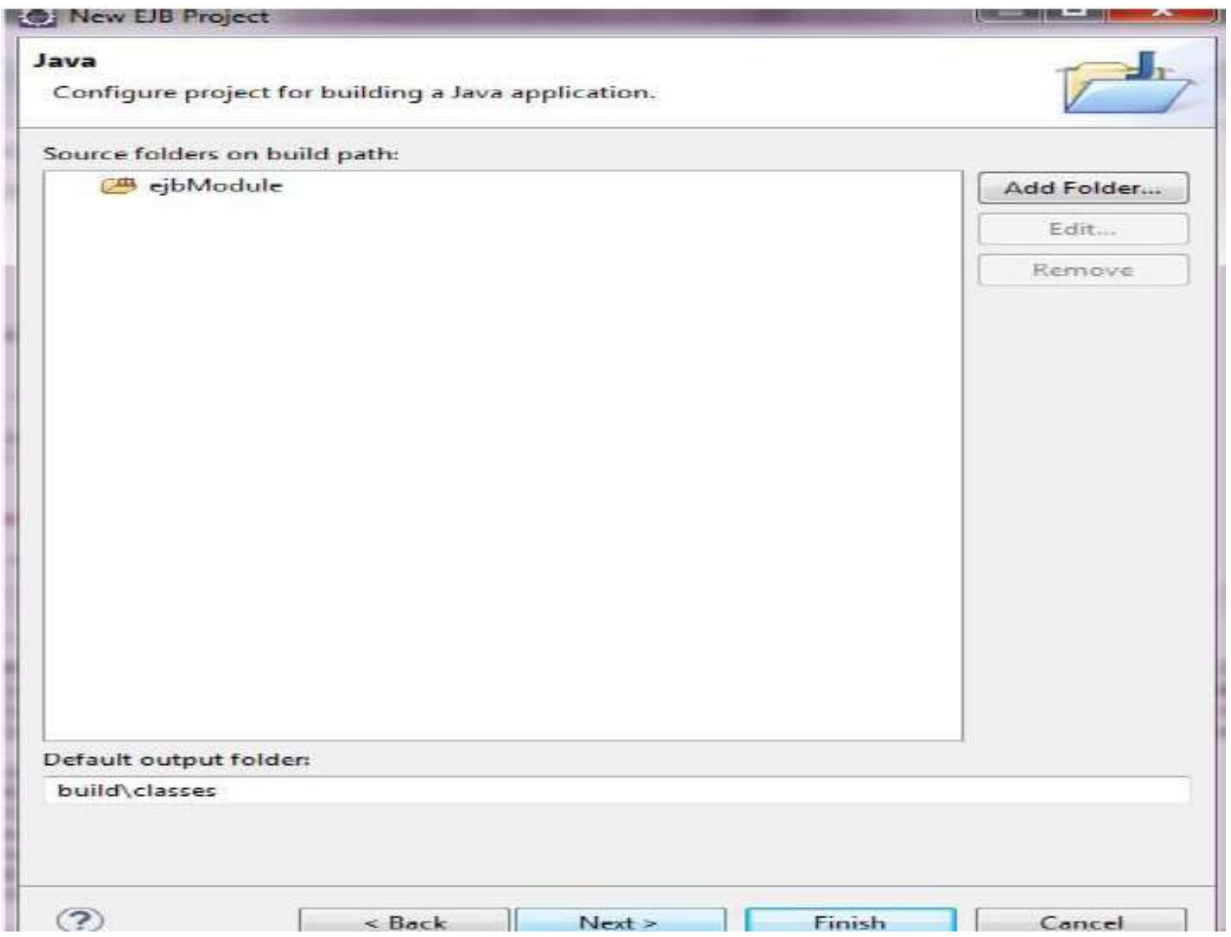
☐ Add project to an EAR

EAR project name: EAR

Working sets

☐ Add project to working sets

Working sets:



New EJB Project

Java

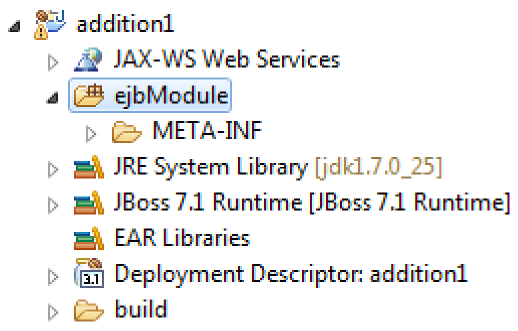
Configure project for building a Java application.

Source folders on build path:

ejbModule

Default output folder:

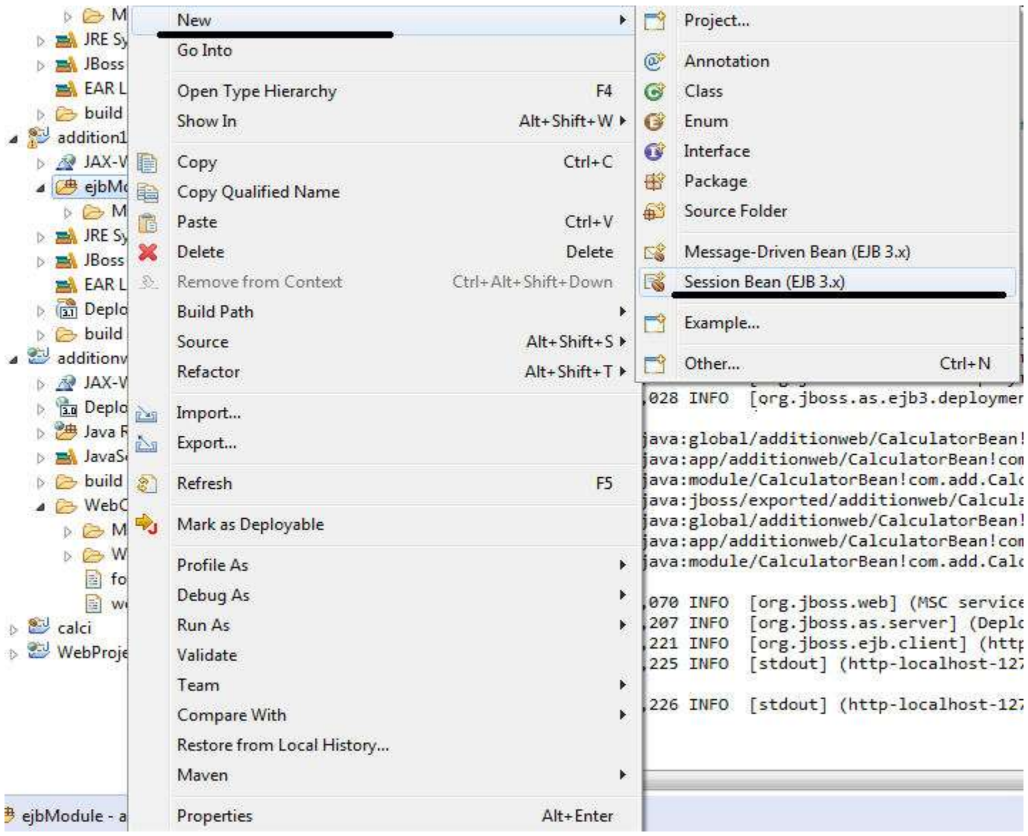
build\classes



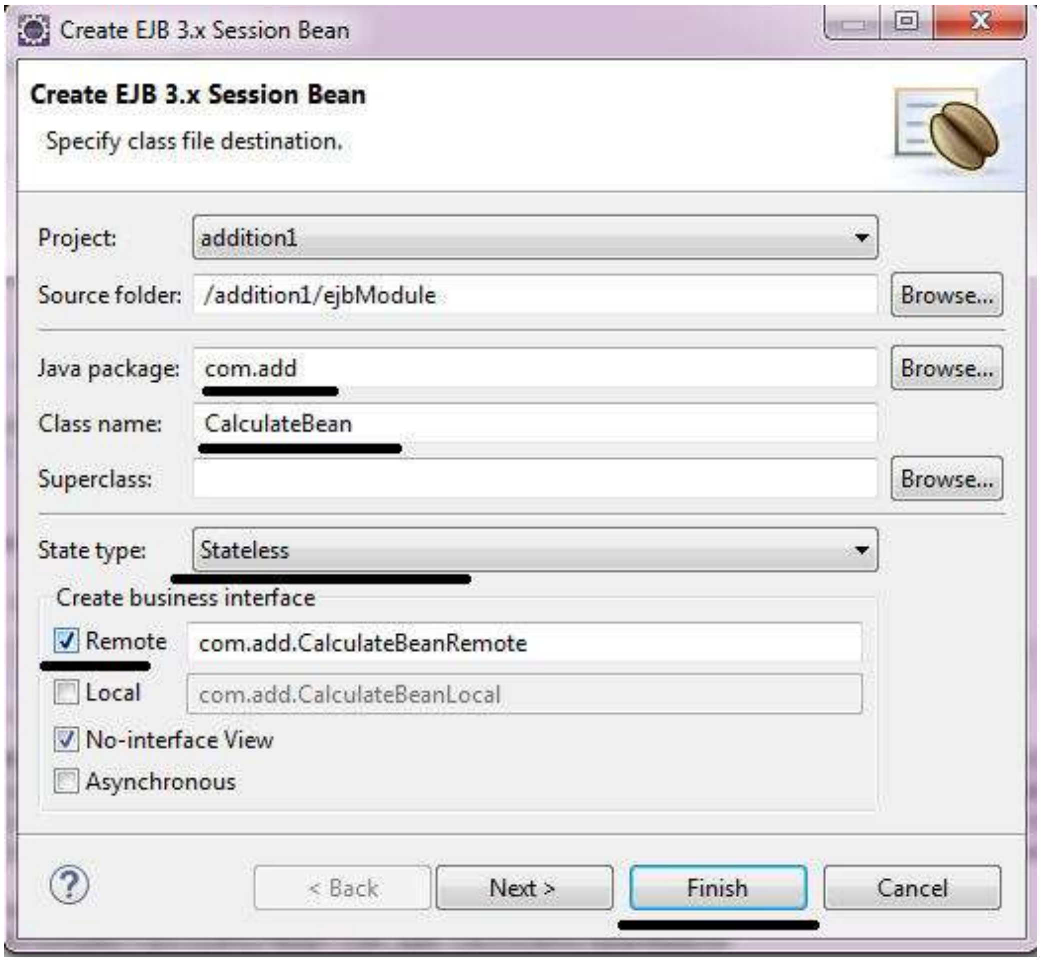
```
addition1
  JAX-WS Web Services
  ejbModule
    META-INF
  JRE System Library [jdk1.7.0_25]
  JBoss 7.1 Runtime [JBoss 7.1 Runtime]
  EAR Libraries
  Deployment Descriptor: addition1
  build
```

Step 2 :

Now create Stateless session bean with its remote interface. Expand project –> expande ejbModule –> Right click Session Bean –> New –> Session



```
,028 INFO  [org.jboss.as.ejb3.deploymer
java:global/additionweb/CalculatorBean!
java:app/additionweb/CalculatorBean!com
java:module/CalculatorBean!com.add.Calc
java:jboss/exported/additionweb/Calcula
java:global/additionweb/CalculatorBean!
java:app/additionweb/CalculatorBean!com
java:module/CalculatorBean!com.add.Calc

,070 INFO  [org.jboss.web] (MSC service
,207 INFO  [org.jboss.as.server] (Deplo
,221 INFO  [org.jboss.ejb.client] (http
,225 INFO  [stdout] (http-localhost-127

,226 INFO  [stdout] (http-localhost-127
```



Create EJB 3.x Session Bean

Specify class file destination.

Project: addition1

Source folder: /addition1/ejbModule

Java package: com.add

Class name: CalculateBean

Superclass:

State type: Stateless

Create business interface

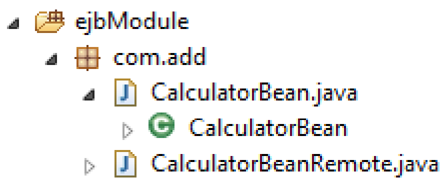
[✓] Remote com.add.CalculateBeanRemote

[ ] Local com.add.CalculateBeanLocal

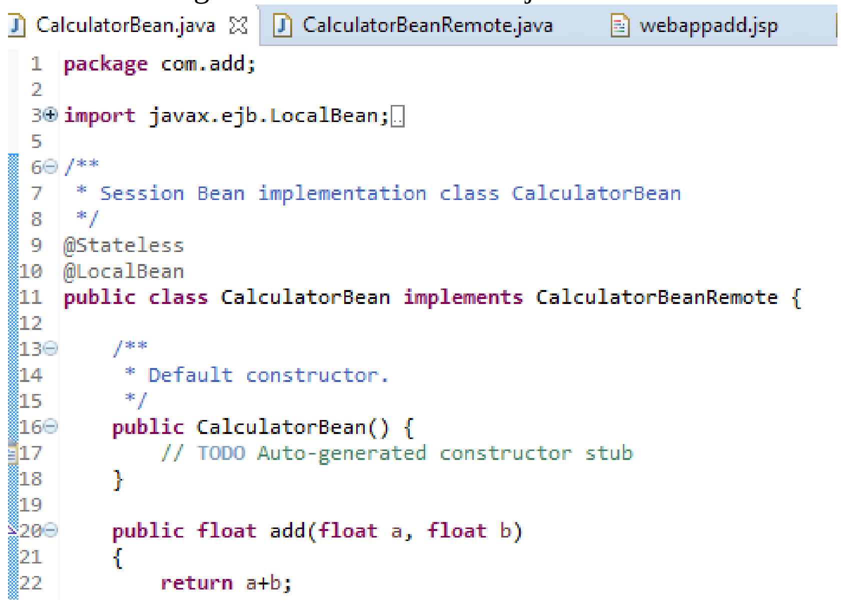
[✓] No-interface View

[ ] Asynchronous

In ejbModule 2 java files are going to create after Finish button.



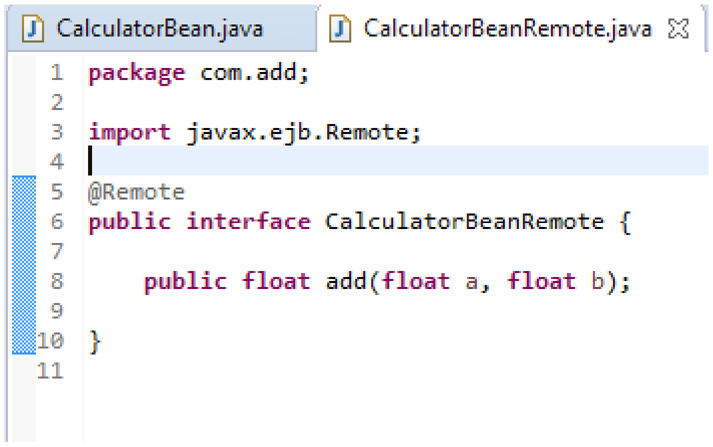
Write following code in CalculatorBean.java



```
package com.add;

import javax.ejb.LocalBean;

/**
 * Session Bean implementation class CalculatorBean
 */
@Stateless
@LocalBean
public class CalculatorBean implements CalculatorBeanRemote {

    /**
     * Default constructor.
     */
    public CalculatorBean() {
        // TODO Auto-generated constructor stub
    }

    public float add(float a, float b)
    {
        return a+b;
```

```
        }
  }
```

Write Following code in CalculatorBeanRemote.java



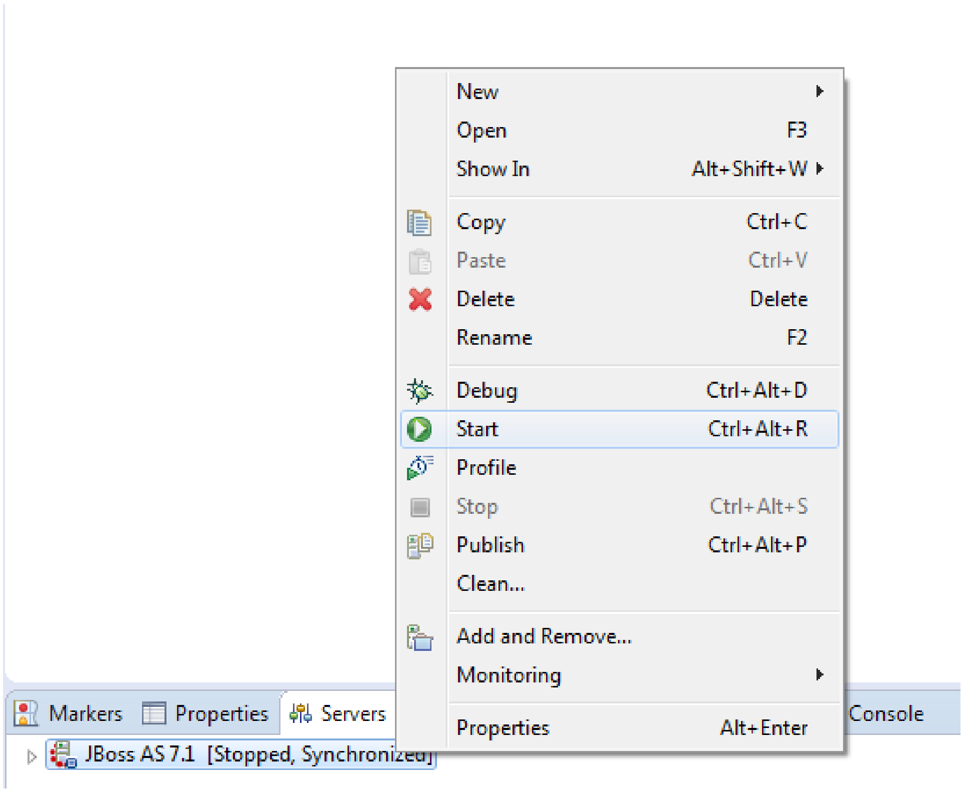
```
package com.add;

import javax.ejb.Remote;

@Remote
public interface CalculatorBeanRemote {

    public float add(float a, float b);

}
```

Step 3:
Deploying the project :

Now we need to deploy the EJB "addition" on server. Follow the steps mentioned bellow to deploy this project on server.

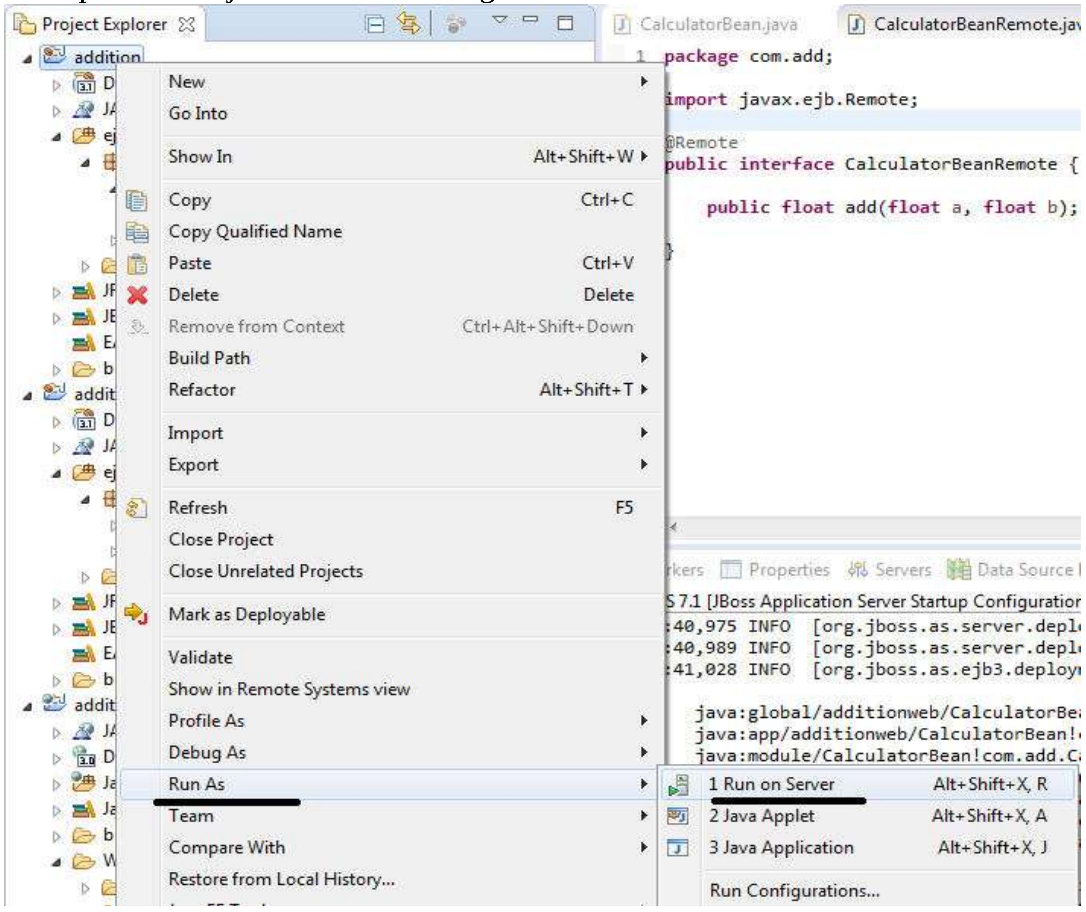
Start the server

Right click on "JBoss 7.1 Runtime Server" from Servers view and click on Start.



Step 4:

Now next step Go to Project-> addition -> right click -> run-> Run on server

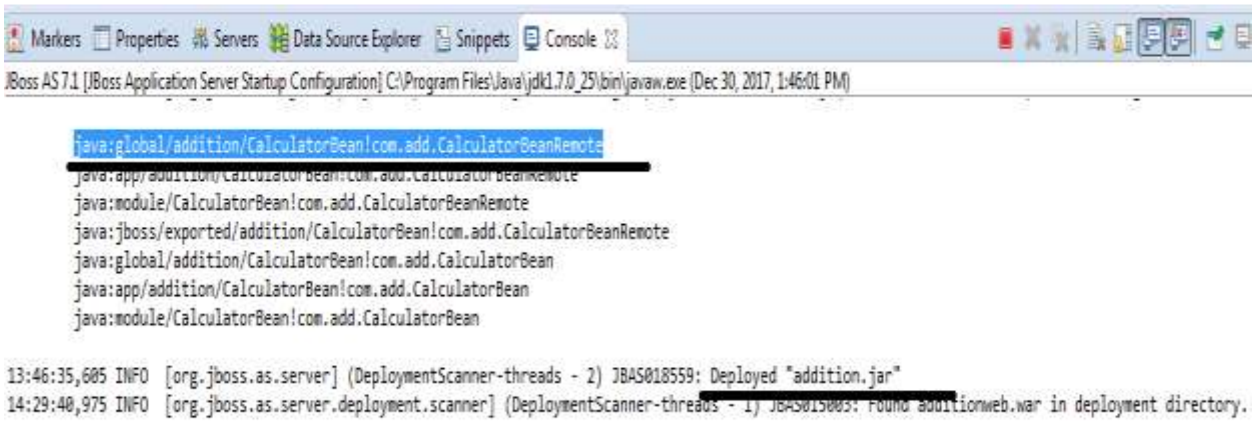


```
package com.add;

import javax.ejb.Remote;

@Remote
public interface CalculatorBeanRemote {

    public float add(float a, float b);
}
```

```
S 7.1 [JBoss Application Server Startup Configuratior
:40,975 INFO  [org.jboss.as.server.depl
:40,989 INFO  [org.jboss.as.server.depl
:41,028 INFO  [org.jboss.as.ejb3.deploy

    java:global/additionweb/CalculatorBe
    java:app/additionweb/CalculatorBean!
    java:module/CalculatorBean!com.add.C
```

Step 5:

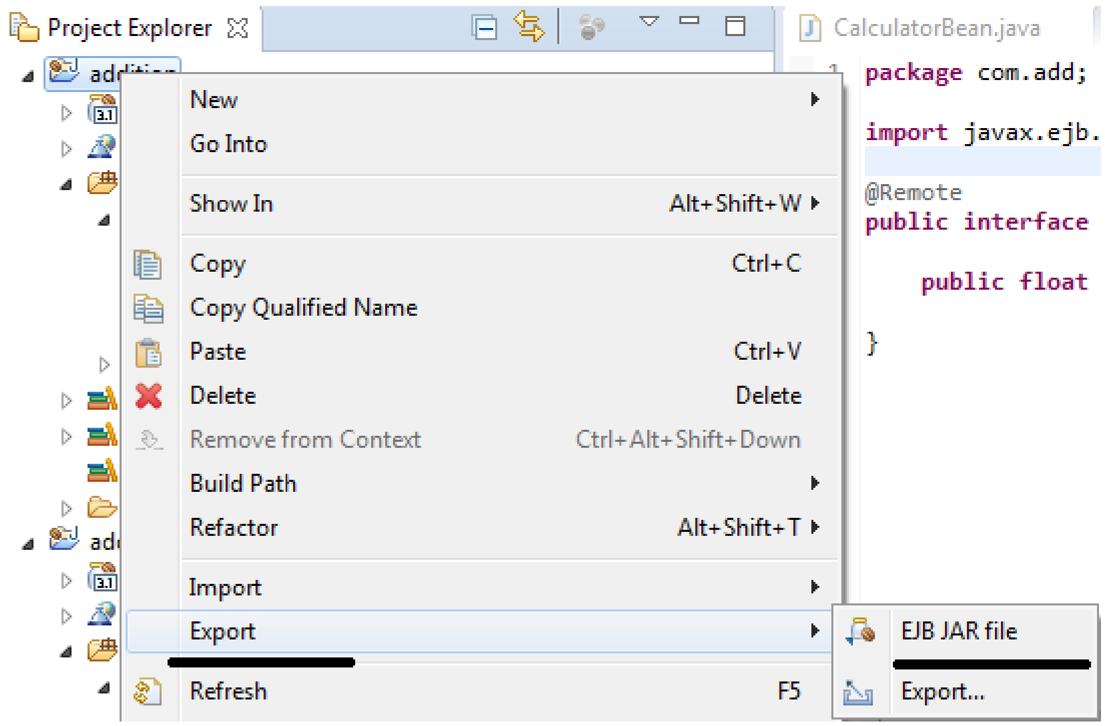
After running the program you can see following message on console



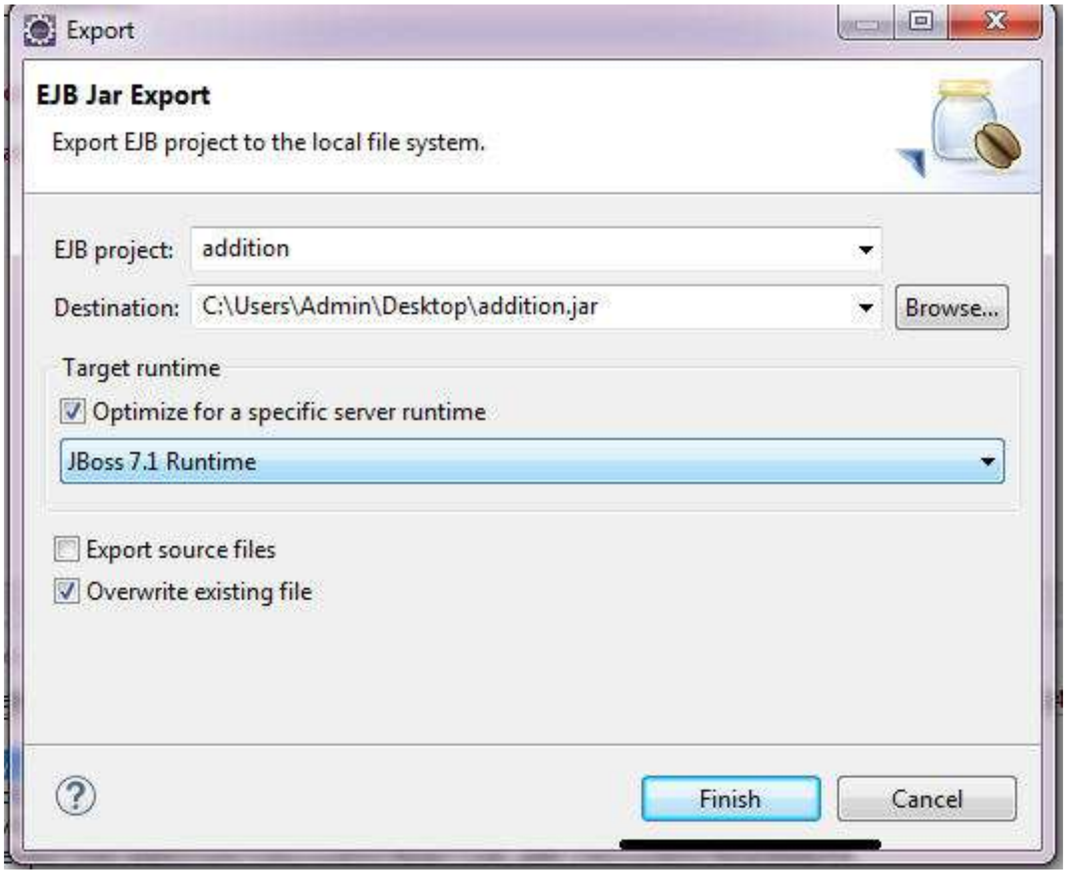
```
JBoss AS 7.1 [JBoss Application Server Startup Configuration] C:\Program Files\Java\jdk1.7.0_25\bin\javaw.exe (Dec 30, 2017, 1:46:01 PM)

        java:global/addition/CalculatorBean!com.add.CalculatorBeanRemote
        java:app/addition/CalculatorBean!com.add.CalculatorBeanRemote
        java:module/CalculatorBean!com.add.CalculatorBeanRemote
        java:jboss/exported/addition/CalculatorBean!com.add.CalculatorBeanRemote
        java:global/addition/CalculatorBean!com.add.CalculatorBean
        java:app/addition/CalculatorBean!com.add.CalculatorBean
        java:module/CalculatorBean!com.add.CalculatorBean

13:46:35,605 INFO  [org.jboss.as.server] (DeploymentScanner-threads - 2) JBAS018559: Deployed "addition.jar"
14:29:40,975 INFO  [org.jboss.as.server.deployment.scanner] (DeploymentScanner-threads - 1) JBAS015003: Found additionweb.war in deployment directory.
```

Step 6:

Once this jar file is deployed to server now export EJB jar file save it in desktop -> Finish.



```
package com.add;

import javax.ejb.

@Remote
public interface

    public float

}
```


Export

EJB Jar Export

Export EJB project to the local file system.

EJB project: addition

Destination: C:\Users\Admin\Desktop\addition.jar

Target runtime

[✓] Optimize for a specific server runtime

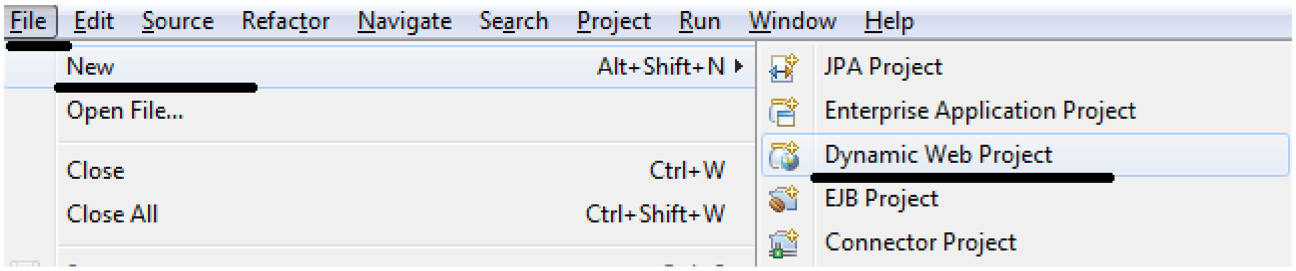
JBoss 7.1 Runtime

[ ] Export source files

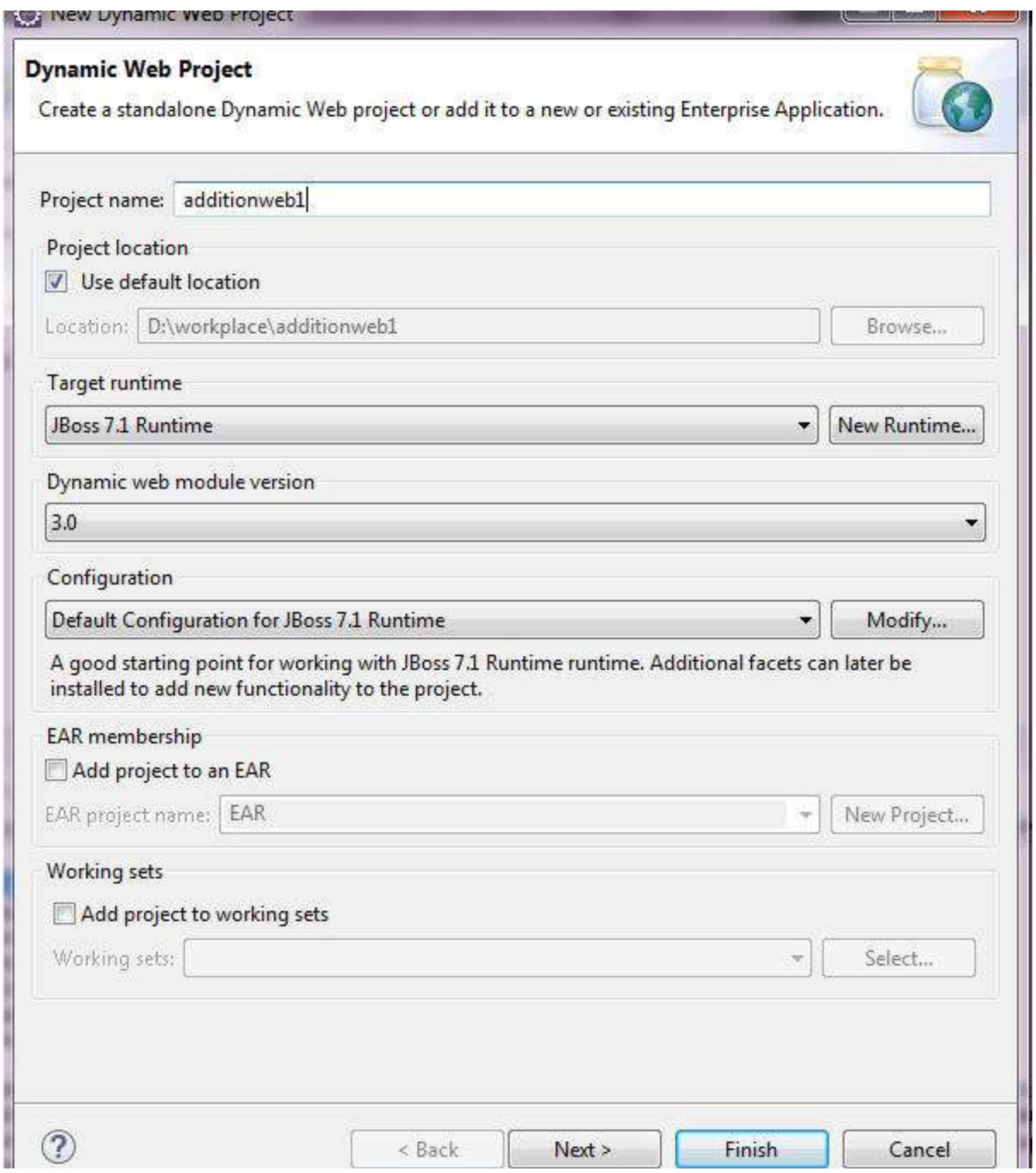
[✓] Overwrite existing file

Step 7:

Now create another project



Write project name-> as addition web -> finish



Dynamic Web Project

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name: additionweb1

Project location

[✓] Use default location

Location: D:\workplace\additionweb1

Target runtime: JBoss 7.1 Runtime

Dynamic web module version: 3.0

Configuration: Default Configuration for JBoss 7.1 Runtime

A good starting point for working with JBoss 7.1 Runtime runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership

[ ] Add project to an EAR

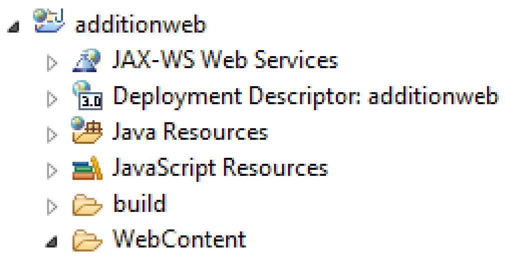
EAR project name: EAR

Working sets

[ ] Add project to working sets

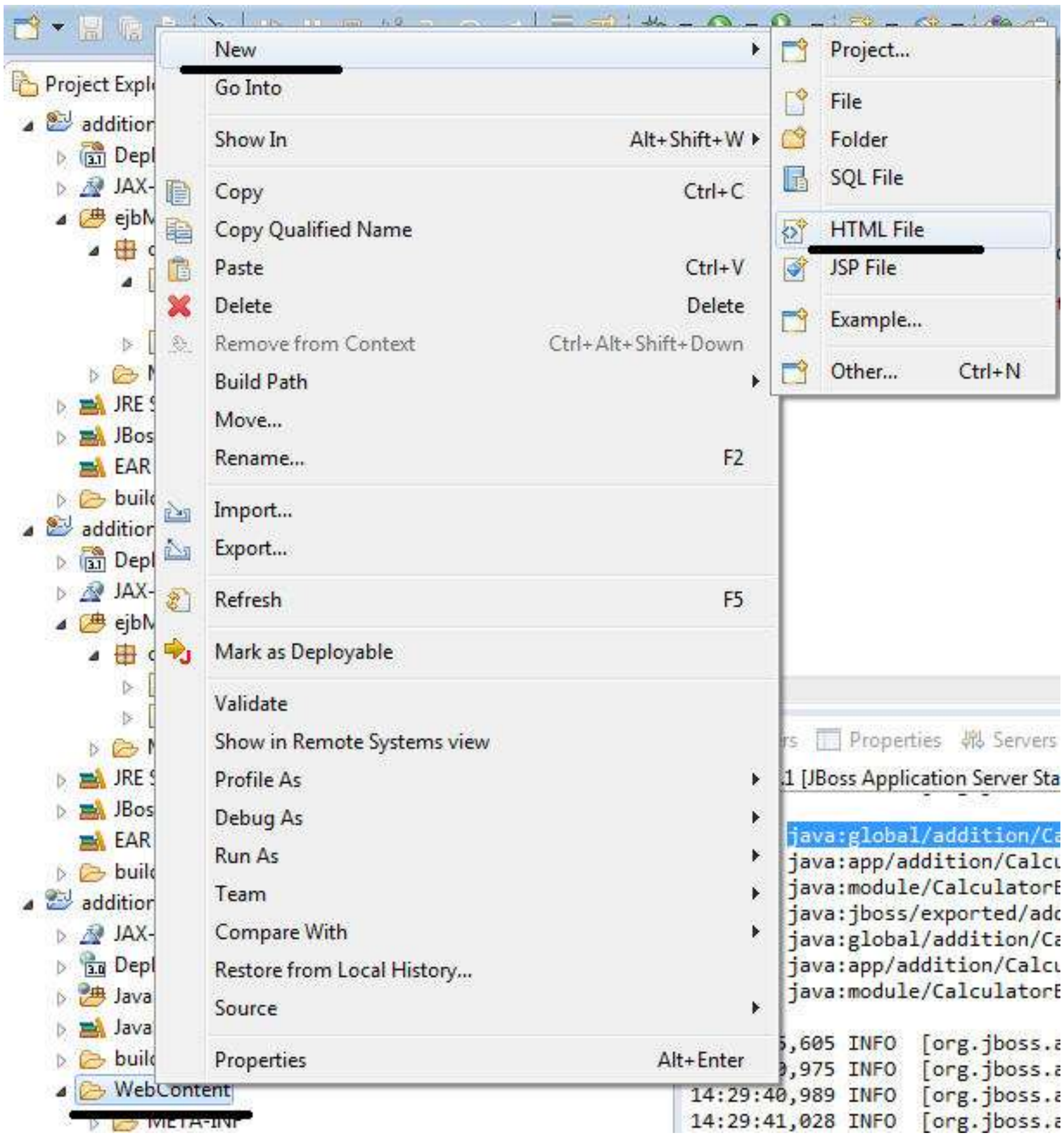
Working sets:

Get a file structure as follow



```
◢ additionweb
    ▷ JAX-WS Web Services
    ▷ Deployment Descriptor: additionweb
    ▷ Java Resources
    ▷ JavaScript Resources
    ▷ build
    ◢ WebContent
```

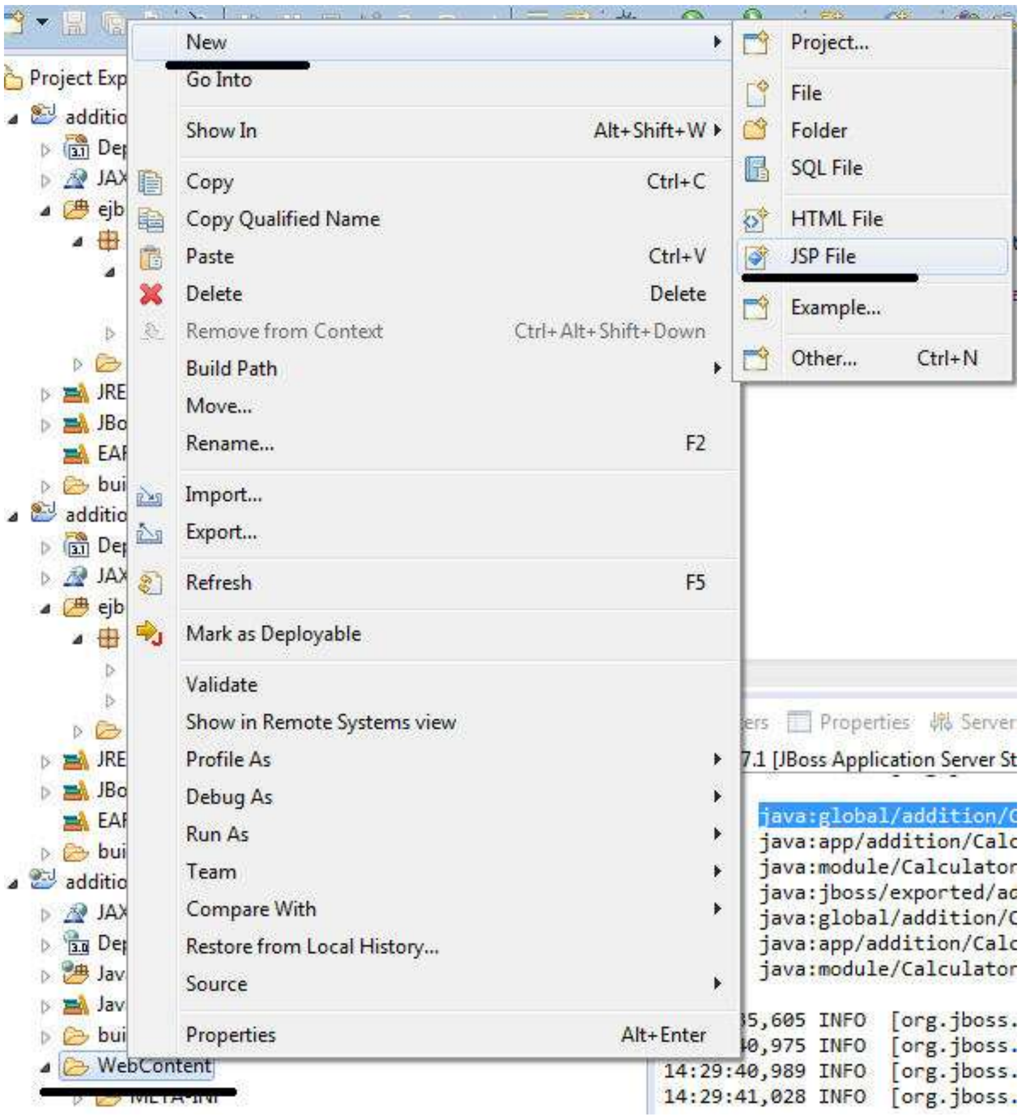
Step 8:
Web Content -> right click->new -> html page



Write file name -> form.html->Finish

Step 9:

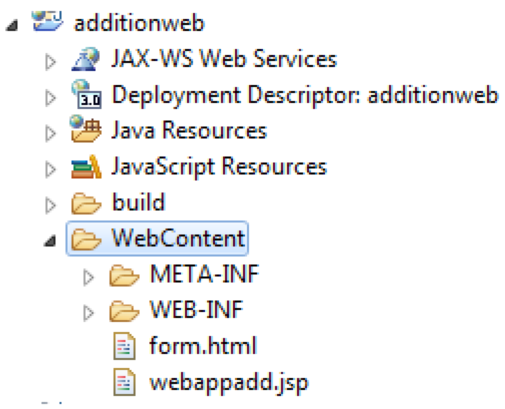
Web Content -> right click->new -> jsp page



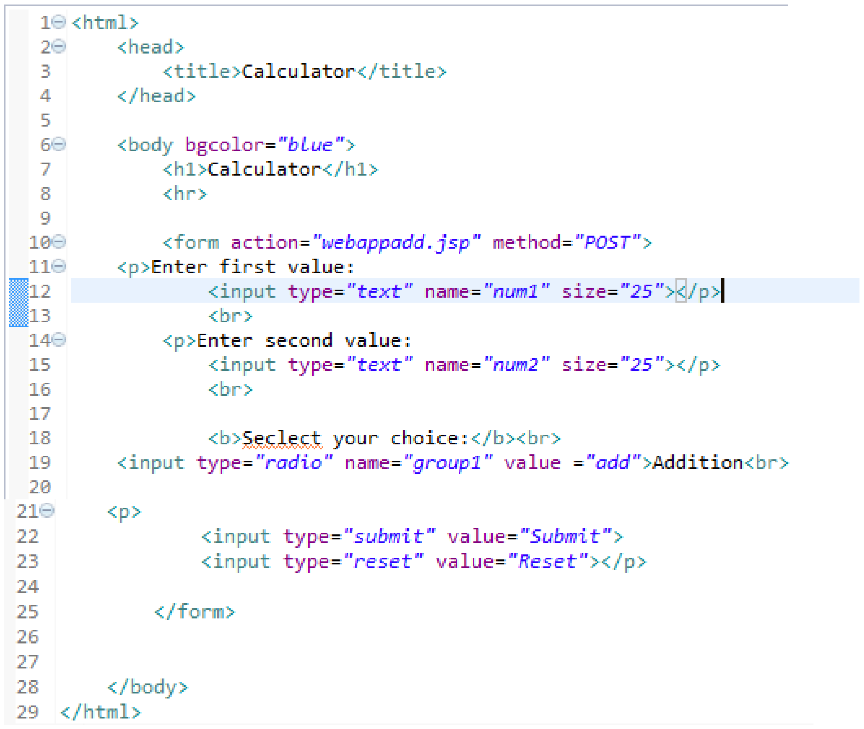
```
java:global/addition/C
java:app/addition/Calc
java:module/Calculator
java:jboss/exported/ad
java:global/addition/C
java:app/addition/Calc
java:module/Calculator

5,605 INFO  [org.jboss.
0,975 INFO  [org.jboss.
14:29:40,989 INFO  [org.jboss.
14:29:41,028 INFO  [org.jboss.
```

Write file name -> webappadd.jsp->Finish

Get the file structure in project window as follows

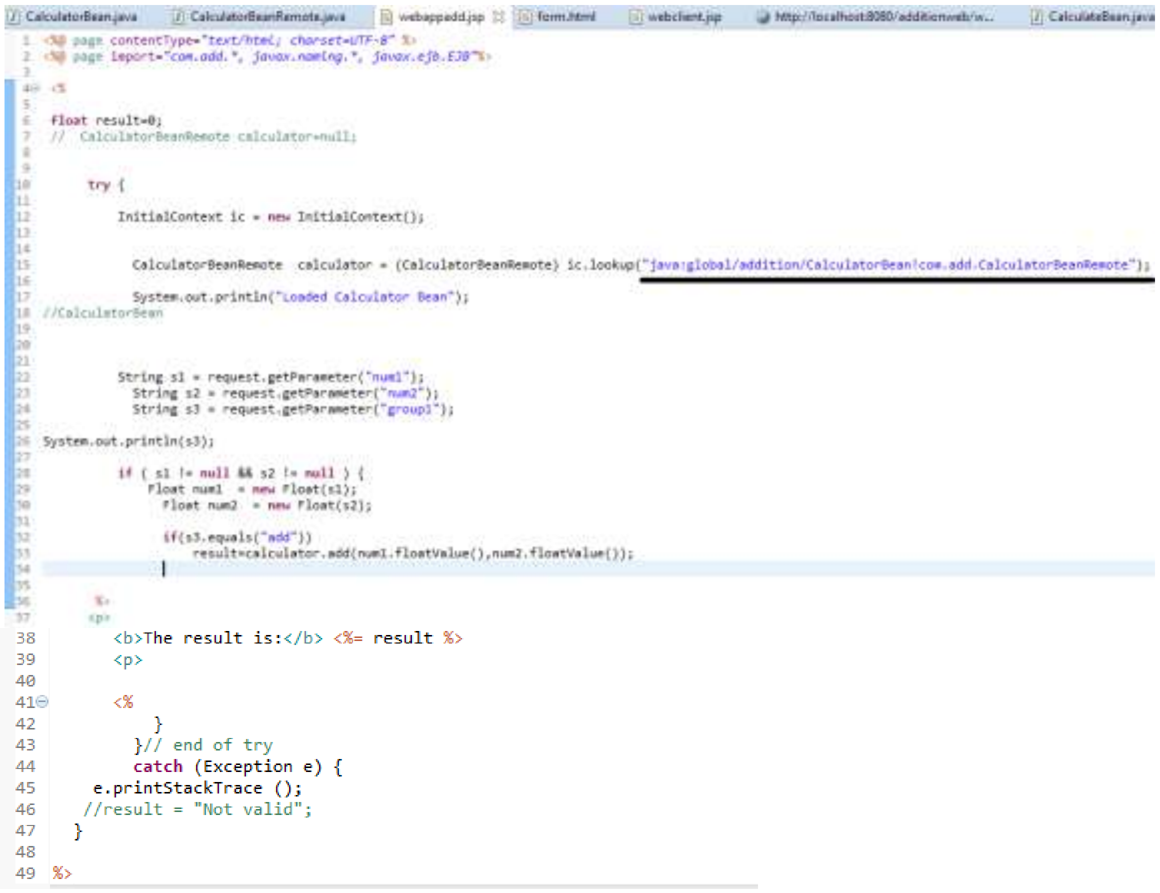


Write the following code in form.html
//form.html



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

    <body bgcolor="blue">
        <h1>Calculator</h1>
        <hr>

        <form action="webappadd.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>Seclect your choice:</b><br>
    <input type="radio" name="group1" value ="add">Addition<br>

    <p>
            <input type="submit" value="Submit">
            <input type="reset" value="Reset"></p>

        </form>


    </body>
</html>
```

Write following code in webappadd.jsp



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

<%

Float result=0;
//  CalculatorBeanRemote calculator=null;


    try {

        InitialContext ic = new InitialContext();


          CalculatorBeanRemote  calculator = (CalculatorBeanRemote) ic.lookup("java:global/addition/CalculatorBean!com.add.CalculatorBeanRemote");

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



        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=calculator.add(num1.floatValue(),num2.floatValue());



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

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

%>
```

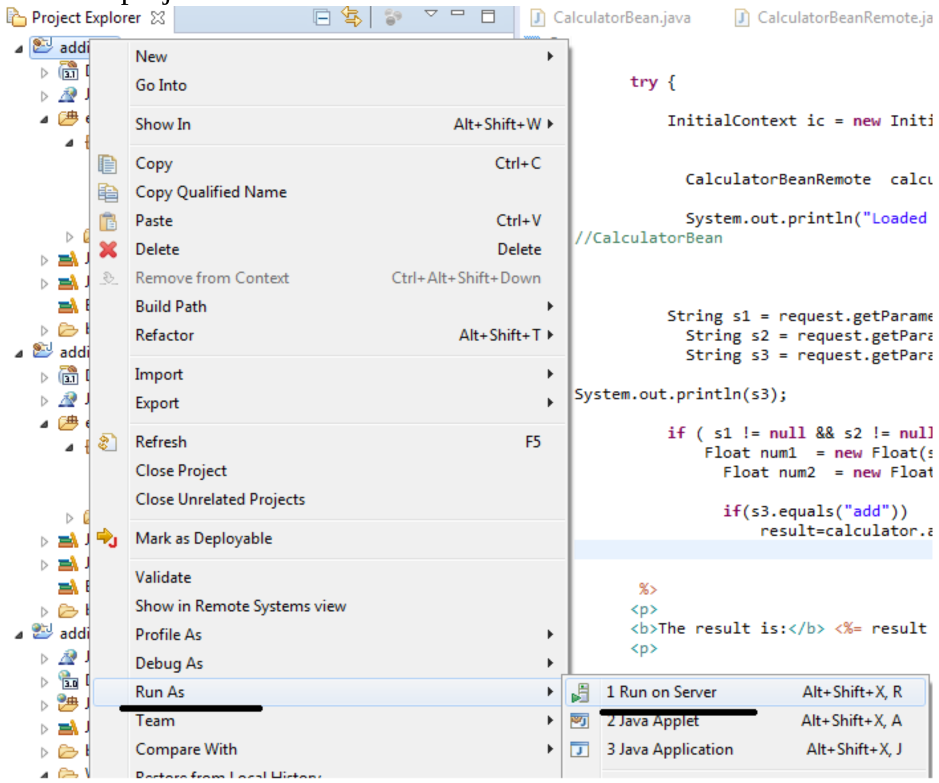
Step 10 :

Copy the url from step 5 and add that url in wepappadd code as given above.

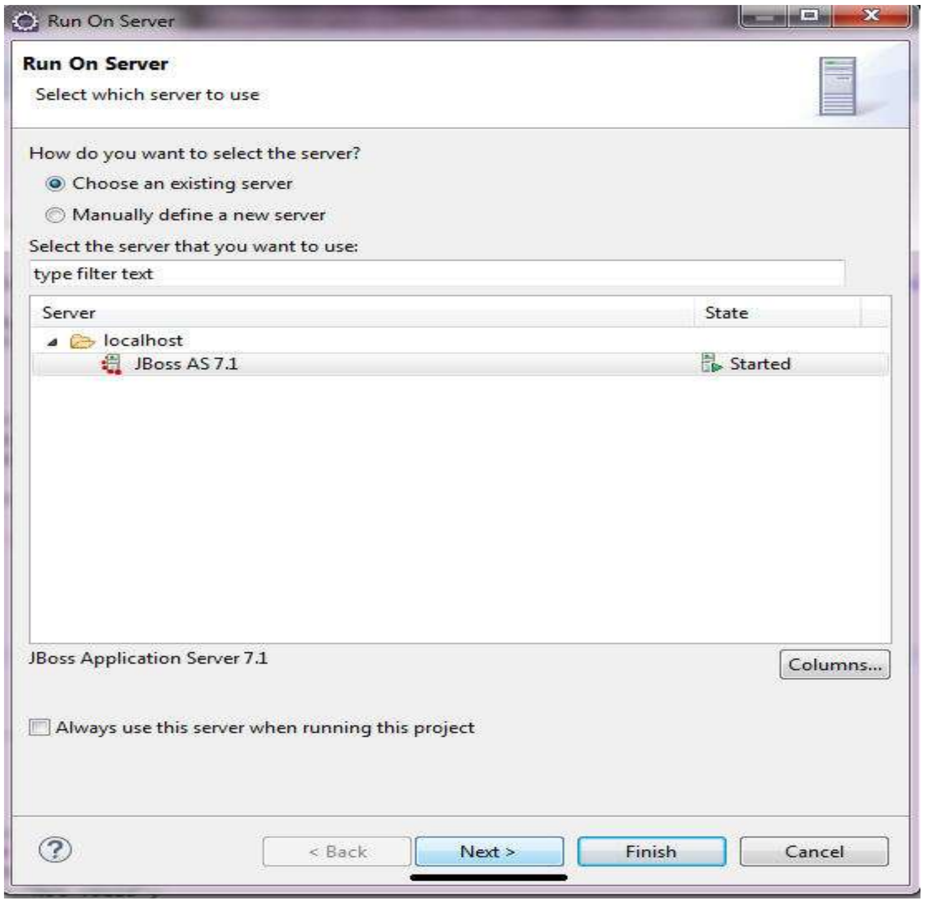
Step 11 :

Running the application :

Right click on project addition-> run as -> run on server



```
try {
        InitialContext ic = new Initi
          CalculatorBeanRemote  calcu
          System.out.println("Loaded
//CalculatorBean
        String s1 = request.getParame
          String s2 = request.getPara
          String s3 = request.getPara
System.out.println(s3);
        if ( s1 != null && s2 != null
            Float num1  = new Float(s
              Float num2  = new Float
              if(s3.equals("add"))
                result=calculator.a
   %>
<p>
<b>The result is:</b> <%= result
<p>
```



How do you want to select the server?

[✓ Choose an existing server]

Manually define a new server

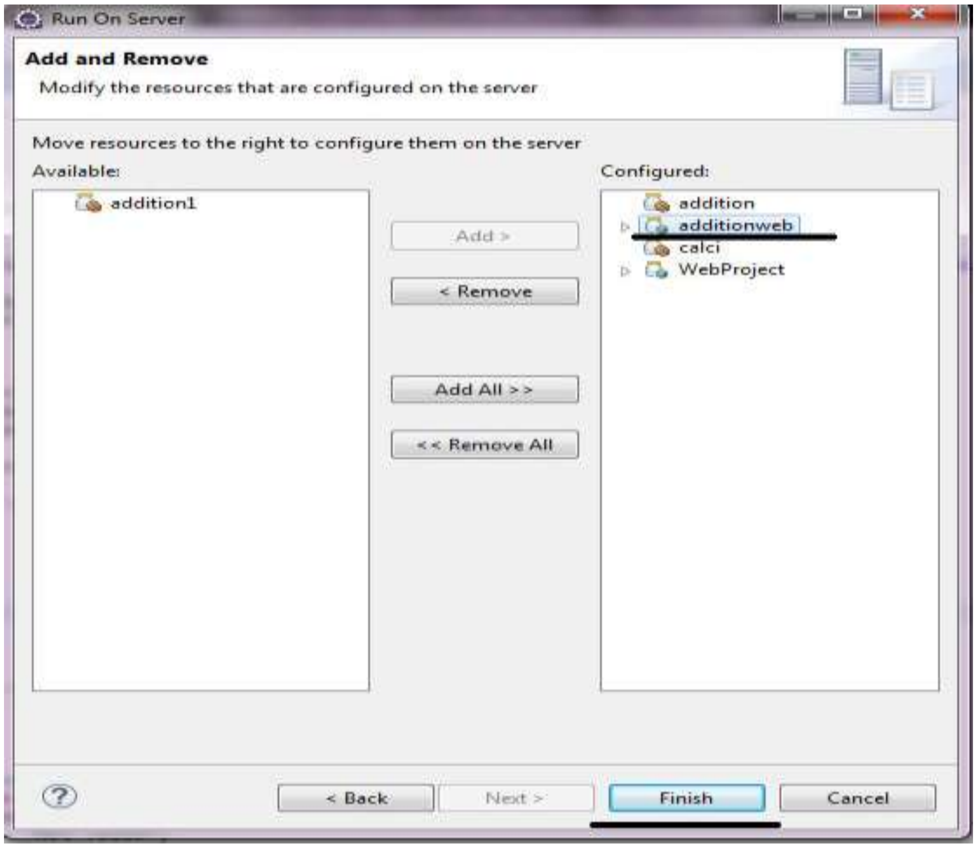
Select the server that you want to use:

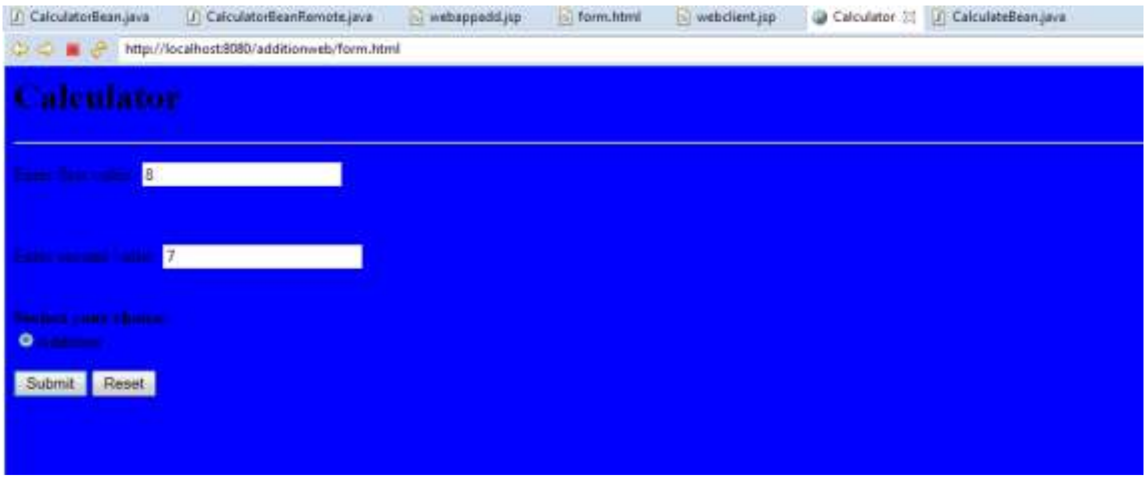
type filter text

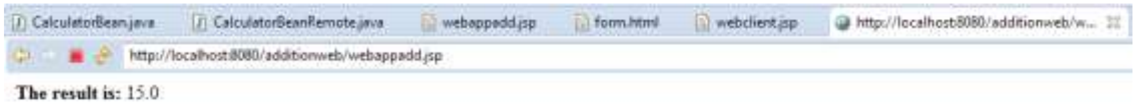
Server	State
localhost	
JBoss AS 7.1	Started

JBoss Application Server 7.1

Always use this server when running this project






CalculatorBean.java
CalculatorBeanRemote.java
webappadd.jsp
form.html
webclient.jsp
http://localhost:8080/additionweb/w...
http://localhost:8080/additionweb/webappadd.jsp

The result is: 15.0

CONCLUSION: Hence we design and implemented a business interface web application using EJB.