1 Introduction

In this chapter we introduce the student to the concept of Stateless Session Beans.

1.1 Stateless Session Beans

Stateless Session Beans are session beans that do not maintain the conversational state between method calls of the client and the bean. This means, when a client calls the method of a bean (EJB), that invocation is serviced only once and thereafter the bean forgets about the client. When the same client makes a subsequent call, the bean treats it as a new invocation. The previous state of the conversation is not kept. Consequently stateless session beans are used to run independent tasks that don't need the results of previous calls for them to execute successfully. An example could be determining the validity of an Identity Document.

1.1.1 How does a Stateless Session Bean work?

The container creates a pool of stateless session beans. When a request is made for a stateless session bean, the container takes one from the pool and service the request. After the request has been serviced, the bean is not destroyed, rather is it is returned back to the pool for reuse. So the advantage of using stateless session beans is that there is an efficient use of resources. Clients get to share the beans.

1.1.2 How to create a Stateless Session Bean?

A Stateless Session Bean is a Plain Old Java Object (POJO) annotated with the **@Stateless** annotation.

```
1  import java.ejb.Stateless;
2
3  @Stateless
4  public class PropertyManagerSB {
5
6  }
7
```

The **@Stateless** annotation turns a normal Java class into a Stateless Session Bean. The annotation is found in the **java.ejb** package.

All Session Beans are accessed through an interface. This means we need to always define an interface for a Session Bean and have the Bean implement it.

```
1
2
3  public interface PropertyManagerInterface {
4      public String determineLowestIncomeInfo(String areaCode);
5      public String determineHighestIncomeInfo(String monthCode);
6  }
7
```

Then we will have a bean that implements the interface

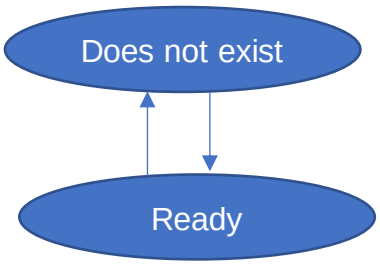
```
1
2
3  @Stateless
4  public class PropertyManagerSB implements PropertyManagerInterface{
5      public String determineLowestIncomeInfo(String areaCode){
6          //code
7      }
8
9      public String determineHighestIncomeInfo(String monthCode){
10         //code
11     }
12 }
```

1.1.3 Lifecycle of a stateless session bean

A stateless session bean has two states, namely:

- Does not exist; and
- Ready

The figure below shows the two states of a stateless session bean.



A stateless session bean moves between the two states. The container is responsible for creating a pool of stateless session beans. The beans are created either at startup

of the container or at first invocation. Before then, the beans are not existing. After creation, the stateless session beans are ready to service client requests. During instantiation, the container performs any depency injection required by the beans and thereafter execute methods annotated with **@PostConstruct**. These are methods that must be executed after the container has constructed the beans. This could be seen as code that a programmer might want to run to initialise a bean, like the opening of connections to databases.

Also, a bean method can be annotated with an **@PreDestroy** annotation. This is code that must be ran before a bean is destroyed. This could be seen as more of clean-up code, closing connections to databases before a bean is destroyed by the container.

Example

In this example we are going to create a web application that uses EJBs to convert between currencies. The web application will have the following functionalities:

- Convert a dollar to a rand.
- Convert a rand to a dollar.

The relationship between the currencies is that $1 is equal to R20.

Solution approach

There are two ways in which this problem can be solved. We can either have a servlet for each functionality or have one servlet for all the functionalities. In solution approach 1 we will use the former approach and in solution approach 2 we use the latter.

Solution approach 1

The solution to this problem is going to be done in three sequential parts. In **Part A** we will create an **EJB module** project. This project will have a stateless session bean that implements two methods for converting between the currencies. A jar file of the EJB module will be created.

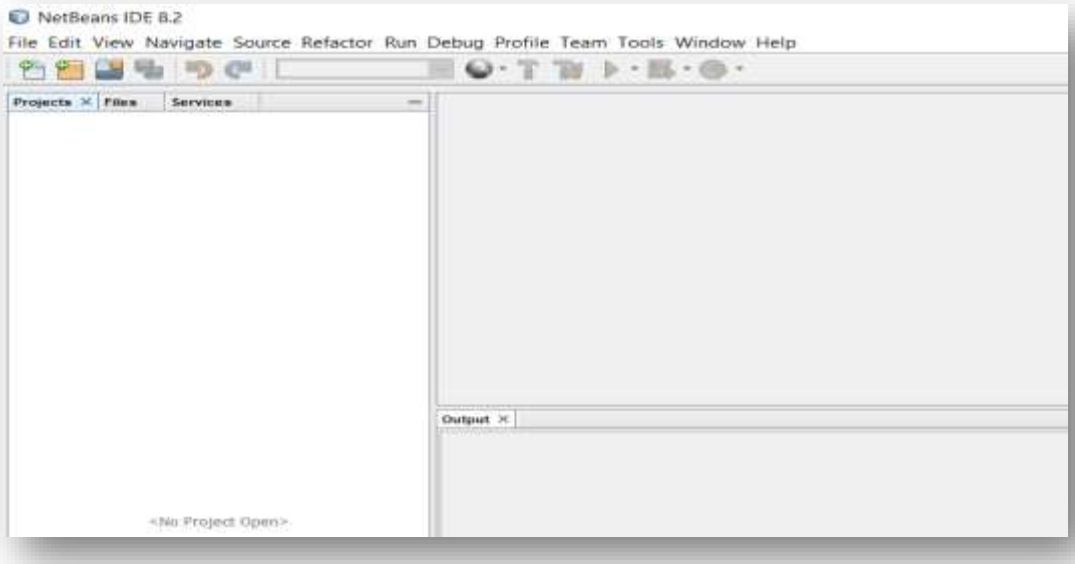
In **Part B** we will create a **web client** project to the **EJB module**. The project will mainly consist of two servlets which will serve as clients to the EJB module. The first servlet will consume/use/invoke the **dollar to rand** conversion method of the EJB, and

the second servlet will consume the **rand to dollar** conversion method of the EJB. This step requires us to have the **jar** file of the **EJB** module installed as a library in the web client project. In **Part C** we will run the client project using a browser.

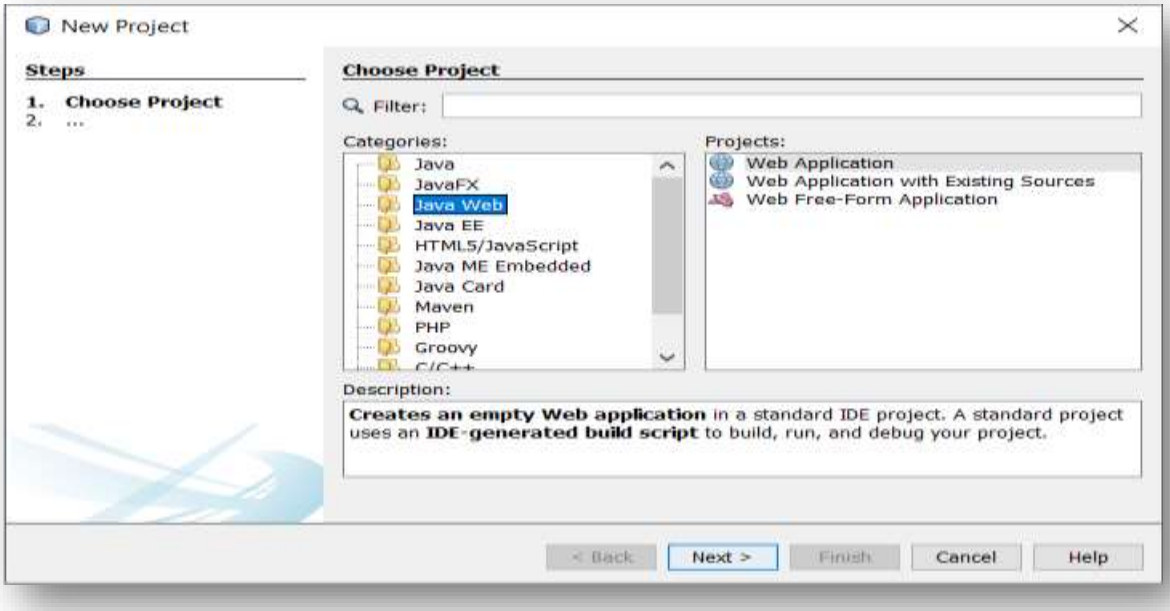
Part A - Create an EJB module project.

To successfully create a working EJB project, perform the following steps:

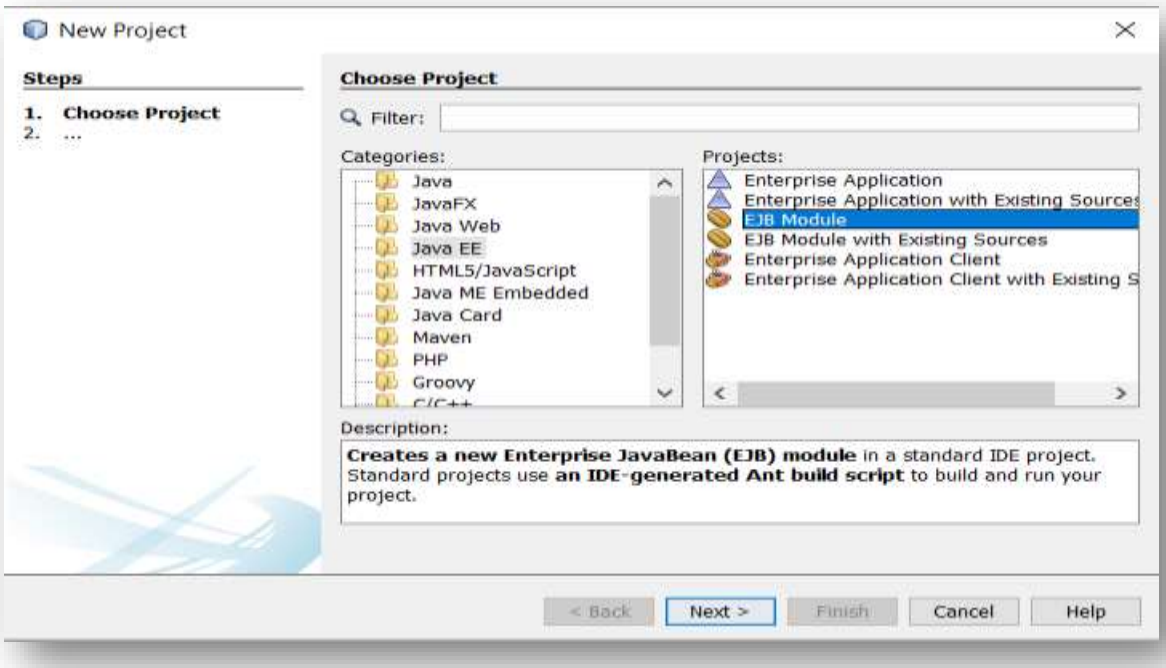
1. Launch NetBeans.



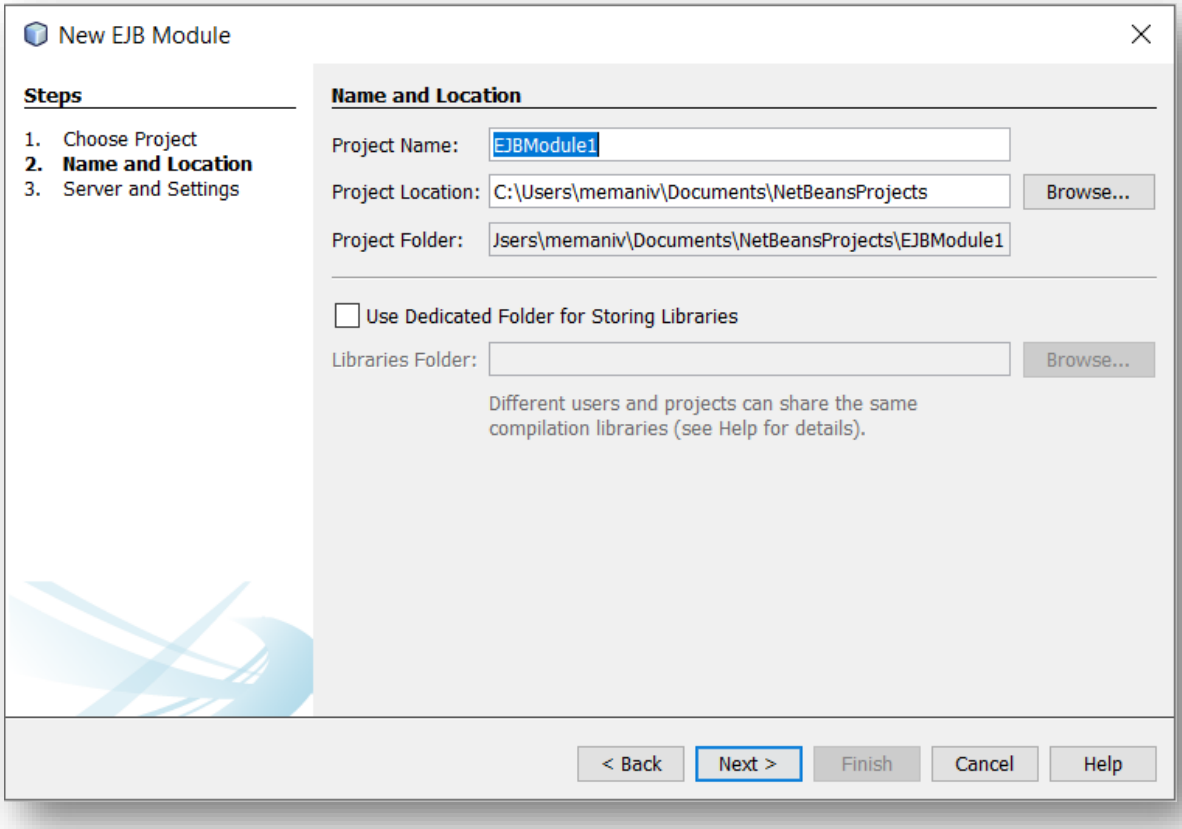
2. Click on **File | New Project**.



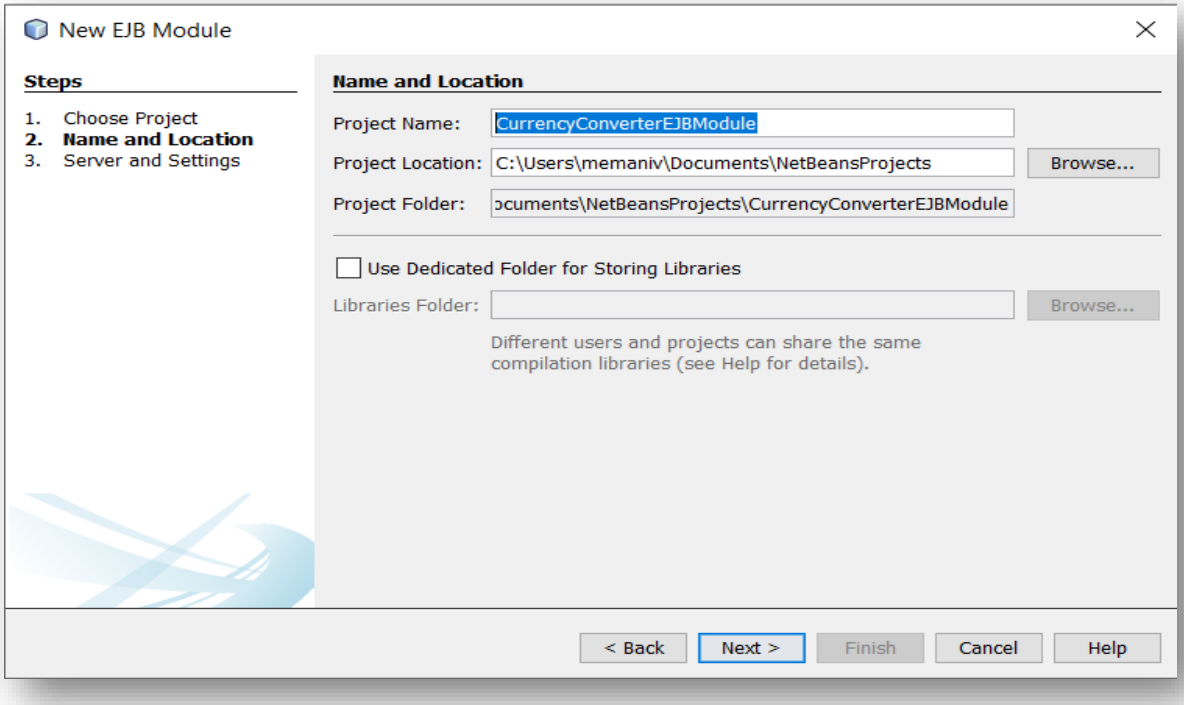
3. Select **Java EE** under **Categories** and **EJB Module** under **Projects**.



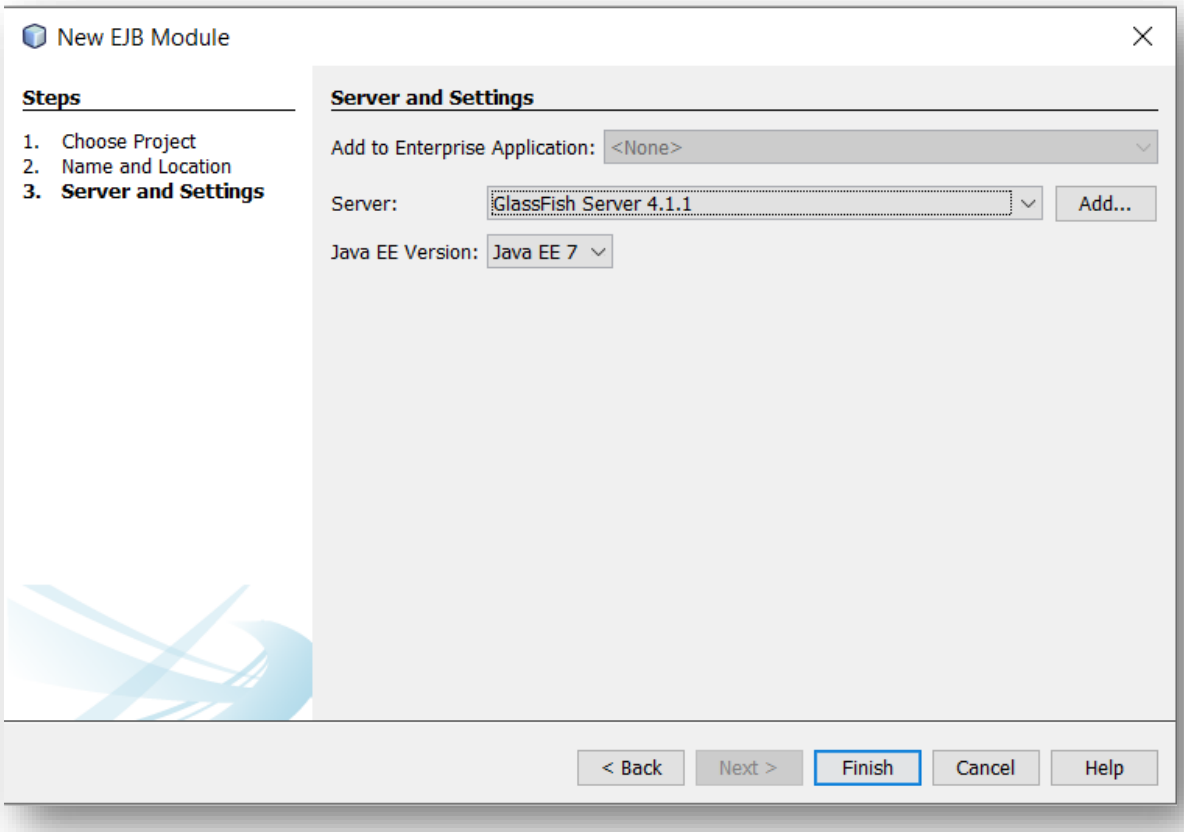
4. Click **Next**.



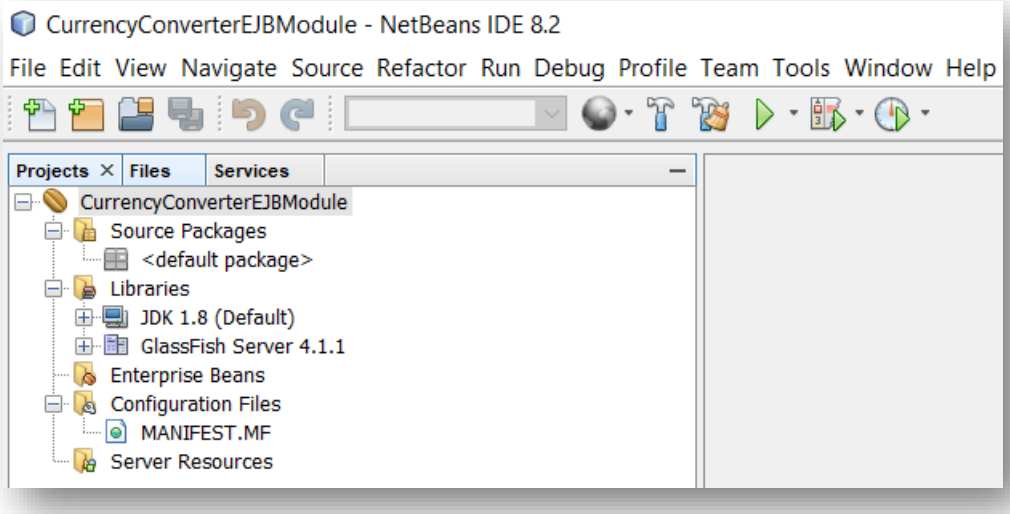
5. Name the project as **CurrencyConverterEJBModule**.



6. Click **Next**.

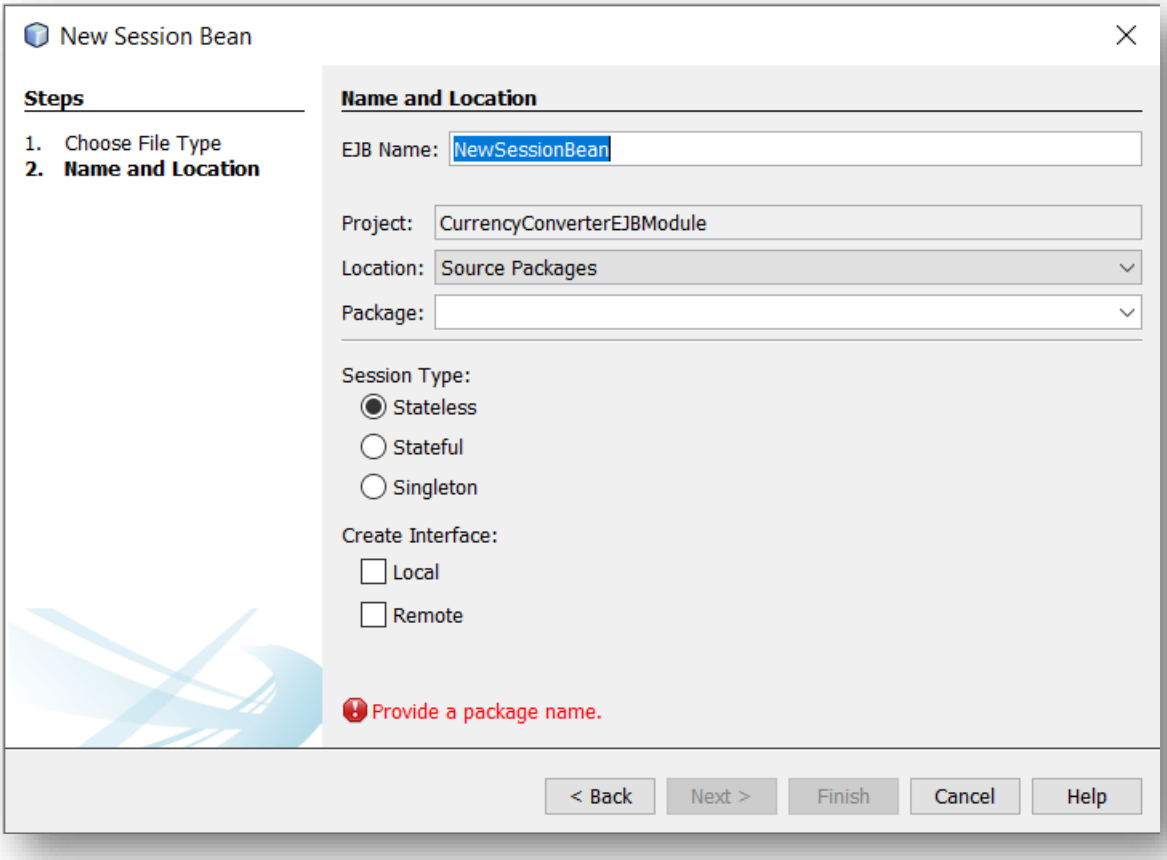


7. Click **Finish**.

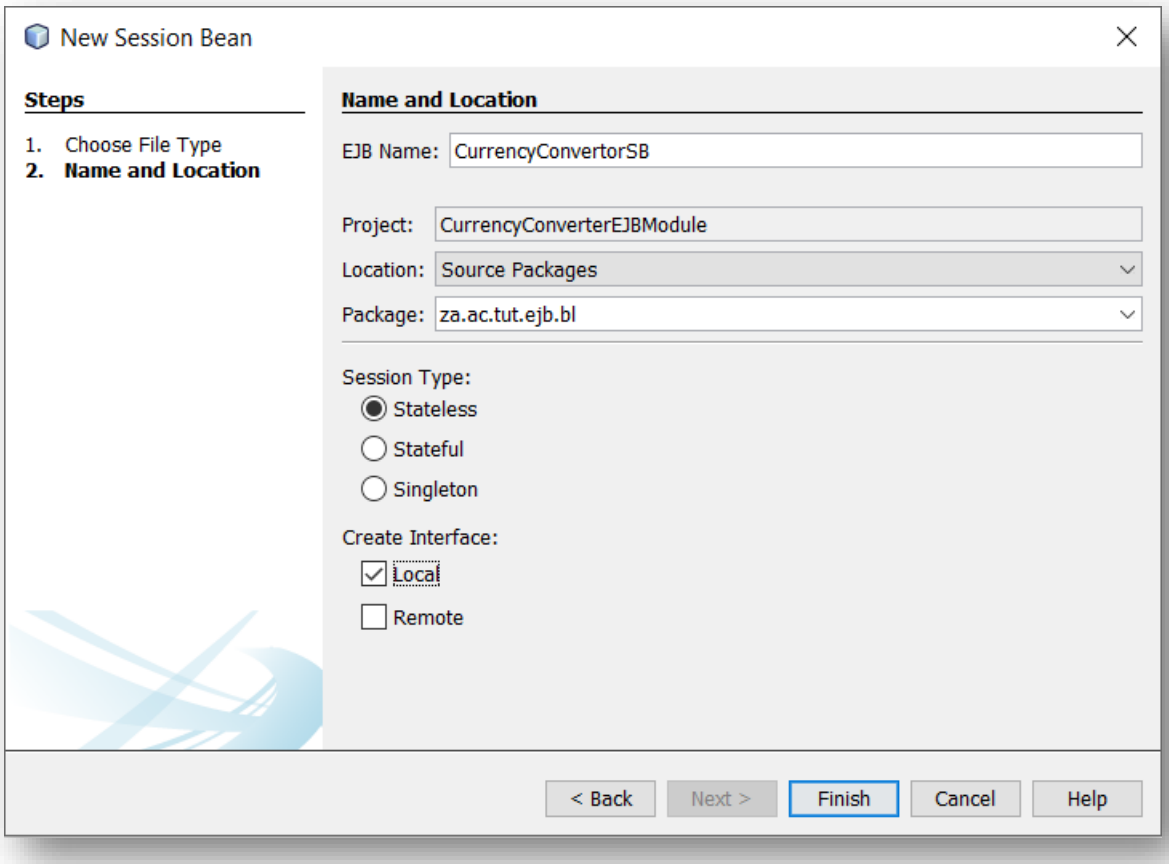


8. Create a stateles session bean. Perform the following steps:

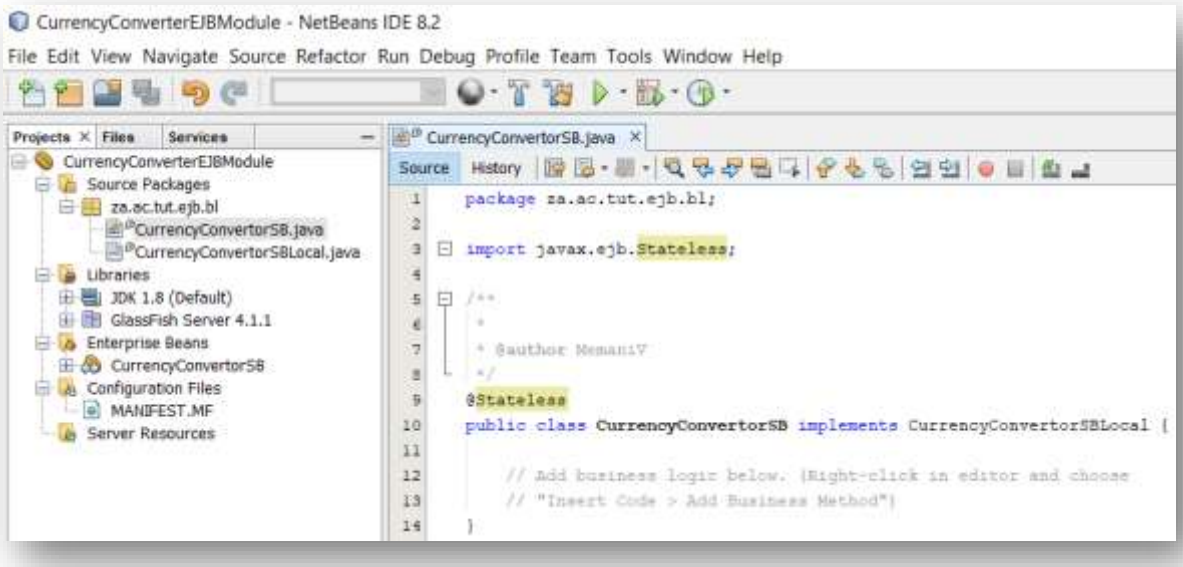
   8.1 Right click on the project and select **New | Session Bean**.



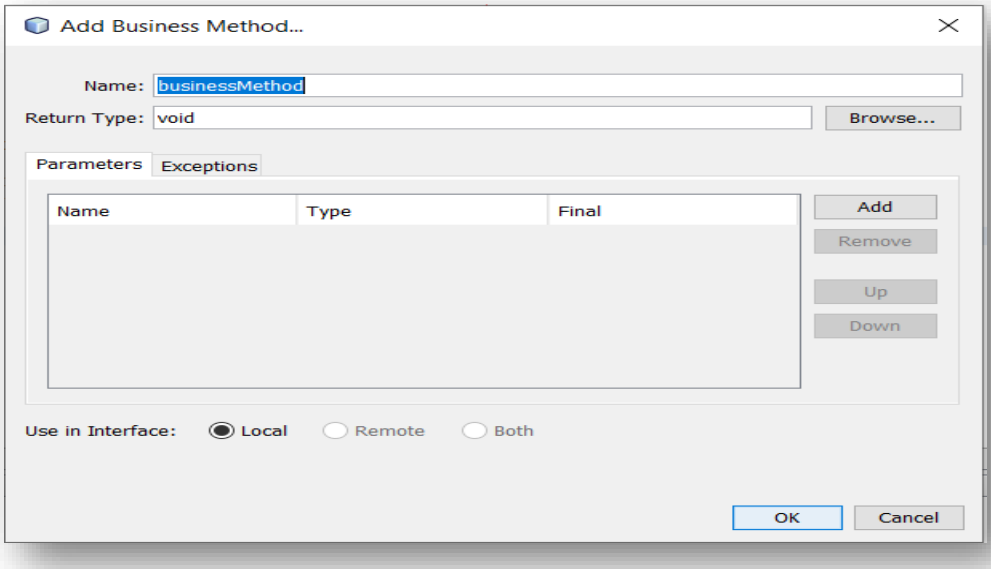
8.2 Name the EJB as **CurrencyConvertorSB**, package it under **za.ac.tut.ejb.bl,** select **Stateless** under **Session Type,** and select the **Local** interface. The **bl** acronym stands for **business logic**.



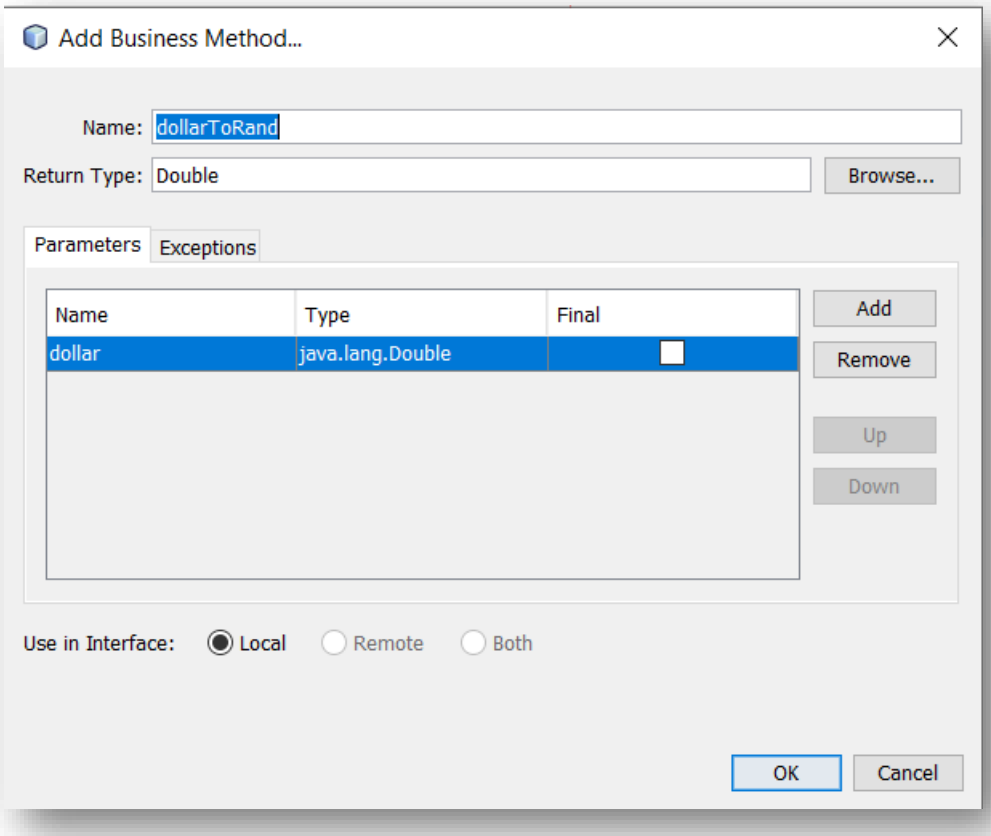
8.3 Click **Finish**.



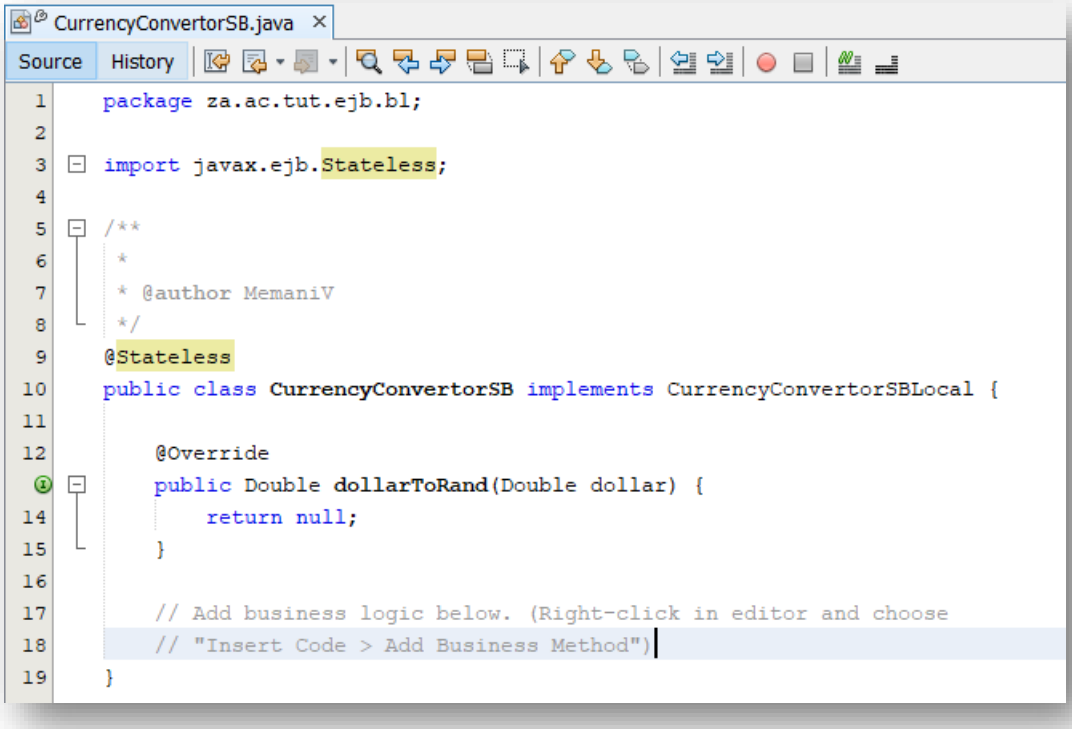
8.4 Right click inside the stateless session bean and select **Insert Code | Add Business Method**.



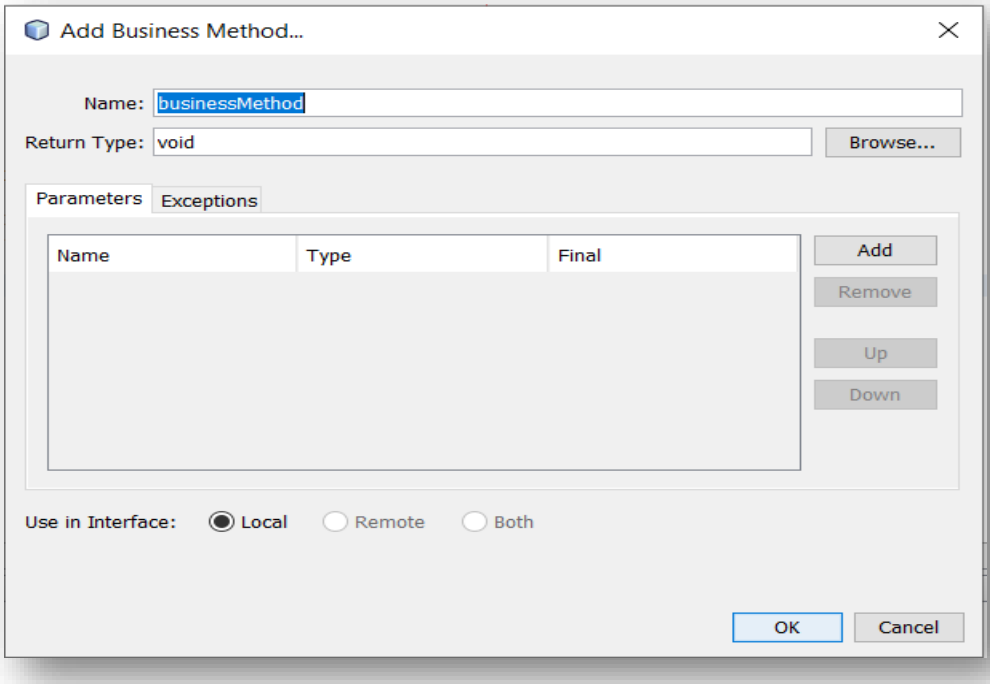
8.5 Name the method as **dollarToRand**, which returns a **Double**. Click on the **Add** button to add the parameters.



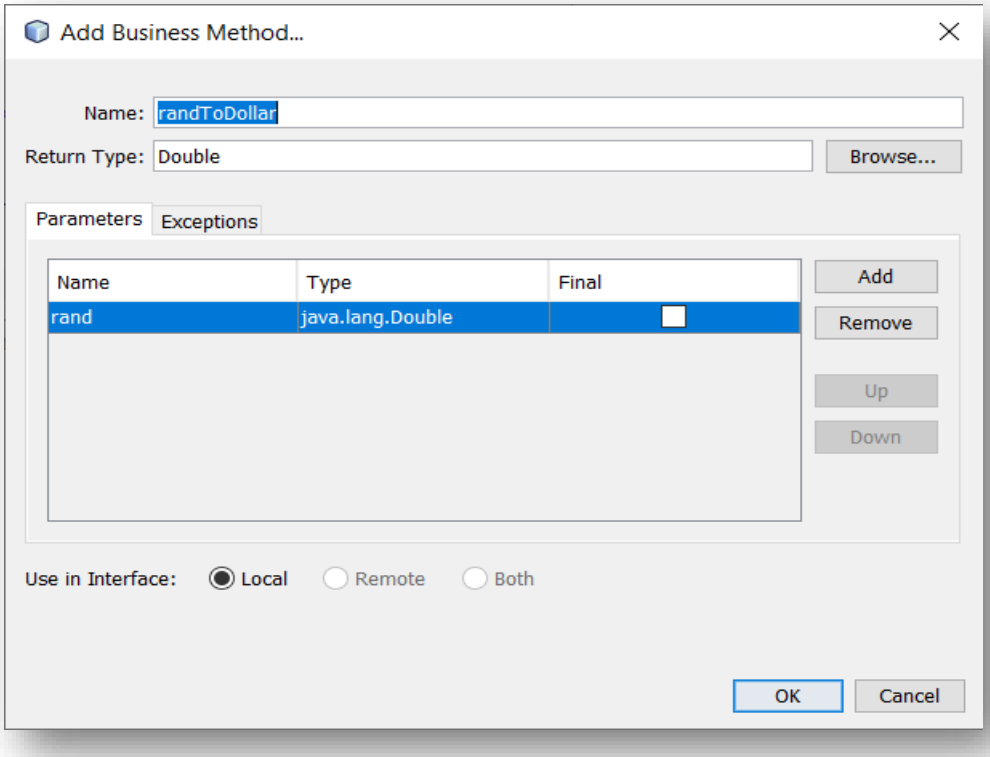
8.6 Click **OK**.



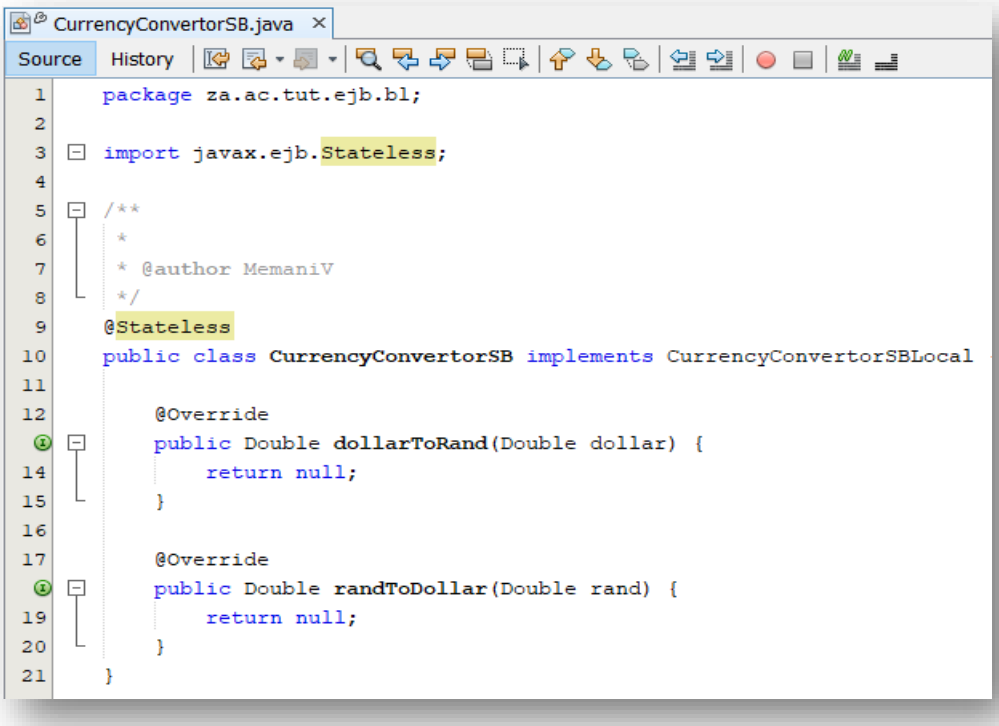
8.7 Add the second method. Right click inside the stateless session bean and select **Insert Code | Add Business Method**.



8.8 Name the method as **randToDollar**, which returns a **Double**. Click on the **Add** button to add the parameters.



8.9 Click **OK**.



8.10 Complete the source code of the Stateless Session Bean.

CurrencyConvertorSB.java

```java
package za.ac.tut.ejb.bl;

import javax.ejb.Stateless;

/**
 *
 * @author MemaniV
 */
@Stateless
public class CurrencyConvertorSB implements CurrencyConvertorSBLocal {
    private final Double DOLLAR_RATE = 20.00;

    @Override
    public Double dollarToRand(Double dollar) {
        double rand = dollar * DOLLAR_RATE;
        return rand;
    }

    @Override
    public Double randToDollar(Double rand) {
        double dollar = rand / DOLLAR_RATE;
        return dollar;
    }
}
```

8.11 View the interface.

CurrencyConverterEJBModule - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects | Files | Services

- CurrencyConverterEJBModule
  - Source Packages
    - za.ac.tut.ejb.bl
      - CurrencyConvertorSB.java
      - CurrencyConvertorSBLocal.java
  - Libraries
    - JDK 1.8 (Default)
    - GlassFish Server 4.1.1
  - Enterprise Beans
    - CurrencyConvertorSB
  - Configuration Files
    - MANIFEST.MF
  - Server Resources

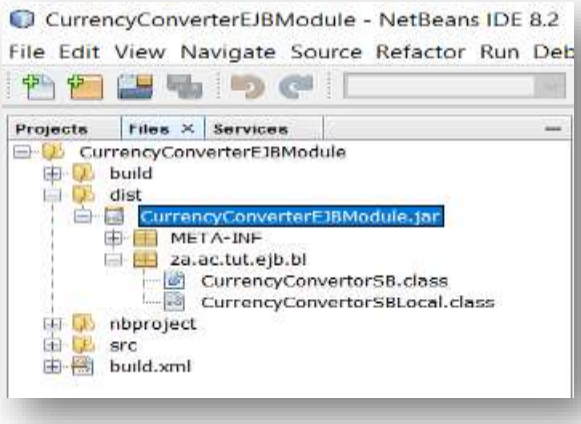
CurrencyConvertorSBLocal.java

```java
package za.ac.tut.ejb.bl;

import javax.ejb.Local;

/**
 *
 * @author MemaniV
 */
@Local
public interface CurrencyConvertorSBLocal {
    Double dollarToRand(Double dollar);
    Double randToDollar(Double rand);
}
```

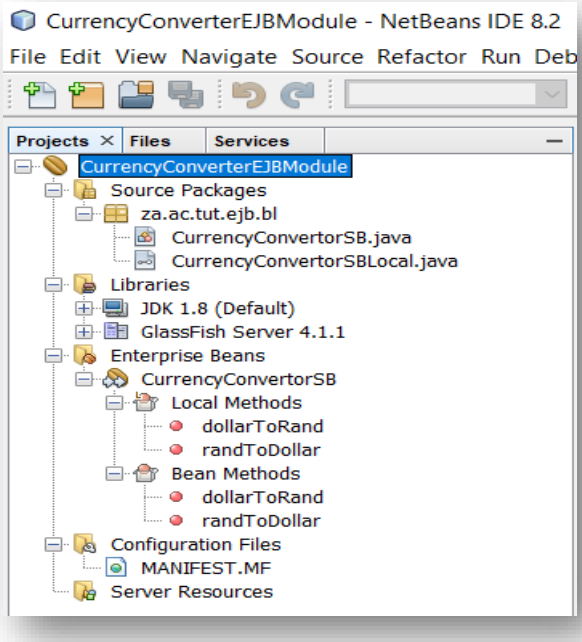
9. Compile the EJB project. Perform the following steps:

9.1 Right click on the project and select **Clean and Build**.

9.2 Click on the **Files** tab and view the contents of the **dist** folder.



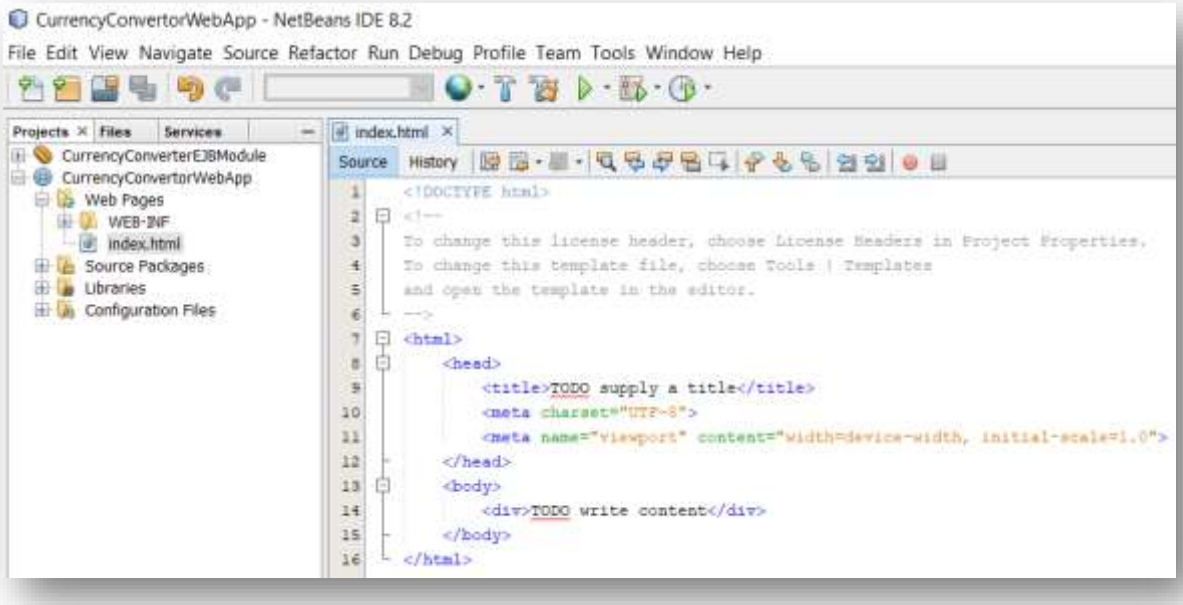
9.3 View the complete EJB project structure.



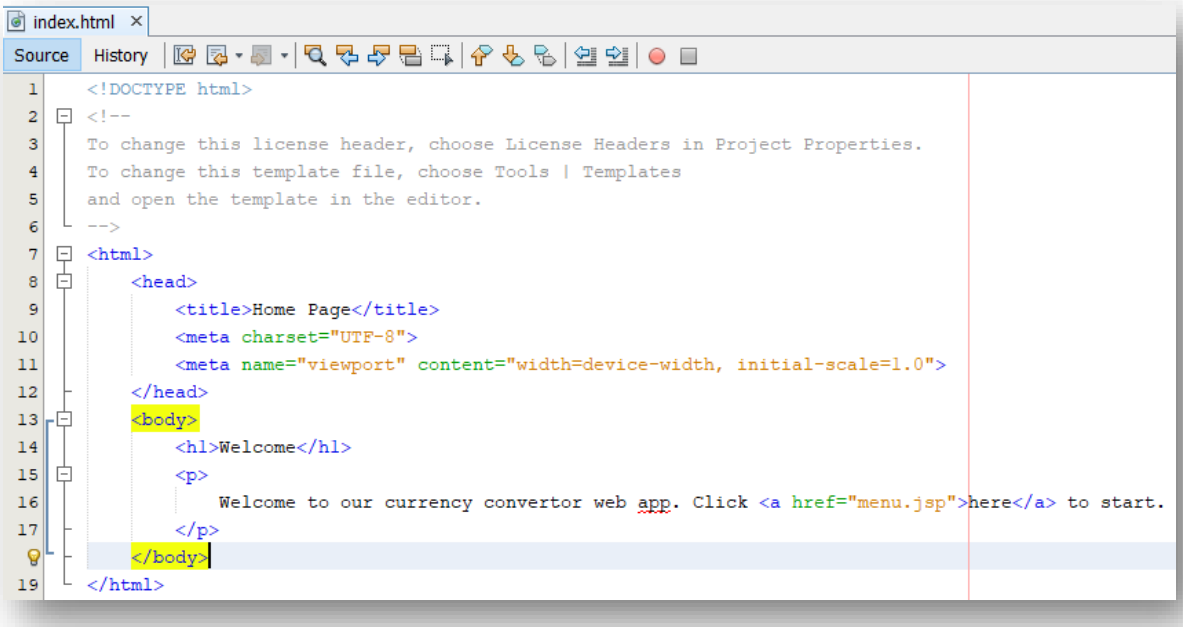
Part B - Create a Web client project

To successfully create a working web client project, perform the following steps:

1. Create a web project called **CurrencyConvertorWebApp.**



2. Edit the **index.html** file.



3. Create the **menu.jsp** file.

```
menu.jsp
Source  History
<%--
    Document    : menu
    Created on  : 19 Mar 2023, 5:31:00 PM
    Author      : MemaniV
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Menu Page</title>
    </head>
    <body>
        <h1>Menu</h1>
        <p>
            Please select one of the following options:
        </p>
        <ol>
            <li>Click <a href="rand_to_dollar.jsp">here</a> to do a rand to dollar conversion</li>
            <li>Click <a href="dollar_to_rand.jsp">here</a> to do a dollar to rand conversion</li>
        </ol>
    </body>
</html>
```

4. Create the **rand_to_dollar.jsp** file.

```
rand_to_dollar.jsp
Source  History
<%--
    Document    : rand_to_dollar
    Created on  : 19 Mar 2023, 5:35:26 PM
    Author      : MemaniV
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Rand to Dollar Conversion Page</title>
    </head>
    <body>
        <h1>Rand amount</h1>
        <p>
            Please enter the Rand amount below to convert.
        </p>
        <form action="RandToDollarConversionServlet.do" method="POST">
            <table>
                <tr>
                    <td>Rand amount: </td>
                    <td><input type="text" name="rand"/></td>
                </tr>
                <tr>
                    <td></td>
                    <td><input type="submit" value="CONVERT"/></td>
                </tr>
            </table>
        </form>
    </body>
</html>
```

5. Create the **dollar_to_rand.jsp** file.

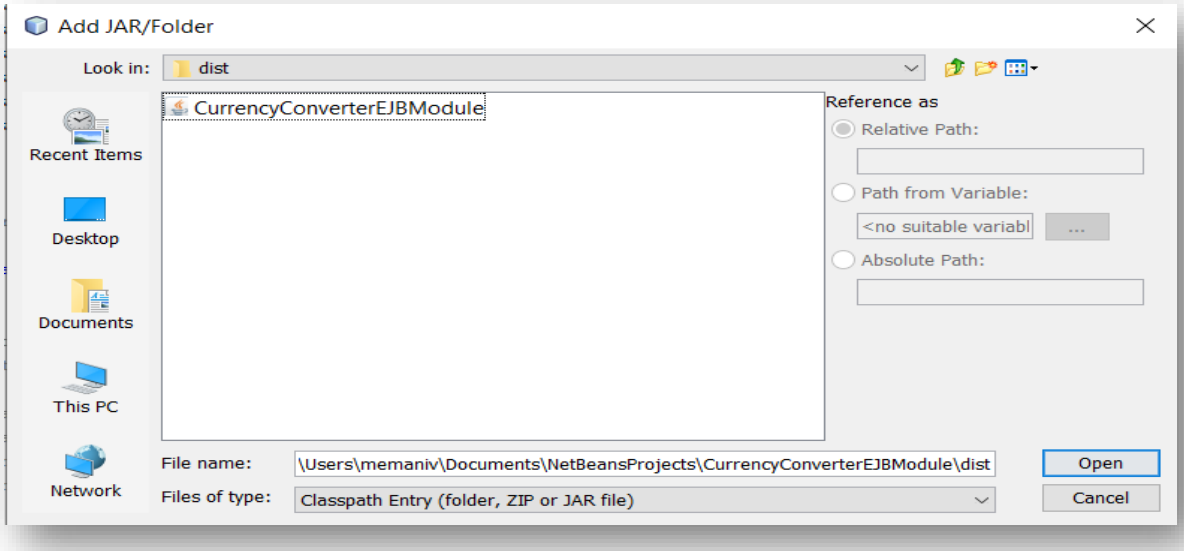
dollar_to_rand.jsp

Source History

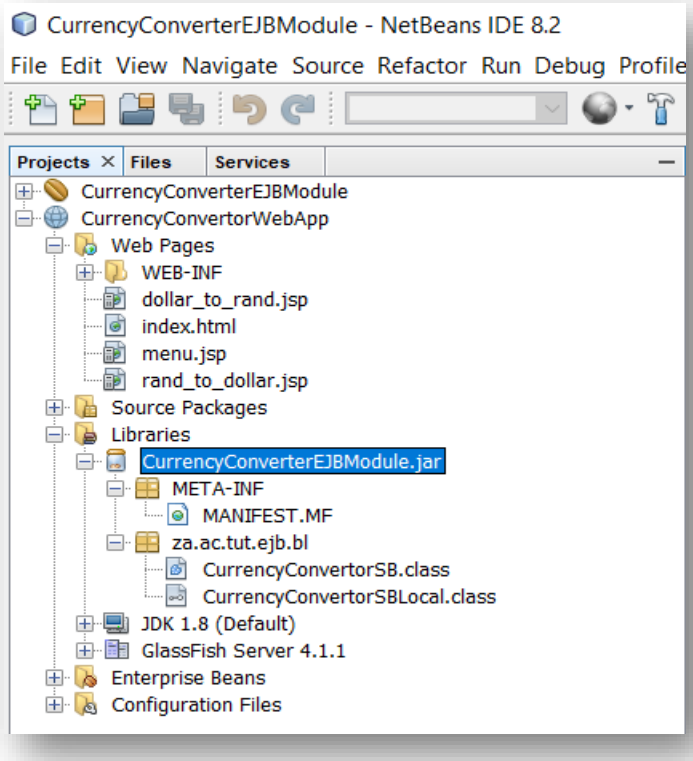
```
<%--
    Document   : dollar_to_rand
    Created on : 19 Mar 2023, 5:41:32 PM
    Author     : MemaniV
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Dollar to Rand Conversion Page</title>
    </head>
    <body>
        <hl>Dollar amount</hl>
        <p>
            Please enter the Dollar amount below to convert.
        </p>
        <form action="DollarToRandConversionServlet.do" method="POST">
            <table>
                <tr>
                    <td>Dollar amount: </td>
                    <td><input type="text" name="dollar"/></td>
                </tr>
                <tr>
                    <td></td>
                    <td><input type="submit" value="CONVERT"/></td>
                </tr>
            </table>
        </form>
    </body>
</html>
```

6. Add the EJB module library to the web app.

6.1 Right-click on the **Libraries** folder of the Web App project and select **Add JAR/Folder.** Navigate to the **dist** folder of **CurrencyConverterEJBModule.**



6.2 Select the jar file and click **Open**.



7. Create the **DollarToRandConversionServlet.java** file.

DollarToRandConversionServlet.java

Source History

```java
package za.ac.tut.web.controller;

import java.io.IOException;
import javax.ejb.EJB;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import za.ac.tut.ejb.bl.CurrencyConvertorSBLocal;

/**
 *
 * @author MemaniV
 */
public class DollarToRandConversionServlet extends HttpServlet {
    @EJB
    private CurrencyConvertorSBLocal sb;

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        Double dollar = Double.parseDouble(request.getParameter("dollar"));
        Double rand = sb.dollarToRand(dollar);

        request.setAttribute("dollar", dollar);
        request.setAttribute("rand", rand);

        RequestDispatcher disp = request.getRequestDispatcher("dollar_to_rand_outcome.jsp");
        disp.forward(request, response);
    }
}
```

8. Create the **RandToDollarConversionServlet.java** file.

```java
package za.ac.tut.web.controller;

import java.io.IOException;
import javax.ejb.EJB;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import za.ac.tut.ejb.bl.CurrencyConvertorSBLocal;

/**
 *
 * @author MemaniV
 */
public class RandToDollarConversionServlet extends HttpServlet {
    @EJB
    private CurrencyConvertorSBLocal sb;

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        Double rand = Double.parseDouble(request.getParameter("rand"));
        Double dollar = sb.randToDollar(rand);

        request.setAttribute("rand", rand);
        request.setAttribute("dollar", dollar);

        RequestDispatcher disp = request.getRequestDispatcher("rand_to_dollar_outcome.jsp");
        disp.forward(request, response);
    }
}
```

9. Create the **dollar_to_rand_outcome.jsp** file.

```jsp
<!--
    Document   : dollar_to_rand_outcome
    Created on : 19 Mar 2023, 6:15:28 PM
    Author     : MemaniV
-->

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Dollar to Rand Outcome Page</title>
    </head>
    <body>
        <h1>Rand outcome</h1>
        <%
            Double dollar = (Double)request.getAttribute("dollar");
            Double rand = (Double)request.getAttribute("rand");
        %>
        <p>
            <b>$<%=dollar%></b> is equivalent to <b>R<%=rand%></b>.
            Please click <a href="menu.jsp">here</a> to go back to the menu page or <a href="index.html">here</a> to main page.
        </p>
    </body>
</html>
```

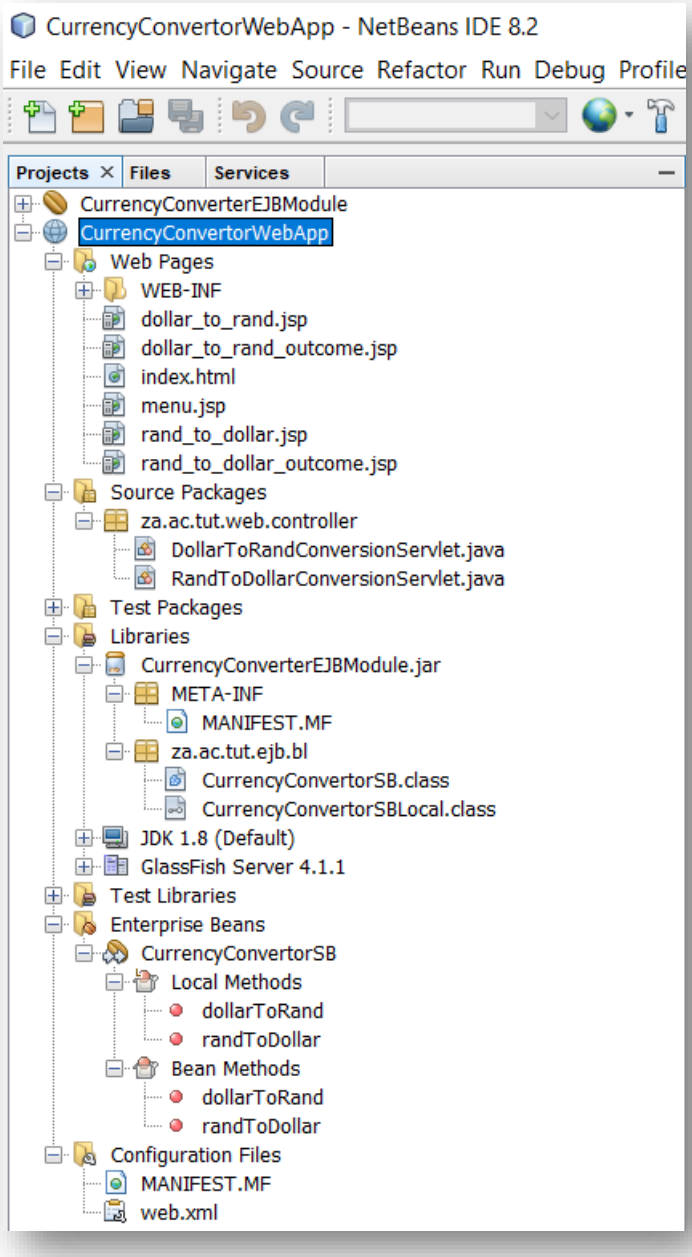
10. Create the **rand_to_dollar_outcome.jsp** file.

```
rand_to_dollar_outcome.jsp
Source  History
<%--
    Document   : rand_to_dollar_outcome
    Created on : 19 Mar 2023, 6:20:50 PM
    Author     : MemaniV
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Rand to Dollar Outcome Page</title>
    </head>
    <body>
        <h1>Dollar outcome</h1>
        <%
            Double rand = (Double)request.getAttribute("rand");
            Double dollar = (Double)request.getAttribute("dollar");
        %>
        <p>
            <b>R<%=rand%></b> is equivalent to <b>$<%=dollar%></b>.
            Please click <a href="menu.jsp">here</a> to go back to the menu page or <a href="index.html">here</a> to main page.
        </p>
    </body>
</html>
```

11. Compile the web project.

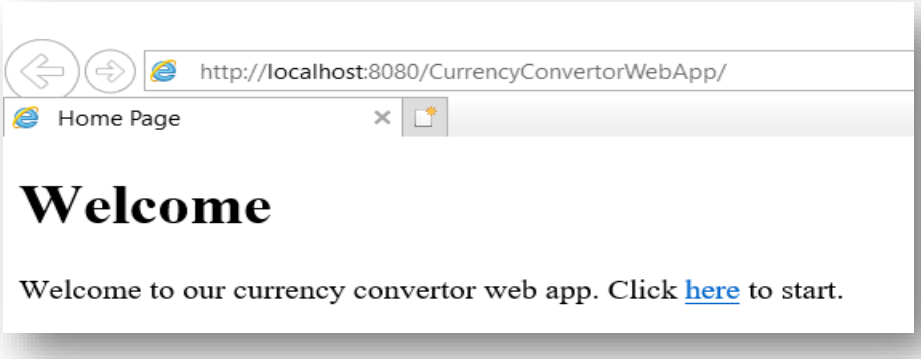
```
Output
Java DB Database Process   CurrencyConvertorWebApp (clean,dist)
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConvertorWebApp\build\web\META-INF
Copying 1 file to C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConvertorWebApp\build\web\META-INF
Copying 7 files to C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConvertorWebApp\build\web
library-inclusion-in-archive:
Copying 1 file to C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConvertorWebApp\build\web\WEB-INF\lib
library-inclusion-in-manifest:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConvertorWebApp\build\empty
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConvertorWebApp\build\generated-sources\ap-source-output
Compiling 2 source files to C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConvertorWebApp\build\web\WEB-INF\classes
compile:
compile-jsps:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConvertorWebApp\dist
Building jar: C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConvertorWebApp\dist\CurrencyConvertorWebApp.war
do-dist:
dist:
BUILD SUCCESSFUL (total time: 0 seconds)
```

12. View the entire structure of the web project.

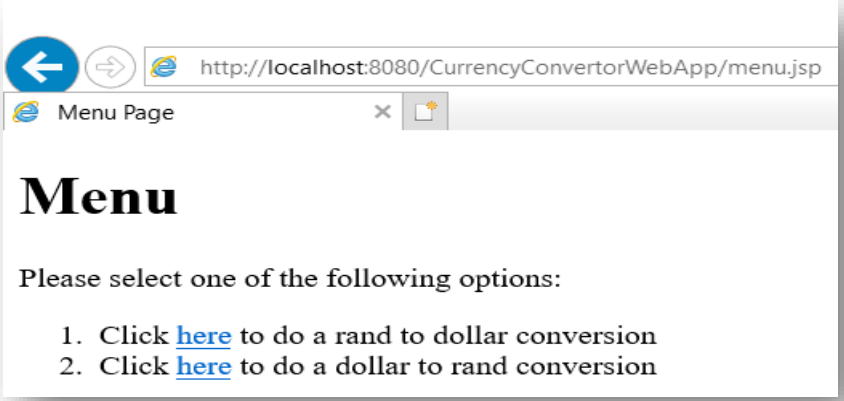


Part C - Run the web client project.

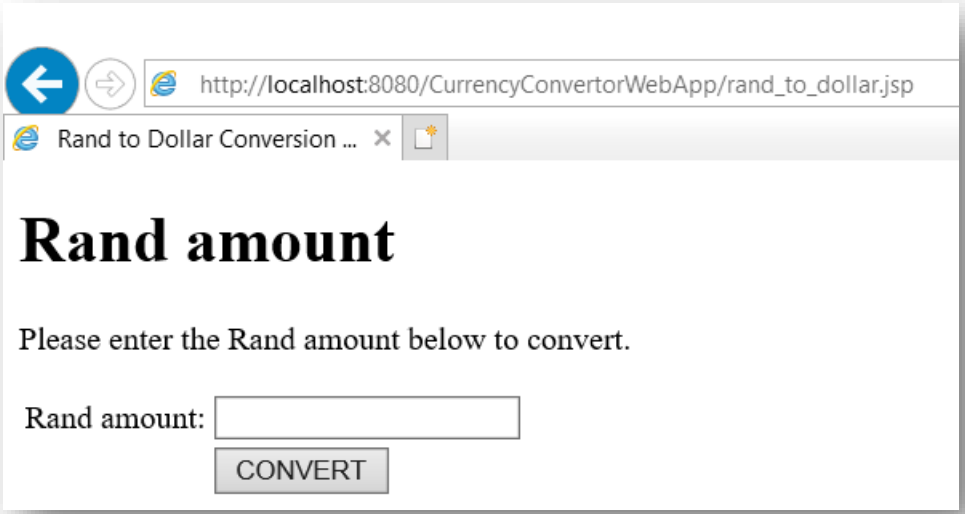
Launch the web app.



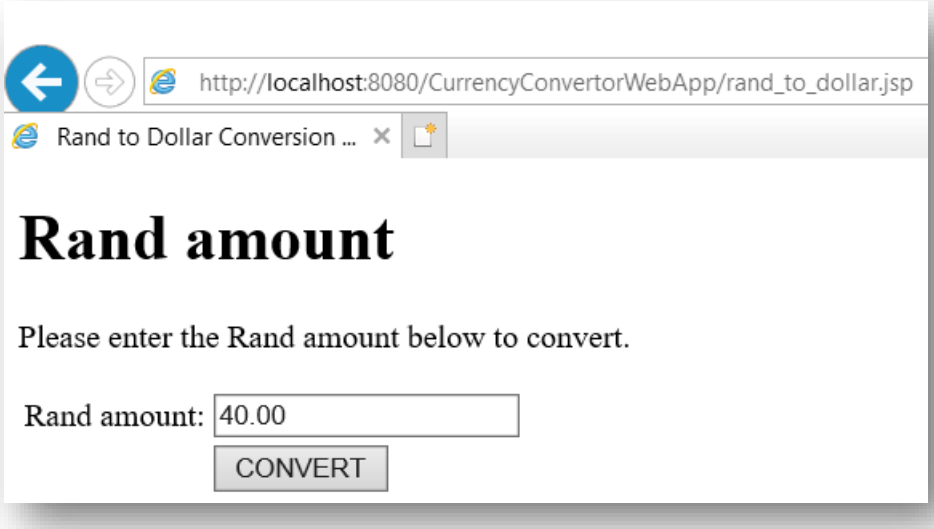
Click on the link.



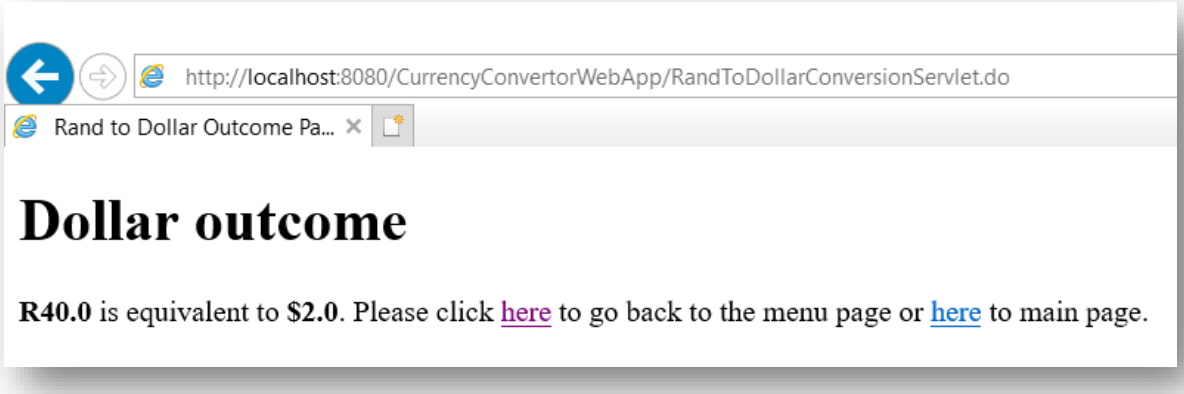
Click on the first link.



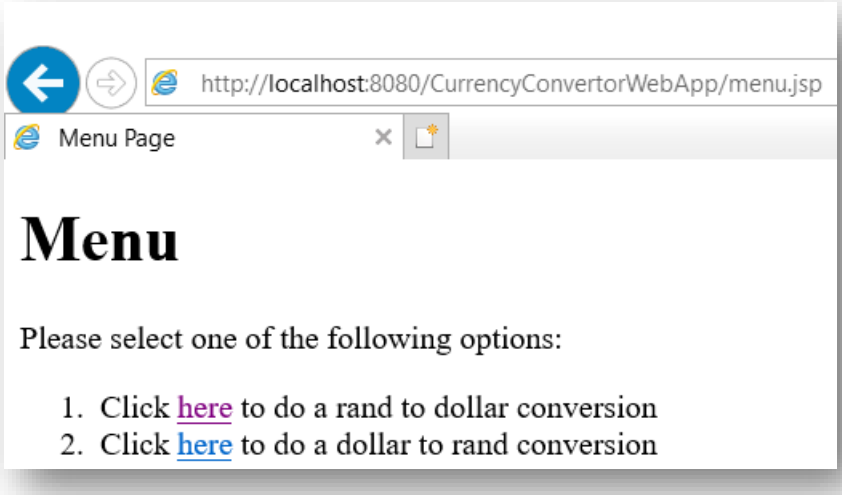
Enter the amount.



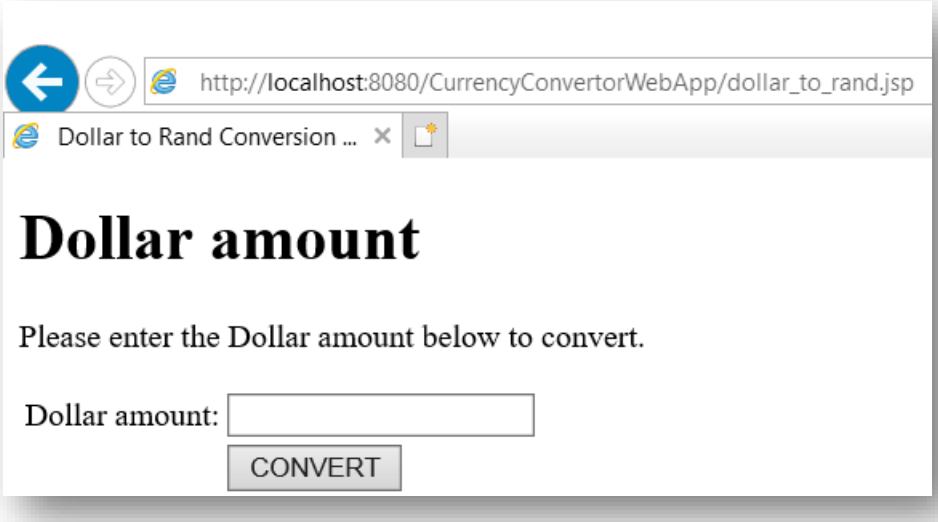
Click on the button.



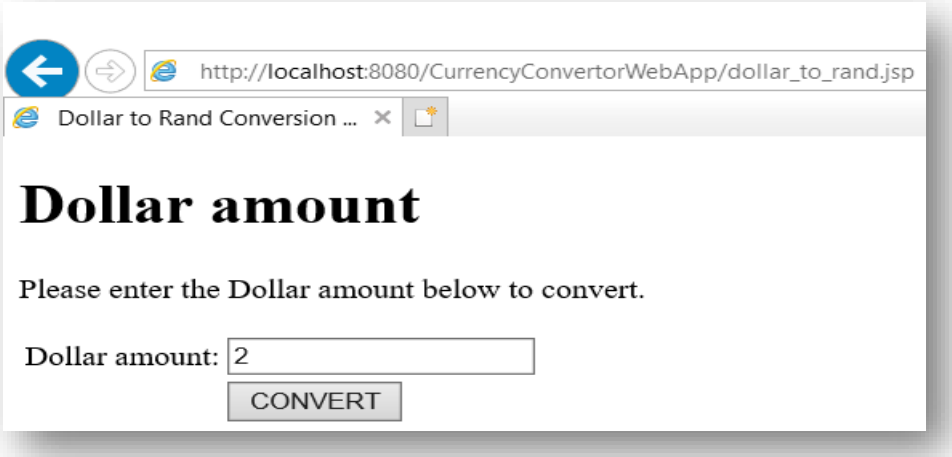
Go back to the menu page.



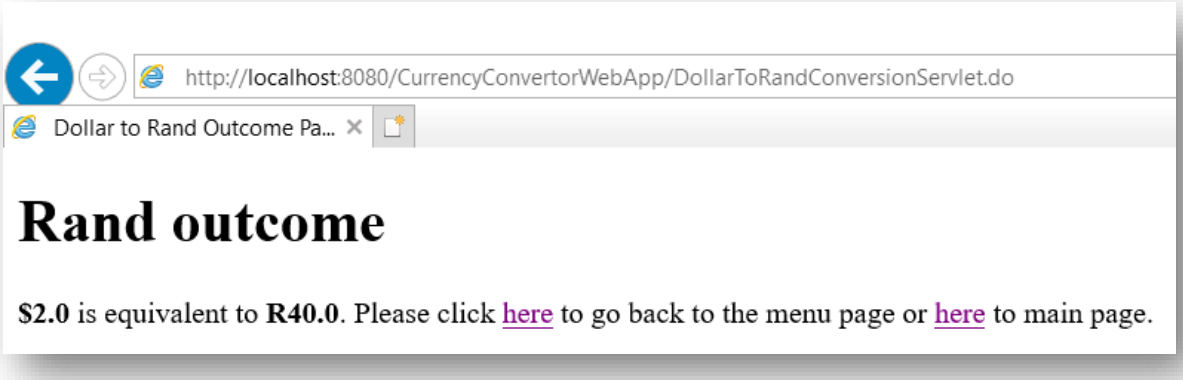
Click on the second link.



Enter the dollar amount



Click on the button.



Solution approach 2

As said earlier on, the only major change we will bring in this approach is the introduction of a single servlet for processing data. We will call the servlet **ConversionServlet**. The **EJB module** will remain the same. The minor changes will entail changing the name of the web client project to **CurrencyConvertWebAppV2**.

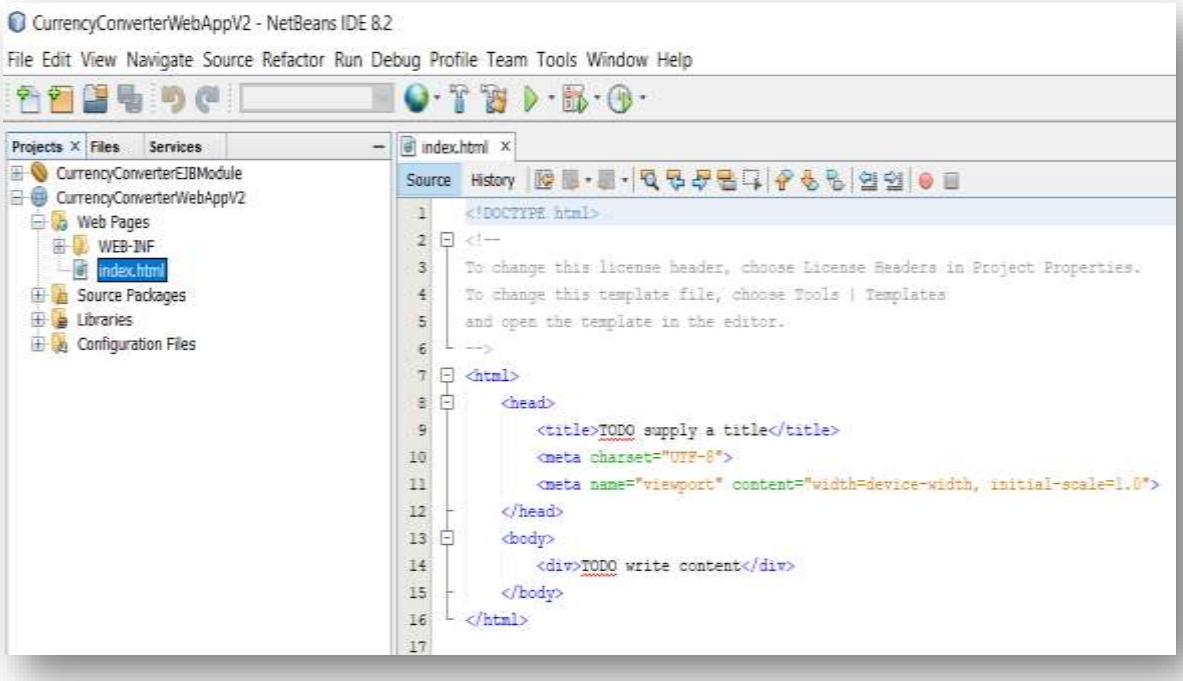
Part A - Create an EJB module project.

This remains the same as in **approach 1**. Make sure that you have the jar file of the EJB module.

Part B - Create a Web client project

To successfully create a working web client project, perform the following steps:

1. Create a web project called **CurrencyConverterWebAppV2.**



2. Edit the **index.html** file.

index.html

Source History

```html
<!DOCTYPE html>
<!--
To change this license header, choose License Headers in Project Properties.
To change this template file, choose Tools | Templates
and open the template in the editor.
-->
<html>
    <head>
        <title>Home Page</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <h1>Welcome</h1>
        <p>
            Welcome to our currency converter web app. Click <a href="menu.jsp">here</a> to start.
        </p>
    </body>
</html>
```

3. Create the **menu.jsp** file.

menu.jsp

Source History

```jsp
<%--
    Document   : menu
    Created on : 19 Mar 2023, 5:31:00 PM
    Author     : MemaniV
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Menu Page</title>
    </head>
    <body>
        <h1>Menu</h1>
        <p>
            Please select one of the following options:
        </p>
        <ol>
            <li>Click <a href="rand_to_dollar.jsp">here</a> to do a rand to dollar conversion</li>
            <li>Click <a href="dollar_to_rand.jsp">here</a> to do a dollar to rand conversion</li>
        </ol>
    </body>
</html>
```

4. Create the **rand_to_dollar.jsp** file.

rand_to_dollar.jsp

Source History

```jsp
<%--
    Document   : rand_to_dollar
    Created on : 19 Mar 2023, 5:35:26 PM
    Author     : MemaniV
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Rand to Dollar Conversion Page</title>
    </head>
    <body>
        <h1>Rand amount</h1>
        <p>
            Please enter the Rand amount below to convert.
        </p>
        <form action="ConversionServlet.do" method="POST">
            <table>
                <tr>
                    <td>Rand amount: </td>
                    <td><input type="text" name="value"/></td>
                    <td><input type="text" name="op" value="rtd" hidden="true"/></td>
                </tr>
                <tr>
                    <td></td>
                    <td><input type="submit" value="CONVERT"/></td>
                </tr>
            </table>
        </form>
    </body>
</html>
```

5. Create the **dollar_to_rand.jsp** file.

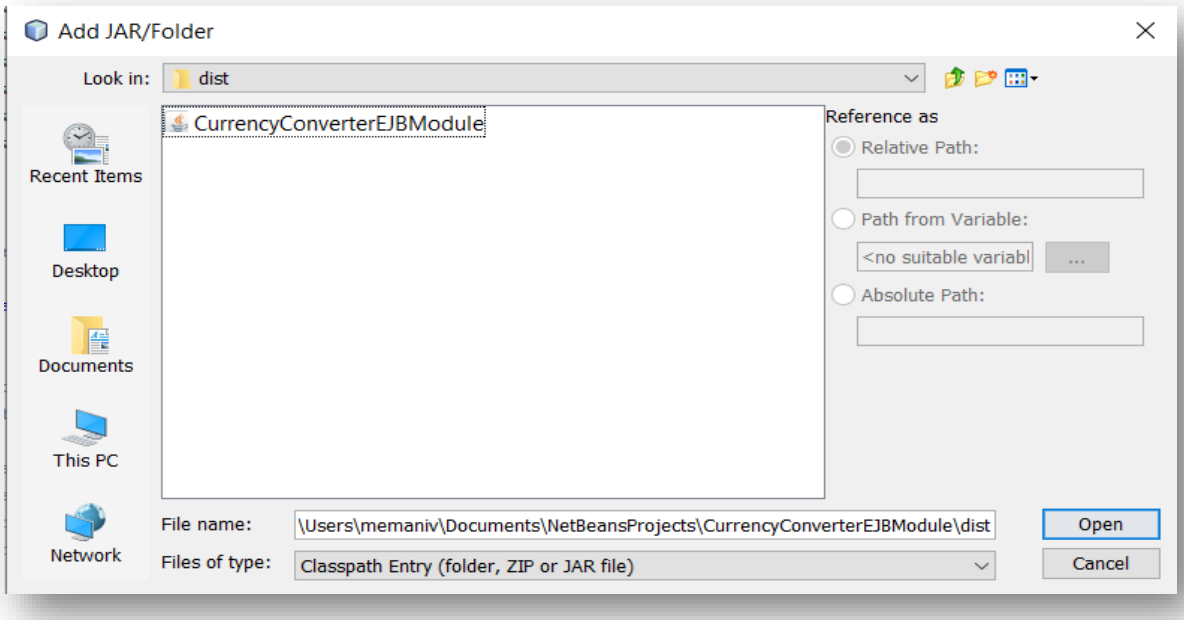
dollar_to_rand.jsp

Source History

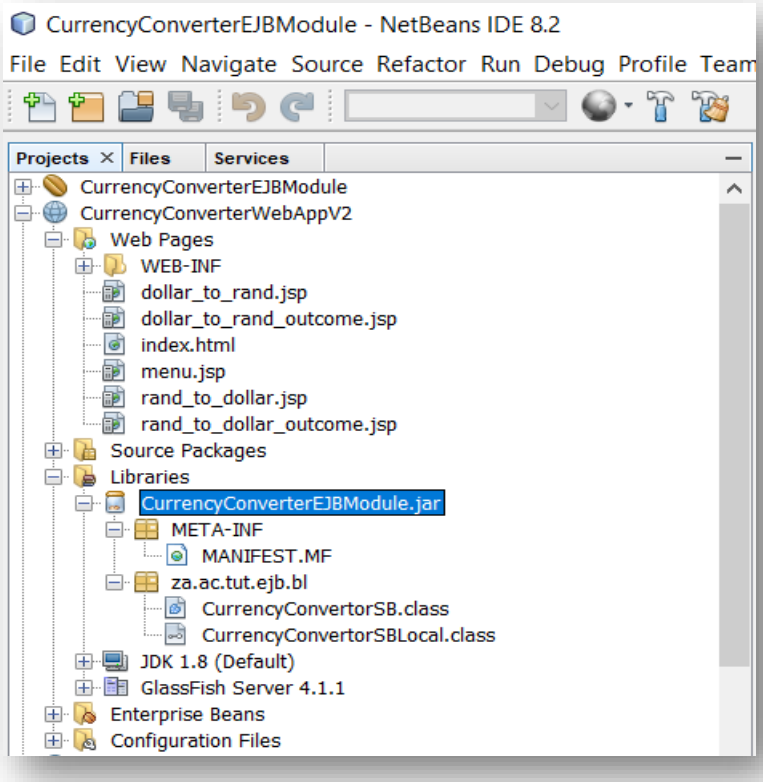
```jsp
<%--
    Document   : dollar_to_rand
    Created on : 19 Mar 2023, 5:41:32 PM
    Author     : MemaniV
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Dollar to Rand Conversion Page</title>
    </head>
    <body>
        <h1>Dollar amount</h1>
        <p>
            Please enter the Dollar amount below to convert.
        </p>
        <form action="ConversionServlet.do" method="POST">
            <table>
                <tr>
                    <td>Dollar amount: </td>
                    <td><input type="text" name="value"/></td>
                    <td><input type="text" name="op" value="dtr" hidden="true"/></td>
                </tr>
                <tr>
                    <td></td>
                    <td><input type="submit" value="CONVERT"/></td>
                </tr>
            </table>
        </form>
    </body>
</html>
```

6. Add the EJB module library to the web app.

6.1 Right-click on the **Libraries** folder of the Web App project and select **Add JAR/Folder.** Navigate to the **dist** folder of **CurrencyConverterEJBModule.**



6.2 Select the jar file and click **Open**.



7. Create the **ConversionServlet.java** file.

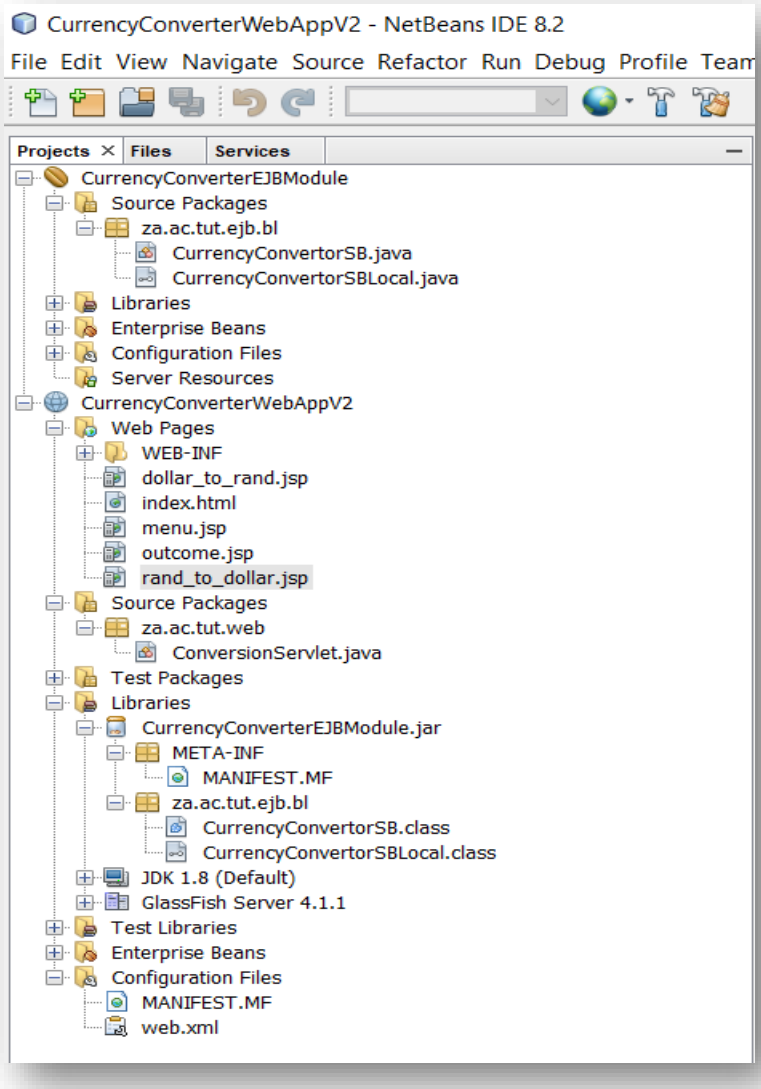
```java
/**
 *
 * @author MemaniV
 */
public class ConversionServlet extends HttpServlet {
    @EJB
    private CurrencyConvertorSBLocal ccl;

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        Double value = Double.parseDouble(request.getParameter("value"));
        String op = request.getParameter("op");
        String outcome;

        if(op.equals("dtr")){
            //perform dollar to rand operation
            Double rand = ccl.dollarToRand(value);
            outcome = "$" + value + " is equivalent to R" + rand;
        } else {
            //perform rand to dollar operation
            Double dollar = ccl.randToDollar(value);
            outcome = "R" + value + " is equivalent to R" + dollar;
        }

        request.setAttribute("outcome", outcome);

        RequestDispatcher disp = request.getRequestDispatcher("outcome.jsp");
        disp.forward(request, response);
    }
}
```

8. Create the **outcome.jsp** file.

```
outcome.jsp
Source  History
<%--
    Document   : outcome
    Created on : 19 Mar 2023, 6:15:28 PM
    Author     : MemaniV
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Currency Conversion Outcome Page</title>
    </head>
    <body>
        <h1>Currency conversion outcome</h1>
        <%
            String outcome = (String)request.getAttribute("outcome");
        %>
        <p>
            <b><%=outcome%></b>. Please click <a href="menu.jsp">here</a>
            to go back to the menu page or <a href="index.html">here</a> to main page.
        </p>
    </body>
</html>
```

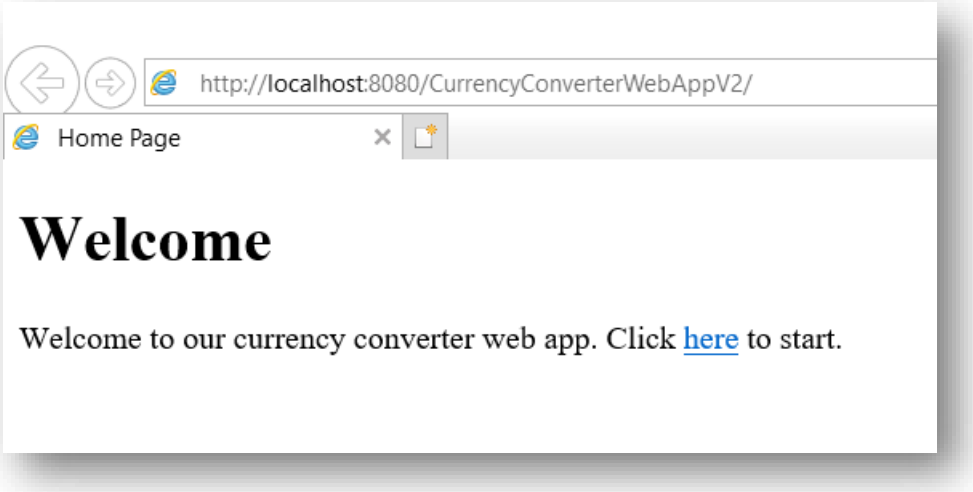
11. Compile the web project.

12. View the entire structure of the web project.

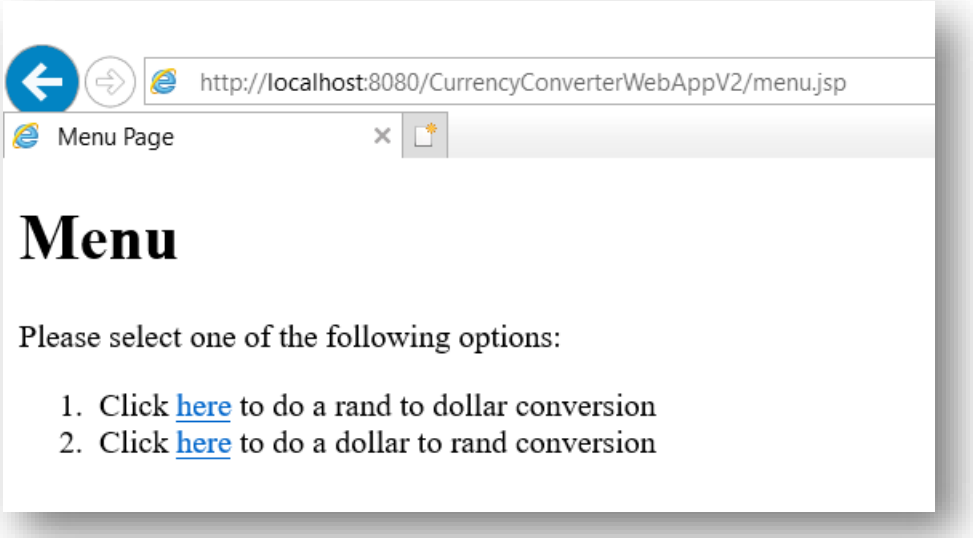


Part C - Run the web client project.

Launch the web app.

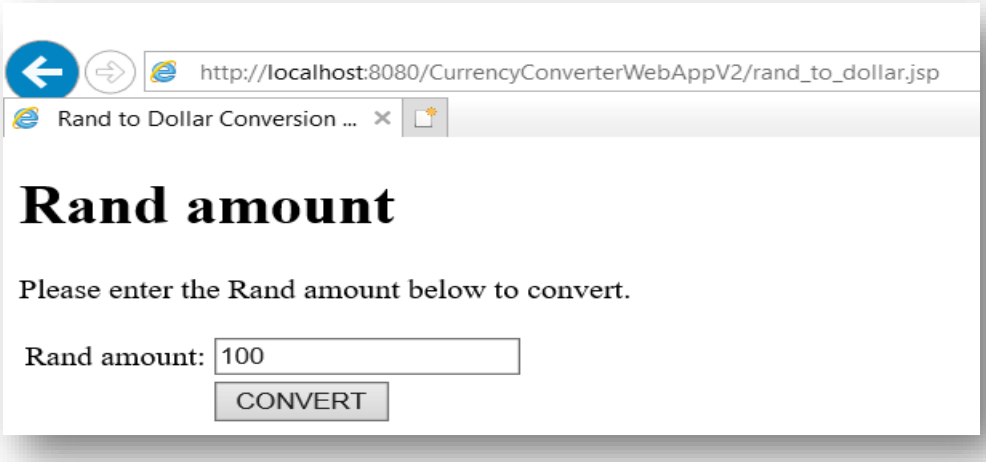


Click on the link.

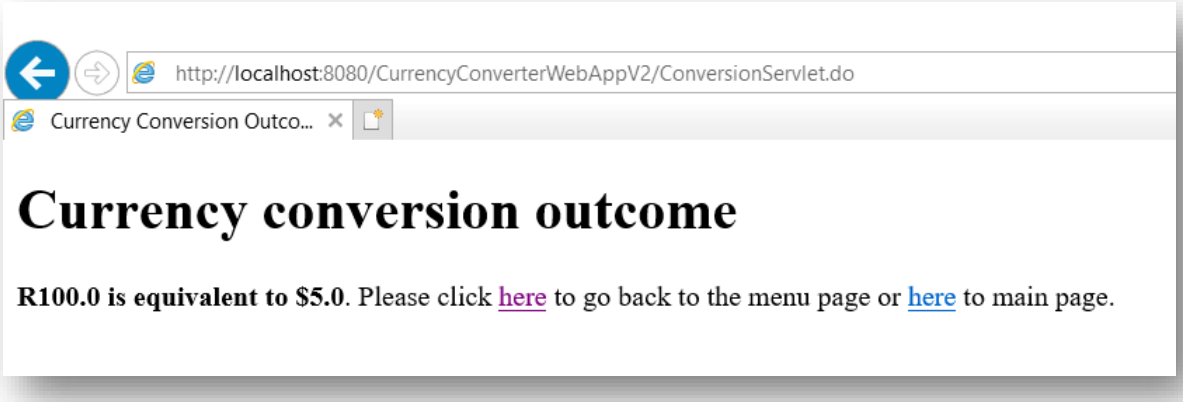


Click on the first link.

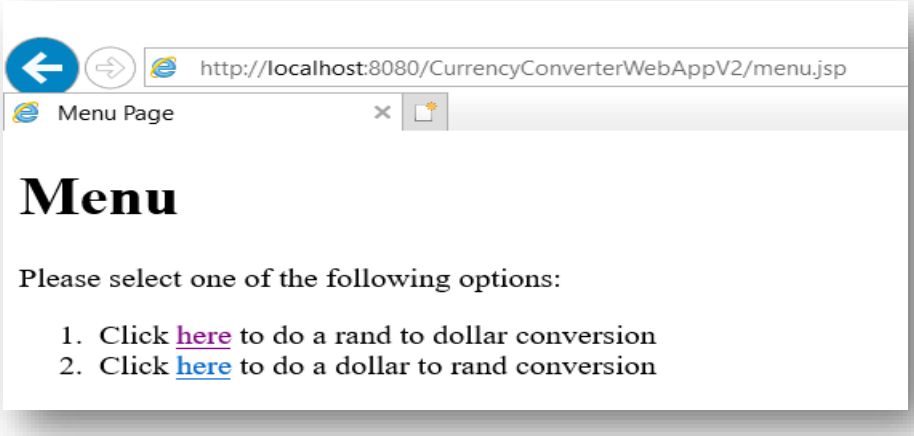
Enter the amount.



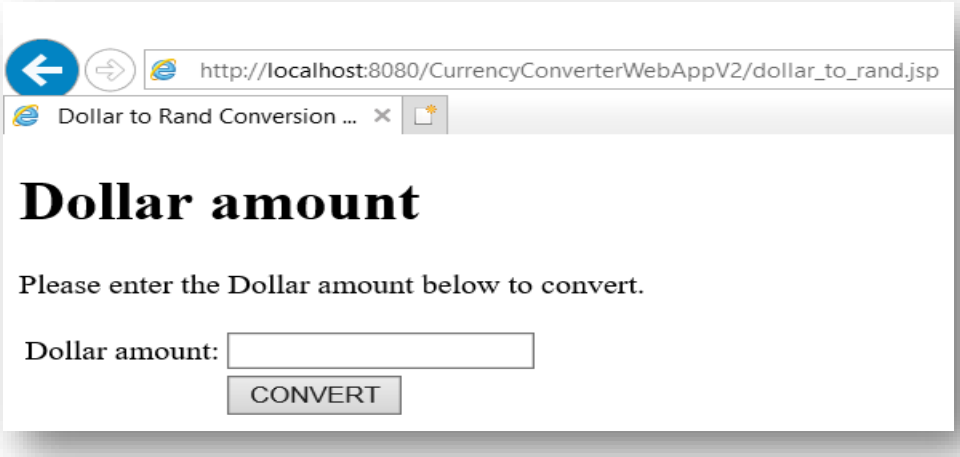
Click on the button.



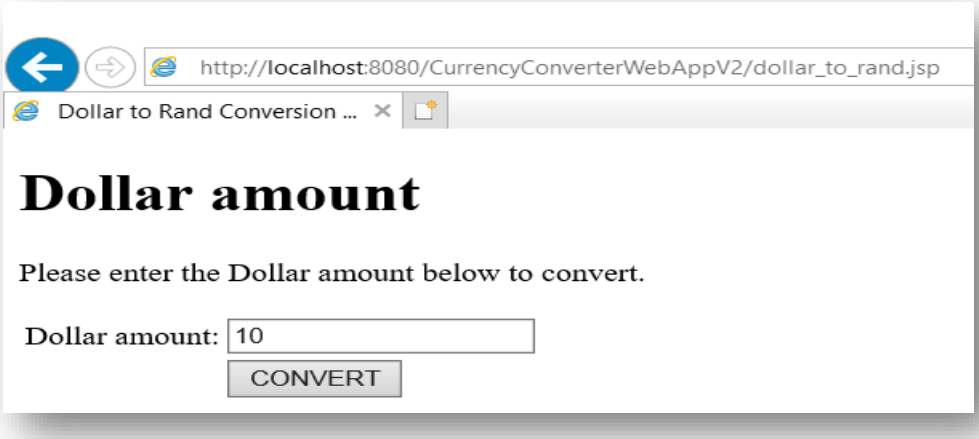
Go back to the menu page.



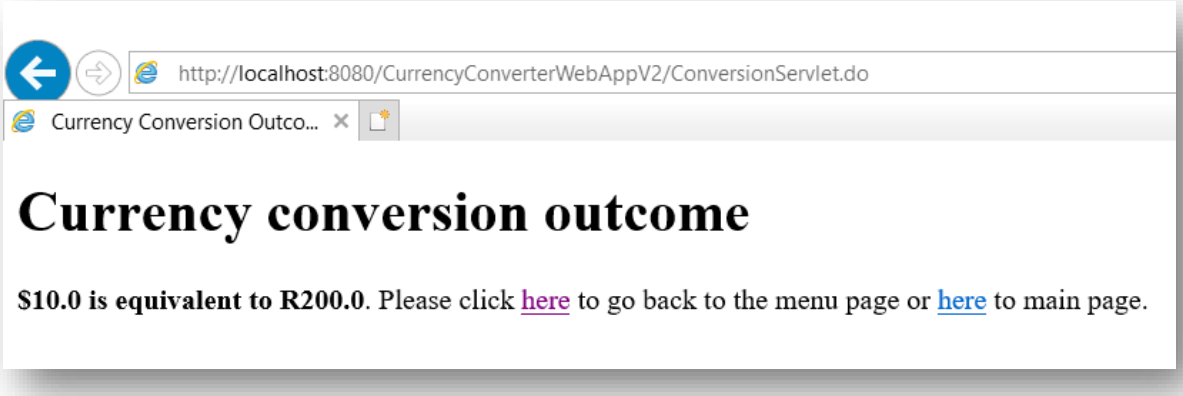
Click on the second link.



Enter the dollar amount



Click on the button.



1.2 DIY (Do It Yourself)

In this chapter we introduced you to Stateless Session Beans. We explained the concept and demonstrated how they are used in web applications. In this DIY, we want you to undertake two tasks in line with what you have learnt.

Task #1

Mujinga is an intern at a text analysis company called ***TextSpectacles***. The company specializes in analysing text messages for customers. Customers provide the company with text messages that consist of words, commas and periods. The customers are interested in gaining insights contained in the text messages. Specifically, they want to know the frequency of occurrence of words in a given text message.

As a new intern at ***TextSpectacles***, Mujinga is given the task of creating a web application that will help analyse text messages according to the stated requirements. So, given a text message, her application is expected to do the following:

- ✓ Determine and display the frequency of occurrence of each word in the sentence.

To do

Assuming that you are Mujinga, create such an application for ***TextSpectacles***. Use a **stateless** session bean to accomplish the task.

Things to note

Exceptions must be handled from the server side for the following situations:

- When a user clicks the submit button without providing a text message. Meaning an empty value is sent to the server.
- When the text message contains digits (numbers).

Task #2

Lerato is a programming intern at WeDoWebApps company. The company specializes in developing conversational web apps that are fail-safe (handle exceptions) and personalized for their clients. Mr Maluleke, a senior developer at WeDoWebApps, has been assigned the responsibility of mentoring Mulumba. As his first task, Mr Maluleke gives Mulumba an opportunity of creating a personalized, fail-safe conversational web app for a client. The client is Ms Skosana, a teacher by profession.

Ms Skosana wants a personalized web application that will allow students to take a test. Ms Skosana has already prepared a test for the students. It has seven (7) multiple choice questions with corresponding answers. The table below shows the test.

No.	Question	Answer
1.	1 + 1 = ? A. 1 B. 11 C. 2 D. 0	C
2.	1 * 1 = ? A. 1 B. 11 C. 2 D. 0	A
3.	1 / 1 = ? A. 1 B. 11 C. 2 D. 0	A
4.	1 - 1 = ? A. 1 B. 11	D

	C. 2 D. 0	
5.	1 % 1 = ? A. 1 B. 11 C. 2 D. 0	D
6.	(1 + 1) *2 = ? A. 1 B. 2 C. 4 D. 6	C
7.	(1 + 1) / 2 = ? A. 1 B. 2 C. 4 D. 6	A

Ms Skosana wants the web application to randomly select five (5) questions from the test and give them to a student for answering. The student must be given one question at a time to answer. After all the questions have been answered, the web application is required to do two things, namely:

- Mark the work of the student.
- Display a summary report. The report should include the following information:

✓ The name of the student.
✓ The number of questions asked.
✓ The number of correct answers.
✓ The percentage mark obtained.
✓ The list of questions asked.
✓ The correct answer for each question.
✓ The answers provided by the student for each question.
✓ The marking outcome for each student answer ("**Correct**" or "**Wrong**").

<u>To do</u>

Assuming that you are Mulumba, create such a web application for Ms Skosana. Use Staless Session Beans.

Things to note

Exceptions must be handled from the server side for the following situations:

- When a student clicks the submit button without providing an answer. Meaning an empty value is sent to the server.
- When a student provides a letter other than A, B, C or D as an answer to a question.

1.3 Conclusion

In this chapter we managed to introduce the student to EJBs. In the next chapter we discuss another type of EJBs, Stateful Session Beans.

Thank you very much for having taken time to go through this chapter. Enjoy the rest of the day and God bless you.