

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 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

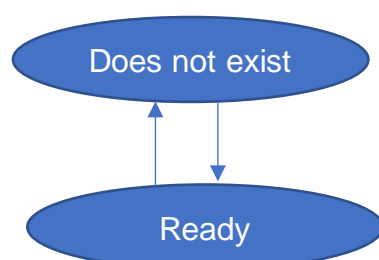
```
1
2
3 @Stateless
4 public class PropertyManagerSB implements PropertyManagerInterface{
5     public String determineLowestIncomeInfo(String areaCode) {
6         //code
7     }
8
9     public String determineHighestIncomeInfo(String monthCode) {
10        //code
11    }
12 }
```

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 dependency 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.

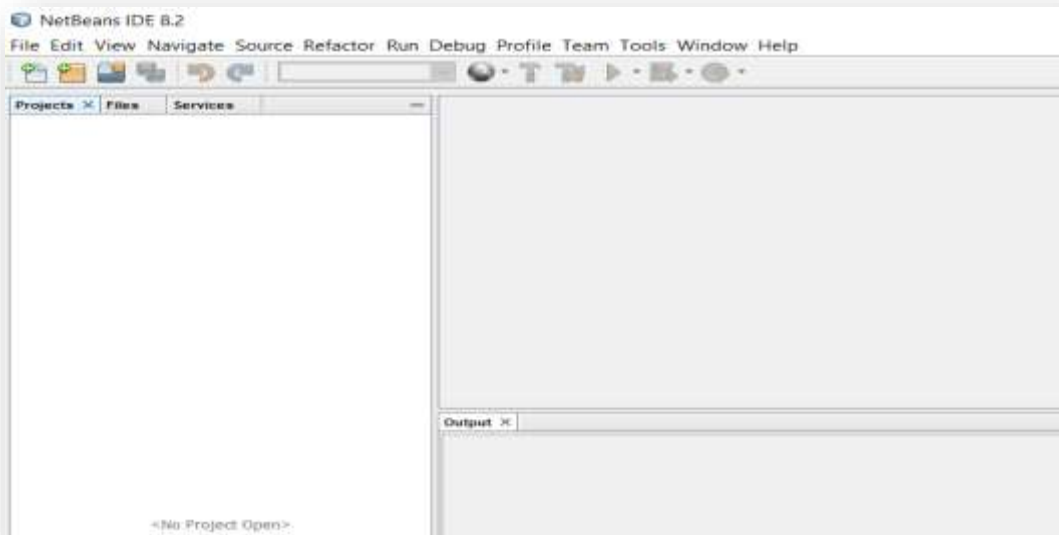
In **Part B** we will create a **web client** project to the **EJB module**. The project will mainly consist of two servlets which will serve as clients to the EJB module. The first servlet will consume/use/invoke the **dollar to rand** conversion method of the EJB, and

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.

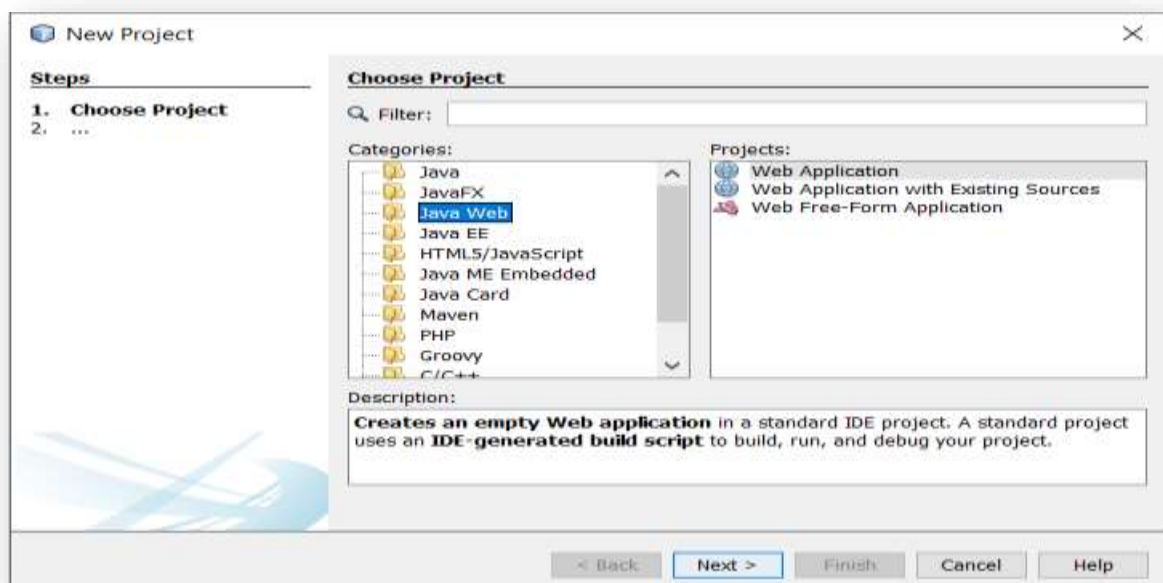
Part A - Create an EJB module project.

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

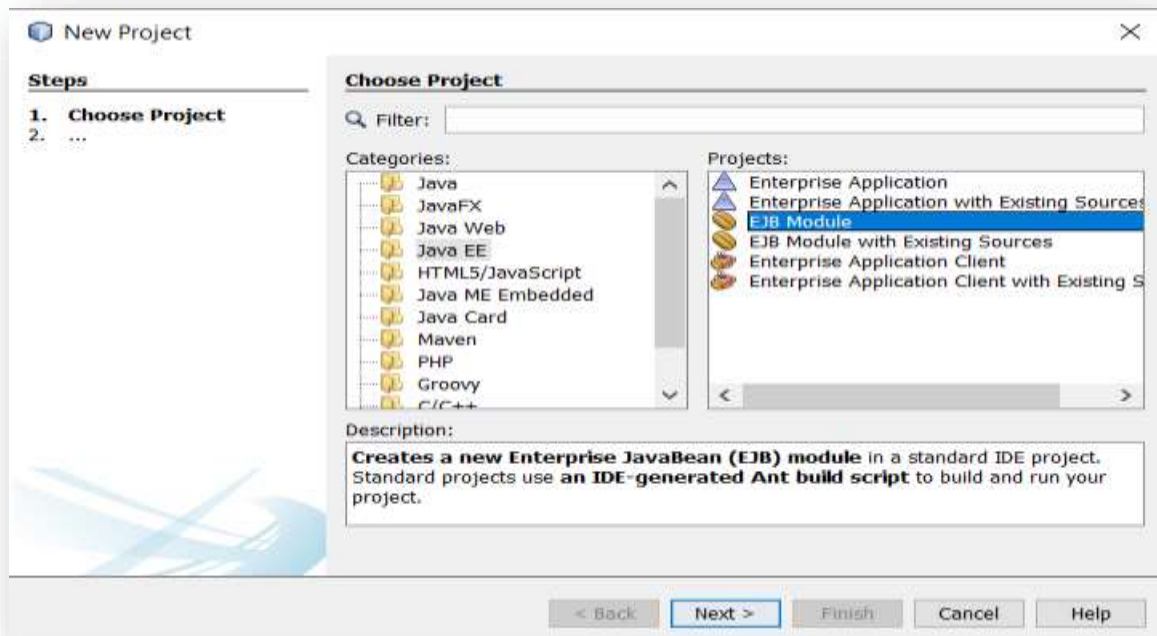
1. Launch NetBeans.



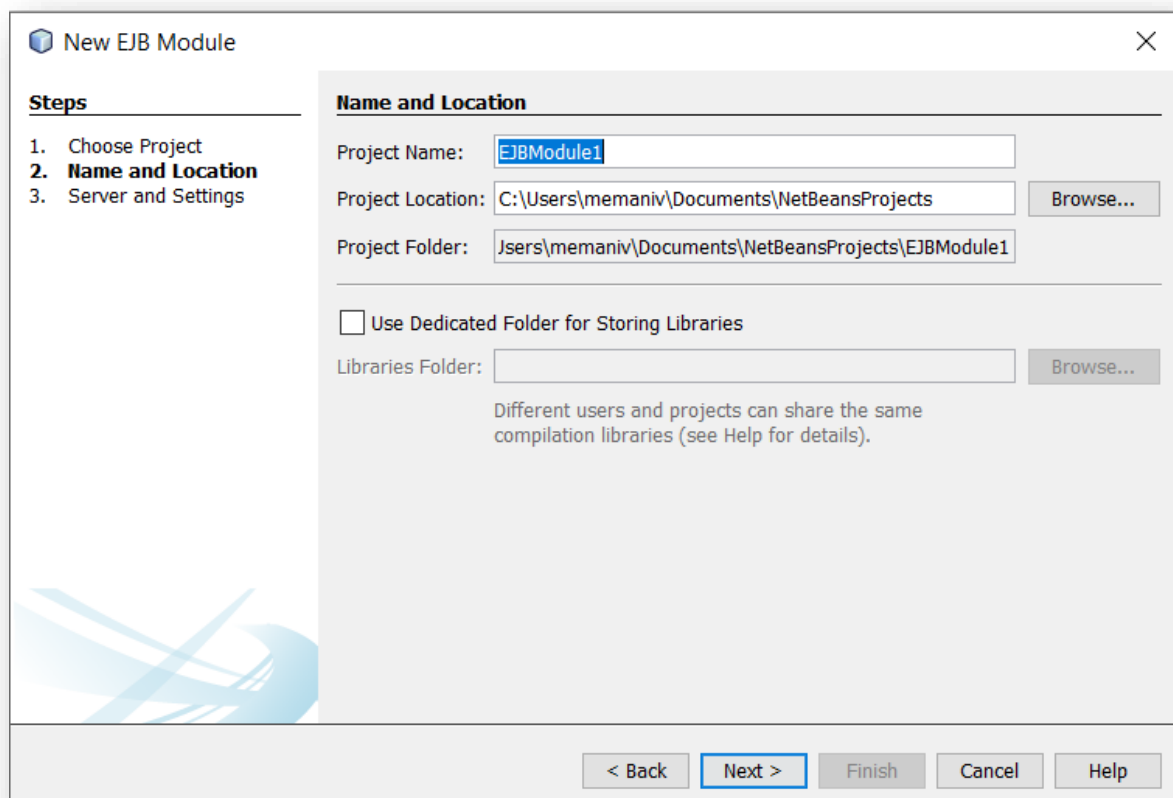
2. Click on **File | New Project**.



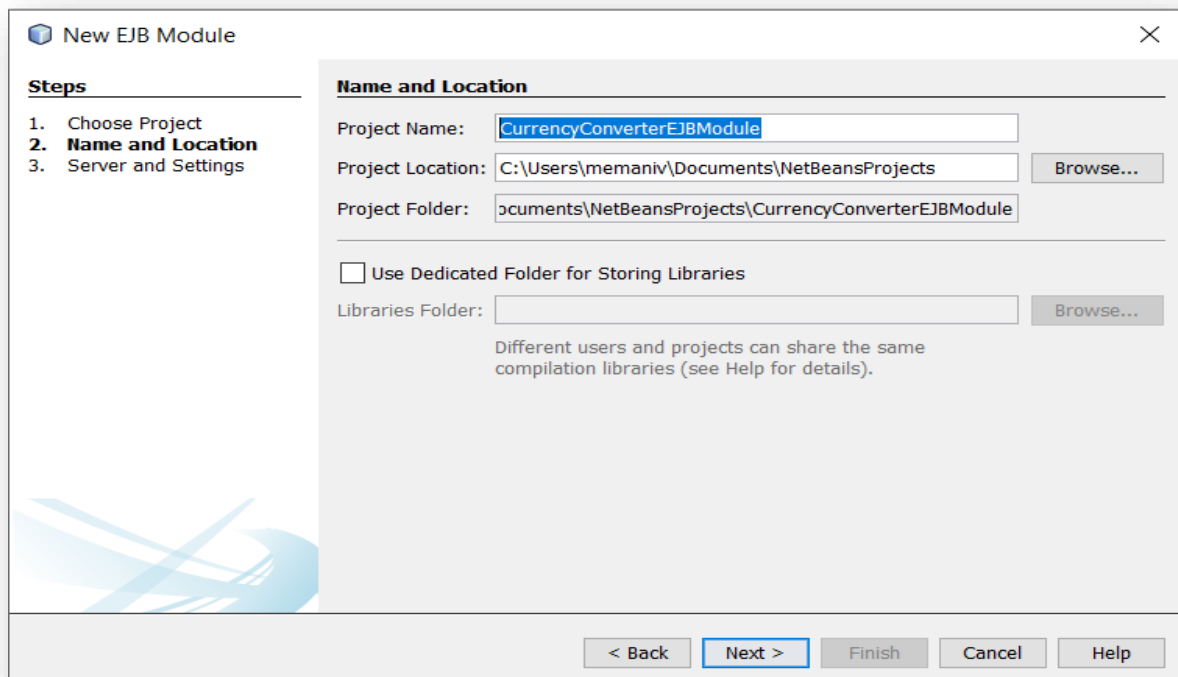
3. Select **Java EE** under **Categories** and **EJB Module** under **Projects**.



4. Click **Next**.



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



The 'New EJB Module' dialog box is shown with the 'Name and Location' tab selected. The 'Steps' list on the left indicates the current step is '2. Name and Location'. The 'Project Name' field is filled with 'CurrencyConverterEJBModule'. The 'Project Location' field shows 'C:\Users\memaniv\Documents\NetBeansProjects' with a 'Browse...' button. The 'Project Folder' field shows 'Documents\NetBeansProjects\CurrencyConverterEJBModule'. There is an unchecked checkbox for 'Use Dedicated Folder for Storing Libraries' and a 'Libraries Folder' field with a 'Browse...' button. A note states: 'Different users and projects can share the same compilation libraries (see Help for details)'. At the bottom, the 'Next >' button is highlighted.

New EJB Module

Steps

1. Choose Project
2. **Name and Location**
3. Server and Settings

Name and Location

Project Name: CurrencyConverterEJBModule

Project Location: C:\Users\memaniv\Documents\NetBeansProjects Browse...

Project Folder: Documents\NetBeansProjects\CurrencyConverterEJBModule

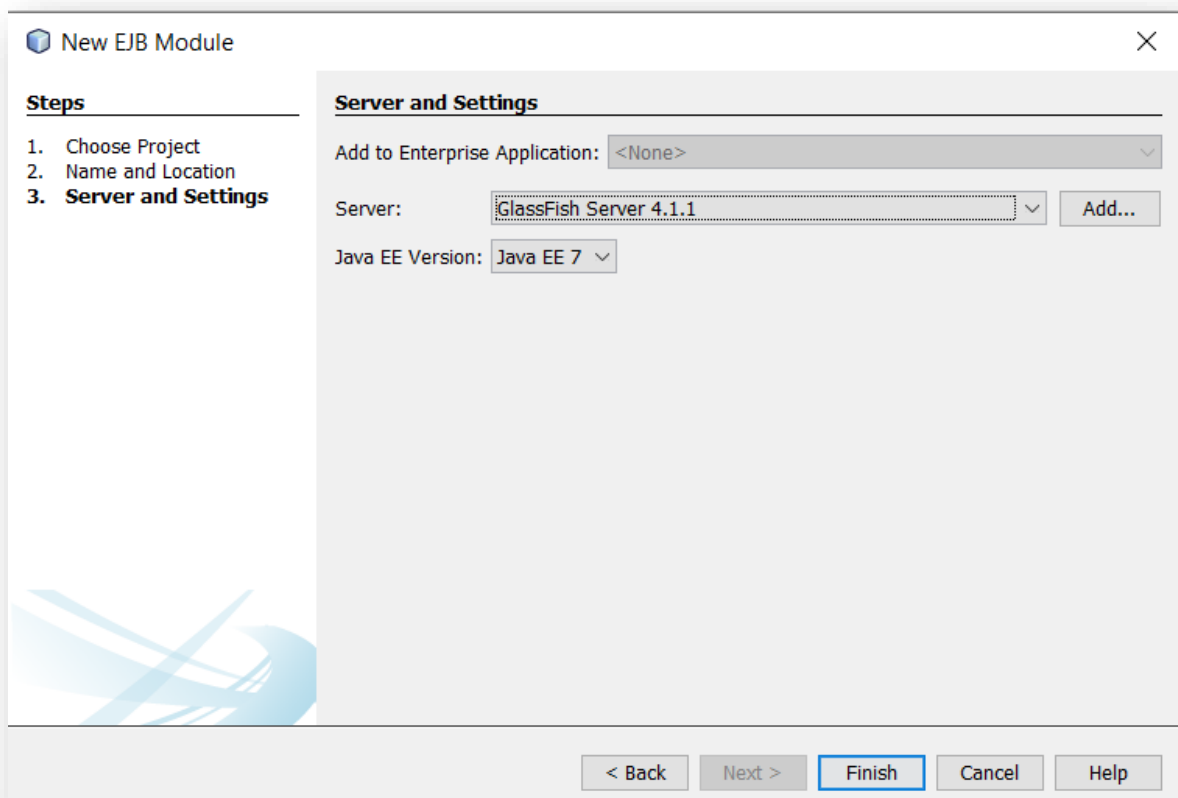
☐ Use Dedicated Folder for Storing Libraries

Libraries Folder: Browse...

Different users and projects can share the same compilation libraries (see Help for details).

< Back Next > Finish Cancel Help

6. Click **Next**.



The 'New EJB Module' dialog box is shown with the 'Server and Settings' tab selected. The 'Steps' list on the left indicates the current step is '3. Server and Settings'. The 'Add to Enterprise Application' dropdown is set to '<None>'. The 'Server' dropdown is set to 'GlassFish Server 4.1.1' with an 'Add...' button. The 'Java EE Version' dropdown is set to 'Java EE 7'. At the bottom, the 'Finish' button is highlighted.

New EJB Module

Steps

1. Choose Project
2. Name and Location
3. **Server and Settings**

Server and Settings

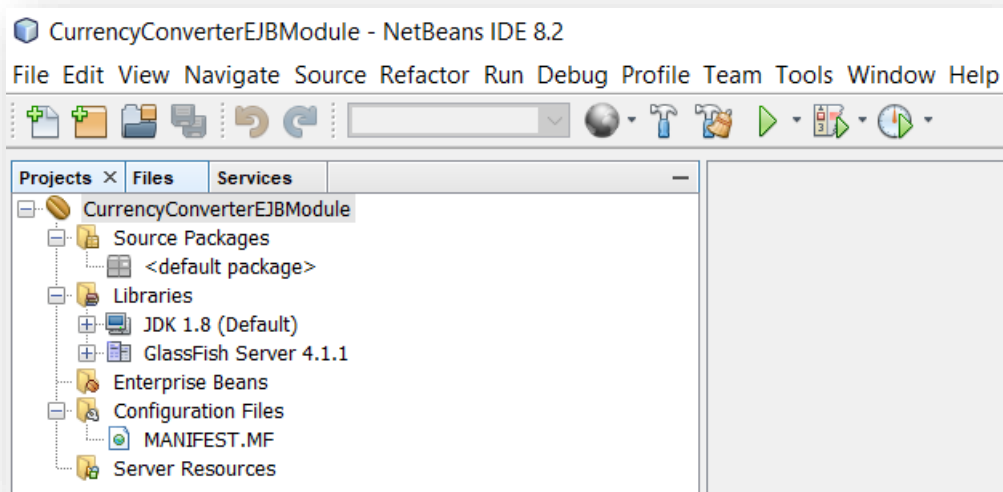
Add to Enterprise Application: <None>

Server: GlassFish Server 4.1.1 Add...

Java EE Version: Java EE 7

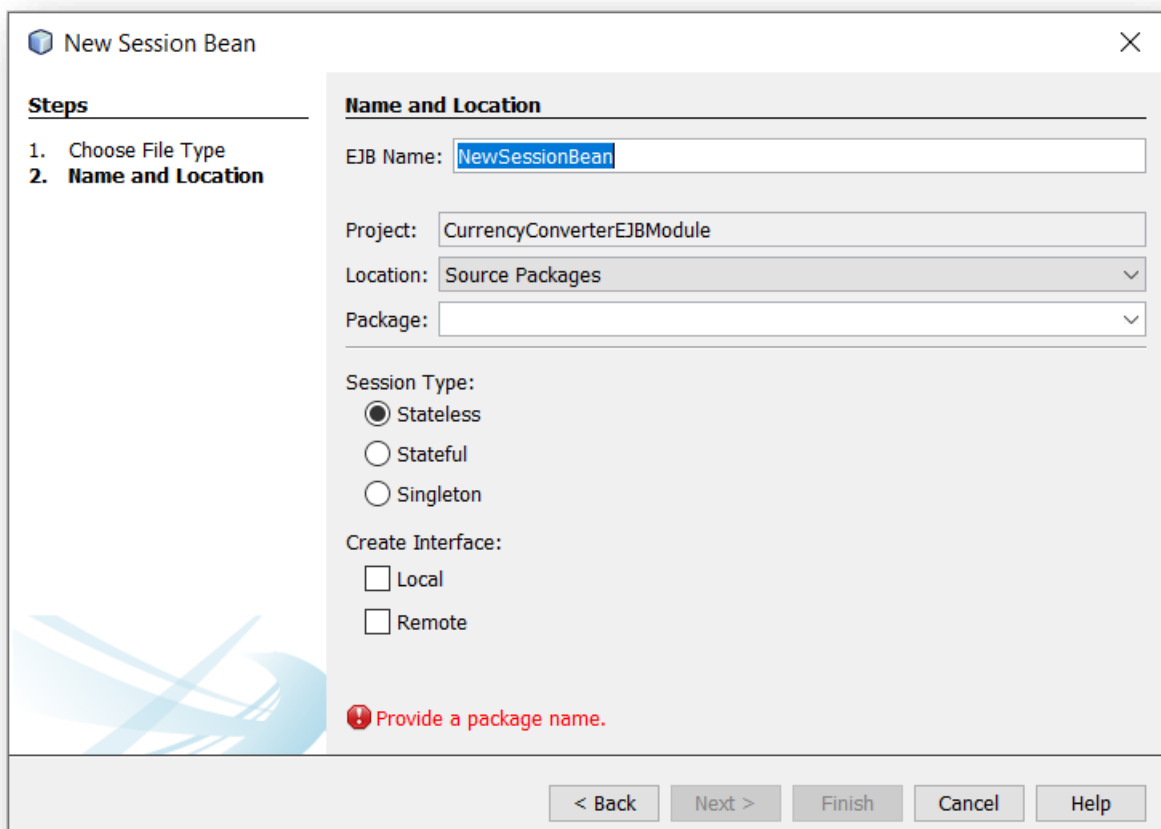
< Back Next > Finish Cancel Help

7. Click **Finish**.

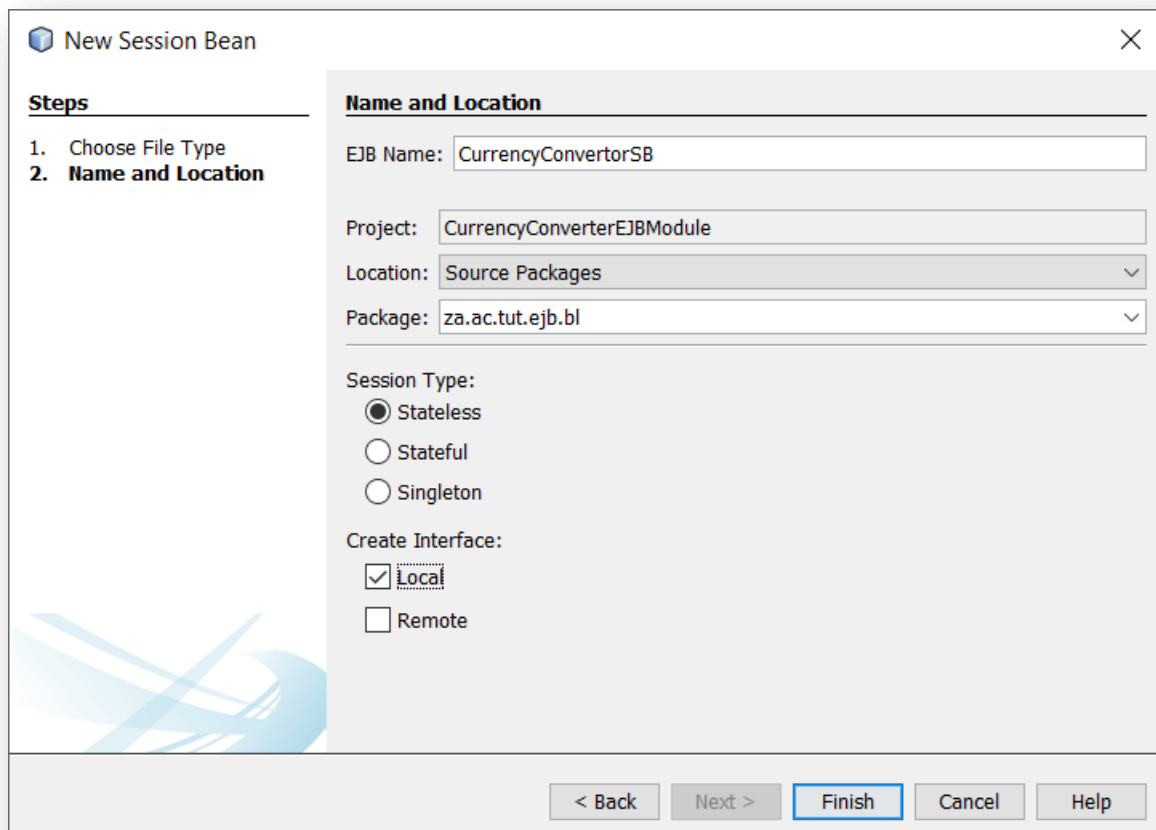


8. Create a stateless session bean. Perform the following steps:

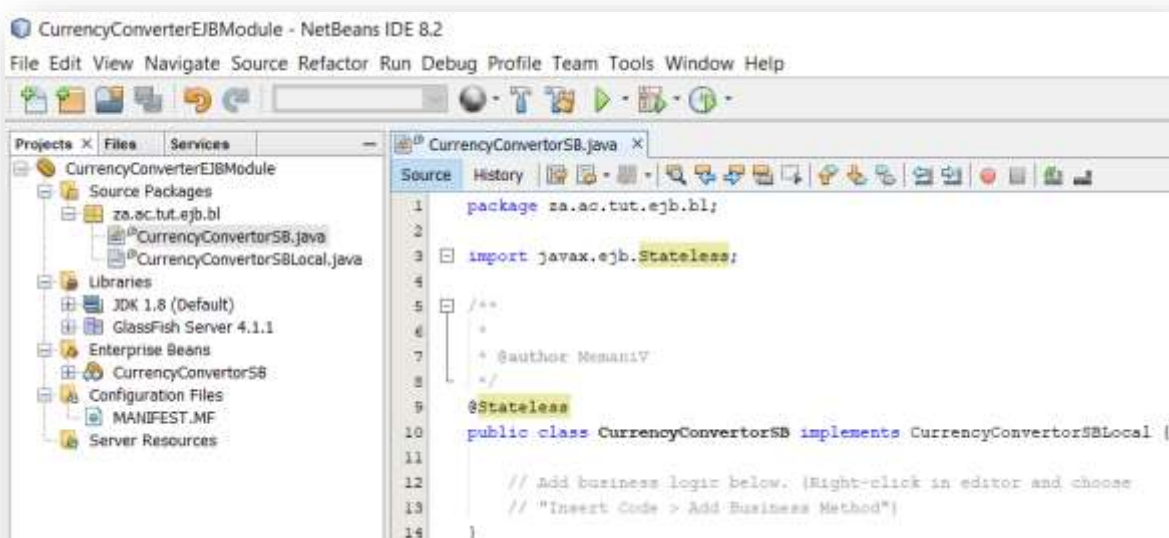
8.1 Right click on the project and select **New | Session Bean**.



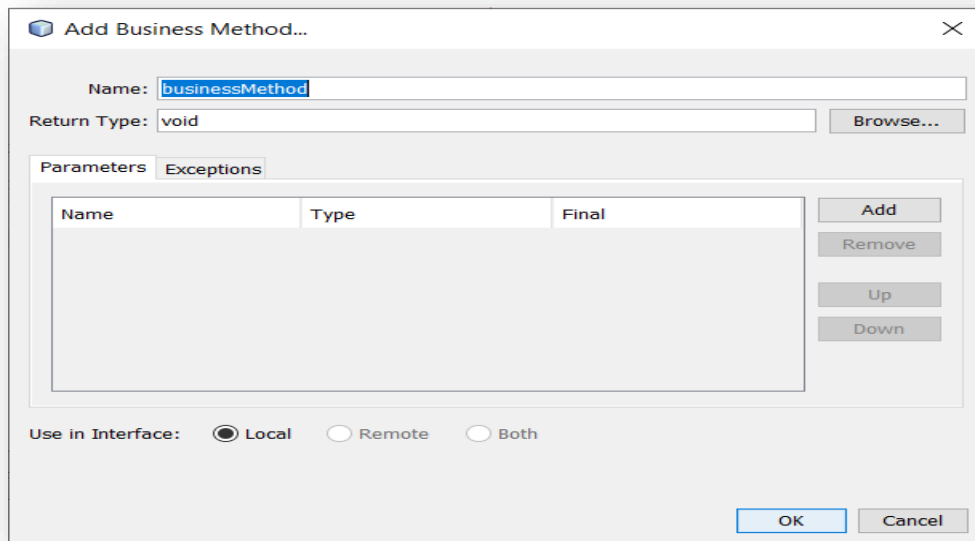
8.2 Name the EJB as **CurrencyConvertorSB**, package it under **za.ac.tut.ejb.bl**, select **Stateless** under **Session Type**, and select the **Local** interface. The **bl** acronym stands for **business logic**.



8.3 Click **Finish**.



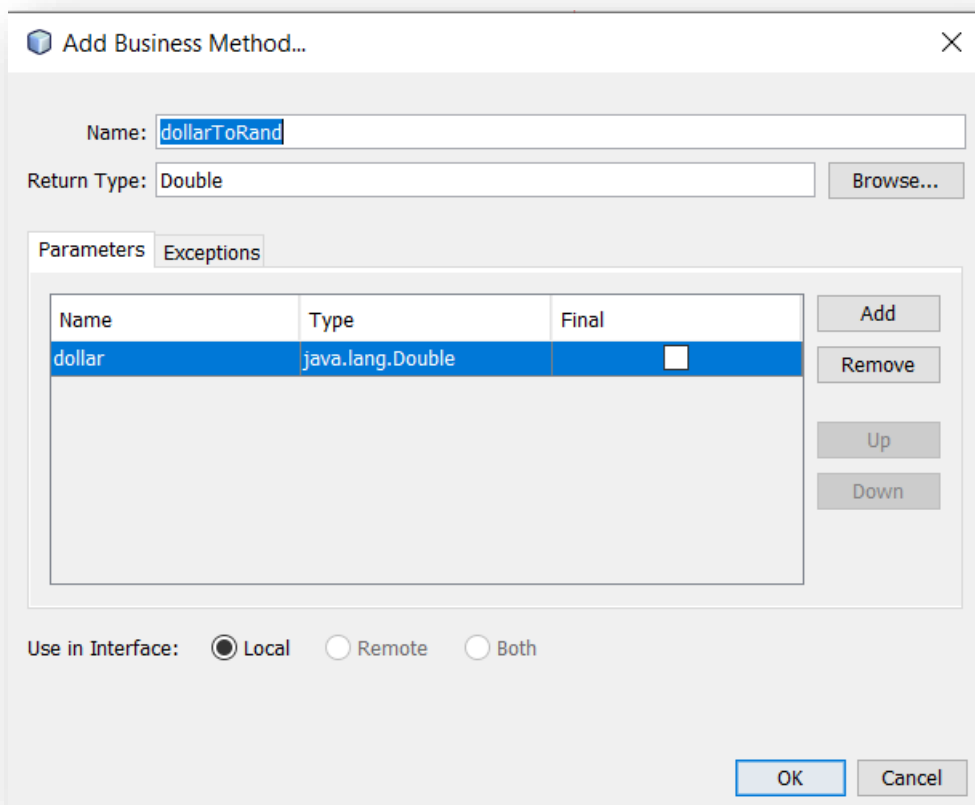
8.4 Right click inside the stateless session bean and select **Insert Code | Add Business Method**.



The 'Add Business Method...' dialog box is shown. The 'Name' field contains 'businessMethod'. The 'Return Type' is set to 'void'. The 'Parameters' tab is selected, and the table below is empty. The 'Use in Interface' section has 'Local' selected. The 'OK' button is highlighted.

Name	Type	Final
------	------	-------

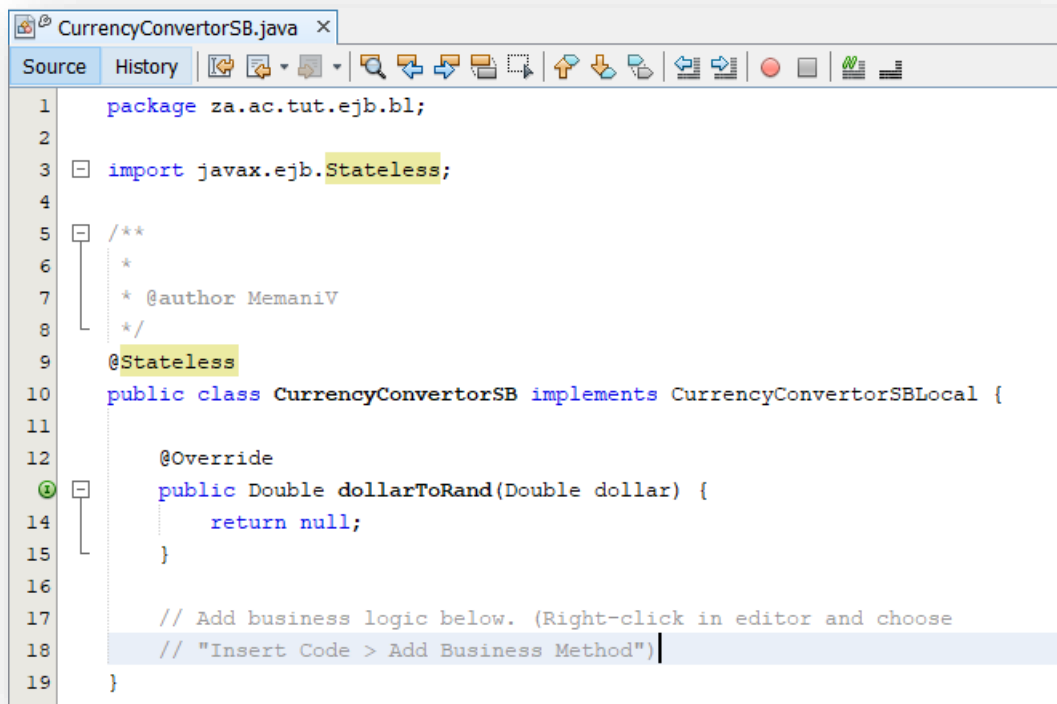
8.5 Name the method as **dollarToRand**, which returns a **Double**. Click on the **Add** button to add the parameters.



The 'Add Business Method...' dialog box is shown. The 'Name' field contains 'dollarToRand'. The 'Return Type' is set to 'Double'. The 'Parameters' tab is selected, and the table below has one row: 'dollar' with type 'java.lang.Double' and 'Final' checkbox unchecked. The 'Use in Interface' section has 'Local' selected. The 'OK' button is highlighted.

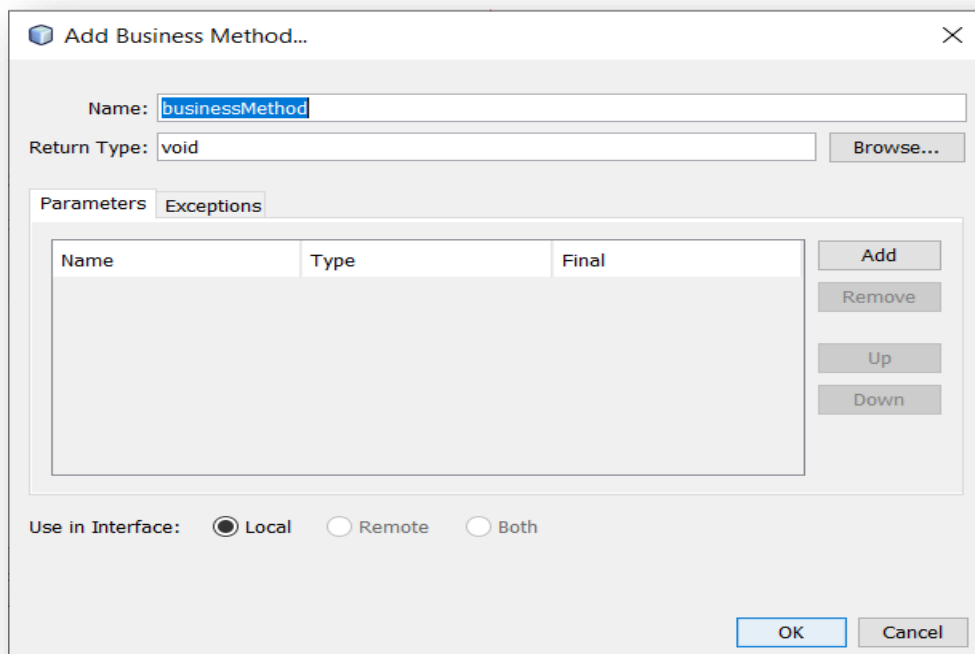
Name	Type	Final
dollar	java.lang.Double	<input type="checkbox"/>

8.6 Click **OK**.



```
1 package za.ac.tut.ejb.bl;
2
3 import javax.ejb.Stateless;
4
5 /**
6  *
7  * @author MemaniV
8  */
9 @Stateless
10 public class CurrencyConvertorSB implements CurrencyConvertorSBLocal {
11
12     @Override
13     public Double dollarToRand(Double dollar) {
14         return null;
15     }
16
17     // Add business logic below. (Right-click in editor and choose
18     // "Insert Code > Add Business Method")
19 }
```

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



Add Business Method...

Name:

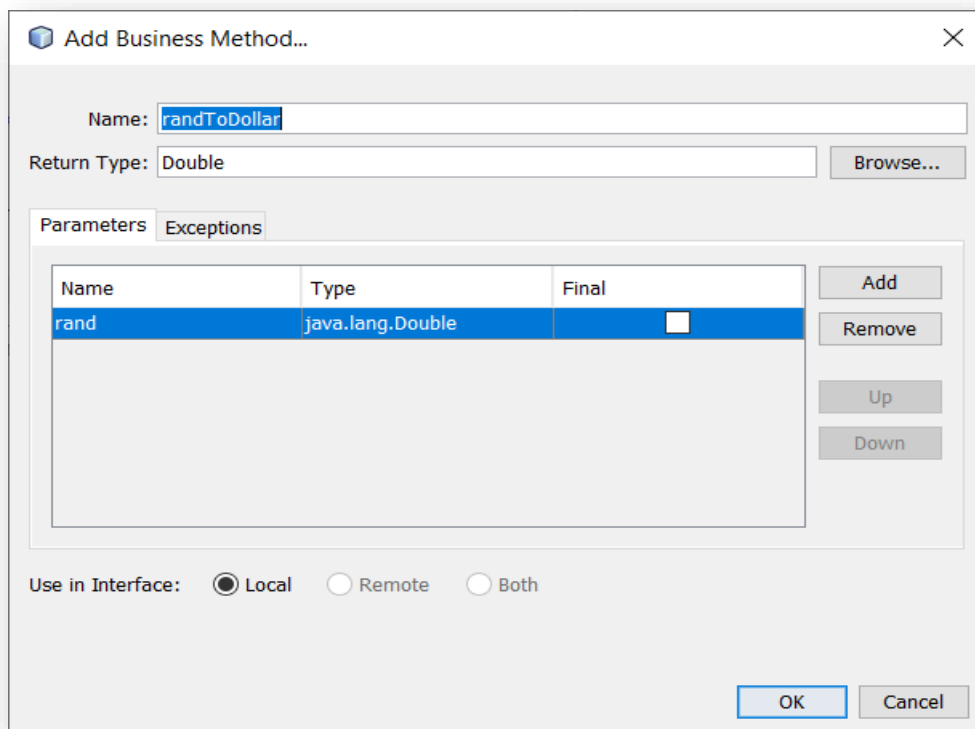
Return Type:

Parameters ☒ Exceptions ☐

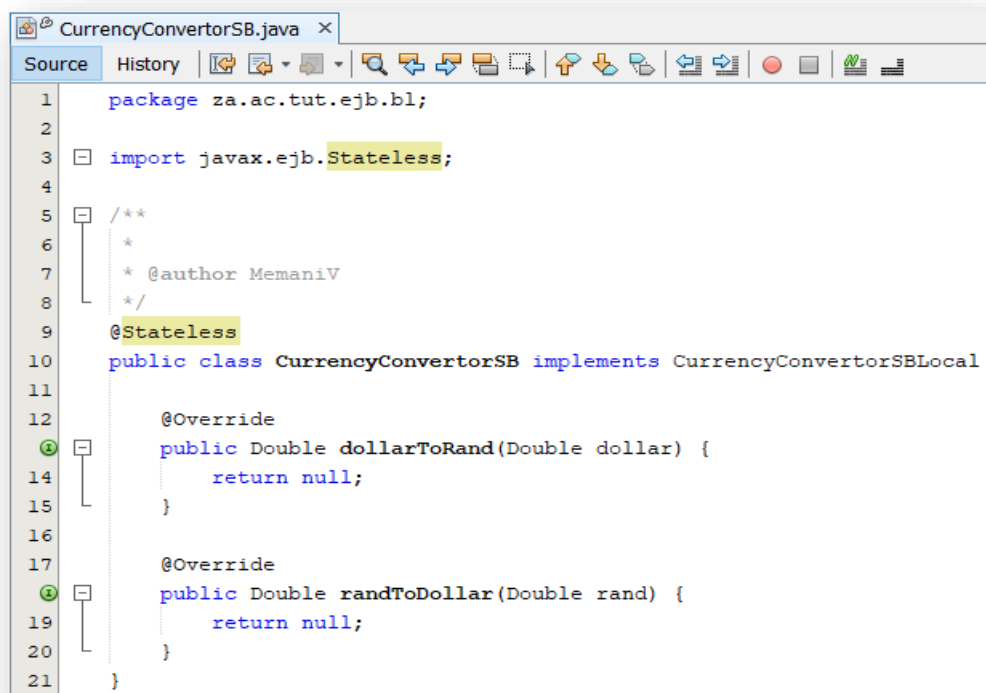
Name	Type	Final
------	------	-------

Use in Interface: ☒ Local ☐ Remote ☐ Both

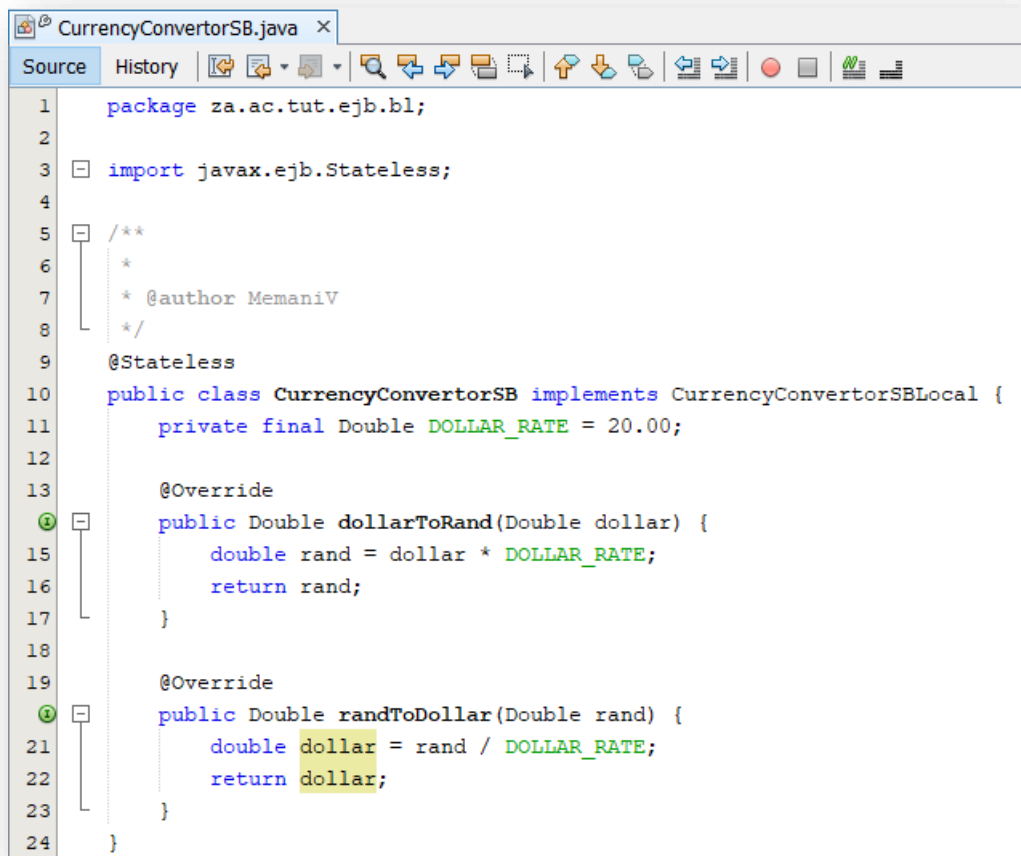
8.8 Name the method as **randToDollar**, which returns a **Double**. Click on the **Add** button to add the parameters.



8.9 Click **OK**.

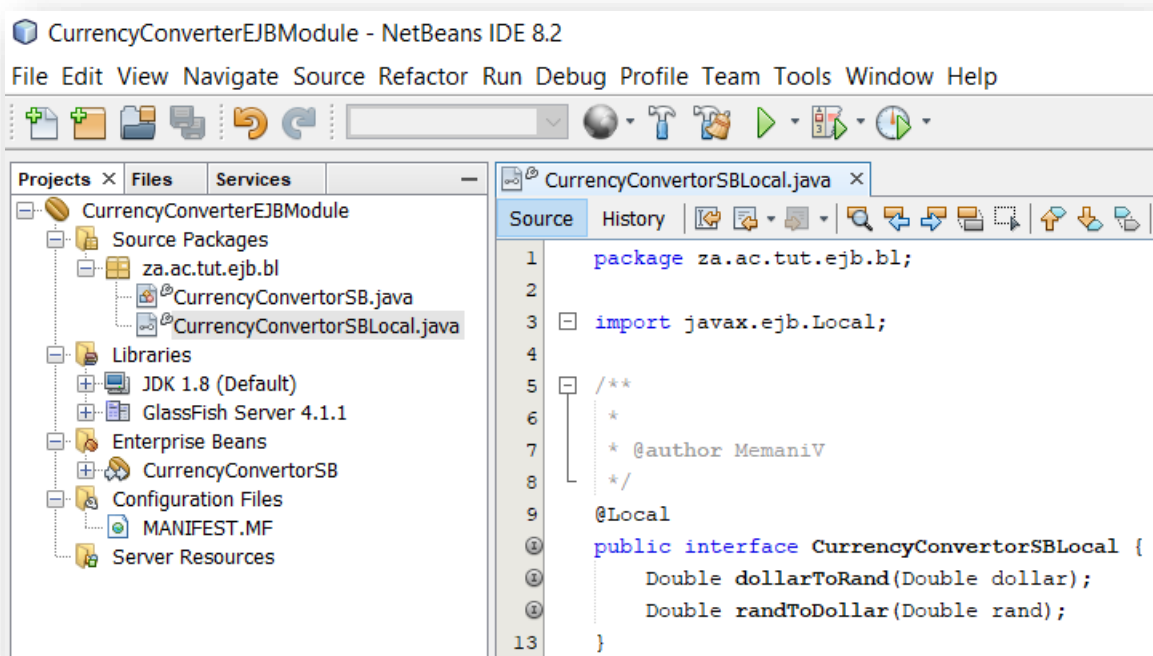


8.10 Complete the source code of the Stateless Session Bean.



```
1 package za.ac.tut.ejb.bl;
2
3 import javax.ejb.Stateless;
4
5 /**
6  *
7  * @author MemaniV
8  */
9 @Stateless
10 public class CurrencyConvertorSB implements CurrencyConvertorSBLocal {
11     private final Double DOLLAR_RATE = 20.00;
12
13     @Override
14     public Double dollarToRand(Double dollar) {
15         double rand = dollar * DOLLAR_RATE;
16         return rand;
17     }
18
19     @Override
20     public Double randToDollar(Double rand) {
21         double dollar = rand / DOLLAR_RATE;
22         return dollar;
23     }
24 }
```

8.11 View the interface.



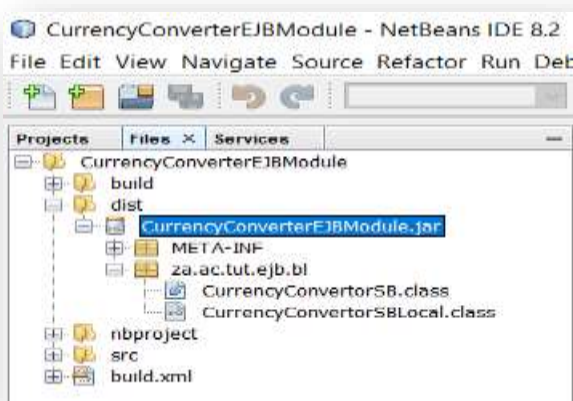
```
1 package za.ac.tut.ejb.bl;
2
3 import javax.ejb.Local;
4
5 /**
6  *
7  * @author MemaniV
8  */
9 @Local
10 public interface CurrencyConvertorSBLocal {
11     Double dollarToRand(Double dollar);
12     Double randToDollar(Double rand);
13 }
```

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

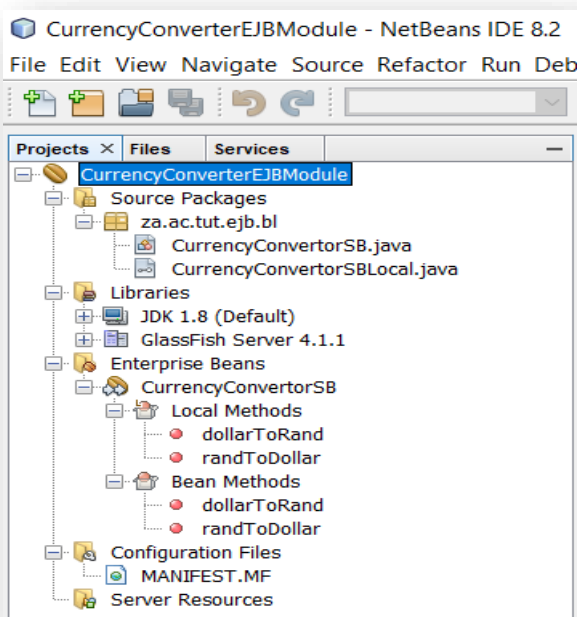
9.1 Right click on the project and select **Clean and Build**.



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



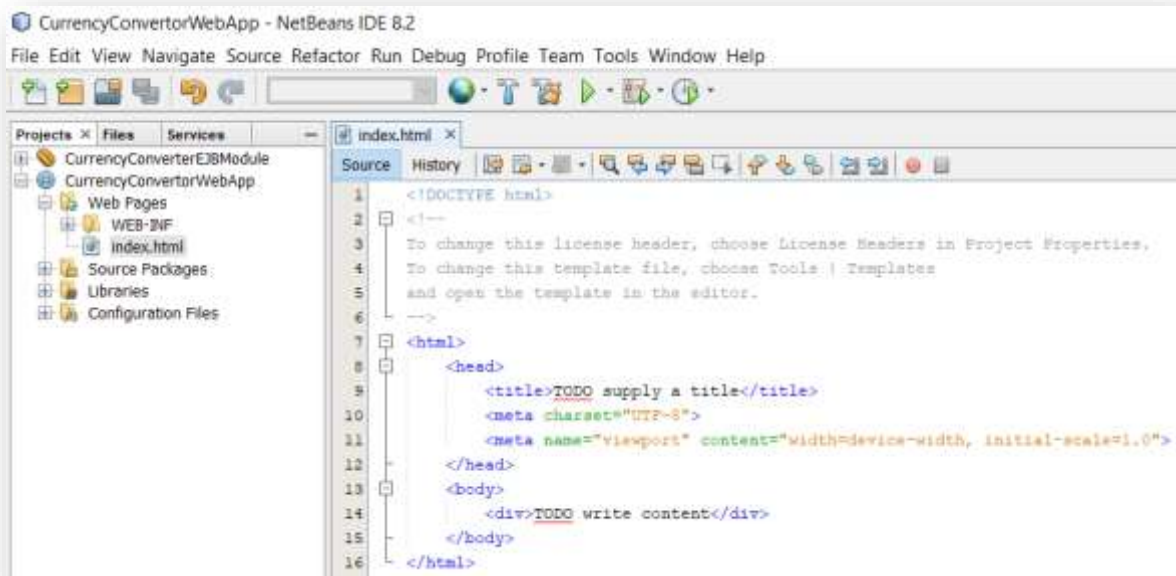
9.3 View the complete EJB project structure.



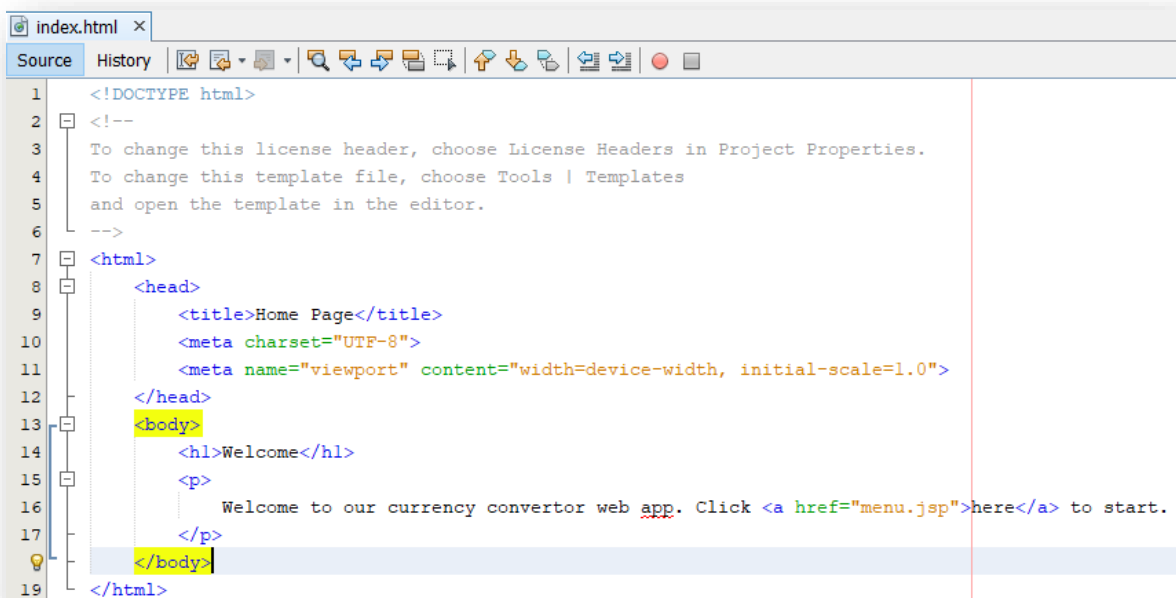
Part B - Create a Web client project

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.



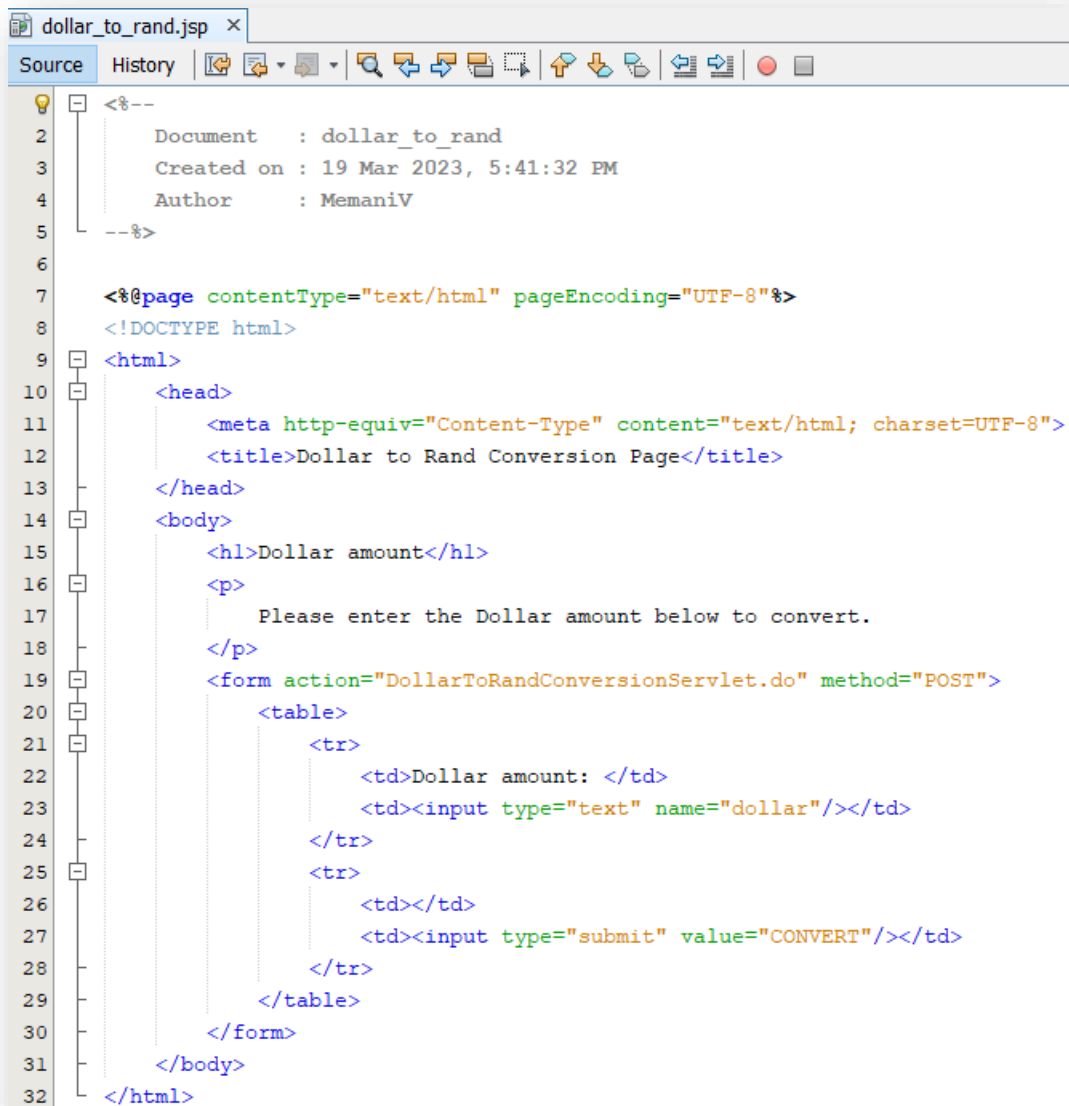
3. Create the **menu.jsp** file.

```
menu.jsp x
Source History
<!--
2      Document    : menu
3      Created on   : 19 Mar 2023, 5:31:00 PM
4      Author      : MemaniV
5  -->
6
7  <%@page contentType="text/html" pageEncoding="UTF-8"%>
8  <!DOCTYPE html>
9  <html>
10     <head>
11         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12         <title>Menu Page</title>
13     </head>
14     <body>
15         <h1>Menu</h1>
16         <p>
17             Please select one of the following options:
18         </p>
19         <ol>
20             <li>Click <a href="rand_to_dollar.jsp">here</a> to do a rand to dollar conversion</li>
21             <li>Click <a href="dollar_to_rand.jsp">here</a> to do a dollar to rand conversion</li>
22         </ol>
23     </body>
24 </html>
```

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

```
rand_to_dollar.jsp x
Source History
<!--
2      Document    : rand to dollar
3      Created on   : 19 Mar 2023, 5:35:26 PM
4      Author      : MemaniV
5  -->
6
7  <%@page contentType="text/html" pageEncoding="UTF-8"%>
8  <!DOCTYPE html>
9  <html>
10     <head>
11         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12         <title>Rand to Dollar Conversion Page</title>
13     </head>
14     <body>
15         <h1>Rand amount</h1>
16         <p>
17             Please enter the Rand amount below to convert.
18         </p>
19         <form action="RandToDollarConversionServlet.do" method="POST">
20             <table>
21                 <tr>
22                     <td>Rand amount: </td>
23                     <td><input type="text" name="rand"/></td>
24                 </tr>
25                 <tr>
26                     <td></td>
27                     <td><input type="submit" value="CONVERT"/></td>
28                 </tr>
29             </table>
30         </form>
31     </body>
32 </html>
```

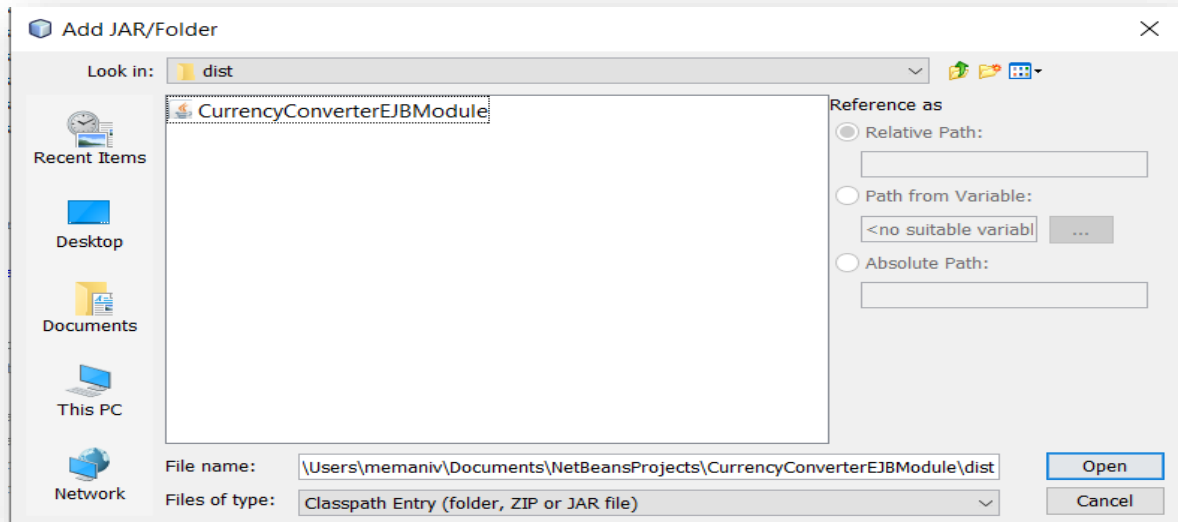
5. Create the **dollar_to_rand.jsp** file.



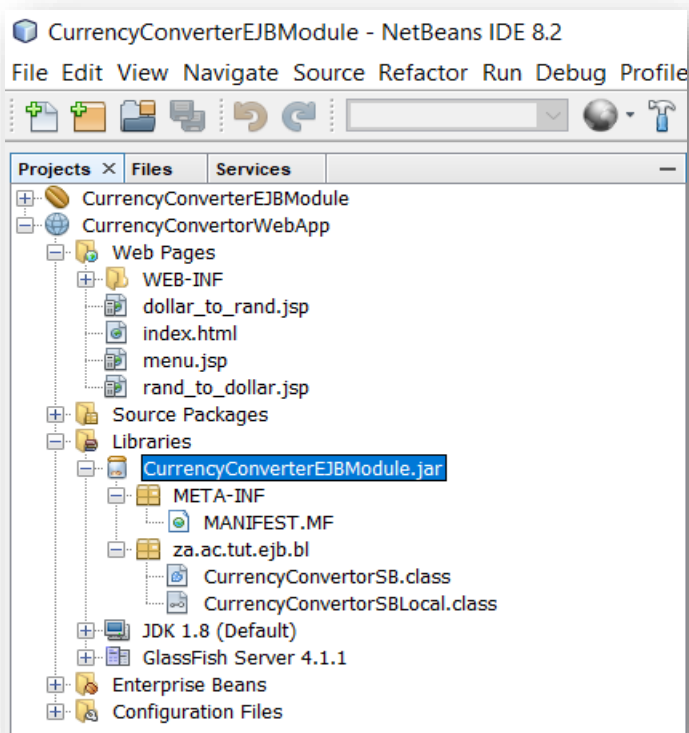
```
dollar_to_rand.jsp x
Source History
<%--
2      Document    : dollar_to_rand
3      Created on  : 19 Mar 2023, 5:41:32 PM
4      Author     : MemaniV
5  --%>
6
7  <%@page contentType="text/html" pageEncoding="UTF-8"%>
8  <!DOCTYPE html>
9  <html>
10     <head>
11         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12         <title>Dollar to Rand Conversion Page</title>
13     </head>
14     <body>
15         <h1>Dollar amount</h1>
16         <p>
17             Please enter the Dollar amount below to convert.
18         </p>
19         <form action="DollarToRandConversionServlet.do" method="POST">
20             <table>
21                 <tr>
22                     <td>Dollar amount: </td>
23                     <td><input type="text" name="dollar"/></td>
24                 </tr>
25                 <tr>
26                     <td></td>
27                     <td><input type="submit" value="CONVERT"/></td>
28                 </tr>
29             </table>
30         </form>
31     </body>
32 </html>
```


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

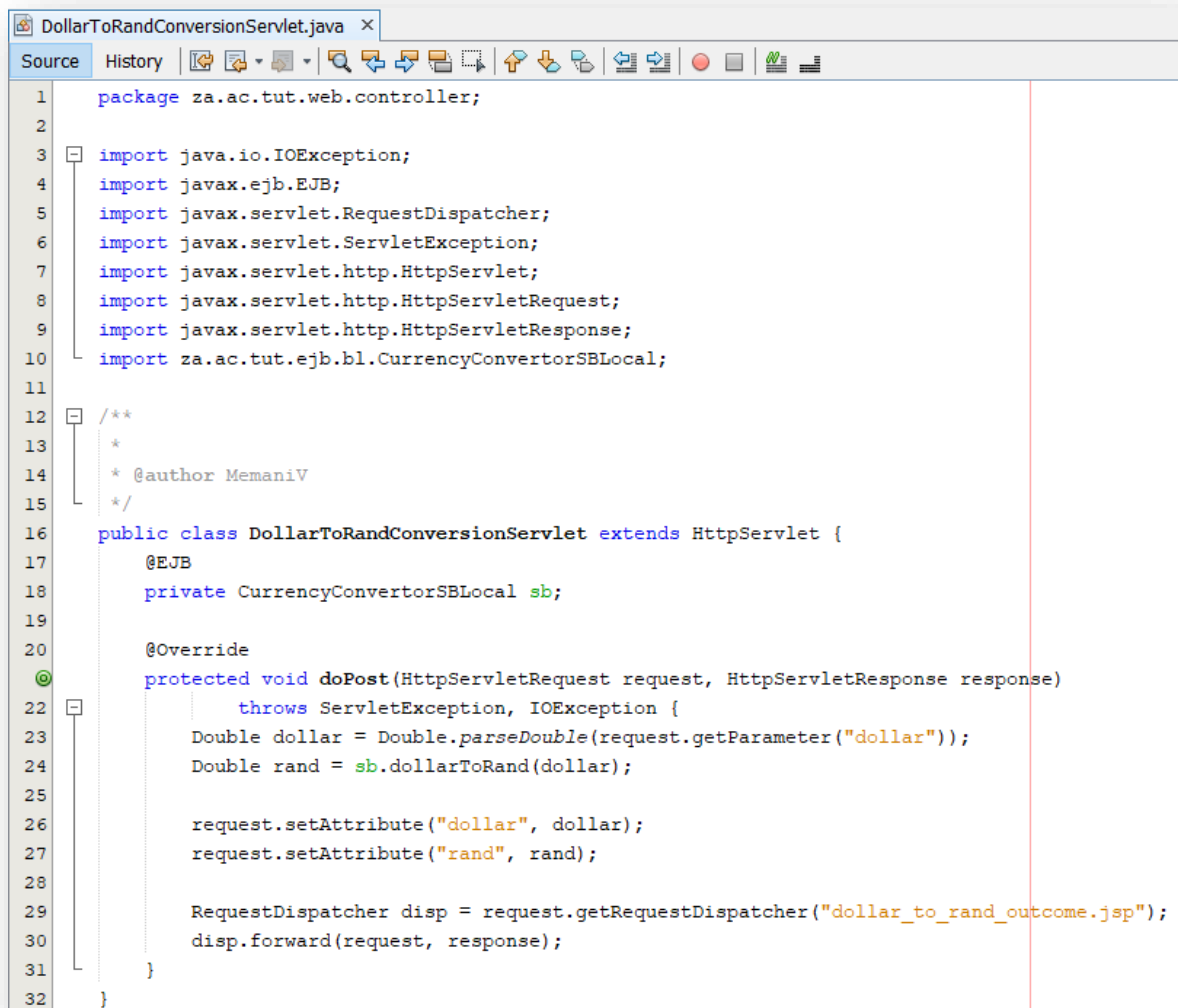
6.1 Right-click on the **Libraries** folder of the Web App project and select **Add JAR/Folder**. Navigate to the **dist** folder of **CurrencyConverterEJBModule**.



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

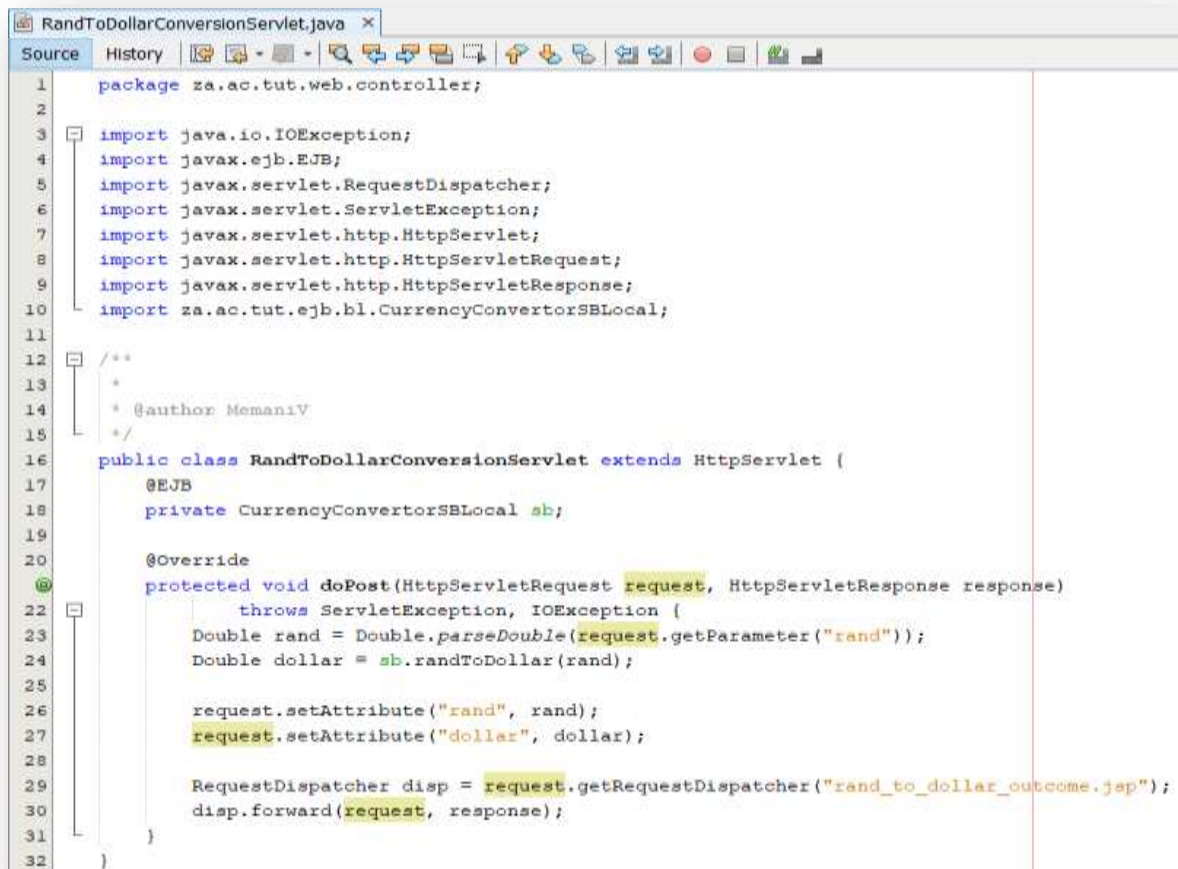


7. Create the **DollarToRandConversionServlet.java** file.



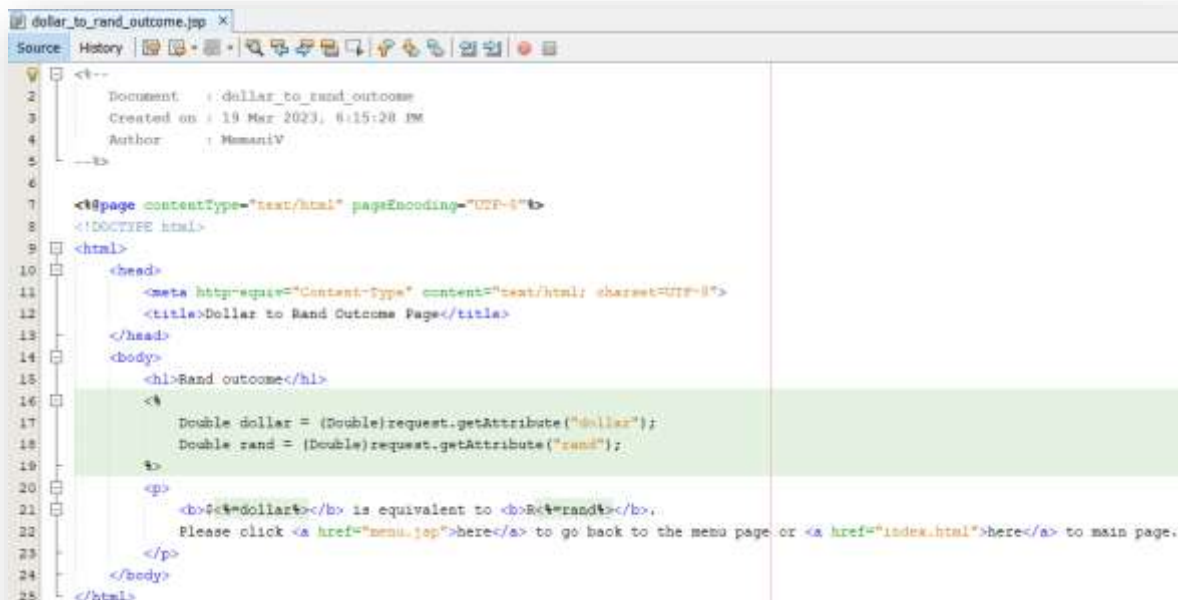
```
1 package za.ac.tut.web.controller;
2
3 import java.io.IOException;
4 import javax.ejb.EJB;
5 import javax.servlet.RequestDispatcher;
6 import javax.servlet.ServletException;
7 import javax.servlet.http.HttpServlet;
8 import javax.servlet.http.HttpServletRequest;
9 import javax.servlet.http.HttpServletResponse;
10 import za.ac.tut.ejb.bl.CurrencyConvertorSBLocal;
11
12 /**
13  *
14  * @author MemaniV
15  */
16 public class DollarToRandConversionServlet extends HttpServlet {
17     @EJB
18     private CurrencyConvertorSBLocal sb;
19
20     @Override
21     protected void doPost(HttpServletRequest request, HttpServletResponse response)
22         throws ServletException, IOException {
23         Double dollar = Double.parseDouble(request.getParameter("dollar"));
24         Double rand = sb.dollarToRand(dollar);
25
26         request.setAttribute("dollar", dollar);
27         request.setAttribute("rand", rand);
28
29         RequestDispatcher disp = request.getRequestDispatcher("dollar_to_rand_outcome.jsp");
30         disp.forward(request, response);
31     }
32 }
```

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



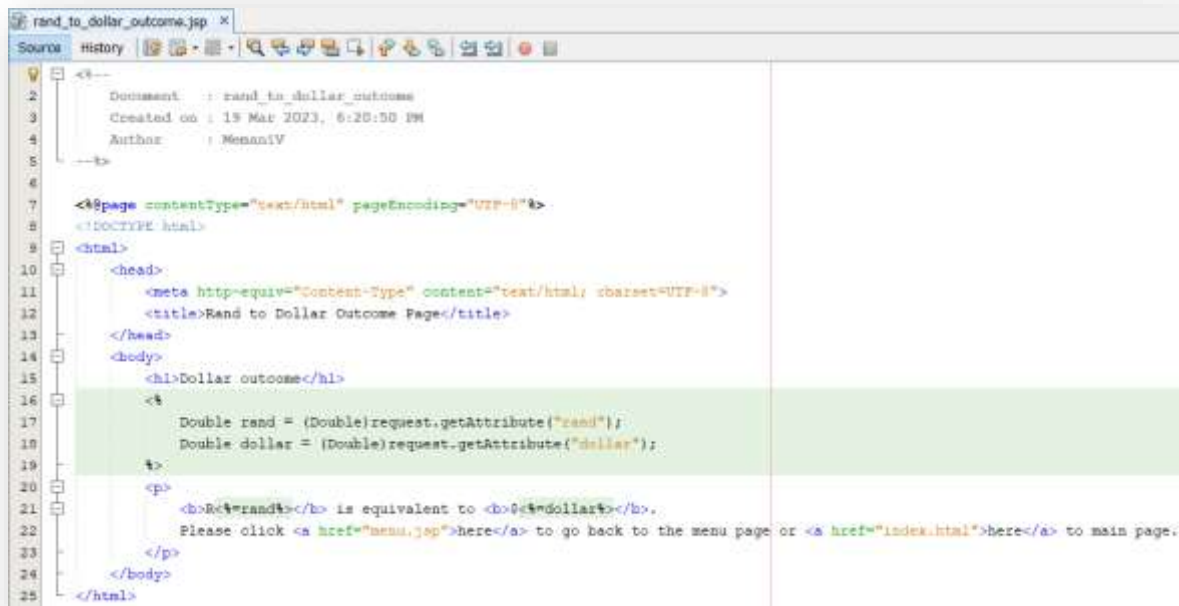
```
1 package za.ac.tut.web.controller;
2
3 import java.io.IOException;
4 import javax.ejb.EJB;
5 import javax.servlet.RequestDispatcher;
6 import javax.servlet.ServletException;
7 import javax.servlet.http.HttpServlet;
8 import javax.servlet.http.HttpServletRequest;
9 import javax.servlet.http.HttpServletResponse;
10 import za.ac.tut.ejb.bl.CurrencyConvertorSBLocal;
11
12 /**
13  *
14  * @author MemaniV
15  */
16 public class RandToDollarConversionServlet extends HttpServlet {
17     @EJB
18     private CurrencyConvertorSBLocal sb;
19
20     @Override
21     protected void doPost(HttpServletRequest request, HttpServletResponse response)
22         throws ServletException, IOException {
23         Double rand = Double.parseDouble(request.getParameter("rand"));
24         Double dollar = sb.randToDollar(rand);
25
26         request.setAttribute("rand", rand);
27         request.setAttribute("dollar", dollar);
28
29         RequestDispatcher disp = request.getRequestDispatcher("rand_to_dollar_outcome.jsp");
30         disp.forward(request, response);
31     }
32 }
```

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



```
1 <!--
2     Document      : dollar_to_rand_outcome
3     Created on    : 19 Mar 2023, 6:15:28 PM
4     Author       : MemaniV
5 -->
6
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>
8 <!DOCTYPE html>
9 <html>
10 <head>
11     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12     <title>Dollar to Rand Outcome Page</title>
13 </head>
14 <body>
15     <h1>Rand outcome</h1>
16
17     <%
18         Double dollar = (Double)request.getAttribute("dollar");
19         Double rand = (Double)request.getAttribute("rand");
20     %>
21
22     <p>
23         <b><%=dollar%></b> is equivalent to <b>R<%=rand%></b>.
24         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.
25     </p>
26 </body>
27 </html>
```

10. Create the `rand_to_dollar_outcome.jsp` file.



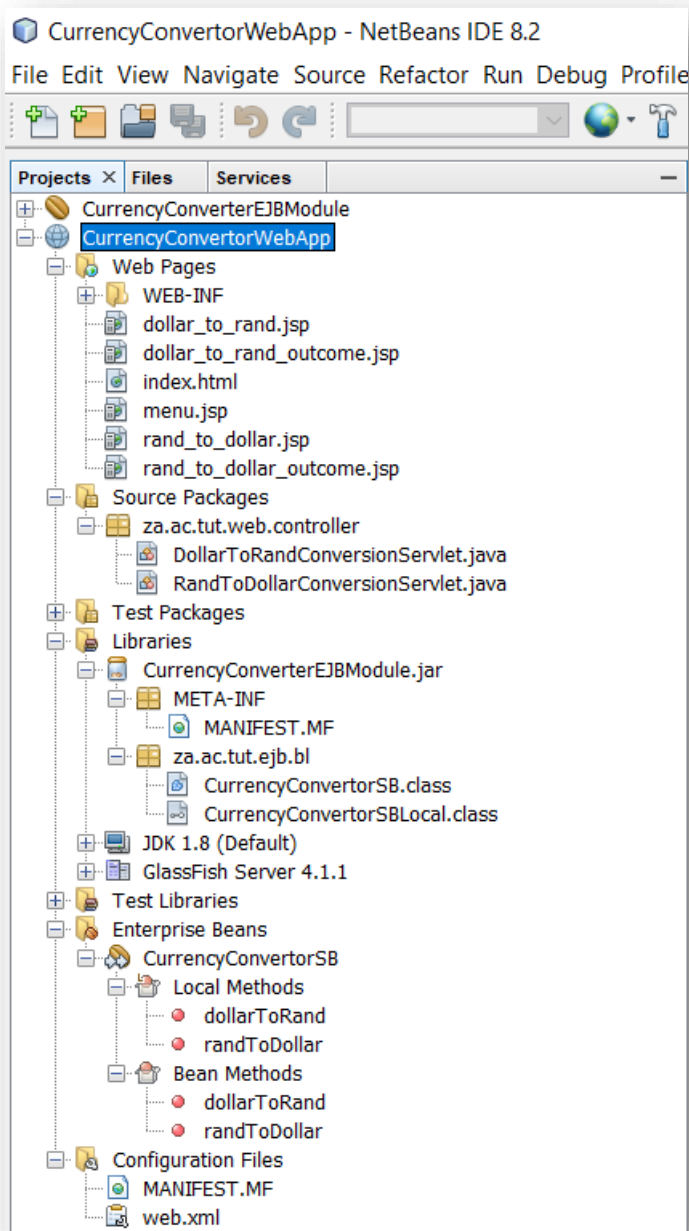
```
1  <!--
2      Document   : rand_to_dollar_outcome
3      Created on : 15 Mar 2023, 6:28:50 PM
4      Author    : Menaniv
5  -->
6
7  <%page contentType="text/html" pageEncoding="UTF-8"%>
8  <!DOCTYPE html>
9  <html>
10     <head>
11         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12         <title>Rand to Dollar Outcome Page</title>
13     </head>
14     <body>
15         <h1>Dollar outcome</h1>
16
17         <%
18             Double rand = (Double)request.getAttribute("rand");
19             Double dollar = (Double)request.getAttribute("dollar");
20
21             <p>
22                 <b>Rand</b> is equivalent to <b>Dollar</b>.
23                 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.
24             </p>
25         </body>
26     </html>
```

11. Compile the web project.



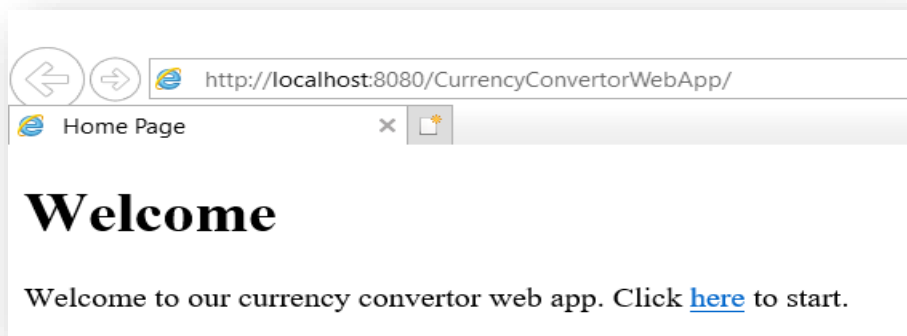
```
Output:
Area DB Database Process: CurrencyConverterWebApp [dev, dist]
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConverterWebApp\build\web\META-INF
Copying 1 file to C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConverterWebApp\build\web\META-INF
Copying 7 files to C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConverterWebAp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

12. View the entire structure of the web project.

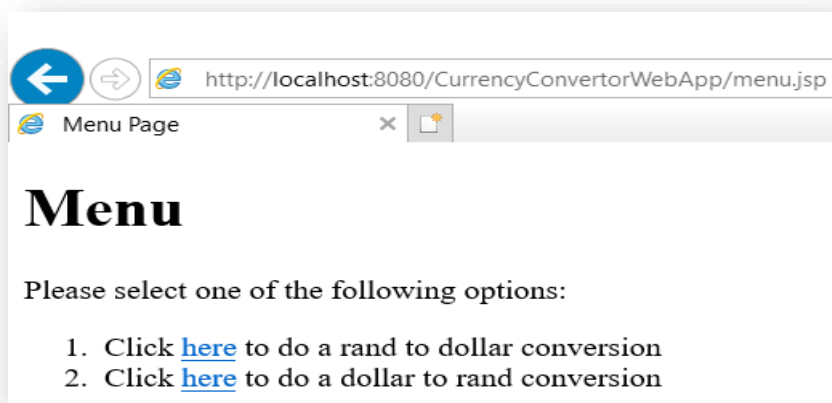


Part C - Run the web client project.

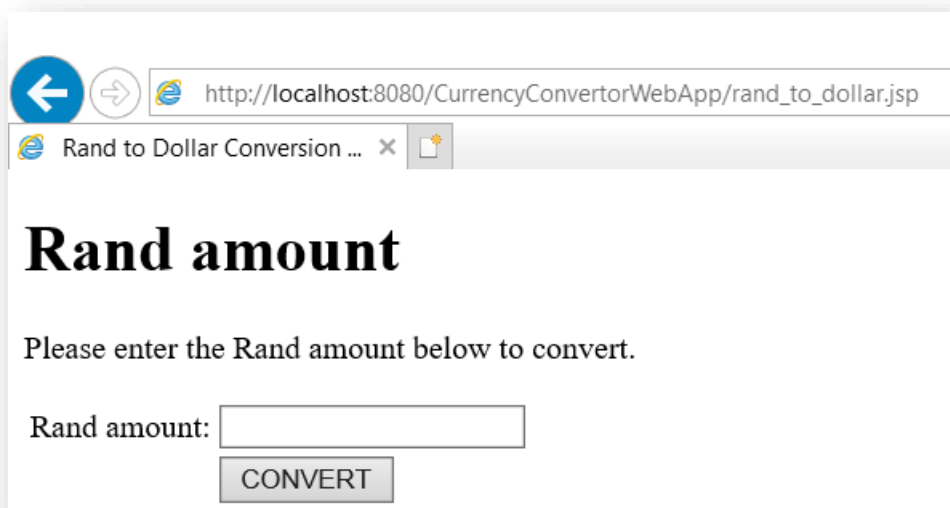
Launch the web app.



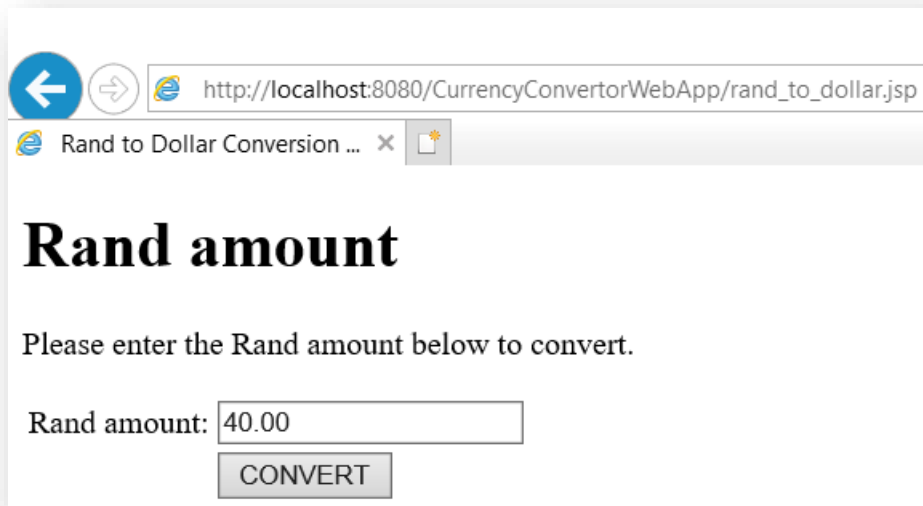
Click on the link.



Click on the first link.

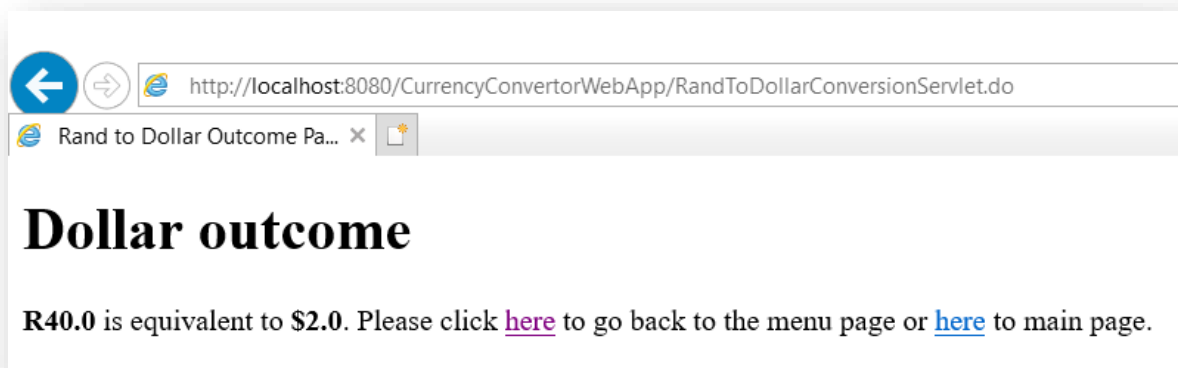


Enter the amount.



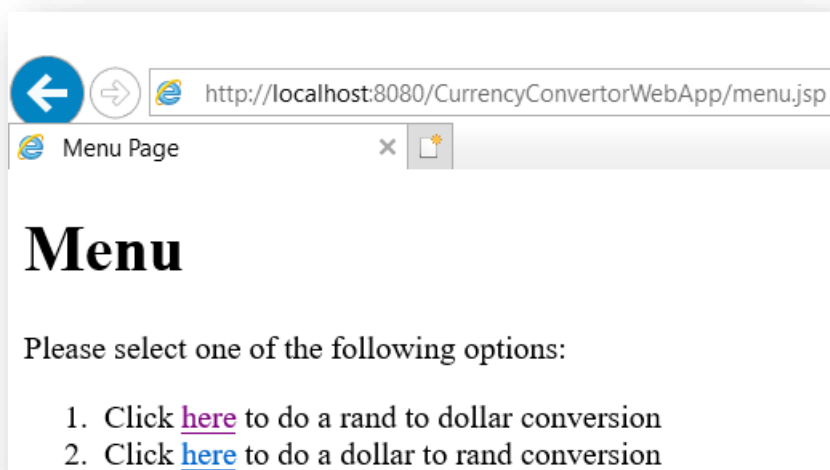
A screenshot of a web browser window. The address bar shows the URL `http://localhost:8080/CurrencyConvertorWebApp/rand_to_dollar.jsp`. The browser tab is titled "Rand to Dollar Conversion ...". The page content features a large heading "Rand amount" in bold black font. Below the heading, a message reads "Please enter the Rand amount below to convert." There is a text input field labeled "Rand amount:" containing the value "40.00". Below the input field is a grey button labeled "CONVERT".

Click on the button.



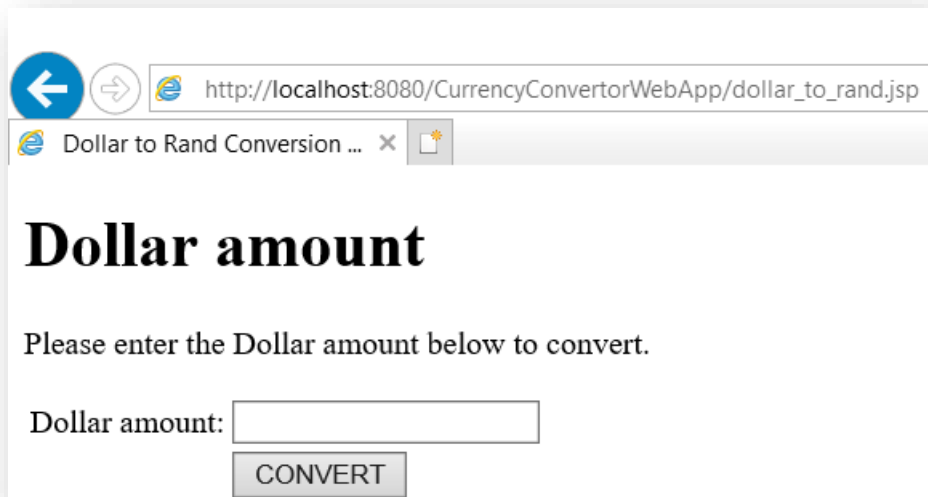
A screenshot of a web browser window. The address bar shows the URL `http://localhost:8080/CurrencyConvertorWebApp/RandToDollarConversionServlet.do`. The browser tab is titled "Rand to Dollar Outcome Pa...". The page content features a large heading "Dollar outcome" in bold black font. Below the heading, a message reads "R40.0 is equivalent to \$2.0. Please click [here](#) to go back to the menu page or [here](#) to main page." The word "here" is underlined and colored purple, while the second "here" is underlined and colored blue.

Go back to the menu page.



A screenshot of a web browser window. The address bar shows the URL `http://localhost:8080/CurrencyConvertorWebApp/menu.jsp`. The browser tab is titled "Menu Page". The page content features a large heading "Menu" in bold black font. Below the heading, a message reads "Please select one of the following options:". There is a numbered list with two items: "1. Click [here](#) to do a rand to dollar conversion" and "2. Click [here](#) to do a dollar to rand conversion". The word "here" in the first item is underlined and colored purple, while the "here" in the second item is underlined and colored blue.

Click on the second link.



← → http://localhost:8080/CurrencyConvertorWebApp/dollar_to_rand.jsp

Dollar to Rand Conversion ... ×

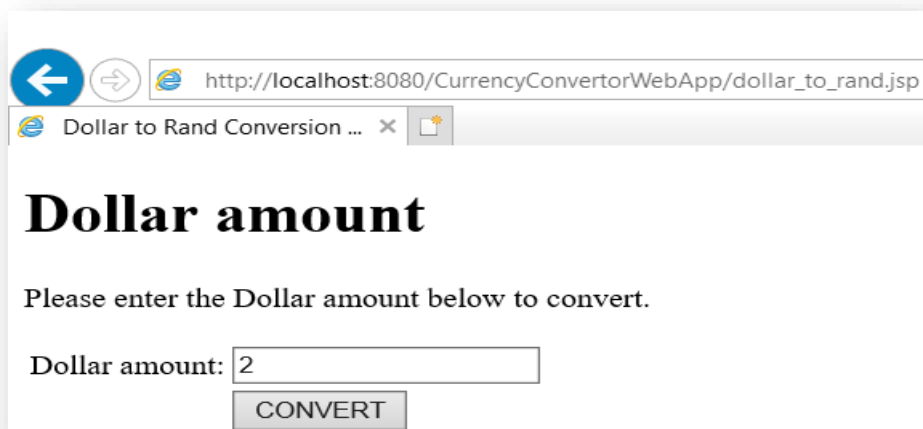
Dollar amount

Please enter the Dollar amount below to convert.

Dollar amount:

CONVERT

Enter the dollar amount



← → http://localhost:8080/CurrencyConvertorWebApp/dollar_to_rand.jsp

Dollar to Rand Conversion ... ×

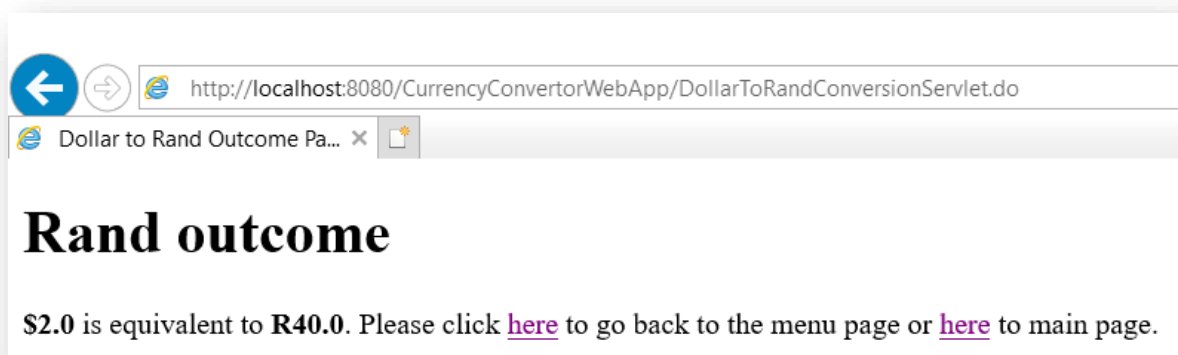
Dollar amount

Please enter the Dollar amount below to convert.

Dollar amount:

CONVERT

Click on the button.



← → http://localhost:8080/CurrencyConvertorWebApp/DollarToRandConversionServlet.do

Dollar to Rand Outcome Pa... ×

Rand outcome

\$2.0 is equivalent to R40.0. Please click [here](#) to go back to the menu page or [here](#) to main page.

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**.

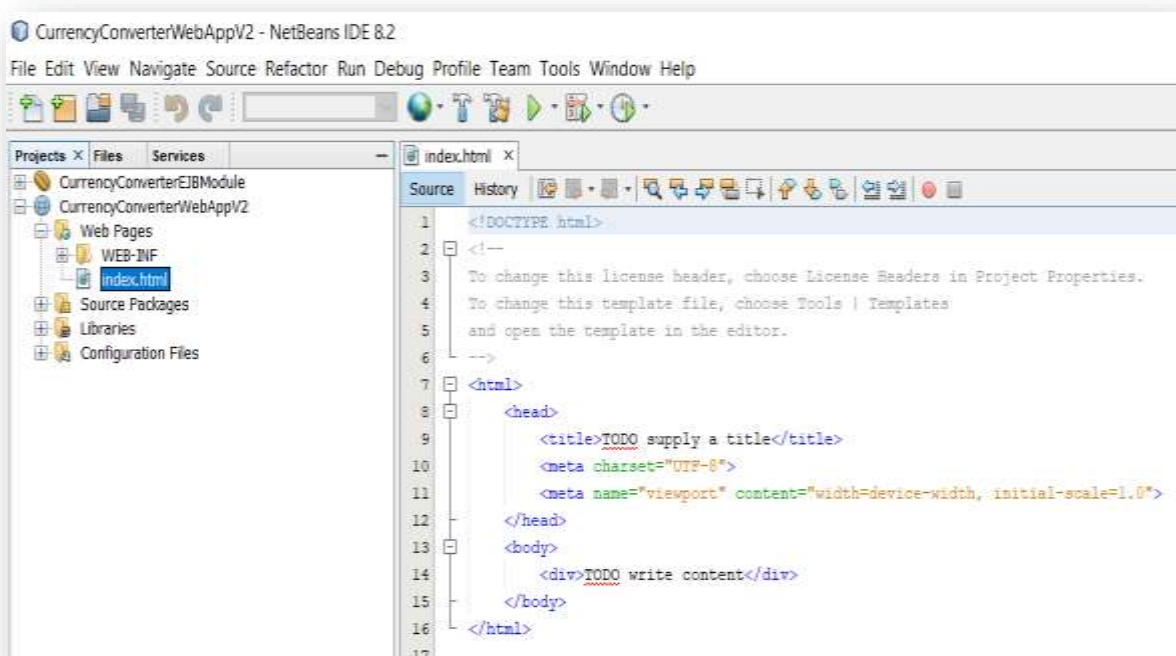
Part A - Create an EJB module project.

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

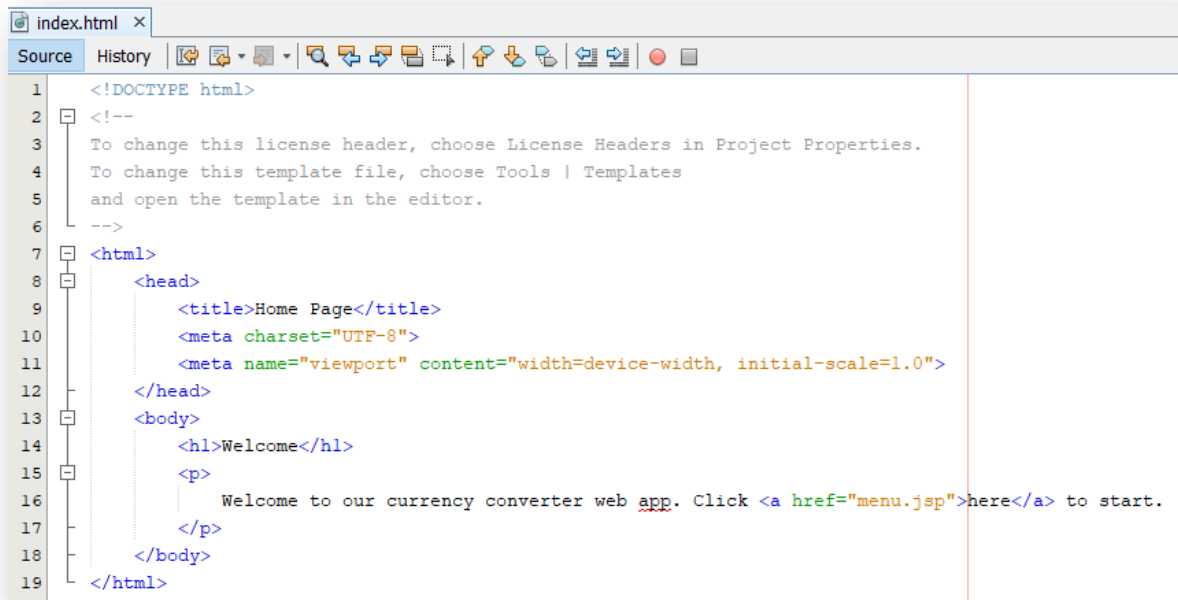
Part B - Create a Web client project

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

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

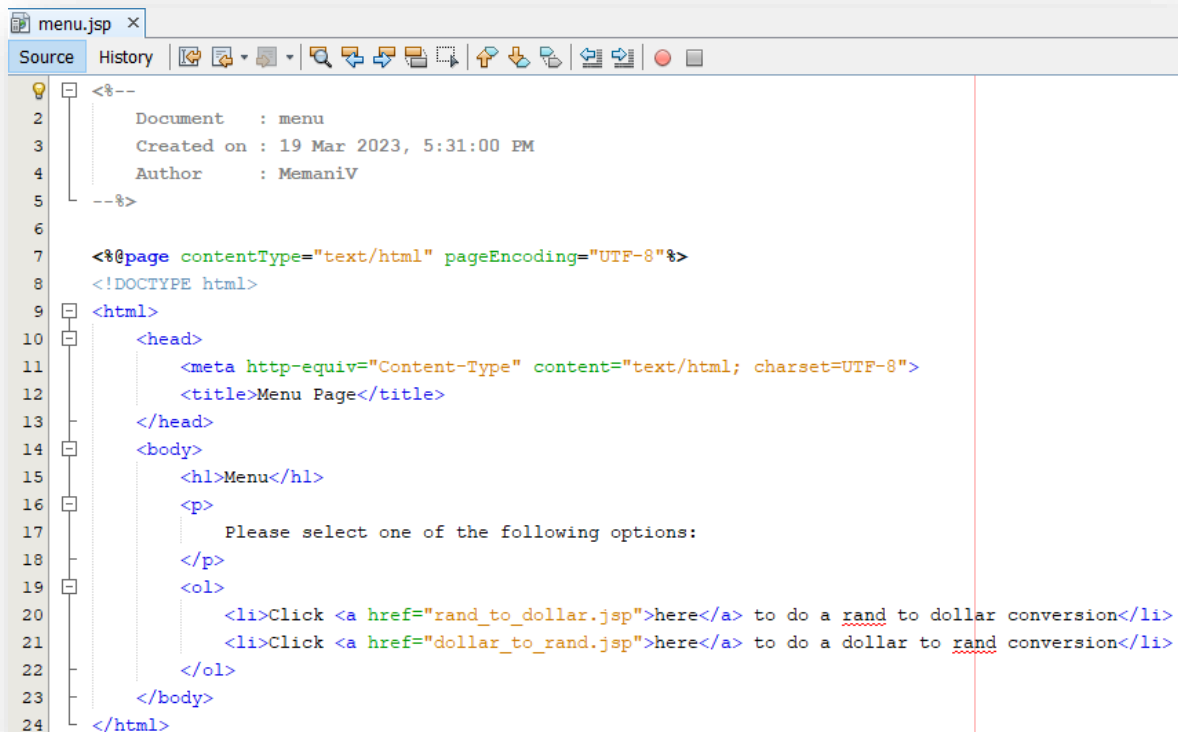


2. Edit the **index.html** file.



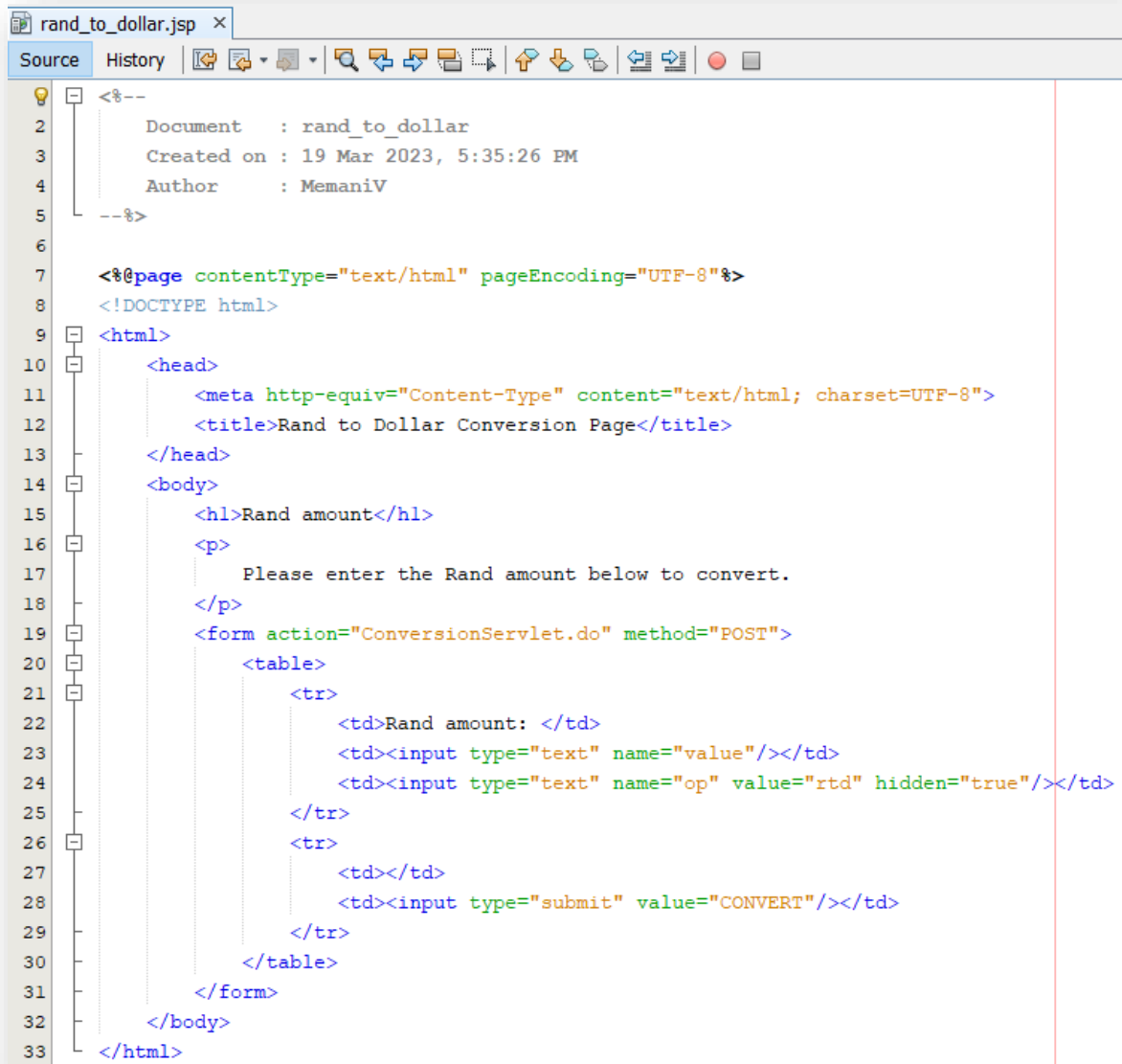
```
1 <!DOCTYPE html>
2 <!--
3 To change this license header, choose License Headers in Project Properties.
4 To change this template file, choose Tools | Templates
5 and open the template in the editor.
6 -->
7 <html>
8 <head>
9 <title>Home Page</title>
10 <meta charset="UTF-8">
11 <meta name="viewport" content="width=device-width, initial-scale=1.0">
12 </head>
13 <body>
14 <h1>Welcome</h1>
15 <p>
16 Welcome to our currency converter web app. Click <a href="menu.jsp">here</a> to start.
17 </p>
18 </body>
19 </html>
```

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



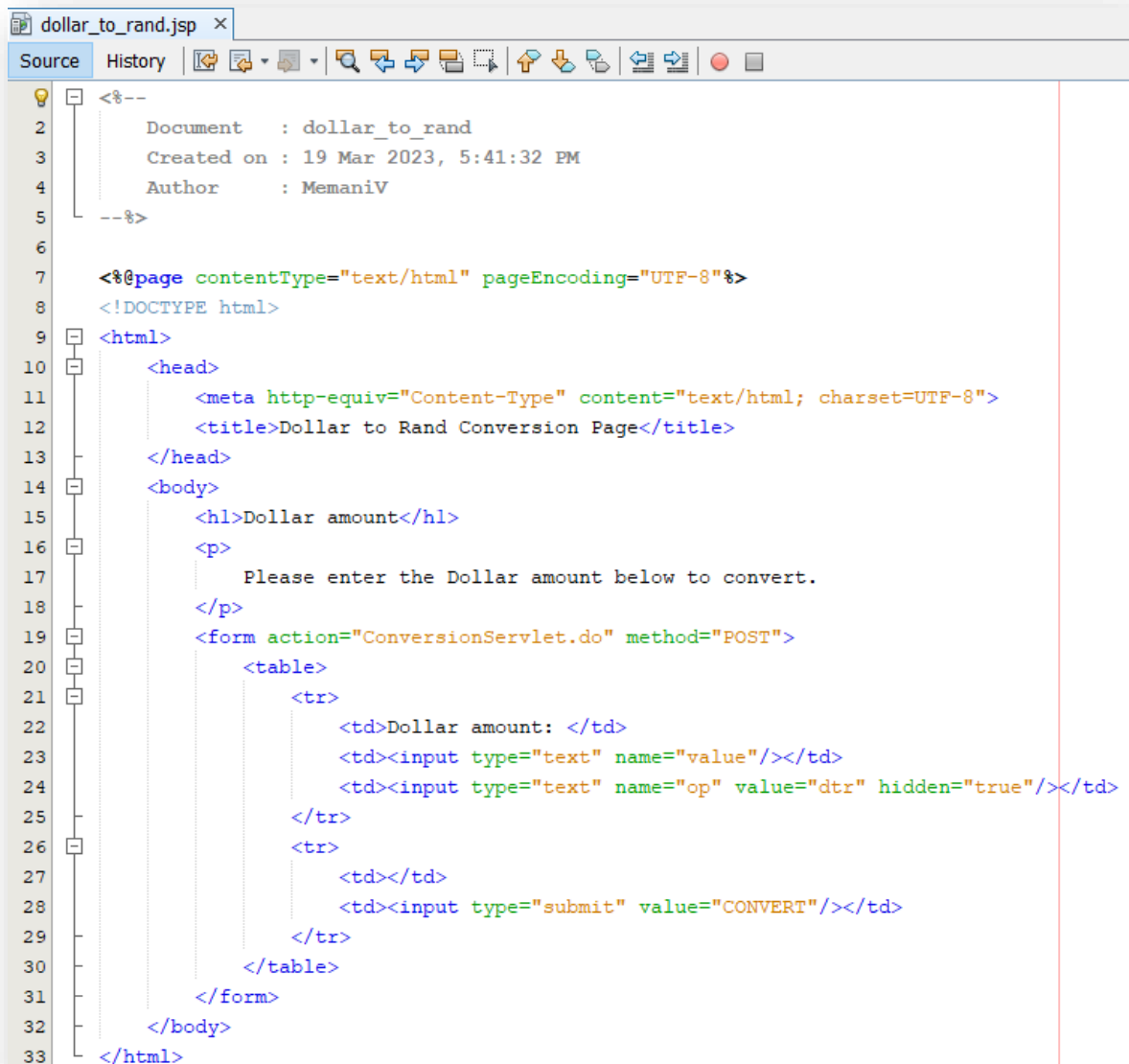
```
1 <%--
2 Document : menu
3 Created on : 19 Mar 2023, 5:31:00 PM
4 Author : MemaniV
5 --%>
6
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>
8 <!DOCTYPE html>
9 <html>
10 <head>
11 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12 <title>Menu Page</title>
13 </head>
14 <body>
15 <h1>Menu</h1>
16 <p>
17 Please select one of the following options:
18 </p>
19 <ol>
20 <li>Click <a href="rand_to_dollar.jsp">here</a> to do a rand to dollar conversion</li>
21 <li>Click <a href="dollar_to_rand.jsp">here</a> to do a dollar to rand conversion</li>
22 </ol>
23 </body>
24 </html>
```

4. Create the `rand_to_dollar.jsp` file.



```
1  <%--
2      Document      : rand_to_dollar
3      Created on    : 19 Mar 2023, 5:35:26 PM
4      Author       : MemaniV
5  --%>
6
7  <%@page contentType="text/html" pageEncoding="UTF-8"%>
8  <!DOCTYPE html>
9  <html>
10     <head>
11         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12         <title>Rand to Dollar Conversion Page</title>
13     </head>
14     <body>
15         <h1>Rand amount</h1>
16         <p>
17             Please enter the Rand amount below to convert.
18         </p>
19         <form action="ConversionServlet.do" method="POST">
20             <table>
21                 <tr>
22                     <td>Rand amount: </td>
23                     <td><input type="text" name="value"/></td>
24                     <td><input type="text" name="op" value="rtd" hidden="true"/></td>
25                 </tr>
26                 <tr>
27                     <td></td>
28                     <td><input type="submit" value="CONVERT"/></td>
29                 </tr>
30             </table>
31         </form>
32     </body>
33 </html>
```

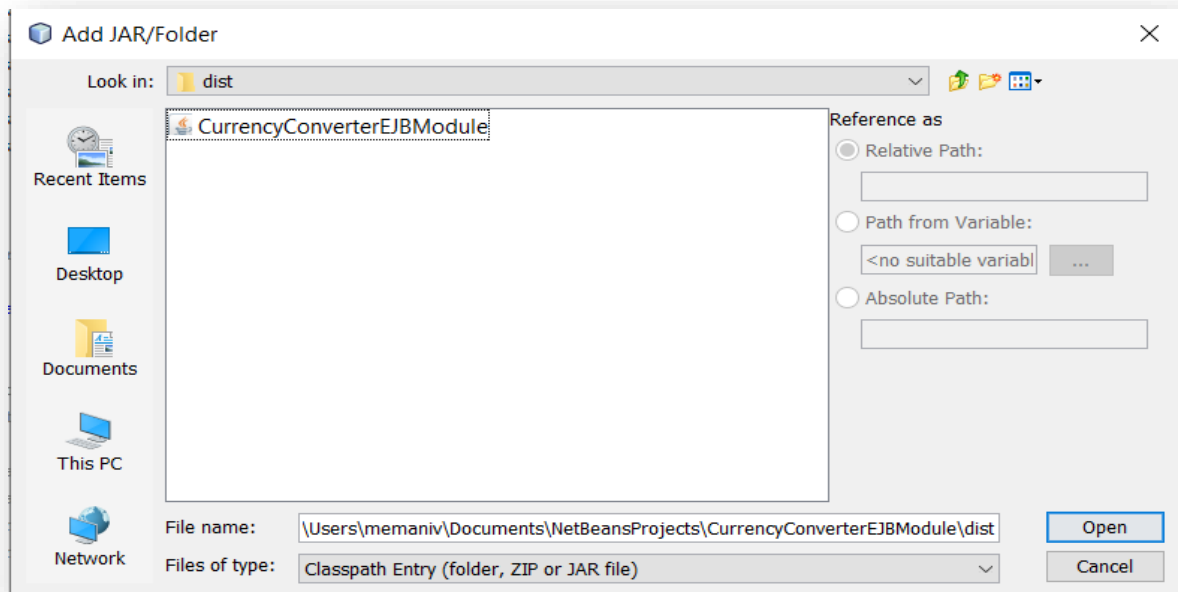
5. Create the **dollar_to_rand.jsp** file.



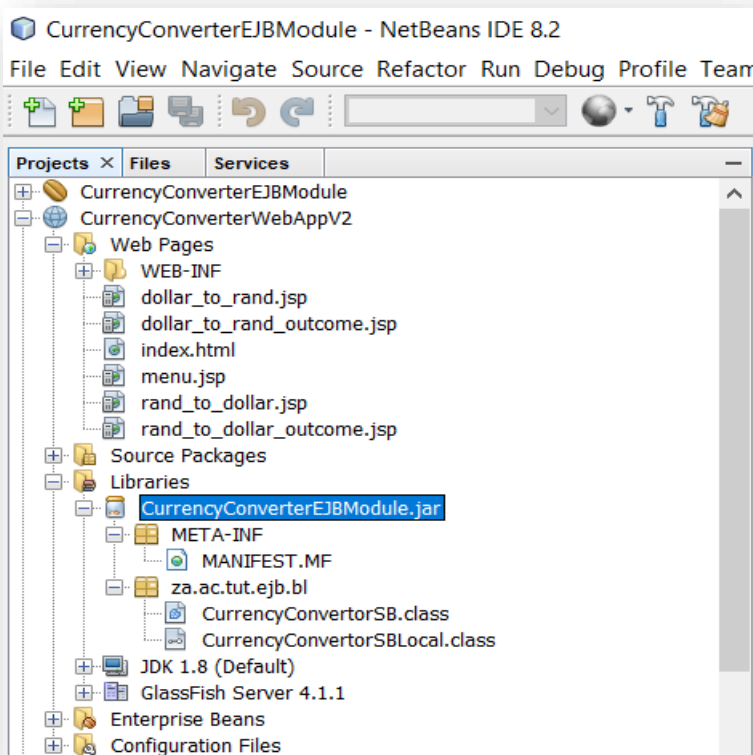
```
dollar_to_rand.jsp x
Source History
<%--
2      Document    : dollar_to_rand
3      Created on  : 19 Mar 2023, 5:41:32 PM
4      Author     : MemaniV
5  --%>
6
7  <%@page contentType="text/html" pageEncoding="UTF-8"%>
8  <!DOCTYPE html>
9  <html>
10     <head>
11         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12         <title>Dollar to Rand Conversion Page</title>
13     </head>
14     <body>
15         <h1>Dollar amount</h1>
16         <p>
17             Please enter the Dollar amount below to convert.
18         </p>
19         <form action="ConversionServlet.do" method="POST">
20             <table>
21                 <tr>
22                     <td>Dollar amount: </td>
23                     <td><input type="text" name="value"/></td>
24                     <td><input type="text" name="op" value="dtr" hidden="true"/></td>
25                 </tr>
26                 <tr>
27                     <td></td>
28                     <td><input type="submit" value="CONVERT"/></td>
29                 </tr>
30             </table>
31         </form>
32     </body>
33 </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.

```
12  /**
13   *
14   * @author MemaniV
15   */
16  public class ConversionServlet extends HttpServlet {
17      @EJB
18      private CurrencyConvertorSBLocal ccl;
19
20      @Override
21      protected void doPost(HttpServletRequest request, HttpServletResponse response)
22          throws ServletException, IOException {
23          Double value = Double.parseDouble(request.getParameter("value"));
24          String op = request.getParameter("op");
25          String outcome;
26
27          if(op.equals("dtr")){
28              //perform dollar to rand operation
29              Double rand = ccl.dollarToRand(value);
30              outcome = "$" + value + " is equivalent to R" + rand;
31          } else {
32              //perform rand to dollar operation
33              Double dollar = ccl.randToDollar(value);
34              outcome = "R" + value + " is equivalent to R" + dollar;
35          }
36
37          request.setAttribute("outcome", outcome);
38
39          RequestDispatcher disp = request.getRequestDispatcher("outcome.jsp");
40          disp.forward(request, response);
41      }
42  }
```

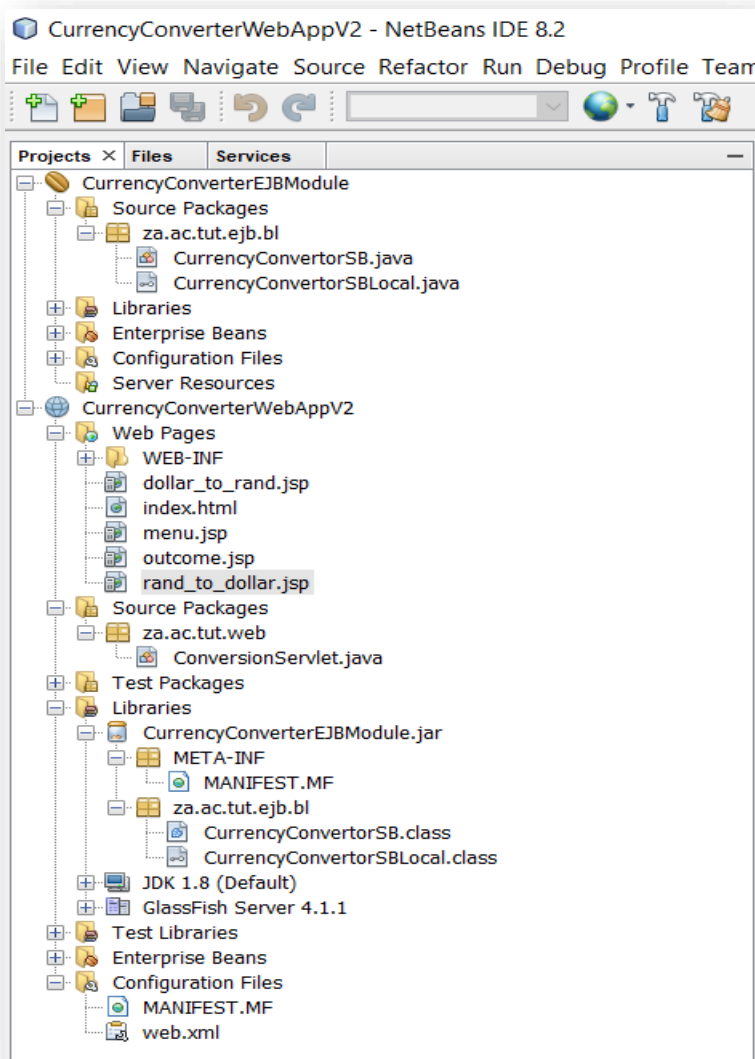
8. Create the **outcome.jsp** file.

```
outcome.jsp x
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.

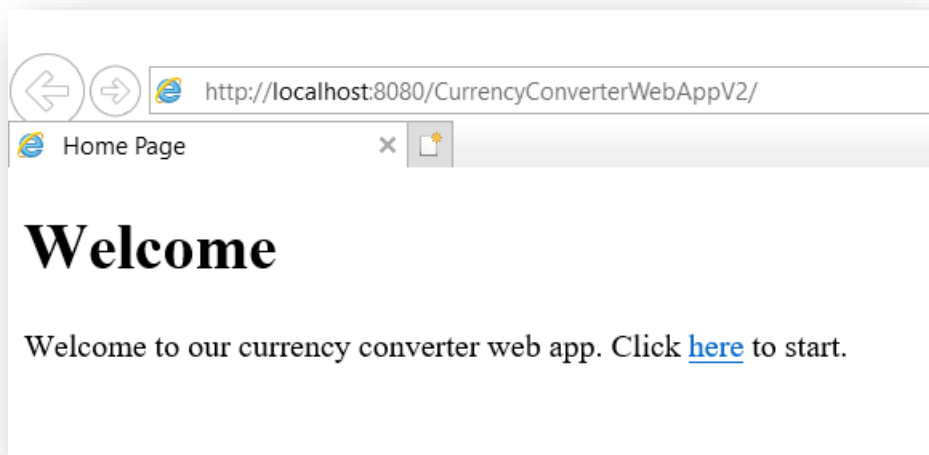
```
Output - CurrencyConverterWebAppV2 [cleaned] x
depc-swt-jar:
depc-jar:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConverterWebAppV2\build\web\WEB-INF\classes
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConverterWebAppV2\build\web\META-INF
Copying 1 file to C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConverterWebAppV2\build\web\META-INF
Copying 7 files to C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConverterWebAppV2\build\web
library-inclusion-in-archive:
Copying 1 file to C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConverterWebAppV2\build\web\WEB-INF\lib
library-inclusion-in-manifest:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConverterWebAppV2\build\empty
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConverterWebAppV2\build\generated-sources\ap-source-output
Compiling 1 source file to C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConverterWebAppV2\build\web\WEB-INF\classes
compile:
compile-jsp:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConverterWebAppV2\dist
Building jar: C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConverterWebAppV2\dist\CurrencyConverterWebAppV2.war
dist:
dist:
BUILD SUCCESSFUL (total time: 0 seconds)
```

12. View the entire structure of the web project.

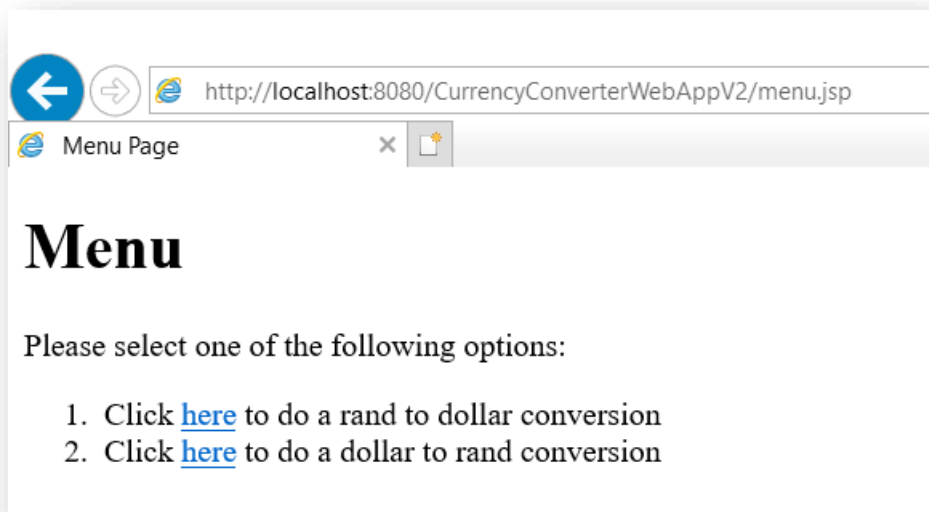


Part C - Run the web client project.

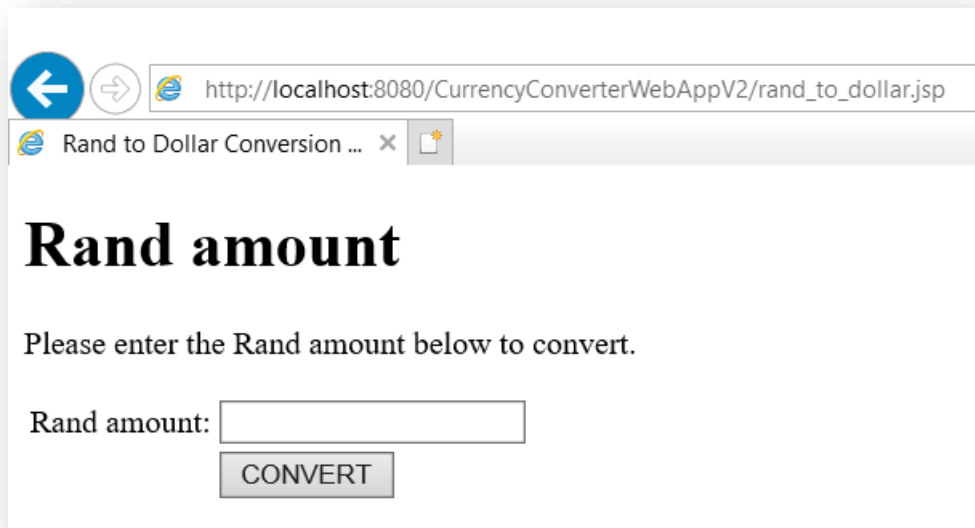
Launch the web app.



Click on the link.

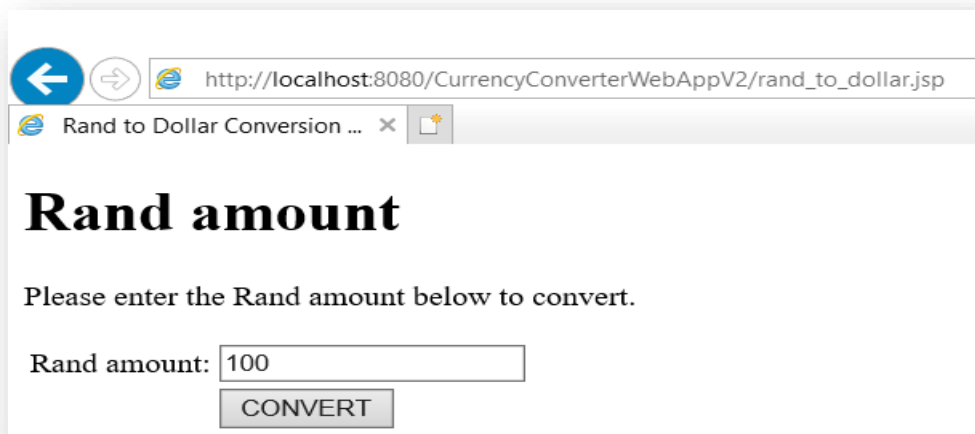


Click on the first link.



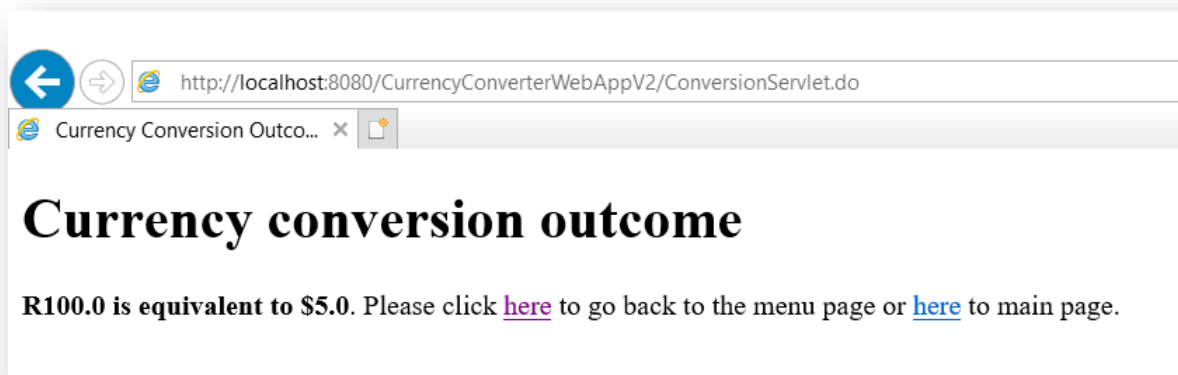
A screenshot of a web browser window. The address bar shows the URL `http://localhost:8080/CurrencyConverterWebAppV2/rand_to_dollar.jsp`. The browser tab is titled "Rand to Dollar Conversion ...". The page content features a large heading "Rand amount" in bold black font. Below the heading is the instruction "Please enter the Rand amount below to convert." followed by a text input field labeled "Rand amount:". The input field is empty. Below the input field is a grey button labeled "CONVERT".

Enter the amount.



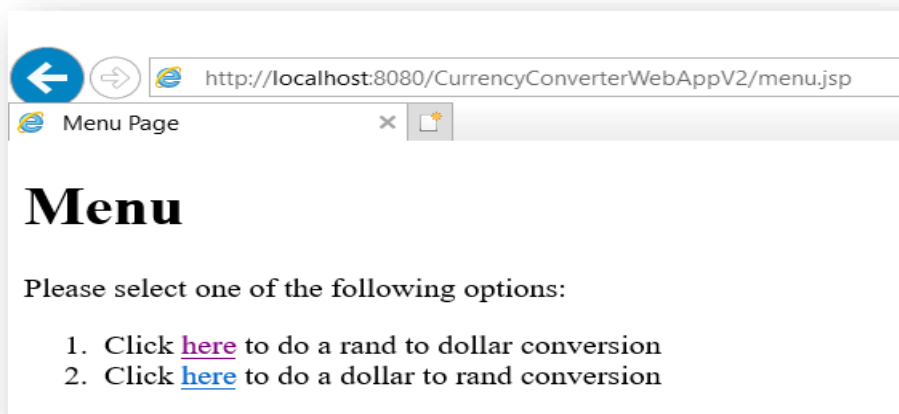
A screenshot of the same web browser window. The address bar and browser tab remain the same. The page content is identical to the previous screenshot, but the text input field now contains the number "100". The "CONVERT" button remains visible below the input field.

Click on the button.

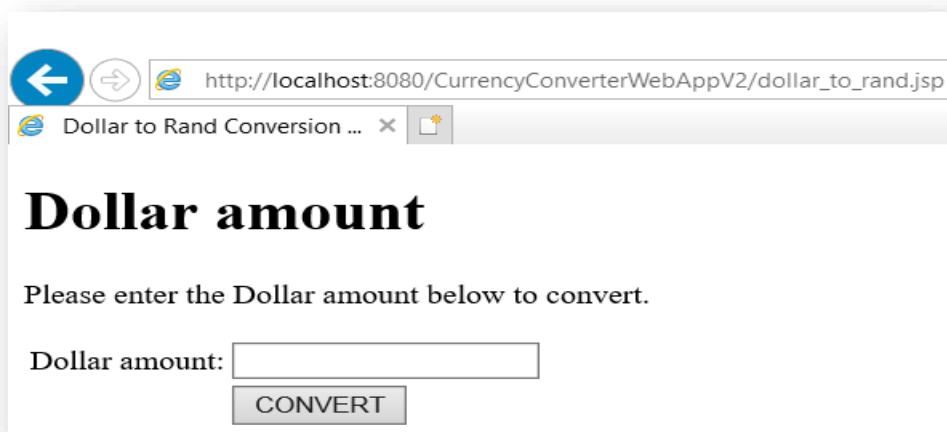


A screenshot of a web browser window showing the result of the conversion. The address bar shows the URL `http://localhost:8080/CurrencyConverterWebAppV2/ConversionServlet.do`. The browser tab is titled "Currency Conversion Outco...". The page content features a large heading "Currency conversion outcome" in bold black font. Below the heading is the text "R100.0 is equivalent to \$5.0. Please click [here](#) to go back to the menu page or [here](#) to main page." where the links are underlined and colored (purple and blue respectively).

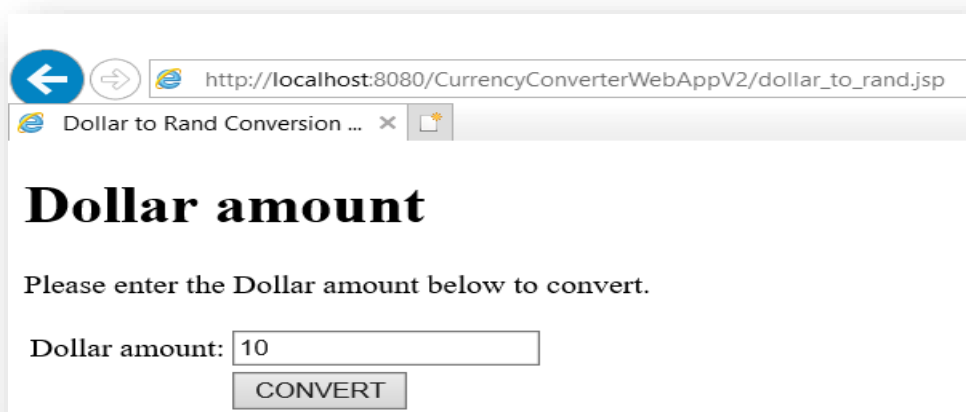
Go back to the menu page.



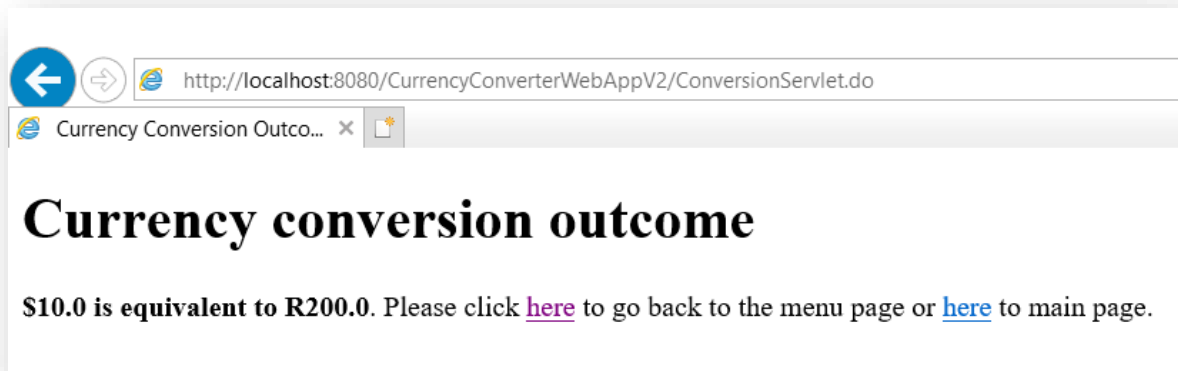
Click on the second link.



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**”).

To do

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