



**Tshwane University
of Technology**

We empower people

**Faculty of
Information and Communication Technology**

Department of Computer Science

SUBJECT NOTES

NQF LEVEL	NQF CREDITS	MODULE NAME	MODULE CODE	YEAR	SEMESTER
6	15	Internet Programming	INT316D	2023	1
QUALIFICATION			SAQA ID:		
Diploma in Computer Science			H/H16/E089CAN		
Diploma in Multimedia Computing					
Diploma in Information Technology					
Diploma in informatics					

*Compiled by Vuyisile Memani
Revised on: January 2023*



©COPYRIGHT: *Tshwane University of Technology*
 Private Bag X680
 PRETORIA
 0001

All rights reserved. Apart from any reasonable quotations for the purposes of research criticism or review as permitted under the Copyright Act, no part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy and recording, without permission in writing from the publisher.

Table of contents

1	Introduction to INT316D and JEE	1
1.1	What is the purpose of INT316D?	1
1.1.1	Enterprise.....	1
1.1.2	Distributed.....	1
1.1.3	Scalability.....	1
1.1.4	Fault-tolerant.....	2
1.1.5	High availability	2
1.1.6	Secured.....	2
1.2	How do we accomplish the module's purpose?	2
1.2.1	Presentation layer	2
1.2.2	Business layer.....	3
1.2.3	Persistence/Database layer	3
1.3	Typical web applications to be developed.....	3
1.4	Software to be used in the module.....	4
1.5	What is JEE?.....	5
1.5.1	Web Server	5
1.5.2	Application Server.....	6
1.6	JEE architecture.....	6
1.7	Benefits of using JEE compliant servers	7
1.8	Example	8
1.9	DIY (Do It Yourself)	14
1.10	Conclusion	14
2	MVC.....	15
2.1	What is MVC?	15
2.1.1	Models	15
2.1.2	Views	15
2.1.3	Controllers.....	16
2.2	Examples	16
2.2.1	Example 1	16
2.2.2	Example 2	20
2.3	DIY (Do It Yourself)	23
2.4	Conclusion	24
3	JSP	25
3.1	What is JSP?.....	25

3.2 Relationship between JSPs and Servlets.....	25
3.3 Implicit objects of JSP	25
3.4 How to write a scriptlet and an expression in JSP?.....	26
3.5 How to declare variables, write comments and import packages in JSP?	26
3.6 Example	28
3.7 DIY (Do It Yourself)	37
3.8 Conclusion	38
4 Servlets.....	39
4.1 What is a servlet?.....	39
4.2 Servlet life cycle	40
4.3 How does a servlet work?.....	41
4.4 Servlet API	41
4.5 Example 1	44
4.6 Example 2	56
4.7 DIY (Do It Yourself)	71
4.8 Conclusion	73
5 Conversational web apps	74
5.1 What is a conversational web app?.....	74
5.2 HTTP.....	75
5.2.1 Anatomy of HTTP request messages	75
5.2.2 Anatomy of HTTP response messages.....	77
5.2.3 HTTP methods.....	78
5.3 Example	78
5.4 DIY (Do It Yourself)	106
5.5 Conclusion	108
6 Exceptions and exception handling	109
6.1 What is an exception?	109
6.2 What is exception handling?	110
6.2.1 Try catch block.....	110
6.2.2 Exception rethrowing methods.....	111
6.3 Predefined exceptions.....	111
6.3.1 Exception class	111
6.3.2 NumberFormatException class	112
6.3.3 RuntimeException.....	112
6.4 User-defined exceptions	113
6.5 How to handle exceptions in a web development environment?.....	113

6.6 Example	114
6.7 DIY (Do It Yourself)	129
6.8 Conclusion	130
7 Enterprise Java Beans (EJBs)	131
7.1 What are EJBs?	131
7.2 Session Beans	132
7.2.1 Stateless Session Bean	132
7.2.2 Stateful Session Bean.....	168
7.2.3 Singleton	193
7.3 DIY (Do It Yourself)	214
7.4 Conclusion	218
8 Java Persistence API (JPA)].....	219
8.1 What is JPA?.....	219
8.2 Annotations	220
8.2.1 @Entity	220
8.2.2 @Table	221
8.2.3 @Id	221
8.2.4 @GeneratedValue	222
8.2.5 @Column.....	222
8.2.6 @Transient	223
8.2.7 @Temporal	224
8.2.8 @SecondaryTable	224
8.2.9 @SecondaryTables	225
8.3 Implementation of JPA annotations.....	226
8.3.1 Using basic annotations	226
8.3.2 Using basic annotations and secondary tables.....	283
8.3.3 Using basic annotations and collections.	326
8.3.4 Using basic annotations and maps	359
8.4 Mapping relationships	394
8.4.1 OneToOne	394
8.4.2 One-to-many	436
8.5 Inheritance and JPA.....	479
8.5.1 Single table per class hierarchy strategy.....	479
8.5.2 Joined-subclass strategy.....	525
8.5.3 Table per concrete class strategy	556
9 JPQL.....	593

9.1 What is JPQL?	593
9.2 JPQL syntax.....	594
9.2.1 SELECT , FROM and WHERE	594
9.2.2 DELETE	595
9.2.3 UPDATE	595
9.3 Binding parameters	596
9.4 JPQL examples.....	598
9.4.1 Dynamic queries	598
9.4.2 Named queries.....	640
10Appendix A: How to download and install JDK	677
11Appendix B: How to download and install Notepad++	682
12Appendix C: How to download and install Tomcat.....	687
13Appendix D: How to download and install MySQL.....	694
14References	695
14.1 Prescribed textbook	695
14.2 Recommended Additional Resources	695

1 Introduction to INT316D and JEE

In this chapter we introduce the student to our INT316D (Internet Programming) module and JEE (Java Enterprise Edition) which is also known as J2EE (Java 2 Enterprise Edition). In our discussion we will cover the following:

- The purpose of the module.
- How we accomplish the module's purpose.
- Relevant software for the module.
- The JEE framework.

1.1 What is the purpose of INT316D?

The purpose of this module is to capacitate students with the requisite skills to develop enterprise web applications that are **distributed**, **scalable**, **fault-tolerant**, **highly-available**, and **secured** using the **JEE** framework. JEE is an acronym for Java Enterprise Edition. These web applications can either be developed using an IDE such as **NetBeans** or a simple editor such as **Notepad++**. Ultimately the web applications are deployed in a **Web Server** such as **Tomcat** or an **Application Server** such as **GlassFish**. The user then accesses the web applications through a browser.

1.1.1 Enterprise

By enterprise we mean that these are business applications. These are e-commerce systems that allows companies to trade over the internet.

1.1.2 Distributed

By distributed we mean that the components of the web application are not located in the same JVM (Java Virtual Machine), they can be distributed over different machines to accomplish a task. This has great benefits in that even during disaster, the system continues working (Never put all your eggs in one basket!!!).

1.1.3 Scalability

Scalable applications continue to function even when new components are added to the system. They don't require the whole system to be rewritten just because a new component is being added.

1.1.4 Fault-tolerant

By fault-tolerant we mean that the enterprise web applications can handle exceptions that may occur during program execution. This makes the enterprise web applications to be robust, meaning, they fail safe in the midst of unexpected error conditions.

1.1.5 High availability

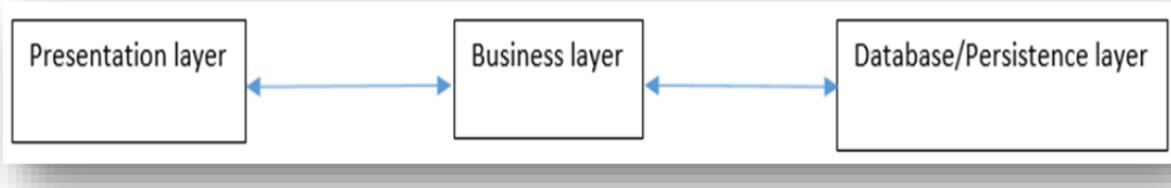
Highly availability simply means the enterprise web applications are available 24/7. This high availability over the web is due to the deployment of the business applications in servers which is normally kept running all the time.

1.1.6 Secured

By secured we mean that the web applications we create implement security measures. In a secured system, users are authenticated and authorised to access resources based on the role they play in the system.

1.2 How do we accomplish the module's purpose?

We develop enterprise web applications that conform to the 3-tier model of creating software. The 3-tier model simply says our web applications should consist of components that can be categorised into three tiers/layers, namely the presentation layer; business layer, and the database/persistence layer. The figure below shows the 3-tier model.



1.2.1 Presentation layer

The presentation layer or tier is mainly responsible for viewing of data. Through this layer users of the system are allowed to enter data, and also view processed data. In this module we are going to use the following presentation layer components:

- JSP
- JSTL

- Servlets
- HTML
- JavaScript

1.2.2 Business layer

The business layer or tier is responsible for the processing of data. In this layer the actual business logic is performed. Say we want determine the sum of two numbers. The summation of the numbers is the business logic, and it is done in this layer. We mainly use the following components to perform business logic:

- POJOs (Plain Old Java Objects, which are classes);
- EJBs (Enterprise Java Beans);

1.2.3 Persistence/Database layer

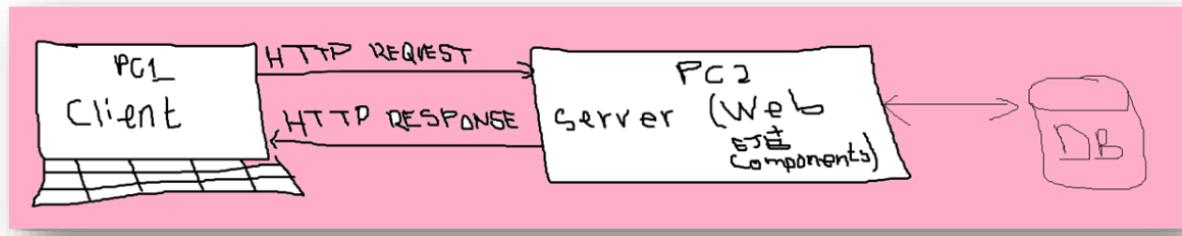
The persistence layer is responsible for database interaction. Through this layer, data is stored in the database and again retrieved from it. We use entities (special EJBs) to interact with databases through an API called JPA (Java Persistence API). There are various DBMS (Database Management System) available out there such as Derby, MariaDB, Oracle, MySQL etc. Any of these will be used for database interaction with the web applications.

1.3 Typical web applications to be developed

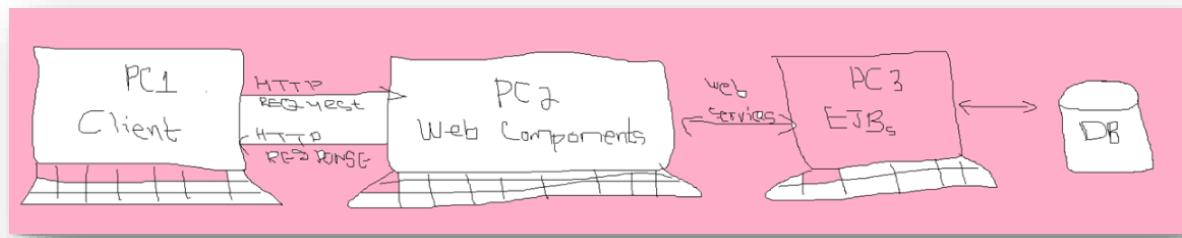
The web applications to be developed in this module will conform to the 3-tier model. We will use two PCs. The first PC will be a client that makes requests and then receives responses. The requests are made through a browser and the responses get rendered by the same browser.

The second PC will be a server that receives requests and send back responses to the client. The server will interact with the database. The protocol of communication between the client and the server will be **http**.

The figure below shows the infrastructure of a typical web application we will develop.



Time permitting, we will change the infrastructure to the diagram below.



In this architecture we deploy the components of an application in distributed machines. Web components are kept in their own machine, different from EJBs.

1.4 Software to be used in the module

In this module we are going to use **Notepad++** and **Tomcat** to develop and deploy enterprise web applications. Prior to installing the two software packages, we download and install JDK Java Development Kit). Version 8 or the latest version will work fine. In Appendix A we show the student how to download and install JDK in the computer.

In Appendix B we take the student through the steps of downloading and installing Notepad++. In Appendix C we show how to download and install Tomcat. Appendix D shows how to download and install MySQL.

In as much as Notepad++, Tomcat, and MySQL are the software packages of choice in this module, feel free to use any IDE (NetBeans, BlueJ etc), server (GlassFish, TomEE, WildFly, Jetty etc) or DBMS (Postgress, Derby, MariaDB etc).

1.5 What is JEE?

JEE is a set or collection of APIs (class libraries) used to develop business applications. The APIs are also known as standards. Through the JEE framework, we are able to develop business applications that are distributed, secured, robust, scalable, and highly available. There are other frameworks in existence for developing business applications. Some of the frameworks are:

- Structs;
- Spring; and
- Hibernate.

The JEE APIs are implemented in containers or servers. Servers are divided into two, depending on which part of the APIs they support or implement. If a server only implements the web part of the APIs, we call it a Web Server. If it supports both the Web and EJB components, we call it an Application Server. So servers are reference implementors (RI) JEE APIs or standards.

1.5.1 Web Server

A Web Server is a container that implements or supports the web components of JEE. Some of the web components of JEE are the following:

- Servlets;
- JSPs;
- JSF pages;
- Filters;
- Listeners;
- EJBs Lite; and
- Web Services

An example of a web server is **Tomcat**. You can get Tomcat from the following location:

[Apache Tomcat® - Welcome!](#)

In **Appendix C** we show the student how to download, install, start and stop Tomcat.

1.5.2 Application Server

An Application Server is a container that supports both the web and EJB components of JEE. An example of an Application Server is **GlassFish**. You can get GlassFish from the following location:

<https://javaee.github.io/glassfish/>

Other than GlassFish, we also have the following popular Application Servers:

WildFly/JBOSS

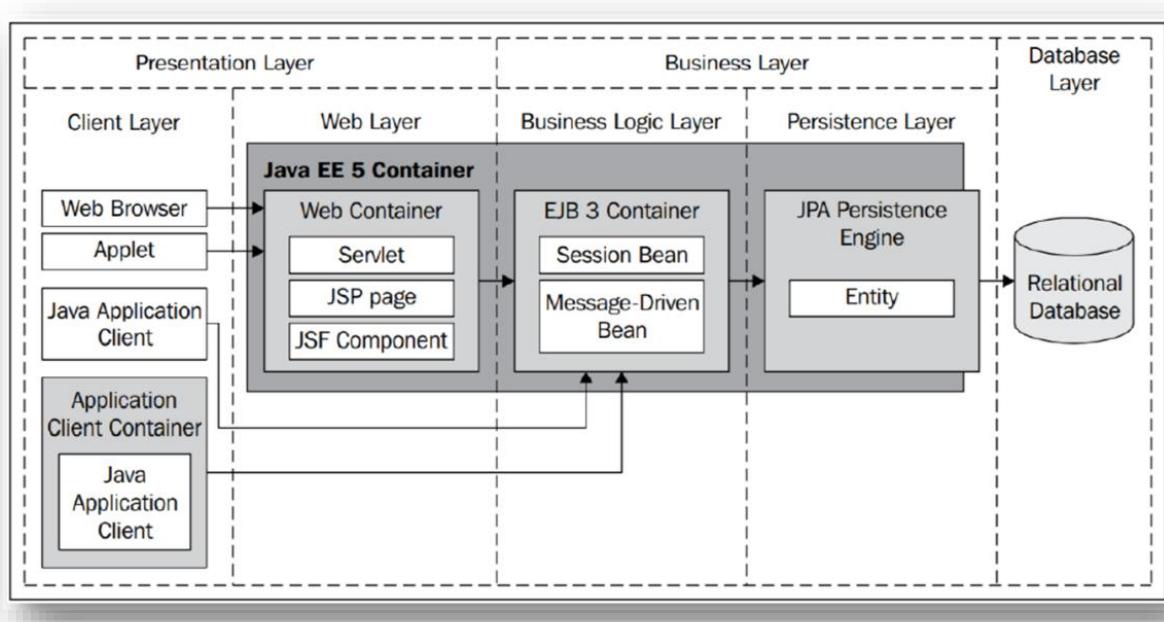
<https://www.wildfly.org/>

TomEE

<https://tomee.apache.org/>

1.6 JEE architecture

The figure below shows the JEE architecture.



In the Java EE architecture diagram depicted above, the commonly known 3 layer model has become 5 layers. This is so because the presentation layer is subdivided into the client layer and the web layer. The business layer is also divided into the business logic layer and the persistence layer. In practice, the distinction between client/web and business logic/persistence layers is not always made. The JEE architecture is simply referred to as n layer or multi-layer architecture.

The Web Browser and Applet can talk directly to the web container. The web components can then talk to the EJB container, which can talk to the database. The Java Application Client can run as a standalone Java class with the main method or be deployed in the Application Client Container of the server as a Java Client project. The application client can talk directly to the EJB container, which in turn can talk to the database.

1.7 Benefits of using JEE compliant servers

The business web applications we create are ultimately deployed on the server. The server gives us the following benefits:

- The applications are available 24/7, so long the server is running.
- The server runs the deployed applications for us. The applications we develop don't have the "**main**" method. It is the server that runs them.
- The server manages the life cycle of components such as **Servlets** and the **EJBs**. This means the instantiation and destruction of these components is done by the server, not us.
- The server supports multi-threading, we don't need to create threads for every request we get.
- The server supports both declarative and programmatic security.
- The server supports seamless communication with ports. We don't need to write code that listens to ports.

1.8 Example

Problem statement

Tomcat comes up with examples of web applications. These examples are found in the **webapp** directory of **Tomcat**. So, in this example we are going to pick one example, deploy it on Tomcat, and run on the browser.

Solution

For you to successfully complete this example, do the following tasks:

- Open the root directory of Tomcat.

Name	Date modified	Type	Size
bin	2022/12/12 10:11	File folder	
conf	2022/12/12 10:11	File folder	
lib	2022/12/12 10:11	File folder	
logs	2023/01/19 14:47	File folder	
temp	2022/12/12 10:11	File folder	
webapps	2023/01/20 13:19	File folder	
work	2022/12/12 10:11	File folder	
LICENSE	2022/11/16 15:34	File	57 KB
NOTICE	2022/11/16 15:34	File	2 KB
RELEASE-NOTES	2022/11/16 15:34	File	8 KB
tomcat	2022/11/16 15:34	Icon	22 KB
Uninstall	2022/11/16 15:34	Application	86 KB

- Open the **webapps** directory. This directory contains all the web applications that come along with Tomcat.

Name	Date modified	Type	Size
docs	2022/12/12 10:11	File folder	
examples	2022/12/12 10:11	File folder	
host-manager	2022/12/12 10:11	File folder	
manager	2022/12/12 10:11	File folder	
ROOT	2022/12/12 10:11	File folder	

All our created applications will also be deployed in this folder (**webapps**). Next, we run the **examples** app.

- Go back to the root folder of Tomcat.

Name	Date modified	Type	Size
bin	2022/12/12 10:11	File folder	
conf	2022/12/12 10:11	File folder	
lib	2022/12/12 10:11	File folder	
logs	2023/01/19 14:47	File folder	
temp	2022/12/12 10:11	File folder	
webapps	2023/01/20 13:19	File folder	
work	2022/12/12 10:11	File folder	
LICENSE	2022/11/16 15:34	File	57 KB
NOTICE	2022/11/16 15:34	File	2 KB
RELEASE-NOTES	2022/11/16 15:34	File	8 KB
tomcat	2022/11/16 15:34	Icon	22 KB
Uninstall	2022/11/16 15:34	Application	86 KB

- Open the **bin** directory.

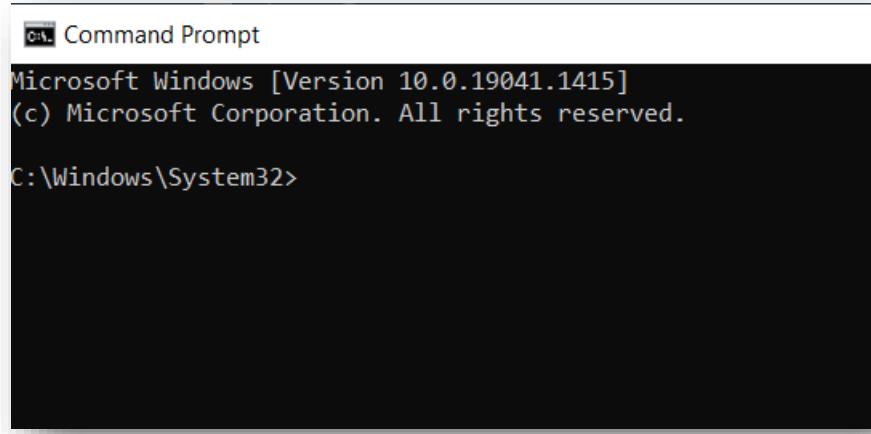
Name	Date modified	Type	Size
bootstrap	2022/11/16 15:34	Executable Jar File	36 KB
catalina	2022/11/16 15:34	Windows Batch File	17 KB
ciphers	2022/11/16 15:34	Windows Batch File	3 KB
configtest	2022/11/16 15:34	Windows Batch File	2 KB
digest	2022/11/16 15:34	Windows Batch File	3 KB
service	2022/11/16 15:34	Windows Batch File	9 KB
setclasspath	2022/11/16 15:34	Windows Batch File	4 KB
shutdown	2022/11/16 15:34	Windows Batch File	2 KB
startup	2022/11/16 15:34	Windows Batch File	2 KB
Tomcat8	2022/11/16 15:34	Application	142 KB
Tomcat8w	2022/11/16 15:34	Application	126 KB
tomcat-juli	2022/11/16 15:34	Executable Jar File	52 KB
tool-wrapper	2022/11/16 15:34	Windows Batch File	5 KB
version	2022/11/16 15:34	Windows Batch File	2 KB

There are many applications in this folder. We are going to use the **startup** application to **start** Tomcat. The **shutdown** application will be used to **stop** Tomcat.

- Copy the address of the **bin** directory.

C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin

- Open the command line.

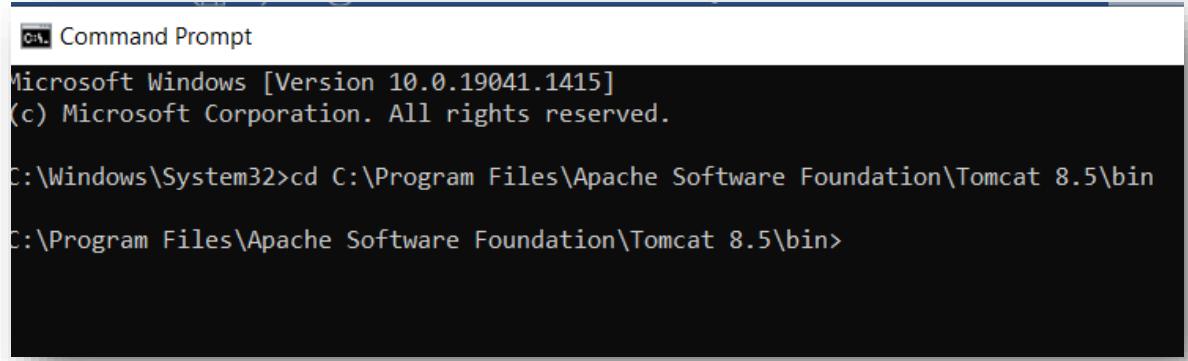


```
Command Prompt
Microsoft Windows [Version 10.0.19041.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>
```

- Change the directory to the **bin** directory of Tomcat by typing the following command:

cd C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin



```
Command Prompt
Microsoft Windows [Version 10.0.19041.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin

C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>
```

- Invoke the **startup** application by typing “**startup**”

```

C:\ Command Prompt
Microsoft Windows [Version 10.0.19041.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin
C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>startup
Using CATALINA_BASE: "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_HOME: "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_TMPDIR: "C:\Program Files\Apache Software Foundation\Tomcat 8.5\temp"
Using JRE_HOME: "C:\Program Files\Java\jdk1.8.0_111"
Using CLASSPATH: "C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\bootstrap.jar;C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\tomcat-juli.jar"
Using CATALINA_OPTS: ""

C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>

```

Tomcat

```

eb application directory [C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\docs]
20-Jan-2023 15:01:57.194 WARNING [localhost-startStop-1] org.apache.catalina.util.SessionIdGeneratorBase.createSecureRandom Creation of SecureRandom instance for session ID generation using [SHA1PRNG] took [212] milliseconds.
20-Jan-2023 15:01:57.242 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application directory [C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\docs] has finished in [578] ms
20-Jan-2023 15:01:57.242 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory [C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\examples]
20-Jan-2023 15:01:57.923 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application directory [C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\examples] has finished in [681] ms

```

- Open the browser and type the following url:

<http://localhost:8080/examples>



- Click on the **Servlets examples** link.

Servlet Examples with Code

This is a collection of examples which demonstrate some of the more frequently used parts of the Servlet API. Familiarity with the Java(tm) Programming Language is assumed. These examples will only work when viewed via an http URL. They will not work if you are viewing these pages via a "file://" URL. Please refer to the *README* file provided with this Tomcat release regarding how to configure and start the provided web server.

Wherever you see a form, enter some data and see how the servlet reacts. When playing with the Cookie and Session Examples, jump back to the Headers Example to see exactly what your browser is sending the server.

To navigate your way through the examples, the following icons will help:

Execute the example	Source
Look at the source code for the example	Source
Return to this screen	Source

Tip: To see the cookie interactions with your browser, try turning on the "notify when setting a cookie" option in your browser preferences. This will let you see when a session is created and give some feedback when looking at the cookie demo.

Hello World	Execute	Source
Request Info	Execute	Source
Request Headers	Execute	Source
Request Parameters	Execute	Source
Cookies	Execute	Source
Sessions	Execute	Source

- Click on the **Execute** link corresponding to **HelloWorld**.



The **HelloWorldExample** app simply prints out the message “**Hello World!**”

- Go back.

Servlet Examples with Code

This is a collection of examples which demonstrate some of the more frequently used parts of the Servlet API. Familiarity with the Java(tm) Programming Language is assumed.

These examples will only work when viewed via an http URL. They will not work if you are viewing these pages via a "file:///..." URL. Please refer to the *README* file provide with this Tomcat release regarding how to configure and start the provided web server.

Wherever you see a form, enter some data and see how the servlet reacts. When playing with the Cookie and Session Examples, jump back to the Headers Example to see exactly what your browser is sending the server.

To navigate your way through the examples, the following icons will help:

- Execute the example
- Look at the source code for the example
- Return to this screen

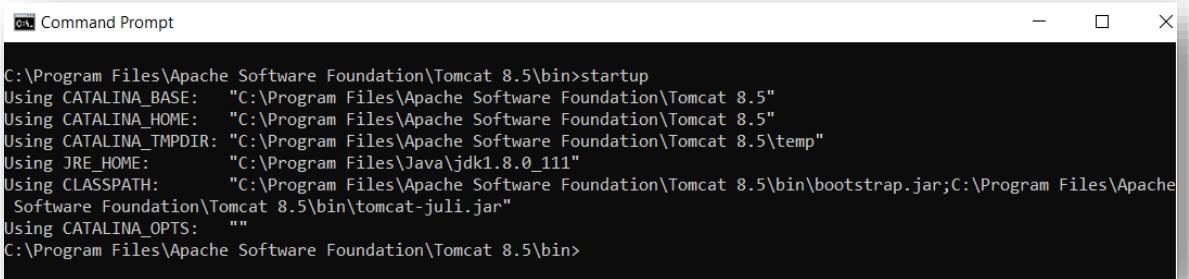
Tip: To see the cookie interactions with your browser, try turning on the "notify when setting a cookie" option in your browser preferences. This will let you see when a session is created and give some feedback when looking at the cookie demo.

Hello World	Execute	Source
Request Info	Execute	Source
Request Headers	Execute	Source
Request Parameters	Execute	Source
Cookies	Execute	Source
Sessions	Execute	Source

- Click on the **Source** link of **HelloWorld** to view the source code of the application.



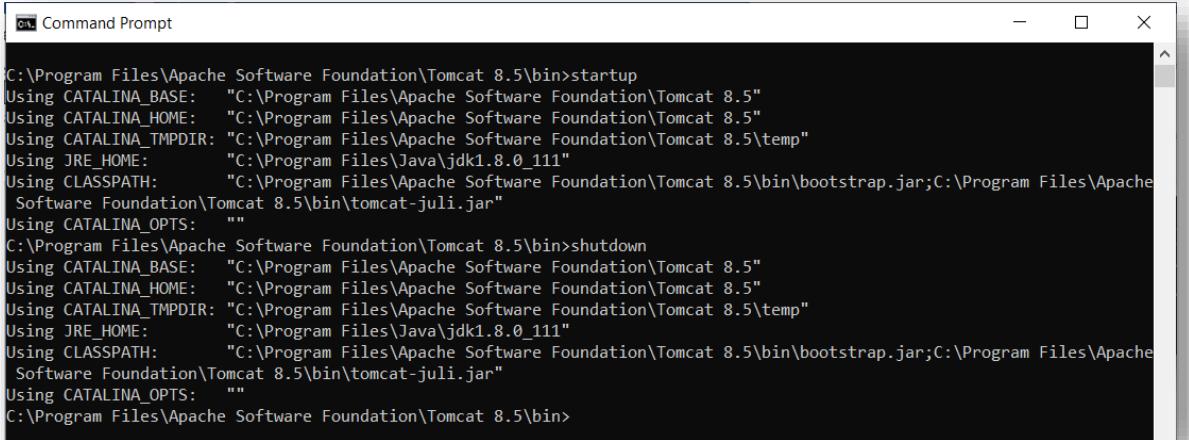
- Go back to the command line.



```
C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>startup
Using CATALINA_BASE:  "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_HOME:  "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_TMPDIR: "C:\Program Files\Apache Software Foundation\Tomcat 8.5\temp"
Using JRE_HOME:        "C:\Program Files\Java\jdk1.8.0_111"
Using CLASSPATH:       "C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\bootstrap.jar;C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\tomcat-juli.jar"
Using CATALINA_OPTS:   ""

C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>
```

- Launch the shutdown application by entering “**shutdown**” to stop Tomcat.



```
C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>startup
Using CATALINA_BASE:  "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_HOME:  "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_TMPDIR: "C:\Program Files\Apache Software Foundation\Tomcat 8.5\temp"
Using JRE_HOME:        "C:\Program Files\Java\jdk1.8.0_111"
Using CLASSPATH:       "C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\bootstrap.jar;C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\tomcat-juli.jar"
Using CATALINA_OPTS:   ""

C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>shutdown
Using CATALINA_BASE:  "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_HOME:  "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_TMPDIR: "C:\Program Files\Apache Software Foundation\Tomcat 8.5\temp"
Using JRE_HOME:        "C:\Program Files\Java\jdk1.8.0_111"
Using CLASSPATH:       "C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\bootstrap.jar;C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\tomcat-juli.jar"
Using CATALINA_OPTS:   ""

C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>
```

1.9 DIY (Do It Yourself)

In this DIY we want you to run some of the web applications under the **examples** folder of **Tomcat**.

Task #1

Launch Tomcat and run all the web applications under the **examples** folder of **webapps**.

1.10 Conclusion

In this chapter we managed to introduce the student to INT316D module and the JEE environment. We referred the student to appendices A – D on how to download and install the software packages needed in this module. We also demonstrated how to start and stop Tomcat. We deployed the example web application on Tomcat and executed it using the browser.

In the next chapter we are going to introduce the student to the concept of MVC (Model View Controller). Thank you very much for having taken time to go through this chapter. Enjoy the rest of the day and God bless you.

2 MVC

In this chapter we introduce the student to MVC, a design pattern. A design pattern is a manner in which software is developed. We will explain the concept and provide examples to illuminate it further.

2.1 What is MVC?

MVC is an acronym that stands for **Model View Controller**. This a software design pattern, it tells us how software should be developed. Through MVC, software is categorized into three components, namely:

- Model;
- View; and
- Control.

2.1.1 Models

Models are components that are used to represent objects ("things") and process them. Models are responsible for the business logic of the application. Examples of models are your simple **Java classes (POJOs)**, **interfaces**, and **EJBs**.

2.1.2 Views

Views are components that allow our applications to interact with the user in an easy manner. They perform input/output operations by allowing a user to provide data to an application through a user interface, and also display outcome through the same user interface. Examples of views are **HTML** pages and **JSPs**.

The difference between an HTML page and a JSP is that an HTML page works with static data. Static data is data that is not changing, it is known before hand. An example of static data is the sun, it never changes, it keeps on shining. On the other JSP works with dynamic data. Dynamic data is data that is not constant, it changes depending on what is requested. An example of dynamic data is the room temperature. This values is not static, it changes frequently.

2.1.3 Controllers

Controllers are components that control the flow of data in an application. They receive data from views and thereafter decide what to do with the data. They can pass data to models for processing or simply pass the data to views for display. An example of a controller is a **Servlet**.

2.2 Examples

In this section we are going to do two examples. The purpose of the examples is to demonstrate to the student how to design a solution to a web problem. So, given a problem statement, we will do three things, namely:

- Discuss the flow of our solution.
- Identify MVC components emanating from the discussion.
- Show the ultimate project structure which will mainly consist of the identified components.

2.2.1 Example 1

Problem statement

Say we want to create a web application that will display a personalised greeting based on the name of the user. The application must allow a user to enter her/his name, and upon receiving the input, it must generate and display a personalised greeting such as “**Hello X. Welcome to the world of Web Applications Development.**”, where X is a placeholder for the entered name.

To do

Design a solution to the problem. The solution must entail a discussion of the program flow, identification of MVC components which emanate from the discussion, and a depiction of the project structure which shows the identified components.

Program flow

The program needs a home or landing page which will welcome users. This page will have static content. As a result an HTML page will be used for this purpose. The page will also have a link to another page that will allow users to enter their names. This too can be an HTML page or a JSP. The page will have a link to a controller which will determine the flow of the data.

The controller will be a servlet file. The servlet will call a model to generate a personalised greeting. The servlet will then forward the personalised greeting to a JSP for viewing. The JSP is the right component for this purpose because the generated data is dynamic, not static. The JSP will display the dynamic data. Lastly, the page will have a link to the home page.

MVC components

Consequently our design will consist of the following MVC components:

Views

- index.html → a home page.
- name_entry.jsp → page that allows a user to enter her/his name.
- personalised_greeting.jsp → page that displays a personalised greeting.

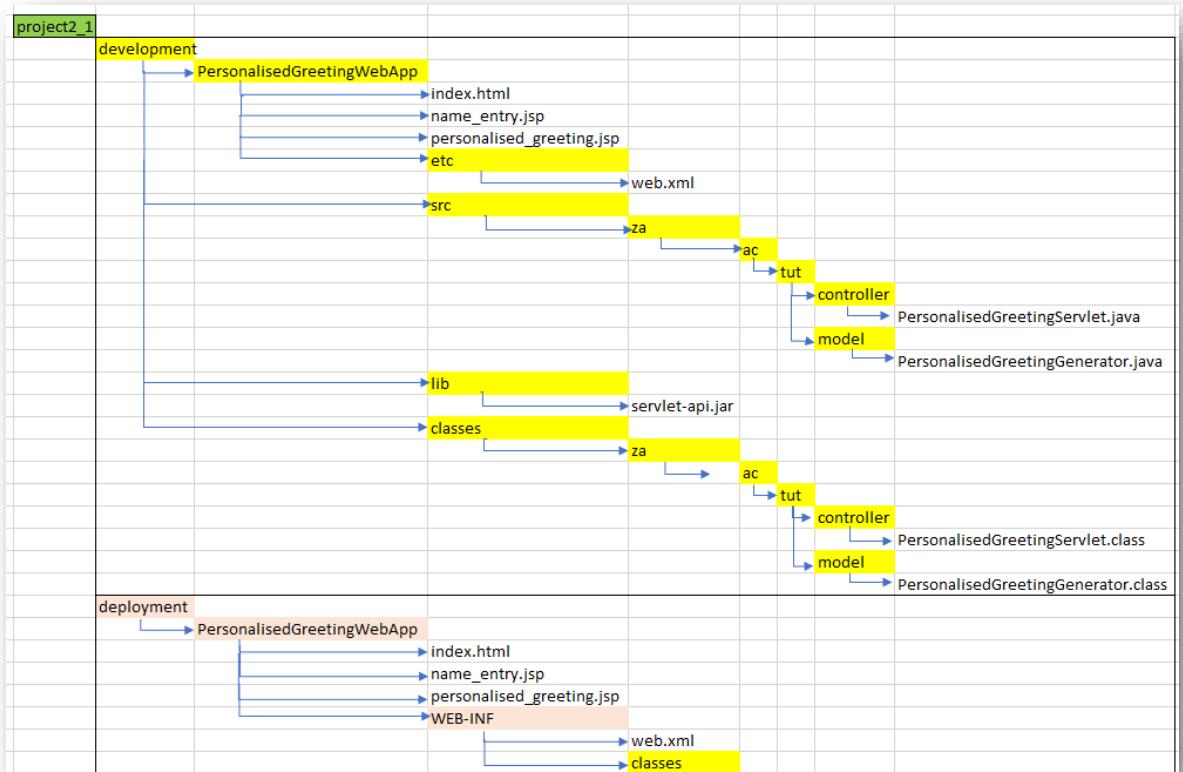
Controller

- PersonalisedGreetingServlet.java → controls the flow of greeting data.

Model

- PersonalisedGreetingGenerator.java → generates the personalised greeting.

Project structure



Description of the project's structure:

We can notice the following from the above figure:

- The name of the project is **project2_1**. The **2** stands for chapter **2** and the **1** for **example number 1**. So this is the first example of chapter 2.
- The project has two main folders, **development** and **deployment**.
- In the development directory we keep code that is under construction. After we are done, we then transfer it to the **deployment** folder. The code in the deployment folder is code that is ready to be deployed in the **Web Server**, that is **Tomcat**.
- The contents of the development folder are copied to the **deployment** folder. There are two major changes in the deployment folder. The **etc** folder is changed to **WEB-INF**, and in this folder we keep the **web.xml** file, and the **classes** folder together with its contents from the development folder.

Description of the project's files:

- The name of the web application is **PersonalisedGreetingWebApp**.
- The web application has six types of files, namely:
 - html**
 - The **index.html** file is the landing page. It has a link to the **name_entry.jsp** file.
 - jsp**
 - The **name_entry.jsp** file allows a user to enter her/his name.
 - The **personalised_greeting.jsp** reads the greeting from the **request** object and displays it.
 - servlet**
 - The **PersonalisedGreetingServlet.java** reads parameter data from the **request** object, generates a personalised greeting through a model and then forwards it to the **personalised_greeting.jsp** file.
 - model**
 - The **PersonalisedGreetingGenerator.java** generates a personalised greeting and sends it back to the servlet.
 - xml**
 - The **web.xml** file is a configuration file of the web application. It is also known as a Deployment Descriptor (DD) file. It describes the web application through XML elements. This file is read by the web server.
 - jar**
 - The **servlet-api.jar** file is a Servlet API. We need it to compile and run the servlets.

2.2.2 Example 2

Problem statement

Say we want to create a web application that will determine and display the sum of two numbers. The application must allow the user to enter the two numbers. Design a solution to the problem.

To do

You are required to design a solution for the given problem. The solution must entail a discussion of the program flow, identification of MVC components which emanate from the discussion, and a depiction of the project's structure which shows the identified components.

Program flow

The program needs a home or landing page which will welcome users. This page will have static content. As a result an HTML page will be used for this purpose. The page will also have a link to another page that will allow users to enter two numbers for summation. This too can be an HTML page or a JSP. The page will have a link to a controller which will determine the flow of the data.

The controller will be a servlet file. The servlet will call a model to determine and return the sum of the two numbers. The servlet will then forward the sum to a JSP for viewing. The JSP will display the dynamic data. Lastly, the page will have a link to the home page.

MVC components

Consequently our design will consist of the following MVC components:

Views

- index.html → a home page.
- numbers.jsp → the page allows users to enter two numbers.
- sum.jsp → the page that displays the sum.

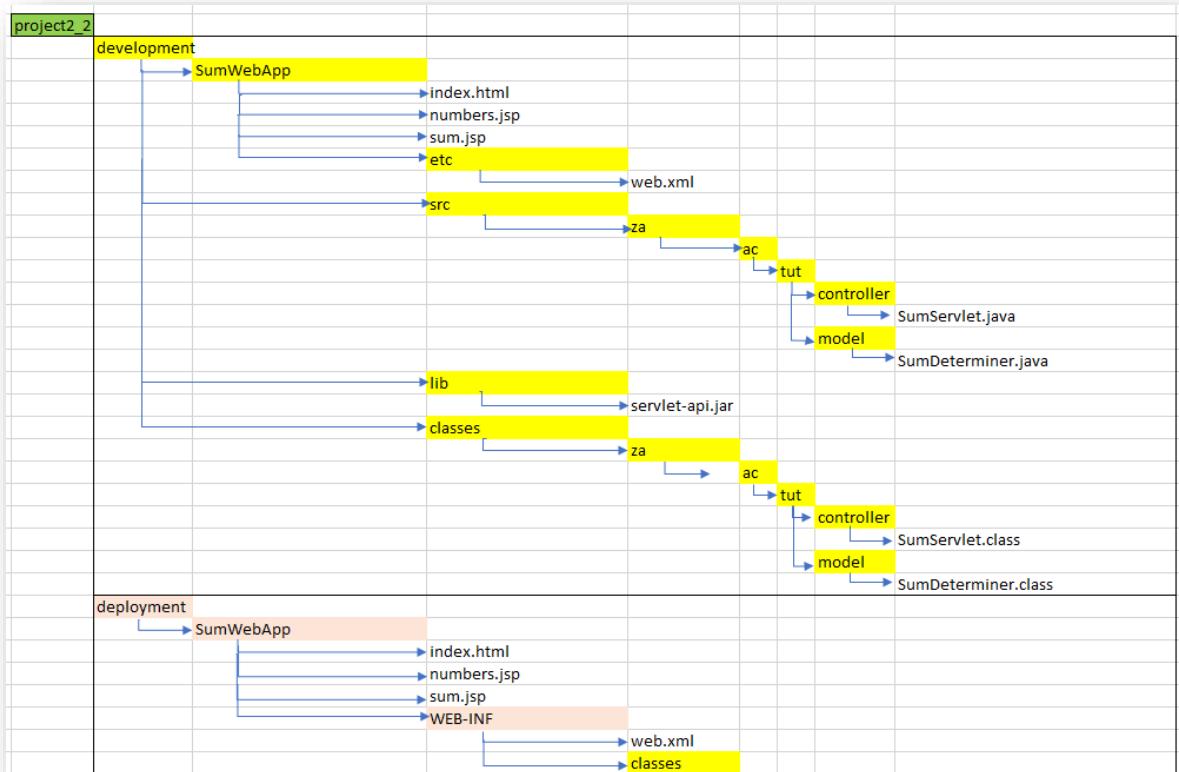
Controller

- SumServlet.java → controls the flow of sum data.

Model

- SumDeterminer.java → determines and returns sum.

Project structure



Description of the project's structure:

We can notice the following from the above figure:

- The name of the project is **project2_2**. The **2** stands for chapter **2** and the **2** for **example number 2**. So this is the second example of chapter **2**.
- The project has two main folders, **development** and **deployment**.
- In the development directory we keep code that is under construction. After we are done, we then transfer it to the **deployment** folder. The code in the deployment folder is code that is ready to be deployed in the **Web Server**, that is **Tomcat**.
- The contents of the development folder are copied to the **deployment** folder. There are two major changes in the deployment folder. The **etc** folder is changed to **WEB-INF**, and in this folder we keep the **web.xml** file, and the **classes** folder together with its contents from the development folder.

Description of the project's files:

- The name of the web application is **SumWebApp**.
- The web application has six types of files, namely:

html

- The **index.html** file is the landing page. It has a link to the **name_entry.jsp** file.

jsp

- The **numbers.jsp** file allows a user to enter two numbers.
- The **sum.jsp** reads the sum values from the **request** object and displays them.

servlet

- The **SumServlet.java** reads parameter data from the **request** object, determines the sum through a model and then forwards it to the **sum.jsp** file.

model

- The **SumDeterminer.java** file determines and returns the sum of two numbers back to the servlet.

xml

- The **web.xml** file is a configuration file of the web application. It is also known as a Deployment Descriptor (DD) file. It describes the web application through XML elements. This file is read by the web server.

jar

- The **servlet-api.jar** file is a Servlet API. We need it to compile and run the servlets.

2.3 DIY (Do It Yourself)

In this DIY we want you to design a solution to a given problem using MVC.

Task #1

Say we want to create a web application that will read and display the current date and time from the computer. The application must allow the user to enter her/his name, and then respond with the current date and time. Design an MVC solution to the problem.

To do

You are required to create a design a solution for the given problem. The solution must entail a discussion of the program flow, identification of MVC components which emanate from the discussion, and a depiction of the project structure which shows the identified components.

Task #2

In the past two years the world was under attack by COVID-19. The pandemic drastically changed the way people led their lives. Suddenly people were expected to wear face masks in public, sanitise their hands and practice social distancing. Entry to public institutions such as schools also required people to have their temperatures checked. Temperatures above 38 degrees Celsius resulted in people being refused entry into public buildings and advised to seek medical help.

Though the pandemic has immensely subsided, and as a result regulations have been done away with, some private institutions still require people to observe the COVID-19 prevention protocols. People are expected to wear their face masks, sanitise hands, and take temperature measurements.

To do

Design an MVC solution for a private institution, XYZ, that will allow access into the premises of XYZ if a person is wearing a face mask, is willing to sanitise hands, and is having a temperature value not more than 38 degrees Celsius. If a person is not meeting these requirements, access must be denied. The reason for denying access must be clearly spelt out.

2.4 Conclusion

In this chapter, we managed to discuss the concept of MVC. We defined the concept, and gave appropriate examples to further illuminate the concept. In the next chapter we are going to introduce the JSP concept.

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

3 JSP

In this chapter we introduce the student to the concept of JSP, a web component that is used for presentation purposes.

3.1 What is JSP?

JSP is an acronym that stands for **Java Server Page**. A JSP is a server-side component, meaning that it resides in the container. It allows Java code to be included in HTML. This makes HTML dynamic.

For example, with normal HTML, we can't read the current time from the computer. But with the introduction of Java into HTML (JSP), we are able to work with dynamically generated data, not just static data.

3.2 Relationship between JSPs and Servlets

A JSP is actually a servlet. When a JSP is first called by the container, it is converted into an equivalent **Java source code**. The source code is **compiled** into **byte code** (class file). An instance of the class is **created** and **initialized**, and a **Servlet** is **created**.

3.3 Implicit objects of JSP

JSP has objects that are implicitly (indirectly) imported to it. The objects are readily available, by default to a JSP. Below are the objects:

- request
- response
- session
- exception
- application
- config
- page
- pageContext
- out

3.4 How to write a scriptlet and an expression in JSP?

A scriptlet is a piece of Java code written inside a JSP. We use the special braces, to denote Java code inside JSP. The open brace percentage sign, `<%`, shows the start of the scriptlet, and the percentage sign close brace, `%>`, signals the end of the scriptlet.

For example, if we want to retrieve some data from a session object, we can accomplish the task as follows:

```
<%
    String name = (String)session.getAttribute("name");
    Integer age = (Integer)session.getAttribute("age");
%>
```

The same with an expression, if we want to write a Java expression in JSP, we use the following special braces with an equal sign inside:

```
<%=%>
```

Say we want to determine and display the sum of two numbers, `num1` and `num2`. We can accomplish the task by doing the following:

```
<%= (num1 + num2)%>
```

3.5 How to declare variables, write comments and import packages in JSP?

To declare a variable in JSP, we use the following generic syntax:

```
<%! variableName=value; %>
```

Say we want to declare the a string variable to hold the **name** of a person as **Thato**, and an **age** variable to hold the value **18**. The declaration of the two variables will be done as follows:

```
<%!  
    String name = "Thato";  
    int age=18;  
%>
```

To write comments in JSP we use the following generic syntax:

```
<%-- my comment --%>
```

For example we can comment the declaration of the person variables as follows:

```
<%!  
    <%-- name of the person --%>  
    String name = "Thato";  
    <%-- age of the person --%>  
    int age=18;  
%>
```

The generic syntax for importing packages into a JSP file is as follows:

```
<% @page importKeyword="fully-qualified-class-name"%>
```

If you are importing more than one class, you will have the following:

```
<% @page importKeyword="fully-qualified-class-name1, fully-qualified-class-name2,.."%>
```

For example, say we want to import both the **DecimalFormat** found in the **java.text** package and the **Random** class found in the **java.util** package. Then we will have the following directive:

```
<% @page import = "java.text.DecimalFormat, java.util.Random"%>
```

3.6 Example

Problem statement

Create a web application that will determine and display the **sum** of two numbers, **num1** and **num2**. The application must allow the user to enter the two numbers. Upon receiving the two numbers, the application must determine their sum and display.

To do

Create such an application. First discuss program flow, then identify MVC components which should emanate from the discussion, depict the project's structure of the solution which should show the identified components, and lastly do the actual coding.

Program flow

The program flow is as explained in example 1 of chapter 2. The only thing to note is that we won't use servlets and models in our implementation as yet. Everything will be done using HTML and JSPs.

MVC components

The MVC components are as determined in example 1 of chapter 2.

Project structure

The project structure is as shown in example 1 of chapter 2.

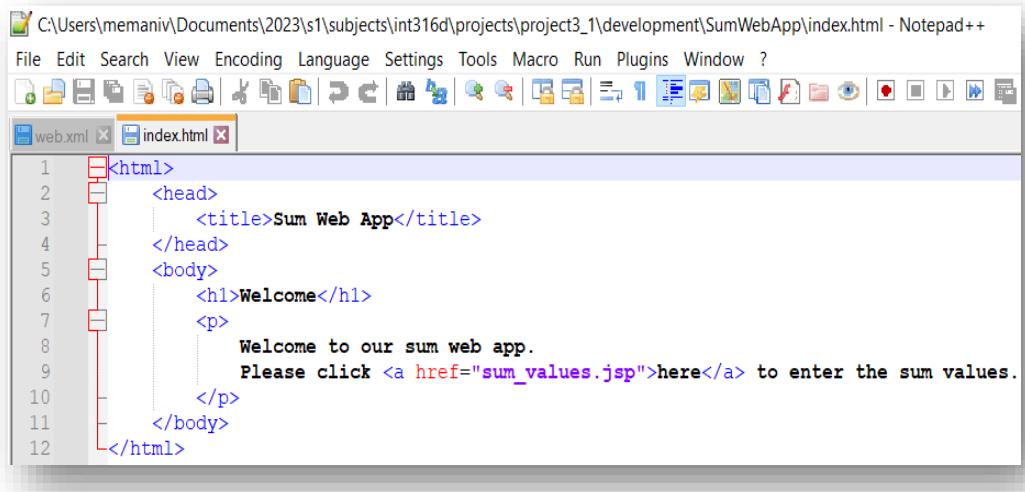
Implementation

In the next page we implement the project structure.

1. Create the files.

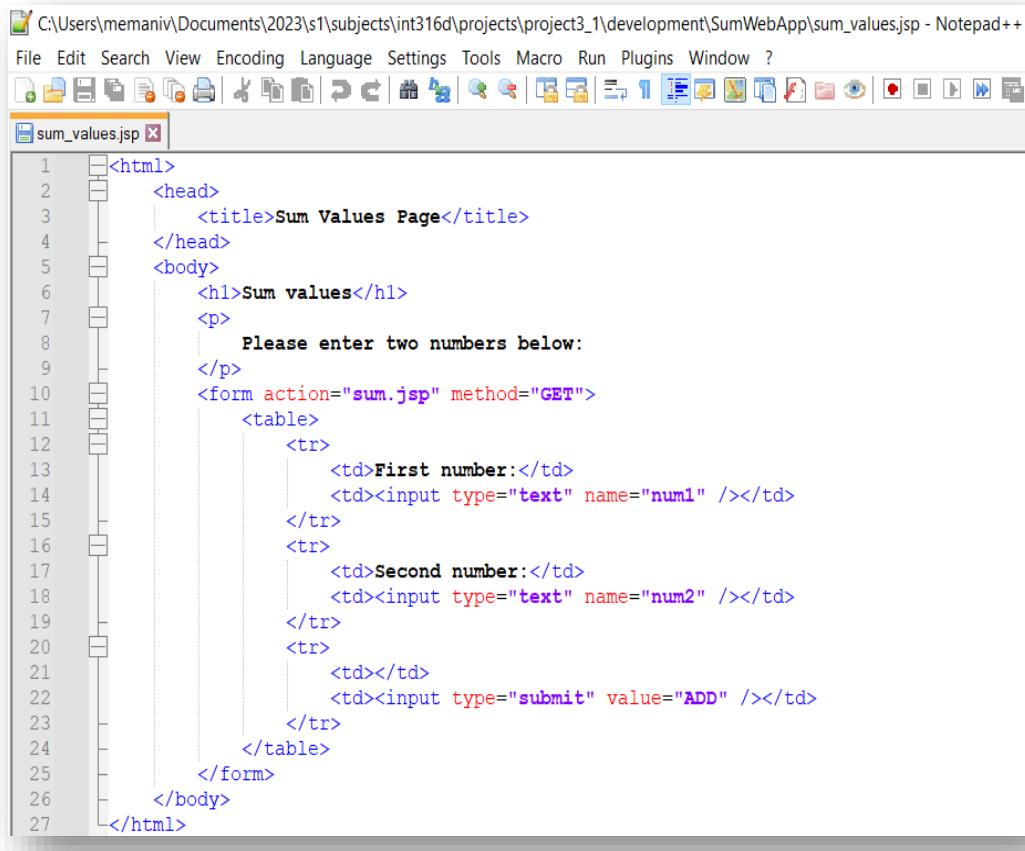
Create the files as follows:

- **html**
 - **index.html**



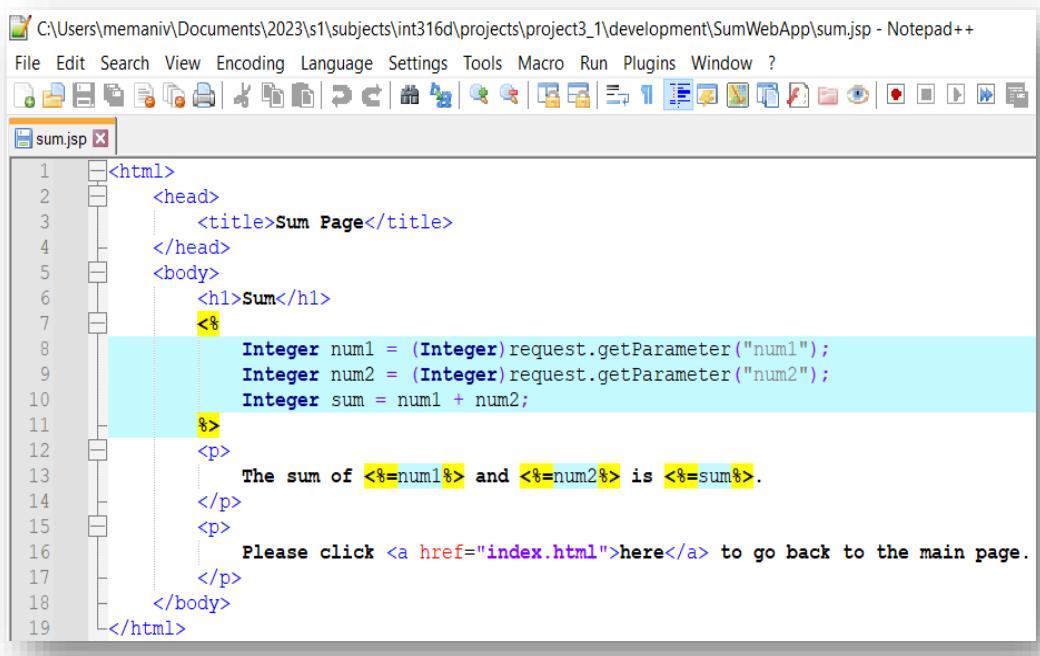
```
C:\Users\memaniv\Documents\2023\s1\subjects\int316d\projects\project3_1\development\SumWebApp\index.html - Notepad++  
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?  
index.html  
1 <html>  
2   <head>  
3     <title>Sum Web App</title>  
4   </head>  
5   <body>  
6     <h1>Welcome</h1>  
7     <p>  
8       Welcome to our sum web app.  
9       Please click <a href="sum_values.jsp">here</a> to enter the sum values.  
10    </p>  
11  </body>  
12 </html>
```

- **jsp**
 - **sum_values.jsp**



```
C:\Users\memaniv\Documents\2023\s1\subjects\int316d\projects\project3_1\development\SumWebApp\sum_values.jsp - Notepad++  
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?  
sum_values.jsp  
1 <html>  
2   <head>  
3     <title>Sum Values Page</title>  
4   </head>  
5   <body>  
6     <h1>Sum values</h1>  
7     <p>  
8       Please enter two numbers below:  
9     </p>  
10    <form action="sum.jsp" method="GET">  
11      <table>  
12        <tr>  
13          <td>First number:</td>  
14          <td><input type="text" name="num1" /></td>  
15        </tr>  
16        <tr>  
17          <td>Second number:</td>  
18          <td><input type="text" name="num2" /></td>  
19        </tr>  
20        <tr>  
21          <td></td>  
22          <td><input type="submit" value="ADD" /></td>  
23        </tr>  
24      </table>  
25    </form>  
26  </body>  
27 </html>
```

▪ sum.jsp

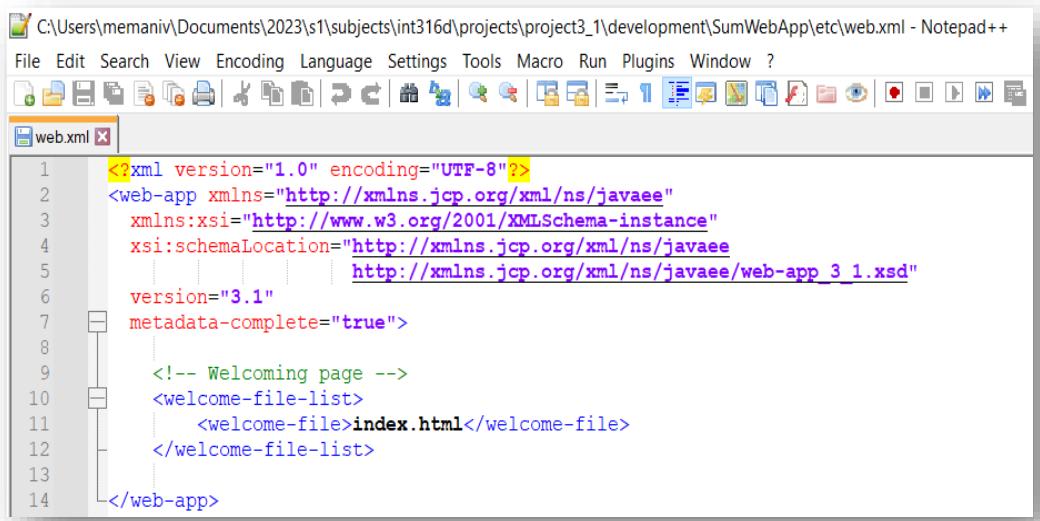


The screenshot shows the Notepad++ editor with the file "sum.jsp" open. The code is a JSP page that displays the sum of two integers. It includes an HTML structure with a title and body, a scriptlet block to calculate the sum, and two plain text blocks containing descriptive messages and a link to the main page.

```
<html>
    <head>
        <title>Sum Page</title>
    </head>
    <body>
        <h1>Sum</h1>
        <%>
        Integer num1 = (Integer) request.getParameter("num1");
        Integer num2 = (Integer) request.getParameter("num2");
        Integer sum = num1 + num2;
        <%>
        <p>
            The sum of <%=num1%> and <%=num2%> is <%=sum%>.
        </p>
        <p>
            Please click <a href="index.html">here</a> to go back to the main page.
        </p>
    </body>
</html>
```

○ XML

▪ web.xml

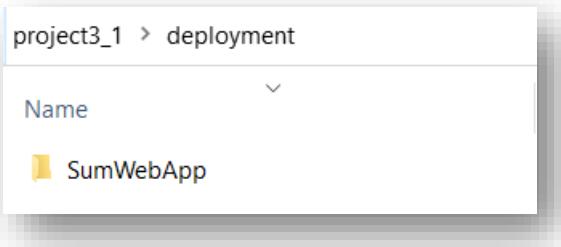


The screenshot shows the Notepad++ editor with the file "web.xml" open. This is a Java Web Application descriptor file. It defines a web application with version 3.1, a welcome file list containing "index.html", and specifies the XML schema location for validation.

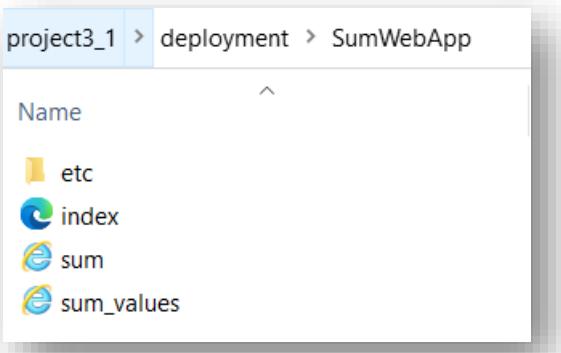
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                               http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
          version="3.1"
          metadata-complete="true">
    <!-- Welcoming page -->
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
    </welcome-file-list>
</web-app>
```

2. Prepare for deployment

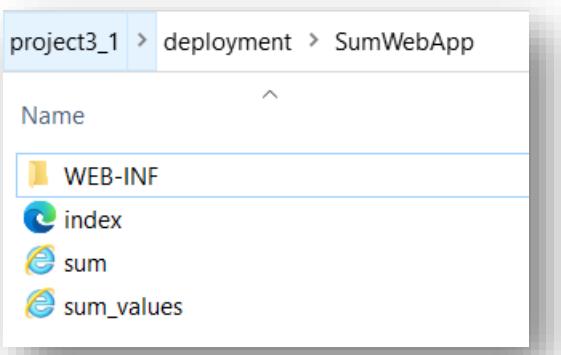
Copy the **SumWebApp** folder and paste it under the **deployment** folder.



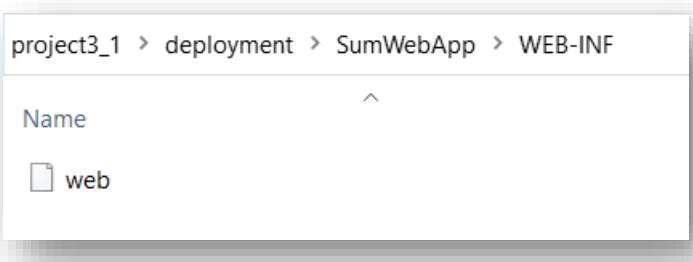
Expand **SumWebApp**.



Change **etc** to **WEB-INF**.

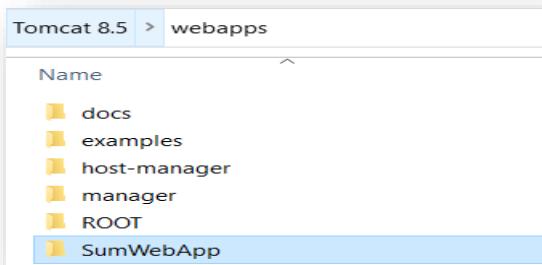


Expand **WEB-INF**.

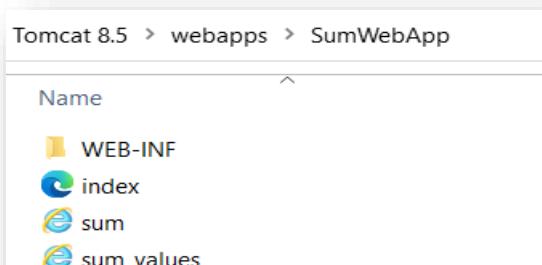


3. Deploy the web application

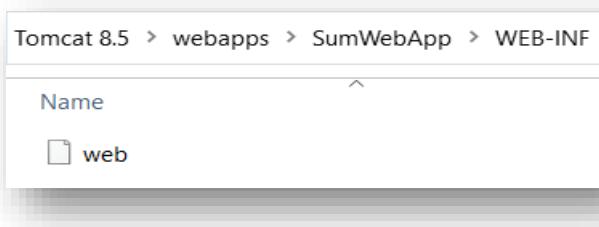
Copy the **SumWebApp** folder under deployment and paste it on the **webapps** folder of **Tomcat**.



Expand **SumWebApp**.



Expand **WEB-INF**.



Go to the **bin** directory of **Tomcat**.

Name	Date modified	Type	Size
bootstrap	2022/11/16 15:34	Executable Jar File	36 KB
catalina	2022/11/16 15:34	Windows Batch File	17 KB
ciphers	2022/11/16 15:34	Windows Batch File	3 KB
configtest	2022/11/16 15:34	Windows Batch File	2 KB
digest	2022/11/16 15:34	Windows Batch File	3 KB
service	2022/11/16 15:34	Windows Batch File	9 KB
setclasspath	2022/11/16 15:34	Windows Batch File	4 KB
shutdown	2022/11/16 15:34	Windows Batch File	2 KB
startup	2022/11/16 15:34	Windows Batch File	2 KB
Tomcat8	2022/11/16 15:34	Application	142 KB
Tomcat8w	2022/11/16 15:34	Application	126 KB
tomcat-juli	2022/11/16 15:34	Executable Jar File	52 KB
tool-wrapper	2022/11/16 15:34	Windows Batch File	5 KB
version	2022/11/16 15:34	Windows Batch File	2 KB

Copy the address of the **bin** directory.

C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin

Launch the command line.

```
Command Prompt
Microsoft Windows [Version 10.0.19041.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>
```

Change location to the **bin** directory of **Tomcat**.

```
Command Prompt
Microsoft Windows [Version 10.0.19041.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin

C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>
```

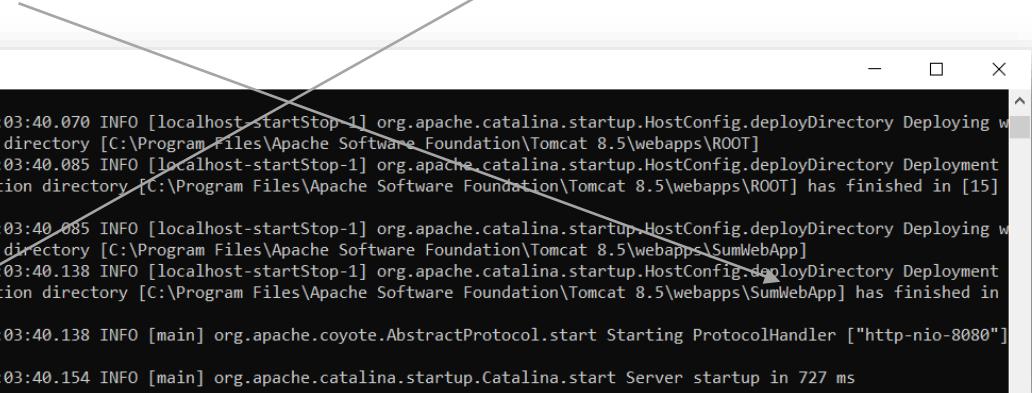
Start Tomcat by typing "**startup**" on the command line and then click the **Enter** key.

```
Command Prompt
Microsoft Windows [Version 10.0.19041.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin
C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>startup
Using CATALINA_BASE: "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_HOME: "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_TMPDIR: "C:\Program Files\Apache Software Foundation\Tomcat 8.5\temp"
Using JRE_HOME: "C:\Program Files\Java\jdk1.8.0_111"
Using CLASSPATH: "C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\bootstrap.jar;C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\tomcat-juli.jar"
Using CATALINA_OPTS: ""
C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>
```

Output:

The **SumWebApp** application is deployed in **53 ms**.



```
[Tomcat] 6] ms  
21-Jan-2023 16:03:40.070 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory [C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\ROOT]  
21-Jan-2023 16:03:40.085 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application directory [C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\SumWebApp] has finished in [15] ms  
21-Jan-2023 16:03:40.085 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory [C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\SumWebApp]  
21-Jan-2023 16:03:40.138 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application directory [C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\SumWebApp] has finished in [53] ms  
21-Jan-2023 16:03:40.138 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["http-nio-8080"]  
21-Jan-2023 16:03:40.154 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in 727 ms
```

4. Run the web application

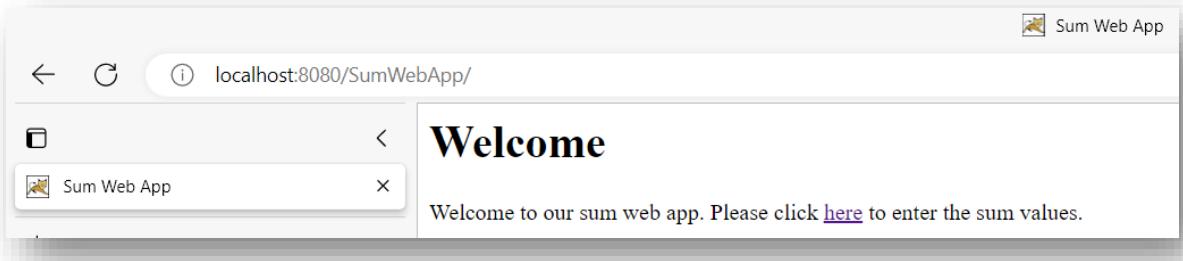
Open the browser and type:

<http://localhost:8080/SumWebApp>

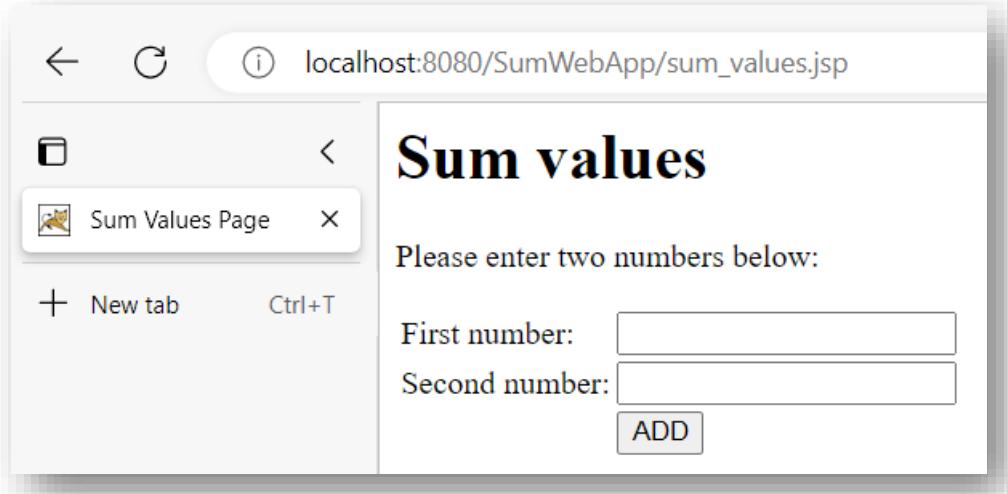
Note:

- **http**: is the protocol used to communicate between the browser and the Web Server, Tomcat.
- **localhost**: is the address of the machine that hosts the Web Server. In this instance the Web Server is running on the local machine.
- **8080**: is the port where the Web Server is running.
- **SumWebApp**: is the name of the web application deployed on the server that we want to talk to.

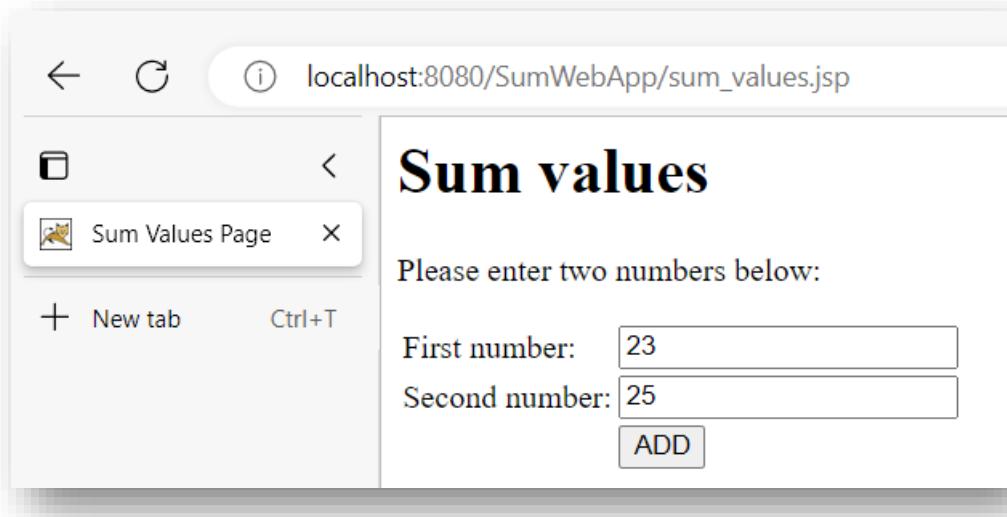
Output:



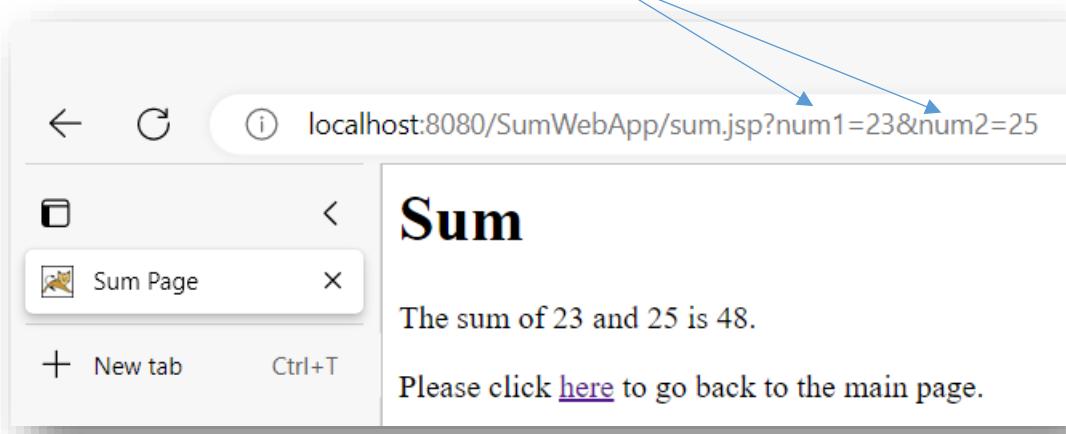
Click on the link.



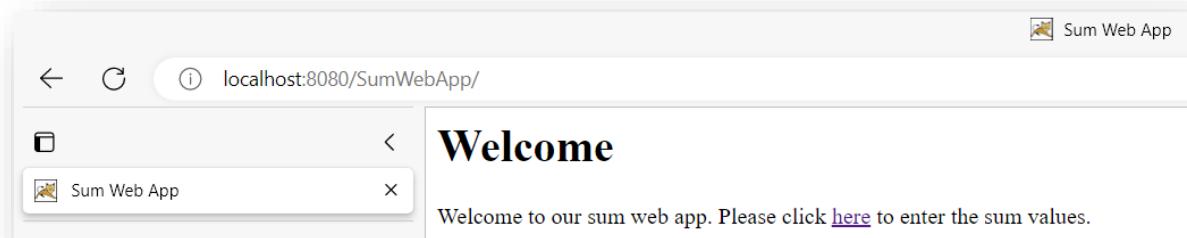
Enter the two numbers.



Click on the **ADD** button. Notice that the two parameters are now part of the url line.



Click on the link to go back to the main page.



Stop Tomcat

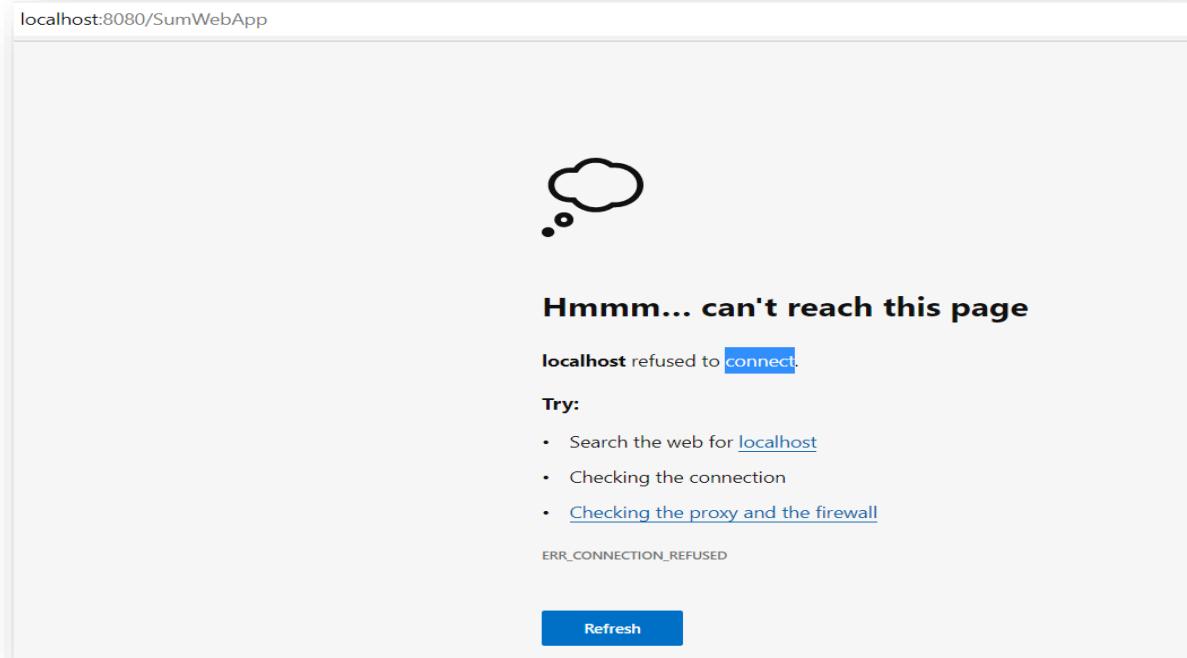
Stop Tomcat by typing "**shutdown**" on the command line and then click the **Enter** key.

```
C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>startup
Using CATALINA_BASE:  "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_HOME:  "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_TMPDIR:  "C:\Program Files\Apache Software Foundation\Tomcat 8.5\temp"
Using JRE_HOME:  "C:\Program Files\Java\jdk1.8.0_111"
Using CLASSPATH:  "C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\bootstrap.jar;C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\tomcat-juli.jar"
Using CATALINA_OPTS:  ""

C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>shutdown
Using CATALINA_BASE:  "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_HOME:  "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_TMPDIR:  "C:\Program Files\Apache Software Foundation\Tomcat 8.5\temp"
Using JRE_HOME:  "C:\Program Files\Java\jdk1.8.0_111"
Using CLASSPATH:  "C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\bootstrap.jar;C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\tomcat-juli.jar"
Using CATALINA_OPTS:  ""

C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>
```

Try to run the web application whilst the server is down. An error takes place.



3.7 DIY (Do It Yourself)

In this DIY we want you to develop web applications using HTML and JSP. The web application should be developed on Notepad++, deployed on Tomcat and executed through the browser.

Task #1

Create a web application called **HelloWorldWebApp** which will display a customised greeting. A user should be allowed to enter their name, and the app must then take the name value and generate a personalised greeting such as “**Hello X. Welcome to the world of web applications.**” where X is the entered name.

Task #2

Create a web application called **AgeCategoryWebApp** that will classify people according to their ages. The classification must be done according to the table below:

Age	Category
0 - 14	Child
15 - 24	Youth
25 - 64	Adult
65 and over	Senior

Task #3

In the past two years the world was under attack by COVID-19. The pandemic drastically changed the way people led their lives. Suddenly people were expected to wear face masks in public, sanitise their hands and practice social distancing. Entry to public institutions such as schools also required people to have their temperatures checked. Temperatures above 38 degrees Celsius resulted in people being refused entry into public buildings and advised to seek medical help.

Though the pandemic has immensely subsided, and as a result regulations have been done away with, some private institutions still require people to observe the COVID-19 prevention protocols. People are expected to wear their face masks, sanitise hands, and take temperature measurements.

Create a web application for a private institution, XYZ, that will allow access into the premises of XYZ if a person is wearing a face mask, is willing to sanitise hands, and

is having a temperature value not more than 38 degrees Celsius. If a person is not meeting these requirements, access must be denied. The reason for denying access must be clearly spelt out.

3.8 Conclusion

In this chapter we managed to introduce the student to the concept of JSP. In the next chapter we are going to introduce the student to Servlets. Thank you very much for having taken time to go through this chapter. Enjoy the rest of the day and God bless you.

4 Servlets

In this chapter, we introduce the student to the concept of servlets. A servlet is a web component that is used for presentation purposes. It servlet resides in the web container and services HTTP requests coming from the client.

4.1 What is a servlet?

A servlet is a web component used to service HTTP requests by the server. A client sends an HTTP request to the server, the server gets an appropriate servlet to service the request, the servlet services the request, and thereafter returns an HTTP response to the client. The figure below shows the exchange of messages between the client and server.



The server/container is responsible for the lifecycle of the servlet. This entails the following:

- The creation of a servlet.
- The loading of the servlet into memory. This is either done when the servlet is first requested or when the server reads the contents of the deployment descriptor (DD).
- The allocation of a servlet to a request in accordance to the client's request.
- The servicing of a request when the servlet is fully initialized.
- The end of life of a servlet.

The servlet technology is fully explained in a Java Specification Request (JSR) document called **JSR315**. The document can be found in the following website:

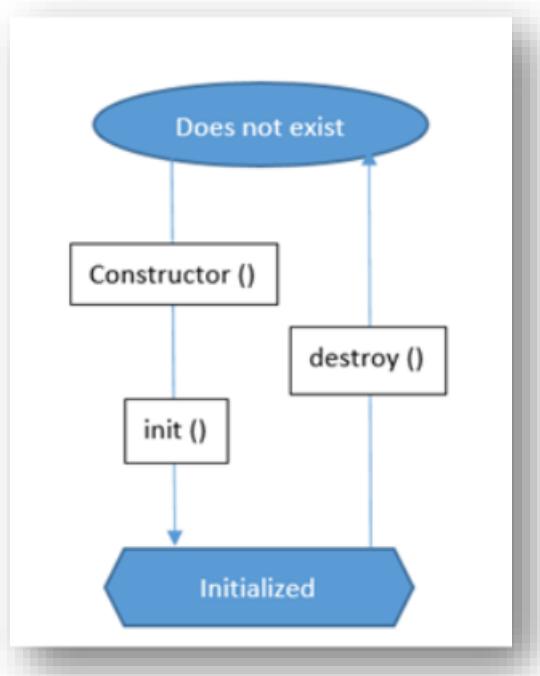
<http://jcp.org/en/jsr/detail?id=315>

4.2 Servlet life cycle

The servlet has two major life cycle stages, namely:

- The does not exist stage; and
- The initialized stage.

The figure below shows the two stages in the life cycle of a servlet.



In the "**Does not exist**" stage, the servlet is non-existent. When a servlet is either called by a client request or read from the deployment descriptor file, this triggers its creation and initialization. The container calls the no parameters constructor of the servlet to create the servlet. The `init()` method is then called to initialize the servlet.

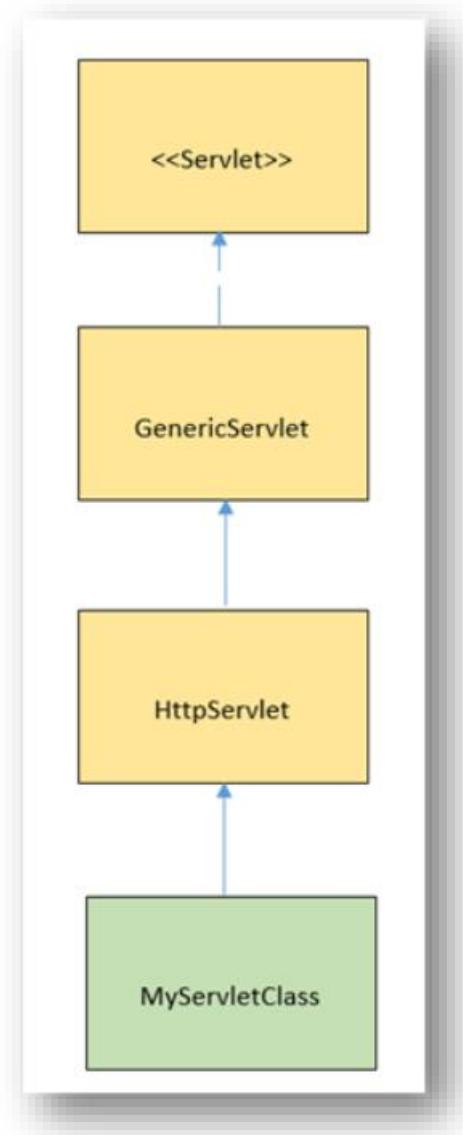
The servlet then enters its state of servicing requests, which is the **initialized** state. This is where the servlet spends most of its time. When the container decides to end the life of the servlet, it either destroys it by calling the `destroy()` method, or just takes the servlet to garbage collection.

4.3 How does a servlet work?

The web container receives an HTTP request message from a client (browser). The container sees that the request is for a specific servlet through the provided URL. The container calls the servlet and allocates it a thread to run. The container then creates and initializes two objects, **HttpServletResponse** and **HttpServletRequest**. It calls the **service** method of the servlet and passes to it the two objects as arguments. The service method calls the appropriate HTTP method (**doGet**, **doPost**) to service the request.

4.4 Servlet API

According to the JEE documentation, any class that extends the **HttpServlet** is a **Servlet**. The inheritance hierarchy of the **Servlet** is as follows:



The class you create, **MyServletClass**, becomes a servlet on the basis of inheriting the **HttpServlet** class which is found in the **javax.servlet.http** package. The HttpServlet class has the following methods that your servlet class needs to override, depending on the HTTP request of the client:

- doGet()
- doPost()
- doDelete()
- doPut() etc.

Mostly we work with the doGet() and doPost() method. If a client sends an HTTP get request, the servlet must override the doGet() methods. Alternatively, if the request is a post, the servlet must override the doPost() method. Below is the method summary of the **HttpServlet** class.

Method Summary		
All Methods	Instance Methods	Concrete Methods
Modifier and Type	Method and Description	
protected void	doDelete(HttpServletRequest req, HttpServletResponse resp)	Called by the server (via the service method) to allow a servlet to handle a DELETE request.
protected void	doGet(HttpServletRequest req, HttpServletResponse resp)	Called by the server (via the service method) to allow a servlet to handle a GET request.
protected void	doHead(HttpServletRequest req, HttpServletResponse resp)	Receives an HTTP HEAD request from the protected service method and handles the request.
protected void	doOptions(HttpServletRequest req, HttpServletResponse resp)	Called by the server (via the service method) to allow a servlet to handle a OPTIONS request.
protected void	doPost(HttpServletRequest req, HttpServletResponse resp)	Called by the server (via the service method) to allow a servlet to handle a POST request.
protected void	doPut(HttpServletRequest req, HttpServletResponse resp)	Called by the server (via the service method) to allow a servlet to handle a PUT request.
protected void	doTrace(HttpServletRequest req, HttpServletResponse resp)	Called by the server (via the service method) to allow a servlet to handle a TRACE request.
protected long	getLastModified(HttpServletRequest req)	Returns the time the HttpServletRequest object was last modified, in milliseconds since midnight January 1, 1970 GMT.
protected void	service(HttpServletRequest req, HttpServletResponse resp)	Receives standard HTTP requests from the public service method and dispatches them to the doXXX methods defined in this class.
void	service(ServletRequest req, ServletResponse res)	Dispatches client requests to the protected service method.

All of the above methods are called or invoked by the server when a specific method is requested by a client. The server acts as our “**main**” method.

The **HttpServlet** class inherits the **GenericServlet** class. **GenericServer** class defines methods that are independent of any communication protocol. This means this class, contrary to **HttpServlet**, supports any communication protocol such as ftp. Because of inheritance, the HttpServlet class inherits all the methods defined in GenericServlet.

Below is the summary of some of the **GenericServlet** class methods.

All Methods	Instance Methods	Abstract Methods	Concrete Methods
Modifier and Type	Method and Description		
void	<code>destroy()</code> Called by the servlet container to indicate to a servlet that the servlet is being taken out of service.		
String	<code>getInitParameter(String name)</code> Returns a String containing the value of the named initialization parameter, or <code>null</code> if the parameter does not exist.		
Enumeration<String>	<code>getInitParameterNames()</code> Returns the names of the servlet's initialization parameters as an Enumeration of String objects, or an empty Enumeration if the servlet has no initialization parameters.		
ServletConfig	<code>getServletConfig()</code> Returns this servlet's ServletConfig object.		
ServletContext	<code>getServletContext()</code> Returns a reference to the ServletContext in which this servlet is running.		
String	<code>getServletInfo()</code> Returns information about the servlet, such as author, version, and copyright.		
String	<code>getServletName()</code> Returns the name of this servlet instance.		
void	<code>init()</code> A convenience method which can be overridden so that there's no need to call <code>super.init(config)</code> .		
void	<code>init(ServletConfig config)</code> Called by the servlet container to indicate to a servlet that the servlet is being placed into service.		
void	<code>log(String msg)</code> Writes the specified message to a servlet log file, prepended by the servlet's name.		
void	<code>log(String message, Throwable t)</code>		

The **GenericServlet** class implements the **Servlet** interface. Below are the methods defined by the Servlet interface:

Method Summary		
All Methods	Instance Methods	Abstract Methods
Modifier and Type	Method and Description	
void	<code>destroy()</code> Called by the servlet container to indicate to a servlet that the servlet is being taken out of service.	
ServletConfig	<code>getServletConfig()</code> Returns a <code>ServletConfig</code> object, which contains initialization and startup parameters for this servlet.	
String	<code>getServletInfo()</code> Returns information about the servlet, such as author, version, and copyright.	
void	<code>init(ServletConfig config)</code> Called by the servlet container to indicate to a servlet that the servlet is being placed into service.	
void	<code>service(ServletRequest req, ServletResponse res)</code> Called by the servlet container to allow the servlet to respond to a request.	

4.5 Example 1

In this example we test some of the **Servlet** methods. We mainly look at the **ServletConfig** and the **ServletContext** interfaces. The **ServletConfig** is used to initialise parameters that we want to be available to a specific Servlet and JSP after the container has read the DD (Deployment Descriptor → web.xml) file. Through the **ServletContext** we are able to initialise parameters that will be available to all the files in the web application after loading of the DD file.

So in this example we are going to create a web application that is going to provide subject information to the student. Upon running the application, the following information will be displayed about INT316D:

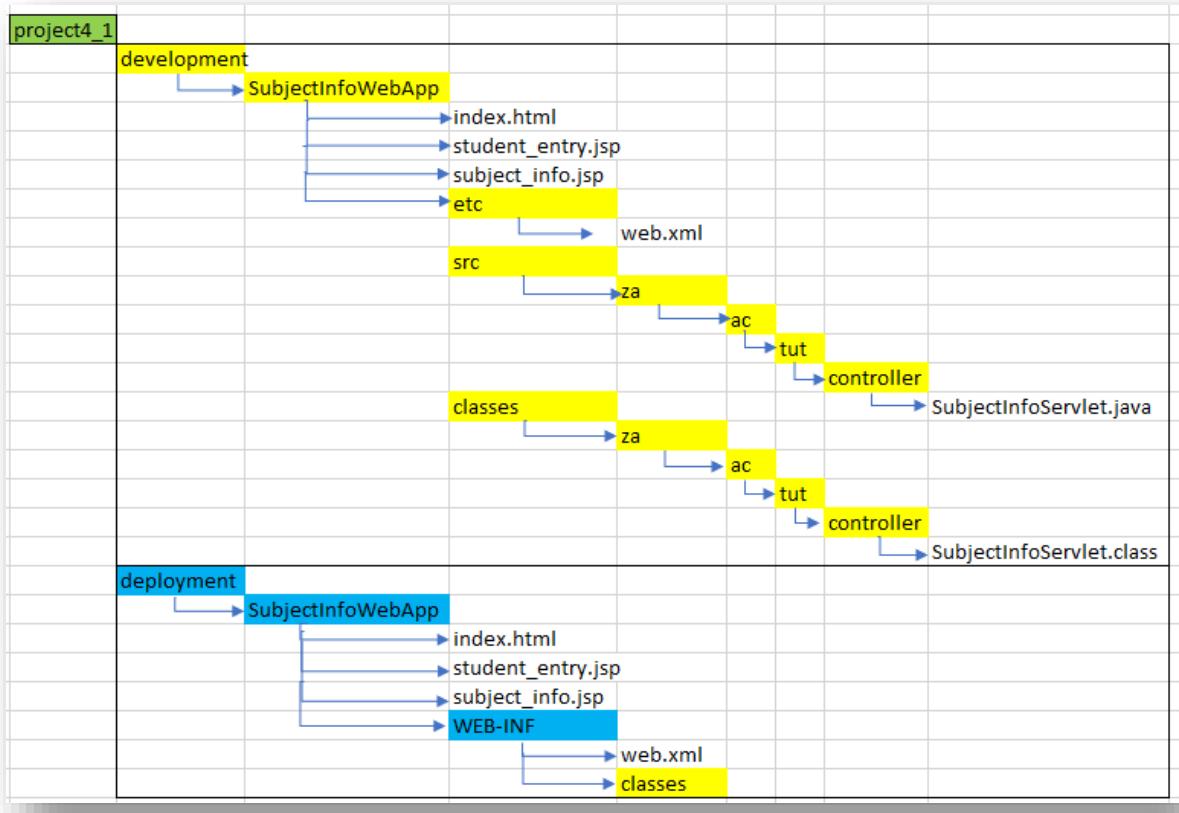
- Name and email address of the Subject Head;
- Name and code of the subject;
- Name and email address of the eMalahleni lecturer;
- Name and email address of the Polokwane lecturer; and
- Name and email address of the Soshanguve lecturer.

To demonstrate the difference between **ServletConfig (local variables)** and **ServletContext (global variables)**, the **name and email address of the Subject Head** will be accessed via the **ServletConfig**, and the rest through the **ServletContext**.

Solution

1. Project structure.

Below is the project structure which acts as the solution to the problem we are solving.



Description of the project's structure:

We can notice the following from the above figure:

- The name of the project is **project4_1**. The **4** stands for chapter **4** and the **1** stands for **example number 1**. So this is the first example of chapter 4.
- The project has two main folders, **development** and **deployment**.
- In the development directory we keep the code that is under construction. After we are done, we then transfer the code to the deployment folder. The code in the deployment folder is code that is ready to be deployed in the Web Server, that is Tomcat.
- The contents of the development folder are copied to the deployment folder. There are two major changes in the deployment folder, **etc** folder is changed to **WEB-INF**, and in this folder we keep the **DD** file, the **classes** folder together with its contents from the development folder.

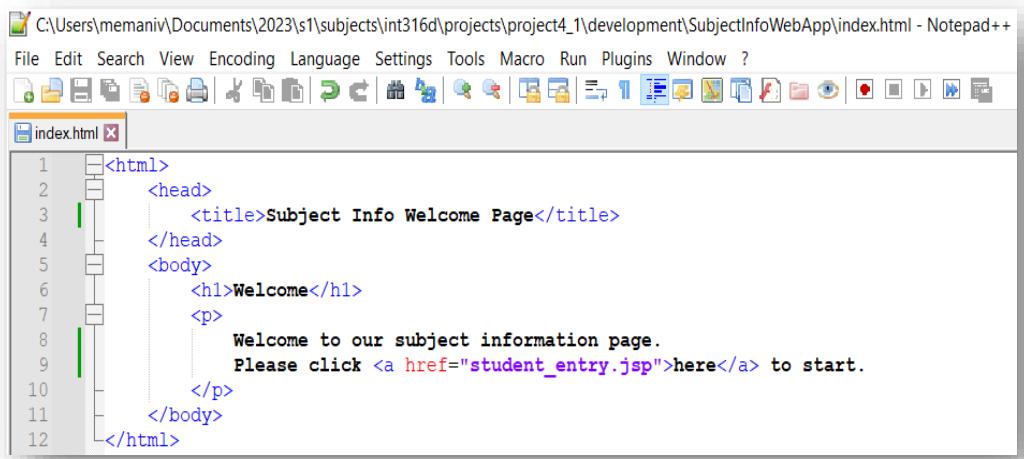
Description of the project's files:

- The name of the web application is **SubjectInfoWebApp**.
- The web application has three types of files, namely:
 - html**
 - The **index.html** file is the landing page. It has a link to the **student_entry.jsp** file.
 - jsp**
 - The **student_entry.jsp** file allows a student to enter her/his name.
 - The **student_info.jsp** reads subject information from the **request**, **ServletConfig** and **ServletContext** objects. The information is then displayed.
 - servlet**
 - The **SubjectInfoServlet.java** reads data from the request and ServletConfig objects, and displays it. It then forwards execution to the **student_info.jsp** file.
 - xml**
 - The **web.xml** file is the configuration file of the web application. It defines the **welcome file**, **servlet**, **servlet initialisation parameters**, and **servlet context parameters**.

2. Create the files.

Create the files as follows:

- **html**
 - **index.html**

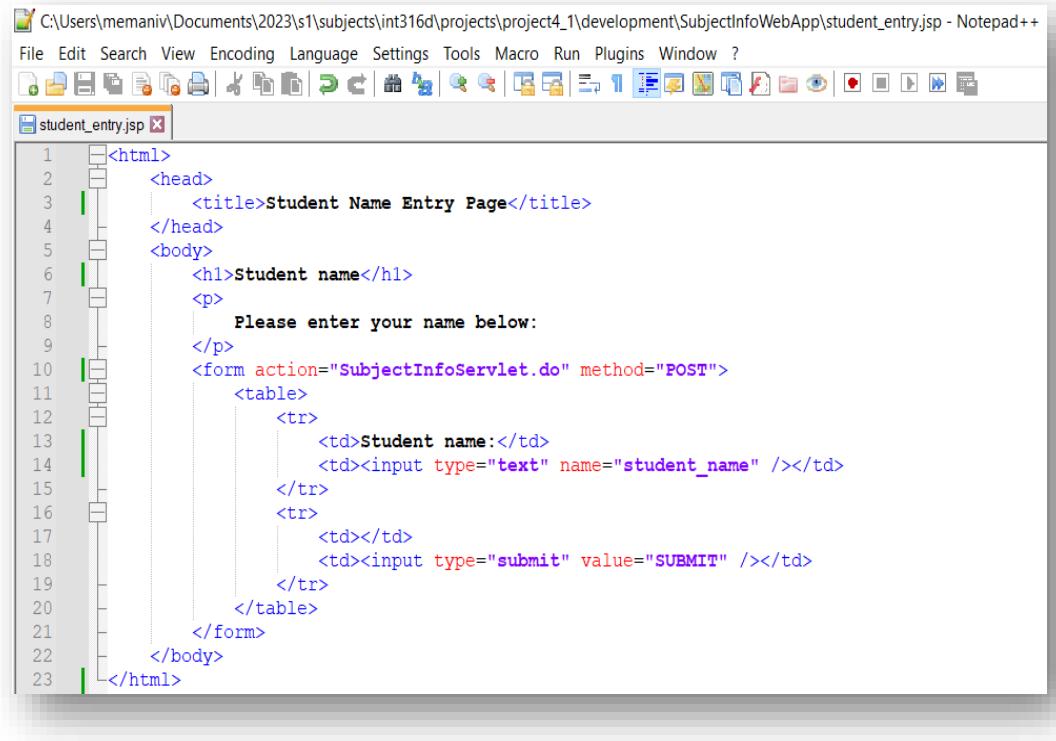


The screenshot shows the Notepad++ editor window with the file 'index.html' open. The code is as follows:

```
<html>
  <head>
    <title>Subject Info Welcome Page</title>
  </head>
  <body>
    <h1>Welcome</h1>
    <p>
      Welcome to our subject information page.
      Please click <a href="student_entry.jsp">here</a> to start.
    </p>
  </body>
</html>
```

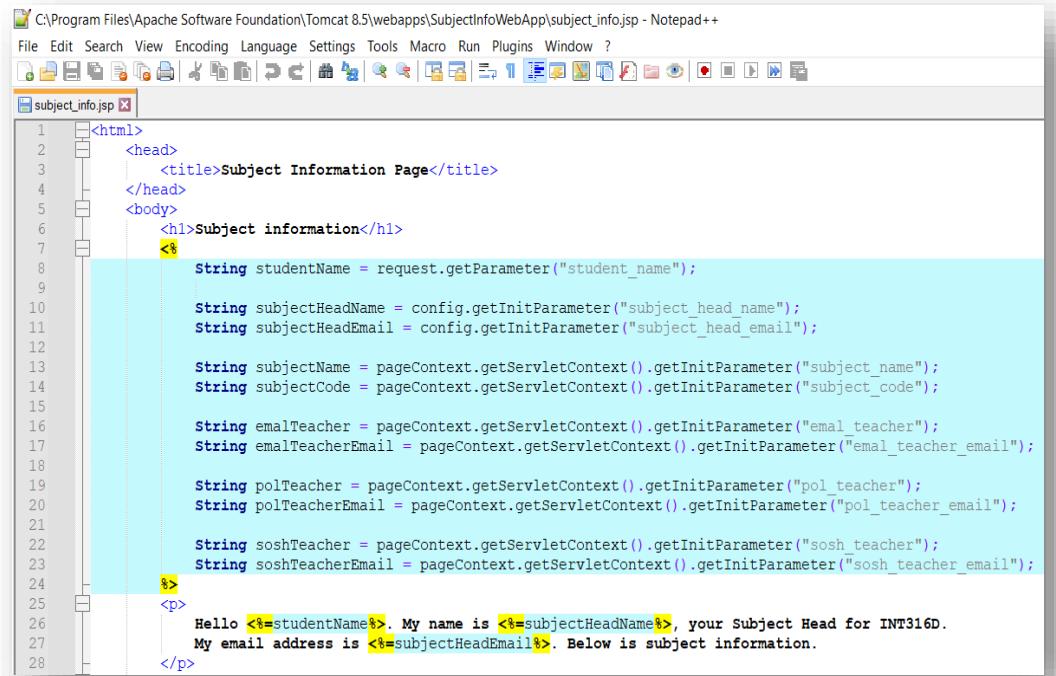
- **jsp**

- **student_entry.jsp**



```
C:\Users\memani\n\Documents\2023\s1\subjects\int316d\projects\project4_1\development\SubjectInfoWebApp\student_entry.jsp - Notepad++  
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?  
student_entry.jsp  
  
<html>  
  <head>  
    <title>Student Name Entry Page</title>  
  </head>  
  <body>  
    <h1>Student name</h1>  
    <p>  
      Please enter your name below:  
    </p>  
    <form action="SubjectInfoServlet.do" method="POST">  
      <table>  
        <tr>  
          <td>Student name:</td>  
          <td><input type="text" name="student_name" /></td>  
        </tr>  
        <tr>  
          <td></td>  
          <td><input type="submit" value="SUBMIT" /></td>  
        </tr>  
      </table>  
    </form>  
  </body>  
</html>
```

- **subject_info.jsp**

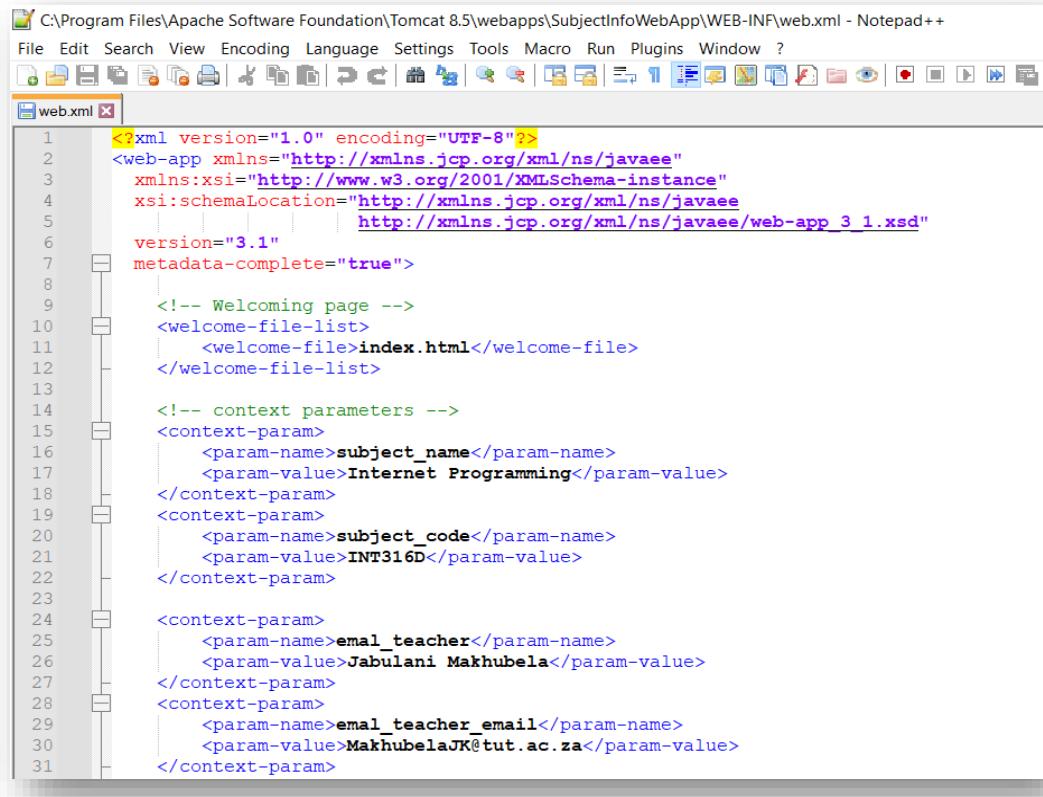


```
C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\SubjectInfoWebApp\subject_info.jsp - Notepad++  
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?  
subject_info.jsp  
  
<html>  
  <head>  
    <title>Subject Information Page</title>  
  </head>  
  <body>  
    <h1>Subject information</h1>  
    <%>  
    String studentName = request.getParameter("student_name");  
  
    String subjectHeadName = config.getInitParameter("subject_head_name");  
    String subjectHeadEmail = config.getInitParameter("subject_head_email");  
  
    String subjectName = pageContext.getServletContext().getInitParameter("subject_name");  
    String subjectCode = pageContext.getServletContext().getInitParameter("subject_code");  
  
    String emalTeacher = pageContext.getServletContext().getInitParameter("emal_teacher");  
    String emalTeacherEmail = pageContext.getServletContext().getInitParameter("emal_teacher_email");  
  
    String polTeacher = pageContext.getServletContext().getInitParameter("pol_teacher");  
    String polTeacherEmail = pageContext.getServletContext().getInitParameter("pol_teacher_email");  
  
    String soshTeacher = pageContext.getServletContext().getInitParameter("sosh_teacher");  
    String soshTeacherEmail = pageContext.getServletContext().getInitParameter("sosh_teacher_email");  
    <%>  
    <p>  
      Hello <%=studentName%>. My name is <%=subjectHeadName%>, your Subject Head for INT316D.  
      My email address is <%=subjectHeadEmail%>. Below is subject information.  
    </p>
```

```
29      <table>
30      |   <tr>
31      |       <td><b><em>Subject name:</em></b></td>
32      |       <td><%=subjectName%></td>
33      |   </tr>
34      |   <tr>
35      |       <td><b><em>Subject code:</em></b></td>
36      |       <td><%=subjectCode%></td>
37      |   </tr>
38      |   <tr>
39      |       <td><b><em>eMalahleni teacher:</em></b></td>
40      |       <td><%=emalTeacher%></td>
41      |   </tr>
42      |   <tr>
43      |       <td><b><em>Email address:</em></b></td>
44      |       <td><%=emalTeacherEmail%></td>
45      |   </tr>
46      |   <tr>
47      |       <td><b><em>Polokwane teacher:</em></b></td>
48      |       <td><%=polTeacher%></td>
49      |   </tr>
50      |   <tr>
51      |       <td><b><em>Email address:</em></b></td>
52      |       <td><%=polTeacherEmail%></td>
53      |   </tr>
54      |   <tr>
55      |       <td><b><em>Soshanguve teacher:</em></b></td>
56      |       <td><%=soshTeacher%></td>
57      |   </tr>
58      |   <tr>
59      |       <td><b><em>Email address:</em></b></td>
60      |       <td><%=soshTeacherEmail%></td>
61      |   </tr>
62      | </table>
63      | <p>
64      |     Please click <a href="index.html">here</a> to go back to the main page.
65      | </p>
66      | </body>
67  </html>
```

- **xml**

- **web.xml (DD)**



The screenshot shows the Notepad++ editor with the file C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\SubjectInfoWebApp\WEB-INF\web.xml open. The code is color-coded XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                               http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
          version="3.1"
          metadata-complete="true">

    <!-- Welcoming page -->
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
    </welcome-file-list>

    <!-- context parameters -->
    <context-param>
        <param-name>subject_name</param-name>
        <param-value>Internet Programming</param-value>
    </context-param>
    <context-param>
        <param-name>subject_code</param-name>
        <param-value>INT316D</param-value>
    </context-param>

    <context-param>
        <param-name>email_teacher</param-name>
        <param-value>Jabulani Makhubela</param-value>
    </context-param>
    <context-param>
        <param-name>email_teacher_email</param-name>
        <param-value>MakhubelaJK@tut.ac.za</param-value>
    </context-param>

    <context-param>
        <param-name>pol_teacher</param-name>
        <param-value>Michael Tshitake</param-value>
    </context-param>
    <context-param>
        <param-name>pol_teacher_email</param-name>
        <param-value>TshitakeFM@tut.ac.za</param-value>
    </context-param>

    <context-param>
        <param-name>sosh_teacher</param-name>
        <param-value>Vuyisile Memani</param-value>
    </context-param>
    <context-param>
        <param-name>sosh_teacher_email</param-name>
        <param-value>MemaniV@tut.ac.za</param-value>
    </context-param>

    <!-- servlet class -->
    <servlet>
        <servlet-name>SubjectInfoServlet</servlet-name>
        <servlet-class>za.ac.tut.controller.SubjectInfoServlet</servlet-class>
        <jsp-file>/subject_info.jsp</jsp-file>
        <init-param>
            <param-name>subject_head_name</param-name>
            <param-value>Vuyisile Memani</param-value>
        </init-param>
        <init-param>
            <param-name>subject_head_email</param-name>
            <param-value>MemaniV@tut.ac.za</param-value>
        </init-param>
    </servlet>

    <!-- servlet-mapping -->
    <servlet-mapping>
        <service-name>SubjectInfoServlet</service-name>
        <url-pattern>/SubjectInfoServlet.do</url-pattern>
    </servlet-mapping>
</web-app>
```

3. Prepare for deployment

Copy the **SubjectInfoWebApp** folder and paste on the **deployment** folder.

project4_1 > deployment		
Name	Date modified	Type
SubjectInfoWebApp	2023/01/23 20:58	File folder

Expand **SubjectInfoWebApp**

?????

Change **etc** to **WEB-INF** and ensure that the **classes** folder and the **web.xml** file are located inside this folder.

project4_1 > deployment > SubjectInfoWebApp			
Name	Date modified	Type	Size
WEB-INF	2023/01/23 21:01	File folder	
index	2023/01/23 17:40	Microsoft Edge HT...	1 KB
student_entry	2023/01/23 17:43	JSP File	1 KB
subject_info	2023/01/23 18:01	JSP File	3 KB

project4_1 > deployment > SubjectInfoWebApp > WEB-INF			
Name	Date modified	Type	Size
classes	2023/01/23 20:58	File folder	
web	2023/01/23 18:12	XML Document	2 KB

4. Deploy web application

Copy the **SubjectInfoWebApp** folder **deployment** and paste it under under the **webapps** folder of Tomcat.

Tomcat 8.5 > webapps		
Name	Date modified	Type
docs	2022/12/12 10:11	File folder
examples	2022/12/12 10:11	File folder
host-manager	2022/12/12 10:11	File folder
manager	2022/12/12 10:11	File folder
ROOT	2022/12/12 10:11	File folder
SubjectInfoWebApp	2023/01/23 21:09	File folder

Get to the **bin** directory of Tomcat.

Name	Date modified	Type	Size
bootstrap	2022/11/16 15:34	Executable Jar File	36 KB
catalina	2022/11/16 15:34	Windows Batch File	17 KB
ciphers	2022/11/16 15:34	Windows Batch File	3 KB
configtest	2022/11/16 15:34	Windows Batch File	2 KB
digest	2022/11/16 15:34	Windows Batch File	3 KB
service	2022/11/16 15:34	Windows Batch File	9 KB
setclasspath	2022/11/16 15:34	Windows Batch File	4 KB
shutdown	2022/11/16 15:34	Windows Batch File	2 KB
startup	2022/11/16 15:34	Windows Batch File	2 KB
Tomcat8	2022/11/16 15:34	Application	142 KB
Tomcat8w	2022/11/16 15:34	Application	126 KB
tomcat-juli	2022/11/16 15:34	Executable Jar File	52 KB
tool-wrapper	2022/11/16 15:34	Windows Batch File	5 KB
version	2022/11/16 15:34	Windows Batch File	2 KB

Copy the address of the bin directory

C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin

Launch the command line.

```
Command Prompt
Microsoft Windows [Version 10.0.19041.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>
```

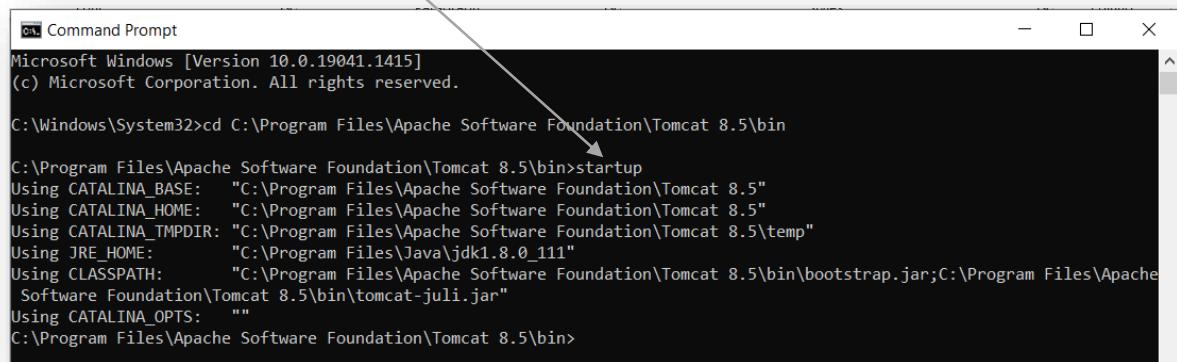
Change drive to the **bin** directory of Tomcat.

```
Command Prompt
Microsoft Windows [Version 10.0.19041.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin

C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>
```

Start Tomcat by typing "**startup**" on the command line and then click the **Enter** key.



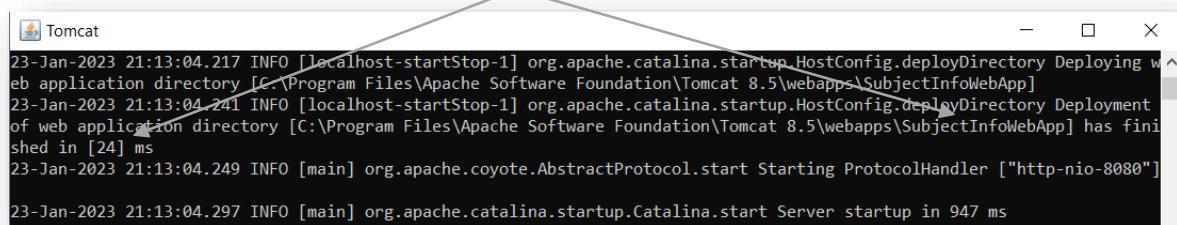
```
Command Prompt
Microsoft Windows [Version 10.0.19041.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin
C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>startup
Using CATALINA_BASE: "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_HOME: "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_TMPDIR: "C:\Program Files\Apache Software Foundation\Tomcat 8.5\temp"
Using JRE_HOME: "C:\Program Files\Java\jdk1.8.0_111"
Using CLASSPATH: "C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\bootstrap.jar;C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\tomcat-juli.jar"
Using CATALINA_OPTS: ""

C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>
```

Output:

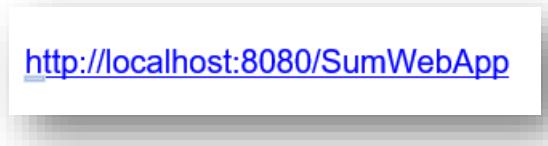
The **SubjectInfoWebApp** application is deployed in **24 ms**.



```
Tomcat
23-Jan-2023 21:13:04.217 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory [C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\SubjectInfoWebApp]
23-Jan-2023 21:13:04.241 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application directory [C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\SubjectInfoWebApp] has finished in [24] ms
23-Jan-2023 21:13:04.249 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["http-nio-8080"]
23-Jan-2023 21:13:04.297 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in 947 ms
```

5. Run the web application

Open the browser and type: (**Correct → SubjectInfoWebApp**)

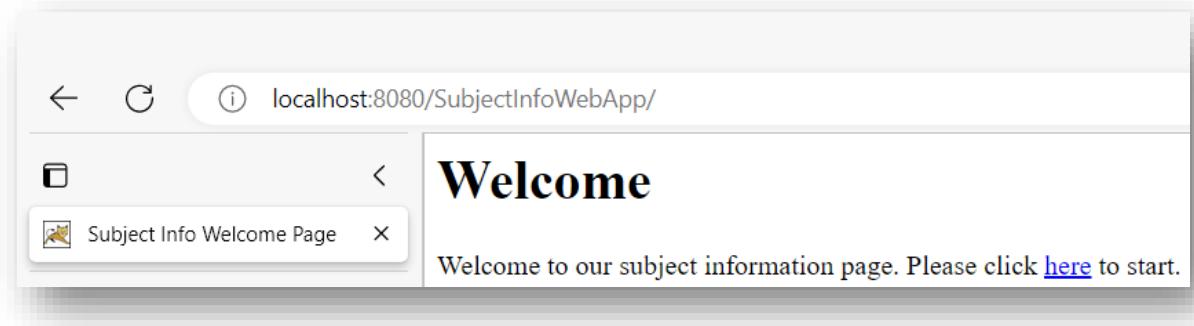


<http://localhost:8080/SumWebApp>

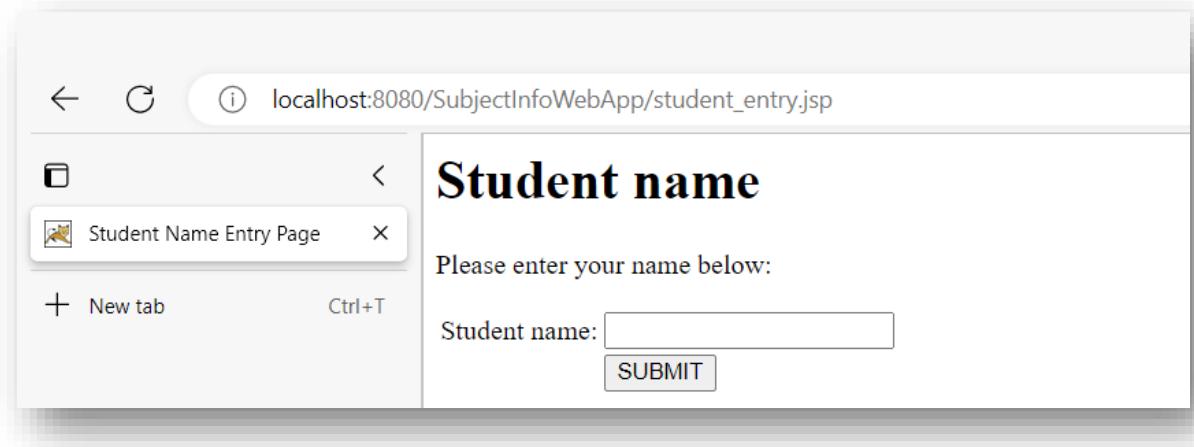
Note:

- **http**: is the protocol used to communicate between the browser and the Web Server, Tomcat.
- **localhost**: is the address of the machine that hosts the Web Server. In this instance the Web Server is running on the local machine.
- **8080**: is the port where the Web Server is running.
- **SubjectInfoWebApp**: is the name of the web application deployed on the server that we want to talk to.

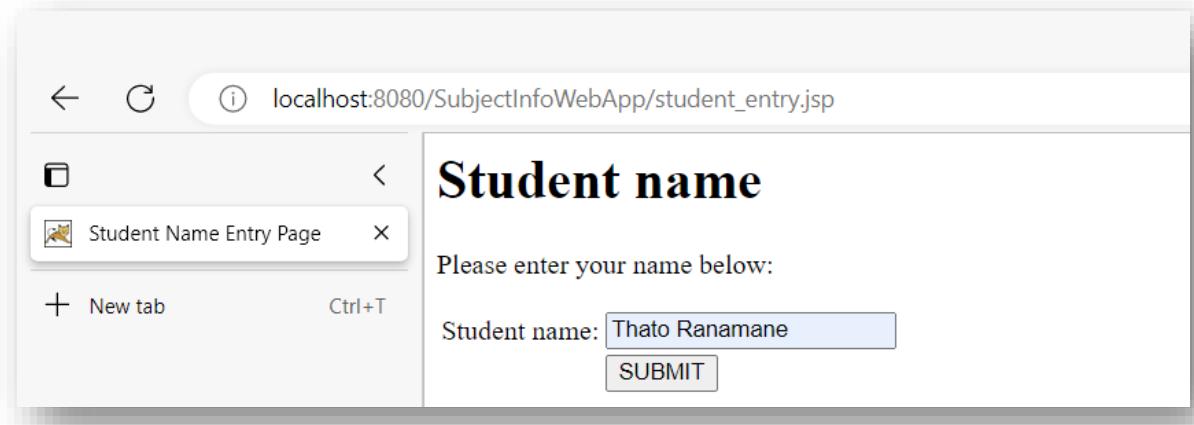
Output:



Click on the link.



Enter name.



Click on the **SUBMIT** button.

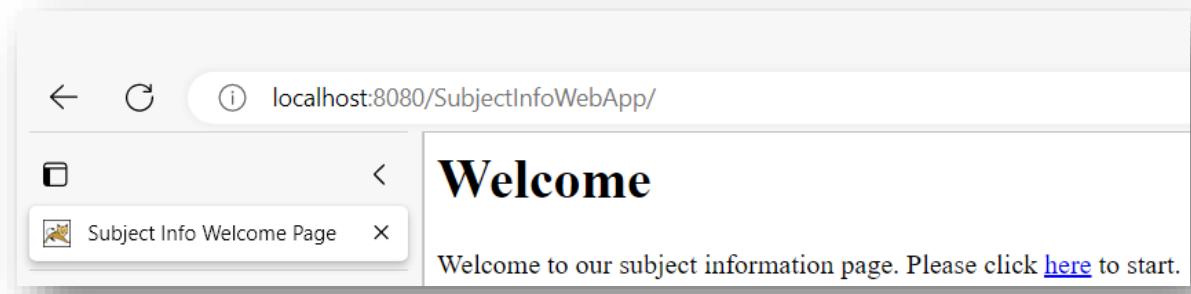
Subject information

Hello Thato Ramamane. My name is Vuyisile Memani, your Subject Head for INT316D. My email address is MemaniV@tut.ac.za. Below is subject information.

Subject name: Internet Programming
Subject code: INT316D
eMalahleni teacher: Jabulani Makhubela
Email address: MakhubelaJK@tut.ac.za
Polokwane teacher: Michael Tshitake
Email address: TshitakeFM@tut.ac.za
Soshangwe teacher: Vuyisile Memani
Email address: MemaniV@tut.ac.za

Please click [here](#) to go back to the main page.

Click on the link to go back to the main page.



6. Stop Tomcat

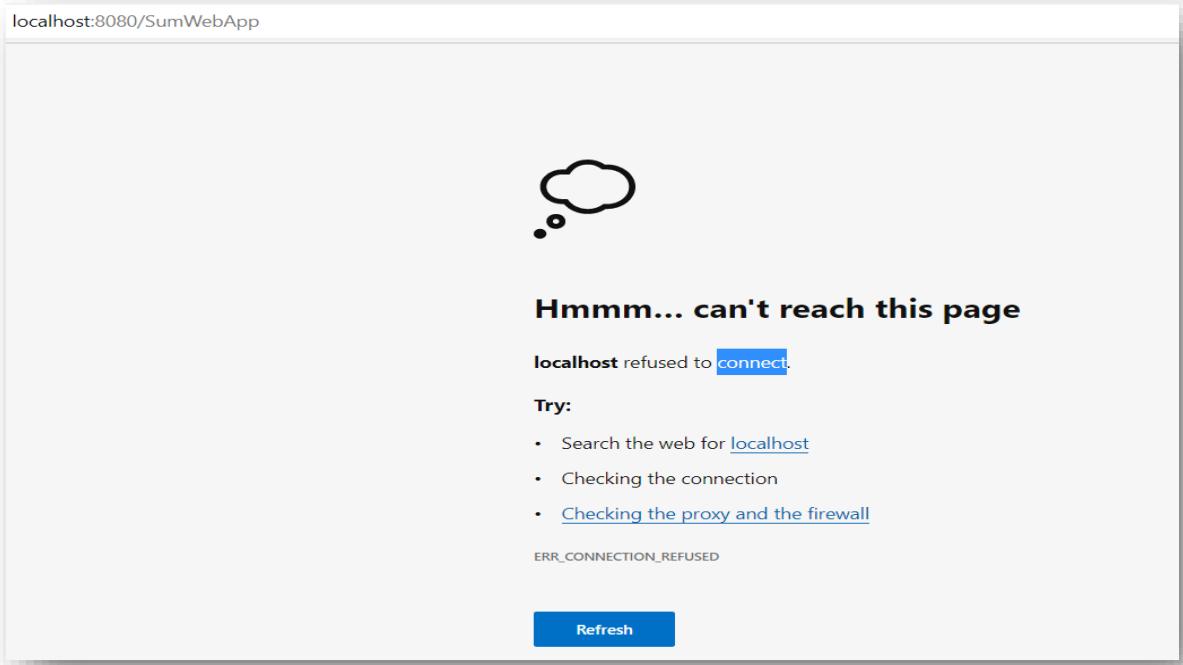
Stop Tomcat by typing "**shutdown**" on the command line and then click the **Enter** key.

A screenshot of a Windows Command Prompt window titled 'Command Prompt'. The window shows the following text:

```
C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>startup
Using CATALINA_BASE: "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_HOME: "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_TMPDIR: "C:\Program Files\Apache Software Foundation\Tomcat 8.5\temp"
Using JRE_HOME: "C:\Program Files\Java\jdk1.8.0_111"
Using CLASSPATH: "C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\bootstrap.jar;C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\tomcat-juli.jar"
Using CATALINA_OPTS: ""
C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>shutdown
Using CATALINA_BASE: "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_HOME: "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_TMPDIR: "C:\Program Files\Apache Software Foundation\Tomcat 8.5\temp"
Using JRE_HOME: "C:\Program Files\Java\jdk1.8.0_111"
Using CLASSPATH: "C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\bootstrap.jar;C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\tomcat-juli.jar"
Using CATALINA_OPTS: ""
C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>
```

A red arrow points from the text 'Please click [here](#) to start.' in the previous screenshot to the word 'shutdown' in this command prompt window.

Try to run the web application whilst the server is down. An error takes place.
: (Correct → SubjectInfoWebApp)



4.6 Example 2

Problem statement

Create a web application that will determine and display the sum of two numbers. The application must allow the user to enter the two numbers. Upon receiving the two numbers, the application must determine their sum and display.

To do

Create such an application. First discuss program flow, then identify MVC components which should emanate from the discussion, depict the project's structure of the solution which should show the identified components, and lastly do the actual coding.

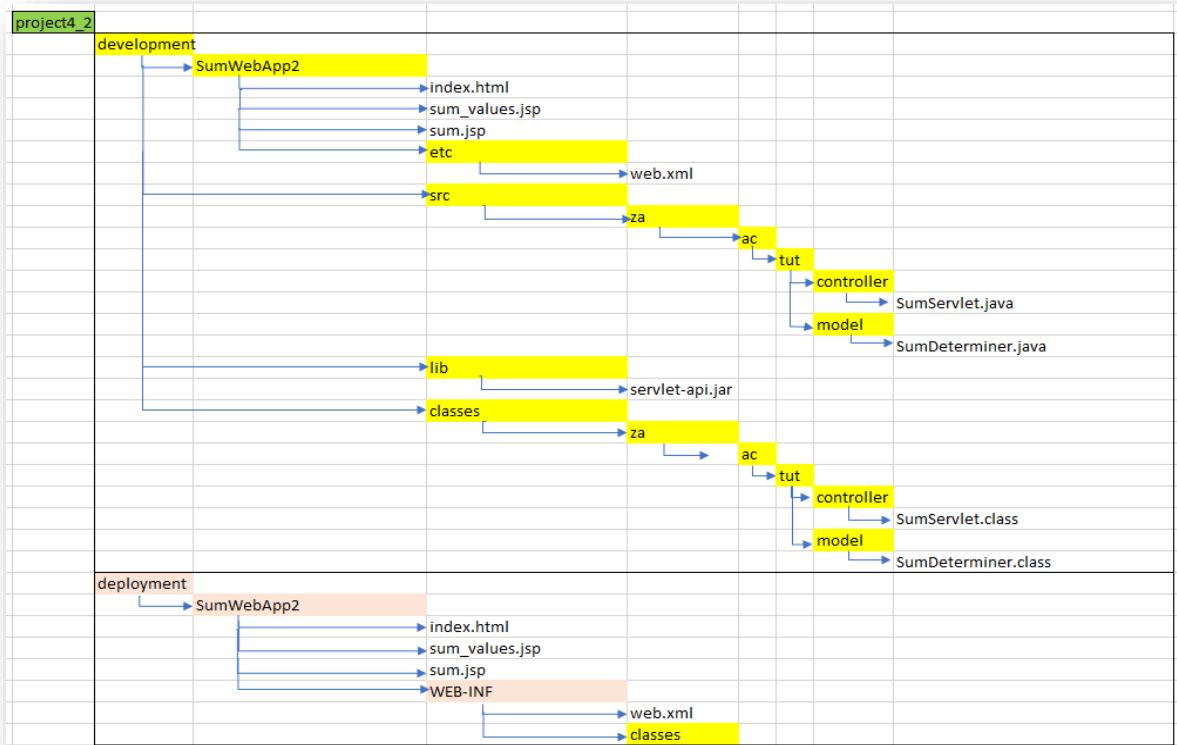
Program flow

The program flow is as explained in example 1 of chapter 2. In this solution we will also include Servlets and models.

MVC components

The MVC components are as determined in example 1 of chapter 2.

Project structure



Implementation

1. Create the files.

Create the files as follows:

- **html**
 - **index.html**

```

C:\Users\memani\Documents\2023\s1\subjects\int316d\projects\project4_2\development\SumWebApp2\index.html - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
index.html x
1 <html>
2   <head>
3     <title>Sum Web App</title>
4   </head>
5   <body>
6     <h1>Welcome</h1>
7     <p>
8       Welcome to our sum web app.
9       Please click <a href="sum_values.jsp">here</a> to enter the sum values.
10    </p>
11  </body>
12 </html>

```

- **jsp**
 - **sum_values.jsp**

```
<html>
    <head>
        <title>Sum Values Page</title>
    </head>
    <body>
        <h1>Sum values</h1>
        <p>
            Please enter two numbers below:
        </p>
        <form action="SumServlet.do" method="GET">
            <table>
                <tr>
                    <td>First number:</td>
                    <td><input type="text" name="num1" /></td>
                </tr>
                <tr>
                    <td>Second number:</td>
                    <td><input type="text" name="num2" /></td>
                </tr>
                <tr>
                    <td></td>
                    <td><input type="submit" value="ADD" /></td>
                </tr>
            </table>
        </form>
    </body>
</html>
```

▪ sum.jsp

```
<html>
    <head>
        <title>Sum Page</title>
    </head>
    <body>
        <h1>Sum</h1>
        <%>
        Integer num1 = (Integer)request.getParameter("num1");
        Integer num2 = (Integer)request.getParameter("num2");
        Integer sum = (Integer)request.getParameter("sum");
        <%>
        <p>
            The sum of <%=num1%> and <%=num2%> is <%=sum%>.
        </p>
        <p>
            Please click <a href="index.html">here</a> to go back to the main page.
        </p>
    </body>
</html>
```

○ servlet

▪ SumServlet.java

```

C:\Users\memaniv\Documents\2023\sem1\subjects\int316d\projects\project4_2\development\SumWebApp2\src\za\ac\tut\controller\SumServlet.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
SumServlet.java x
1 package za.ac.tut.controller;
2
3 import java.io.IOException;
4 import za.ac.tut.model.SumDeterminer;
5 import javax.servlet.ServletException;
6 import javax.servlet.http.HttpServlet;
7 import javax.servlet.RequestDispatcher;
8 import javax.servlet.http.HttpServletRequest;
9 import javax.servlet.http.HttpServletResponse;
10
11 public class SumServlet extends HttpServlet
12 {
13     public void doPost(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException
14     {
15         Integer num1 = Integer.parseInt(request.getParameter("num1"));
16         Integer num2 = Integer.parseInt(request.getParameter("num2"));
17
18         SumDeterminer sd = new SumDeterminer();
19         Integer sum = sd.determineSum(num1, num2);
20
21         request.setAttribute("num1", num1);
22         request.setAttribute("num2", num2);
23         request.setAttribute("sum", sum);
24
25         RequestDispatcher disp = request.getRequestDispatcher("sum.jsp");
26         disp.forward(request, response);
27     }
28 }

```

- **model**

- **SumDeterminer.java**

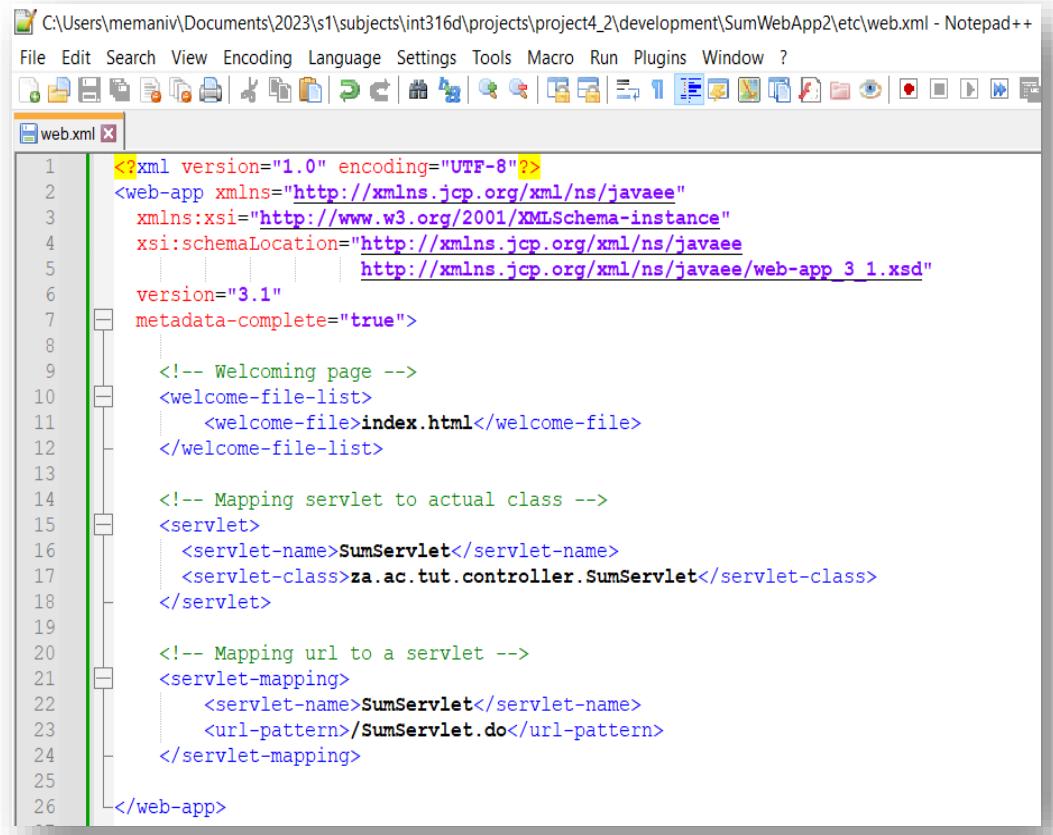
```

C:\Users\memaniv\Documents\2023\sem1\subjects\int316d\projects\project4_2\development\SumWebApp2\src\za\ac\tut\model\SumDeterminer.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
SumDeterminer.java x
1 package za.ac.tut.model;
2
3 public class SumDeterminer {
4
5     public SumDeterminer() {
6     }
7
8     public Integer determineSum(Integer num1, Integer num2){
9         Integer sum;
10
11         sum = num1 + num2;
12
13         return sum;
14     }
15 }

```

- **xml**

- **web.xml**



The screenshot shows the Notepad++ application window with the file 'web.xml' open. The code in the editor is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
        http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
    version="3.1"
    metadata-complete="true">

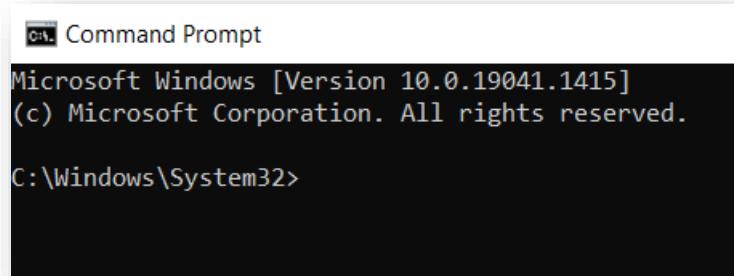
    <!-- Welcoming page -->
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
    </welcome-file-list>

    <!-- Mapping servlet to actual class -->
    <servlet>
        <servlet-name>SumServlet</servlet-name>
        <servlet-class>za.ac.tut.controller.SumServlet</servlet-class>
    </servlet>

    <!-- Mapping url to a servlet -->
    <servlet-mapping>
        <servlet-name>SumServlet</servlet-name>
        <url-pattern>/SumServlet.do</url-pattern>
    </servlet-mapping>
</web-app>
```

2. Compile source code

Open the command line.



The screenshot shows a Windows Command Prompt window. The text displayed is:

```
Command Prompt
Microsoft Windows [Version 10.0.19041.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>
```

Change directory to the **src** directory of **SumWebApp2**.

```
Command Prompt
Microsoft Windows [Version 10.0.19041.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd C:\Users\memaniv\Documents\2023\s1\subjects\int316d\projects\project4_2\development\SumWebApp2\src>c

C:\Users\memaniv\Documents\2023\s1\subjects\int316d\projects\project4_2\development\SumWebApp2\src>
```

Compile SumDeterminer.java

```
Command Prompt
C:\Users\memaniv\Documents\2023\s1\subjects\int316d\projects\project4_2\development\SumWebApp2\src>javac -d ..\classes za\ac\tut\model\SumDeterminer.java

C:\Users\memaniv\Documents\2023\s1\subjects\int316d\projects\project4_2\development\SumWebApp2\src>
```

Compile SumServlet.java

```
Command Prompt
C:\Users\memaniv\Documents\2023\s1\subjects\int316d\projects\project4_2\development\SumWebApp2\src>javac -d ..\classes za\ac\tut\controller\SumServlet.java
za\ac\tut\controller\SumServlet.java:5: error: package javax.servlet does not exist
import javax.servlet.ServletException;
 ^
za\ac\tut\controller\SumServlet.java:6: error: package javax.servlet.http does not exist
import javax.servlet.http.HttpServlet;
 ^
za\ac\tut\controller\SumServlet.java:7: error: package javax.servlet does not exist
import javax.servlet.RequestDispatcher;
 ^
```

An error takes place because the Servlet API is not accessible to the application.

Resolve this problem by setting the classpath to the servlet-api.jar file.

Set the classpath.

```
Command Prompt
C:\Users\memaniv\Documents\2023\s1\subjects\int316d\projects\project4_2\development\SumWebApp2\src>set classpath=.;C:\Users\memaniv\Documents\2023\s1\subjects\int316d\projects\project4_2\development\SumWebApp2\lib\servlet-api.jar

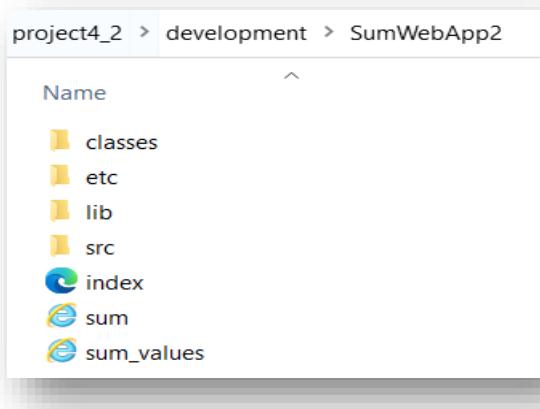
C:\Users\memaniv\Documents\2023\s1\subjects\int316d\projects\project4_2\development\SumWebApp2\src>
```

Compile again. It works now.

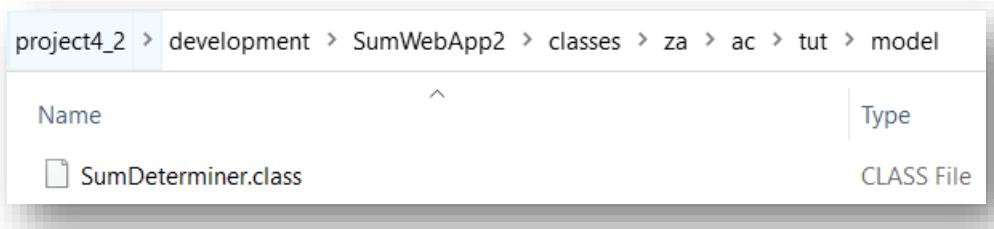
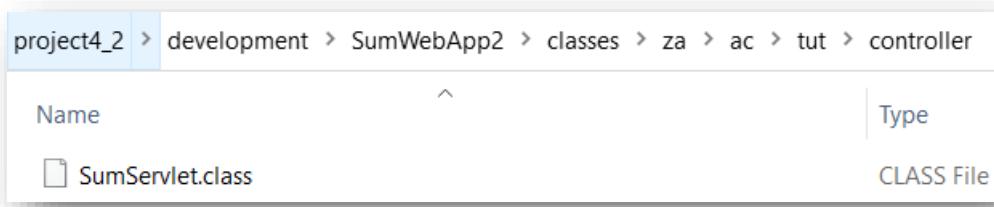
```
Command Prompt  
C:\Users\memaniv\Documents\2023\s1\subjects\int316d\projects\project4_2\development\SumWebApp2\src>javac -d ..\classes za/ac/tut/controller/SumServlet.java  
C:\Users\memaniv\Documents\2023\s1\subjects\int316d\projects\project4_2\development\SumWebApp2\src>
```

3. View development files

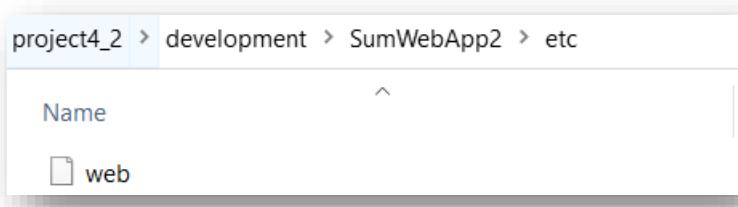
View all the files.



Check under the classes folder.



View inside etc folder.



View inside the lib folder. You can get the servlet-api.jar file from the lib directory of Tomcat.

The screenshot shows a file explorer interface. At the top, there's a breadcrumb navigation: project4_2 > development > SumWebApp2 > lib. Below this is a table with columns: Name, Date modified, Type, and Size. The table lists various Tomcat-related jars. One row, 'servlet-api', is selected and highlighted with a blue border. The 'Name' column shows icons representing the jar types.

Name	Date modified	Type	Size
annotations-api	2022/11/16 15:34	Executable Jar File	13 KB
catalina	2022/11/16 15:34	Executable Jar File	1 686 KB
catalina-ant	2022/11/16 15:34	Executable Jar File	53 KB
catalina-ha	2022/11/16 15:34	Executable Jar File	119 KB
catalina-storeconfig	2022/11/16 15:34	Executable Jar File	76 KB
catalina-tribes	2022/11/16 15:34	Executable Jar File	288 KB
ecj-4.6.3	2022/11/16 15:34	Executable Jar File	2 393 KB
el-api	2022/11/16 15:34	Executable Jar File	87 KB
jasper	2022/11/16 15:34	Executable Jar File	589 KB
jasper-el	2022/11/16 15:34	Executable Jar File	167 KB
jaspic-api	2022/11/16 15:34	Executable Jar File	27 KB
jsp-api	2022/11/16 15:34	Executable Jar File	61 KB
servlet-api	2022/11/16 15:34	Executable Jar File	244 KB

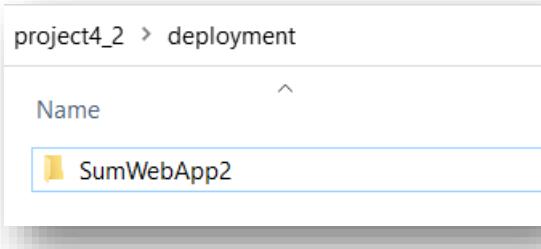
View inside the src folder.

The screenshot shows a file explorer interface. At the top, there's a breadcrumb navigation: project4_2 > development > SumWebApp2 > src > za > ac > tut > controller. Below this is a table with columns: Name and Type. The table lists a single file, 'SumServlet', which is identified as a JAVA File. Another screenshot below shows a similar structure for the 'model' folder, also containing a 'SumDeterminer' file.

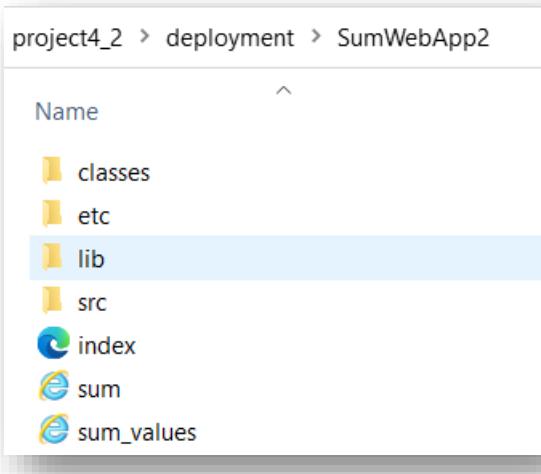
Name	Type
SumServlet	JAVA File

4. Prepare for deployment

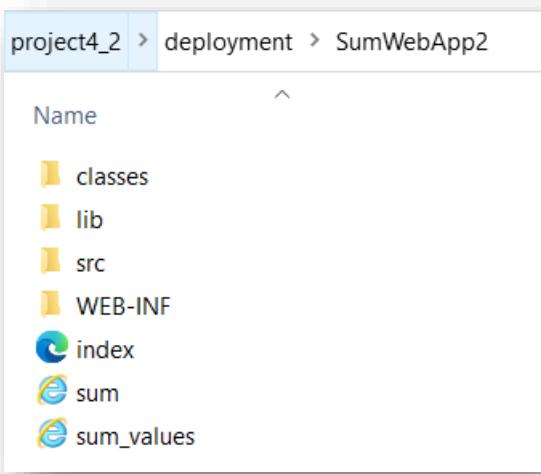
Copy the entire **SumWebApp2** folder and paste it under the **deployment** folder.



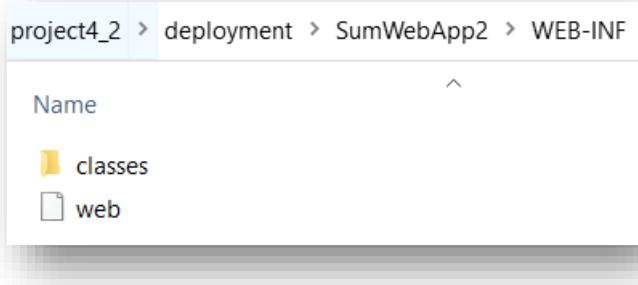
Expand **SumWebApp2**.



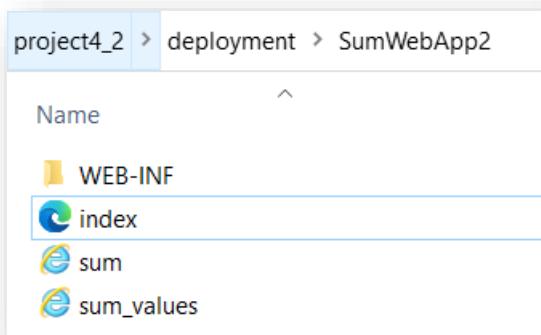
Change **etc** to **WEB-INF**.



Move the **classes** folder to **WEB-INF**.

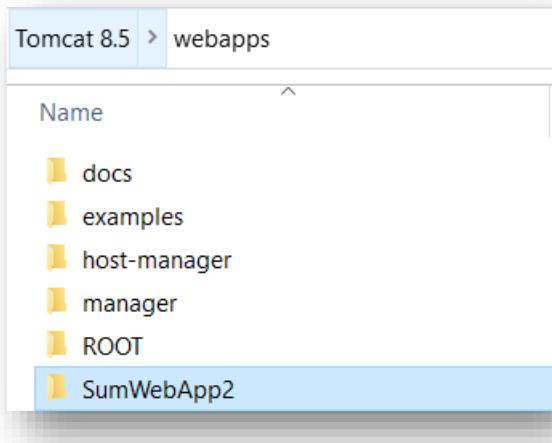


Delete the **src** and **lib** folders.

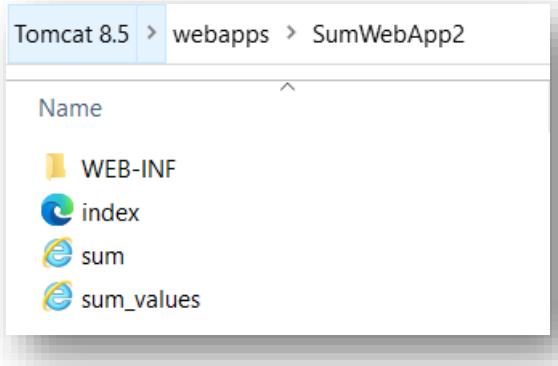


5. Deploy the web application

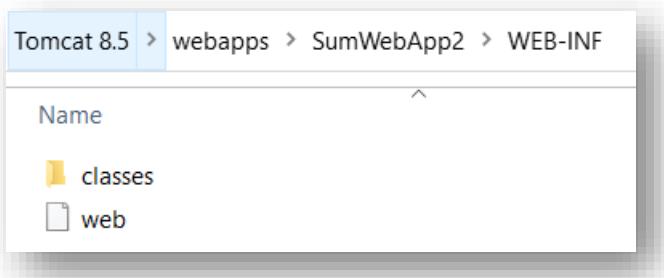
Copy the **SumWebApp2** folder from the **deployment** and paste it on the **webapps** folder of **Tomcat**.



Expand **SumWebApp2**.



Expand **WEB-INF**.



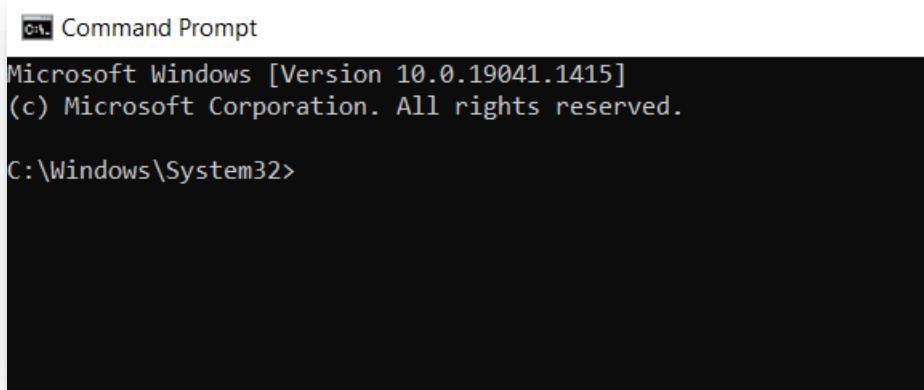
Go to the **bin** directory of **Tomcat**.

Name	Date modified	Type	Size
bootstrap	2022/11/16 15:34	Executable Jar File	36 KB
catalina	2022/11/16 15:34	Windows Batch File	17 KB
ciphers	2022/11/16 15:34	Windows Batch File	3 KB
configtest	2022/11/16 15:34	Windows Batch File	2 KB
digest	2022/11/16 15:34	Windows Batch File	3 KB
service	2022/11/16 15:34	Windows Batch File	9 KB
setclasspath	2022/11/16 15:34	Windows Batch File	4 KB
shutdown	2022/11/16 15:34	Windows Batch File	2 KB
startup	2022/11/16 15:34	Windows Batch File	2 KB
Tomcat8	2022/11/16 15:34	Application	142 KB
Tomcat8w	2022/11/16 15:34	Application	126 KB
tomcat-juli	2022/11/16 15:34	Executable Jar File	52 KB
tool-wrapper	2022/11/16 15:34	Windows Batch File	5 KB
version	2022/11/16 15:34	Windows Batch File	2 KB

Copy the address of the **bin** directory.

C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin

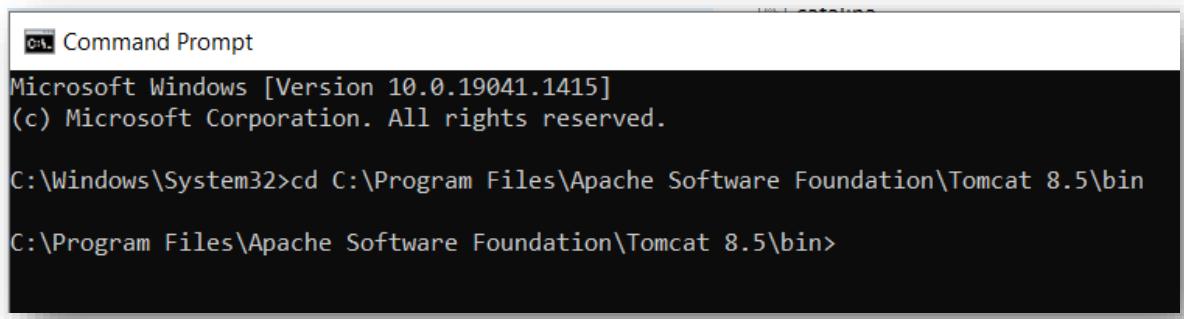
Launch the command line.



```
Windows Command Prompt
Microsoft Windows [Version 10.0.19041.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>
```

Change location to the **bin** directory of **Tomcat**.

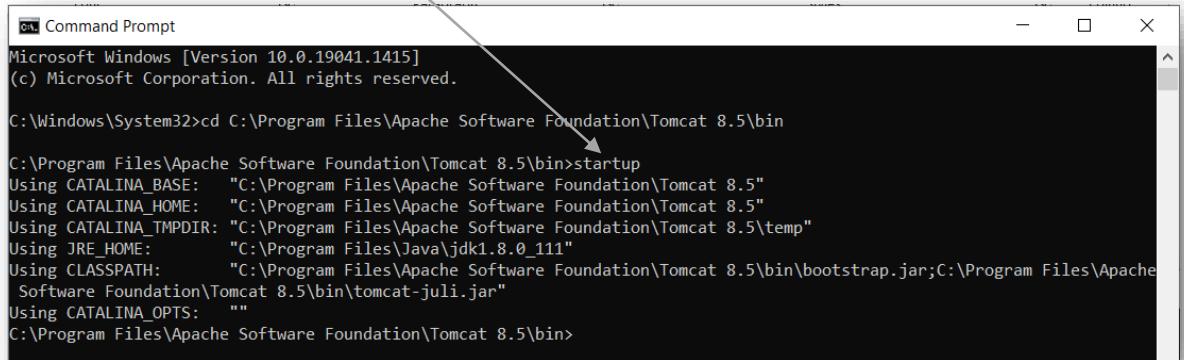


```
Windows Command Prompt
Microsoft Windows [Version 10.0.19041.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin

C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>
```

Start Tomcat by typing "**startup**" on the command line and then click the **Enter** key.

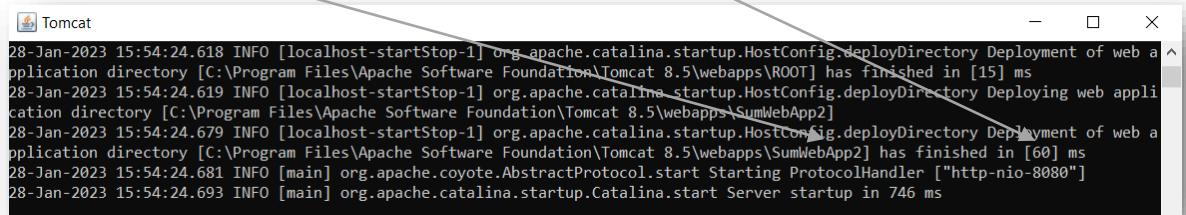


```
Windows Command Prompt
Microsoft Windows [Version 10.0.19041.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin
C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>startup
Using CATALINA_BASE: "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_HOME: "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_TMPDIR: "C:\Program Files\Apache Software Foundation\Tomcat 8.5\temp"
Using JRE_HOME: "C:\Program Files\Java\jdk1.8.0_111"
Using CLASSPATH: "C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\bootstrap.jar;C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\tomcat-juli.jar"
Using CATALINA_OPTS: ""
C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>
```

Output:

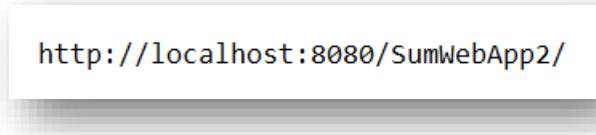
The **SumWebApp2** application is deployed in **60 ms**.



```
Tomcat
28-Jan-2023 15:54:24.618 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application directory [C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\ROOT] has finished in [15] ms
28-Jan-2023 15:54:24.619 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory [C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\SumWebApp2]
28-Jan-2023 15:54:24.679 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application directory [C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\SumWebApp2] has finished in [60] ms
28-Jan-2023 15:54:24.681 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["http-nio-8080"]
28-Jan-2023 15:54:24.693 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in 746 ms
```

6. Run the web application

Open the browser and type:

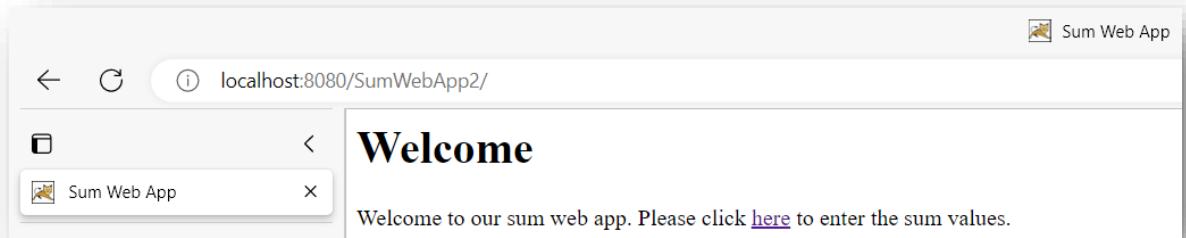


`http://localhost:8080/SumWebApp2/`

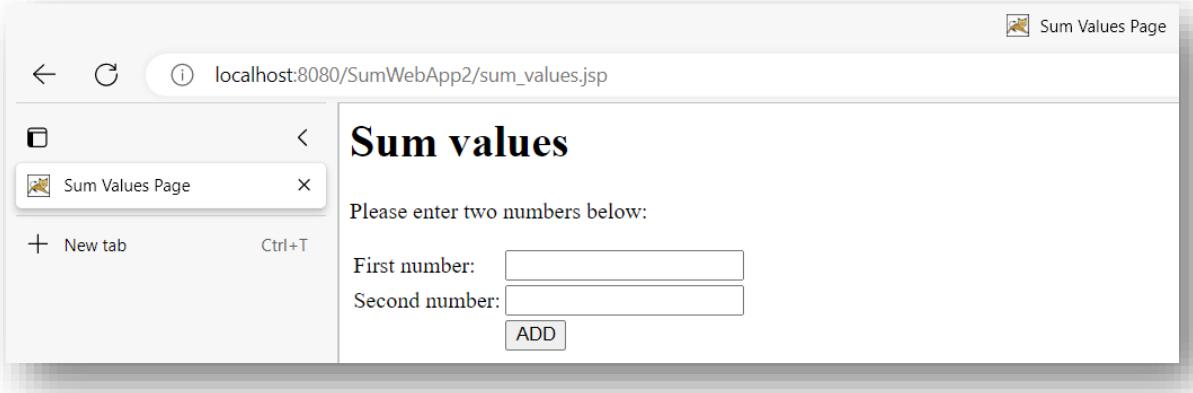
Note:

- **http**: is the protocol used to communicate between the browser and the Web Server, Tomcat.
- **localhost**: is the address of the machine that hosts the Web Server. In this instance the Web Server is running on the local machine.
- **8080**: is the port where the Web Server is running.
- **SumWebApp2**: is the name of the web application deployed on the server that we want to talk to.

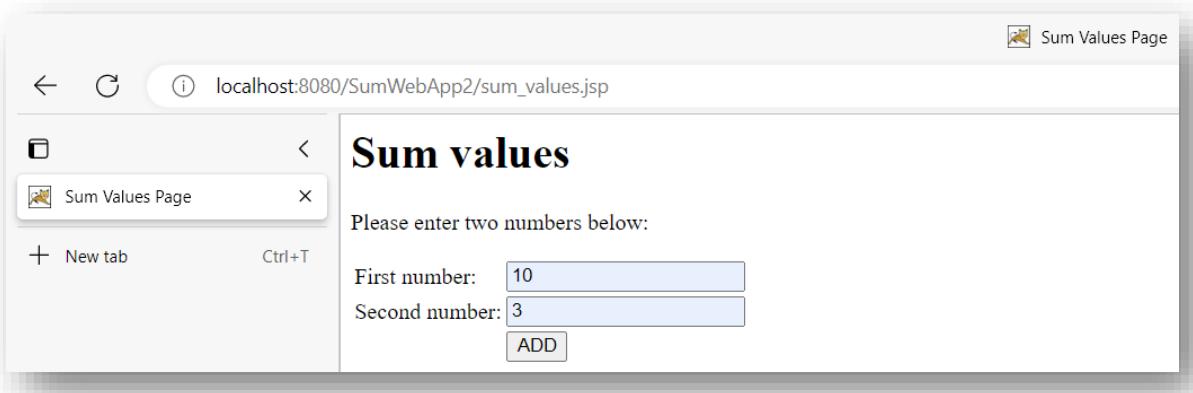
Output:



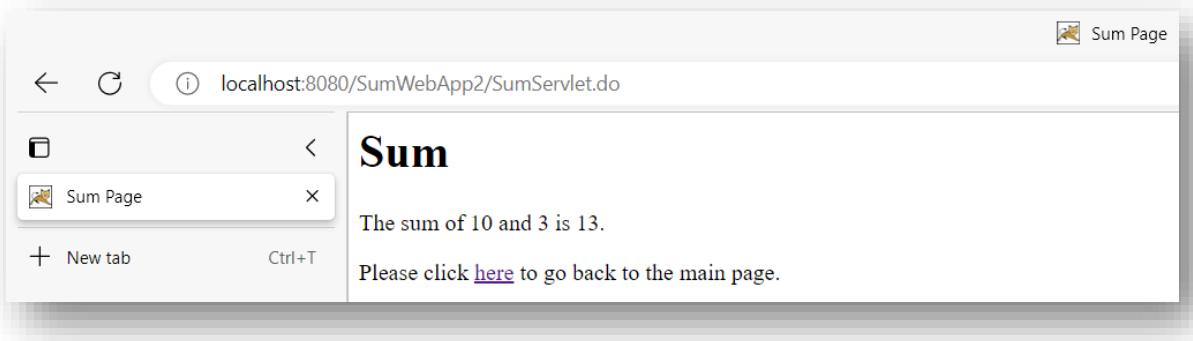
Click on the link.



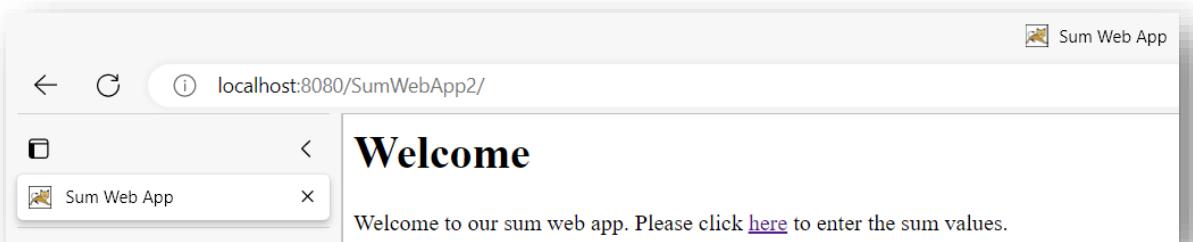
Enter the two numbers.



Click on the **ADD** button.

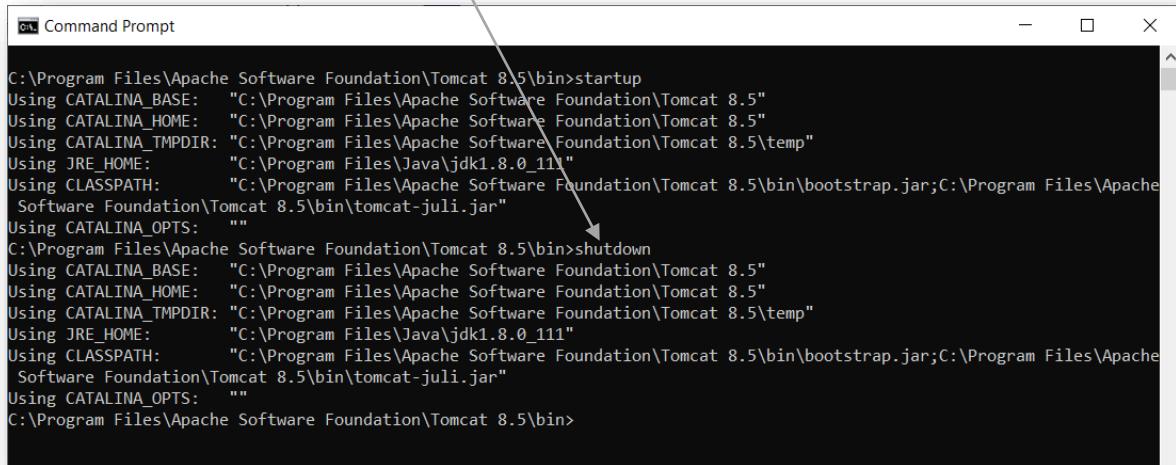


Click on the link to go back to the main page.



7. Stop Tomcat

Stop Tomcat by typing “**shutdown**” on the command line and then click the **Enter** key.

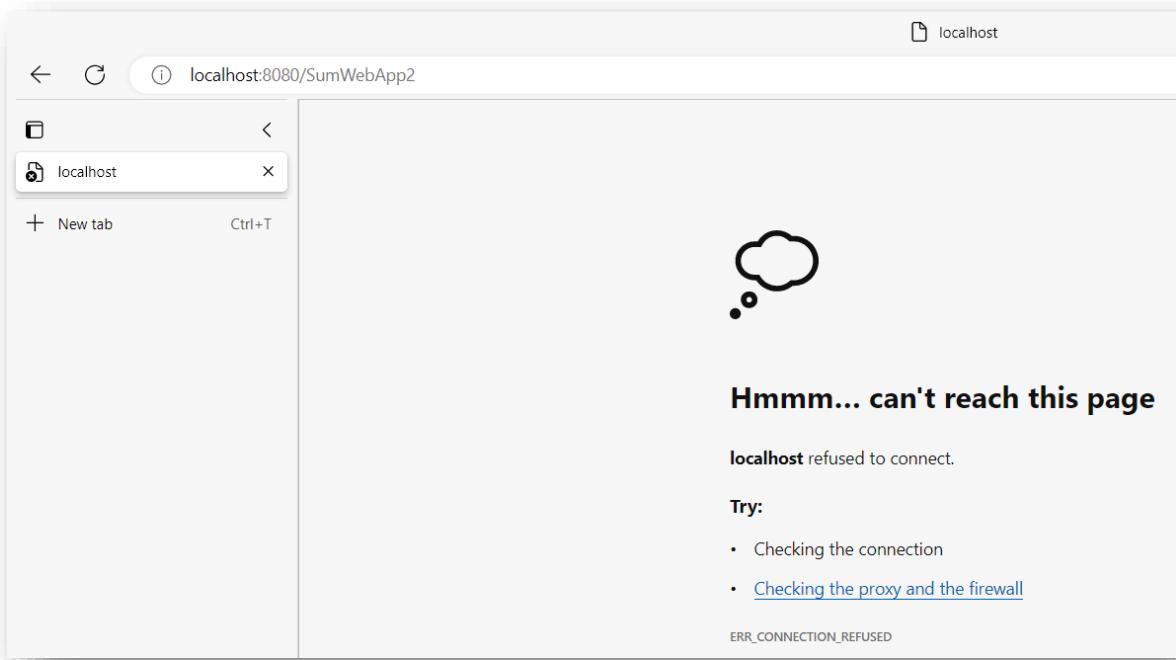


```
C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>startup
Using CATALINA_BASE:  "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_HOME:  "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_TMPDIR: "C:\Program Files\Apache Software Foundation\Tomcat 8.5\temp"
Using JRE_HOME:        "C:\Program Files\Java\jdk1.8.0_111"
Using CLASSPATH:       "C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\bootstrap.jar;C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\tomcat-juli.jar"
Using CATALINA_OPTS:   ""

C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>shutdown
Using CATALINA_BASE:  "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_HOME:  "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_TMPDIR: "C:\Program Files\Apache Software Foundation\Tomcat 8.5\temp"
Using JRE_HOME:        "C:\Program Files\Java\jdk1.8.0_111"
Using CLASSPATH:       "C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\bootstrap.jar;C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\tomcat-juli.jar"
Using CATALINA_OPTS:   ""

C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>
```

Try to run the web application whilst the server is down. An error takes place.



4.7 DIY (Do It Yourself)

In this chapter we introduced you to mathematical operators. We showed you how each work. In this DIY, we want you to undertake three tasks in line with what you have learnt.

Task #1

Create a web application that will perform a rand to dollar conversion, and display the results. Assume that \$1 is equal to R10.

Task #2

Mujinga is an intern at a text analysis company called **TextSpectacles**. The company specializes in analysing text messages for customers. As an intern, Mujinga is given the responsibility of creating a web application that will help in the analysis of short text messages. Given a sentence, she is expected to determine, through the web application, the number of times a word is occurring in the sentence.

To make the process easier, it is assumed that the sentence under analysis will always contain letters and special characters, that is the comma (,) and period (.). So upon receiving a sentence, Mujinga is supposed to remove the special characters from the text, get the individual words making up the sentence, change all the words to lowercases, and store them in a list.

The web application must then determine how many times each word in the list is occurring. The outcome of this determination must be displayed on the screen. As an established programmer at TextSpectacles, Mujinga approaches you with the request to help her accomplish the required task (unfortunately you can't say no).

Task #3

Thato is an entrepreneur involved in the property business. She builds houses and thereafter rent them out. She has a number of properties around Gauteng that she rents out to the general public in need of accommodation. Most of her houses are in the townships where the property market is booming.

Though lucrative, the monthly income is unpredictable because of the changing financial circumstances of people. In some months the demand is high, and in others

it is low. The table below shows the monthly income she has made over the last three months in the respective areas.

Area	Code	Monthly income		
		Jan	Feb	Mar
Garankuwa	GA	R120-000	R100-000	R110-000
Soshanguve	SO	R200-000	R170-000	R190-000
Mabopane	MA	R150-000	R180-000	R100-000

Thato needs a web application that will help her monitor the monthly income generated by her properties. The envisaged application should be able to help her perform the following tasks:

- Store the monthly income for the various properties in a database.
- Retrieve from the database the property information of a specific area (using area code) for a particular month, and thereafter display the results.

Thato approaches you, as an esteemed programmer, to create the web application for her.

Task #4

Company XYZ is putting into place measures to mitigate against the spread of COVID in its premises. When reporting for duty, employees are subjected to temperature scanning and hands sanitization. The **temperature readings**, together with the **names** and **staff numbers** of employees are recorded manually in a notebook. The staff number is used to uniquely identify each employee.

After all the employees have clocked-in for work, the notebook is handed-over to management for processing. Management wants to digitize this manual handling of data. Consequently management approaches you as a programmer to develop for them a web application that will be capable of doing the following:

- Record the temperature information of employees.
- Retrieve the data of all employees from the whose temperature is above a given temperature value.
- Search for temperature information of a specific employee.
- Update the temperature reading of a specific employee.

Create such a web application for management.

Task #5

Busisiwe is a teacher at ABC. She needs a web application that will help her manage assessment queries from students. Normally, after marking an assessment, she gives her students an opportunity to raise complaints regarding the marking of their scripts.

When lodging a complaint, students are required to provide their personal details (name, surname and student number) and the details of the question where they are not satisfied. The question details entail the assessment name, question number, and the description of the specific area of discontent. Below is an example of student queries.

No.	Name	Surname	Student no.	Test code	Question	Complaint
1	Vuyi	Memza	111	CT1	1	My definition has not been marked.
2	Khomo	Mnisi	222	CT1	2	My code has not been marked.
3	Tata	Tau	333	CT1	3	My design is correct.
4	Rara	Ndah	444	CT1	4	Please mark my code.
5	Lira	Nkosi	555	CT1	5	Please mark my database.

The students mainly send their queries through emails. Consequently, Busisiwe needs a web application that will allow her students to raise their queries through it. After her students have entered their test issues, Busisiwe wants to be able to do the following:

- View all the student queries.
- For each query be able to make a comment which will serve as a response.
- Be able to view a specific student query.

Busisiwe approaches your company, **WeCanDoIt**, to develop the web application for her. Ms Livhuwani, a senior developer at the company, task you with the responsibility of creating the web application.

4.8 Conclusion

In this chapter we managed to introduce the student to Servlets. In the next chapter we will discuss conversational web applications. Thank you very much for having taken time to go through this chapter. Enjoy the rest of the day and God bless you.

5 Conversational web apps

In this chapter we discuss the concept of conversational web apps.

5.1 What is a conversational web app?

Web applications are by default non-conversational. This means they are stateless, they are unable to keep track of a conversation with a client. This is due to the usage of HTTP which is a stateless communication protocol. To overcome this challenge, we use sessions.

Sessions are objects that keep track of the conversation between a client and a web application. The **HttpSession** class is used to create sessions. After a session is created, communication between the client and the server becomes stateful (remember previous states). Below is a code snippet for creating a session in servlet.

```
//start a session  
HttpSession session = request.getSession(true);
```

When a session is created, a session id is generated. This session id is sent to the client as part of the response. When the client wants to send a request to the server, it attaches the session id to the request. Through this exchange of session-id, the server is able to determine whether it is dealing with a new client or the same one. The exchange of IDs is done through cookies. Another way of creating conversational web apps is through URL rewrite. A session can be ended when it is no longer needed. Below is a code snippet that ends a session.

```
//get the session  
HttpSession session = request.getSession();  
  
//end the session  
session.invalidate();
```

5.2 HTTP

HTTP (**H**yper**T**ext **T**ransfer **P**rotocol) is a commonly used communication protocol by computers over the World Wide Web. The protocol uses a message-based model for transferring data between clients and servers. The client sends a request message over HTTP to the server, and the server responds in kind with a response message, using HTTP. The figure below shows the message-based model used by HTTP.



The protocol is connectionless or stateless. This means the server treats every request message as an autonomous or independent request. It does not link the current request to previous ones. In the next chapters, we will learn how to overcome this shortfall. This will help us to create web applications that are stateful, which are applications that remember previous states. This makes them to keep the conversation with the client.

5.2.1 Anatomy of HTTP request messages

An HTTP request message has headers with corresponding values. The request message may have a body. The body is the payload (data) that the client wants to send to the server. The reason that a payload is optional, is that client it is not always the case that a user will be sending data to the server, at times the user will simply request for data from the server. In such cases, the payload of the request message will be empty.

I visited the TUT website by entering <http://www.tut.ac.za> on the browser. Below is the information I obtained upon inspection of the request message.



According to the figure above, **GET** is the HTTP method used to make the request, the version of HTTP used is **HTTP/2**, and the resource requested is:

<http://www.tut.ac.za/catalogs/masterpage/TUT/favicon.ico>

The Status of the request is 200, meaning the request has been successfully handled by the server. The request also generated some headers. Below are the headers.

Request Headers (1,530 KB)

Accept: image/avif,image/webp,*/*
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Cookie: _ga=GA1.3.438176779.1598713016; _ga_RG2V4FC9SJ=GS1.1.1646824782.73.1.1646825608.0; _ga_XBKPZ9W35L=GS1.1.1646342295.22.0.1646342303.52; _fbp=fb.2.1628849861402.1098031216; ab.storage.deviceld.ddccb14d-5a22-47e4-b5c8-173a04aed25c=%78%22g%22%3A%2223f55545-372f-78f8-b620-f2c4fddd7ca1%22%2C%22c%22%3A1628849865436%2C%22!%22%3A1628849865436%7D; _jhSessionUser_1584225=eyJpZCI6IjhhZDM5ZT14LWQzM2EtNWl3MC1MzLyLWFhN2ZjNDk5YjMxOClslmNyZWF0ZWQiOjE2Mzk1MDE1Njc2NTEsImV4aXNoaW5nJlp0cnVlfQ==; _jhid=e83591e7-11b4-4732-8177-e...2262898570665352779647%7CMCAID%7CNONE%7CMCOPTOUT-1646341618s%7CNONE%7CMCAAMHL-1646939218%7C6%7CMCAAMB-1646939218%7Cj8Odv6LonN4r3an7LhD3WzrU1bUpAkFkkiY1ncBR96t2PTI%7CMCCIDH%7C-640381294%7CMCSYNCSOP%7C411-19062%7CvVersion%7C5.3.0; s_pers=%20v%8%3D1646334683413%7C1740942683413%3B%20v%_s%3DFirst%2520Visit%7C1646336483413%3B%20c19%3Dsd%253Abrowse%253Ajournalsandbooks%7C1646336483416%3B%20v68%3D1646334617554%7C1646336483422%3B; _gid=GA1.3.2083563719.1647055703; WSS_FullScreenMode=false; _gat_gtag_UA_108380564_1=%
Host: www.tut.ac.za
Referer: https://www.tut.ac.za/
Sec-Fetch-Dest: image
Sec-Fetch-Mode: no-cors
Sec-Fetch-Site: same-origin
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0

A brief description of some of the headers is as follows:

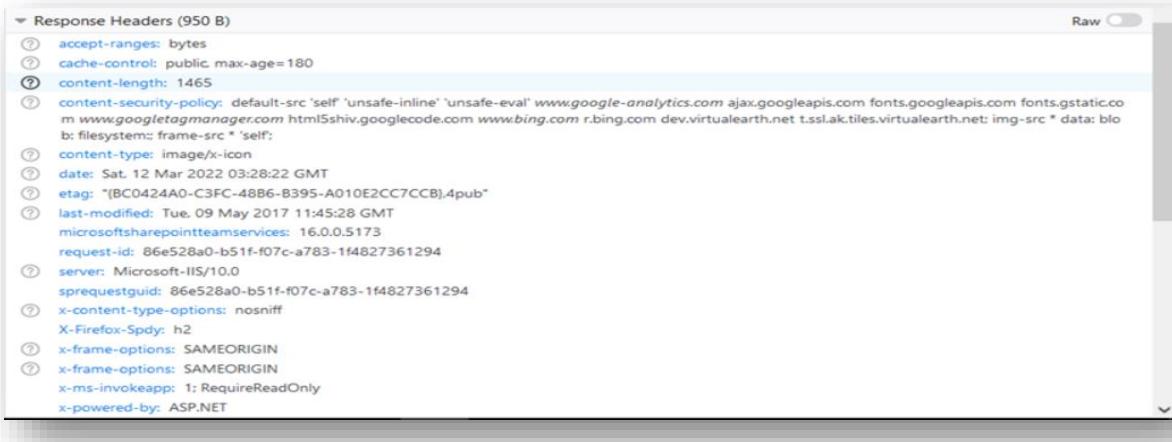
- **Accept:** it describes the kind of data that browser can accept.
- **Cookie:** This is additional information that the server sent to the client to initiate the communication. It is used to track communication between the client and the server.
- **Host:** the host name.
- **User-Agent:** This shows the browser or other client software that generated the request.

5.2.2 Anatomy of HTTP response messages

The first line of the response also has three elements separated by white spaces. The first element is the HTTP version used, the second element is the numerical value of the response status code, and the third element is the description of the status code. Table 2 below shows the applicable HTTP status codes and their description.

HTTP status code	Description
1xx	Information.
2xx	Request has been successfully handled.
3xx	Request is redirect to another resource.
4xx	Request problem.
5xx	Server problem.

Here are the response headers that I obtained when visiting the TUT website:



The screenshot shows the 'Response Headers' section of a browser's developer tools. The headers listed are:

- accept-ranges: bytes
- cache-control: public, max-age=180
- content-length: 1465
- content-security-policy: default-src 'self' 'unsafe-inline' 'unsafe-eval' www.google-analytics.com ajax.googleapis.com fonts.googleapis.com fonts.gstatic.com www.googletagmanager.com html5shiv.googlecode.com www.bing.com r.bing.com dev.virtualearth.net t.ssl.ak.tiles.virtualearth.net; img-src * data: blob: filesystem: frame-src * 'self'; content-type: image/x-icon
- date: Sat, 12 Mar 2022 03:28:22 GMT
- etag: "(BC0424A0-C3FC-48B6-B395-A010E2CC7CCB).4pub"
- last-modified: Tue, 09 May 2017 11:45:28 GMT
- microsoftsharepointteamsservices: 16.0.0.5173
- request-id: 86e528a0-b51f-f07c-a783-1f4827361294
- server: Microsoft-IIS/10.0
- sprequestguid: 86e528a0-b51f-f07c-a783-1f4827361294
- x-content-type-options: nosniff
- X-Firefox-Spdy: h2
- x-frame-options: SAMEORIGIN
- x-frame-options: SAMEORIGIN
- x-msInvokeapp: 1: RequireReadonly
- x-powered-by: ASP.NET

Below is a description of some of the headers:

- **Content-length:** The size of the message in bytes
- **Content-type:** The type of content the server is sending back to the client (browser)
- **Date:** The date and time when the server received the request
- **Server:** The server being used by the host.

5.2.3 HTTP methods

HTTP uses the following methods for communication between computers:

- **GET**: used by the client to get a resource from the server. There's no payload sent to the server.
- **POST**: used by the client to send data to the server for processing. The sent data becomes the payload not visible to the public.
- **DELETE**: used by the client to remove a resource from the server. The method sends unique id of the resource to be removed.
- **PUT**: used by the client to update the state of a resource in the server. The method sends a unique id of the resource to update, and the new data to be used.
- **HEAD**: used by the client to check if certain resources are available in the server. The server responds by sending back the headers, not the actual data.
- **OPTIONS**: used by the client to check the HTTP methods supported by the server.
- **TRACE**: used for debugging/diagnostic purposes.
- **CONNECT**: used to start a two-way communication with the server.

5.3 Example

Problem statement

Create a web application that will allow a user to take an arithmetic test with five questions. The application must first allow the user to enter her/his name. Upon receiving the name, the application must greet the user using their name and explain that there are five questions to be asked. The application must provide a link for the user to start the test.

The application must give the user one question at a time. After the user has answered a question, the application must determine and display the outcome. The outcome must consist of the following information:

- The question asked;
- The answer of the user;
- The correct answer; and
- The outcome, that is whether the user got the question correct or not.

After all the questions have been asked and answered, the application must display a summary report. The report must contain the following information:

- The total number of questions asked;
- The number of questions that the user got correct;
- The number of questions that the user got wrong; and
- The percentage mark obtained.

To do

Create such an application. First discuss the program flow, then identify MVC components which should emanate from the discussion, show the project's structure of the solution. The structure must show identified components. Lastly, do the actual coding.

Program flow

The application must have a welcome page. This should be an **HTML file** for presentation. The HTML file will have a link to a **Servlet** which will be responsible for starting a session. The servlet will start and initialise the session. The servlet will then forward control to a **JSP** that will allow the user to enter their name.

The JSP will send the name to a **Servlet** that will store the given name on a session object. Control will then be forwarded to a **JSP** which will greet the user by name and state that there are five questions to be asked. The JSP will provide a link to a servlet the user to start the test.

The link will take the program flow to a servlet that will get a question and send it back to the user. The question will be displayed in a JSP. The JSP will have a text field to allow the user to provide the answer. The answer of the user will be taken to a servlet that will determine whether the answer is correct or not. The servlet will also update the scores of the user.

The servlet will then forward control to a JSP that will display outcome. If all the questions are exhausted, control will be taken to a summary page which will be a JSP. After the summary has been displayed, program flow will be taken to a servlet that will end the session, and forward control to the home page. If not all the questions have been asked, control will be taken to the servlet that retrieves questions for answering by the user.

MVC components

Consequently our design will consist of the following MVC components:

Views

- index.html → a home page.
- name_entry.jsp → page that allows a user to enter her/his name.
- question.jsp → page that displays a question.
- outcome.jsp → page that displays outcome for each question asked.
- outcome2.jsp → page that displays outcome for the last question asked.
- summary.jsp → page that displays the test summary.

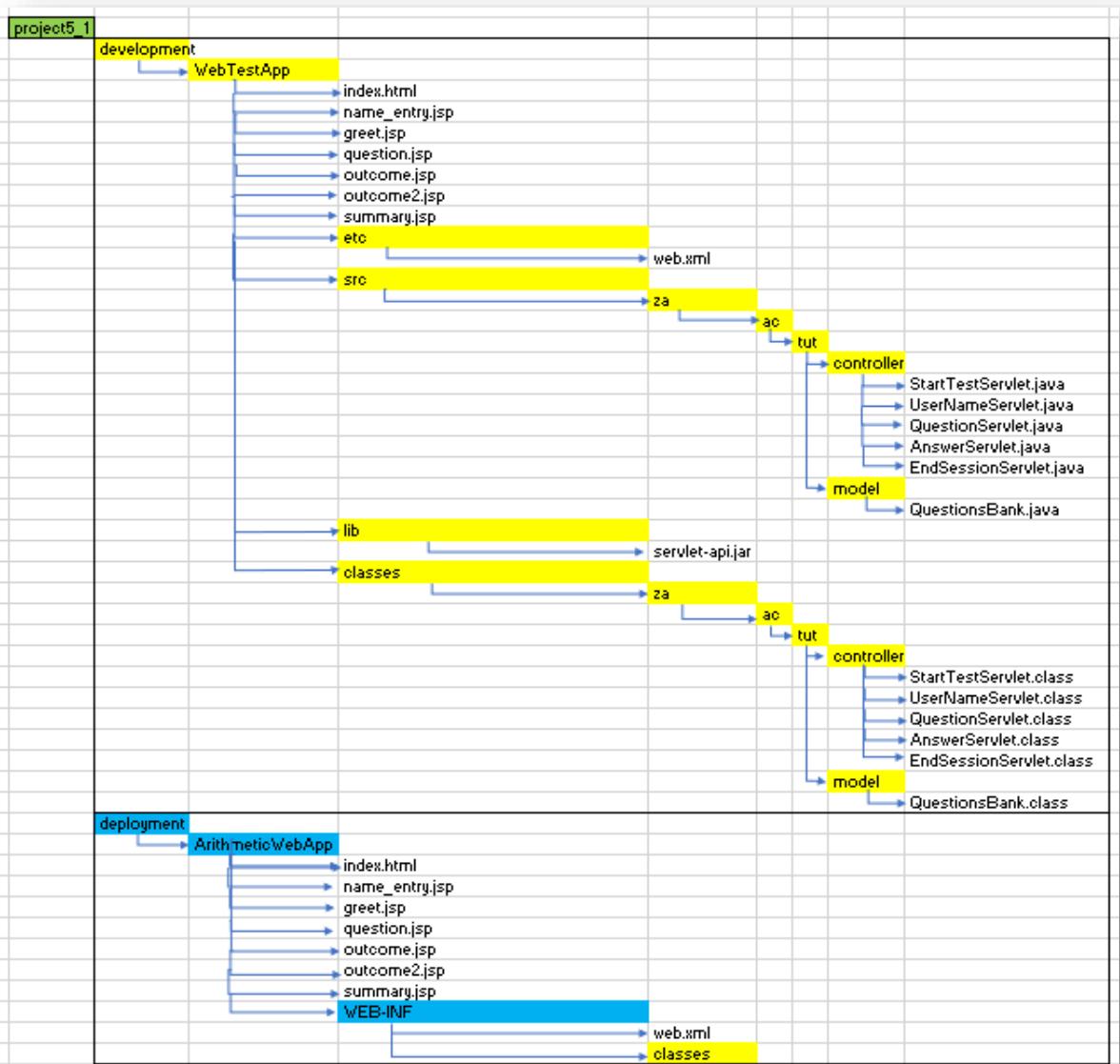
Controllers

- StartTestServlet.java → starts a test session.
- UserNameServlet.java → stores the user name on the session.
- QuestionServlet.java → returns a question.
- AnswerServlet.java → determines if the user's answer is correct or not.
- EndSessionServlet.java → ends a test session.

Model

- QuestionsBank.java → generates questions.

Project structure

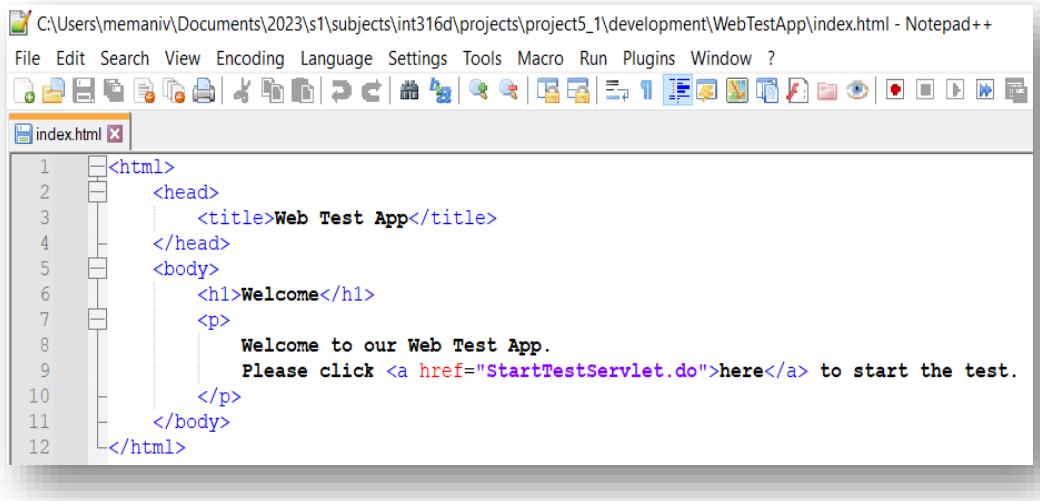


Implementation

1. Create the files.

Create the files as follows:

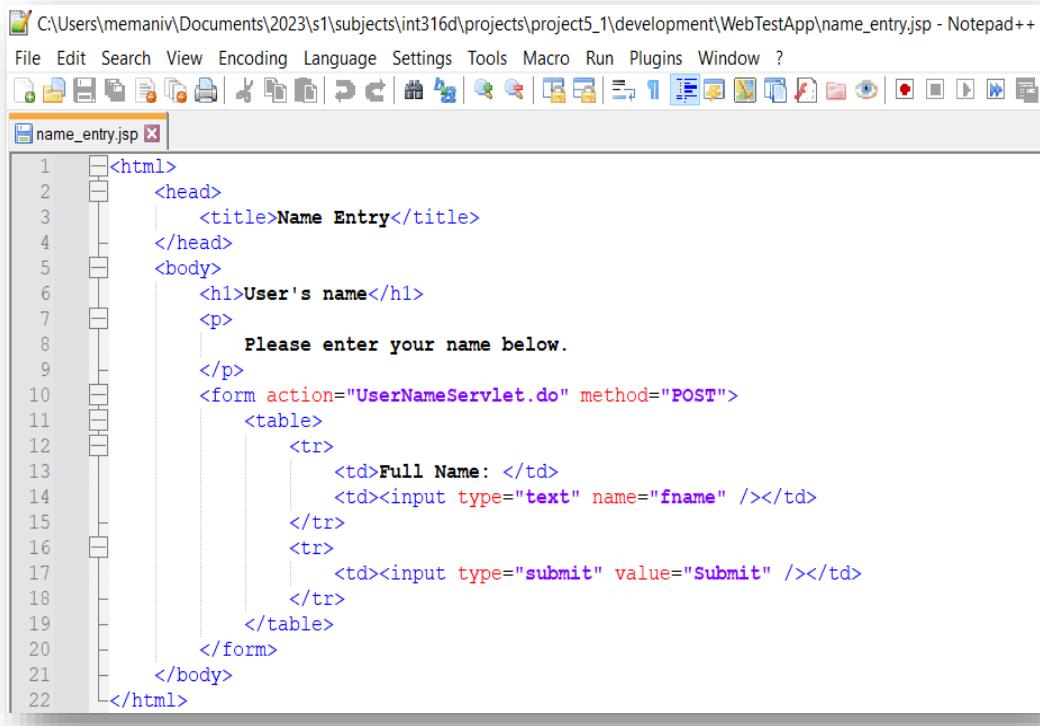
- **html**
 - **index.html**



The screenshot shows the Notepad++ interface with the file 'index.html' open. The code in the editor is:

```
<html>
  <head>
    <title>Web Test App</title>
  </head>
  <body>
    <h1>Welcome</h1>
    <p>
      Welcome to our Web Test App.
      Please click <a href="StartTestServlet.do">here</a> to start the test.
    </p>
  </body>
</html>
```

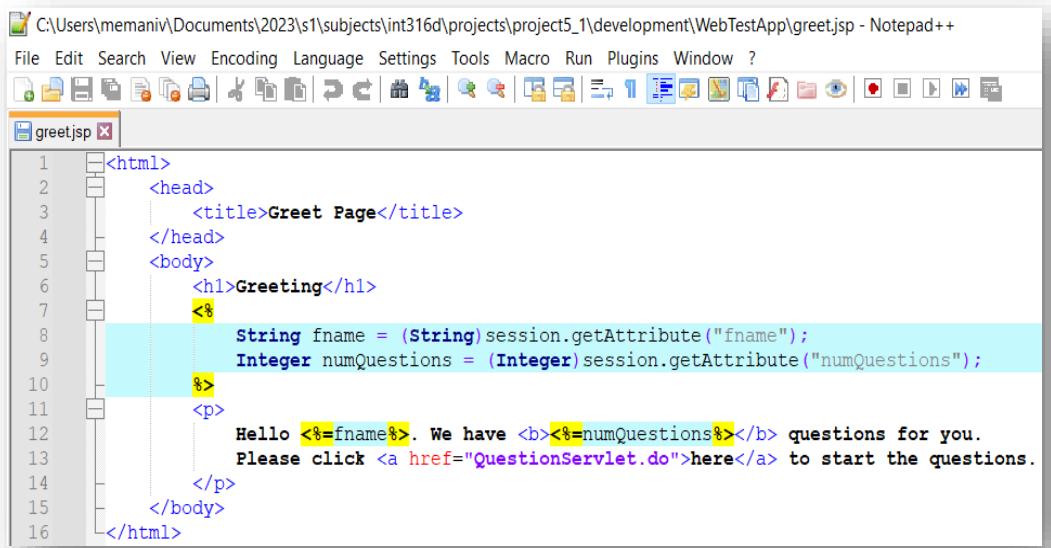
- **jsp**
 - **name_entry.jsp**



The screenshot shows the Notepad++ interface with the file 'name_entry.jsp' open. The code in the editor is:

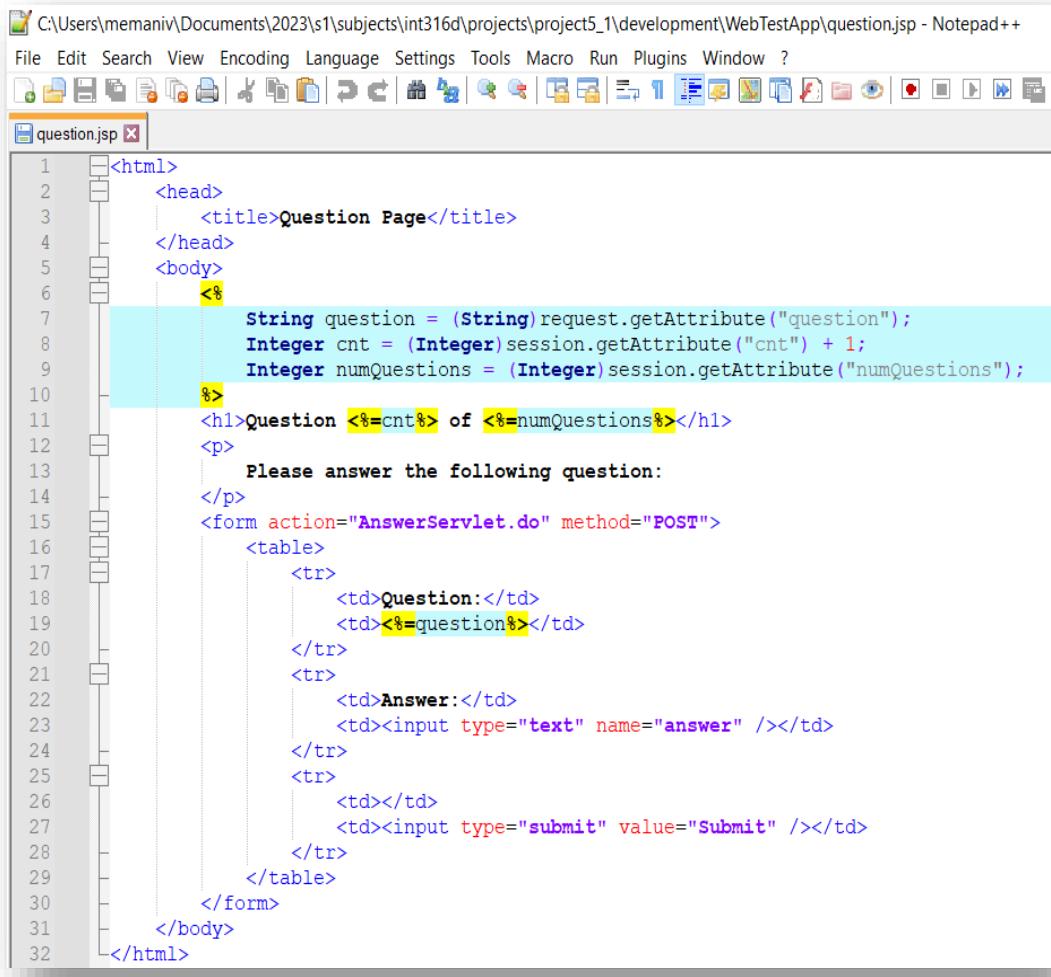
```
<html>
  <head>
    <title>Name Entry</title>
  </head>
  <body>
    <h1>User's name</h1>
    <p>
      Please enter your name below.
    </p>
    <form action="UserNameServlet.do" method="POST">
      <table>
        <tr>
          <td>Full Name:</td>
          <td><input type="text" name="fname" /></td>
        </tr>
        <tr>
          <td><input type="submit" value="Submit" /></td>
        </tr>
      </table>
    </form>
  </body>
</html>
```

▪ greet.jsp



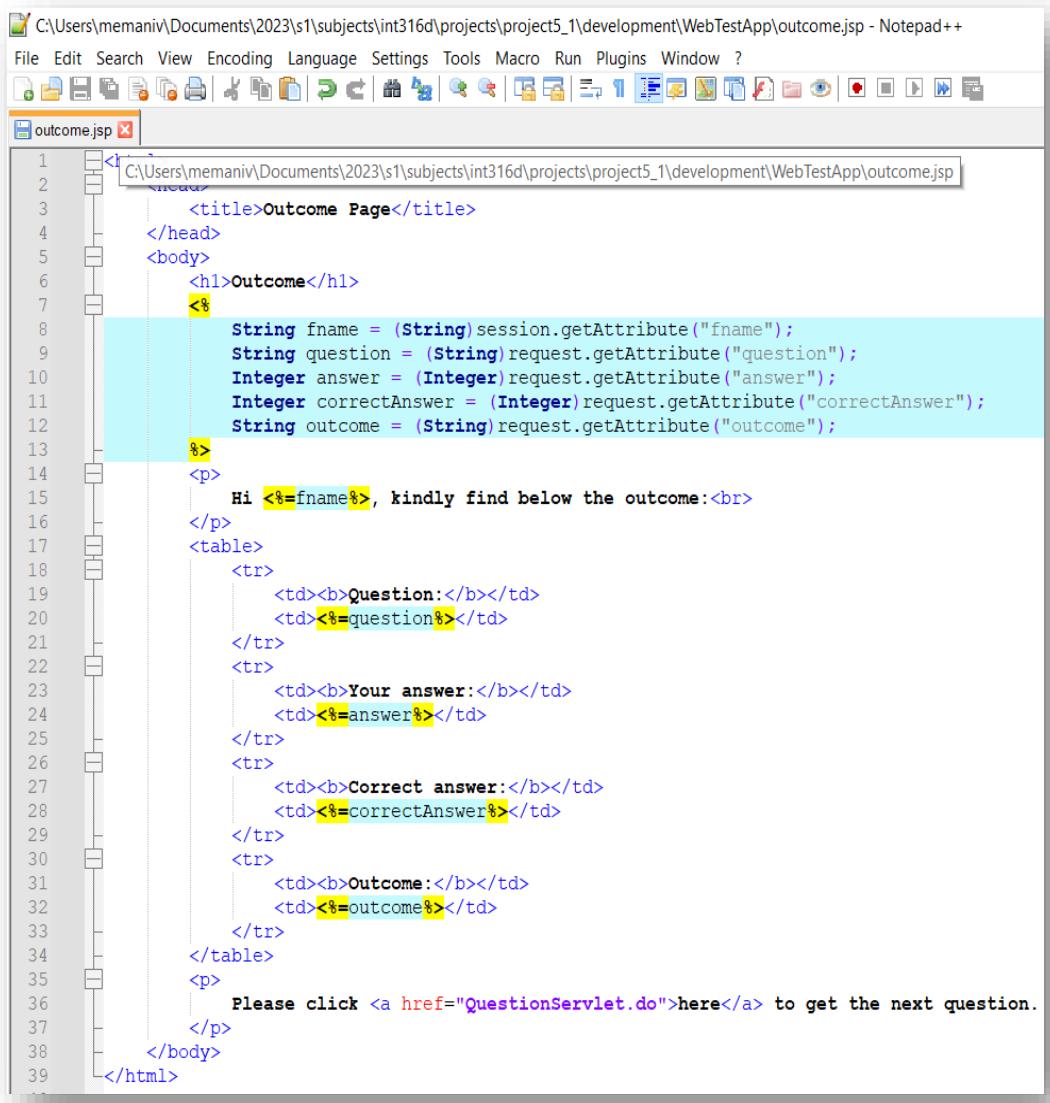
```
C:\Users\memaniiv\Documents\2023\s1\subjects\int316d\projects\project5_1\development\WebTestApp\greet.jsp - Notepad++  
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?  
greet.jsp  
1 <html>  
2   <head>  
3     <title>Greet Page</title>  
4   </head>  
5   <body>  
6     <h1>Greeting</h1>  
7     <%>  
8       String fname = (String)session.getAttribute("fname");  
9       Integer numQuestions = (Integer)session.getAttribute("numQuestions");  
10      %>  
11      <p>  
12        Hello <%=fname%>. We have <b><%=numQuestions%></b> questions for you.  
13        Please click <a href="QuestionServlet.do">here</a> to start the questions.  
14      </p>  
15    </body>  
16  </html>
```

▪ question.jsp



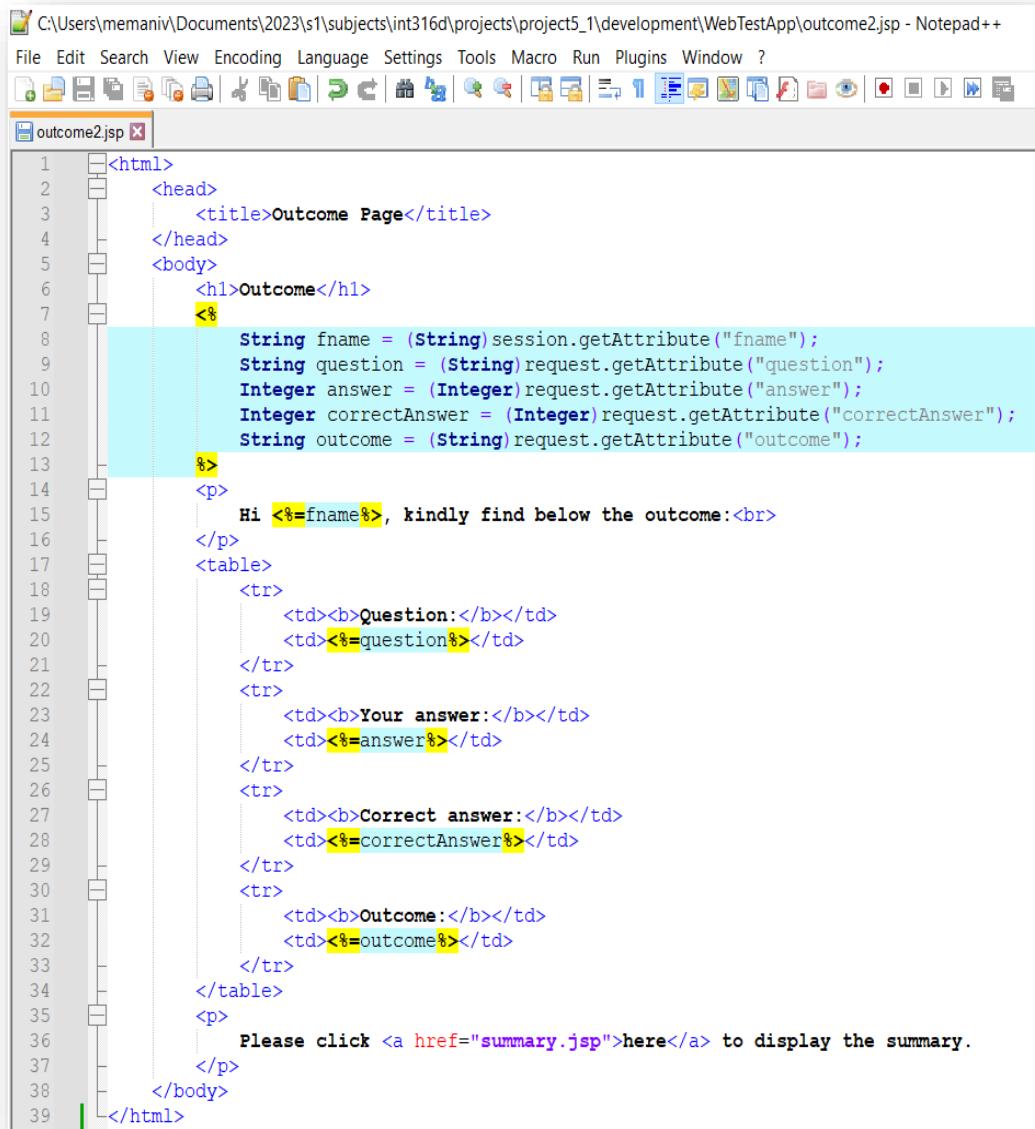
```
C:\Users\memaniiv\Documents\2023\s1\subjects\int316d\projects\project5_1\development\WebTestApp\question.jsp - Notepad++  
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?  
question.jsp  
1 <html>  
2   <head>  
3     <title>Question Page</title>  
4   </head>  
5   <body>  
6     <%>  
7       String question = (String)request.getAttribute("question");  
8       Integer cnt = (Integer)session.getAttribute("cnt") + 1;  
9       Integer numQuestions = (Integer)session.getAttribute("numQuestions");  
10      %>  
11      <h1>Question <%=cnt%> of <%=numQuestions%></h1>  
12      <p>  
13        Please answer the following question:  
14      </p>  
15      <form action="AnswerServlet.do" method="POST">  
16        <table>  
17          <tr>  
18            <td>Question:</td>  
19            <td><%=question%></td>  
20          </tr>  
21          <tr>  
22            <td>Answer:</td>  
23            <td><input type="text" name="answer" /></td>  
24          </tr>  
25          <tr>  
26            <td></td>  
27            <td><input type="submit" value="Submit" /></td>  
28          </tr>  
29        </table>  
30      </form>  
31    </body>  
32  </html>
```

▪ outcome.jsp



```
<html>
    <head>
        <title>Outcome Page</title>
    </head>
    <body>
        <h1>Outcome</h1>
        <%>
        String fname = (String)session.getAttribute("fname");
        String question = (String)request.getAttribute("question");
        Integer answer = (Integer)request.getAttribute("answer");
        Integer correctAnswer = (Integer)request.getAttribute("correctAnswer");
        String outcome = (String)request.getAttribute("outcome");
        <br>
        <p>
            Hi <%=fname%>, kindly find below the outcome:<br>
        </p>
        <table>
            <tr>
                <td><b>Question:</b></td>
                <td><%=question%></td>
            </tr>
            <tr>
                <td><b>Your answer:</b></td>
                <td><%=answer%></td>
            </tr>
            <tr>
                <td><b>Correct answer:</b></td>
                <td><%=correctAnswer%></td>
            </tr>
            <tr>
                <td><b>Outcome:</b></td>
                <td><%=outcome%></td>
            </tr>
        </table>
        <p>
            Please click <a href="QuestionServlet.do">here</a> to get the next question.
        </p>
    </body>
</html>
```

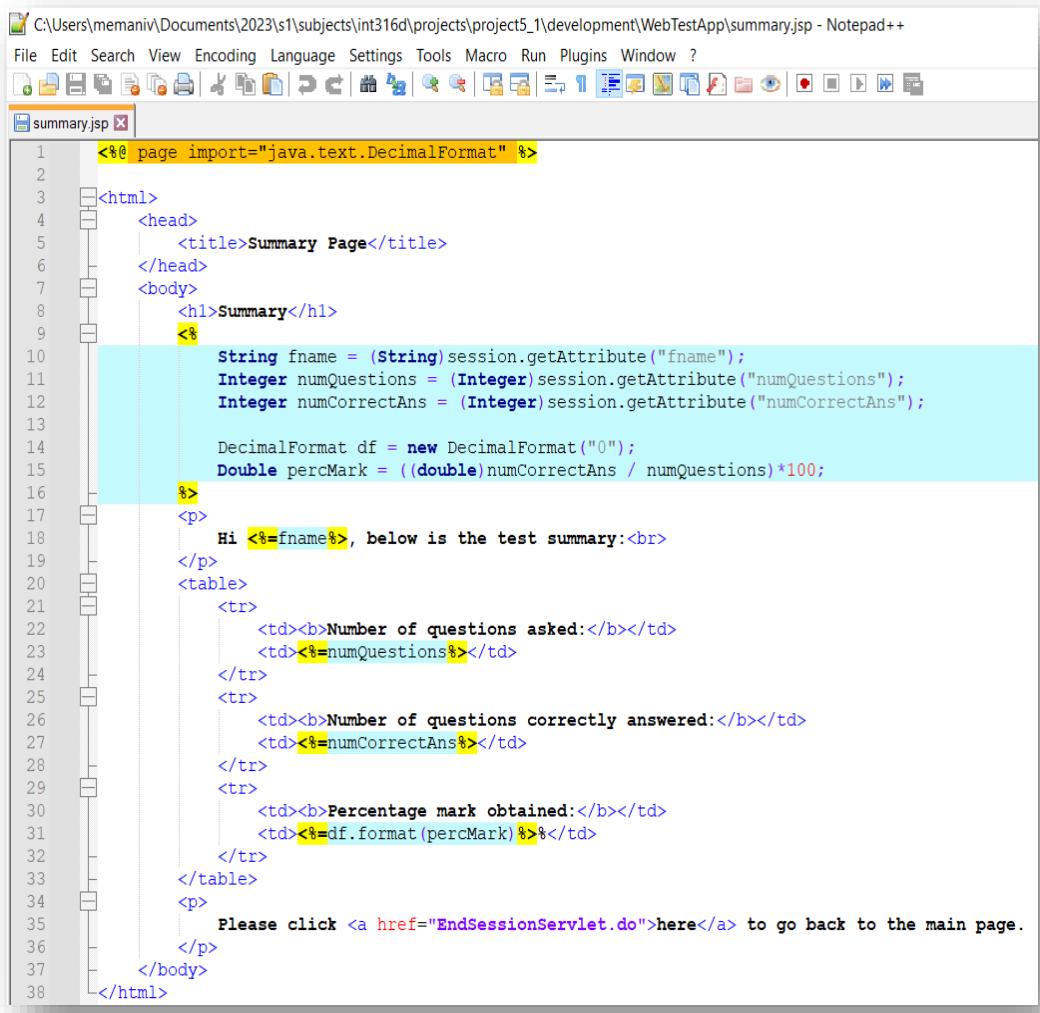
- **outcome2.jsp**



The screenshot shows the Notepad++ editor with the file "outcome2.jsp" open. The code is a JSP page with the following structure:

```
<html>
    <head>
        <title>Outcome Page</title>
    </head>
    <body>
        <h1>Outcome</h1>
        <%>
        String fname = (String)session.getAttribute("fname");
        String question = (String)request.getAttribute("question");
        Integer answer = (Integer)request.getAttribute("answer");
        Integer correctAnswer = (Integer)request.getAttribute("correctAnswer");
        String outcome = (String)request.getAttribute("outcome");
        <%>
        <p>
            Hi <%=fname%>, kindly find below the outcome:<br>
        </p>
        <table>
            <tr>
                <td><b>Question:</b></td>
                <td><%=question%></td>
            </tr>
            <tr>
                <td><b>Your answer:</b></td>
                <td><%=answer%></td>
            </tr>
            <tr>
                <td><b>Correct answer:</b></td>
                <td><%=correctAnswer%></td>
            </tr>
            <tr>
                <td><b>Outcome:</b></td>
                <td><%=outcome%></td>
            </tr>
        </table>
        <p>
            Please click <a href="summary.jsp">here</a> to display the summary.
        </p>
    </body>
</html>
```

▪ summary.jsp



The screenshot shows the Notepad++ editor with the file "summary.jsp" open. The code is a JSP page that displays a test summary for a user based on session attributes. The code uses Java scriptlets and JSTL tags to format and present the data.

```
<%@ page import="java.text.DecimalFormat" %>
<html>
<head>
<title>Summary Page</title>
</head>
<body>
<h1>Summary</h1>
<%
String fname = (String)session.getAttribute("fname");
Integer numQuestions = (Integer)session.getAttribute("numQuestions");
Integer numCorrectAns = (Integer)session.getAttribute("numCorrectAns");

DecimalFormat df = new DecimalFormat("0");
Double percMark = ((double)numCorrectAns / numQuestions)*100;
%>
<p>
Hi <%=fname%>, below is the test summary:<br>
</p>
<table>
<tr>
<td><b>Number of questions asked:</b></td>
<td><%=numQuestions%></td>
</tr>
<tr>
<td><b>Number of questions correctly answered:</b></td>
<td><%=numCorrectAns%></td>
</tr>
<tr>
<td><b>Percentage mark obtained:</b></td>
<td><%=df.format(percMark)%></td>
</tr>
</table>
<p>
Please click <a href="EndSessionServlet.do">here</a> to go back to the main page.
</p>
</body>
</html>
```

- **servlet**

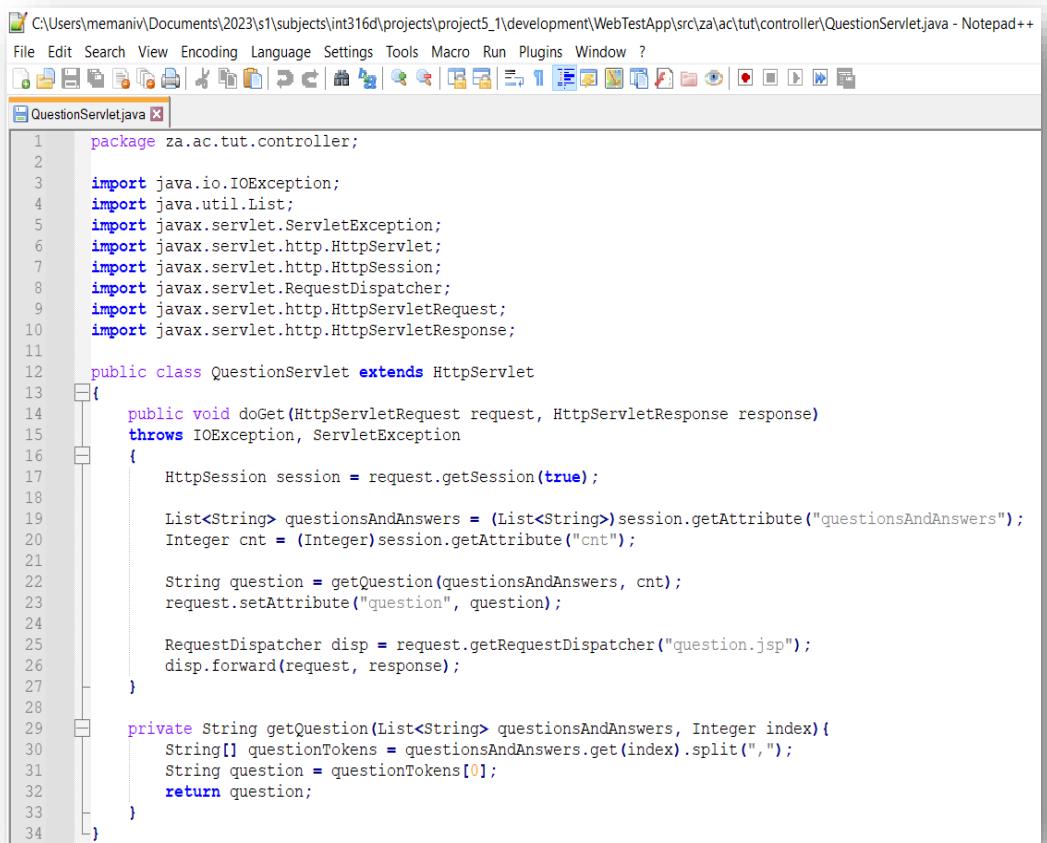
- **StartTestServlet.java**

```
C:\Users\memaniiv\Documents\2023\st\subjects\int316d\projects\project5_1\development\WebTestApp\src\za\ac\tut\controller\StartTestServlet.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
StartTestServlet.java x
1 package za.ac.tut.controller;
2
3 import java.util.List;
4 import java.io.IOException;
5 import za.ac.tut.model.QuestionsBank;
6 import javax.servlet.ServletException;
7 import javax.servlet.http.HttpServlet;
8 import javax.servlet.http.HttpSession;
9 import javax.servlet.RequestDispatcher;
10 import javax.servlet.http.HttpServletRequest;
11 import javax.servlet.http.HttpServletResponse;
12
13 public class StartTestServlet extends HttpServlet
14 {
15     public void doGet(HttpServletRequest request, HttpServletResponse response)
16         throws IOException, ServletException
17     {
18         HttpSession session = request.getSession(true);
19
20         QuestionsBank qb = new QuestionsBank();
21         List<String> questionsAndAnswers = qb.getQuestionsAndAnswers();
22         Integer numQuestions = questionsAndAnswers.size();
23         Integer cnt = 0;
24         Integer numCorrectAns = 0;
25
26         session.setAttribute("questionsAndAnswers", questionsAndAnswers);
27         session.setAttribute("numQuestions", numQuestions);
28         session.setAttribute("cnt", cnt);
29         session.setAttribute("numCorrectAns", numCorrectAns);
30
31         RequestDispatcher disp = request.getRequestDispatcher("name_entry.jsp");
32         disp.forward(request, response);
33     }
34 }
```

- **UserNameServlet.java**

```
C:\Users\memaniiv\Documents\2023\st\subjects\int316d\projects\project5_1\development\WebTestApp\src\za\ac\tut\controller\UserNameServlet.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
UserNameServlet.java x
1 package za.ac.tut.controller;
2
3 import java.io.IOException;
4 import javax.servlet.ServletException;
5 import javax.servlet.http.HttpServlet;
6 import javax.servlet.http.HttpServletRequest;
7 import javax.servlet.RequestDispatcher;
8 import javax.servlet.http.HttpServletResponse;
9
10 public class UserNameServlet extends HttpServlet
11 {
12     public void doPost(HttpServletRequest request, HttpServletResponse response)
13         throws IOException, ServletException
14     {
15         HttpSession session = request.getSession(true);
16         String fname = (String)request.getParameter("fname");
17
18         session.setAttribute("fname", fname);
19
20         RequestDispatcher disp = request.getRequestDispatcher("greet.jsp");
21         disp.forward(request, response);
22     }
23 }
24 }
```

▪ QuestionServlet.java

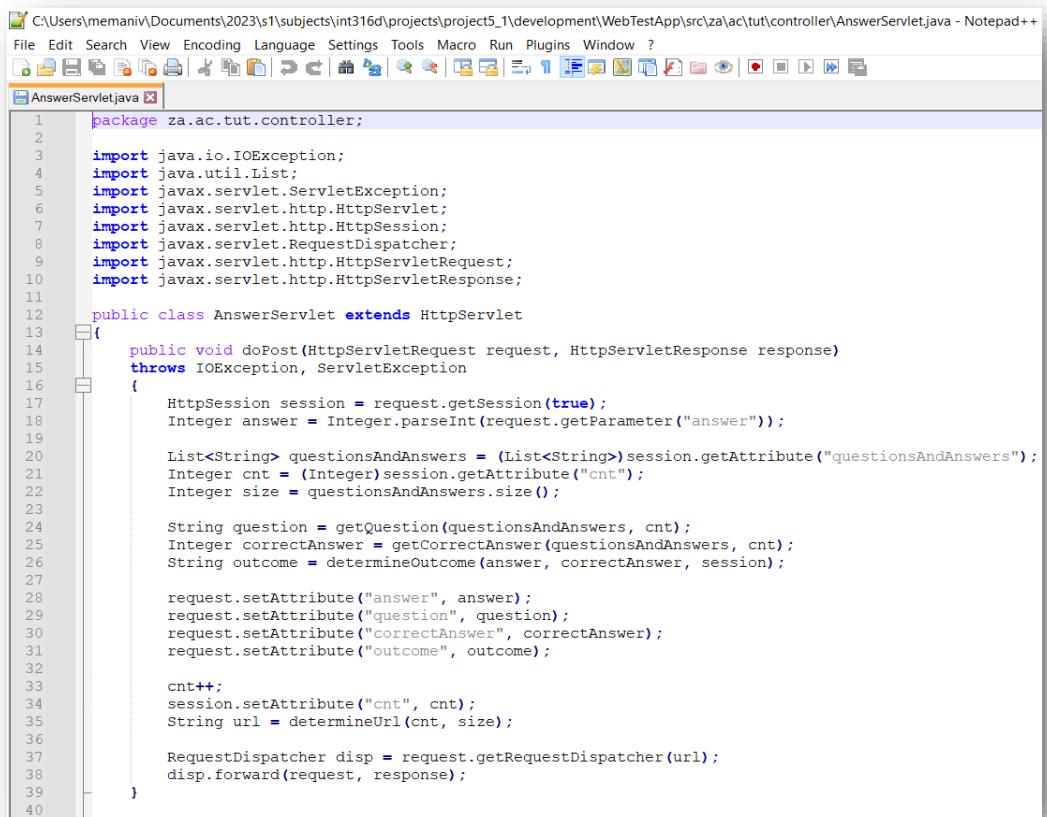


```

1 package za.ac.tut.controller;
2
3 import java.io.IOException;
4 import java.util.List;
5 import javax.servlet.ServletException;
6 import javax.servlet.http.HttpServlet;
7 import javax.servlet.http.HttpSession;
8 import javax.servlet.RequestDispatcher;
9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11
12 public class QuestionServlet extends HttpServlet
13 {
14     public void doGet(HttpServletRequest request, HttpServletResponse response)
15         throws IOException, ServletException
16     {
17         HttpSession session = request.getSession(true);
18
19         List<String> questionsAndAnswers = (List<String>)session.getAttribute("questionsAndAnswers");
20         Integer cnt = (Integer)session.getAttribute("cnt");
21
22         String question = getQuestion(questionsAndAnswers, cnt);
23         request.setAttribute("question", question);
24
25         RequestDispatcher disp = request.getRequestDispatcher("question.jsp");
26         disp.forward(request, response);
27     }
28
29     private String getQuestion(List<String> questionsAndAnswers, Integer index){
30         String[] questionTokens = questionsAndAnswers.get(index).split(",");
31         String question = questionTokens[0];
32         return question;
33     }
34 }

```

▪ AnswerServlet.java



```

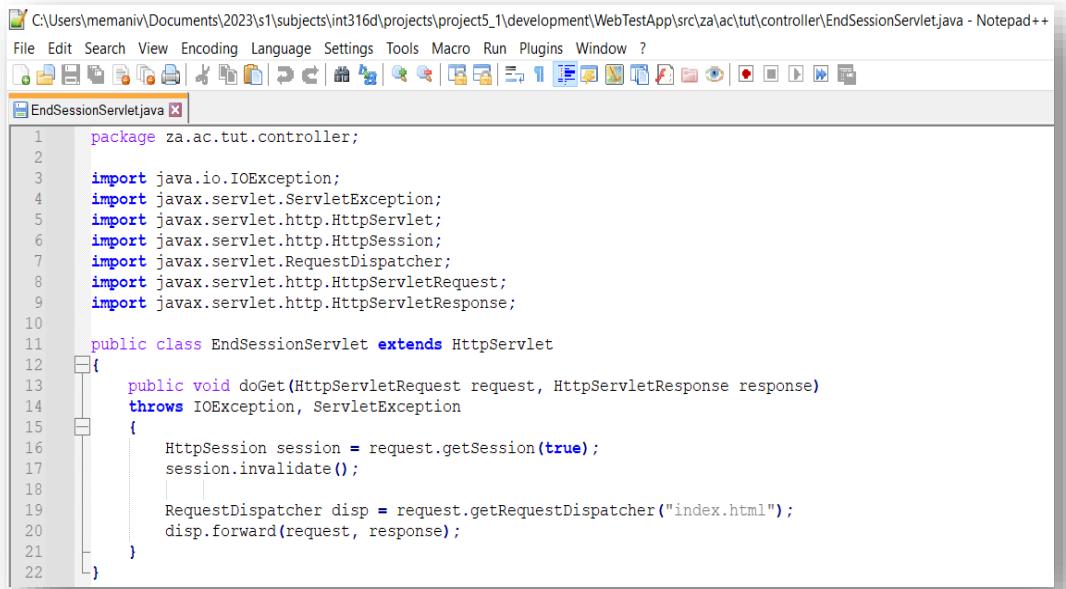
1 package za.ac.tut.controller;
2
3 import java.io.IOException;
4 import java.util.List;
5 import javax.servlet.ServletException;
6 import javax.servlet.http.HttpServlet;
7 import javax.servlet.http.HttpSession;
8 import javax.servlet.RequestDispatcher;
9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11
12 public class AnswerServlet extends HttpServlet
13 {
14     public void doPost(HttpServletRequest request, HttpServletResponse response)
15         throws IOException, ServletException
16     {
17         HttpSession session = request.getSession(true);
18         Integer answer = Integer.parseInt(request.getParameter("answer"));
19
20         List<String> questionsAndAnswers = (List<String>)session.getAttribute("questionsAndAnswers");
21         Integer cnt = (Integer)session.getAttribute("cnt");
22         Integer size = questionsAndAnswers.size();
23
24         String question = getQuestion(questionsAndAnswers, cnt);
25         Integer correctAnswer = getCorrectAnswer(questionsAndAnswers, cnt);
26         String outcome = determineOutcome(answer, correctAnswer, session);
27
28         request.setAttribute("answer", answer);
29         request.setAttribute("question", question);
30         request.setAttribute("correctAnswer", correctAnswer);
31         request.setAttribute("outcome", outcome);
32
33         cnt++;
34         session.setAttribute("cnt", cnt);
35         String url = determineUrl(cnt, size);
36
37         RequestDispatcher disp = request.getRequestDispatcher(url);
38         disp.forward(request, response);
39     }
40 }

```

```

41     private String getQuestion(List<String> questionsAndAnswers, Integer index){
42         String[] questionTokens = questionsAndAnswers.get(index).split(",");
43         String question = questionTokens[0];
44         return question;
45     }
46
47     private Integer getCorrectAnswer(List<String> questionsAndAnswers, Integer index){
48         String[] questionTokens = questionsAndAnswers.get(index).split(",");
49         Integer correctAnswer = Integer.parseInt(questionTokens[1].trim());
50         return correctAnswer;
51     }
52
53     private String determineOutcome(Integer answer, Integer correctAnswer, HttpSession session){
54         String outcome = "";
55
56         if(answer.equals(correctAnswer)){
57             outcome = "You are correct.";
58             Integer numCorrectAns = (Integer)session.getAttribute("numCorrectAns");
59             numCorrectAns++;
60             session.setAttribute("numCorrectAns", numCorrectAns);
61         } else {
62             outcome = "You are wrong.";
63         }
64
65         return outcome;
66     }
67
68     private String determineUrl(Integer cnt, Integer size){
69         String url = "";
70
71         if(cnt == size){
72             url = "outcome2.jsp";
73         } else {
74             url = "outcome.jsp";
75         }
76
77         return url;
78     }
79 }
```

▪ QuestionServlet.java



```

C:\Users\memaniv\Documents\2023\s1\subjects\int316d\projects\project5_1\development\WebTestApp\src\za\ac\tut\controller\EndSessionServlet.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
EndSessionServlet.java
1 package za.ac.tut.controller;
2
3 import java.io.IOException;
4 import javax.servlet.ServletException;
5 import javax.servlet.http.HttpServlet;
6 import javax.servlet.http.HttpSession;
7 import javax.servlet.RequestDispatcher;
8 import javax.servlet.http.HttpServletRequest;
9 import javax.servlet.http.HttpServletResponse;
10
11 public class EndSessionServlet extends HttpServlet
12 {
13     public void doGet(HttpServletRequest request, HttpServletResponse response)
14         throws IOException, ServletException
15     {
16         HttpSession session = request.getSession(true);
17         session.invalidate();
18
19         RequestDispatcher disp = request.getRequestDispatcher("index.html");
20         disp.forward(request, response);
21     }
22 }
```

- o model

- QuestionsBank.java

```
C:\Users\memani\notepad++\C:\Users\memani\Documents\2023\s1\subjects\int316d\projects\project5_1\development\WebTestApp\src\za\ac\tut\model\QuestionsBank.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
QuestionsBank.java x
1 package za.ac.tut.model;
2
3 import java.util.List;
4 import java.util.ArrayList;
5
6 public class QuestionsBank
7 {
8     public QuestionsBank()
9     {
10
11     }
12
13     public List<String> getQuestionsAndAnswers()
14     {
15         List<String> questionsAndAnswers = new ArrayList<>();
16
17         questionsAndAnswers.add("1 * 3 = ?,3");
18         questionsAndAnswers.add("100 * 3 = ?,300");
19         questionsAndAnswers.add("10 / 2 = ?,5");
20         questionsAndAnswers.add("100 - 1 = ?,99");
21         questionsAndAnswers.add("0 + 1 = ?,1");
22
23     }
24 }
```

- o xml

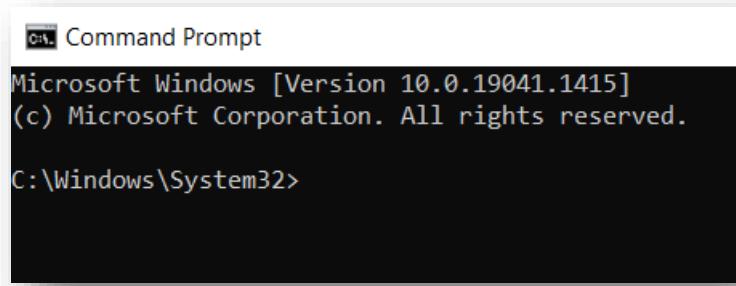
- web.xml

```
C:\Users\memani\notepad++\C:\Users\memani\Documents\2023\s1\subjects\int316d\projects\project5_1\development\WebTestApp\etc\web.xml - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
web.xml x
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
5                         http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
6     version="3.1"
7     metadata-complete="true">
8
9     <!-- Welcoming page -->
10    <welcome-file-list>
11        <welcome-file>index.html</welcome-file>
12    </welcome-file-list>
13
14    <!-- Mapping servlet to actual class -->
15    <servlet>
16        <servlet-name>StartTestServlet</servlet-name>
17        <servlet-class>za.ac.tut.controller.StartTestServlet</servlet-class>
18    </servlet>
19
20    <servlet>
21        <servlet-name>UserNameServlet</servlet-name>
22        <servlet-class>za.ac.tut.controller.UserNameServlet</servlet-class>
23    </servlet>
24
25    <servlet>
26        <servlet-name>QuestionServlet</servlet-name>
27        <servlet-class>za.ac.tut.controller.QuestionServlet</servlet-class>
28    </servlet>
29
30    <servlet>
31        <servlet-name>AnswerServlet</servlet-name>
32        <servlet-class>za.ac.tut.controller.AnswerServlet</servlet-class>
33    </servlet>
34
35    <servlet>
36        <servlet-name>EndSessionServlet</servlet-name>
37        <servlet-class>za.ac.tut.controller.EndSessionServlet</servlet-class>
38    </servlet>
39
```

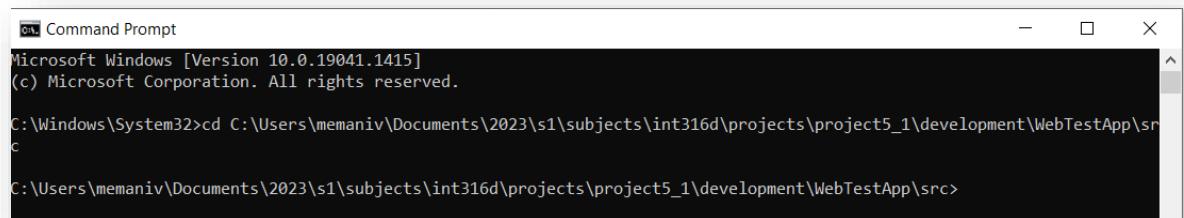
```
40      <!-- Mapping url to a servlet -->
41      <servlet-mapping>
42          <servlet-name>StartTestServlet</servlet-name>
43          <url-pattern>/StartTestServlet.do</url-pattern>
44      </servlet-mapping>
45
46      <servlet-mapping>
47          <servlet-name>UserNameServlet</servlet-name>
48          <url-pattern>/UserNameServlet.do</url-pattern>
49      </servlet-mapping>
50
51      <servlet-mapping>
52          <servlet-name>QuestionServlet</servlet-name>
53          <url-pattern>/QuestionServlet.do</url-pattern>
54      </servlet-mapping>
55
56      <servlet-mapping>
57          <servlet-name>AnswerServlet</servlet-name>
58          <url-pattern>/AnswerServlet.do</url-pattern>
59      </servlet-mapping>
60
61      <servlet-mapping>
62          <servlet-name>EndSessionServlet</servlet-name>
63          <url-pattern>/EndSessionServlet.do</url-pattern>
64      </servlet-mapping>
65
66  </web-app>
```

2. Compile source code

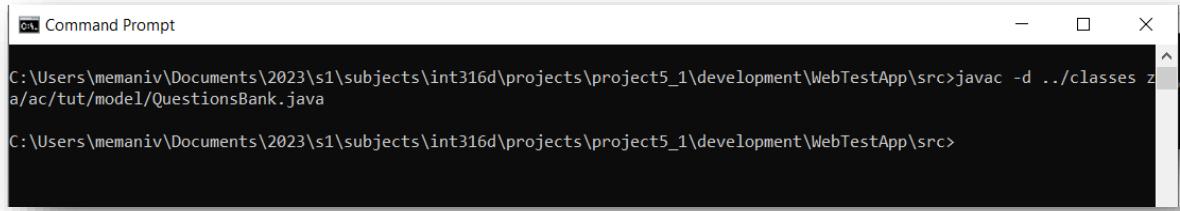
Open the command line.



Change directory to the **src** directory of **WebTestApp**.



Compile QuestionsBank.java

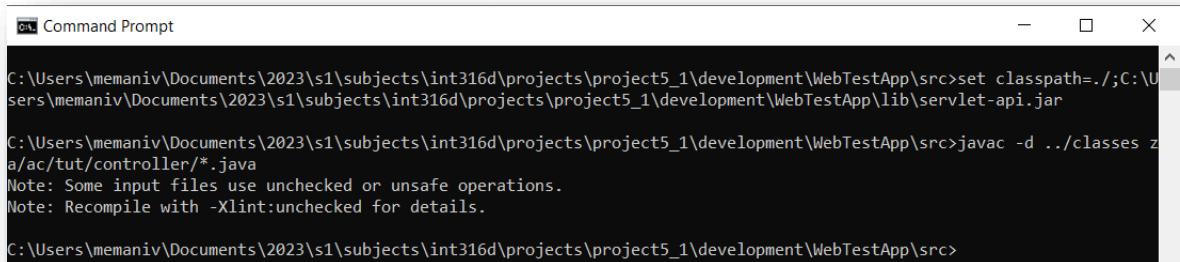


```
Command Prompt

C:\Users\memaniv\Documents\2023\s1\subjects\int316d\projects\project5_1\development\WebTestApp\src>javac -d ..\classes za/ac/tut/model/QuestionsBank.java

C:\Users\memaniv\Documents\2023\s1\subjects\int316d\projects\project5_1\development\WebTestApp\src>
```

Set the classpath and compile all the controllers



```
Command Prompt

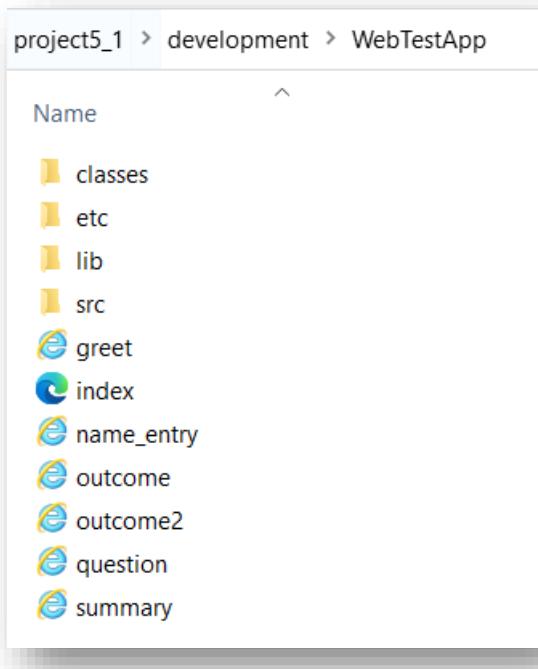
C:\Users\memaniv\Documents\2023\s1\subjects\int316d\projects\project5_1\development\WebTestApp\src>set classpath=.;C:\Users\memaniv\Documents\2023\s1\subjects\int316d\projects\project5_1\development\WebTestApp\lib\servlet-api.jar

C:\Users\memaniv\Documents\2023\s1\subjects\int316d\projects\project5_1\development\WebTestApp\src>javac -d ..\classes za/ac/tut/controller/*.java
Note: Some input files use unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

C:\Users\memaniv\Documents\2023\s1\subjects\int316d\projects\project5_1\development\WebTestApp\src>
```

3. View development files

View all the files.



Check under the **classes** folder.

project5_1 > development > WebTestApp > classes > za > ac > tut > controller
Name

AnswerServlet.class
EndSessionServlet.class
QuestionServlet.class
StartTestServlet.class
UserNameServlet.class

project5_1 > development > WebTestApp > classes > za > ac > tut > model
Name Type

QuestionsBank.class CLASS File

View inside **etc** folder.

project5_1 > development > WebTestApp > etc
Name

web Type XML Document

View inside the **lib** folder.

project5_1 > development > WebTestApp > lib
Name

servlet-api Type Executable Jar File

View inside the **src** folder.

Name	Type
AnswerServlet	JAVA File
EndSessionServlet	JAVA File
QuestionServlet	JAVA File
StartTestServlet	JAVA File
UserNameServlet	JAVA File

Name	Type
QuestionsBank	JAVA File

4. Prepare for deployment

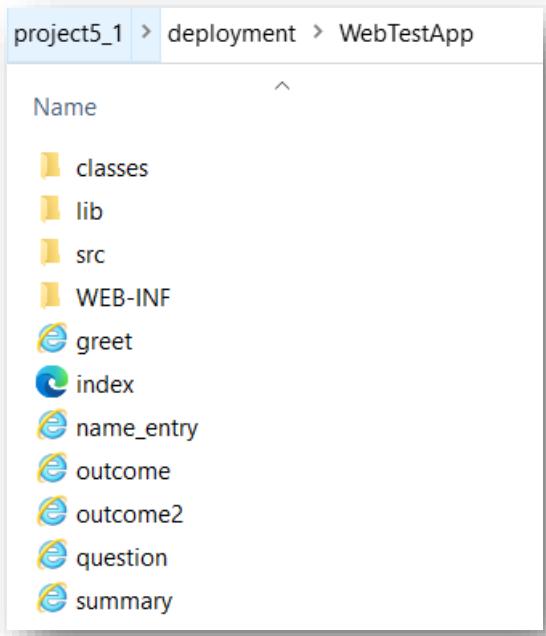
Copy the entire **WebTestApp** folder and paste it under the **deployment** folder.

Name
WebTestApp

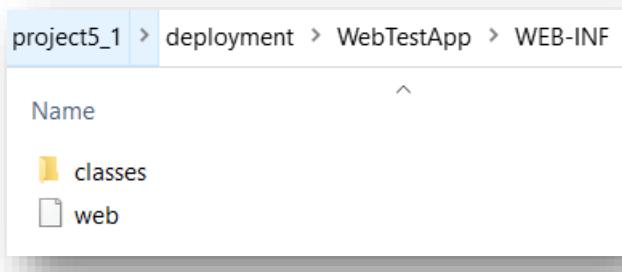
Expand **WebTestApp**.

Name
classes
etc
lib
src
greet
index
name_entry
outcome
outcome2
question
summary

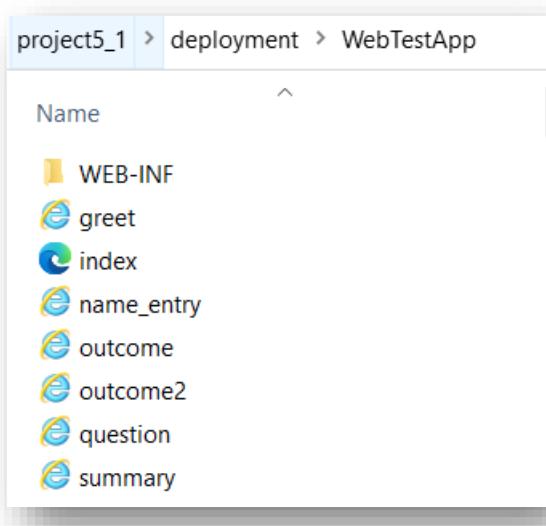
Change **etc** to **WEB-INF**.



Move the **classes** folder to **WEB-INF**.

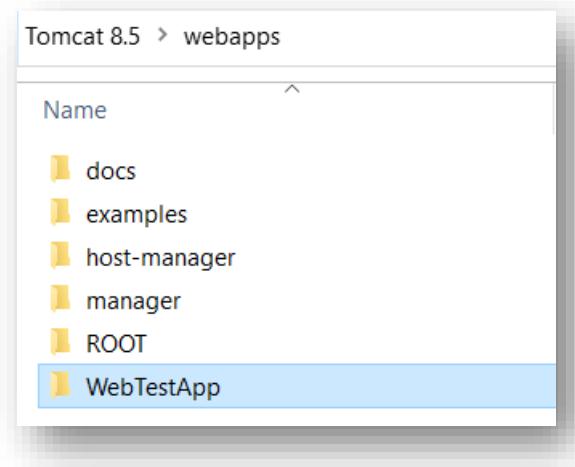


Delete the **src** and **lib** folders.

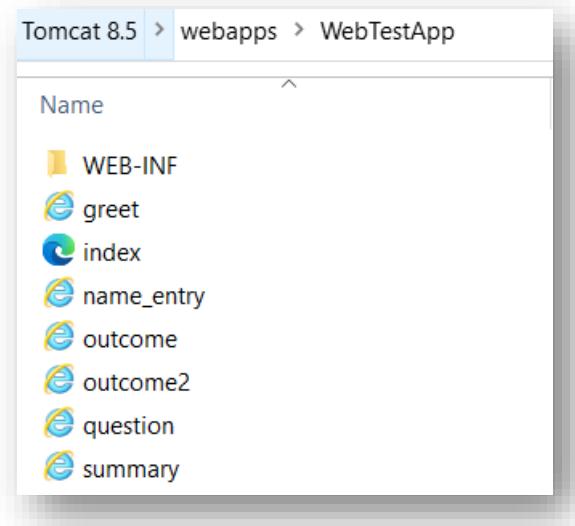


5. Deploy the web application

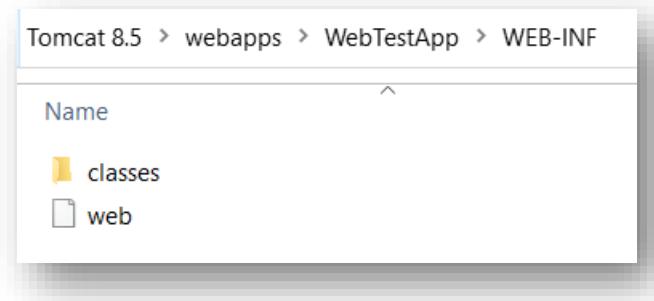
Copy the **WebTestApp** folder from the **deployment** and paste it on the **webapps** folder of **Tomcat**.



Expand **WebTestApp**.



Expand **WEB-INF**.



Go to the **bin** directory of **Tomcat**.

Name	Date modified	Type	Size
bootstrap	2022/11/16 15:34	Executable Jar File	36 KB
catalina	2022/11/16 15:34	Windows Batch File	17 KB
ciphers	2022/11/16 15:34	Windows Batch File	3 KB
configtest	2022/11/16 15:34	Windows Batch File	2 KB
digest	2022/11/16 15:34	Windows Batch File	3 KB
service	2022/11/16 15:34	Windows Batch File	9 KB
setclasspath	2022/11/16 15:34	Windows Batch File	4 KB
shutdown	2022/11/16 15:34	Windows Batch File	2 KB
startup	2022/11/16 15:34	Windows Batch File	2 KB
Tomcat8	2022/11/16 15:34	Application	142 KB
Tomcat8w	2022/11/16 15:34	Application	126 KB
tomcat-juli	2022/11/16 15:34	Executable Jar File	52 KB
tool-wrapper	2022/11/16 15:34	Windows Batch File	5 KB
version	2022/11/16 15:34	Windows Batch File	2 KB

Copy the address of the **bin** directory.

C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin

Launch the command line.

```
cmd Command Prompt
Microsoft Windows [Version 10.0.19041.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>
```

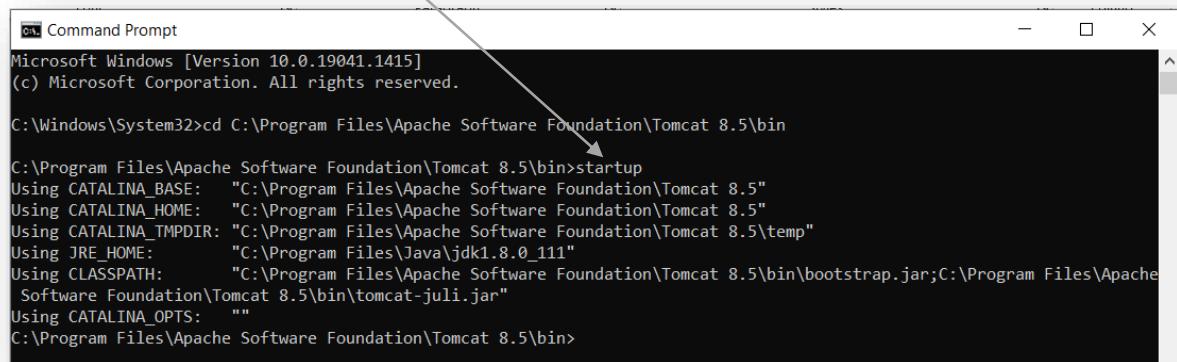
Change location to the **bin** directory of **Tomcat**.

```
cmd Command Prompt
Microsoft Windows [Version 10.0.19041.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin

C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>
```

Start Tomcat by typing "**startup**" on the command line and then click the **Enter** key.



```
Command Prompt
Microsoft Windows [Version 10.0.19041.1415]
(c) Microsoft Corporation. All rights reserved.

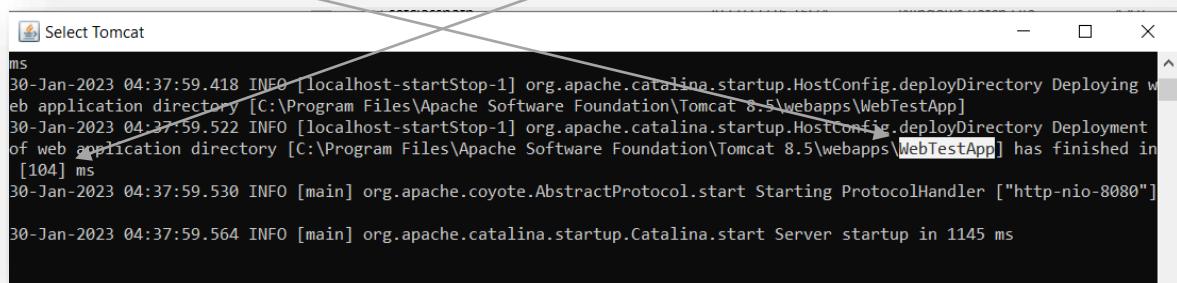
C:\Windows\System32>cd C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin

C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>startup
Using CATALINA_BASE: "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_HOME: "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_TMPDIR: "C:\Program Files\Apache Software Foundation\Tomcat 8.5\temp"
Using JRE_HOME: "C:\Program Files\Java\jdk1.8.0_111"
Using CLASSPATH: "C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\bootstrap.jar;C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\tomcat-juli.jar"
Using CATALINA_OPTS: ""

C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>
```

Output:

The **WebTestApp** application is deployed in **104 ms**.



```
Select Tomcat
ms
30-Jan-2023 04:37:59.418 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory [C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\WebTestApp]
30-Jan-2023 04:37:59.522 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application directory [C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\WebTestApp] has finished in [104] ms
30-Jan-2023 04:37:59.530 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["http-nio-8080"]
30-Jan-2023 04:37:59.564 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in 1145 ms
```

6. Run the web application

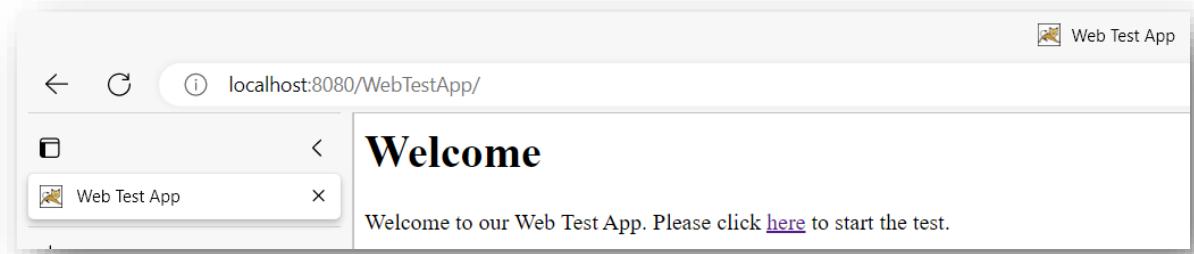
Open the browser and type:

`http://localhost:8080/WebTestApp`

Note:

- **http:** is the protocol used to communicate between the browser and the Web Server, Tomcat.
- **localhost:** is the address of the machine that hosts the Web Server. In this instance the Web Server is running on the local machine.
- **8080:** is the port where the Web Server is running.
- **WebTestApp:** is the name of the web application deployed on the server that we want to talk to.

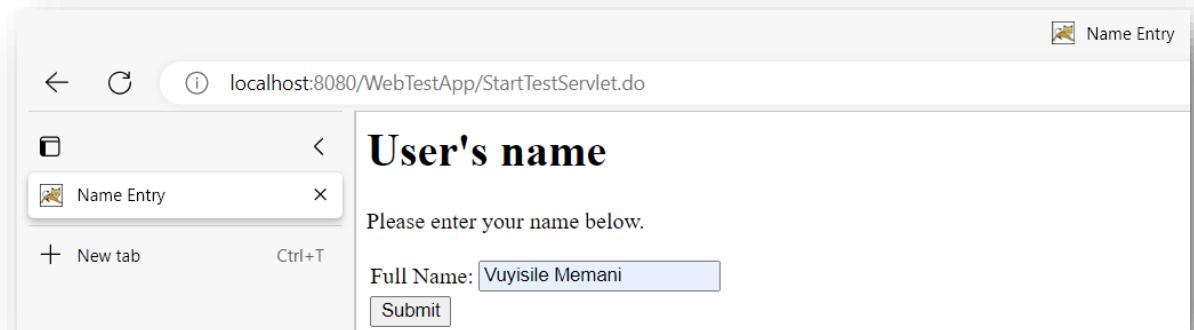
Output:



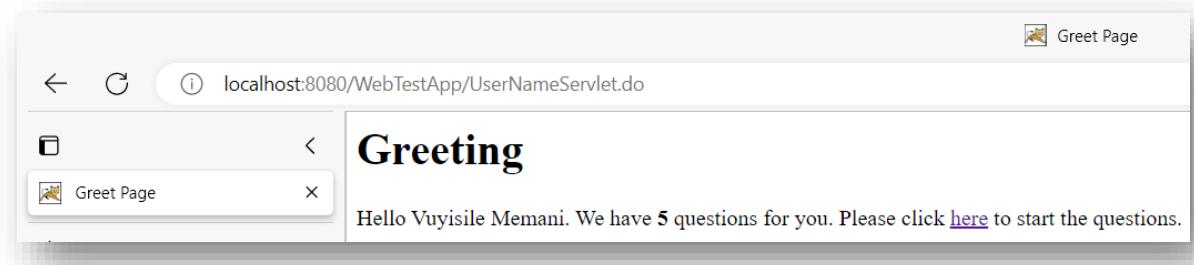
Click on the link.



Enter name.



Click on the **Submit** button.



Click on the link.

A screenshot of a web browser window titled "Question Page". The address bar shows "localhost:8080/WebTestApp/QuestionServlet.do". The main content area displays the heading "Question 1 of 5" and the instruction "Please answer the following question:". Below this, a question "Question: 1 * 3 = ?" is shown, followed by an input field for the answer and a "Submit" button.

Provide the answer.

A screenshot of a web browser window titled "Question Page". The address bar shows "localhost:8080/WebTestApp/QuestionServlet.do". The main content area displays the heading "Question 1 of 5" and the instruction "Please answer the following question:". Below this, a question "Question: 1 * 3 = ?" is shown, followed by an input field containing the answer "1" and a "Submit" button.

Click on the **Submit** button.

A screenshot of a web browser window titled "Outcome Page". The address bar shows "localhost:8080/WebTestApp/AnswerServlet.do". The main content area displays the heading "Outcome" and the message "Hi Vuyisile Memani, kindly find below the outcome:". Below this, it shows the question "Question: 1 * 3 = ?", the user's answer "Your answer: 1", the correct answer "Correct answer: 3", and the outcome "Outcome: You are wrong.". At the bottom, it says "Please click [here](#) to get the next question."

Click on the link.

A screenshot of a web browser window titled "Question Page". The address bar shows "localhost:8080/WebTestApp/QuestionServlet.do". The main content area displays the heading "Question 2 of 5" and the text "Please answer the following question:". Below this is a question "Question: 100 * 3 = ?" and an input field labeled "Answer:" with a "Submit" button below it. The browser's toolbar includes icons for back, forward, and search, along with tabs for "Question Page" and "New tab".

Provide the answer.

A screenshot of a web browser window titled "Question Page". The address bar shows "localhost:8080/WebTestApp/QuestionServlet.do". The main content area displays the heading "Question 2 of 5" and the text "Please answer the following question:". Below this is a question "Question: 100 * 3 = ?" and an input field labeled "Answer:" containing the value "300", with a "Submit" button below it. The browser's toolbar includes icons for back, forward, and search, along with tabs for "Question Page" and "New tab".

Click on the **Submit** button.

A screenshot of a web browser window titled "Outcome Page". The address bar shows "localhost:8080/WebTestApp/AnswerServlet.do". The main content area displays the heading "Outcome" and the text "Hi Vuyisile Memani, kindly find below the outcome:". Below this are several lines of text: "Question: 100 * 3 = ?", "Your answer: 300", "Correct answer: 300", and "Outcome: You are correct.". At the bottom, there is a link "Please click [here](#) to get the next question." The browser's toolbar includes icons for back, forward, and search, along with tabs for "Outcome Page" and "New tab".

Click on the link.

A screenshot of a web browser window titled "Question Page". The address bar shows "localhost:8080/WebTestApp/QuestionServlet.do". The main content area displays the heading "Question 3 of 5" and the text "Please answer the following question:". Below this is a question "Question: 10 / 2 = ?" and an input field with the value "Answer: 5". A "Submit" button is located below the input field.

Provide the answer.

A screenshot of a web browser window titled "Question Page". The address bar shows "localhost:8080/WebTestApp/QuestionServlet.do". The main content area displays the heading "Question 3 of 5" and the text "Please answer the following question:". Below this is a question "Question: 10 / 2 = ?" and an input field containing the value "5". A "Submit" button is located below the input field.

Click on the **Submit** button.

A screenshot of a web browser window titled "Outcome Page". The address bar shows "localhost:8080/WebTestApp/AnswerServlet.do". The main content area displays the heading "Outcome" and the text "Hi Vuyisile Memani, kindly find below the outcome:". Below this, it shows the question "Question: 10 / 2 = ?", the user's answer "Your answer: 5", the correct answer "Correct answer: 5", and the outcome "Outcome: You are correct.". At the bottom, it says "Please click [here](#) to get the next question."

Click on the link.

A screenshot of a web browser window titled "Question Page". The address bar shows "localhost:8080/WebTestApp/QuestionServlet.do". The main content area displays the heading "Question 4 of 5" and the text "Please answer the following question:". Below this is a question box containing "Question: 100 - 1 = ?" and an answer input field with the placeholder "Answer: []". A "Submit" button is located below the input field.

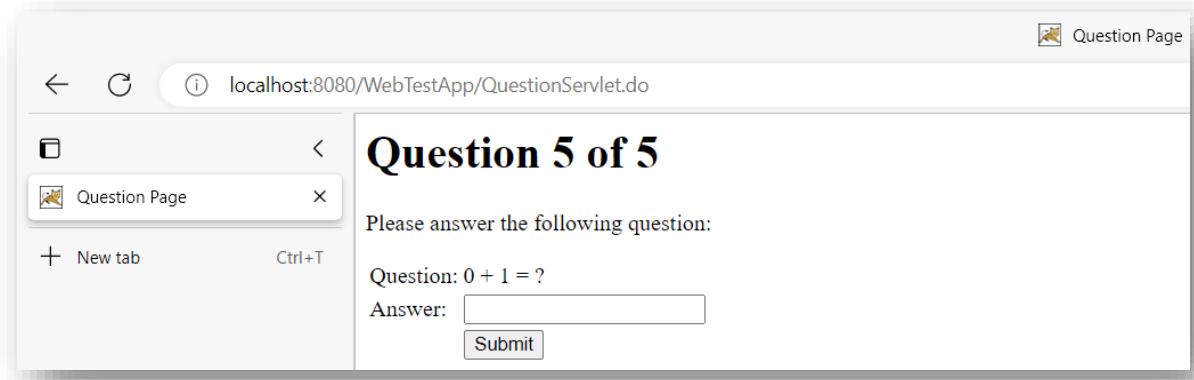
Provide the answer.

A screenshot of a web browser window titled "Question Page". The address bar shows "localhost:8080/WebTestApp/QuestionServlet.do". The main content area displays the heading "Question 4 of 5" and the text "Please answer the following question:". Below this is a question box containing "Question: 100 - 1 = ?" and an answer input field containing "99". A "Submit" button is located below the input field.

Click on the **Submit** button.

A screenshot of a web browser window titled "Outcome Page". The address bar shows "localhost:8080/WebTestApp/AnswerServlet.do". The main content area displays the heading "Outcome" and the text "Hi Vuvisile Memani, kindly find below the outcome:". Below this are several lines of text: "Question: 100 - 1 = ?", "Your answer: 99", "Correct answer: 99", and "Outcome: You are correct.". At the bottom, there is a message "Please click [here](#) to get the next question."

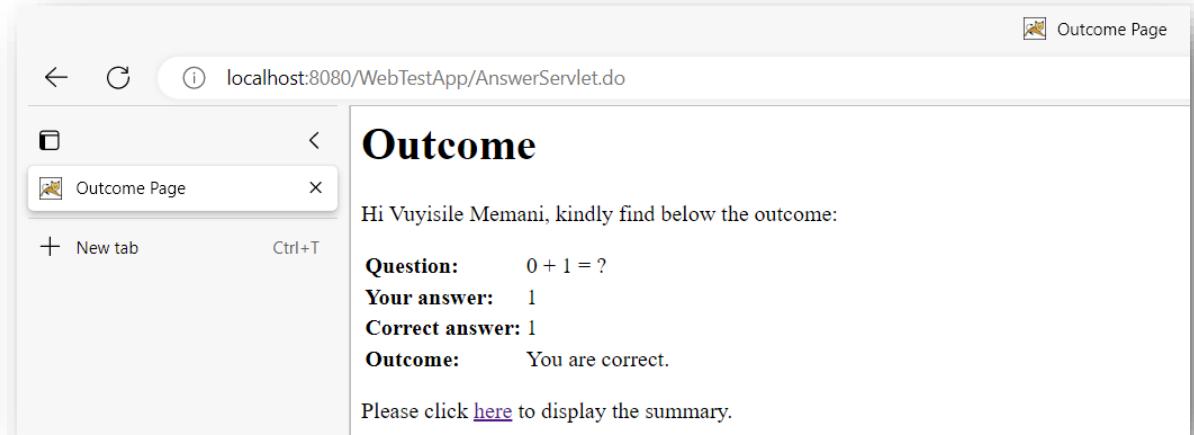
Click on the link.



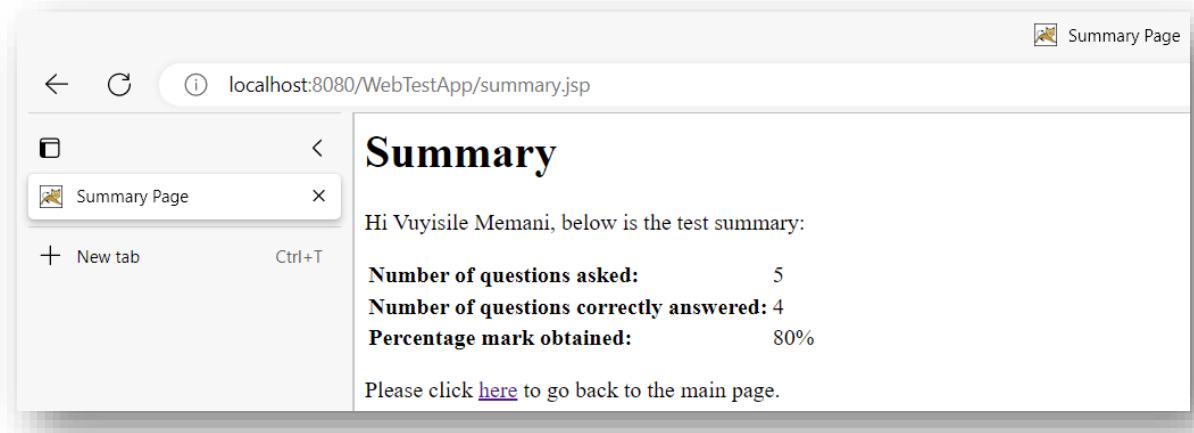
Provide the answer.



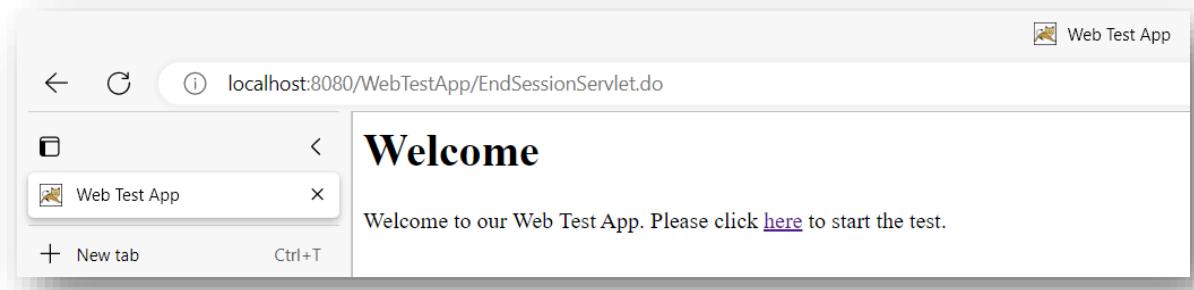
Click on the **Submit** button.



Click on the link.



Click on the link.



7. Stop Tomcat

Stop Tomcat by typing "**shutdown**" on the command line and then click the **Enter** key.

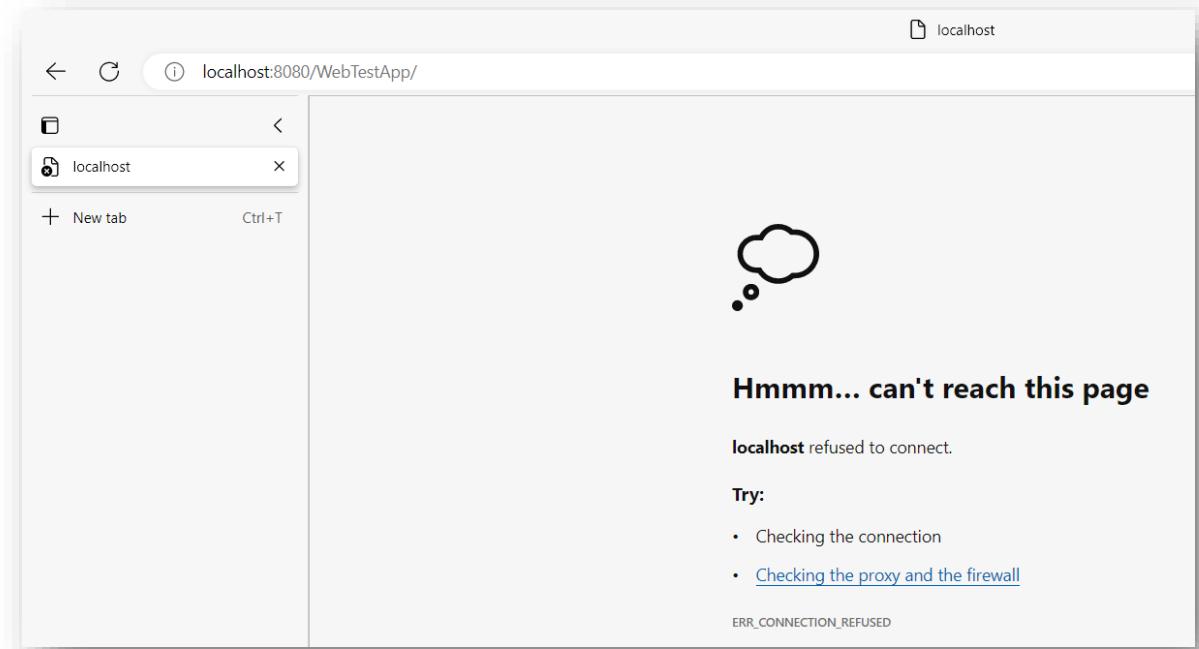
A screenshot of a Windows Command Prompt window titled "Command Prompt". The window shows the output of a Tomcat startup and shutdown process. It starts with "C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>startup" followed by several environment variable definitions. Then it shows "C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>shutdown", which triggers the shutdown of the Tomcat server. The command prompt ends with "C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>". An arrow points from the word "shutdown" in the shutdown command to the "shutdown" link in the "Welcome" message of the previous screenshot.

```
C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>startup
Using CATALINA_BASE:  "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_HOME:  "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_TMPDIR: "C:\Program Files\Apache Software Foundation\Tomcat 8.5\temp"
Using JRE_HOME:        "C:\Program Files\Java\jdk1.8.0_111"
Using CLASSPATH:       "C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\bootstrap.jar;C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\tomcat-juli.jar"
Using CATALINA_OPTS:   ""

C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>shutdown
Using CATALINA_BASE:  "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_HOME:  "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_TMPDIR: "C:\Program Files\Apache Software Foundation\Tomcat 8.5\temp"
Using JRE_HOME:        "C:\Program Files\Java\jdk1.8.0_111"
Using CLASSPATH:       "C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\bootstrap.jar;C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\tomcat-juli.jar"
Using CATALINA_OPTS:   ""

C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>
```

Try to run the web application whilst the server is down. An error takes place.



5.4 DIY (Do It Yourself)

In this chapter we introduced the student to conversational web apps.

Task #1

Tumelo wants a toss-guessing game that can be played by a human and a computer. When the game starts, the computer tosses the coin and ask the user to guess the toss. If the user guesses correctly, his/her score is incremented by one and takes over the toss. Otherwise, if the guess is incorrect, the score of the computer is incremented and it continues with the toss.

The application must keep track of the number of tosses. After each toss, appropriate messages must be displayed confirming the winner and loser. The user must be asked if she/he wants to proceed with the game. If so, the winner of the toss must do the tossing, otherwise the following game summary must be displayed:

- The number of tosses made.
- The number of tosses won by the computer.
- The number of tosses won by the user.

Tumelo requests that you create such a web application for him.

Task #2

Ms Mujinga is a teacher at institution ABC. She needs a web application that will allow her to set a TRUE/FALSE test of 10 questions. This type of test requires that when she sets it, she provides both the question and the answer. She wants her students to able to take a test.

The test questions must be randomly given to the students so that not two students sitting next to each other will get the same question. A student must be given one question at a time. After that the next question must be provided until all the questions have been answered. At the end of the test, a summary report must be displayed. The report must provide the following infomation:

- The question asked, the correct answer , the user's answer, and the mark obtained (1 or 0).
- The percentage mark obtained.

Create such a web application for Ms Mujinga.

Task #3

Mamazala is running a thriving business in Soshanguve, selling kotas/sphatlhos to customers. The table below shows the pricelist for the kotas.

Kota code	Description	Unit price
1	Atchar and cheese	R15-00
2	Atchar, cheese and chips	R20-00
3	Atchar, cheese, chips and polony	R25-00

She needs a web application that will help her run the business. The application must be capable of doing the following:

- Serve a customer by doing the following:
 - ✓ Display the kota menu shown in the above table.
 - ✓ Allow her to enter the kota code and quantity. The application must also allow for the user to add into the order other kotas.
 - ✓ Determine and display the amount due for the order.
 - ✓ Accept money tendered by the client and give appropriate change.
 - ✓ Update the total daily amount made.

- Serve the next customer.
- Display the following information when the shop gets closed:
 - ✓ The total number of kotas bought.
 - ✓ The number of kotas bought per code.
 - ✓ The total daily amount made.

Create such an application for Mamazala.

5.5 Conclusion

In this chapter we managed to explain the concept of conversational web apps. Thank you very much for having taken time to go through this chapter. Enjoy the rest of the day and God bless you.

6 Exceptions and exception handling

In this chapter we are going to discuss exceptions and their handling. We will start the discussion by explaining exceptions and exception handling in general terms. Subsequent to that we will talk about how to work with exceptions and their handling in a web development environment.

6.1 What is an exception?

An exception is an unexpected condition that can take place during programme execution. So this is a condition that a program was not designed to work with. For example you can have a program that adds numbers. If a user provides nonnumeric input instead, the program will malfunction. It is important to design our program in such a way that they are able to continue working even when the unexpected happens.

Exceptions are mainly divided into two, we have runtime exceptions and compile time exceptions. Runtime exceptions are exceptions that can only be detected during runtime, the time when a program is running. They are logical in nature and the programmer is expected to know how to handle them prior execution. Going back to the addition problem, the programmer needs to think ahead as to what must take place when a user enters a nonnumeric value for a number. This is called exception handling. Runtime exceptions are also known as unchecked exceptions. This means they are not checked at program compile time. So a programmer is not forced to handle them.

Compile time exceptions are exceptions that the compiler detects at compile time if they are not handled. The compiler forces the programmer to handle them before proceeding any further. These kinds of exceptions are also known as checked exceptions. Examples of compile time exceptions are all your file I/O (Input/Output) operations.

6.2 What is exception handling?

Exception handling is the mechanism used to handle exceptions when they occur. Through exception handling we avoid a situation whereby the behavior of a program becomes unpredictable due to the occurrence of an exception. So we manage the exception and bring the program back to normalcy. There are two ways of handling exceptions, namely:

- Usage of a try catch block; or
- Design a method that rethrows an exception.

6.2.1 Try catch block

The try catch block is used to handle an exception. Code that is likely to throw an exception is kept inside the try block, and code that must handle the exception is put inside the catch block. Below is the generic syntax of a try catch block.

```
1   try
2   {
3       //code that is likely to throw an exception
4       int a = ...;
5   }
6   catch(MyException e)
7   {
8       //code that must be executed for the occurred exception
9   }
10
11
```

When an exception takes place, an exception object is created. The object is then thrown into the catch blocks. We can have many catch blocks. Each catch block is responsible for handling one exception type. Then a catch block matching the thrown exception is sought. If one is found, the code inside that block is executed. If none is found, the program crashes or simply behaves in an unpredictable manner.

6.2.2 Exception rethrowing methods

Another way of handling an exception is through an exception rethrow. This is accomplished through a method that executes a piece of code that is likely to throw an exception. Rather than handling the exception inside the method, the method gets to rethrow the exception. This approach defers exception handling to other users of the method. The figure below shows exception rethrow through a method.

```
12
13  public void myMethod() throws MyException {
14      //code that is likely to throw an exception
15      int a = ....;
16
17 }
```

6.3 Predefined exceptions

The Java programming language comes along with predefined classes, some of which represent exceptions. These are the exceptions that the JDK creators encountered when developing the language. Some of the exception classes contained in the JDK is the `Exception`, `RuntimeException`, and `NumberFormatException` class.

6.3.1 Exception class

Below is a brief description of the class which is located in the `java.lang` package

```
public class Exception
extends Throwable
```

The class `Exception` and its subclasses are a form of `Throwable` that indicates conditions that a reasonable application might want to catch.

The class `Exception` and any subclasses that are not also subclasses of `RuntimeException` are *checked exceptions*. Checked exceptions need to be declared in a method or constructor's `throws` clause if they can be thrown by the execution of the method or constructor and propagate outside the method or constructor boundary.

Since:
JDK1.0

See Also:
`Error`, `Serialized Form`

See The Java™ Language Specification:
11.2 Compile-Time Checking of Exceptions

Constructor Summary	
Constructors	Modifier
<code>Exception()</code>	Constructs a new exception with <code>null</code> as its detail message.
<code>Exception(String message)</code>	Constructs a new exception with the specified detail message.

6.3.2 NumberFormatException class

Below is a brief description of the class which is located in the java.lang package

```
public class NumberFormatException  
extends IllegalArgumentException
```

Thrown to indicate that the application has attempted to convert a string to one of the numeric types, but that the string does not have the appropriate format.

Since:

JDK1.0

See Also:

`Integer.parseInt(String)`, Serialized Form

Constructor Summary

Constructors

Constructor and Description

`NumberFormatException()`

Constructs a `NumberFormatException` with no detail message.

`NumberFormatException(String s)`

Constructs a `NumberFormatException` with the specified detail message.

6.3.3 RuntimeException

Below is a brief description of the class which is located in the java.lang package

```
public class RuntimeException  
extends Exception
```

`RuntimeException` is the superclass of those exceptions that can be thrown during the normal operation of the Java Virtual Machine.

`RuntimeException` and its subclasses are *unchecked exceptions*. Unchecked exceptions do *not* need to be declared in a method or constructor's `throws` clause if they can be thrown by the execution of the method or constructor and propagate outside the method or constructor boundary.

Since:

JDK1.0

See Also:

Serialized Form

See *The Java™ Language Specification*:

11.2 Compile-Time Checking of Exceptions

Constructor Summary

Constructors

Modifier	Constructor and Description
	<code>RuntimeException()</code> Constructs a new runtime exception with null as its detail message.
	<code>RuntimeException(String message)</code> Constructs a new runtime exception with the specified detail message.

6.4 User-defined exceptions

As a programmer you can also define your own exception classes. These classes will simply be there to represent a specific condition that is not catered for by the predefined exception classes. The handling of the exception will work the same as that of predefined exceptions. The generic syntax for defining a user-defined exception is as follows:

```
17  
18     public class MyException extends RuntimeException {  
19         public MyExceptionClass(String errorMessage) {  
20             super(errorMessage);  
21         }  
22     }  
23 }
```

6.5 How to handle exceptions in a web development environment?

In the web development environment we use a combination of the DD and corresponding files to handle exceptions. In the DD we state the exceptions we want to handle, and also the files that must be called to handle those exceptions. In the DD, an exception can be referenced either by its fully qualified name or code. For example, if we want to catch the NumberFormatException we will have the following in the DD:

```
25 <error-page>  
26     <exception-type>java.lang.NumberFormatException</exception-type>  
27     <location>/numberFormatErrorResponse.jsp</location>  
28 </error-page>  
29  
30 
```

The **<error-page>** element is used to represent exception handling. Inside the **<error-page>** element we define the type of error we want to handle, **<exception-type>**, and the file that will handle the error, **<location>**.

The exception code is used if we know the code that represents an error. For example, if we were to work with the famous 404 FileNotFoundException, we could represent it as follows in the DD:

```
30
31 <error-page>
32   <error-code>404</error-code>
33   <location>/fileNotFoundErrorPage.jsp</location>
34 </error-page>
35
```

Again we define the error handling file inside and the **location** element.

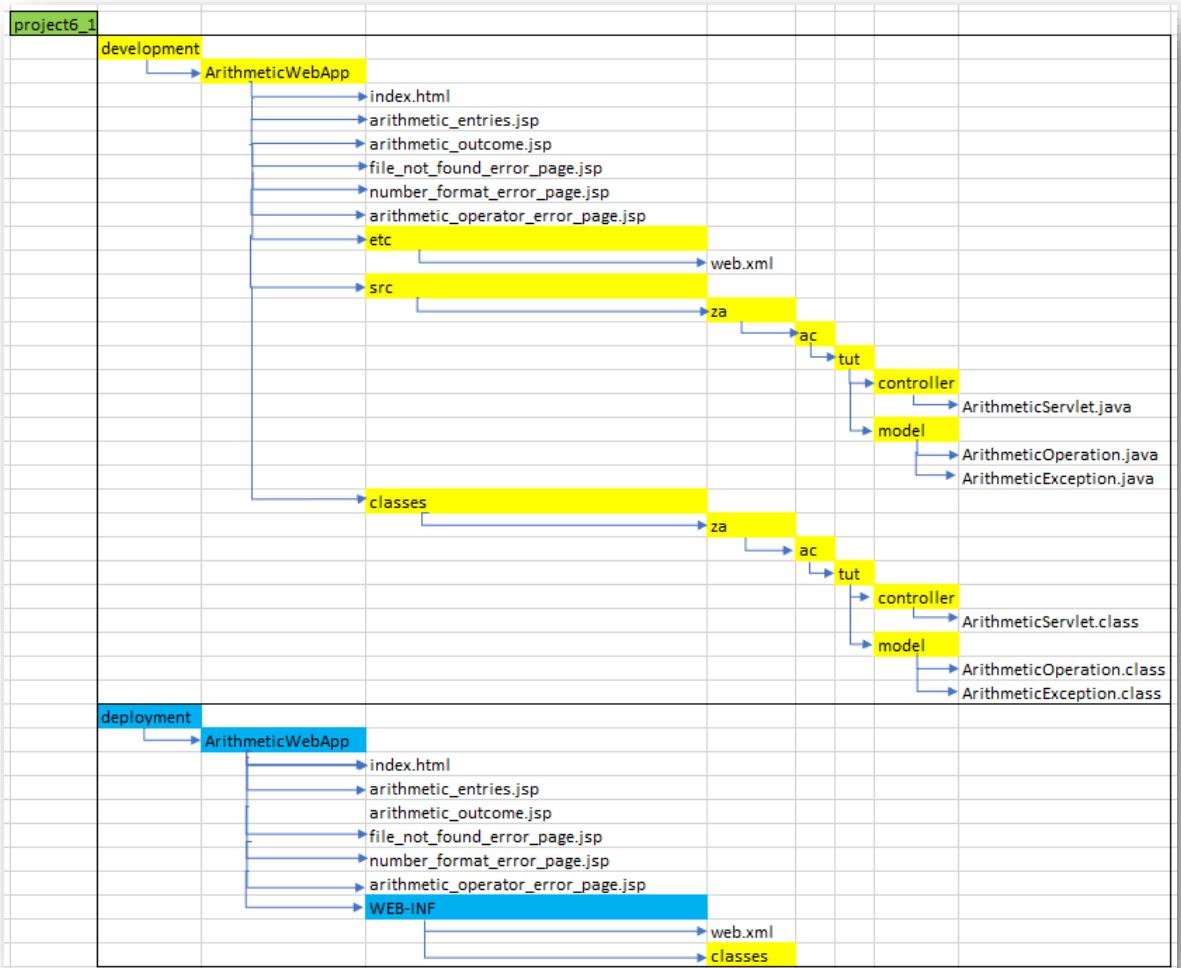
6.6 Example

Create a robust web application that will allow a user to perform a preferred arithmetic operation on two numbers. After the user has entered the two numbers and the preferred operation, the application must perform the operation and return the outcome. If the user enters an invalid input (letter instead of number, not any of the recognised operators or no input at all), an appropriate error message must be displayed.

Solution

1. Project structure.

Below is the project structure which acts as the solution to the problem we are solving.



Description of the project's structure:

We can notice the following from the above figure:

- The name of the project is **project6_1**. The **6** stands for chapter **6** and the **1** stands for **example number 1**. So this is the first example of chapter 6.
 - The project has two main folders, **development** and **deployment**.
 - In the development directory we keep the code that is under construction. After we are done, we then transfer the code to the deployment folder. The code in the deployment folder is code that is ready to be deployed in the Web Server, that is Tomcat.
 - The contents of the development folder are copied to the deployment folder. There are two major changes in the deployment folder, **etc** folder is changed

to **WEB-INF**, and in this folder we keep the **DD** file, the **classes** folder together with its contents from the development folder.

Description of the project's files:

- The name of the web application is **ArithmeticWebApp**.
- The web application has three types of files, namely:

html

- The **index.html** file is the landing page. It has a link to the **arithmetic_entries.jsp** file.

jsp

- The **arithmetic_entries.jsp** file allows a user to enter two numbers and the operation to be performed.
- The **arithmetic_outcome.jsp** displays the outcome of the requested operation.
- The **arithmetic_operator_error_page.jsp** handles an exception that occurs when an invalid operator is entered.
- The **file_not_found_error_page.jsp** handles an exception that occurs when a requested file is not found.
- The **number_format_error_page.jsp** file handles an exception that occurs when a nonnumeric value is entered.

servlet

- The **ArithmeticServlet.java** gets arithmetic data from the **arithmetic.jsp** file. The servlet performs the requested operation. The outcome is forwarded to the **arithmetic_outcome.jsp** file.

models

- The **ArithmeticOperation.java** class defines arithmetic methods for determining sum, product, quotient and difference. This class is instantiated and used inside the servlet class.
- The **ArithmetiException.java** class is thrown when any of the arithmetic entries is invalid. This is caused either by a nonnumeric input value or an operator other than '+', '*', '/' or '-'.

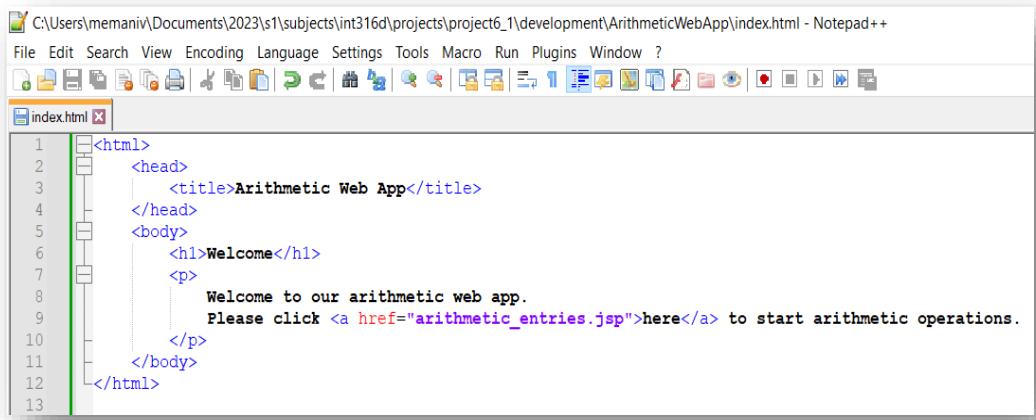
xml

- The **web.xml** file is the configuration file of the web application. It defines the **welcome file**, **servlet**, **servlet initialisation parameters**, and **servlet context parameters**.

2. Create the files.

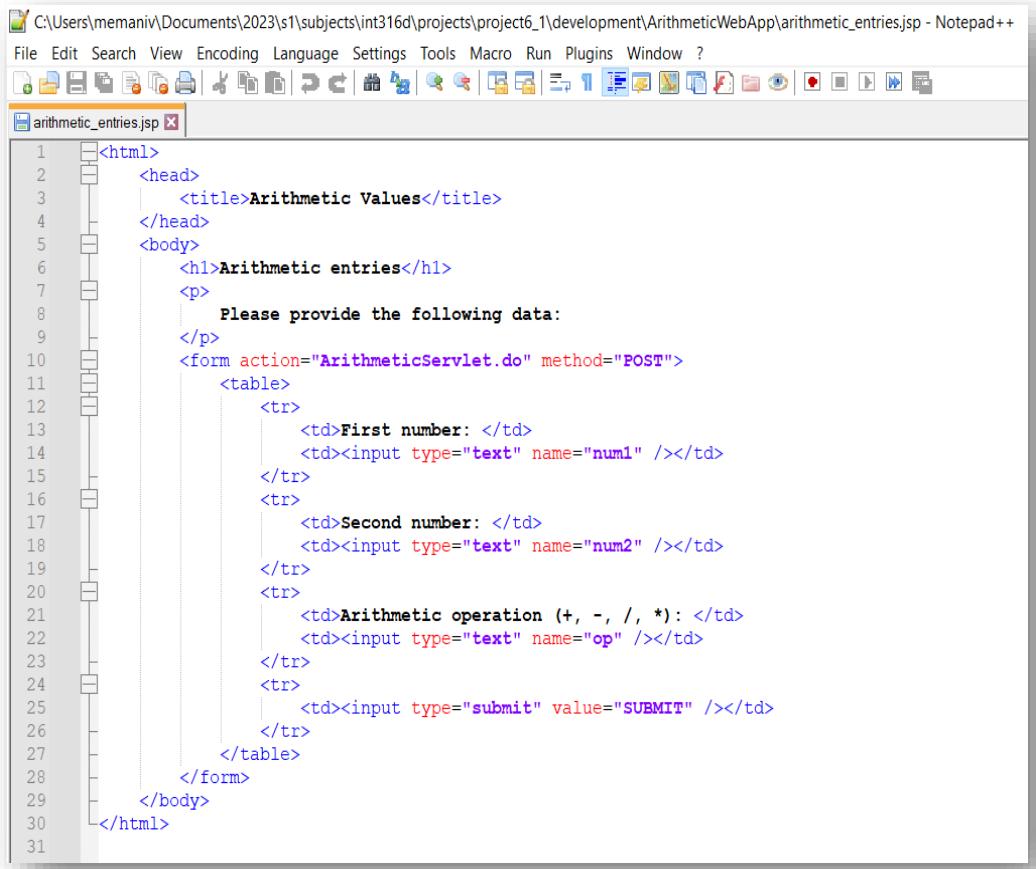
Create the files as follows:

- **html**
 - **index.html**



```
C:\Users\memaniv\Documents\2023\s1\subjects\int316d\projects\project6_1\development\ArithmeticWebApp\index.html - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
index.html [x]
1 <html>
2   <head>
3     <title>Arithmetic Web App</title>
4   </head>
5   <body>
6     <h1>Welcome</h1>
7     <p>
8       Welcome to our arithmetic web app.
9       Please click <a href="arithmetic_entries.jsp">here</a> to start arithmetic operations.
10    </p>
11  </body>
12</html>
13
```

- **jsp**
 - **arithmetic_entries.jsp**



```
C:\Users\memaniv\Documents\2023\s1\subjects\int316d\projects\project6_1\development\ArithmeticWebApp\arithmetic_entries.jsp - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
arithmetic_entries.jsp [x]
1 <html>
2   <head>
3     <title>Arithmetic Values</title>
4   </head>
5   <body>
6     <h1>Arithmetic entries</h1>
7     <p>
8       Please provide the following data:
9     </p>
10    <form action="ArithmeticServlet.do" method="POST">
11      <table>
12        <tr>
13          <td>First number: </td>
14          <td><input type="text" name="num1" /></td>
15        </tr>
16        <tr>
17          <td>Second number: </td>
18          <td><input type="text" name="num2" /></td>
19        </tr>
20        <tr>
21          <td>Arithmetic operation (+, -, /, *): </td>
22          <td><input type="text" name="op" /></td>
23        </tr>
24        <tr>
25          <td><input type="submit" value="SUBMIT" /></td>
26        </tr>
27      </table>
28    </form>
29  </body>
30</html>
31
```

- **arithmetic_outcome.jsp**

```
C:\Users\memani\n\Documents\2023\s1\subjects\int316d\projects\project6_1\development\ArithmeticWebApp\arithmetic_outcome.jsp - Notepad++  
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?  
arithmetic_outcome.jsp  
1 <html>  
2   <head>  
3     <title>Arithmetic Outcome Page</title>  
4   </head>  
5   <body>  
6     <h1>Arithmetic outcome</h1>  
7     <%>  
8       Integer num1 = (Integer)request.getAttribute("num1");  
9       Integer num2 = (Integer)request.getAttribute("num2");  
10      Character op = (Character)request.getAttribute("op");  
11      String outcome = (String)request.getAttribute("outcome");  
12    <%>  
13    <p>  
14      <%=num1%> <%=op%> <%=num2%> = <%=outcome%>.  
15      Please click <a href="index.html">here</a> to go back to the main page.  
16    </p>  
17  </body>  
18</html>
```

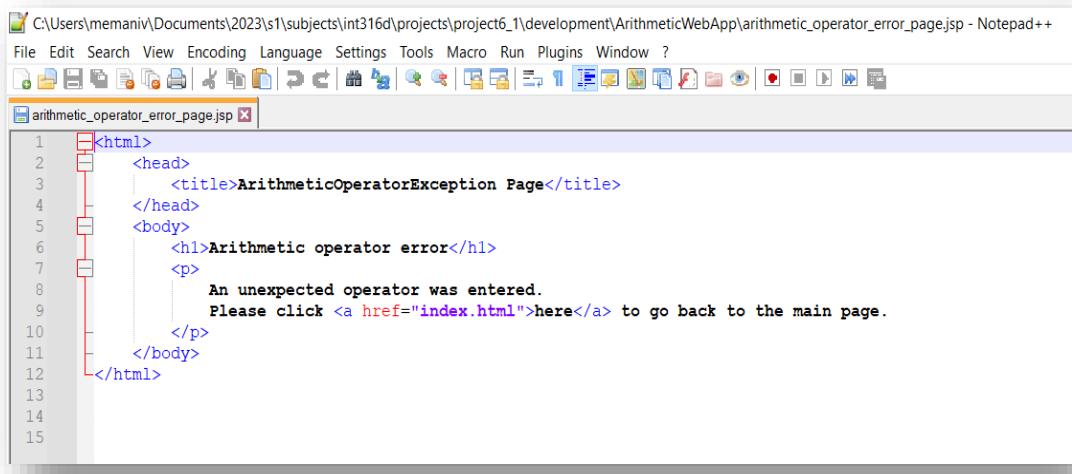
- **file_not_found_error_page.jsp**

```
C:\Users\memani\n\Documents\2023\s1\subjects\int316d\projects\project6_1\development\ArithmeticWebApp\file_not_found_error_page.jsp - Notepad++  
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?  
file_not_found_error_page.jsp  
1 <html>  
2   <head>  
3     <title>FileNotFoundException Page</title>  
4   </head>  
5   <body>  
6     <h1>File not found error</h1>  
7     <p>  
8       The requested file was not found.  
9       Please click <a href="index.html">here</a> to go back to the main page.  
10    </p>  
11  </body>  
12</html>
```

- **number_format_error_page.jsp**

```
C:\Users\memani\n\Documents\2023\s1\subjects\int316d\projects\project6_1\development\ArithmeticWebApp\number_format_error_page.jsp - Notepad++  
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?  
number_format_error_page.jsp  
1 <html>  
2   <head>  
3     <title>NumberFormatException Page</title>  
4   </head>  
5   <body>  
6     <h1>Number format error</h1>  
7     <p>  
8       An unexpected non-numeric number was entered.  
9       Please click <a href="index.html">here</a> to go back to the main page.  
10    </p>  
11  </body>  
12</html>
```

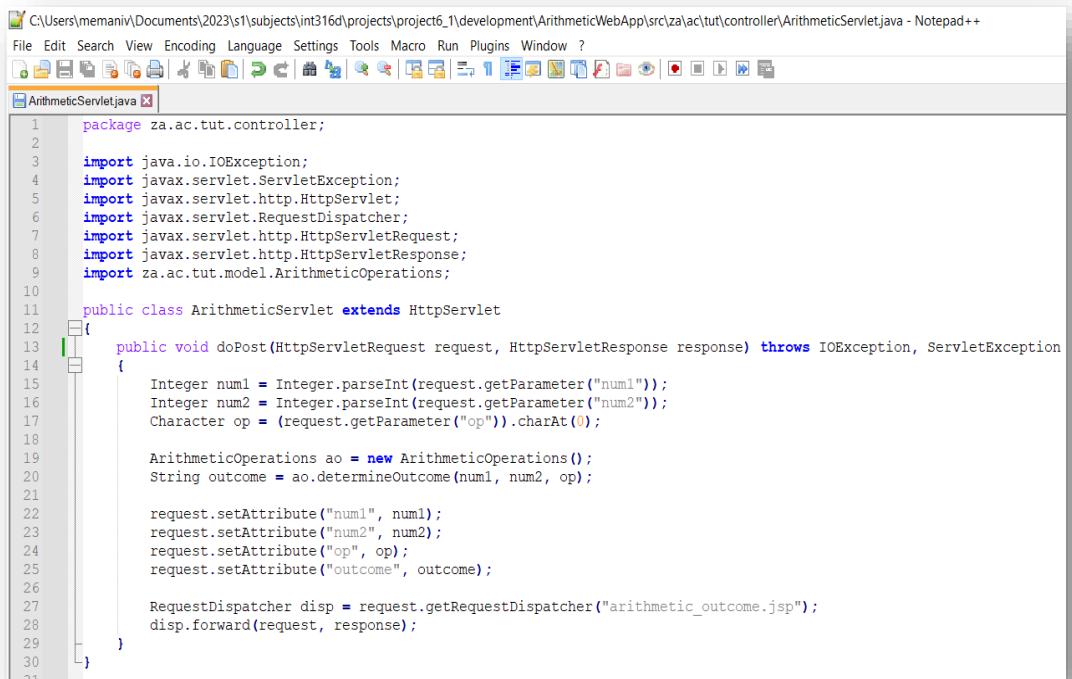
- **arithmetic_operator_error_page.jsp**



```
C:\Users\memaniv\Documents\2023\s1\subjects\int316d\projects\project6_1\development\ArithmeticWebApp\arithmetic_operator_error_page.jsp - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
arithmetic_operator_error_page.jsp
1 <html>
2   <head>
3     <title>ArithmeticOperatorException Page</title>
4   </head>
5   <body>
6     <h1>Arithmetic operator error</h1>
7     <p>
8       An unexpected operator was entered.
9       Please click <a href="index.html">here</a> to go back to the main page.
10    </p>
11  </body>
12</html>
13
14
15
```

- **servlet**

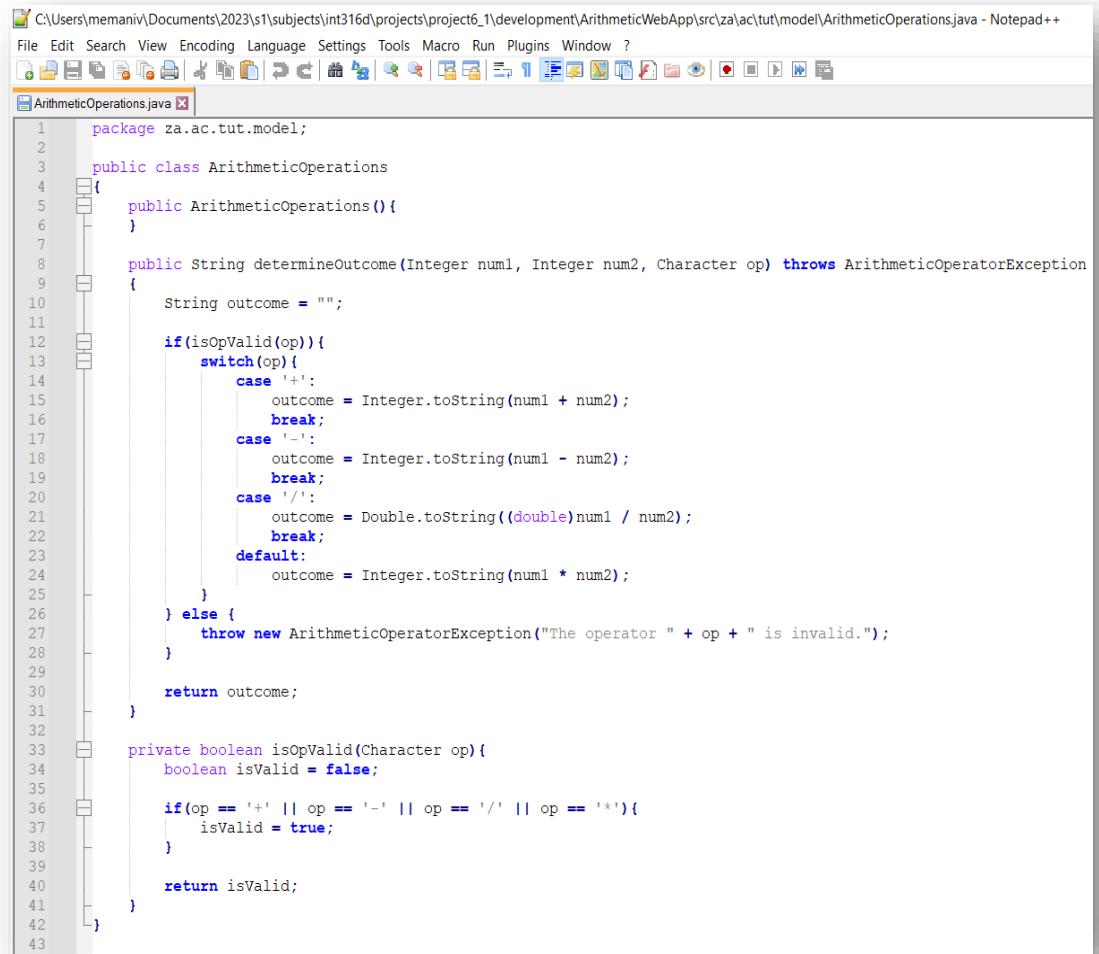
- **ArithmeticServlet.java**



```
C:\Users\memaniv\Documents\2023\s1\subjects\int316d\projects\project6_1\development\ArithmeticWebApp\src\za\ac\tut\controller\ArithmeticServlet.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
ArithmeticServlet.java
1 package za.ac.tut.controller;
2
3 import java.io.IOException;
4 import javax.servlet.ServletException;
5 import javax.servlet.http.HttpServlet;
6 import javax.servlet.http.HttpServletRequest;
7 import javax.servlet.http.HttpServletResponse;
8 import javax.servlet.http.HttpSession;
9 import za.ac.tut.model.ArithmeticOperations;
10
11 public class ArithmeticServlet extends HttpServlet
12 {
13   public void doPost(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException
14   {
15     Integer num1 = Integer.parseInt(request.getParameter("num1"));
16     Integer num2 = Integer.parseInt(request.getParameter("num2"));
17     Character op = (request.getParameter("op")).charAt(0);
18
19     ArithmeticOperations ao = new ArithmeticOperations();
20     String outcome = ao.determineOutcome(num1, num2, op);
21
22     request.setAttribute("num1", num1);
23     request.setAttribute("num2", num2);
24     request.setAttribute("op", op);
25     request.setAttribute("outcome", outcome);
26
27     RequestDispatcher disp = request.getRequestDispatcher("arithmetic_outcome.jsp");
28     disp.forward(request, response);
29   }
30 }
31
```

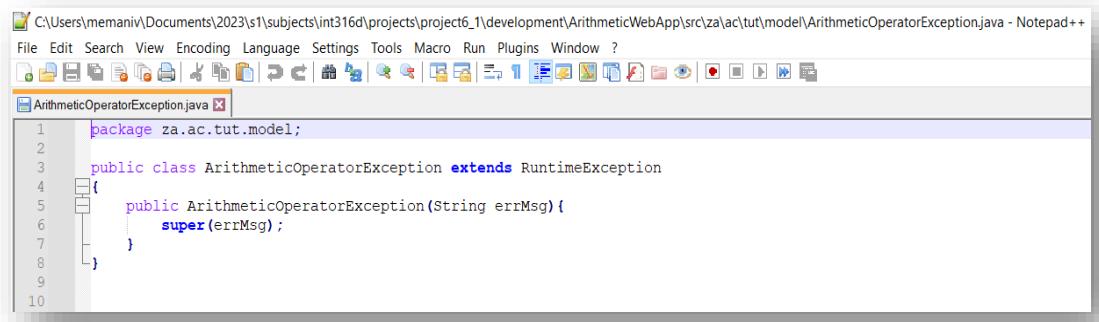
- **models**

- **ArithmeticOperations.java**



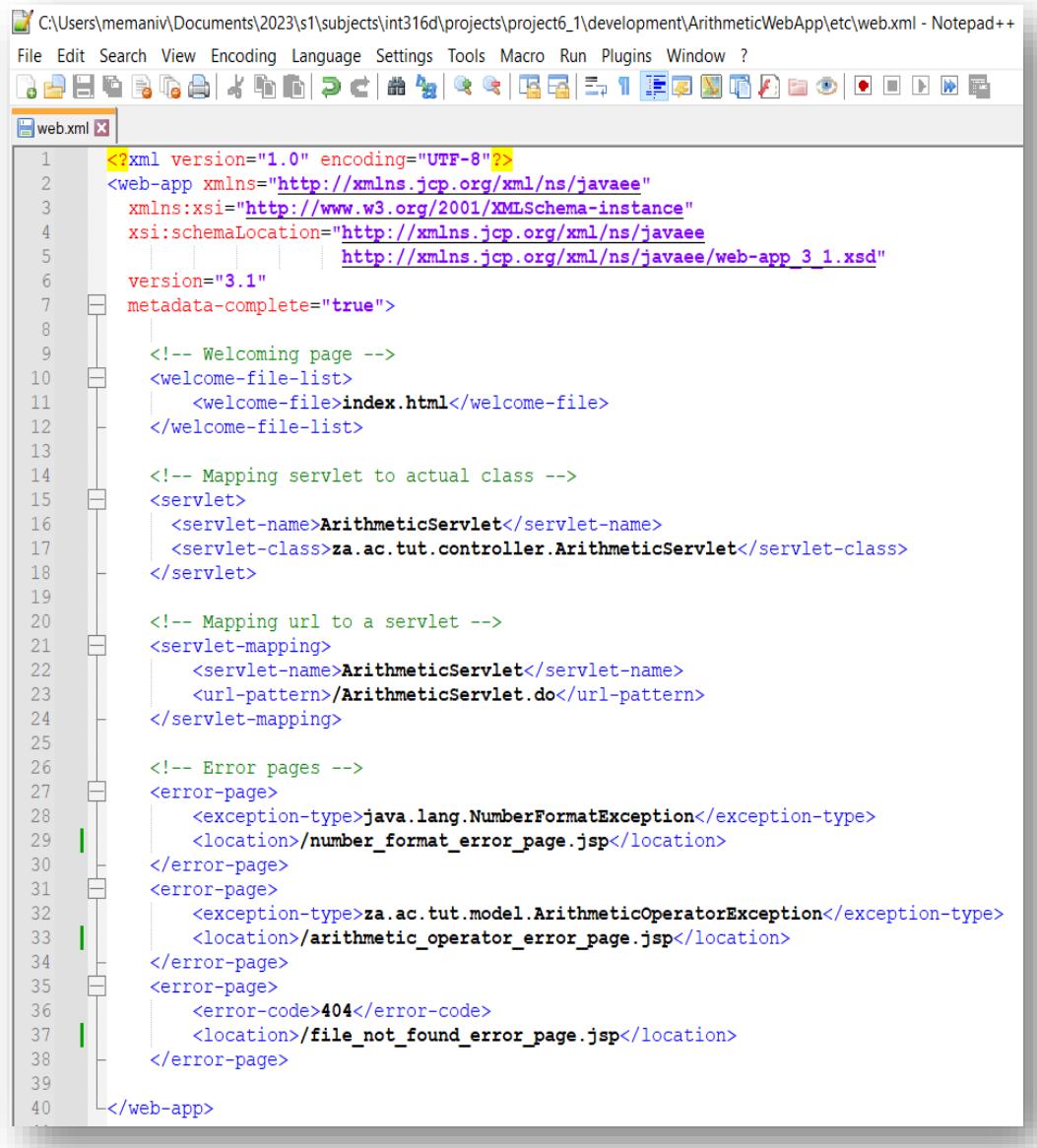
```
1 package za.ac.tut.model;
2
3 public class ArithmeticOperations
4 {
5     public ArithmeticOperations() {
6     }
7
8     public String determineOutcome(Integer num1, Integer num2, Character op) throws ArithmeticOperatorException {
9         String outcome = "";
10
11         if(isOpValid(op)){
12             switch(op){
13                 case '+':
14                     outcome = Integer.toString(num1 + num2);
15                     break;
16                 case '-':
17                     outcome = Integer.toString(num1 - num2);
18                     break;
19                 case '/':
20                     outcome = Double.toString((double)num1 / num2);
21                     break;
22                 default:
23                     outcome = Integer.toString(num1 * num2);
24             }
25         } else {
26             throw new ArithmeticOperatorException("The operator " + op + " is invalid.");
27         }
28
29         return outcome;
30     }
31
32     private boolean isOpValid(Character op) {
33         boolean isValid = false;
34
35         if(op == '+' || op == '-' || op == '/' || op == '*'){
36             isValid = true;
37         }
38
39         return isValid;
40     }
41 }
42
43
```

- **ArithmeticOperatorsException.java**



```
1 package za.ac.tut.model;
2
3 public class ArithmeticOperatorException extends RuntimeException
4 {
5     public ArithmeticOperatorException(String errMsg){
6         super(errMsg);
7     }
8 }
9
10
```

- **xml**
 - **web.xml**



The screenshot shows the Notepad++ application window with the file `web.xml` open. The window title is `C:\Users\memani\n Documents\2023\s1\subjects\int316d\projects\project6_1\development\ArithmeticWebApp\etc\web.xml - Notepad++`. The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Tools, Macro, Run, Plugins, Window, and Help. The toolbar contains various icons for file operations like Open, Save, Find, Copy, Paste, etc. The code editor displays the XML configuration for a Java web application:

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
5          http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
6      version="3.1"
7      metadata-complete="true">
8
9      <!-- Welcoming page -->
10     <welcome-file-list>
11         <welcome-file>index.html</welcome-file>
12     </welcome-file-list>
13
14     <!-- Mapping servlet to actual class -->
15     <servlet>
16         <servlet-name>ArithmeticServlet</servlet-name>
17         <servlet-class>za.ac.tut.controller.ArithmeticServlet</servlet-class>
18     </servlet>
19
20     <!-- Mapping url to a servlet -->
21     <servlet-mapping>
22         <servlet-name>ArithmeticServlet</servlet-name>
23         <url-pattern>/ArithmeticServlet.do</url-pattern>
24     </servlet-mapping>
25
26     <!-- Error pages -->
27     <error-page>
28         <exception-type>java.lang.NumberFormatException</exception-type>
29         <location>/number_format_error_page.jsp</location>
30     </error-page>
31     <error-page>
32         <exception-type>za.ac.tut.model.ArithmeticOperatorException</exception-type>
33         <location>/arithmetic_operator_error_page.jsp</location>
34     </error-page>
35     <error-page>
36         <error-code>404</error-code>
37         <location>/file_not_found_error_page.jsp</location>
38     </error-page>
39
40 </web-app>

```

3. Prepare for deployment

Copy the **ArithmeticWebApp** folder and paste on the **deployment** folder.

project6_1 > deployment		
Name	Date modified	Type
ArithmeticWebApp	2023/01/25 22:09	File folder

Change **etc** to **WEB-INF** and ensure that the **classes** folder and the **web.xml** file are located inside this folder.

project6_1 > deployment > ArithmeticWebApp			
Name	Date modified	Type	Size
WEB-INF	2023/01/25 22:08	File folder	
arithmetic_entries	2023/01/25 20:41	JSP File	1 KB
arithmetic_operator_error_page	2023/01/25 21:29	JSP File	1 KB
arithmetic_outcome	2023/01/25 20:48	JSP File	1 KB
file_not_found_error_page	2023/01/25 20:54	JSP File	1 KB
index	2023/01/25 20:40	Microsoft Edge HT...	1 KB
number_format_error_page	2023/01/25 21:29	JSP File	1 KB

project6_1 > deployment > ArithmeticWebApp > WEB-INF			
Name	Date modified	Type	Size
classes	2023/01/25 22:08	File folder	
web	2023/01/25 22:04	XML Document	2 KB

4. Deploy web application

Copy the **ArithmeticWebApp** folder deployment and paste it under the webapps folder of Tomcat.

Tomcat 8.5 > webapps		
Name	Date modified	Type
ArithmeticWebApp	2023/01/25 22:09	File folder
docs	2022/12/12 10:11	File folder
examples	2022/12/12 10:11	File folder
host-manager	2022/12/12 10:11	File folder
manager	2022/12/12 10:11	File folder
ROOT	2022/12/12 10:11	File folder

Get to the **bin** directory of Tomcat.

Tomcat 8.5 > bin			
Name	Date modified	Type	Size
bootstrap	2022/11/16 15:34	Executable Jar File	36 KB
catalina	2022/11/16 15:34	Windows Batch File	17 KB
ciphers	2022/11/16 15:34	Windows Batch File	3 KB
configtest	2022/11/16 15:34	Windows Batch File	2 KB
digest	2022/11/16 15:34	Windows Batch File	3 KB
service	2022/11/16 15:34	Windows Batch File	9 KB
setclasspath	2022/11/16 15:34	Windows Batch File	4 KB
shutdown	2022/11/16 15:34	Windows Batch File	2 KB
startup	2022/11/16 15:34	Windows Batch File	2 KB
Tomcat8	2022/11/16 15:34	Application	142 KB
Tomcat8w	2022/11/16 15:34	Application	126 KB
tomcat-juli	2022/11/16 15:34	Executable Jar File	52 KB
tool-wrapper	2022/11/16 15:34	Windows Batch File	5 KB
version	2022/11/16 15:34	Windows Batch File	2 KB

Copy the address of the bin directory

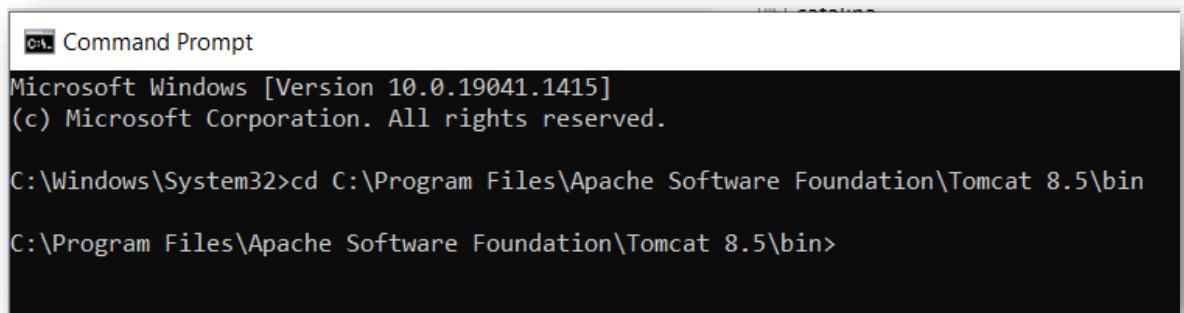
C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin

Launch the command line.

```
Command Prompt
Microsoft Windows [Version 10.0.19041.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>
```

Change to the **bin** directory of Tomcat.

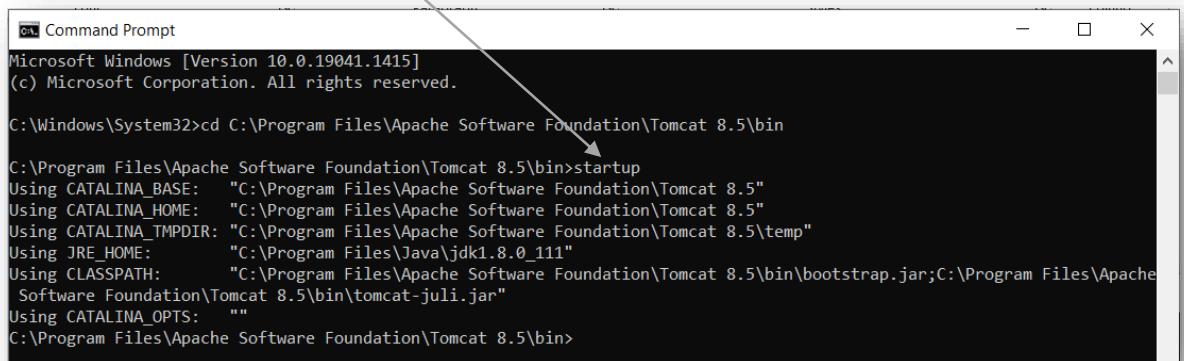


```
Command Prompt
Microsoft Windows [Version 10.0.19041.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin

C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>
```

Start Tomcat by typing "**startup**" on the command line and then click the **Enter** key.



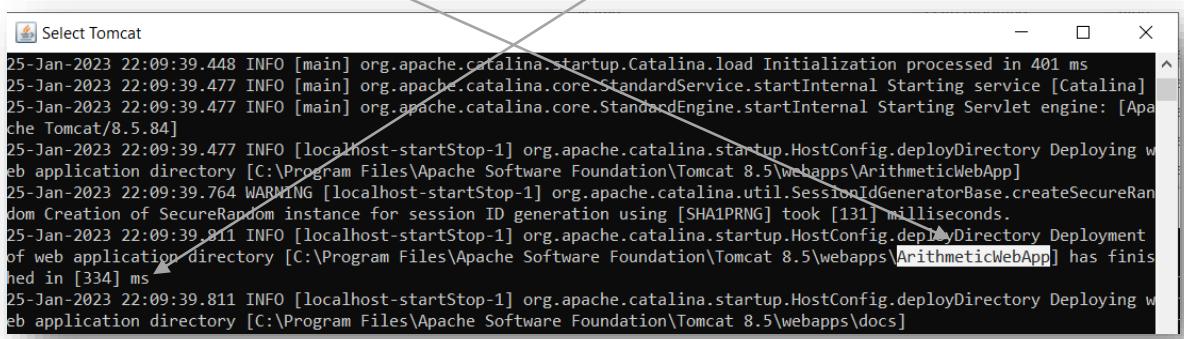
```
Command Prompt
Microsoft Windows [Version 10.0.19041.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin

C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>startup
Using CATALINA_BASE: "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_HOME: "C:\Program Files\Apache Software Foundation\Tomcat 8.5"
Using CATALINA_TMPDIR: "C:\Program Files\Apache Software Foundation\Tomcat 8.5\temp"
Using JRE_HOME: "C:\Program Files\Java\jdk1.8.0_111"
Using CLASSPATH: "C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\bootstrap.jar;C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\tomcat-juli.jar"
Using CATALINA_OPTS: ""
C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>
```

Output:

The **ArithmeticWebApp** application is deployed in **334 ms**.



```
Select Tomcat
25-Jan-2023 22:09:39.448 INFO [main] org.apache.catalina.startup.Catalina.load Initialization processed in 401 ms
25-Jan-2023 22:09:39.477 INFO [main] org.apache.catalina.core.StandardService.startInternal Starting service [Catalina]
25-Jan-2023 22:09:39.477 INFO [main] org.apache.catalina.core.StandardEngine.startInternal Starting Servlet engine: [Apache Tomcat/8.5.84]
25-Jan-2023 22:09:39.477 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory [C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\ArithmeticWebApp]
25-Jan-2023 22:09:39.764 WARNING [localhost-startStop-1] org.apache.catalina.util.SessionIdGeneratorBase.createSecureRandom Creation of SecureRandom instance for session ID generation using [SHA1PRNG] took [131] milliseconds.
25-Jan-2023 22:09:39.911 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application directory [C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\ArithmeticWebApp] has finished in [334] ms
25-Jan-2023 22:09:39.811 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory [C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\docs]
```

5. Run the web application

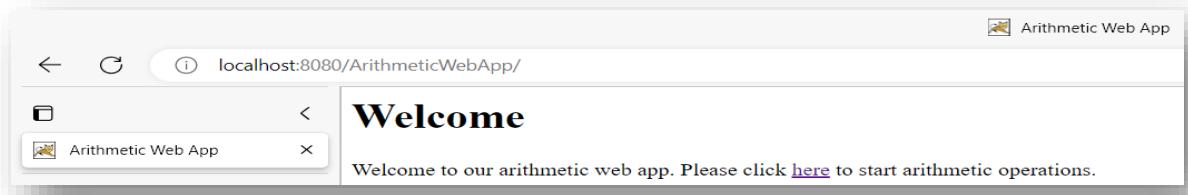
Open the browser and type:

`http://localhost:8080/ArithmeticWebApp/`

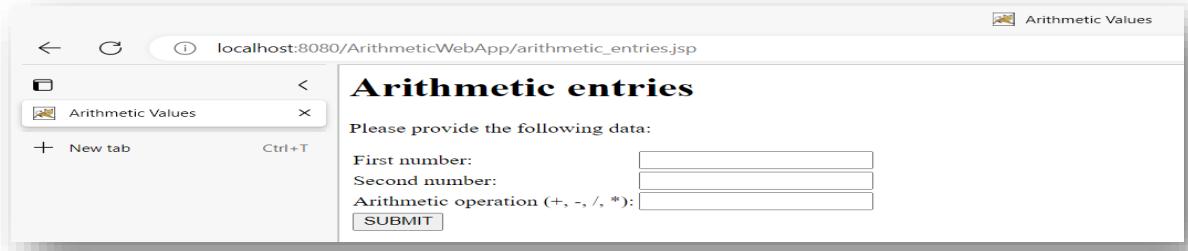
Note:

- **http:** is the protocol used to communicate between the browser and the Web Server, Tomcat.
- **localhost:** is the address of the machine that hosts the Web Server. In this instance the Web Server is running on the local machine.
- **8080:** is the port where the Web Server is running.
- **ArithmeticWebApp:** the name of the web application deployed on the server that we want to communicate with.

Output:



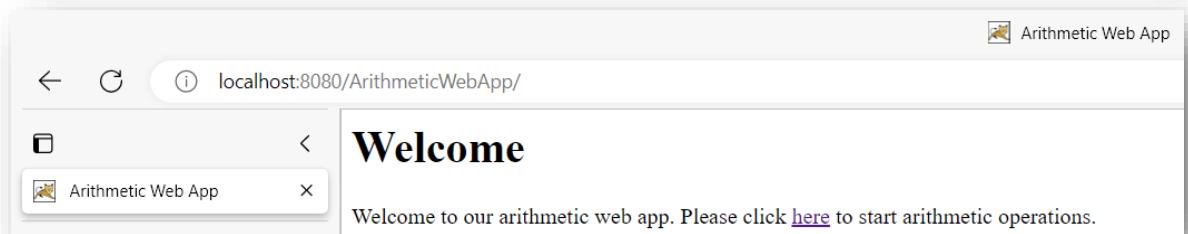
Click on the link.



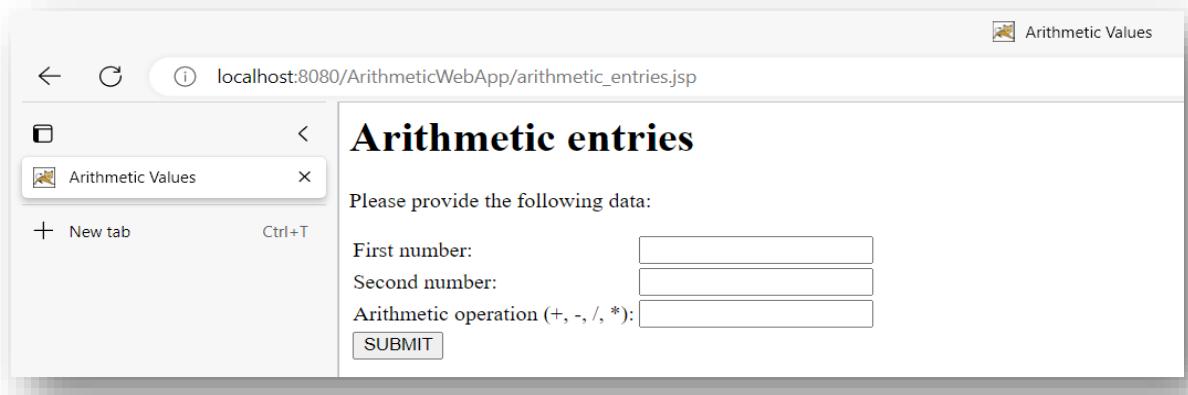
Click on the **SUBMIT** button. An exception is thrown for providing nonnumeric data and subsequently handled through a file.



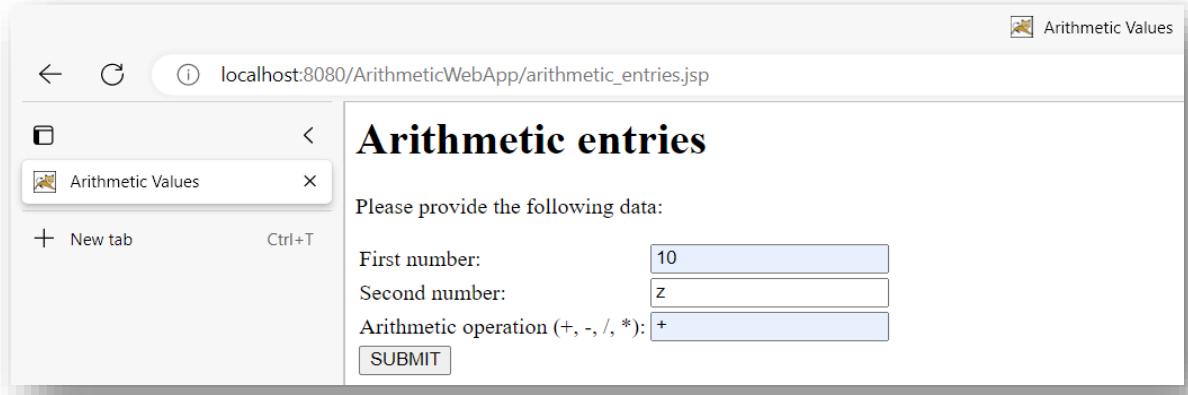
Click on the link.



Click on the link.



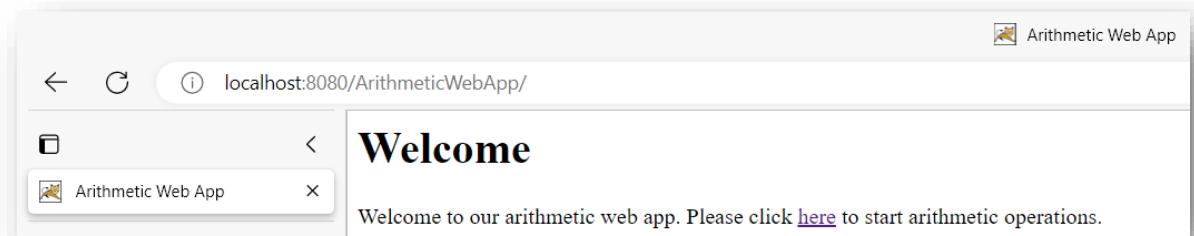
Enter a **number**, a **letter** and a **valid operator**.



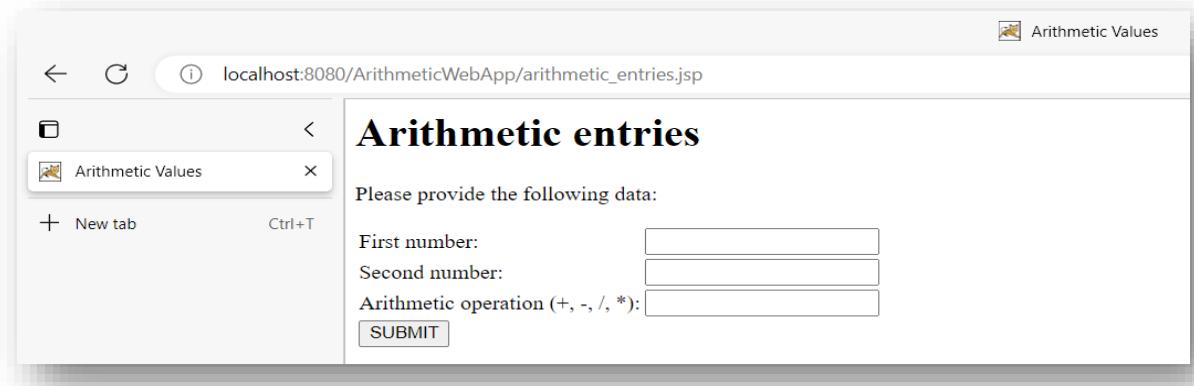
Click on the **SUBMIT** button. An exception is thrown for providing nonnumeric data and subsequently handled through a file.



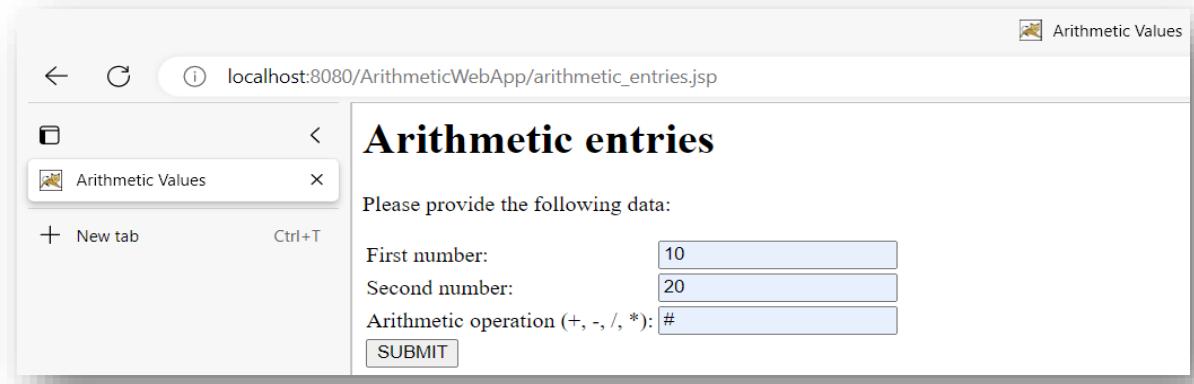
Click on the link.



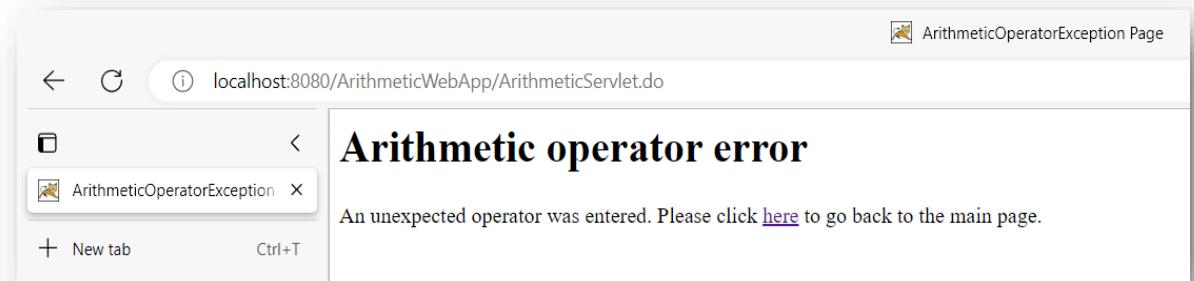
Click on the link.



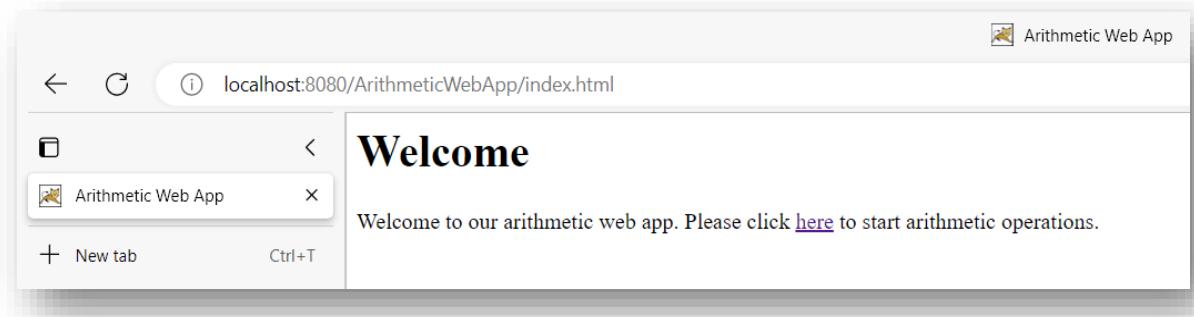
Enter **valid numbers** and an invalid operator.



Click on the **SUBMIT** button. An exception is thrown for providing an invalid operator and subsequently handled through a file.



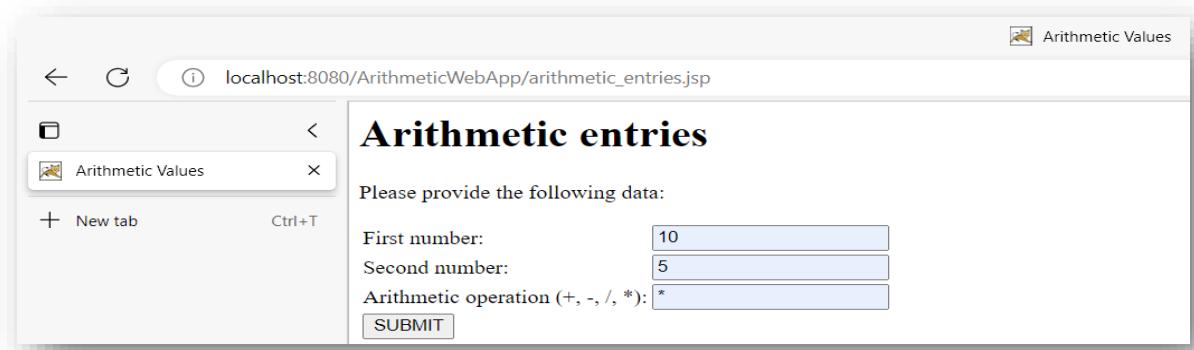
Click on the link.



Click on the link.



Enter **valid numbers** and an valid operator.



Click on the **SUBMIT** button. The expected output is displayed.



6.7 DIY (Do It Yourself)

In this DIY we want you to develop robust web applications through exception handling.

Task #1

Find out about what constitutes a valid South African ID. After that create a web application that can be used to check the validity of an ID given an ID number.

Task #2

Relebogile is an intern student at **CodersThink.Inc** company. The company specialises in developing web applications for companies. As his first task at the company, Relebogile is given the responsibility of creating a web application that will help users come up with strong passwords. A strong password must meet the following requirements:

- The size must be at least 10.
- Consist of letters, special characters and numbers. At least two of the letters must be uppercase, and another two lowercase.
- The @ and # special characters must not be part of the password.

Assuming that you are Relebogile, create such a web application for **CodersThink.Inc**.

Task #3

Mulumba is a newly appointed software developer at an academic institution XYZ. The institution has a set selection criteria for first year intake into their Computer Science programme. Below is the selection criteria.

Selection criteria:

To be considered for this qualification, applicants must have an Admission Point Score (APS) of at least **26** (with Mathematics or Technical Mathematics) or **28** (with Mathematical Literacy). Applicants with a score of **23** (with Mathematics or Technical Mathematics) or **25** (with Mathematical Literacy) will be considered for the extended programme. Life Orientation is excluded for APS calculation. Candidates who successfully completed the National Diploma: Information Technology (Extended) might also be considered.

The institution wants a web application that prospective students can use to calculate their APS (Admission Point Score) before applying. This will save students time and money so that they don't apply if they don't meet the minimum requirements.

The table below shows the relationship between APS and the grades obtained by students at matric (NSC).

APS	NSC	Range	Description
7	7	80 – 100%	Outstanding competent
6	6	70 – 79%	Highly competent
5	5	60 – 69%	Very competent
4	4	50 – 59%	Competent
3	3	40 – 49%	Not yet competent
2	2	30 – 39%	Not achieved
1	1	0 – 29%	

So the web application must allow a prospective student to enter his/her NSC grades, together with the respective subject names. The application must then calculate the APS and tell whether the student qualifies or not. The application must make sure that the NSC grades are in the specified ranges. Also Life Orientation information must not be accepted.

Assuming that you are Mulumba, create such a web application for institution XYZ.

6.8 Conclusion

In this chapter we managed to introduce the student to exceptions and their handling. In the next chapter we discuss the concept of EJBs, the business component of JEE.

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

7 Enterprise Java Beans (EJBs)

In this chapter, we introduce the student to the concept of EJBs, that is your Enterprise Java Beans. We also demonstrate the implementation of EJBs in a web development environment.

7.1 What are EJBs?

To successfully define an EJB we need to go back to the definition we provided for JEE. We said “*JEE is a set or collection of APIs (class libraries) used to develop business applications.*” This set of APIs is divided into parts, Web APIs and EJB APIs. Containers/servers that implement or support the Web APIs are called Web Servers and an example is Tomcat. Some of the Web APIs are JSP and JSF. Containers/servers that implement or support both APIs (Web and EJBs) are called Application Servers and an example is GlassFish.

Now, what is an EJB? An EJB is a server-side component/standard/specification/API that is used to implement business logic of enterprise applications. As a server-side component, the lifecycle of an EJB is under the control of the server. Its start of life and end of life is determined by the container. The container is responsible for instantiating an EJB when needed, and also destroying it when no longer needed.

As a standard, the EJB is defined in a JSR (Java Specification Request) document known as JSR-220. You can download the entire JSR-220 from the following location:

[The Java Community Process\(SM\) Program - JSRs: Java Specification Requests - detail JSR# 220 \(jcp.org\)](https://jcp.org/en/jsr/detail/JSR_220)

There are two types of EJBs, namely:

- Session Beans (SB); and
- Message-Driven Beans.

7.2 Session Beans

Session Beans are implementors of business logic. There are three kinds of Session Beans, namely:

- Stateless;
- Stateful; and
- Singleton.

7.2.1 Stateless Session Bean

Stateless Session Beans do not maintain the conversational state between method calls of the client and the bean. This means that, 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 so. An example could be determining the validity of an Identity Document.

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

How to create a Stateless Session Bean?

A Stateless Session Bean is a plain Java class 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

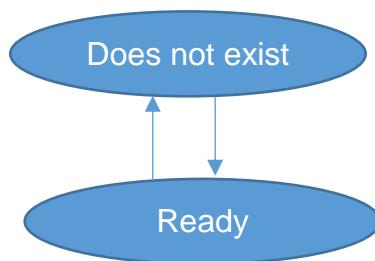
```
1  
2  
3  @Stateless  
4  public class PropertyManagersSB implements PropertyManagerInterface{  
5      public String determineLowestIncomeInfo(String areaCode){  
6          //code  
7      }  
8  
9      public String determineHighestIncomeInfo(String monthCode){  
10         //code  
11     }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

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

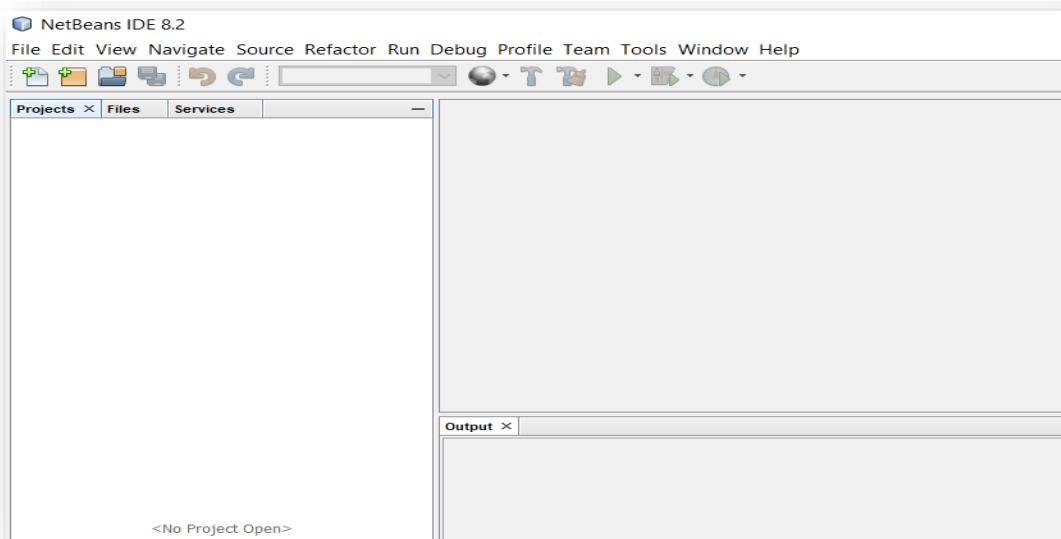
The solution to this problem is going to be done in three sequential parts. In **Part A** we will create an **EJB module** project. This project will have a stateless session bean that implements two methods for converting between the currencies. A jar file of the EJB module will be created.

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.

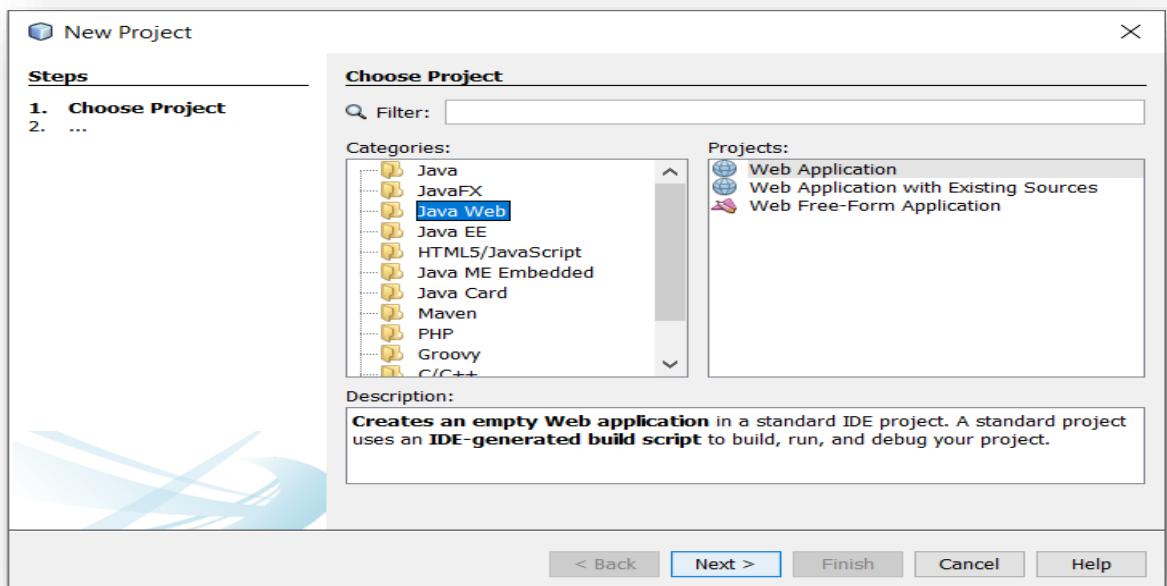
Part A - Create an EJB module project.

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

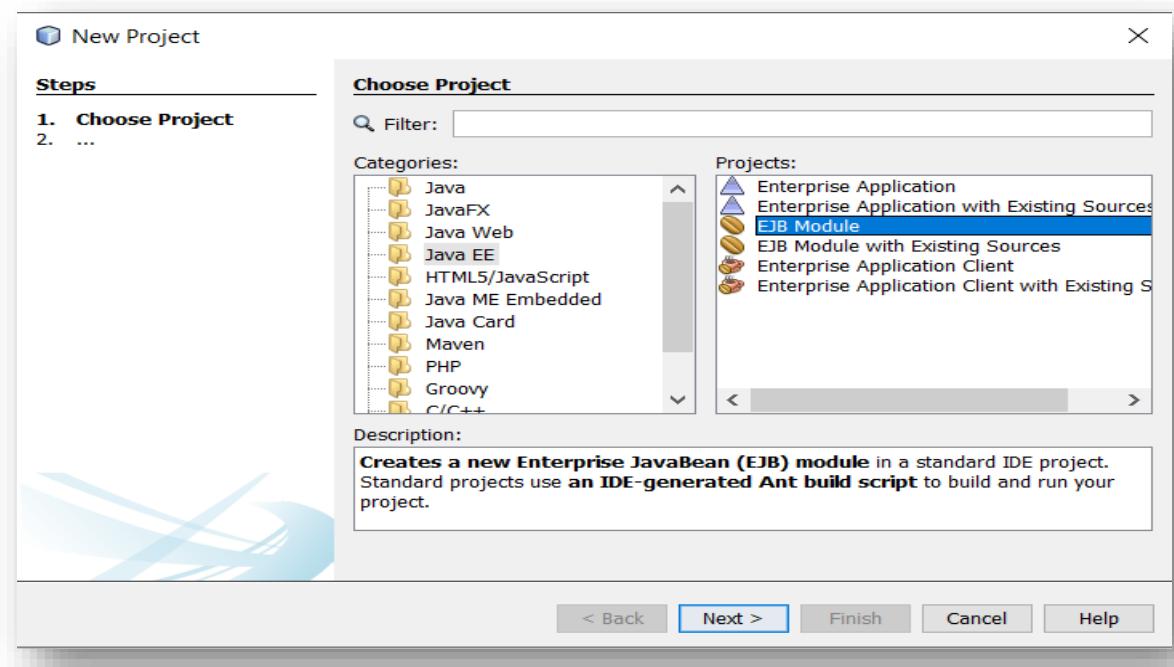
1. Launch NetBeans.



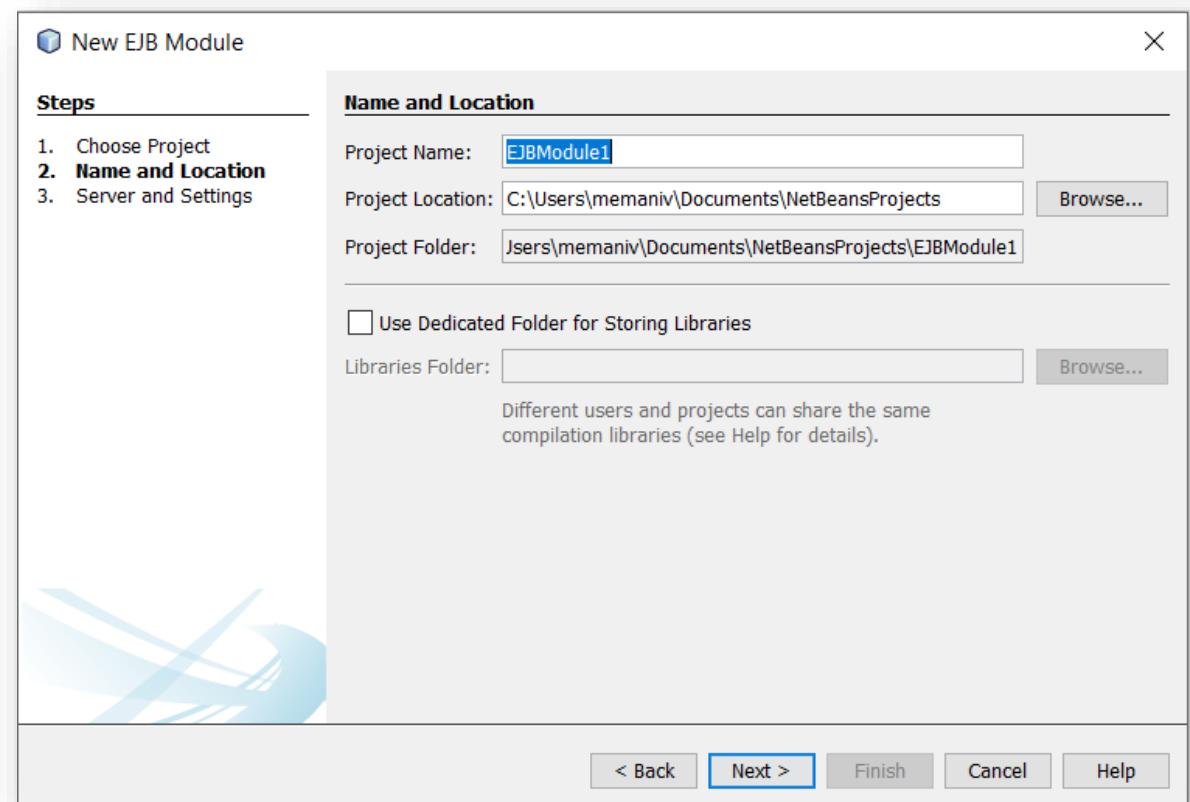
2. Click on **File | New Project**.



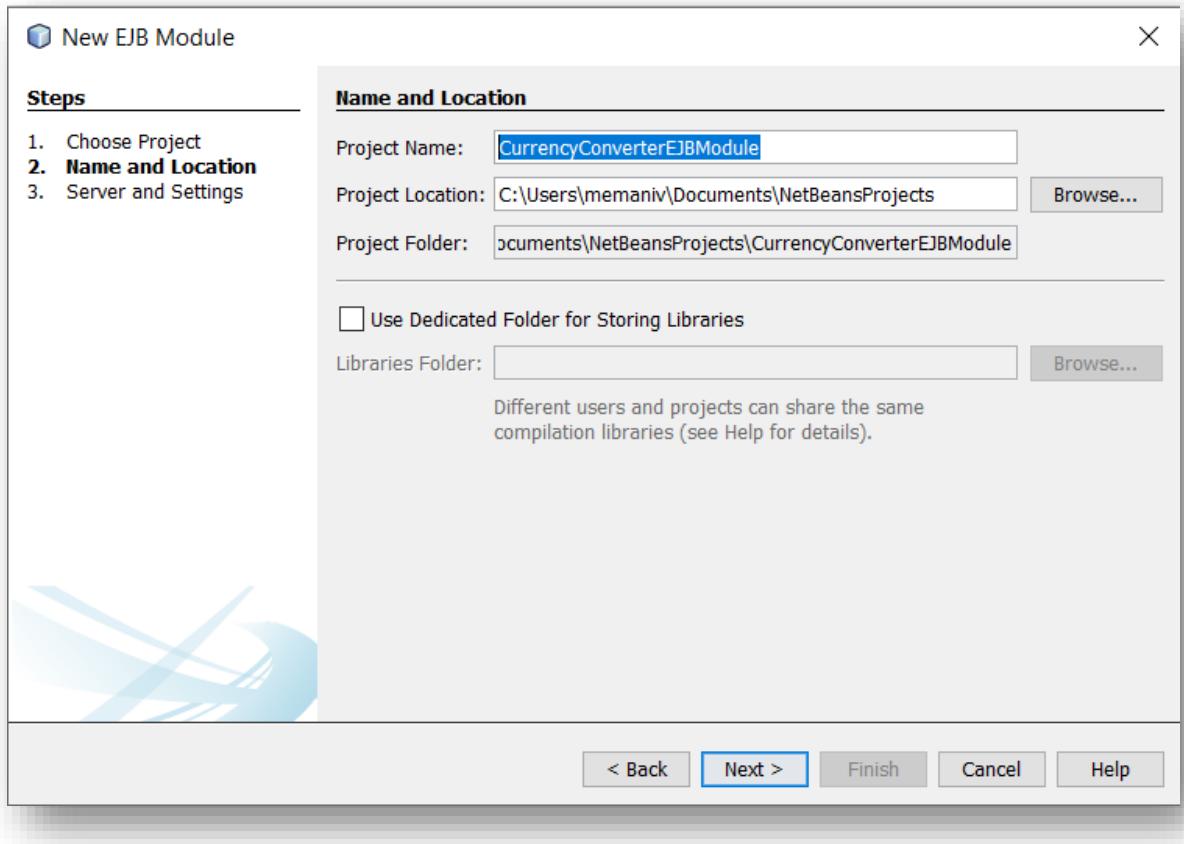
3. Select Java EE under Categories and EJB Module under Projects.



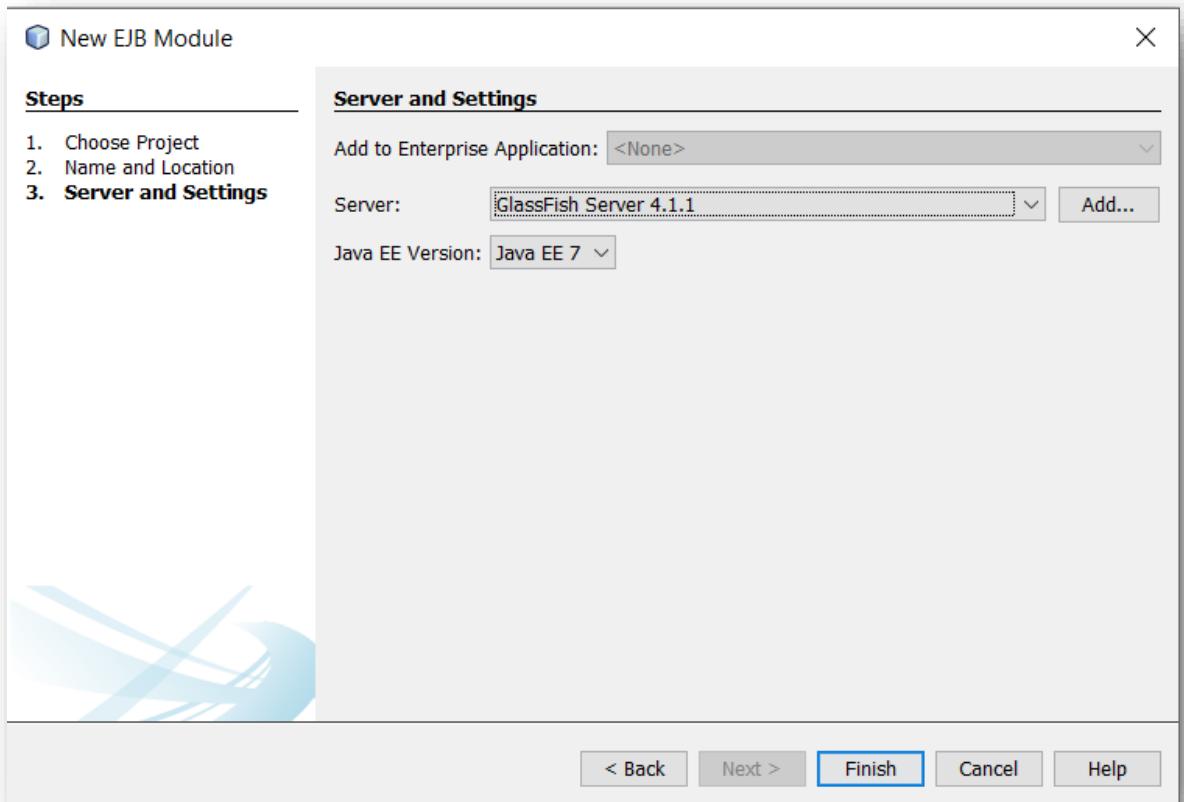
4. Click Next.



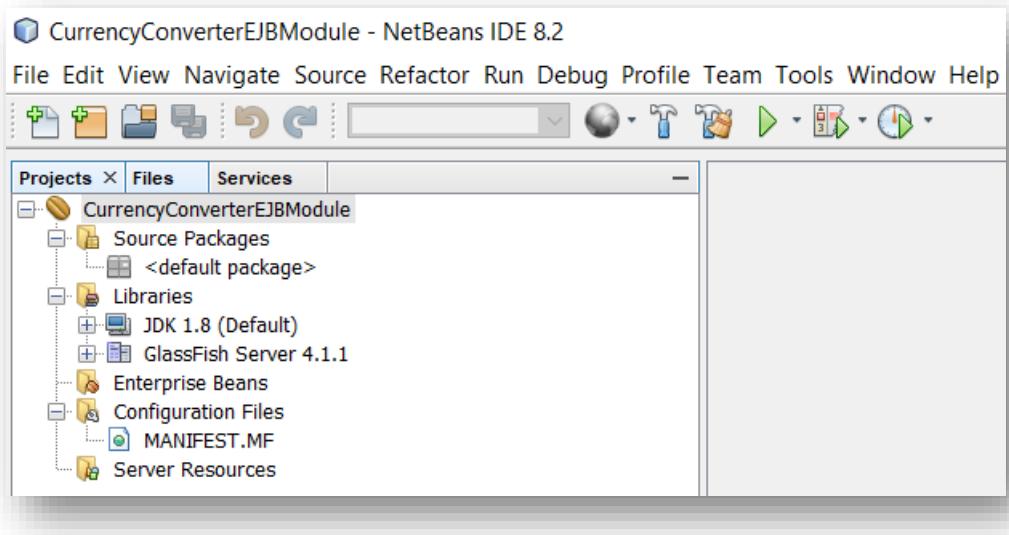
5. Name the project as **CurrencyConverterEJBModule**.



6. Click **Next**.

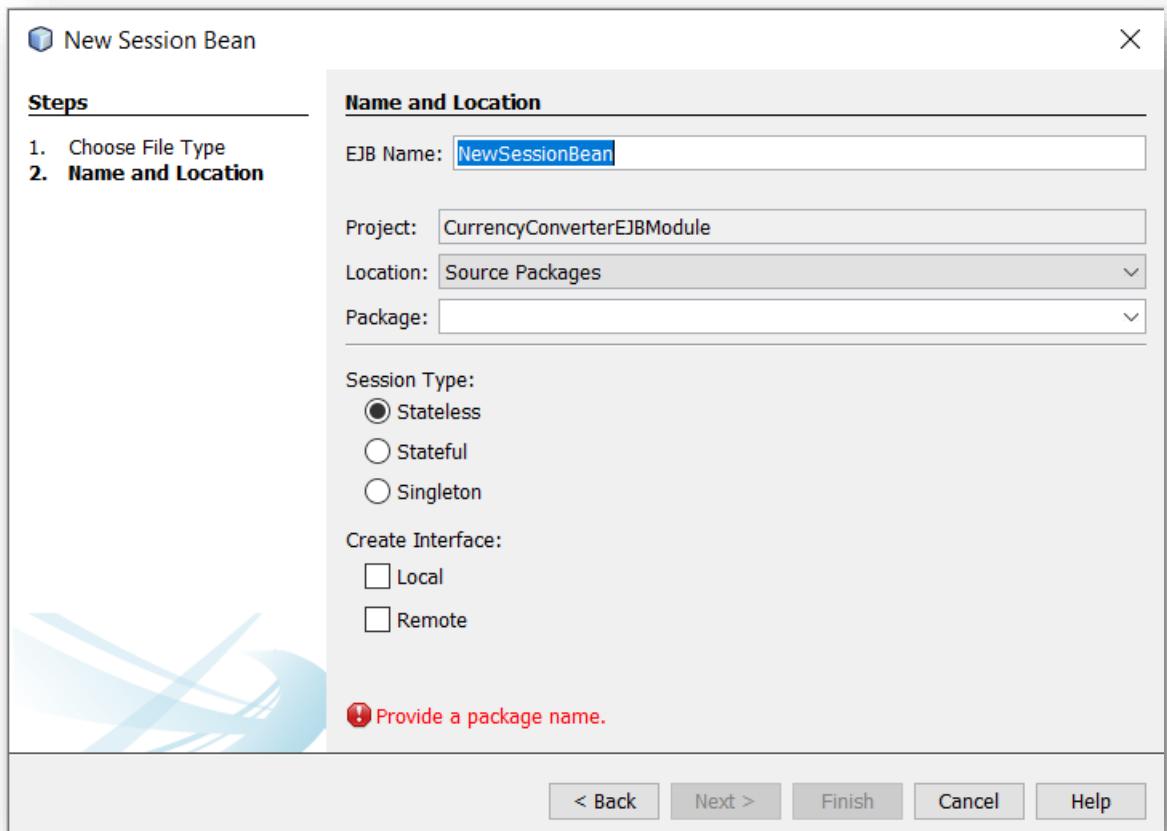


7. Click **Finish**.

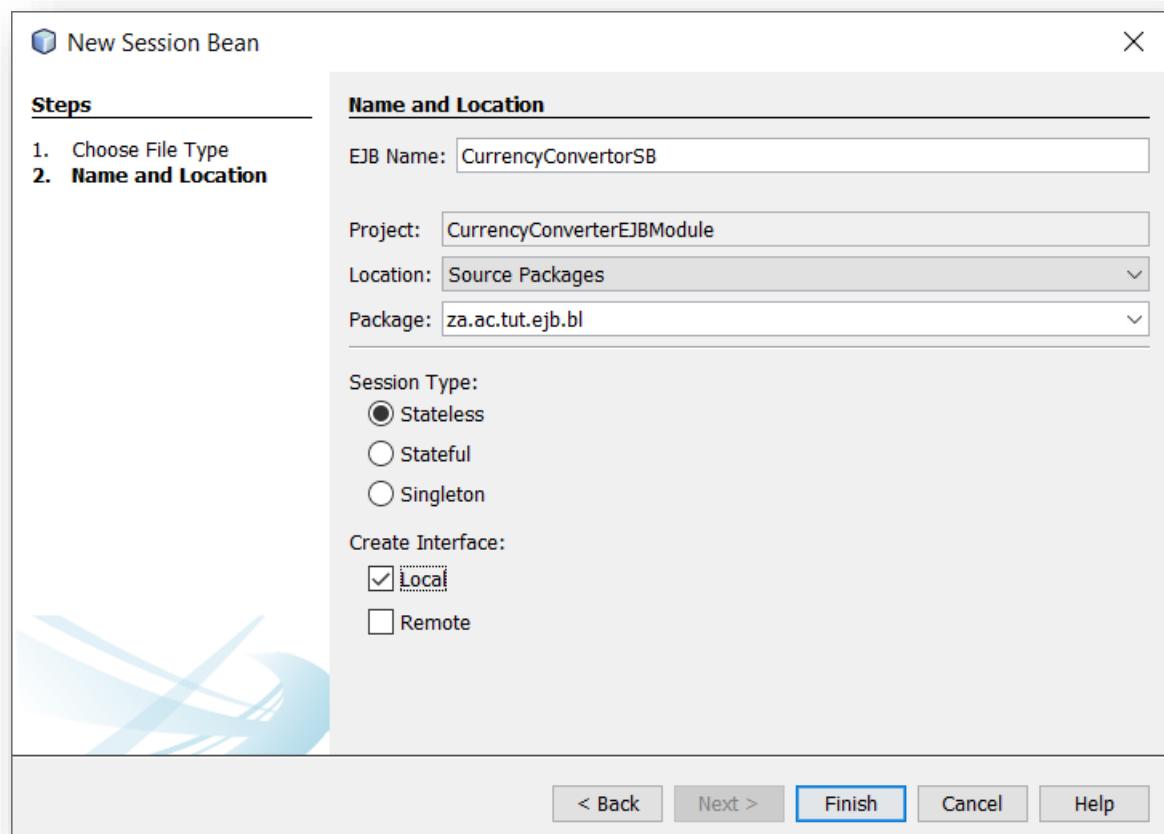


8. Create a stateles session bean. Perform the following steps:

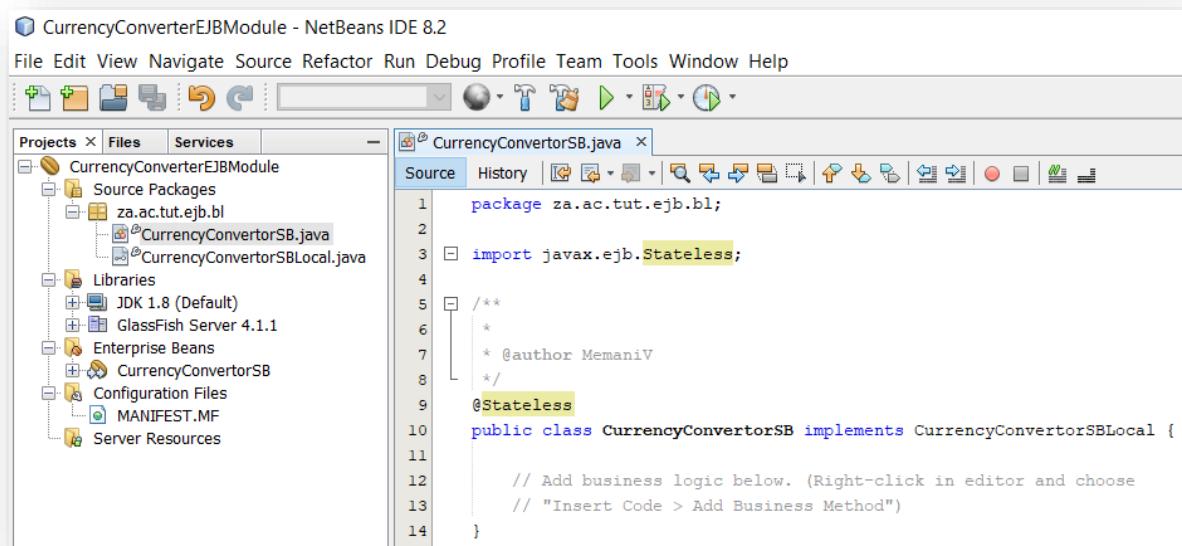
8.1 Right click on the project and select **New | Session Bean**.



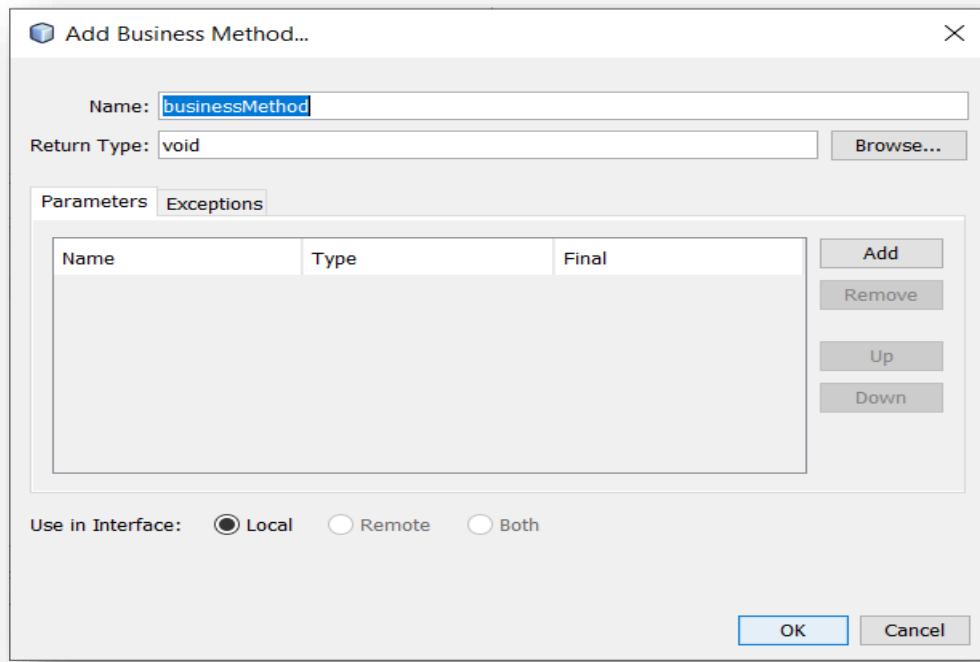
8.2 Name the EJB as **CurrencyConvertorSB**, package it under **za.ac.tut.ejb.bl**, select **Stateless** under **Session Type**, and select the **Local** interface. The **bl** acronym stands for **business logic**.



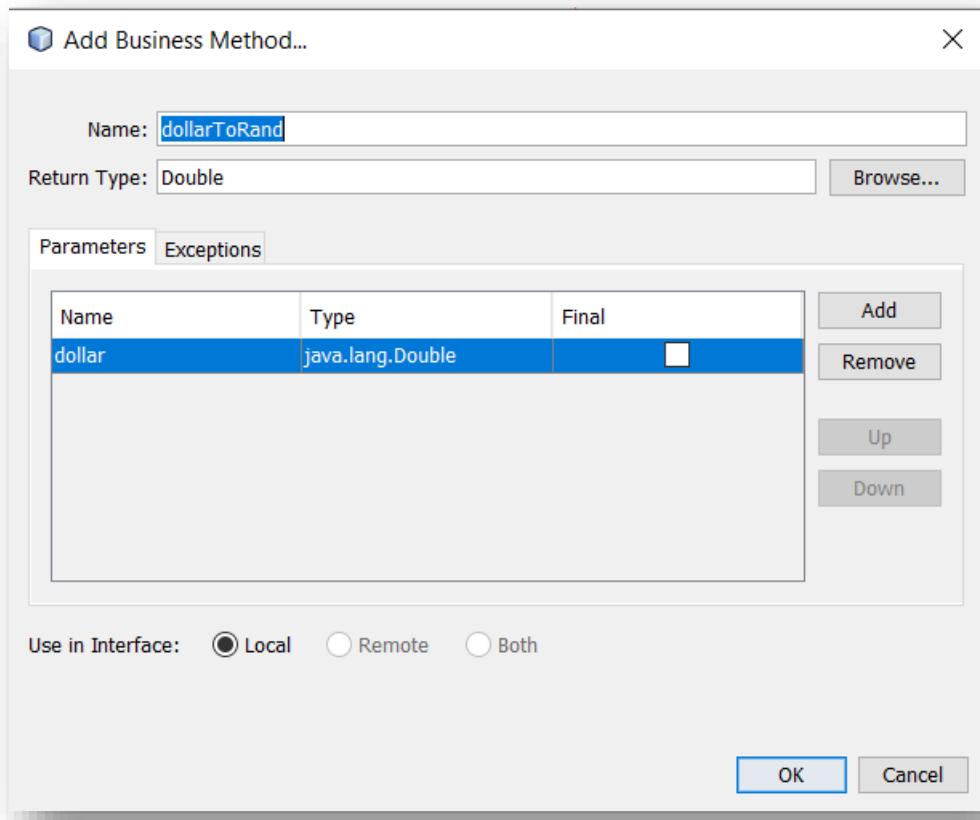
8.3 Click Finish.



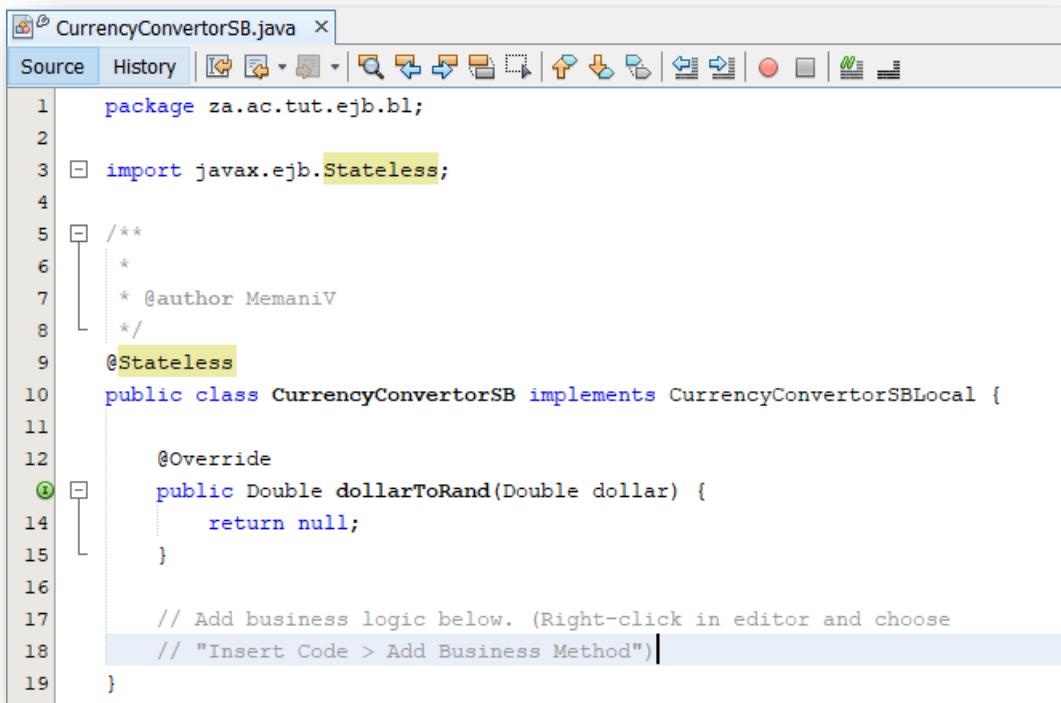
8.4 Right click inside the stateless session bean and select **Insert Code | Add Business Method**.



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

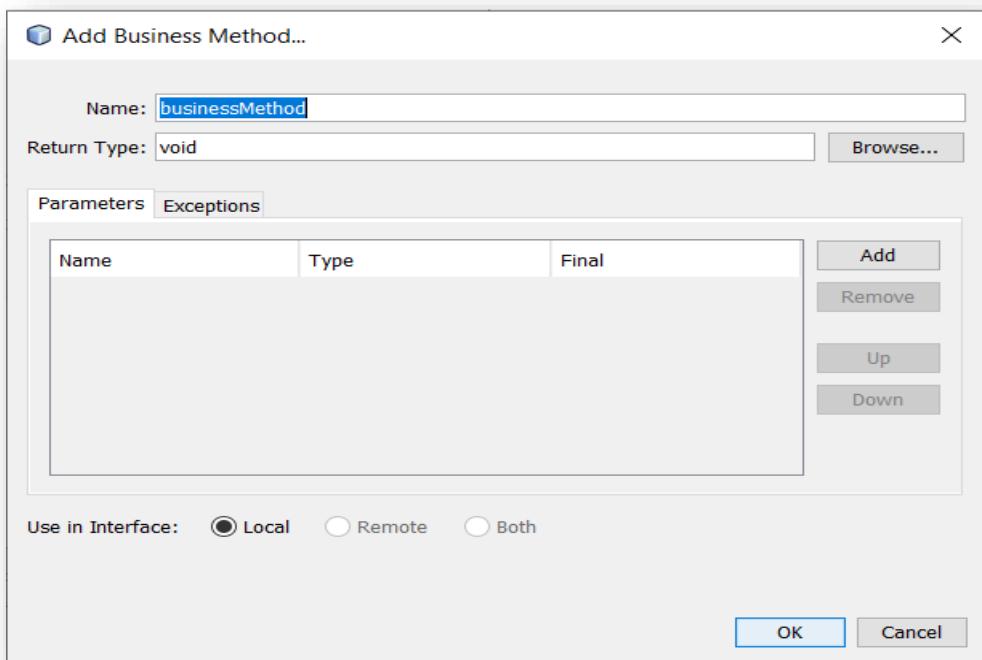


8.6 Click OK.

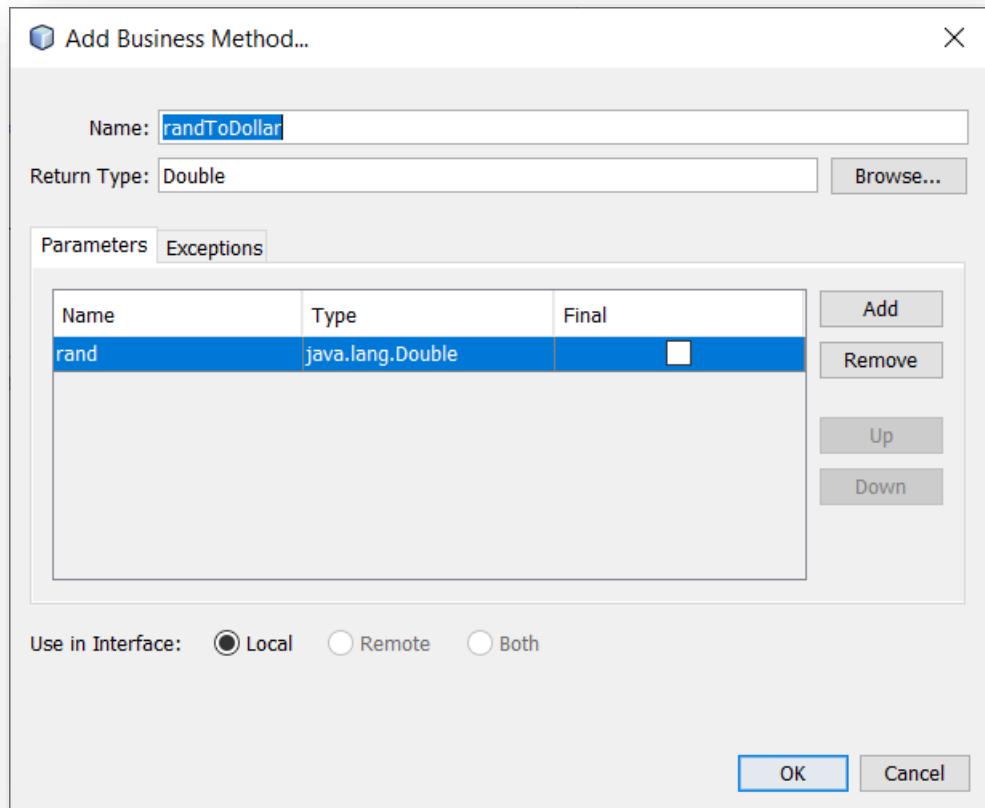


```
1 package za.ac.tut.ejb.bl;
2
3 import javax.ejb.Stateless;
4
5 /**
6 * @author MemaniV
7 */
8 @Stateless
9 public class CurrencyConvertorSB implements CurrencyConvertorSBLocal {
10
11     @Override
12     public Double dollarToRand(Double dollar) {
13         return null;
14     }
15
16     // Add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

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



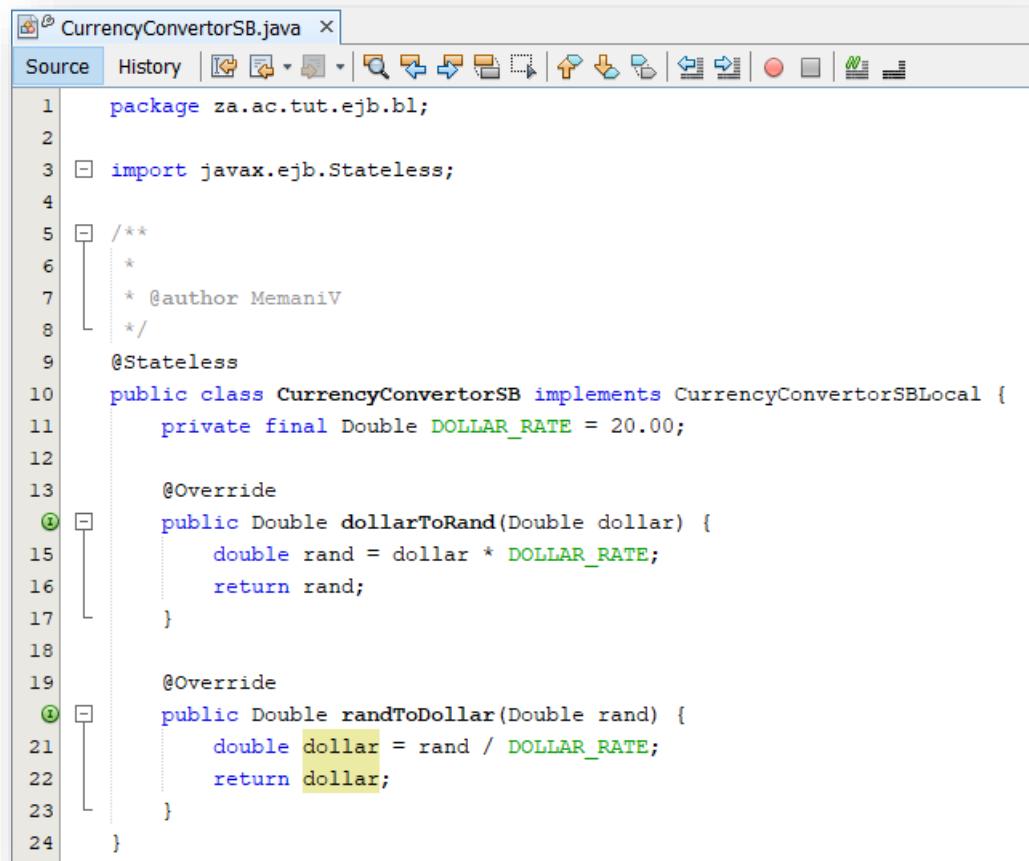
8.8 Name the method as **randToDollar**, which returns a **Double**. Click on the **Add** button to add the parameters.



8.9 Click **OK**.

```
package za.ac.tut.ejb.bl;
import javax.ejb.Stateless;
/**
 * 
 * @author MemaniV
 */
@Stateless
public class CurrencyConvertorSB implements CurrencyConvertorSBLocal {
    @Override
    public Double dollarToRand(Double dollar) {
        return null;
    }
    @Override
    public Double randToDollar(Double rand) {
        return null;
    }
}
```

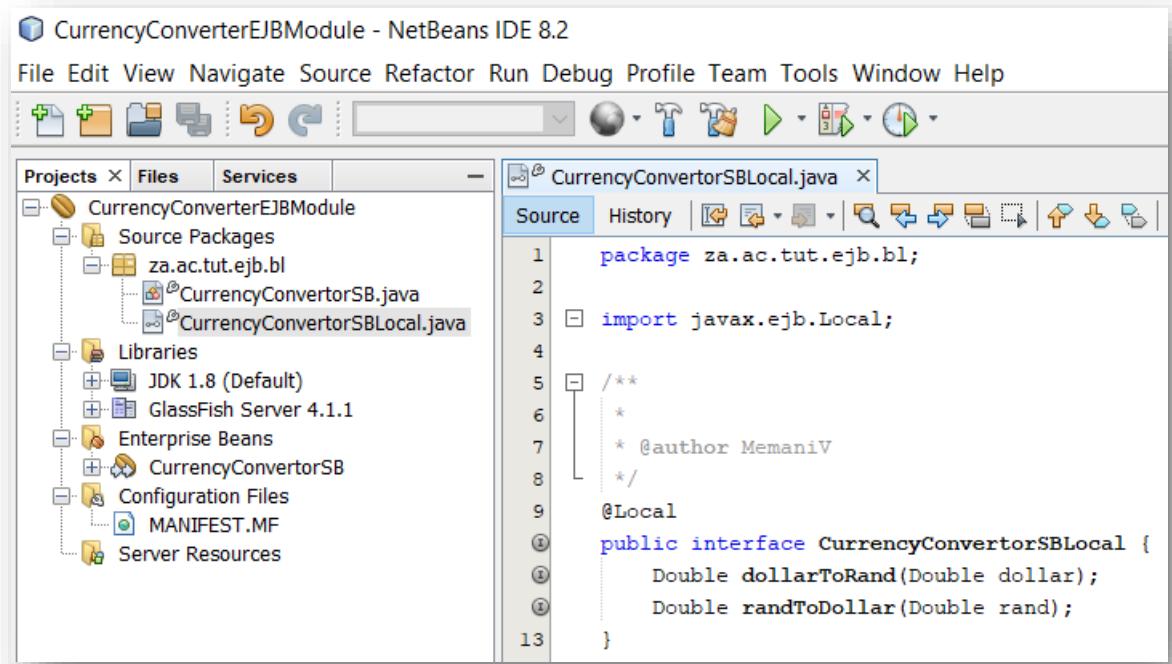
8.10 Complete the source code of the Stateless Session Bean.



The screenshot shows the NetBeans IDE interface with the file `CurrencyConverterSB.java` open. The code defines a stateless session bean named `CurrencyConverterSB` that implements the local interface `CurrencyConverterSBLocal`. The implementation includes two methods: `dollarToRand` and `randToDollar`, both of which use a fixed exchange rate of 20.00.

```
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 CurrencyConverterSB implements CurrencyConverterSBLocal {
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.



The screenshot shows the NetBeans IDE interface with the file `CurrencyConverterSBLocal.java` open. This is the local interface for the `CurrencyConverterSB` session bean, defining two methods: `dollarToRand` and `randToDollar`.

```
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 CurrencyConverterSBLocal {
11     Double dollarToRand(Double dollar);
12     Double randToDollar(Double rand);
13 }
```

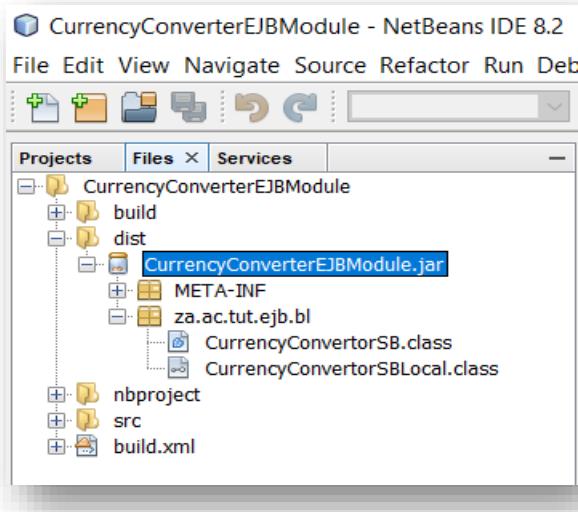
9. Compile the EJB project. Perform the following steps:

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

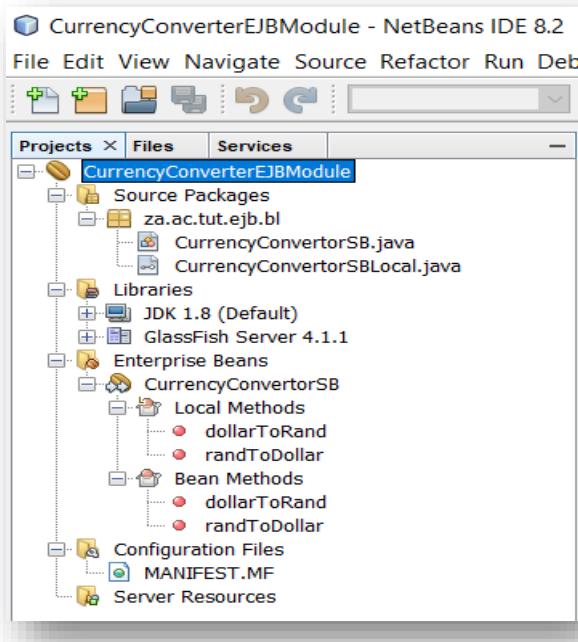


```
Java DB Database Process X CurrencyConverterEJBModule (clean,dist) X
Copying 1 file to C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConverterEJBModule\build\classes\META-INF
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConverterEJBModule\build\empty
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConverterEJBModule\build\generated-sources\ap-source-output
Compiling 2 source files to C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConverterEJBModule\build\classes
compile:
library-inclusion-in-archive:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConverterEJBModule\dist
Building jar: C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConverterEJBModule\dist\CurrencyConverterEJBModule.jar
dist:
BUILD SUCCESSFUL (total time: 1 second)
```

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



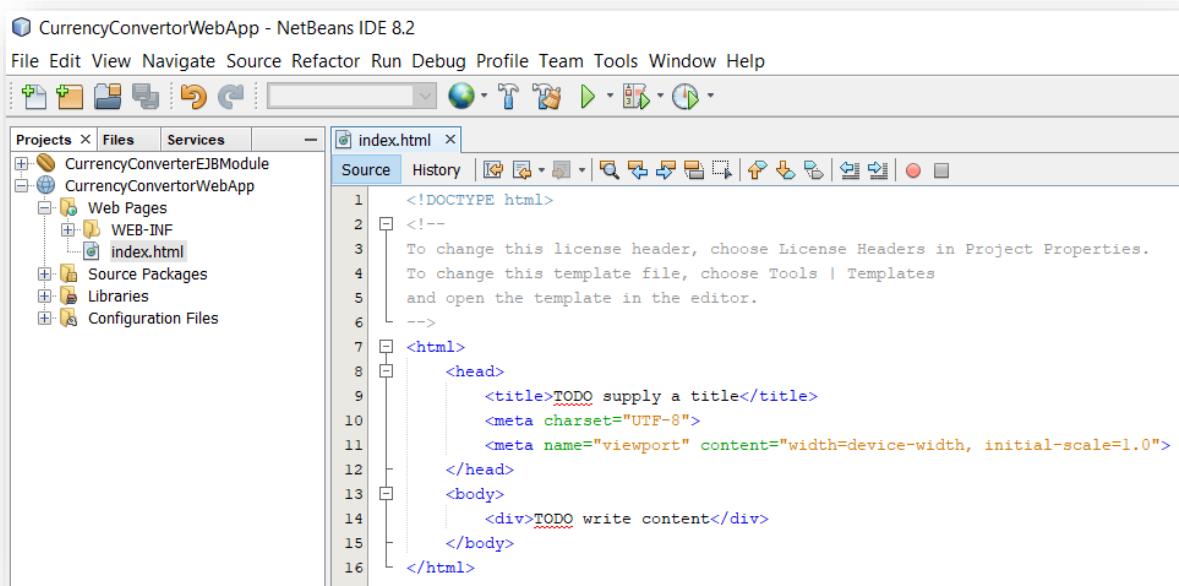
9.3 View the complete EJB project structure.



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**.

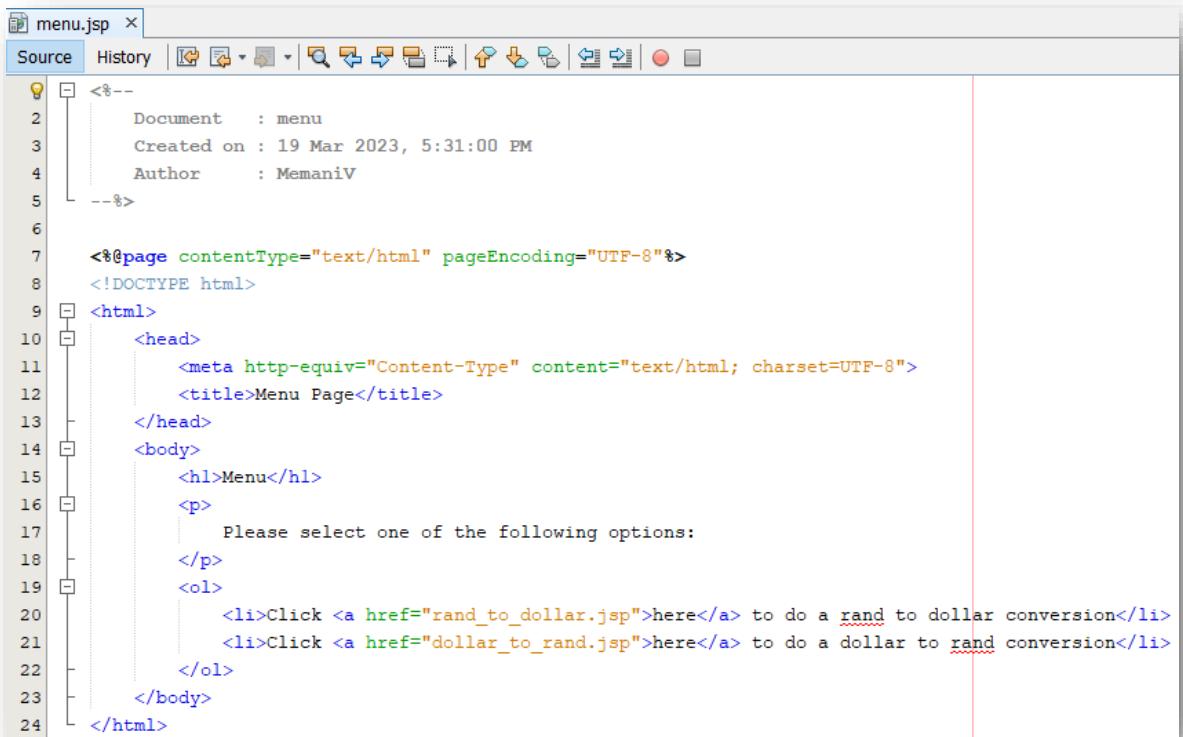


2. Edit the **index.html** file.

The screenshot shows the NetBeans IDE interface with the title bar "index.html". The menu bar includes Source, History, and other icons. The right panel displays the "index.html" source code with changes made to the body content:

```
<!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 convertor web app. Click <a href="#menu.jsp">here</a> to start.
        </p>
    </body>
</html>
```

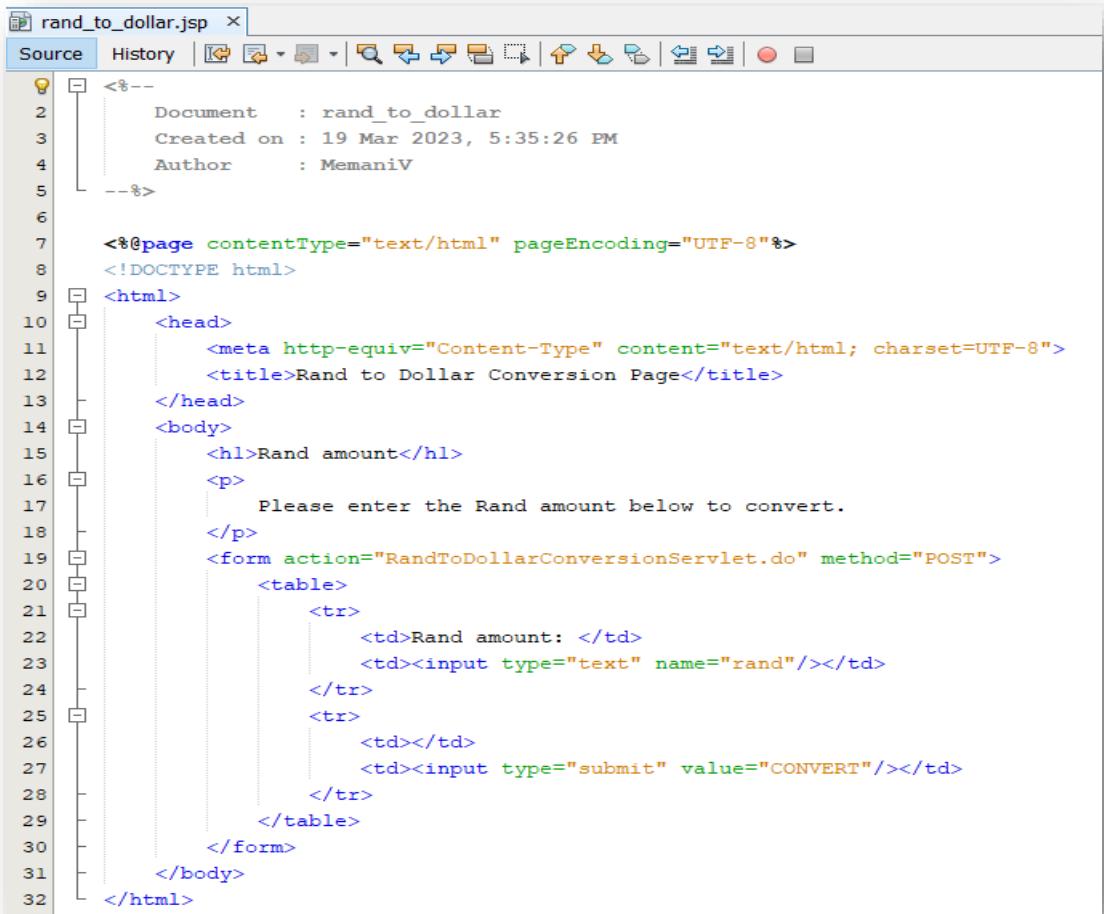
3. Create the **menu.jsp** file.



The screenshot shows the code editor window for the **menu.jsp** file. The code is a JSP page with the following structure:

```
<%--  
    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>
```

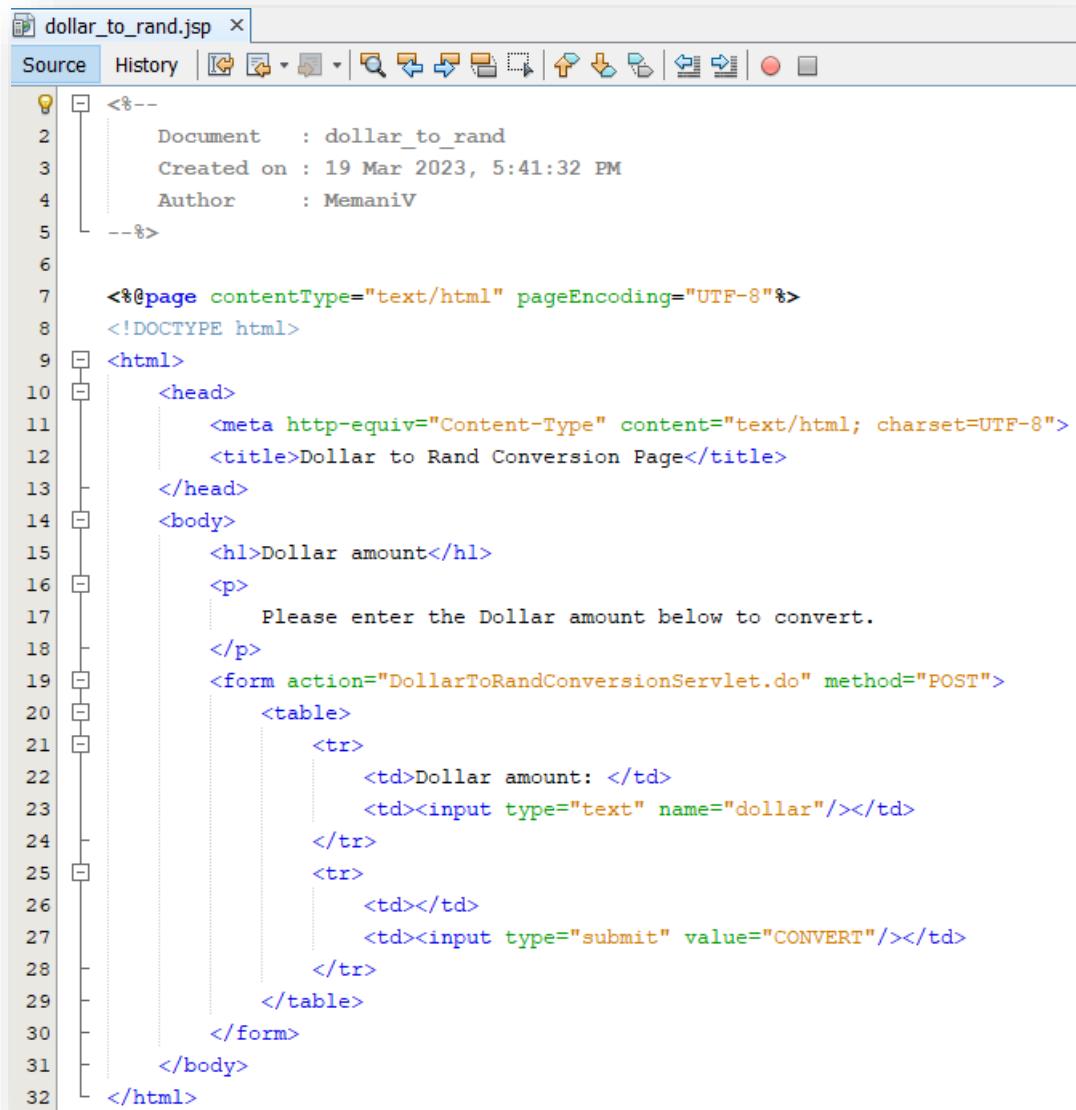
4. Create the **rand_to_dollar.jsp** file.



The screenshot shows the code editor window for the **rand_to_dollar.jsp** file. The code is a JSP page with the following structure:

```
<%--  
    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>
```

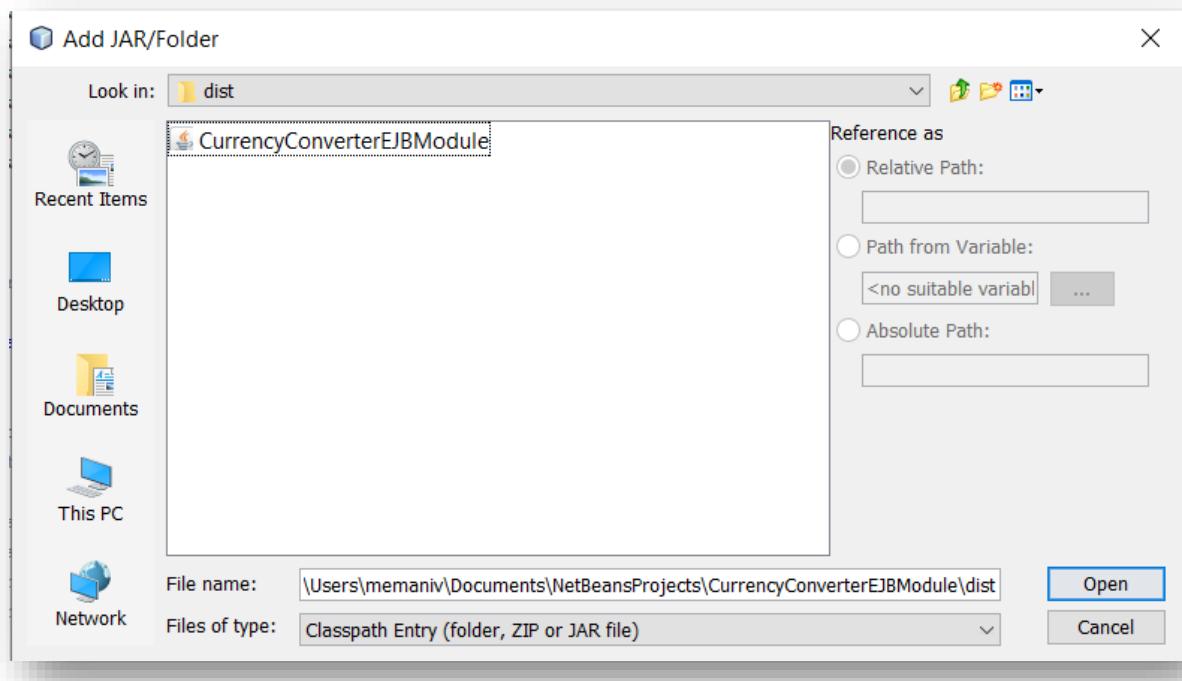
5. Create the **dollar_to_rand.jsp** file.



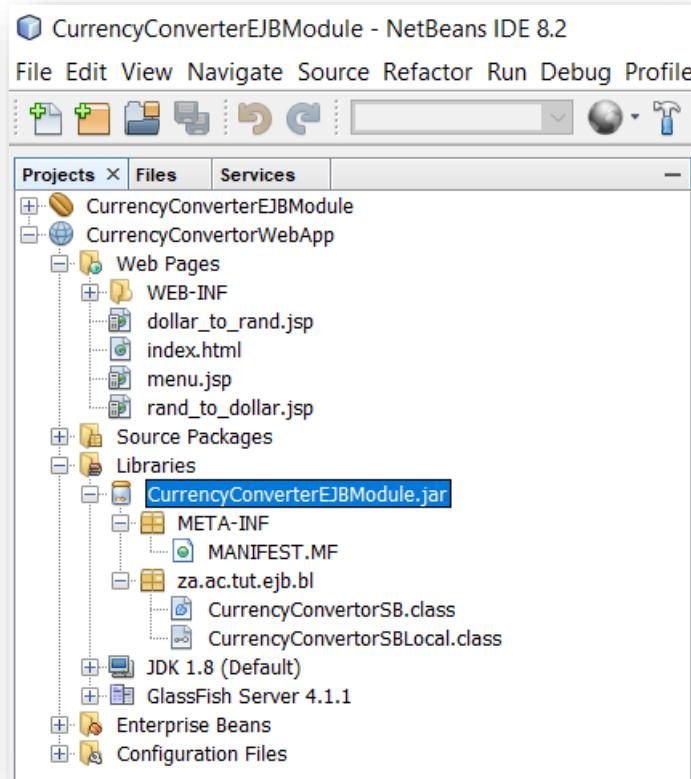
```
<%--  
1 Document : dollar_to_rand  
2 Created on : 19 Mar 2023, 5:41:32 PM  
3 Author : MemaniV  
4 --%>  
5  
6  
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>  
8 <!DOCTYPE html>  
9 <html>  
10 <head>  
11     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
12     <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.

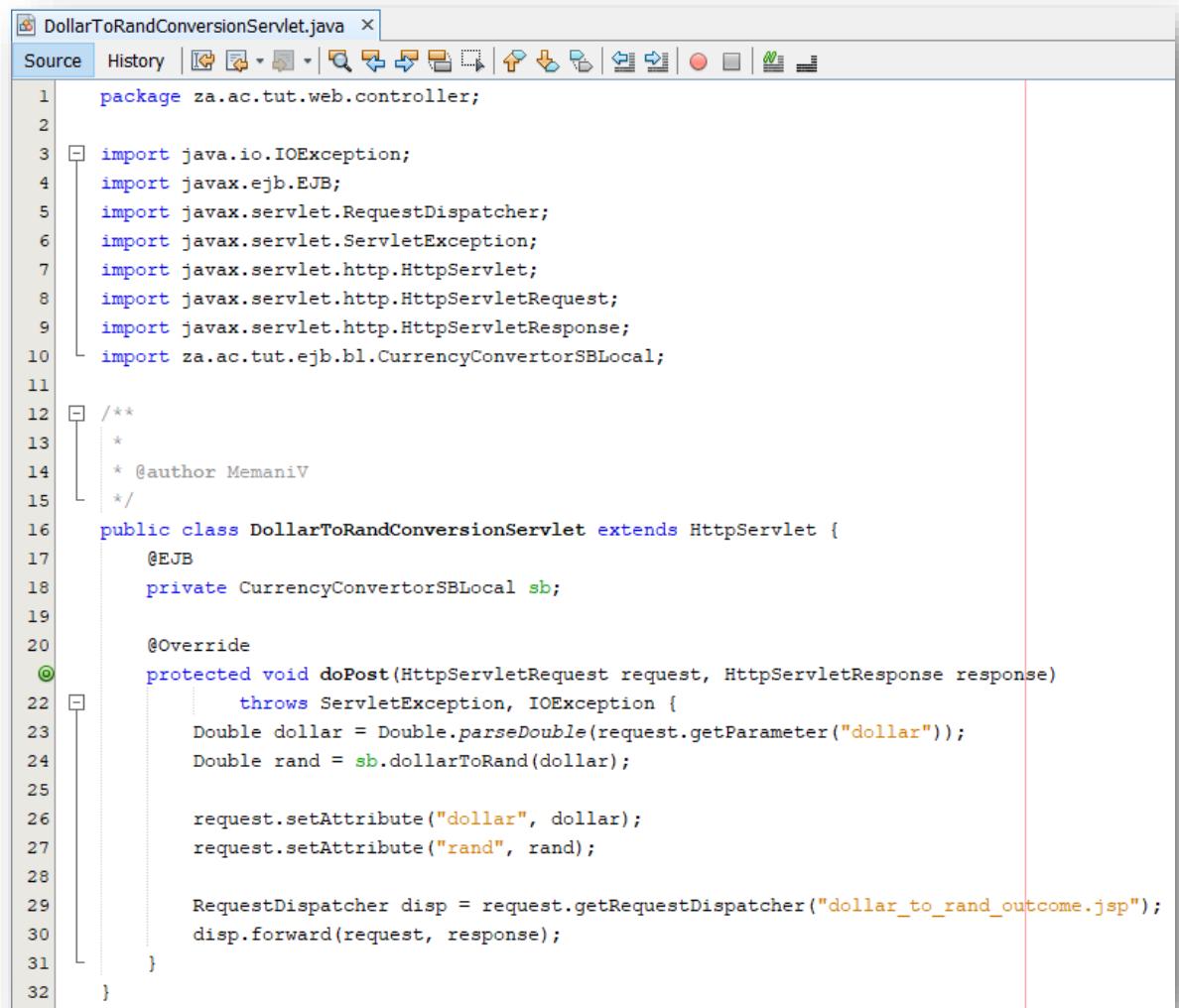
6.1 Right-click on the **Libraries** folder of the Web App project and select **Add JAR/Folder**. Navigate to the **dist** folder of **CurrencyConverterEJBModule**.



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



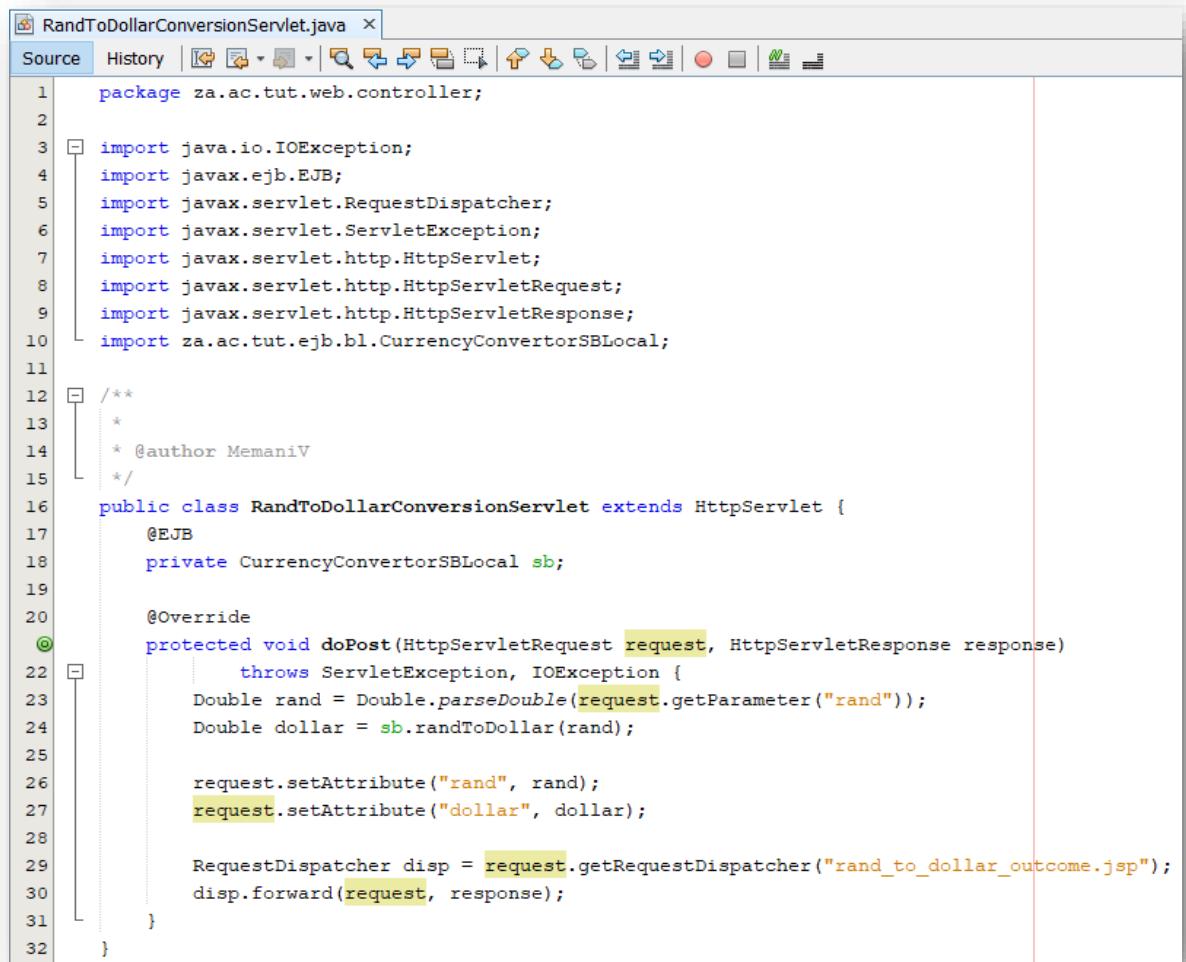
7. Create the **DollarToRandConversionServlet.java** file.



The screenshot shows a Java code editor window with the title bar "DollarToRandConversionServlet.java". The menu bar includes "Source", "History", and various tool icons. The code itself is a Java servlet named "DollarToRandConversionServlet" that extends "HttpServlet". It imports several Java packages and classes, including "java.io.IOException", "javax.ejb.EJB", "javax.servlet.RequestDispatcher", "javax.servlet.ServletException", "javax.servlet.http.HttpServlet", "javax.servlet.http.HttpServletRequest", "javax.servlet.http.HttpServletResponse", and "za.ac.tut.ejb.bl.CurrencyConvertorSBLocal". The class contains a single method, "doPost", which retrieves a "dollar" parameter from the request, converts it to a double using "Double.parseDouble", and then calls the "dollarToRand" method of the "sb" bean. The converted value is then set as an attribute in the request. Finally, the code uses "RequestDispatcher" to forward the request to a JSP page named "dollar_to_rand_outcome.jsp". The code is annotated with Javadoc comments and includes an author tag "MemaniV".

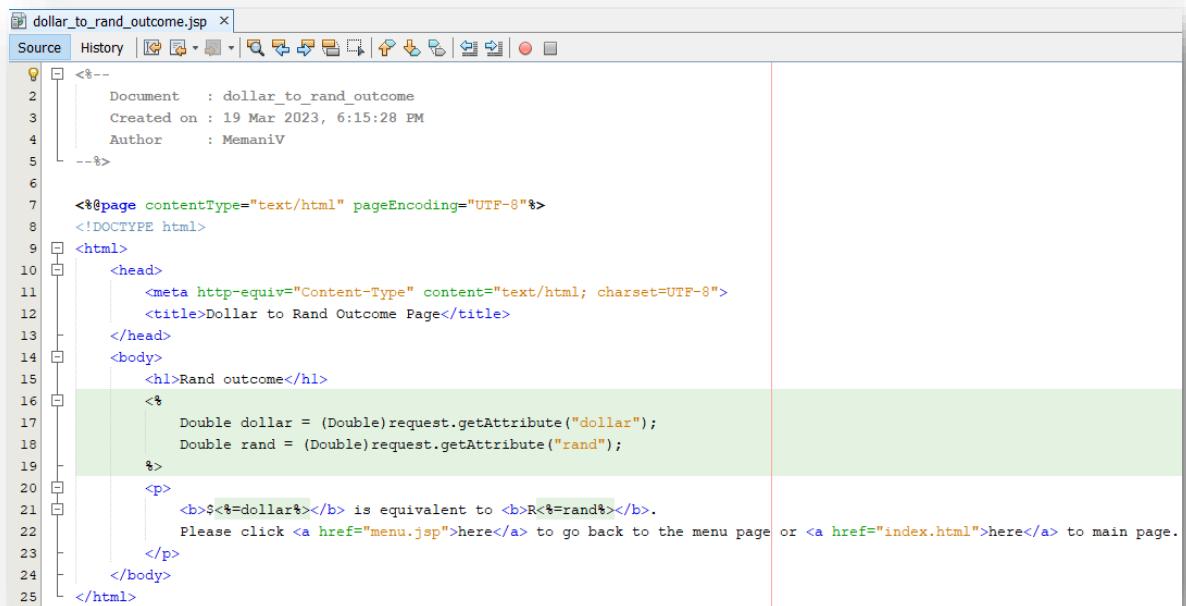
```
1 package za.ac.tut.web.controller;
2
3 import java.io.IOException;
4 import javax.ejb.EJB;
5 import javax.servlet.RequestDispatcher;
6 import javax.servlet.ServletException;
7 import javax.servlet.http.HttpServlet;
8 import javax.servlet.http.HttpServletRequest;
9 import javax.servlet.http.HttpServletResponse;
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.



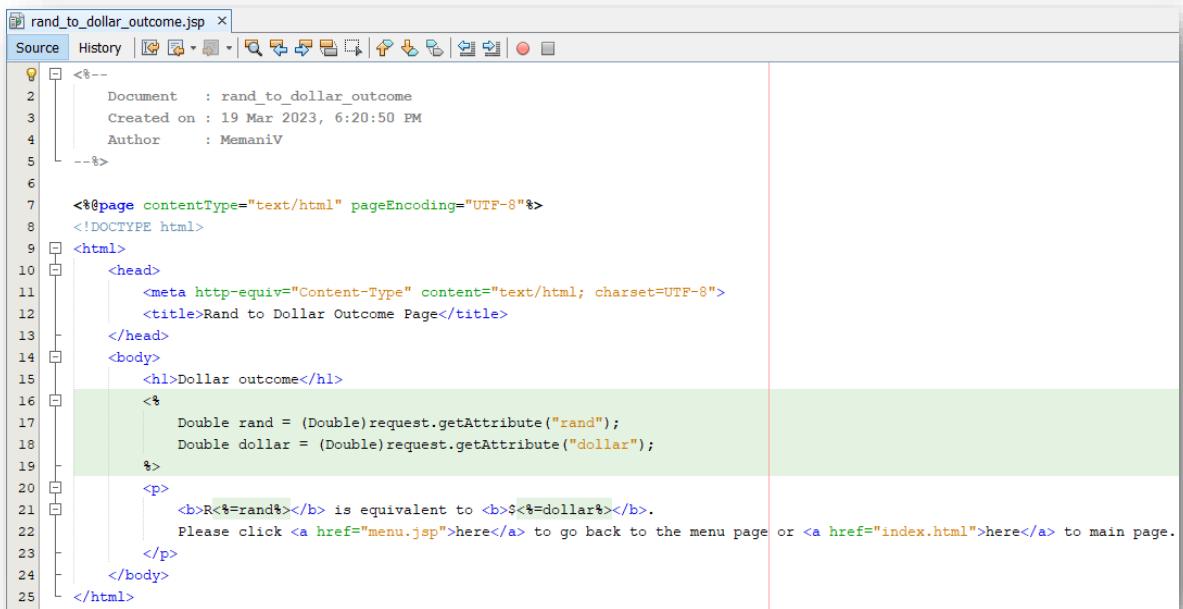
```
1 package za.ac.tut.web.controller;
2
3 import java.io.IOException;
4 import javax.ejb.EJB;
5 import javax.servlet.RequestDispatcher;
6 import javax.servlet.ServletException;
7 import javax.servlet.http.HttpServlet;
8 import javax.servlet.http.HttpServletRequest;
9 import javax.servlet.http.HttpServletResponse;
10 import za.ac.tut.ejb.bl.CurrencyConvertorSBL;
11
12 /**
13 *
14 * @author MemaniV
15 */
16 public class RandToDollarConversionServlet extends HttpServlet {
17     @EJB
18     private CurrencyConvertorSBL 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.



```
<%--  
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>
```

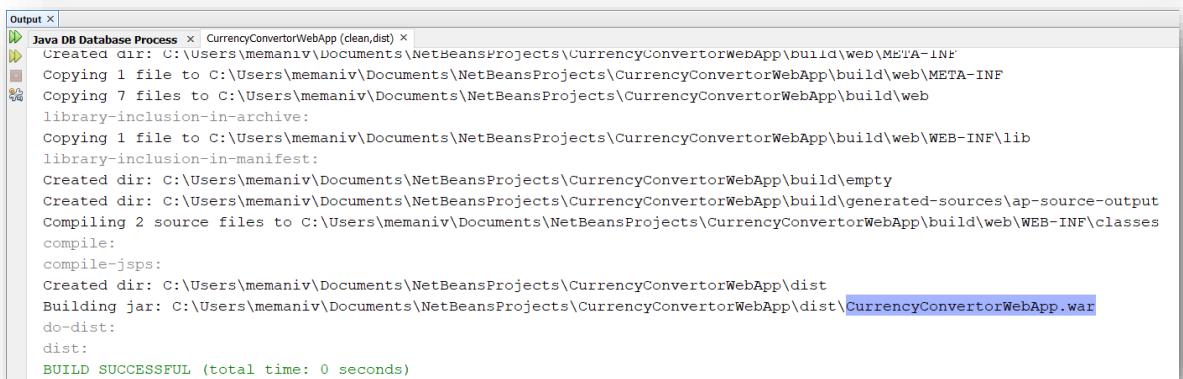
10. Create the `rand_to_dollar_outcome.jsp` file.



The screenshot shows the NetBeans IDE interface with the file `rand_to_dollar_outcome.jsp` open. The code is a JSP page that displays a random dollar amount based on a user input. The code includes imports for `java.util.Random` and `java.lang.Double`, and uses `request.getAttribute` to get values from the request. It also includes links to `menu.jsp` and `index.html`.

```
<%--  
2 Document : rand_to_dollar_outcome  
3 Created on : 19 Mar 2023, 6:20:50 PM  
4 Author : MemaniV  
5 --%>  
  
<%@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>
```

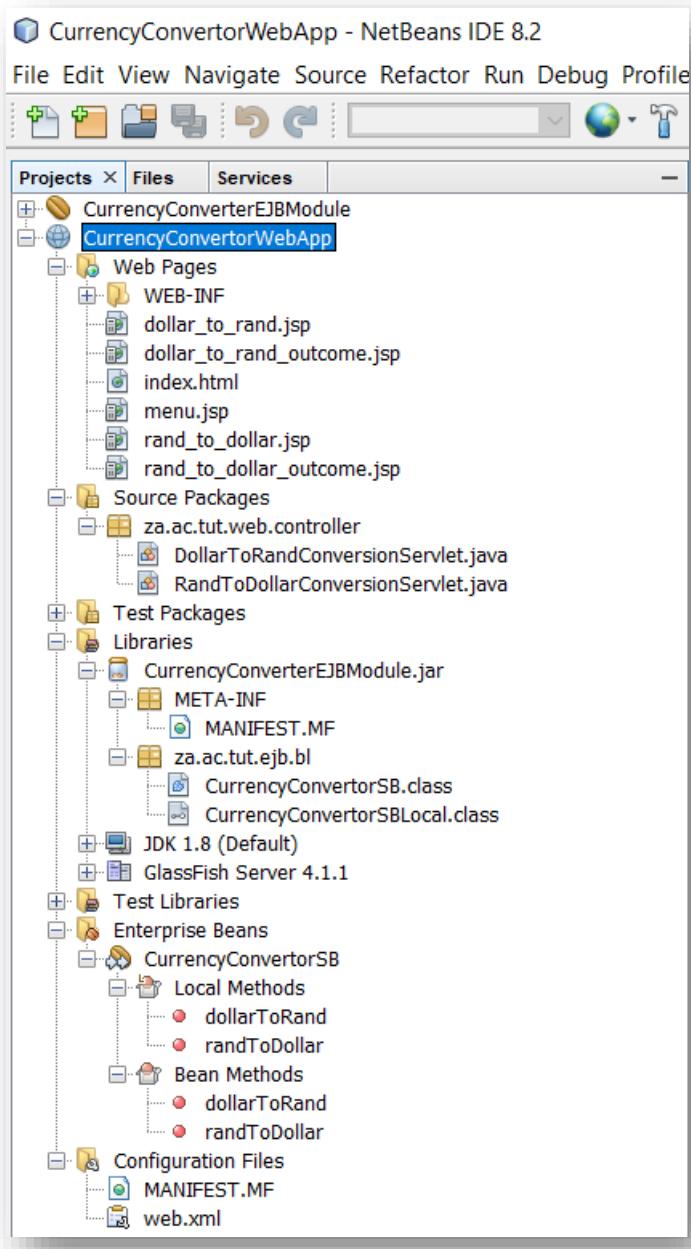
11. Compile the web project.



The screenshot shows the NetBeans IDE Output window displaying the build process for the `CurrencyConvertorWebApp` project. The output shows various build steps such as creating directories, copying files, compiling Java source code, and building a WAR file. The final message indicates a successful build.

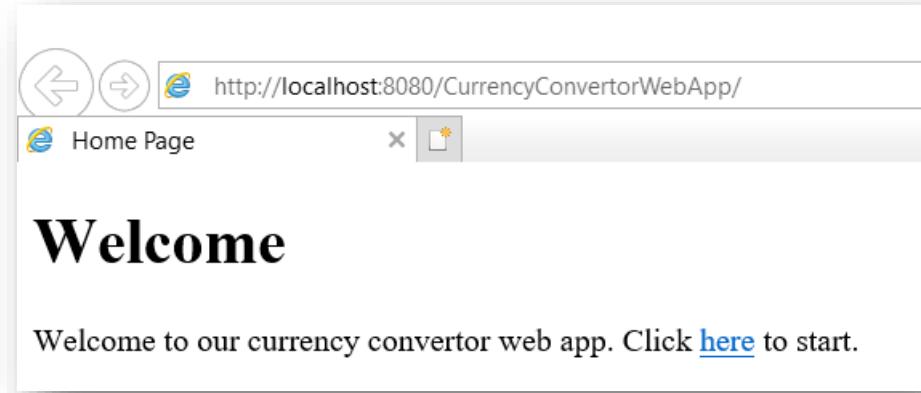
```
Java DB Database Process x CurrencyConvertorWebApp (clean,dist) x  
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConvertorWebApp\build\web\META-INF  
Copying 1 file to C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConvertorWebApp\build\web\META-INF  
Copying 7 files to C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConvertorWebApp\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-jsp:  
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.

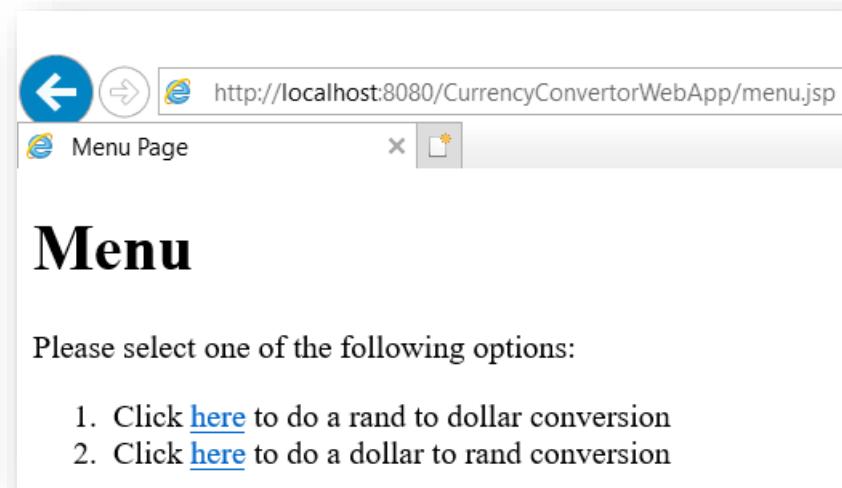


Part C - Run the web client project.

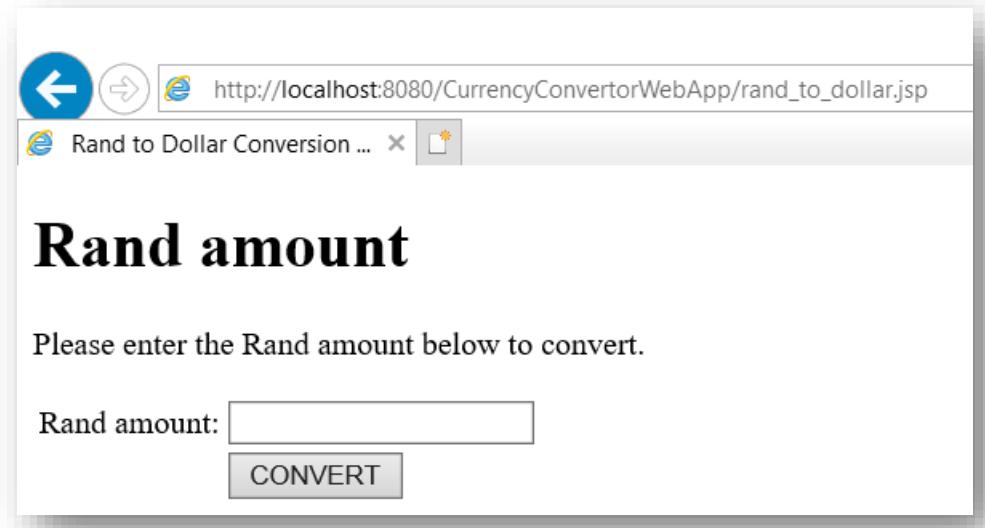
Launch the web app.



Click on the link.



Click on the first link.



Enter the amount.

The screenshot shows a web browser window with the URL http://localhost:8080/CurrencyConvertorWebApp/rand_to_dollar.jsp. The title bar says "Rand to Dollar Conversion ...". The main content area has a large heading "Rand amount". Below it is a text input field containing "40.00" with the placeholder "Please enter the Rand amount below to convert.". A "CONVERT" button is located below the input field.

Click on the button.

The screenshot shows a web browser window with the URL <http://localhost:8080/CurrencyConvertorWebApp/RandToDollarConversionServlet.do>. The title bar says "Rand to Dollar Outcome Pa...". The main content area has a large heading "Dollar outcome". Below it is a message: "R40.0 is equivalent to \$2.0. Please click [here](#) to go back to the menu page or [here](#) to main page."

Go back to the menu page.

The screenshot shows a web browser window with the URL <http://localhost:8080/CurrencyConvertorWebApp/menu.jsp>. The title bar says "Menu Page". The main content area has a large heading "Menu". Below it is a text: "Please select one of the following options:" followed by a numbered list.

1. Click [here](#) to do a rand to dollar conversion
2. Click [here](#) to do a dollar to rand conversion

Click on the second link.

The screenshot shows a web browser window with the URL http://localhost:8080/CurrencyConvertorWebApp/dollar_to_rand.jsp. The title bar says "Dollar to Rand Conversion ...". The main content area has a large heading "Dollar amount". Below it is the text "Please enter the Dollar amount below to convert.". There is an input field labeled "Dollar amount:" and a button labeled "CONVERT".

Enter the dollar amount

The screenshot shows the same web browser window as before, but now the input field contains the value "2". The rest of the interface is identical to the first screenshot.

Click on the button.

The screenshot shows a web browser window with the URL <http://localhost:8080/CurrencyConvertorWebApp/DollarToRandConversionServlet.do>. The title bar says "Dollar to Rand Outcome Pa...". The main content area has a large heading "Rand outcome". Below it is the text "\$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**.

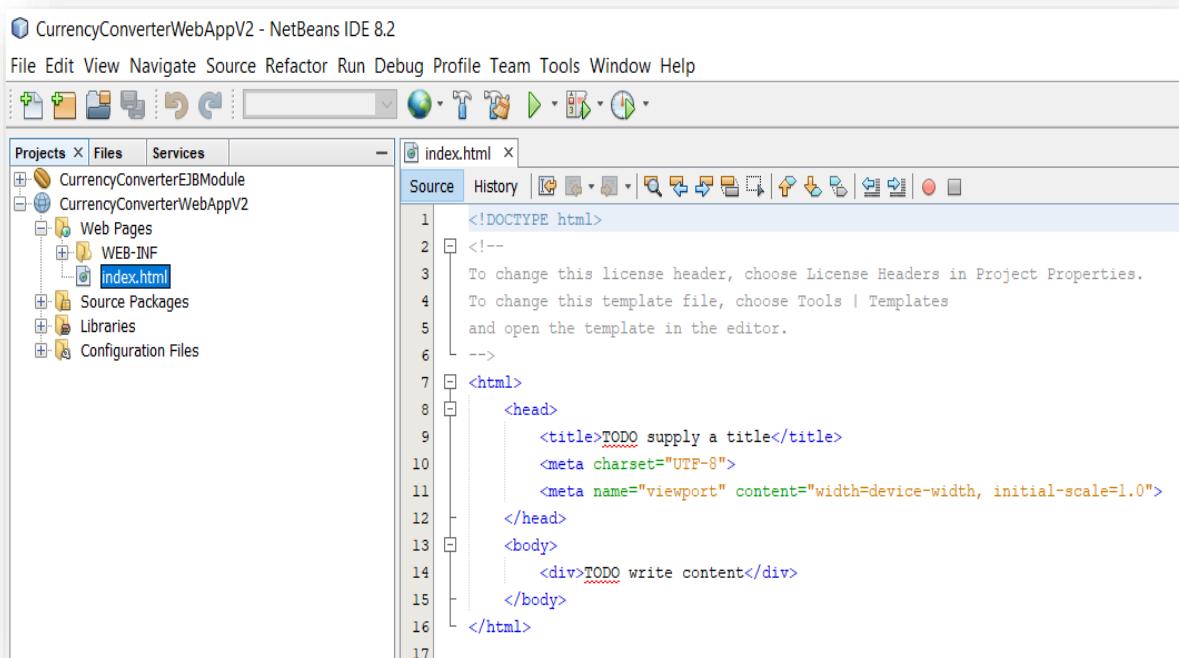
Part A - Create an EJB module project.

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

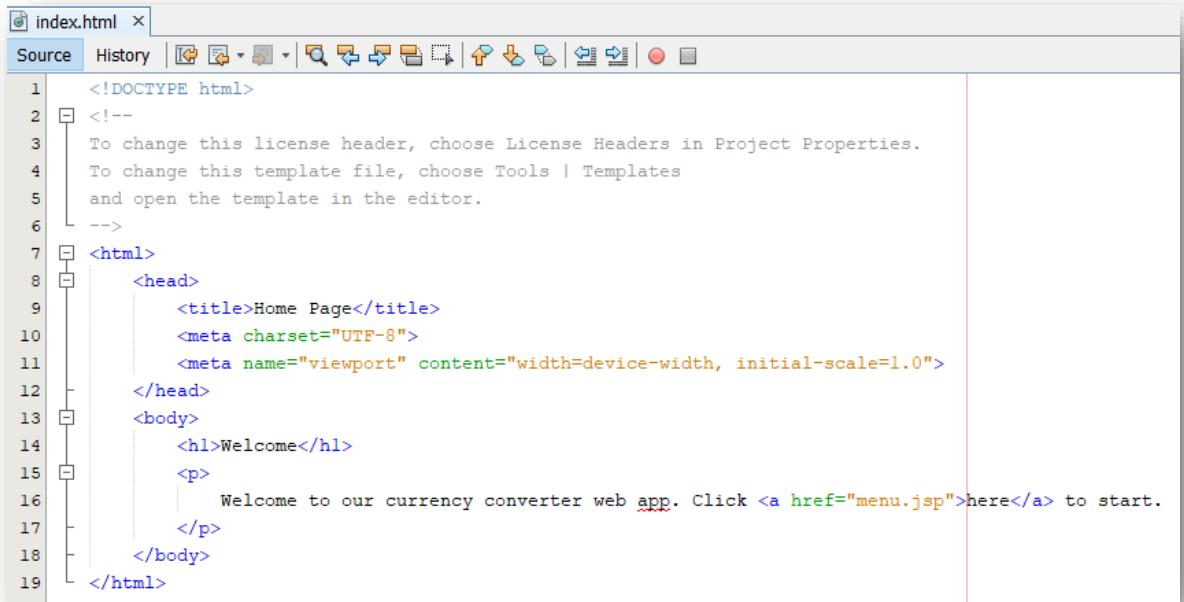
Part B - Create a Web client project

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

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



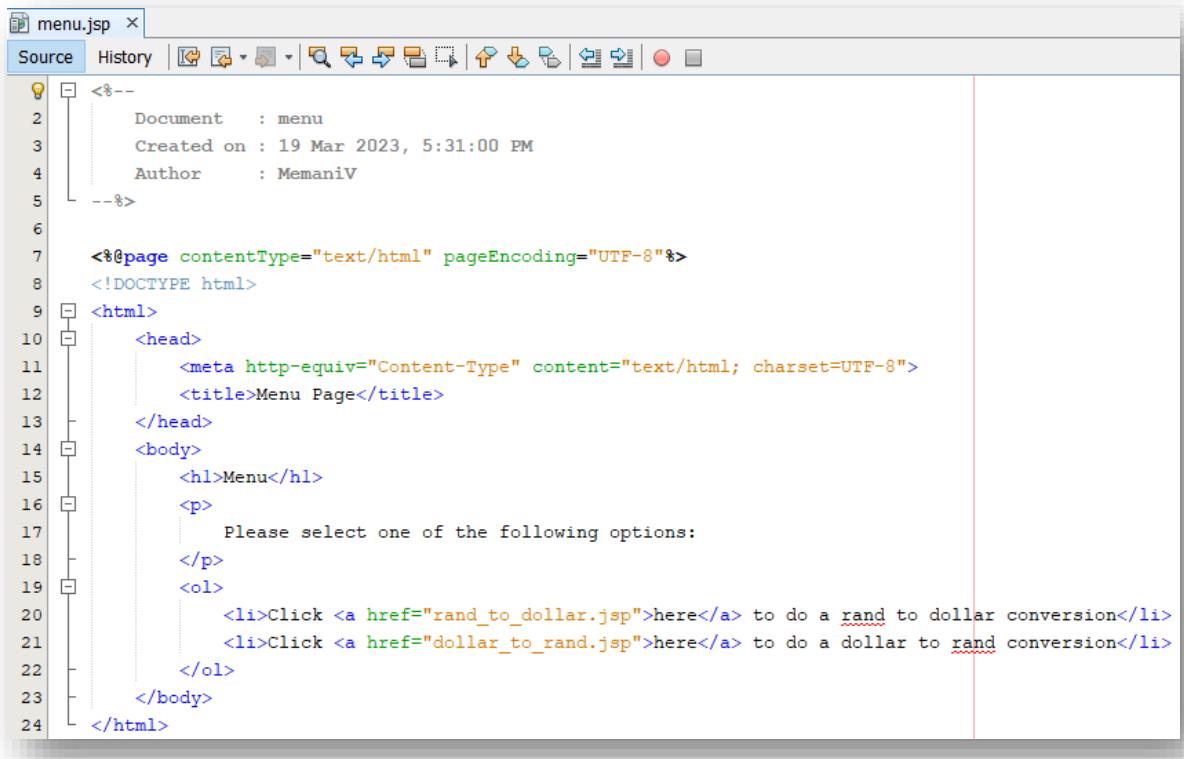
2. Edit the **index.html** file.



The screenshot shows a code editor window titled "index.html x". The "Source" tab is selected. The code is as follows:

```
<!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.

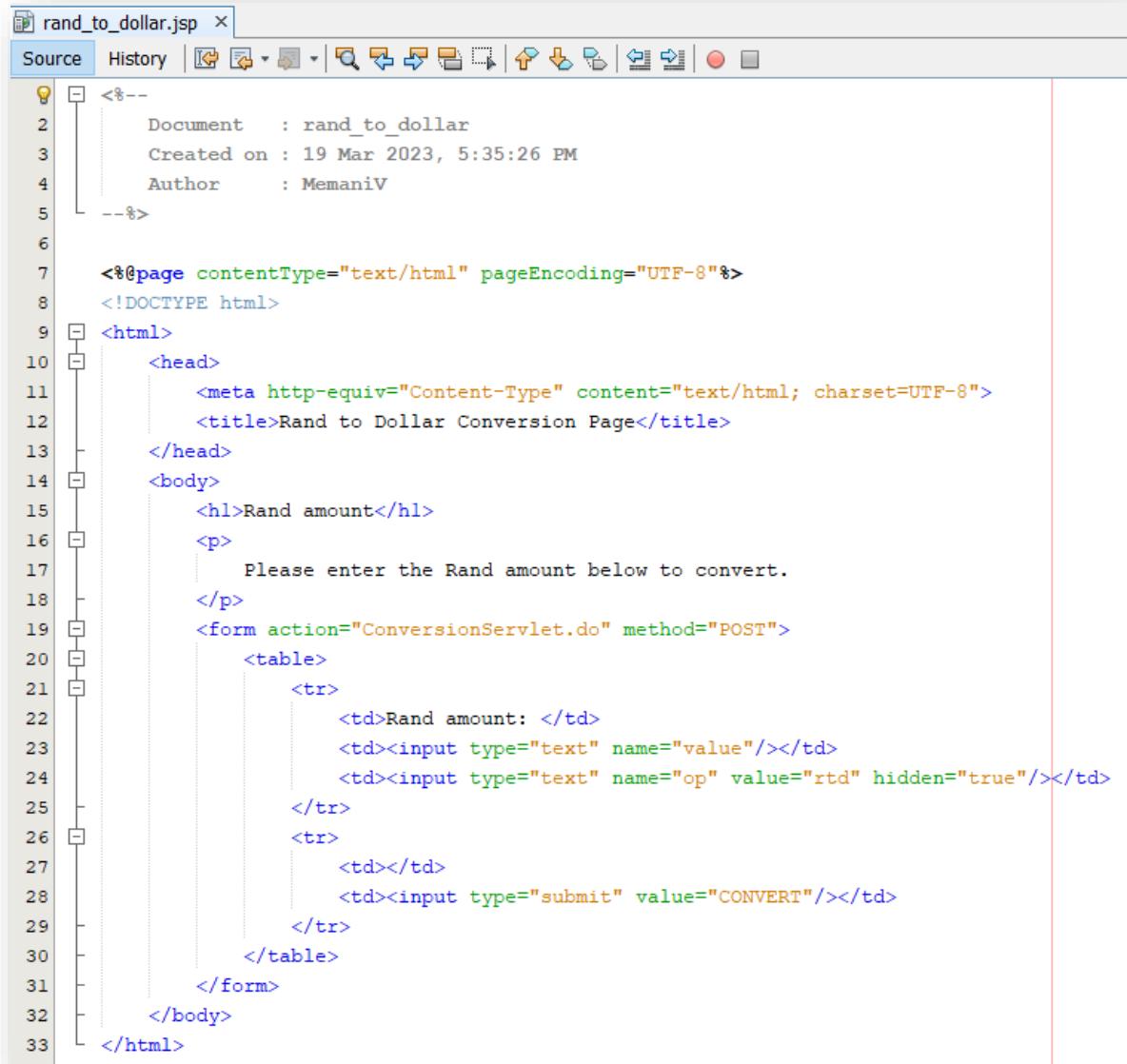


The screenshot shows a code editor window titled "menu.jsp x". The "Source" tab is selected. The code is as follows:

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

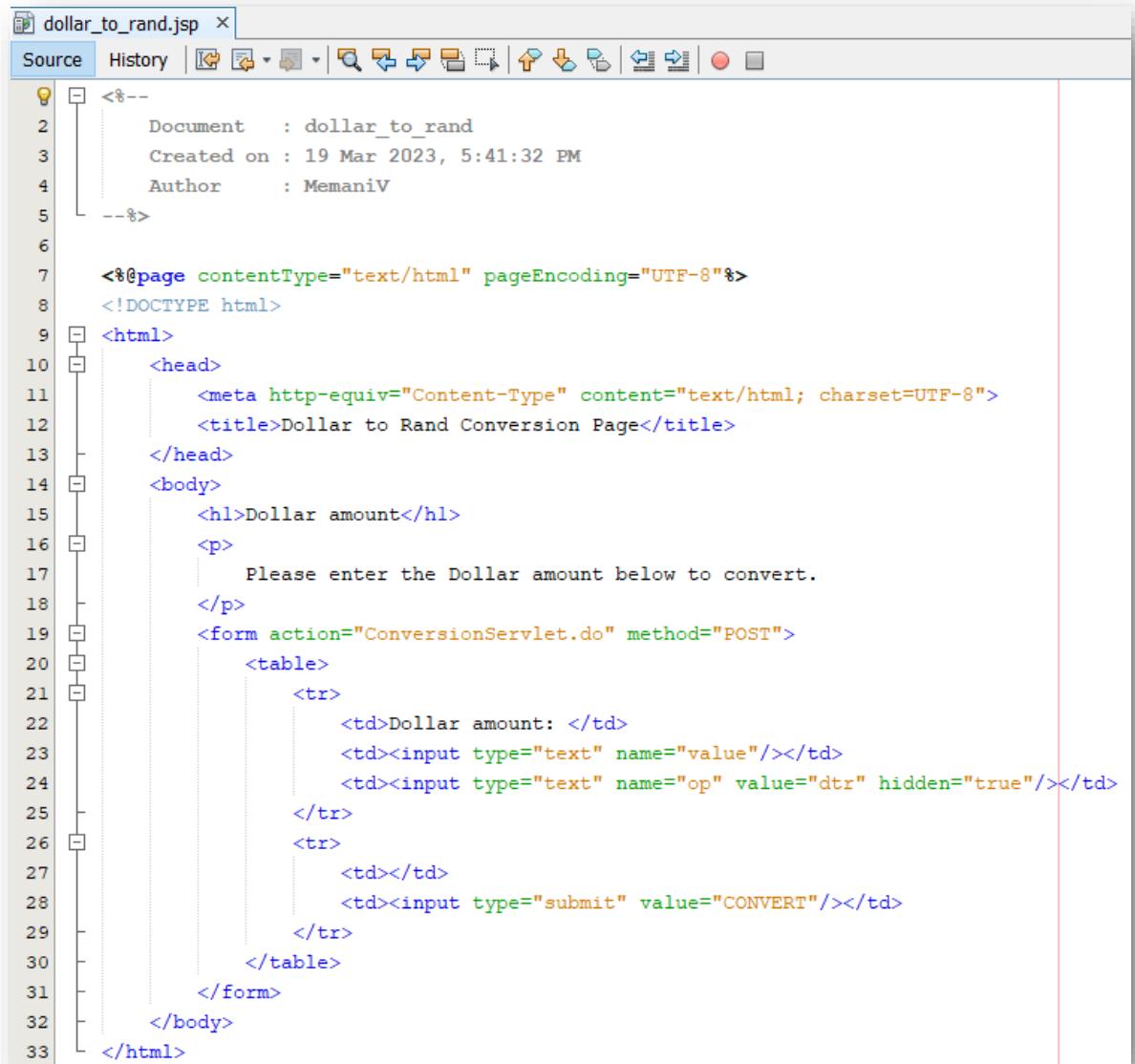
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Menu Page</title>
    </head>
    <body>
        <h1>Menu</h1>
        <p>
            Please select one of the following 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.



```
<%--  
1 Document      : rand_to_dollar  
2 Created on   : 19 Mar 2023, 5:35:26 PM  
3 Author        : MemaniV  
4--%>  
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>
```

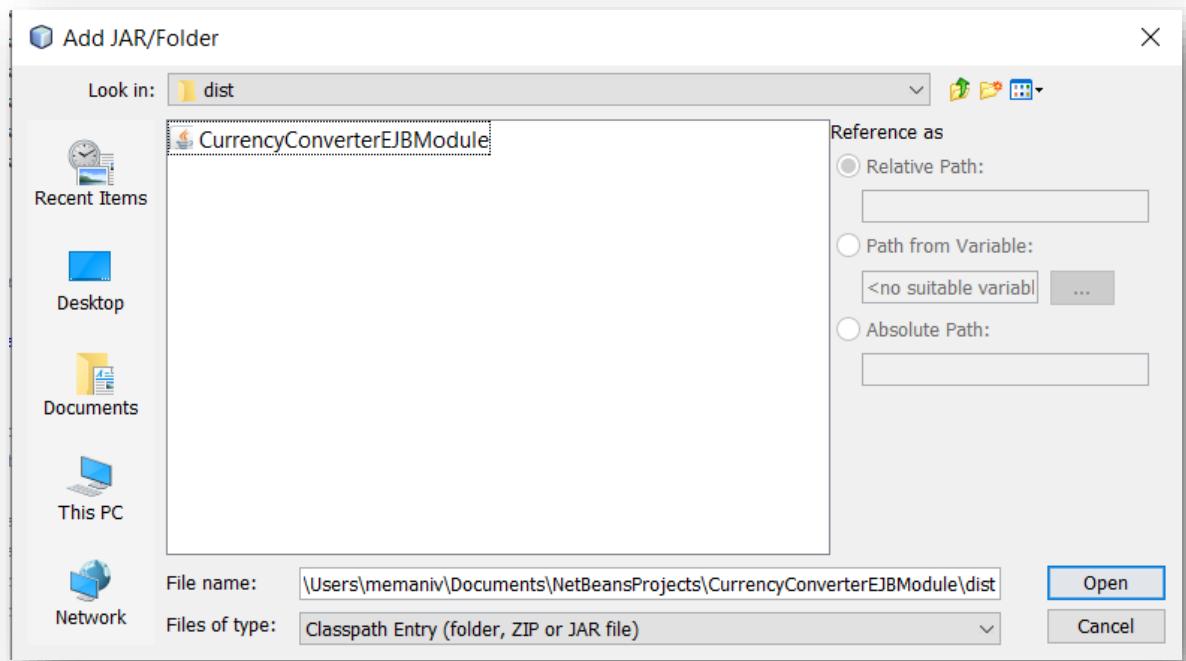
5. Create the dollar_to_rand.jsp file.



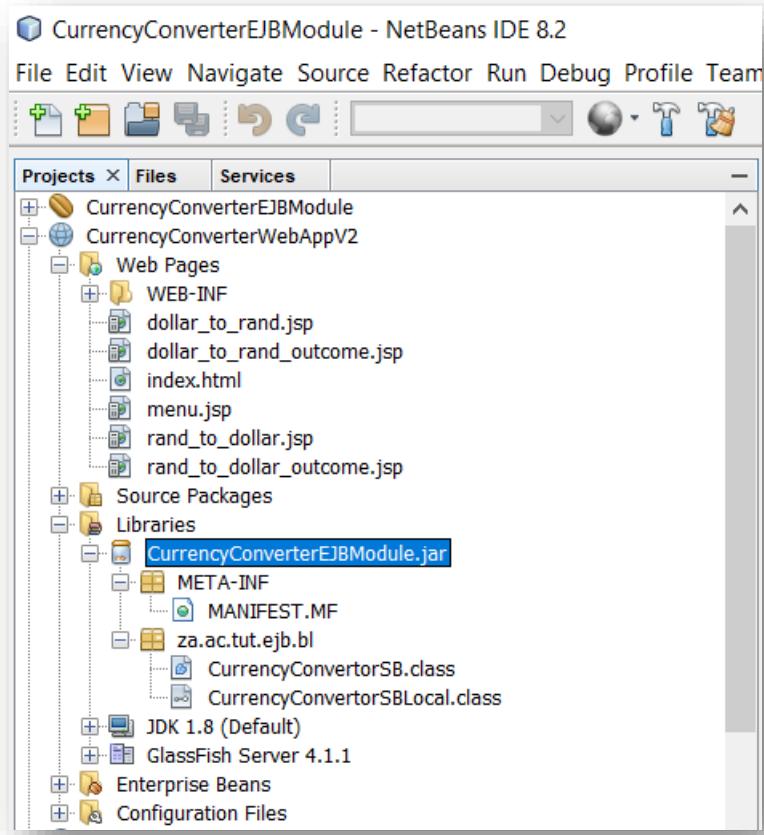
```
<%--  
2 Document : dollar_to_rand  
3 Created on : 19 Mar 2023, 5:41:32 PM  
4 Author : MemaniV  
5 --%>  
6  
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>  
8 <!DOCTYPE html>  
9 <html>  
10 <head>  
11     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
12     <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**.



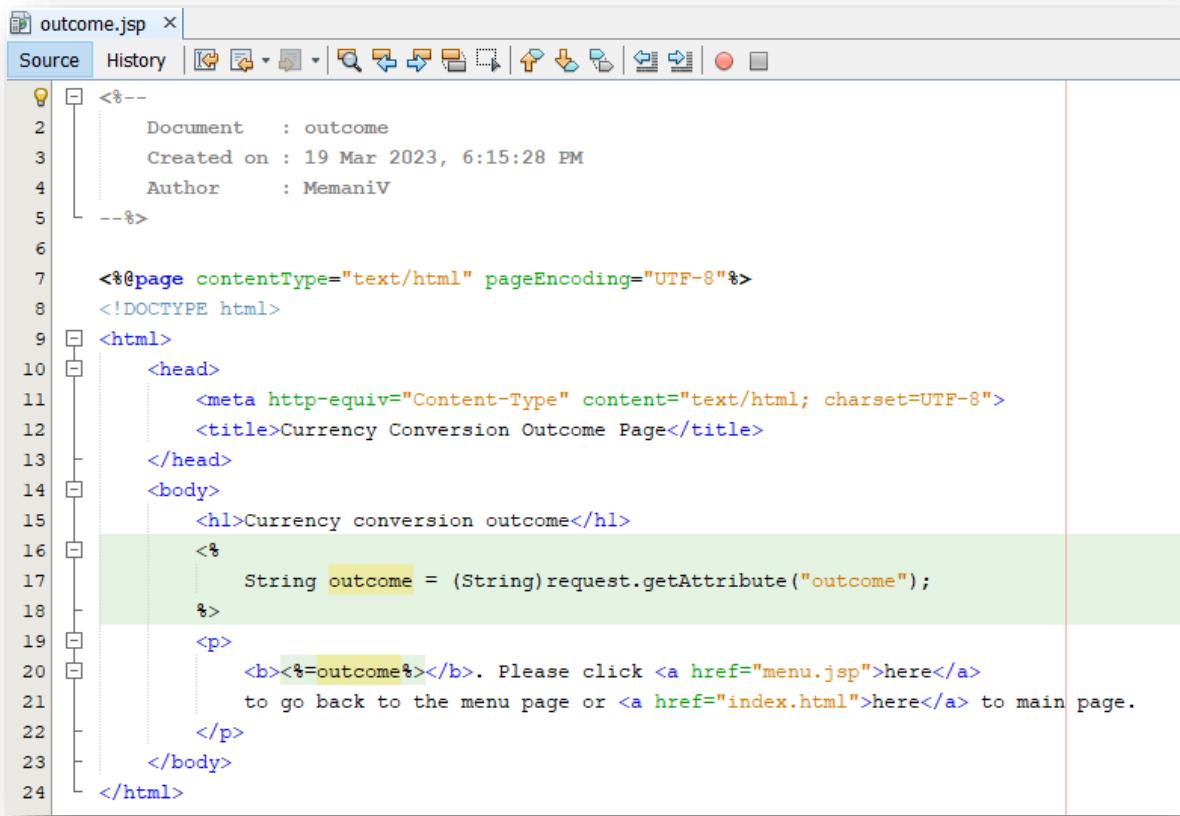
6.2 Select the jar file and click **Open**.



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 $" + 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.



The screenshot shows the NetBeans IDE interface with the 'outcome.jsp' file open in the editor. The code is a JSP page that displays a currency conversion outcome. It includes a header with document information, an HTML structure with a title, and a body containing a heading and a paragraph with a message. A line of code is highlighted in green, showing the retrieval of the 'outcome' attribute from the request.

```
<%--  
1 Document : outcome  
2 Created on : 19 Mar 2023, 6:15:28 PM  
3 Author : MemaniV  
4 --%>  
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>Currency Conversion Outcome Page</title>  
13 </head>  
14 <body>  
15     <h1>Currency conversion outcome</h1>  
16     <%  
17         String outcome = (String)request.getAttribute("outcome");  
18     %>  
19     <p>  
20         <b><%=outcome%></b>. Please click <a href="menu.jsp">here</a>  
21         to go back to the menu page or <a href="index.html">here</a> to main page.  
22     </p>  
23 </body>  
24 </html>
```

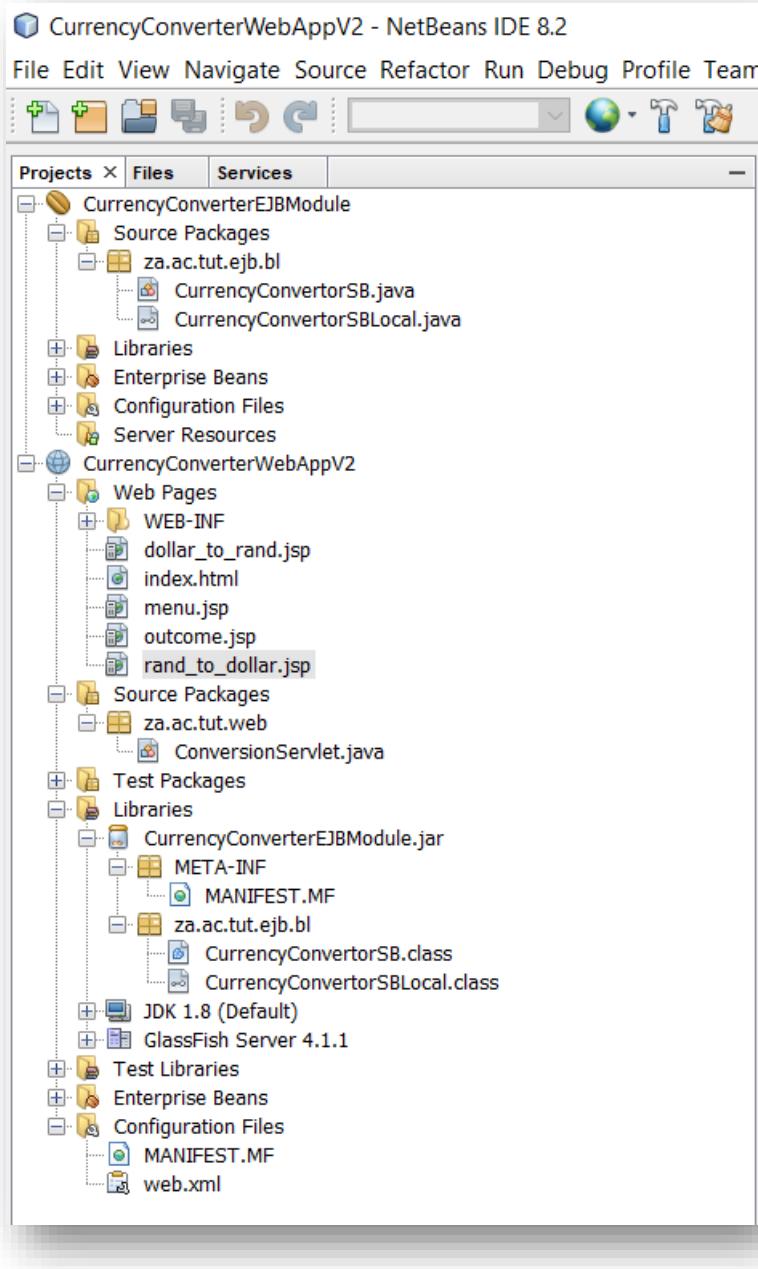
11. Compile the web project.



The screenshot shows the 'Output' window in NetBeans IDE for the 'CurrencyConverterWebAppV2' project. The window displays the build log, which shows the compilation process, including the creation of directories, copying of files, and the final building of a WAR file named 'CurrencyConverterWebAppV2.war'. The log concludes with a 'BUILD SUCCESSFUL' message.

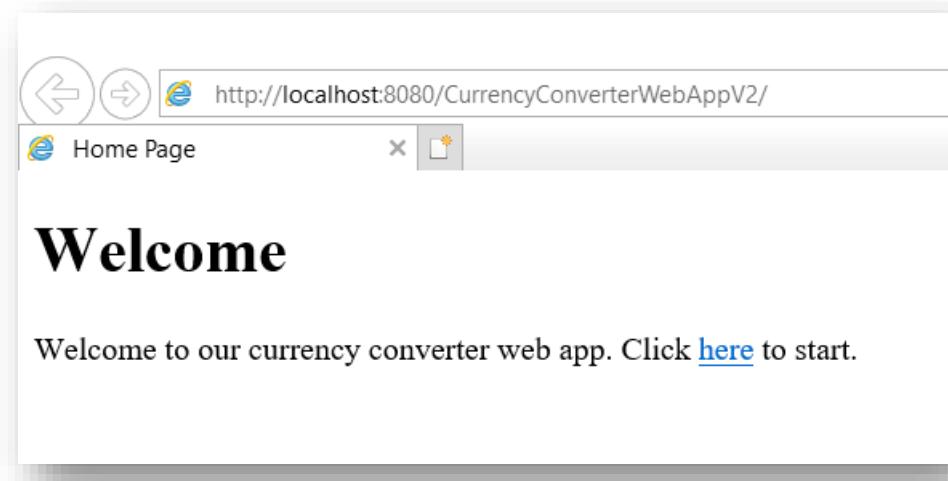
```
deps-maven-jar:  
deps-ear-jar:  
deps-jar:  
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConverterWebAppV2\build\web\WEB-INF\classes  
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConverterWebAppV2\build\web\META-INF  
Copying 1 file to C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConverterWebAppV2\build\web\META-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  
do-dist:  
dist:  
BUILD SUCCESSFUL (total time: 0 seconds)
```

12. View the entire structure of the web project.

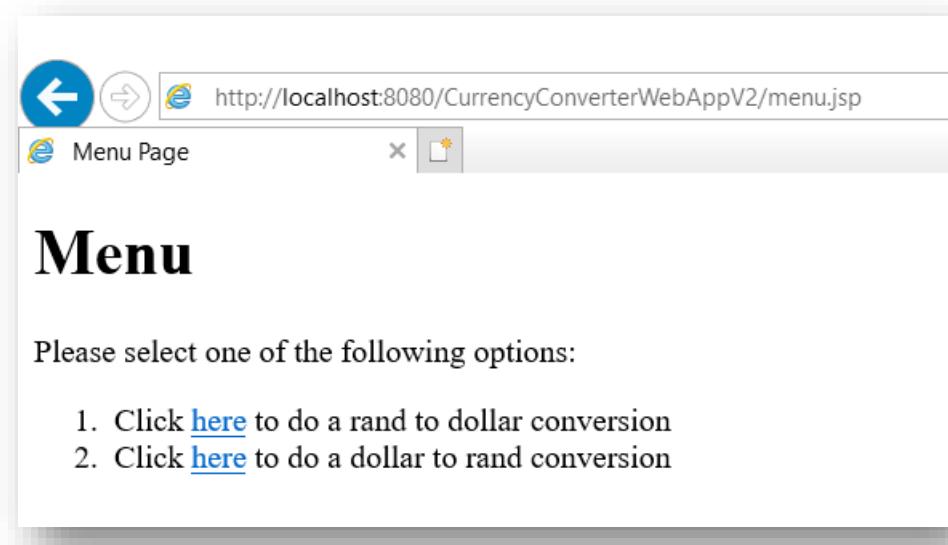


Part C - Run the web client project.

Launch the web app.



Click on the link.



1. Click [here](#) to do a rand to dollar conversion
2. Click [here](#) to do a dollar to rand conversion

Click on the first link.

The screenshot shows a web browser window with the URL http://localhost:8080/CurrencyConverterWebAppV2/rand_to_dollar.jsp. The title bar says "Rand to Dollar Conversion ...". The main content area has a large heading "Rand amount". Below it is a text input field labeled "Rand amount:" followed by a "CONVERT" button.

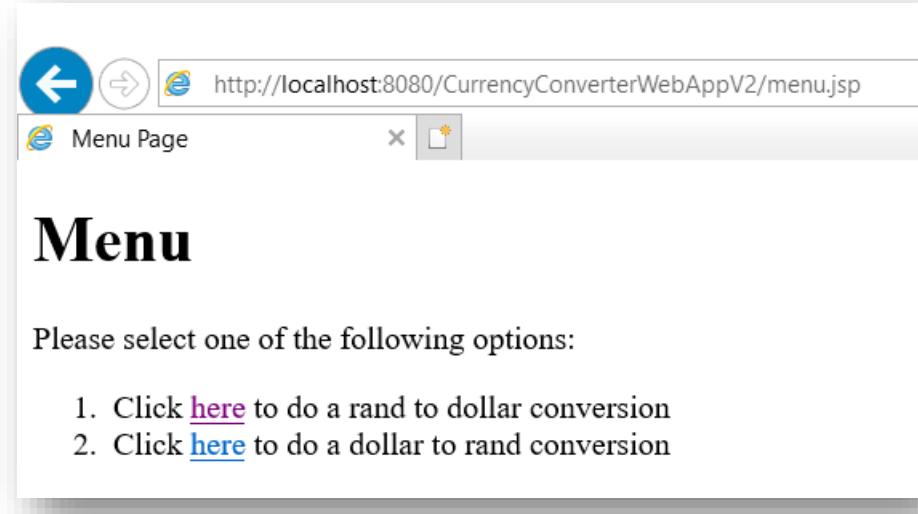
Enter the amount.

The screenshot shows the same web browser window as before, but now the "Rand amount" input field contains the value "100". The "CONVERT" button is visible below the input field.

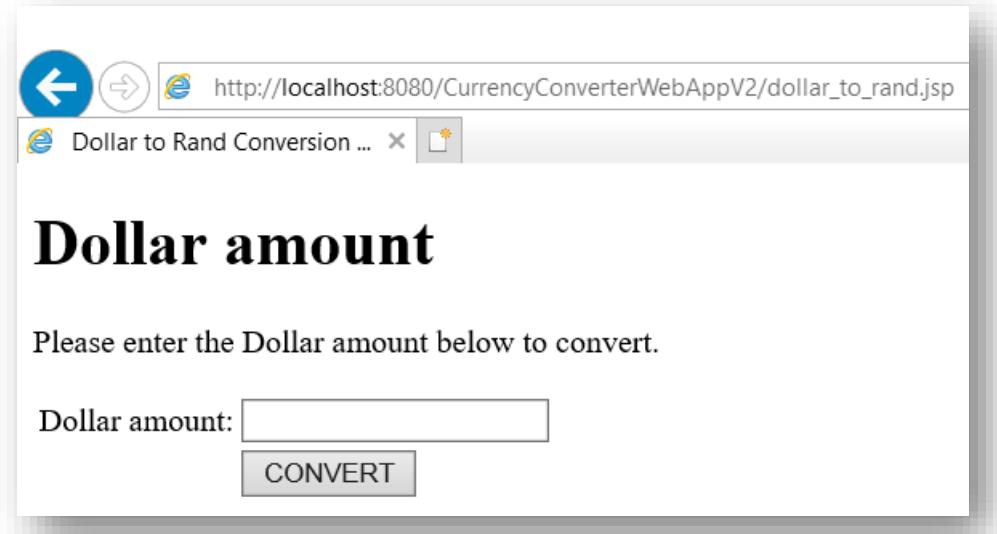
Click on the button.

The screenshot shows a web browser window with the URL <http://localhost:8080/CurrencyConverterWebAppV2/ConversionServlet.do>. The title bar says "Currency Conversion Outco...". The main content area has a large heading "Currency conversion outcome". Below it is a message: "R100.0 is equivalent to \$5.0. Please click [here](#) to go back to the menu page or [here](#) to main page."

Go back to the menu page.



Click on the second link.



Enter the dollar amount

The screenshot shows a web browser window with the URL http://localhost:8080/CurrencyConverterWebAppV2/dollar_to_rand.jsp. The title bar says "Dollar to Rand Conversion ...". The main content area has a large bold heading "Dollar amount". Below it is a text input field containing "10" with the placeholder "Dollar amount:". Underneath the input field is a "CONVERT" button.

Click on the button.

The screenshot shows a web browser window with the URL <http://localhost:8080/CurrencyConverterWebAppV2/ConversionServlet.do>. The title bar says "Currency Conversion Outco...". The main content area has a large bold heading "Currency conversion outcome". Below it is a message: "\$10.0 is equivalent to R200.0. Please click [here](#) to go back to the menu page or [here](#) to main page."

7.2.2 Stateful Session Bean

A Stateful Session Bean keeps track of the method calls between a client and the bean. This means that the bean is able to keep the state of previous calls to its methods. Stateful session beans are useful in tasks that require to be done in several steps, with each step dependent on the results of the previous step(s). A good example is the shopping cart in an online shopping web site. When a client adds items to a cart, their details must be kept at all times. When items are removed or changed, nothing must be lost because of subsequent operations or calls. So the state of the cart must be maintained at all material times.

The container creates an instance of the bean for each new client during an active session. So there's no resource sharing, each client is allocated its own bean by the container. This is an expensive exercise in terms of memory usage. To avoid the presence of so many objects in memory (million users would require a million beans) the container manages the beans through a process of **passivation** and **activation**.

Passivation is the process of temporarily removing a bean from memory and storing it in a persistent object such as a database or file. This is done by the container to free memory for a short time between client calls. The bean is returned back to memory when a client makes a subsequent call to the bean. This reverse process is known as activation.

Lifecycle of stateful session beans

Stateful session beans have three lifecycle states, namely:

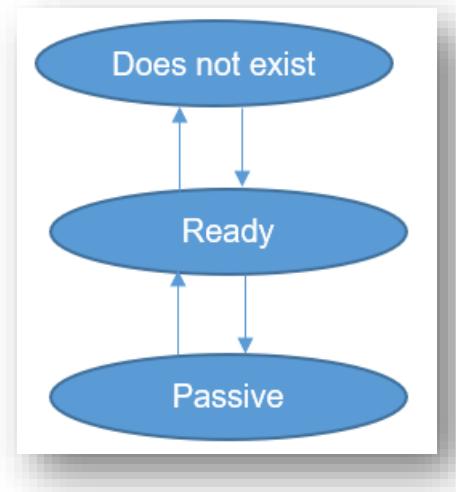
- Does not exist;
- Ready; and
- Passive.

So at all times, stateful session bean would move between the mentioned stages. The figure below shows the states. The bean is brought to life by the container. During construction, dependency injection and post construct methods are executed (@PostConstruct).

At the Ready stage the bean is ready to service client requests. When inactive, the container can move a bean to the Passive state. There's a @PrePassivate annotation that a programmer can annotate a method with. This is code that a

container will have to execute before taking the bean to the Passive state. When a client makes a call for the bean, the container would run first any method annotated with the `@PostActivate` annotation before moving the bean the Ready state.

The client can initiate the destruction of a bean by calling a method annotated with the `@Remove` annotation. The container will then call the `@PreDestroy` annotated method, if any, and put up the bean garbage collection. The figure below shows the explained lifecycle stages of a stateful session bean.



Example

In this example we are going to create a web application that uses a Stateful Session Bean to manage items. An item has an ID, description and a price. The envisaged web application must perform the following functionalities:

- Add items to a list.
- Remove items from the list.
- Get the numbers of items currently in the list.
- Get the list of items in the list.

Solution approach

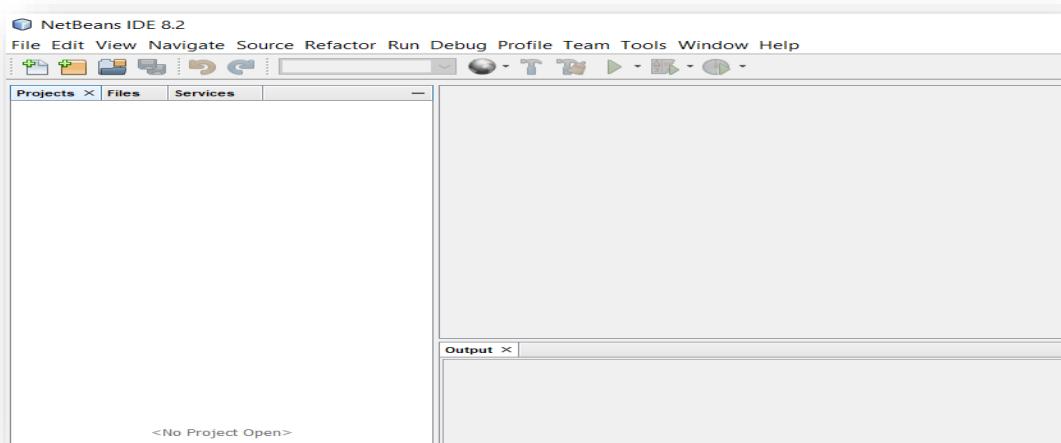
The solution to this problem is divided into three parts, Part A, B and C. In part A we will create an EJB module project with a stateful session bean. The stateful session bean will implement all the needed methods.

In Part B we will create a web client project. The web client will mainly have one servlet to manage the requests to the EJB. In Part C we will run the web application through the browser.

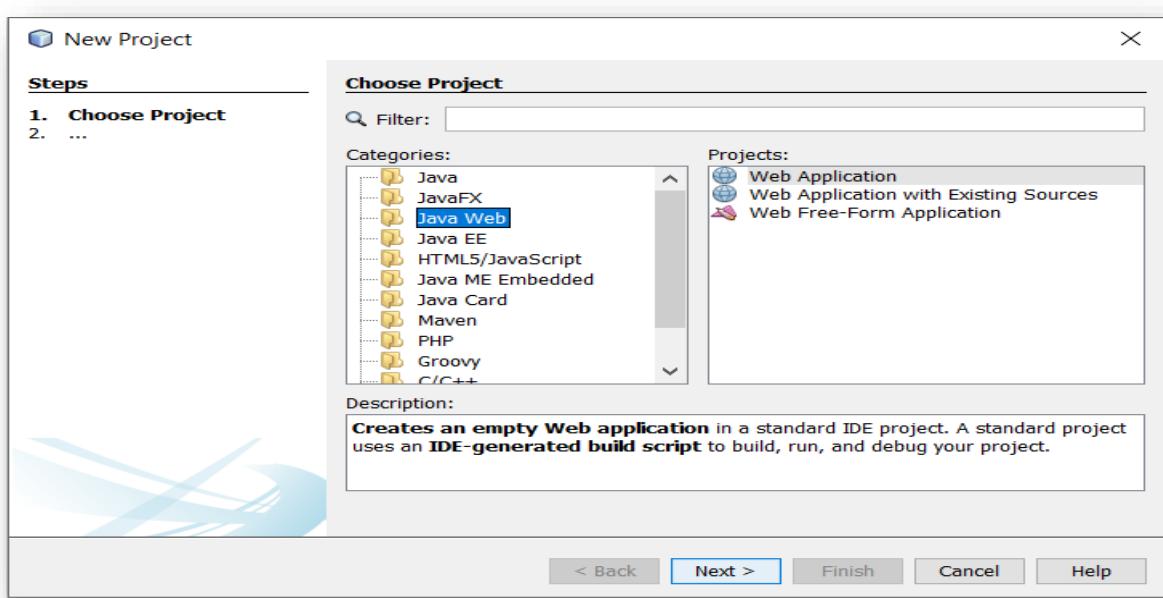
Part A - Create an EJB module project.

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

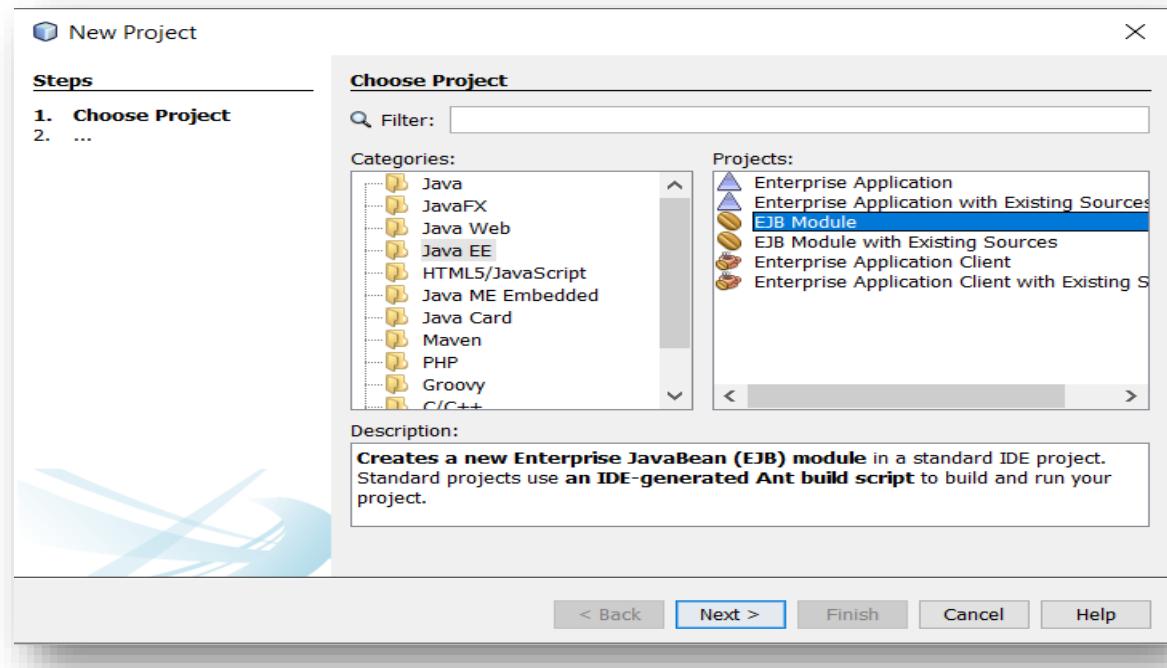
1. Launch NetBeans.



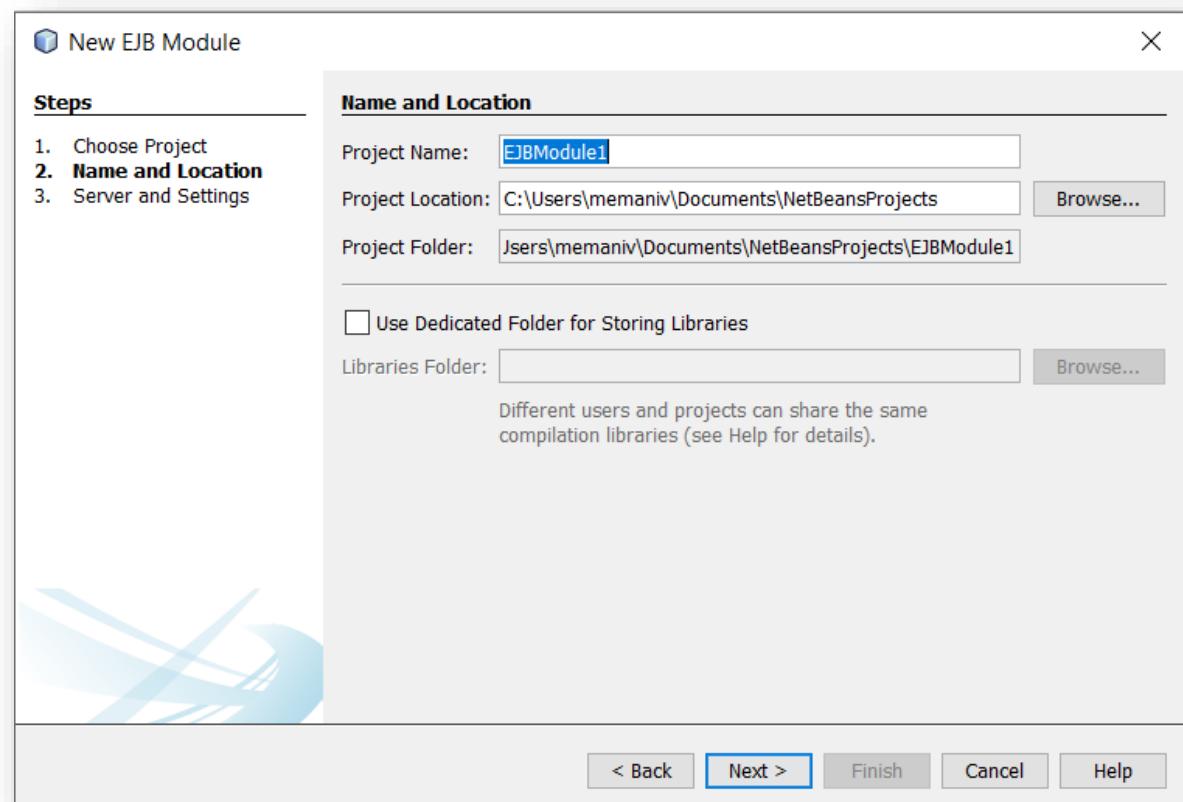
2. Click on **File | New Project**.



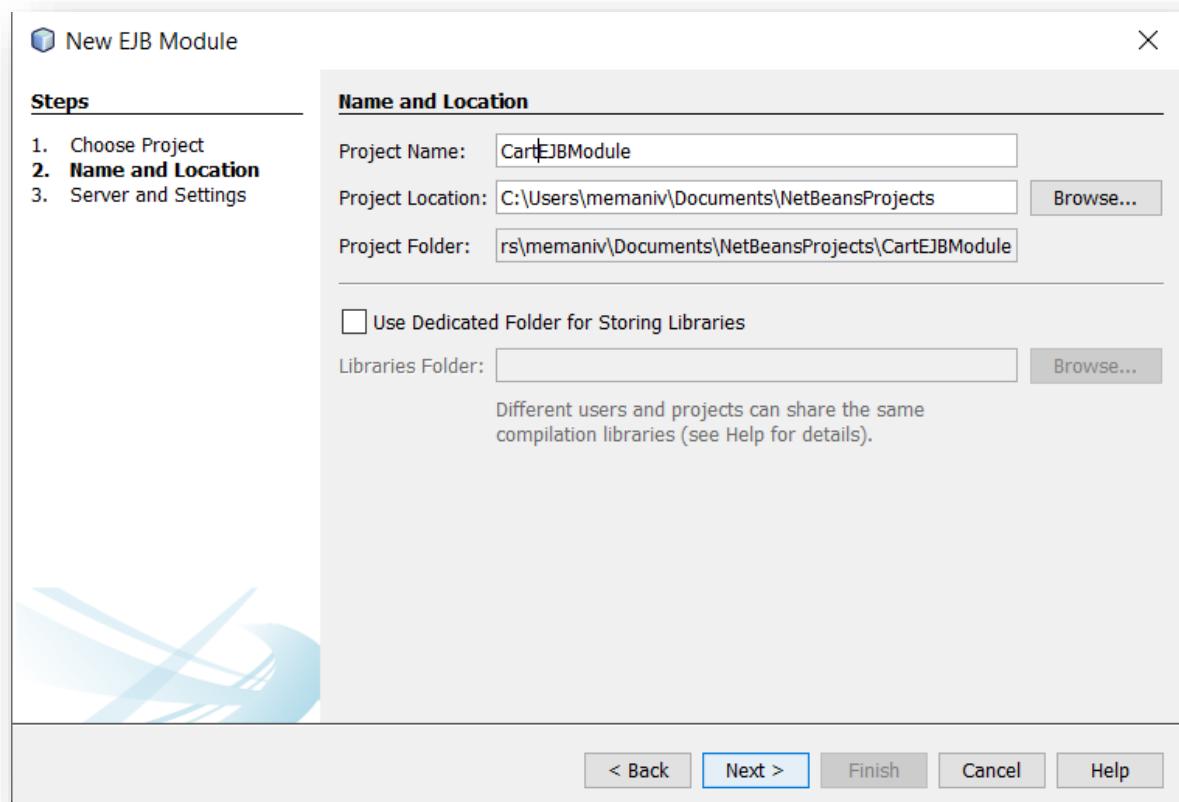
3. Select Java EE under Categories and EJB Module under Projects.



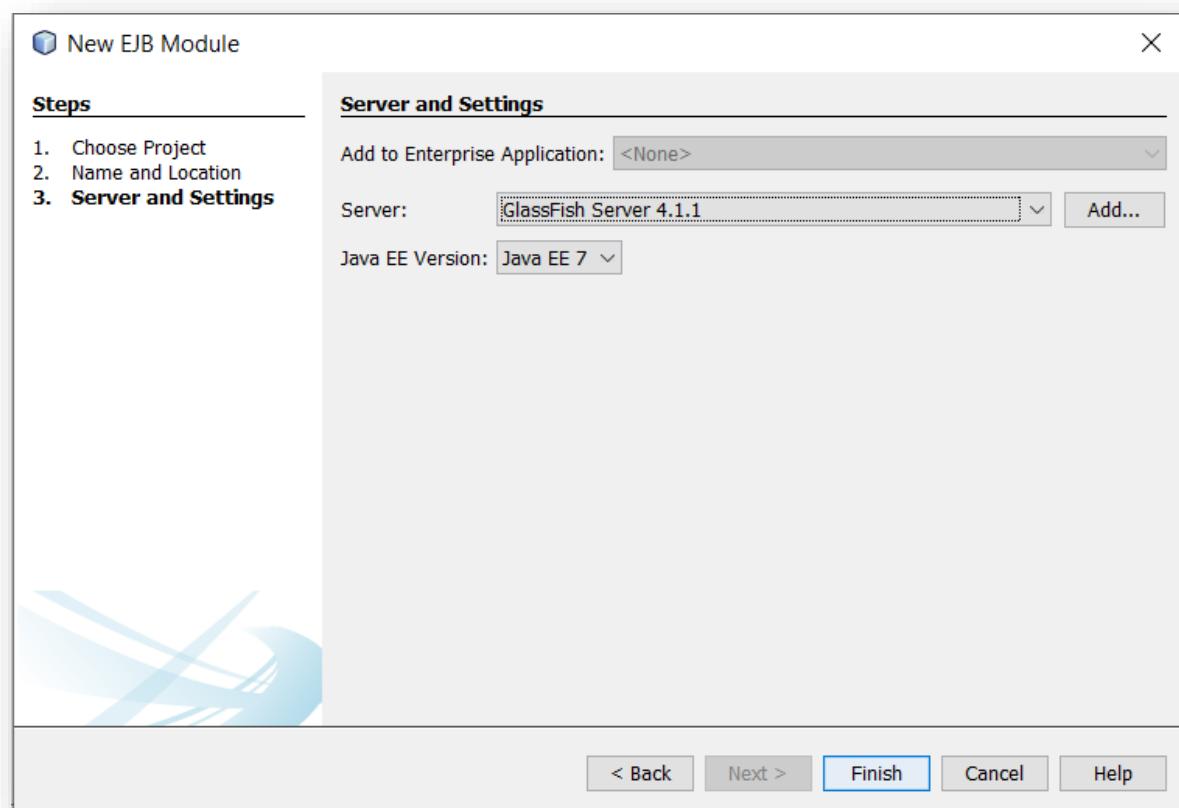
4. Click **Next.**



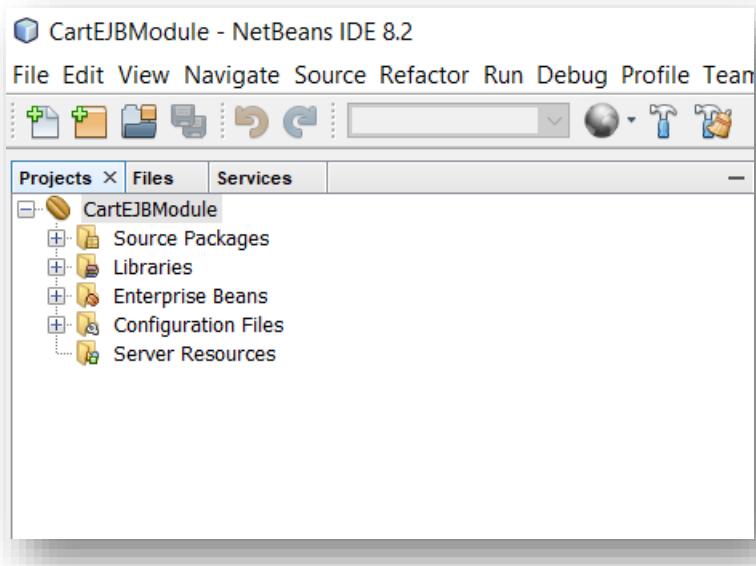
5. Name the project as **CartEJBModule**.



6. Click **Next**.

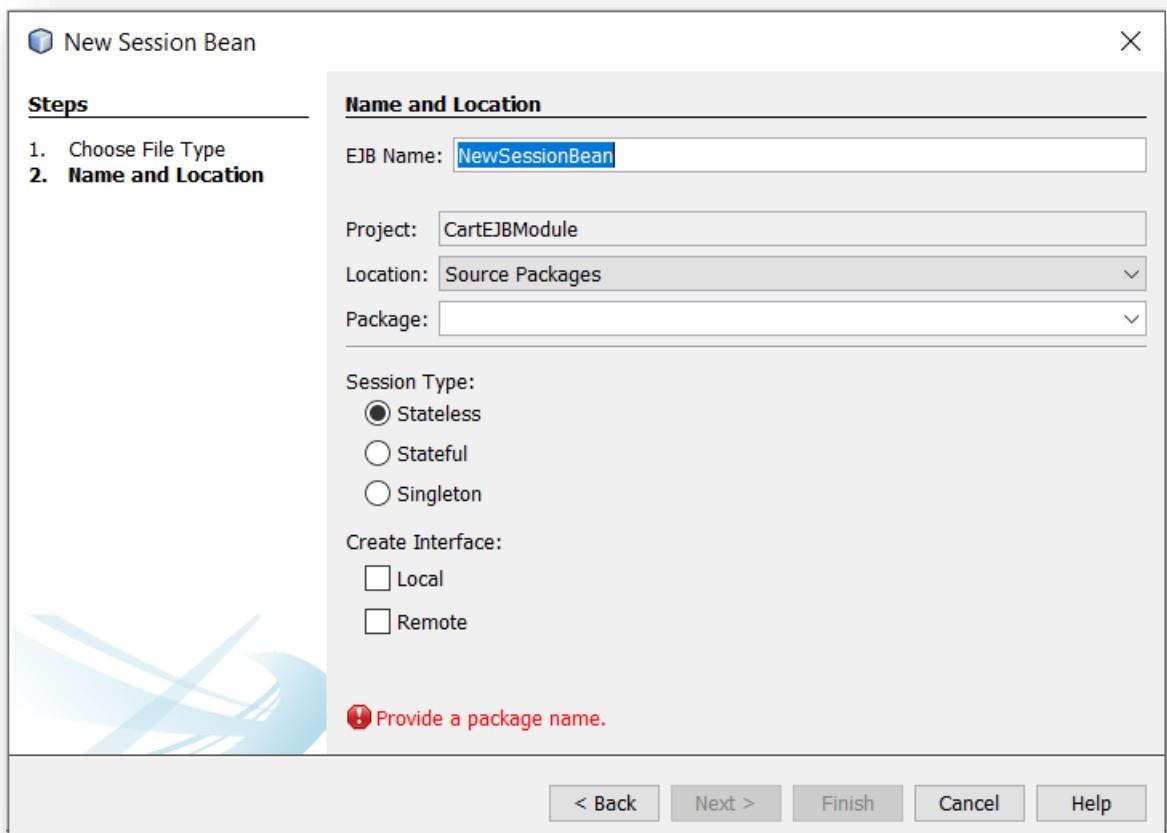


7. Click **Finish**.

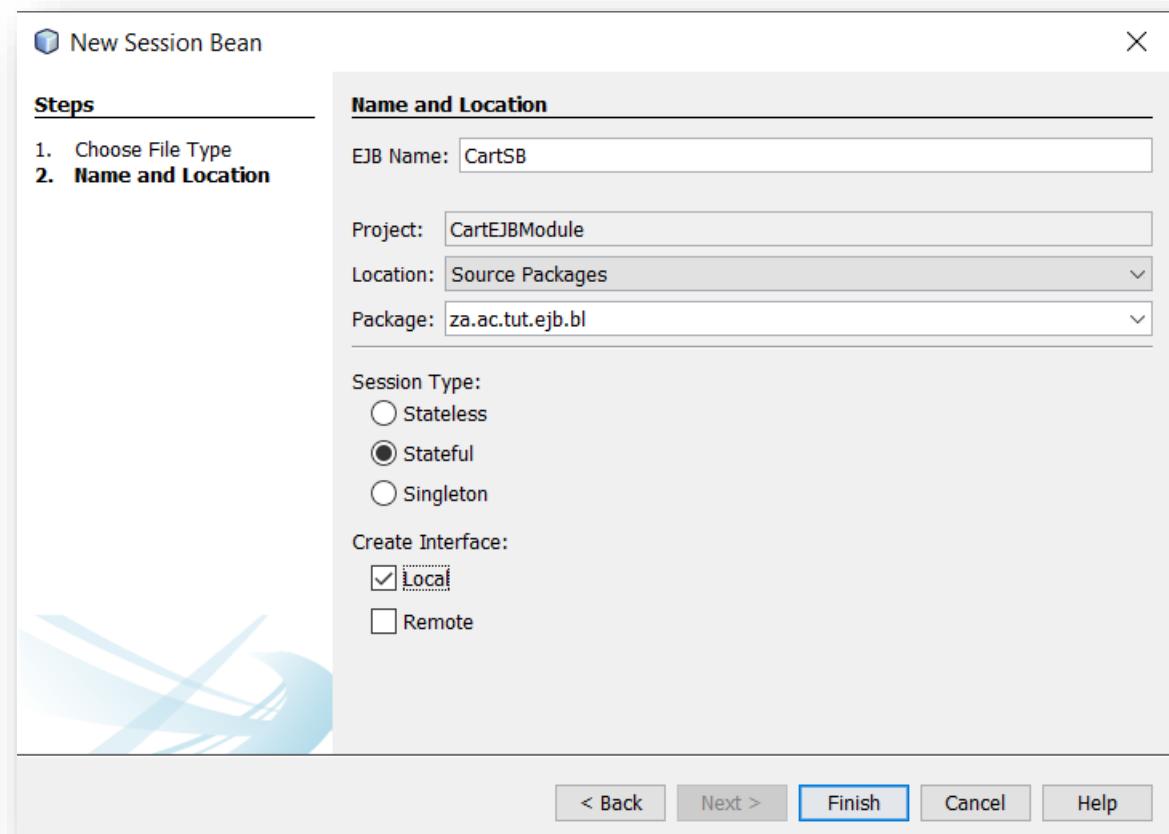


8. Create a stateful session bean. Perform the following steps:

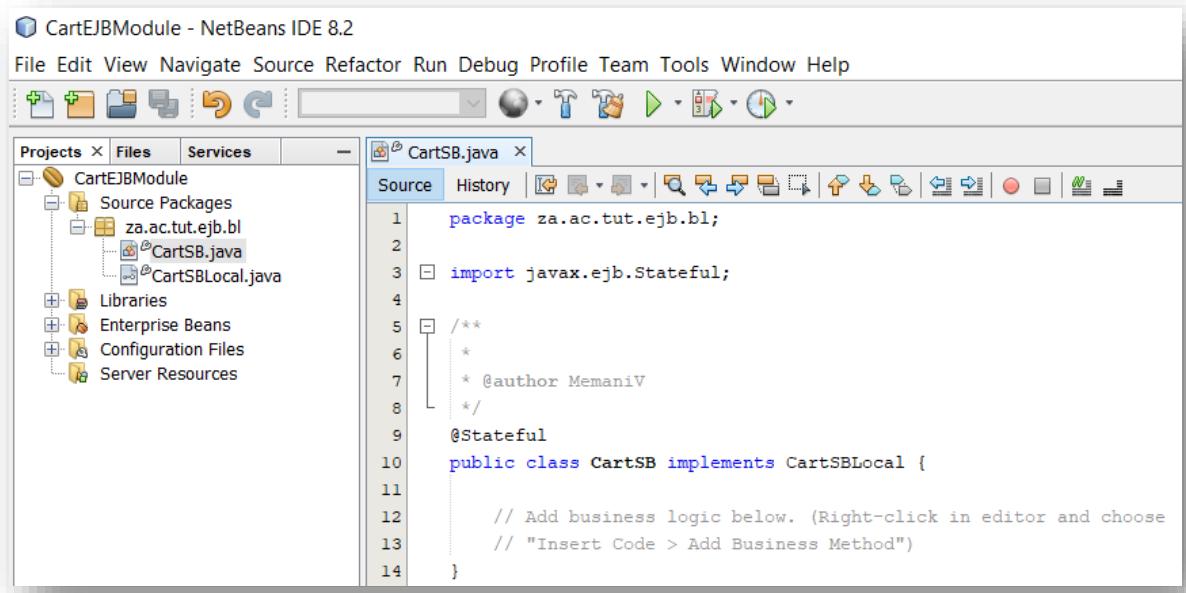
8.1 Right click on the project and select **New | Session Bean**.



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



8.3 Click **Finish**.



9. Create a POJO (Plain Old Java Object) called Item.

The screenshot shows a Java code editor with the file `Item.java` open. The code defines a simple POJO named `Item` with fields `id`, `description`, and `price`, along with their corresponding getters and setters. The code is annotated with Javadoc-style comments and includes an overridden `toString` method.

```
1 package za.ac.tut.model;
2
3 /**
4 * @author MemaniV
5 */
6
7 public class Item {
8     private Integer id;
9     private String description;
10    private Double price;
11
12    public Item(Integer id, String description, Double price) {
13        this.id = id;
14        this.description = description;
15        this.price = price;
16    }
17
18    public Integer getId() {
19        return id;
20    }
21
22    public void setId(Integer id) {
23        this.id = id;
24    }
25
26    public String getDescription() {
27        return description;
28    }
29
30    public void setDescription(String description) {
31        this.description = description;
32    }
33
34    public Double getPrice() {
35        return price;
36    }
37
38    public void setPrice(Double price) {
39        this.price = price;
40    }
41
42    @Override
43    public String toString() {
44        return "(" + getId() + "," + getDescription() + "," + getPrice() + ")";
45    }
46
47}
```

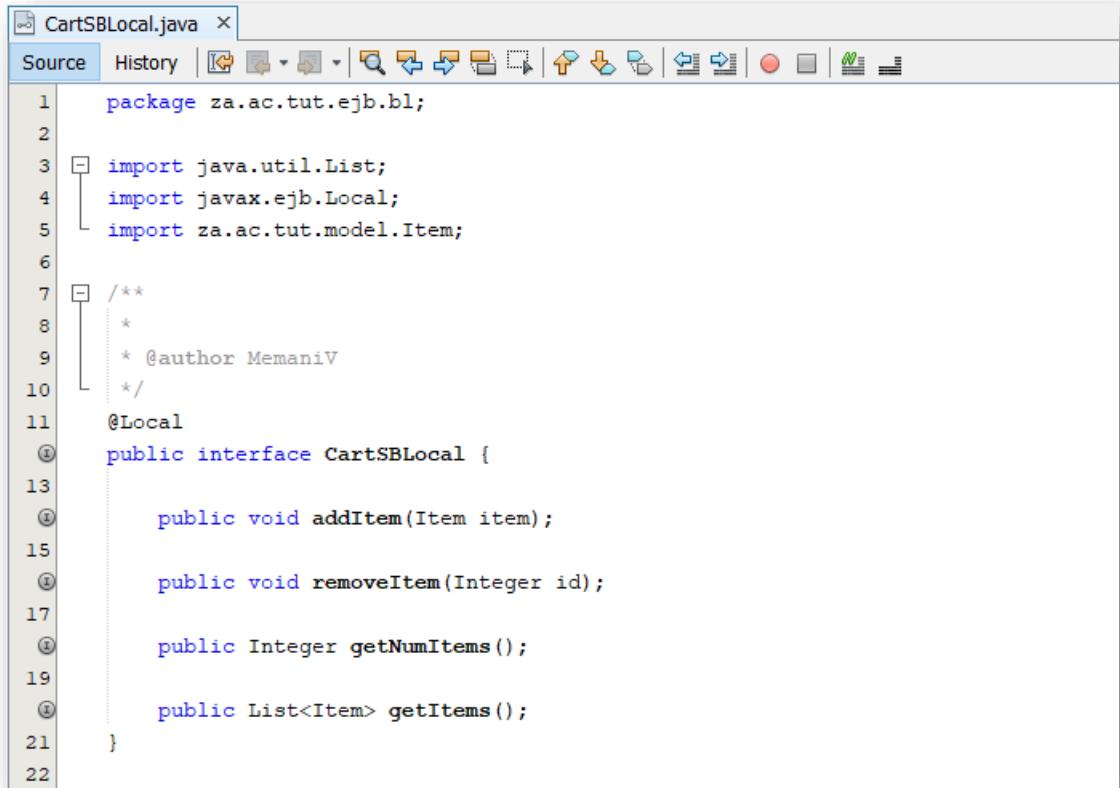
10. Add business methods to the Stateful Bean.



The screenshot shows a Java code editor with the file `CartSB.java` open. The code defines a stateful bean named `CartSB` that implements the `CartSBLocal` interface. The bean maintains a list of items in a `cart` field. It provides methods to add items, remove items by ID, get the number of items, and get the list of items.

```
1 package za.ac.tut.ejb.bl;
2
3 import java.util.ArrayList;
4 import java.util.List;
5 import javax.ejb.Stateful;
6 import za.ac.tut.model.Item;
7
8 /**
9 * @author MemaniV
10 */
11 @Stateful
12 public class CartSB implements CartSBLocal {
13     private List<Item> cart = new ArrayList<>();
14
15     @Override
16     public void addItem(Item item) {
17         cart.add(item);
18     }
19
20     @Override
21     public void removeItem(Integer id) {
22         for(Item item: cart){
23             if(item.getId().equals(id)){
24                 cart.remove(item);
25             }
26         }
27     }
28
29     @Override
30     public Integer getNumItems() {
31         return cart.size();
32     }
33
34     @Override
35     public List<Item> getItems() {
36         return cart;
37     }
38
39 }
40 }
```

11. View the interface.



The screenshot shows the NetBeans IDE interface with the file `CartSBLocal.java` open. The code defines a local EJB interface with methods for adding and removing items from a cart, and getting item counts and lists.

```
1 package za.ac.tut.ejb.bl;
2
3 import java.util.List;
4 import javax.ejb.Local;
5 import za.ac.tut.model.Item;
6
7 /**
8 *
9 * @author MemaniV
10 */
11 @Local
12 public interface CartSBLocal {
13
14     public void addItem(Item item);
15
16     public void removeItem(Integer id);
17
18     public Integer getNumItems();
19
20     public List<Item> getItems();
21 }
22
```

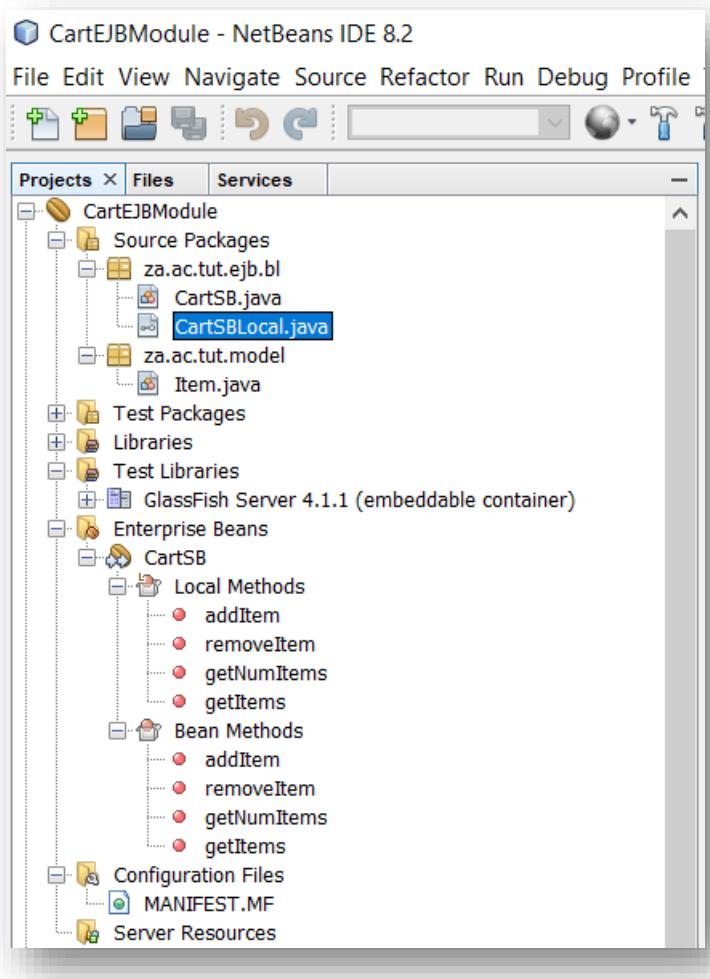
12. Compile the EJB project.



The screenshot shows the NetBeans IDE Output window for the `CartEJBModule` project. It displays the command-line output of the Ant build process, showing the creation of the `dist` jar file.

```
Output - CartEJBModule (clean,dist) ×
ant -f C:\\\\Users\\\\memaniv\\\\Documents\\\\NetBeansProjects\\\\CartEJBModule -Dnb.internal.action.name=rebuild -Duser.p
init:
undeploy-clean:
deps-clean:
clean:
init:
deps-jar:
Created dir: C:\\Users\\memaniv\\Documents\\NetBeansProjects\\CartEJBModule\\build\\classes
Copying 1 file to C:\\Users\\memaniv\\Documents\\NetBeansProjects\\CartEJBModule\\build\\classes\\META-INF
Created dir: C:\\Users\\memaniv\\Documents\\NetBeansProjects\\CartEJBModule\\build\\empty
Created dir: C:\\Users\\memaniv\\Documents\\NetBeansProjects\\CartEJBModule\\build\\generated-sources\\ap-source-output
Compiling 3 source files to C:\\Users\\memaniv\\Documents\\NetBeansProjects\\CartEJBModule\\build\\classes
compile:
library-inclusion-in-archive:
Created dir: C:\\Users\\memaniv\\Documents\\NetBeansProjects\\CartEJBModule\\dist
Building jar: C:\\Users\\memaniv\\Documents\\NetBeansProjects\\CartEJBModule\\dist\\CartEJBModule.jar
dist:
BUILD SUCCESSFUL (total time: 0 seconds)
```

13. View the complete EJB project structure.



Part B - Create a Web client project

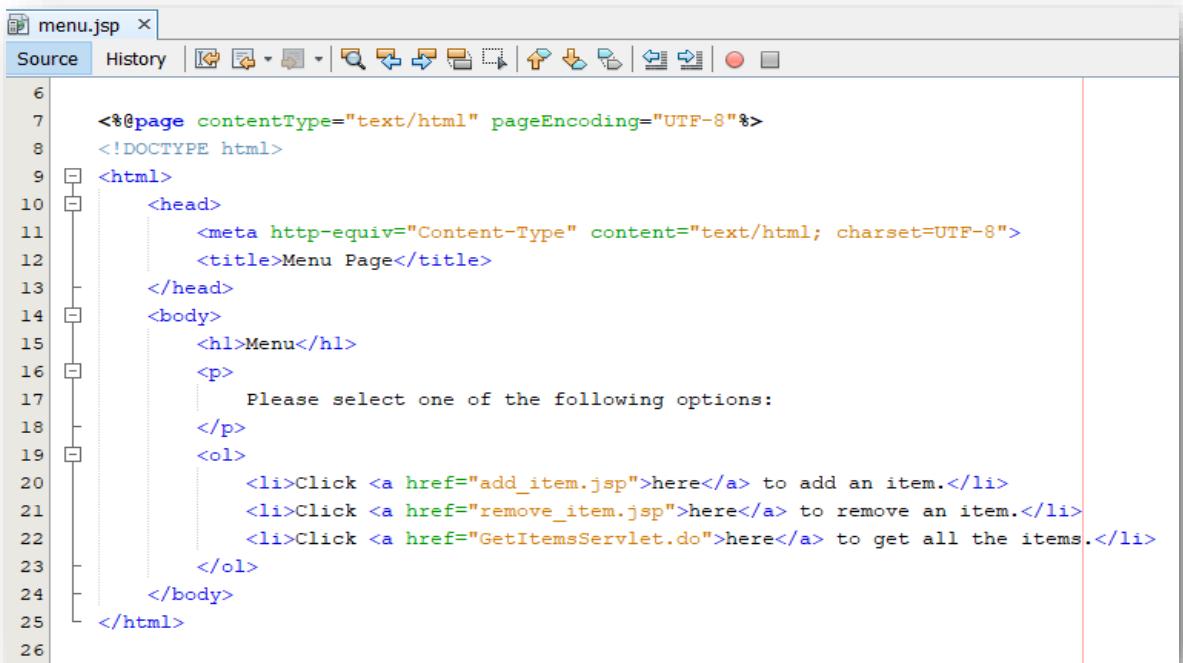
To successfully create a working web client project, perform the following steps:

1. Create a web project called **CartWebApp** and edit the **index** file.

The screenshot shows the NetBeans code editor for the file "index.html". The tab bar at the top shows "index.html x". The toolbar below the tab bar includes icons for Source, History, and various file operations. The code editor displays the following HTML and XML content:

```
<!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>
```

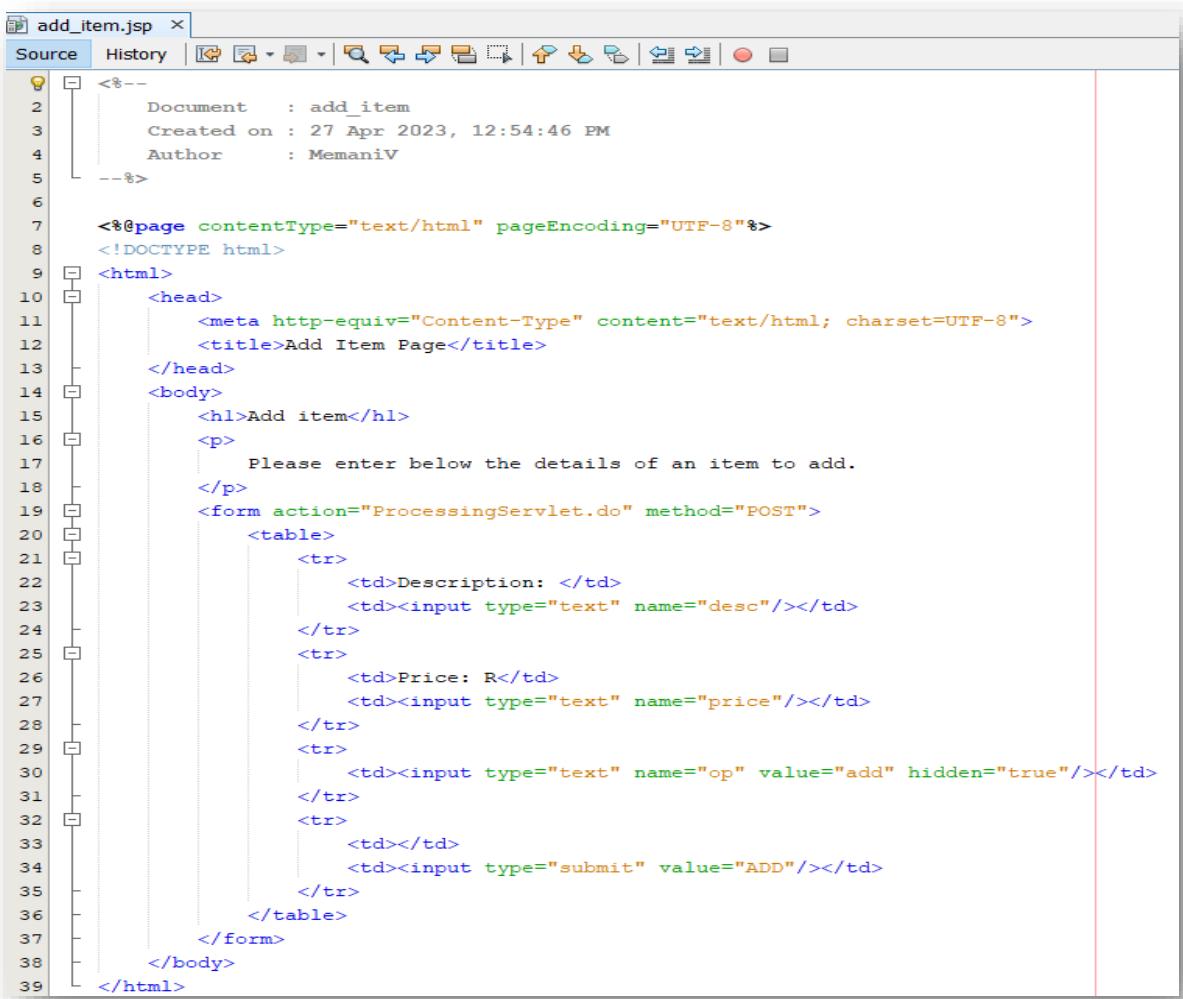
2. Create the menu.jsp file.



The screenshot shows the menu.jsp file in a Java IDE. The code is a simple JSP page that displays a menu. It includes a title, a heading, a paragraph instructing the user to select an option, and a list of three links: 'add_item.jsp', 'remove_item.jsp', and 'GetItemsServlet.do'. The code is well-formatted with line numbers and color-coded syntax highlighting.

```
<%@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="add_item.jsp">here</a> to add an item.</li>
            <li>Click <a href="remove_item.jsp">here</a> to remove an item.</li>
            <li>Click <a href="GetItemsServlet.do">here</a> to get all the items.</li>
        </ol>
    </body>
</html>
```

3. Create the add_item.jsp file.

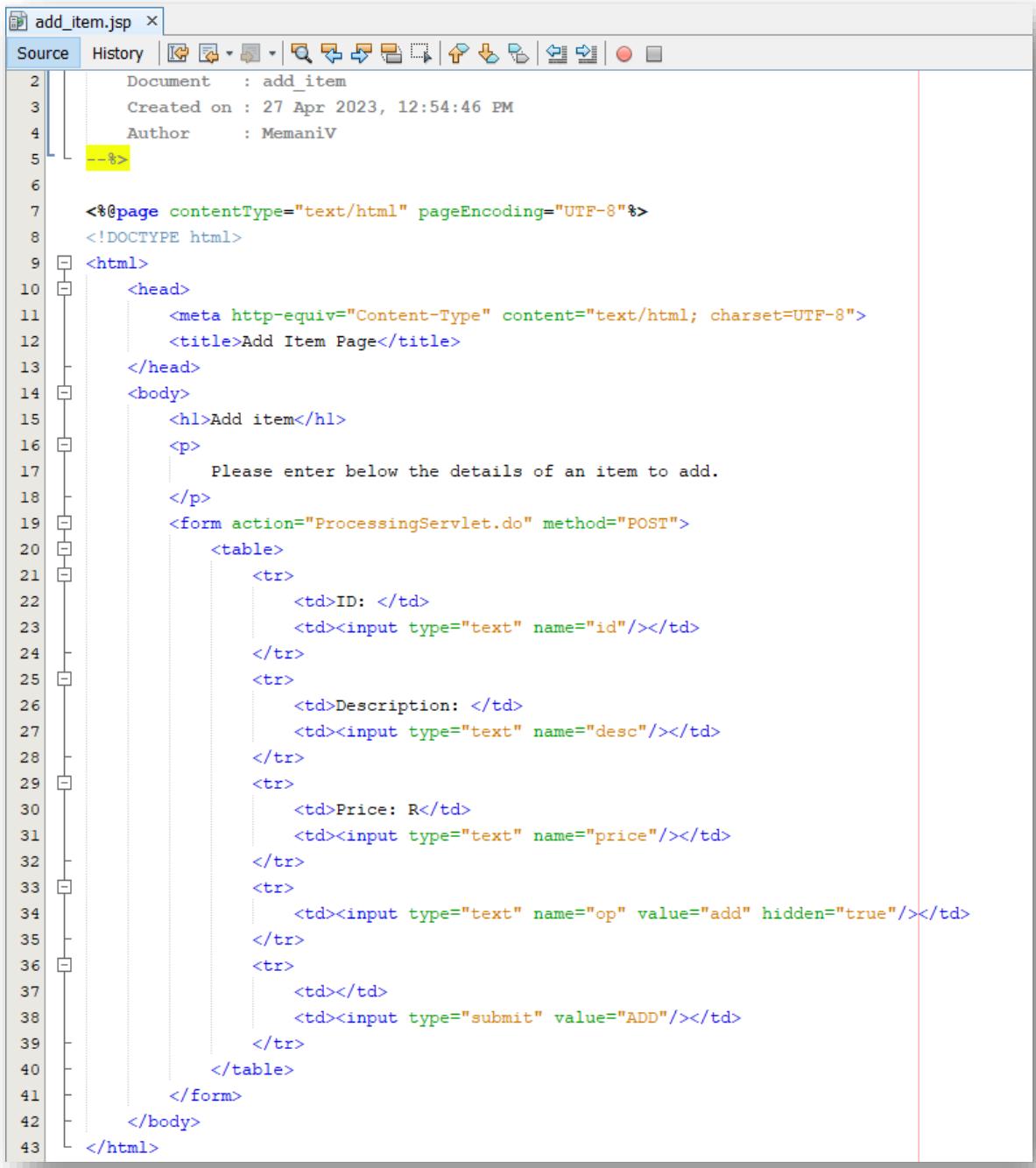


The screenshot shows the add_item.jsp file in a Java IDE. The code is a JSP page for adding an item. It includes a title, a heading, a paragraph asking for item details, and a form with a table containing fields for Description, Price, and a hidden 'op' field set to 'add'. The form has a POST method and points to 'ProcessingServlet.do'. The code is annotated with document information (Document: add_item, Created on: 27 Apr 2023, Author: MemaniV) and includes color-coded syntax highlighting.

```
<%-->
Document      : add_item
Created on   : 27 Apr 2023, 12:54:46 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>Add Item Page</title>
    </head>
    <body>
        <h1>Add item</h1>
        <p>
            Please enter below the details of an item to add.
        </p>
        <form action="ProcessingServlet.do" method="POST">
            <table>
                <tr>
                    <td>Description:</td>
                    <td><input type="text" name="desc"/></td>
                </tr>
                <tr>
                    <td>Price:</td>
                    <td><input type="text" name="price"/></td>
                </tr>
                <tr>
                    <td><input type="text" name="op" value="add" hidden="true"/></td>
                </tr>
                <tr>
                    <td></td>
                    <td><input type="submit" value="ADD"/></td>
                </tr>
            </table>
        </form>
    </body>
</html>
```

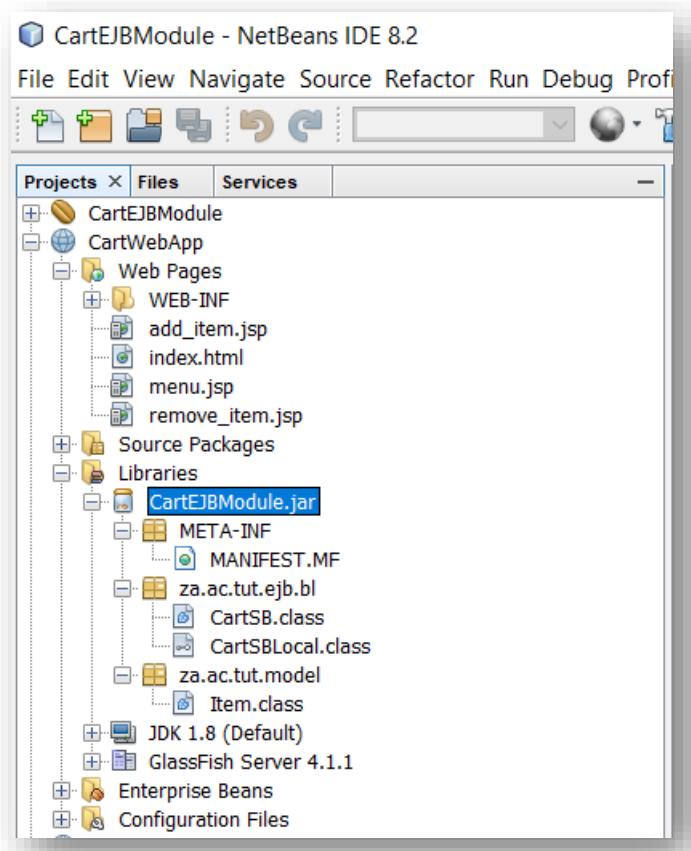
4. Create the **remove_item.jsp** file.



```
add_item.jsp
Source History |  |         |  
```

```
2 Document : add_item
3 Created on : 27 Apr 2023, 12:54:46 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>Add Item Page</title>
13 </head>
14 <body>
15     <h1>Add item</h1>
16     <p>
17         Please enter below the details of an item to add.
18     </p>
19     <form action="ProcessingServlet.do" method="POST">
20         <table>
21             <tr>
22                 <td>ID: </td>
23                 <td><input type="text" name="id"/></td>
24             </tr>
25             <tr>
26                 <td>Description: </td>
27                 <td><input type="text" name="desc"/></td>
28             </tr>
29             <tr>
30                 <td>Price: R</td>
31                 <td><input type="text" name="price"/></td>
32             </tr>
33             <tr>
34                 <td><input type="text" name="op" value="add" hidden="true"/></td>
35             </tr>
36             <tr>
37                 <td></td>
38                 <td><input type="submit" value="ADD"/></td>
39             </tr>
40         </table>
41     </form>
42 </body>
43 </html>
```

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

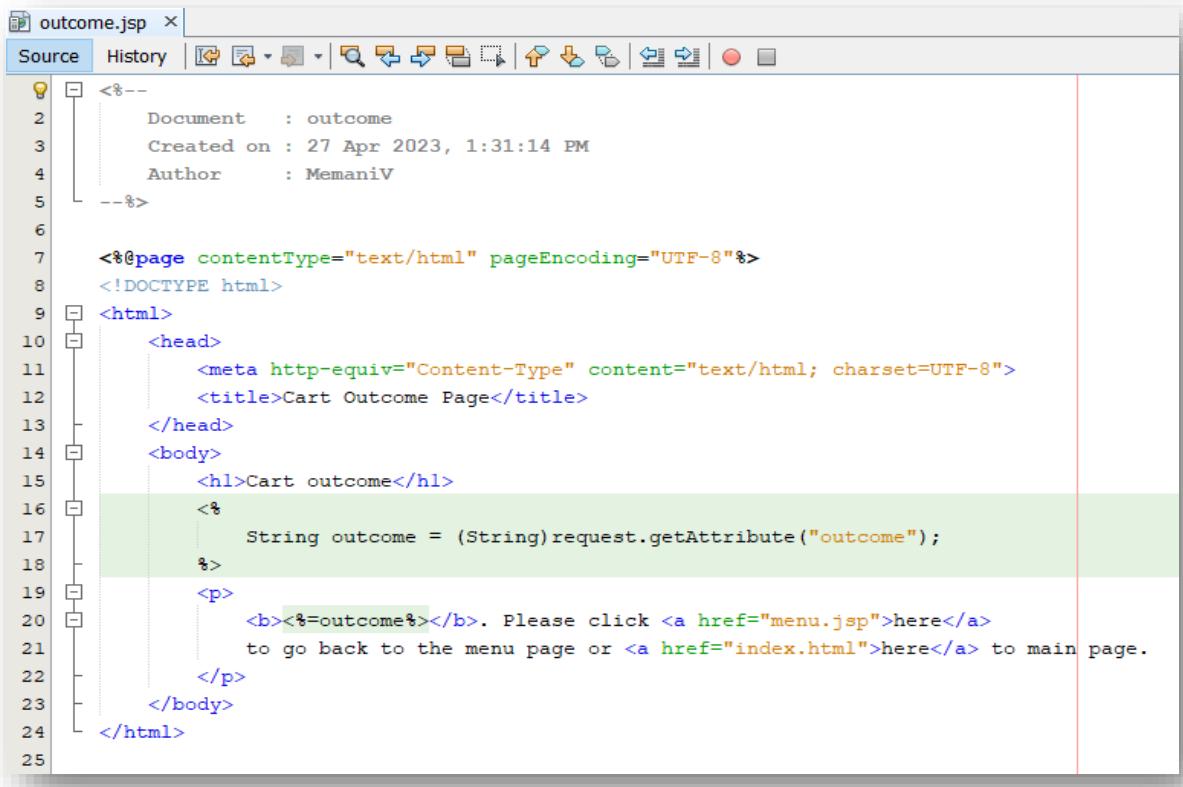


6. Create the **ProcessingServlet.java** file.

```
17 public class ProcessingServlet extends HttpServlet {
18     @EJB
19     private CartSBLocal csl;
20
21     @Override
22     protected void doPost(HttpServletRequest request, HttpServletResponse response)
23             throws ServletException, IOException {
24         String op = request.getParameter("op");
25         String url = "outcome.jsp";
26         String outcome;
27
28         if(op.equals("add")){
29             //add item
30             Integer id = Integer.parseInt(request.getParameter("id"));
31             String description = request.getParameter("desc");
32             Double price = Double.parseDouble(request.getParameter("price"));
33
34             Item item = createItem(id, description, price);
35             csl.addItem(item);
36             outcome = "The item has been added.";
37             request.setAttribute("outcome", outcome);
38         } else if(op.equals("remove")){
39             //remove item
40             Integer id = Integer.parseInt(request.getParameter("id"));
41             csl.removeItem(id);
42             outcome = "The item has been successfully removed.";
43             request.setAttribute("outcome", outcome);
44         } else {
45             //get all the items
46             List<Item> items = csl.getItems();
47             Integer numItems = csl.getNumItems();
48             request.setAttribute("items", items);
49             request.setAttribute("numItems", numItems);
50             url = "items.jsp";
51         }
52
53         RequestDispatcher disp = request.getRequestDispatcher(url);
54         disp.forward(request, response);
55     }
}
```

```
57     private Item createItem(Integer id, String description, Double price) {
58         Item item = new Item(id, description, price);
59         return item;
60     }
61 }
62 }
```

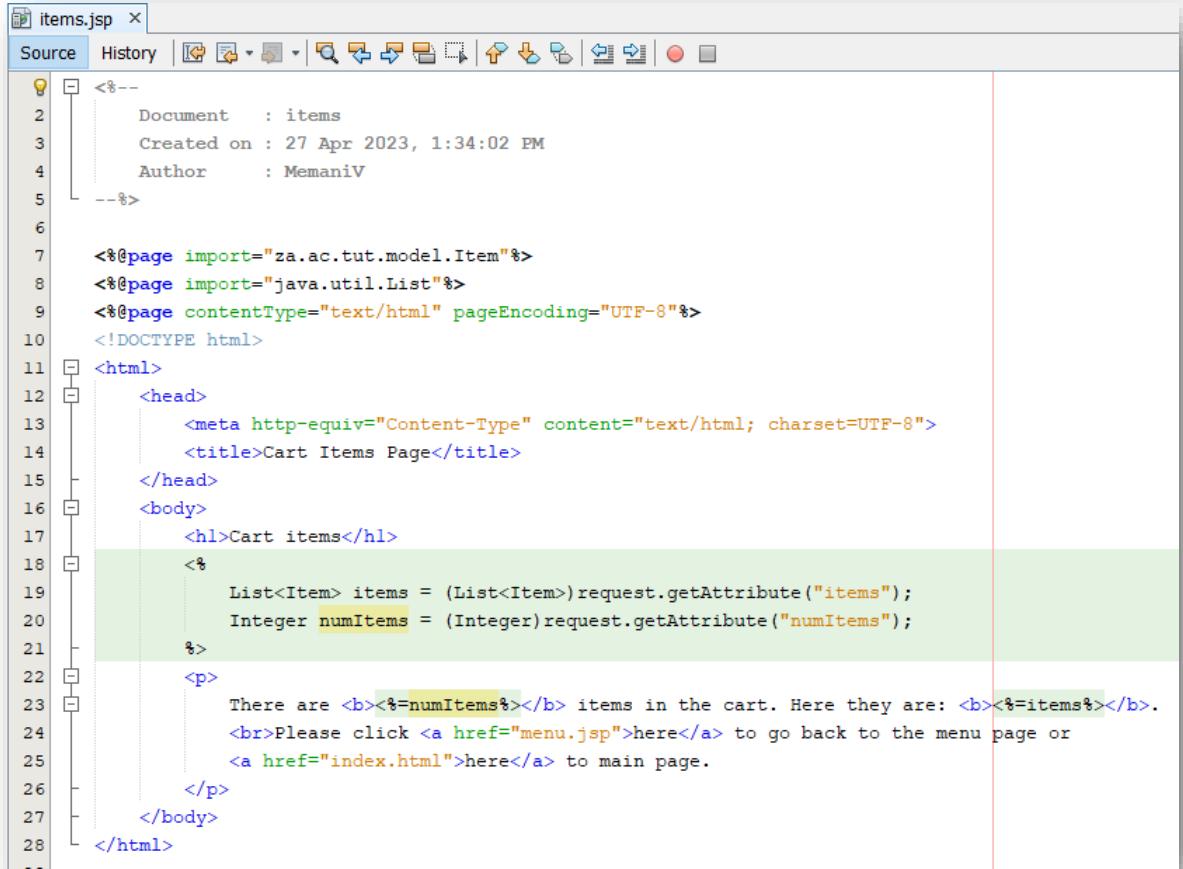
7. Create the **outcome.jsp** file.



The screenshot shows the code editor of a Java IDE with the file "outcome.jsp" open. The code is a JSP page that displays a cart outcome message. It includes imports, page directives, and a scriptlet to get the outcome from the request attribute. The code is well-structured with proper indentation and syntax highlighting.

```
<%--  
1 Document : outcome  
2 Created on : 27 Apr 2023, 1:31:14 PM  
3 Author : MemaniV  
4 --%>  
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>Cart Outcome Page</title>  
13 </head>  
14 <body>  
15     <h1>Cart outcome</h1>  
16     <%  
17         String outcome = (String)request.getAttribute("outcome");  
18     %>  
19     <p>  
20         <b><%=outcome%></b>. Please click <a href="menu.jsp">here</a>  
21         to go back to the menu page or <a href="index.html">here</a> to main page.  
22     </p>  
23 </body>  
24 </html>  
25
```

8. Create the **items.jsp** file.



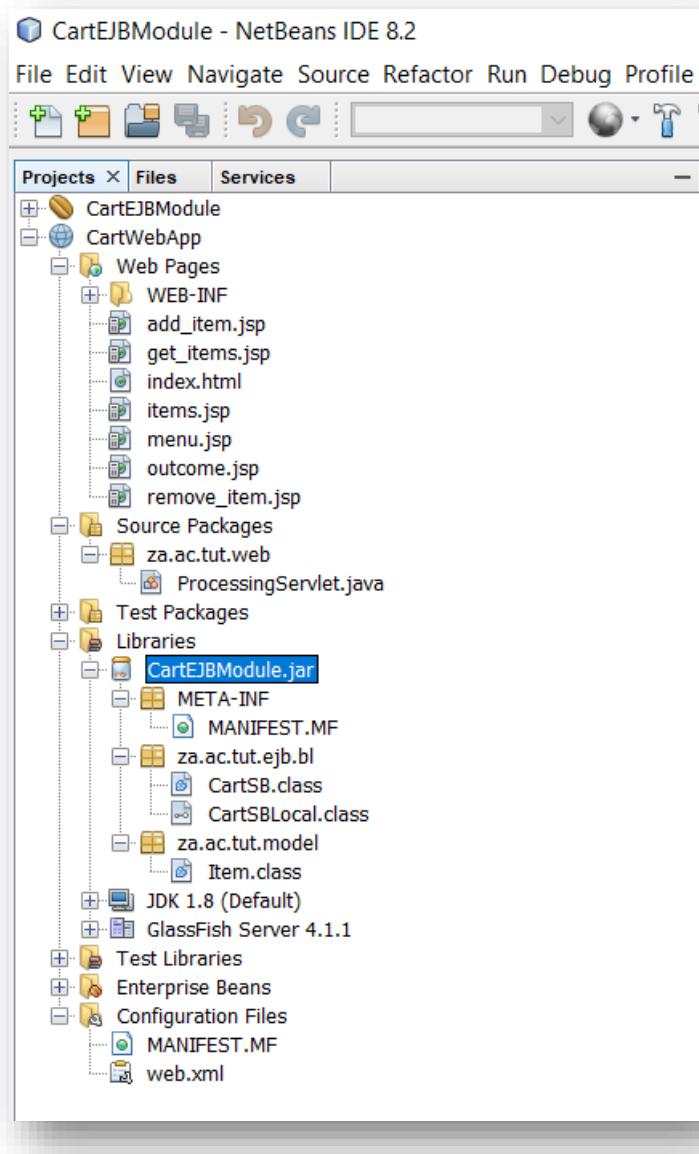
The screenshot shows the code editor of a Java IDE with the file "items.jsp" open. The code is a JSP page that displays a list of items in the cart. It includes imports, page directives, and a scriptlet to get the list of items and the number of items from the request attribute. The code uses JSTL to iterate over the list and print each item. The code is well-structured with proper indentation and syntax highlighting.

```
<%--  
1 Document : items  
2 Created on : 27 Apr 2023, 1:34:02 PM  
3 Author : MemaniV  
4 --%>  
5  
6  
7 <%@page import="za.ac.tut.model.Item"%>  
8 <%@page import="java.util.List"%>  
9 <%@page contentType="text/html" pageEncoding="UTF-8"%>  
10 <!DOCTYPE html>  
11 <html>  
12 <head>  
13     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
14     <title>Cart Items Page</title>  
15 </head>  
16 <body>  
17     <h1>Cart items</h1>  
18     <%  
19         List<Item> items = (List<Item>)request.getAttribute("items");  
20         Integer numItems = (Integer)request.getAttribute("numItems");  
21     %>  
22     <p>  
23         There are <b><%=numItems%></b> items in the cart. Here they are: <b><%=items%></b>.  
24         <br>Please click <a href="menu.jsp">here</a> to go back to the menu page or  
25         <a href="index.html">here</a> to main page.  
26     </p>  
27 </body>  
28 </html>
```

11. Compile the web project.

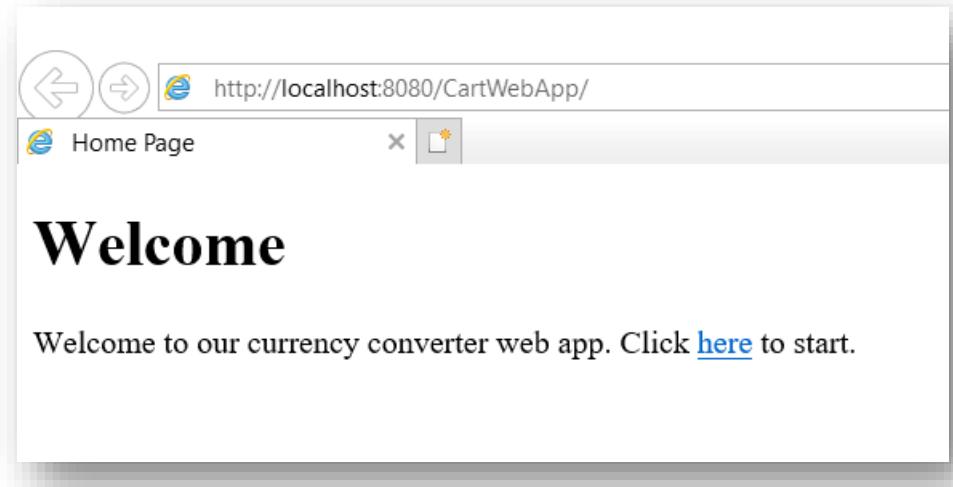
```
Output - CurrencyConverterWebAppV2 (clean,dist) x
deps-ear-jar:
deps-jar:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConverterWebAppV2\build\web\WEB-INF\classes
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\CurrencyConverterWebAppV2\build\web\META-INF
Copying 1 file 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
do-dist:
dist:
BUILD SUCCESSFUL (total time: 0 seconds)
```

12. View the entire structure of the web project.

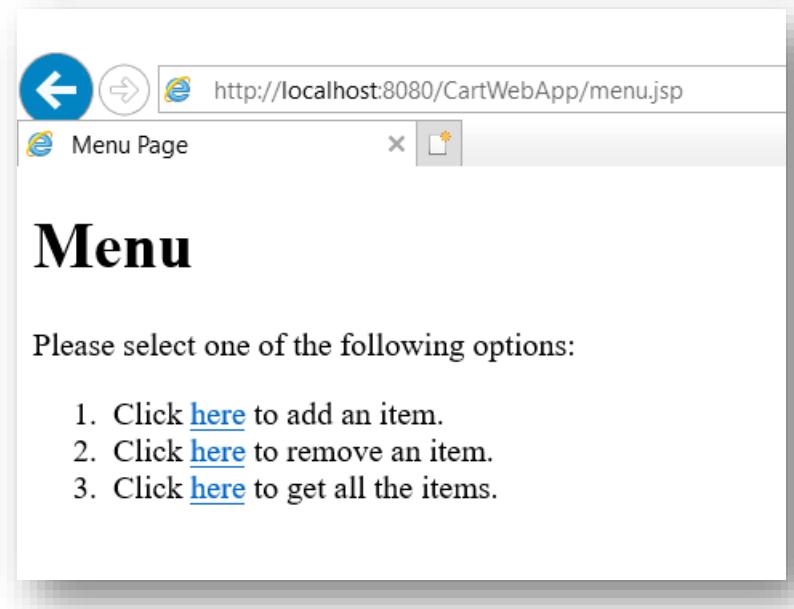


Part C - Run the web client project.

Launch the web app.



Click on the link.



1. Click [here](#) to add an item.
2. Click [here](#) to remove an item.
3. Click [here](#) to get all the items.

Add items

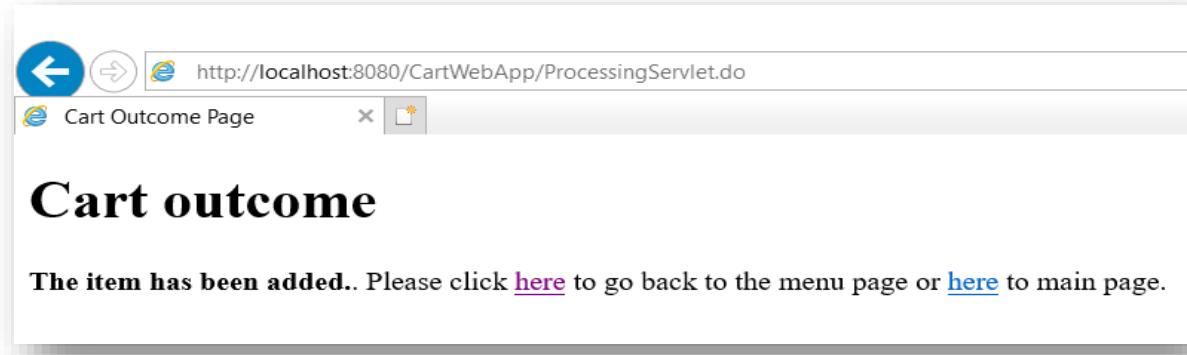
Click on the first link.

The screenshot shows a web browser window with the URL http://localhost:8080/CartWebApp/add_item.jsp. The title bar says "Add Item Page". The main content area has a large heading "Add item". Below it, a message says "Please enter below the details of an item to add.". There are three input fields: "ID:" with an empty box, "Description:" with an empty box, and "Price: R" with an empty box. At the bottom is a "ADD" button.

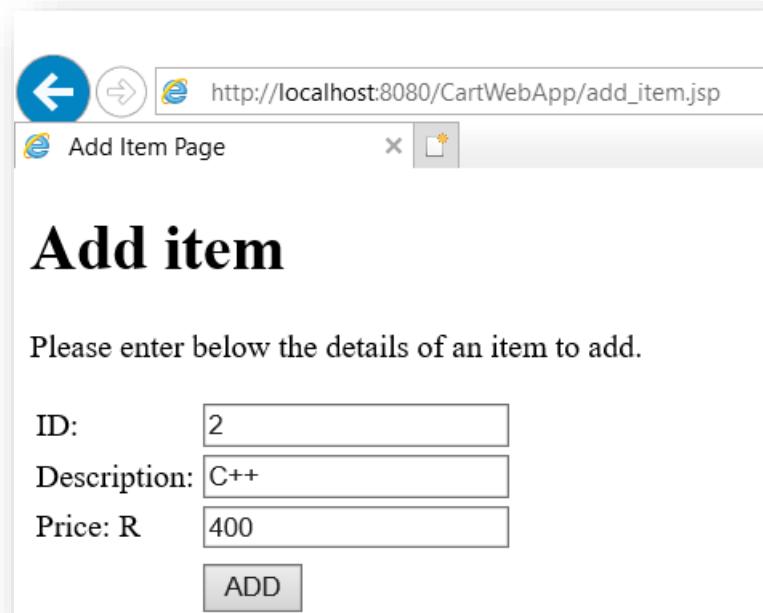
Enter item details.

The screenshot shows the same web browser window as before, but now with data entered into the fields. The "ID:" field contains "1", the "Description:" field contains "JAVA", and the "Price: R" field contains "500". The "ADD" button is still at the bottom.

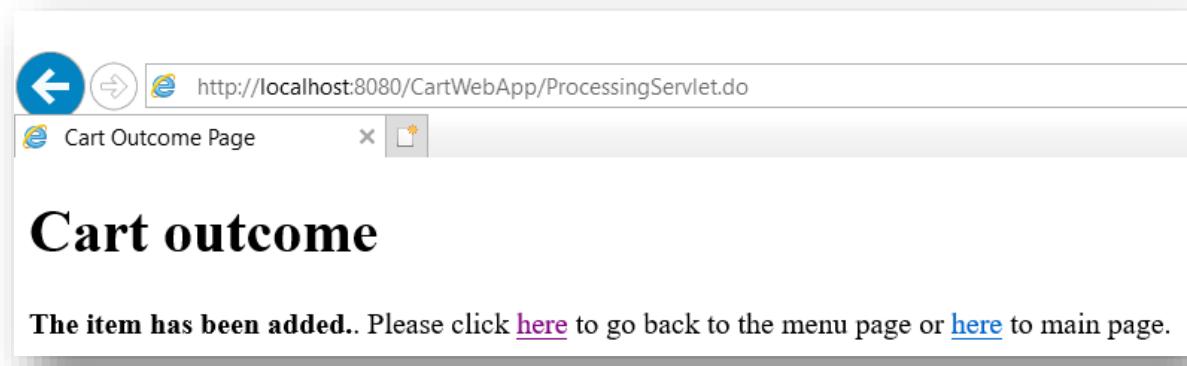
Click on the button.



Add a second book.



Click on the button.



Add a third book

Add item

Please enter below the details of an item to add.

ID:

Description:

Price: R

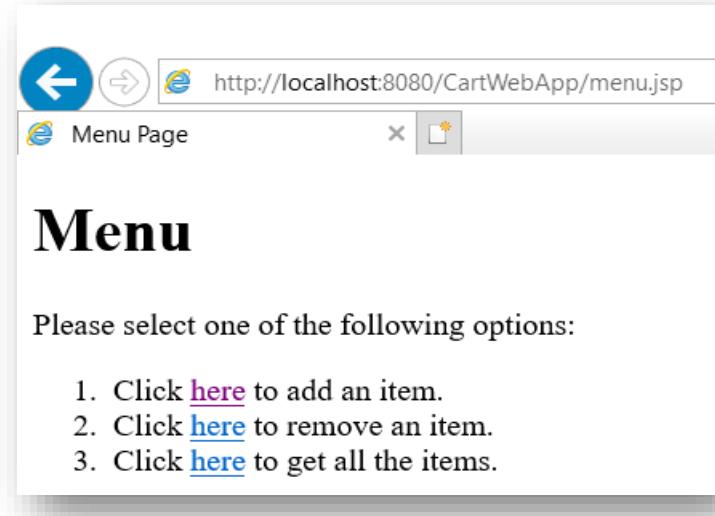
Click on the button.

here to go back to the menu page or [here](#) to main page.'"/>

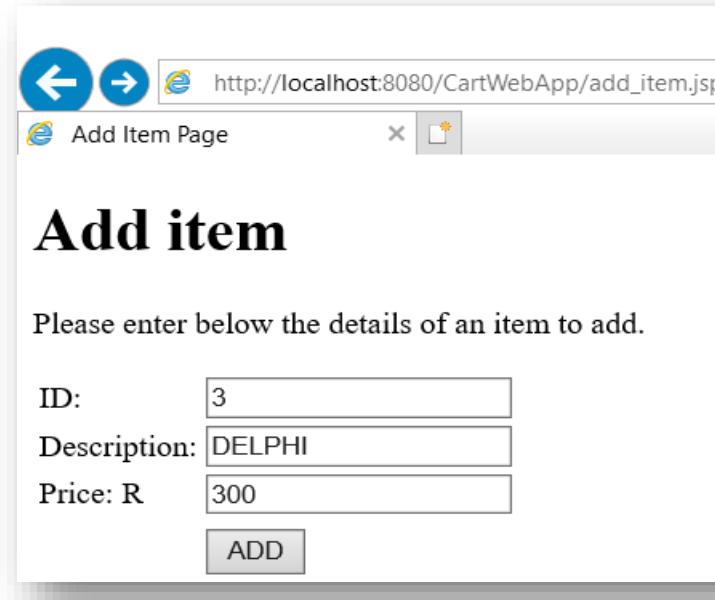
Cart outcome

The item has been added.. Please click [here](#) to go back to the menu page or [here](#) to main page.

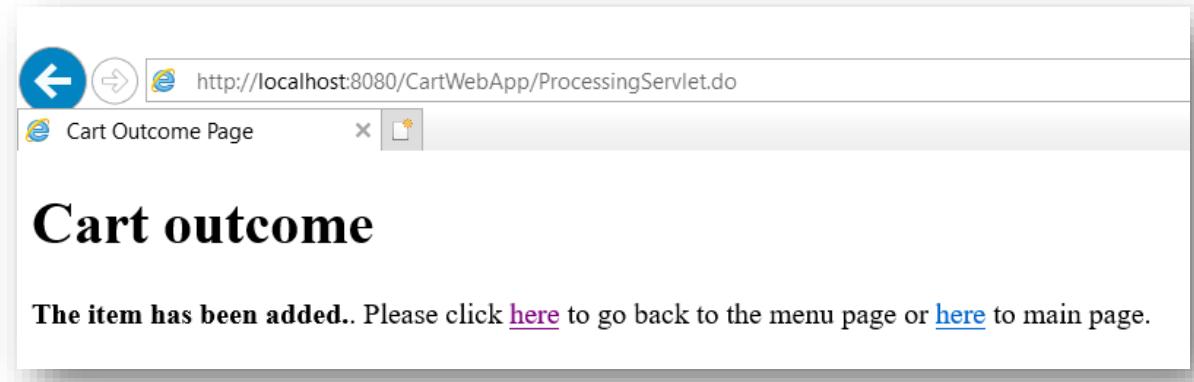
Go back to the menu page.



Click on the third link and add details.



Click on the button.



[View all items](#)

Go back to the menu page

The screenshot shows a web browser window with the URL <http://localhost:8080/CartWebApp/menu.jsp>. The title bar says "Menu Page". The main content area has a large heading "Menu" and the text "Please select one of the following options:" followed by a list:

1. Click [here](#) to add an item.
2. Click [here](#) to remove an item.
3. Click [here](#) to get all the items.

Click on the third link.

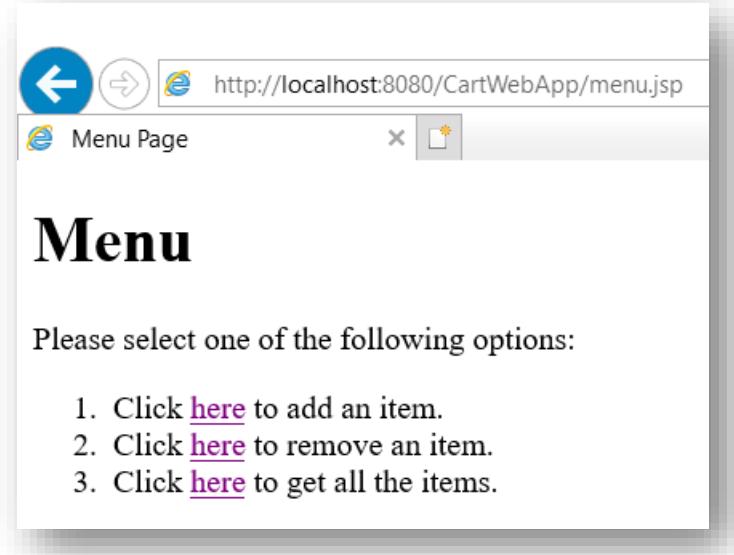
The screenshot shows a web browser window with the URL http://localhost:8080/CartWebApp/get_items.jsp. The title bar says "Get Items Page". The main content area has a large heading "Get items" and the text "Please click the button below." followed by a button labeled "GET ITEMS".

Click on the button.

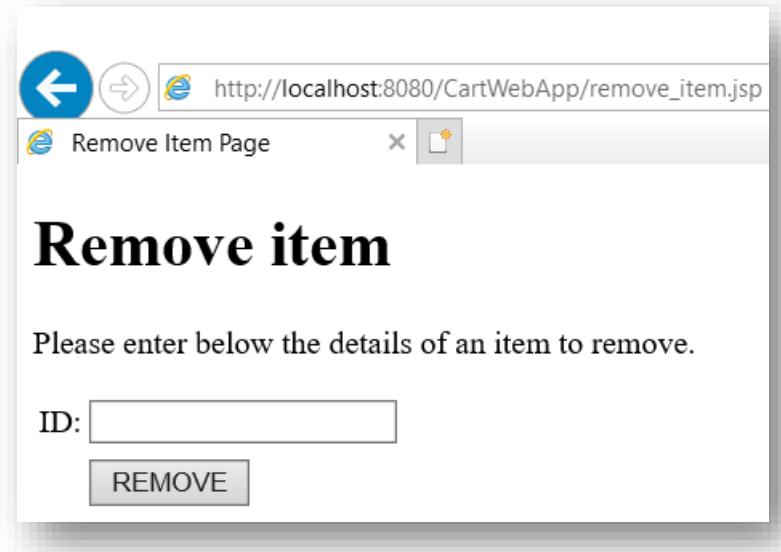
The screenshot shows a web browser window with the URL <http://localhost:8080/CartWebApp/ProcessingServlet.do>. The title bar says "Cart Items Page". The main content area has a large heading "Cart items" and the text "There are 3 items in the cart. Here they are: [{1,JAVA,500.0}, {2,C++,400.0}, {3,DELPHI,300.0}]. Please click [here](#) to go back to the menu page or [here](#) to main page."

Remove item

Go back to the menu page.



Click on the second link.



Enter ID

http://localhost:8080/CartWebApp/remove_item.jsp

Remove Item Page

Remove item

Please enter below the details of an item to remove.

ID:

REMOVE

Click on the button.

http://localhost:8080/CartWebApp/ProcessingServlet.do

Cart Outcome Page

Cart outcome

The item has been successfully removed.. Please click [here](#) to go back to the menu page or [here](#) to main page.

View all the books.

http://localhost:8080/CartWebApp/ProcessingServlet.do

Cart Items Page

Cart items

There are 2 items in the cart. Here they are: [{1, JAVA, 500.0}, {3, DELPHI, 300.0}].
Please click [here](#) to go back to the menu page or [here](#) to main page.

7.2.3 Singleton

The container creates one Singleton for the entire web application. Consequently a Singleton is used when we need multiple clients to manipulate a single resource. A perfect example could be an entertainment event that needs to sell 100 tickets online to patrons.

Since we have a single bean for multiple clients, surely there are concurrency issues that need to be addressed. By default, the container handles concurrent access. But the container also allows programmers to explicitly state that concurrency access must be managed by the container or by the bean itself.

Cocurrency access that is explicitly managed by the container is called CMC (Container Managed Concurrency) and the one managed by the bean is called BMC (Bean Managed Concurrency).

Container Managed Concurrency

When using Container-Managed Concurrency, concurrent access to the singleton bean is managed by the container. The `@Lock` annotation is used to signal that a singleton uses CMC. The annotation takes one of the two values, READ or WRITE. A lock that is of type READ simply means that a method can be called concurrently by clients. A WRITE lock type means a method can only be used by one client at a time, the rest must wait for their turn.

The `@Lock` annotation can be specified at a class level, method level, or even both. When specified at a class level it means it is applicable to all methods. The method specification overrides the class specification for a particular method. Below is an example that shows how to define a singleton with a read concurrency (all class can read at the same time) at class level, and a write concurrency (only one client can write at a time) at a method level. We also specify that the other clients must wait for atleast 20 seconds to access the method, otherwise an exception known as the `ConcurrentAccessTimeoutException` should be thrown.

Example

Write some syntax to represent a Singleton that will allow clients to read the number of show tickets available and also buy the tickets.

```
1  @Singleton
2  @Lock(LockType.READ)
3  public class ShowTickets {
4      private Integer availableTickets = 100;
5
6      public Integer getAvailableTickets(){
7          return availableTickets;
8      }
9
10     @Lock(LockType.WRITE)
11     @AccessTimeout(value = 30, unit = TimeUnit.SECONDS)
12     public Integer buyTickets(Integer quantity){
13         Integer numTicketsBought;
14
15         if(quantity > availableTickets){
16             numTicketsBought = availableTickets;
17         } else {
18             numTicketsBought = quantity;
19         }
20
21         availableTickets -= numTicketsBought;
22
23         return numTicketsBought;
24     }
25
26 }
```

Code explained:

- The `@Singleton` annotation simply says the class `ShowTickets` is a singleton.
- The `@Lock` annotation says that concurrent access to the singleton is managed by the container. The `LockType.READ` says concurrent access to the entire class for reading purposes is granted on all methods.
- The `@Lock(LockType.WRITE)` annotation says concurrent access to this particular method is restricted. Only one client will be allowed to access the method at a time, other clients will have to wait for their turn. If 30 seconds elapses without access being given, an exception must be thrown.

Bean Managed Concurrency

BMC allows full concurrent access to the singleton bean instance. The programmer is then expected to synchronise the activities of the many clients through the usage of Java keywords such as **synchronized** and **volatile**.

As a demonstration we can take the same example of CMC and apply it here. Below is the source code of the modified example.

```
1  @Singleton
2  @ConcurrencyManagement(ConcurrencyManagementType.BEAN)
3  public class ShowTickets {
4      private Integer availableTickets = 100;
5
6      public Integer getAvailableTickets(){
7          return availableTickets;
8      }
9
10     @AccessTimeout(value = 30, unit = TimeUnit.SECONDS)
11     public synchronized Integer buyTickets(Integer quantity){
12         Integer numTicketsBought;
13
14         if(quantity > availableTickets){
15             numTicketsBought = availableTickets;
16         } else {
17             numTicketsBought = quantity;
18         }
19
20         availableTickets -= numTicketsBought;
21
22         return numTicketsBought;
23     }
24
25 }
```

Code explained:

- The `@Singleton` annotation simply says the class `ShowTickets` is a singleton.
- The `@ConcurrencyManagement(ConcurrencyManagement.BEAN)` annotation says that concurrent access to the singleton is managed by the bean.
- The `synchronized` keyword says that access to the method is one client at a time, others will have to wait for their turn. If 30 seconds elapses without access being given, an exception must be thrown.

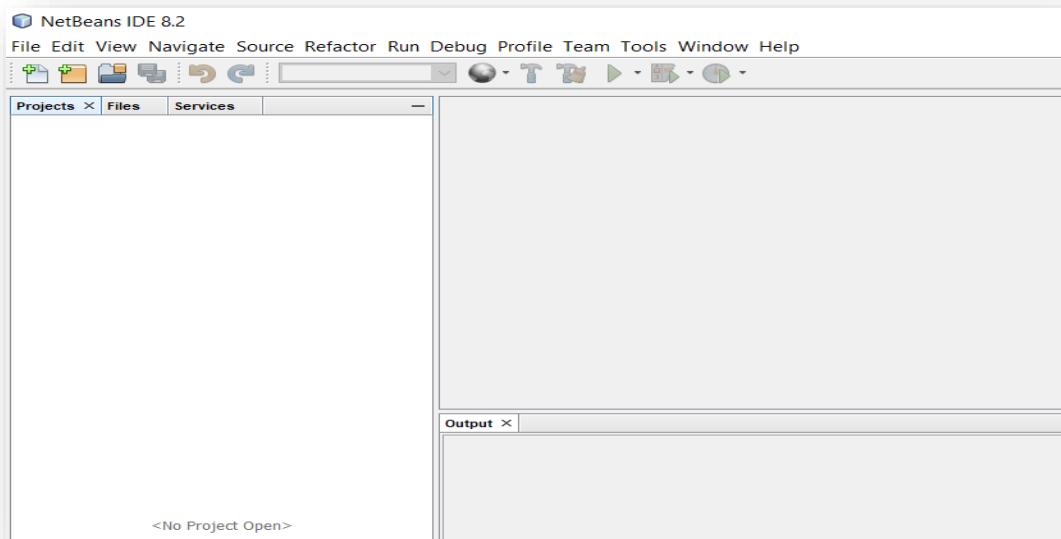
Example

In this example we are going to create a web application that uses a singleton. Our application will sell tickets to users and notify them when the tickets are finished. A ticket costs R20.

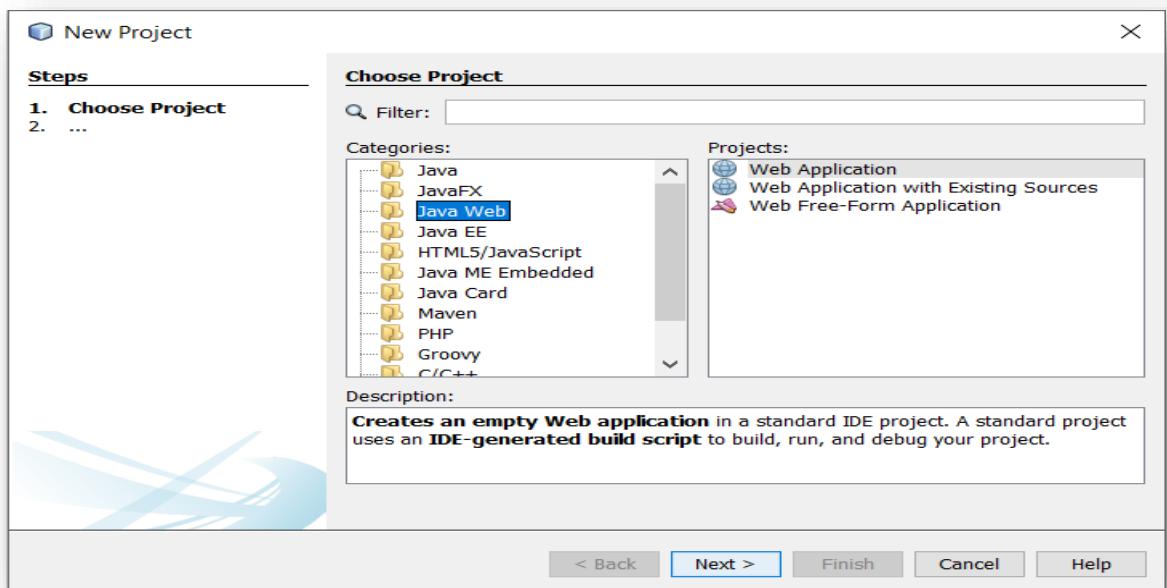
Part A - Create an EJB project

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

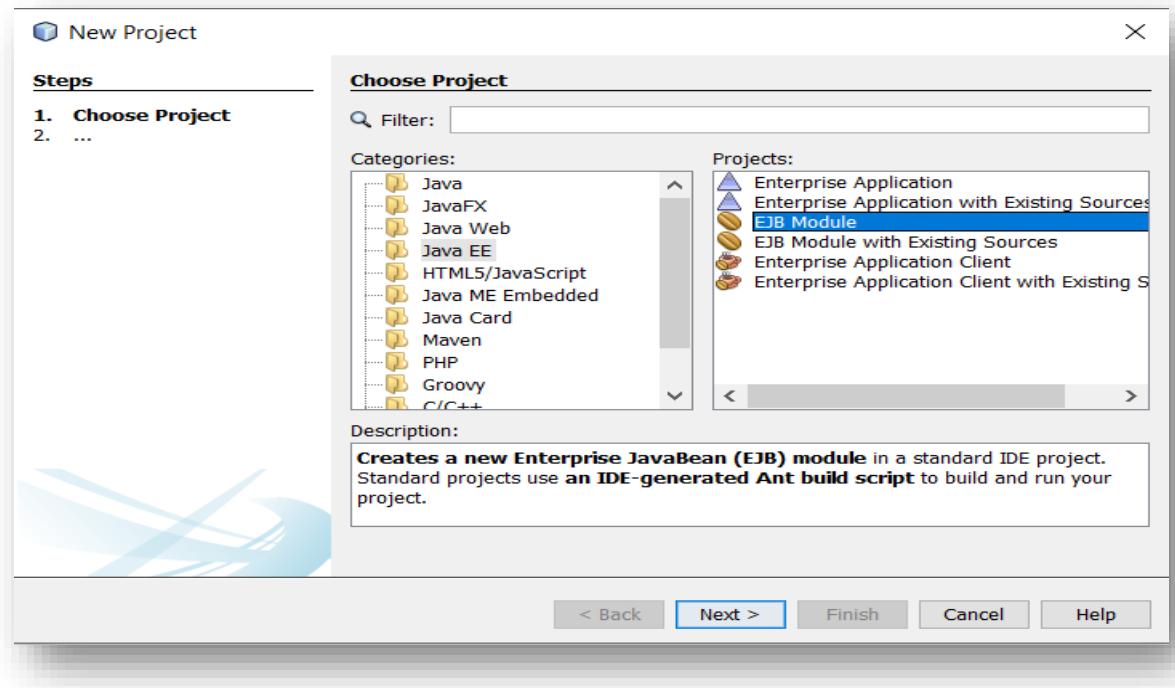
1. Launch NetBeans.



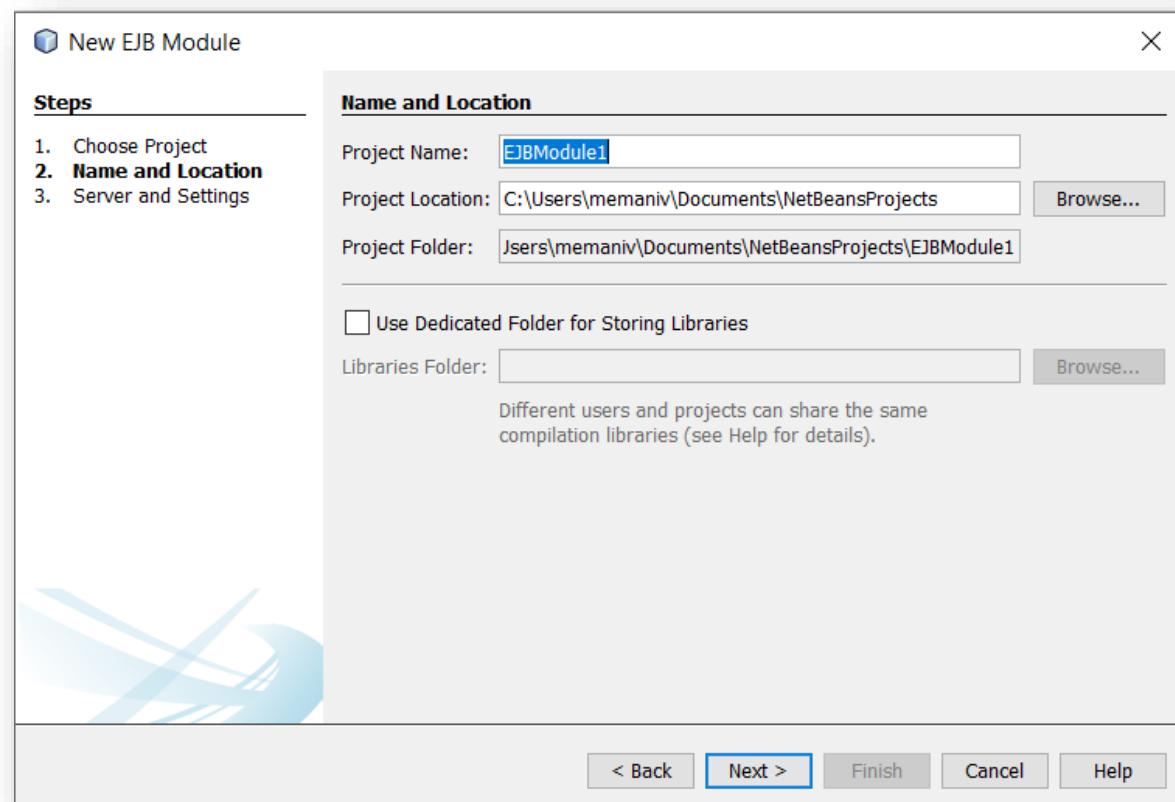
2. Click on **File | New Project**.



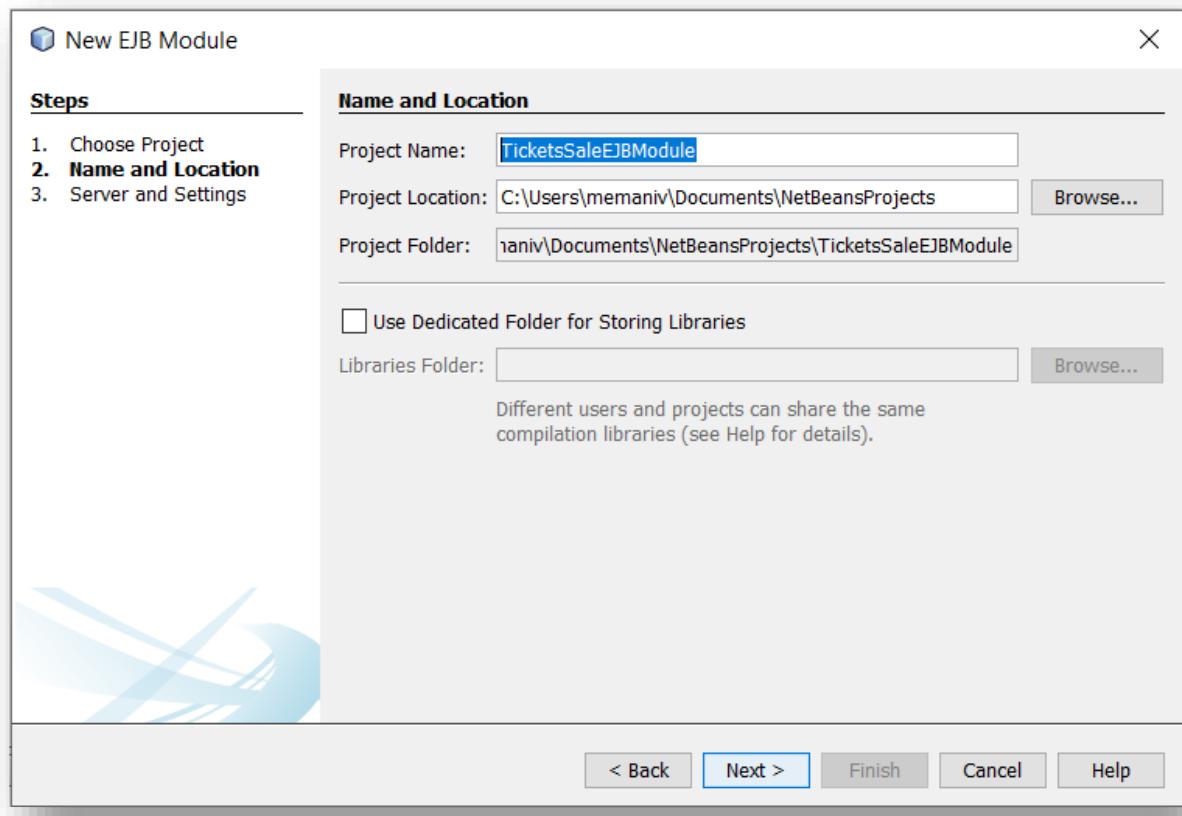
3. Select Java EE under Categories and EJB Module under Projects.



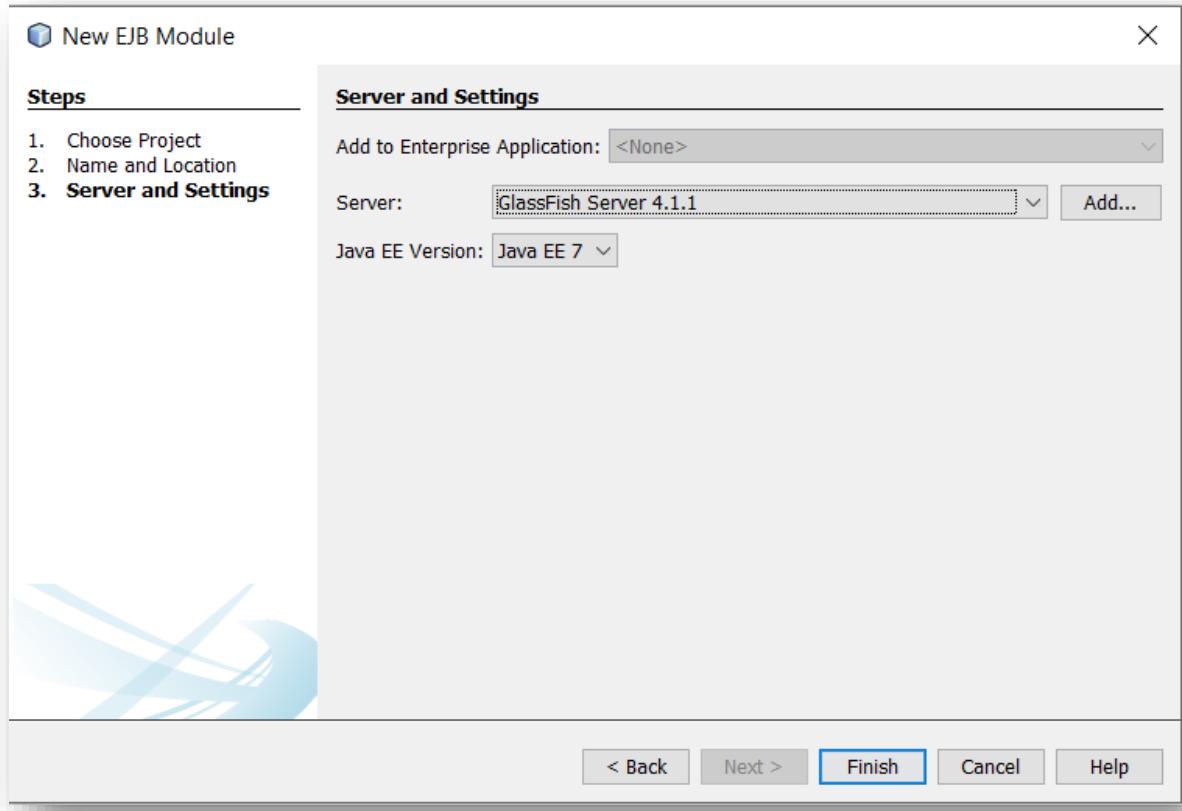
4. Click Next.



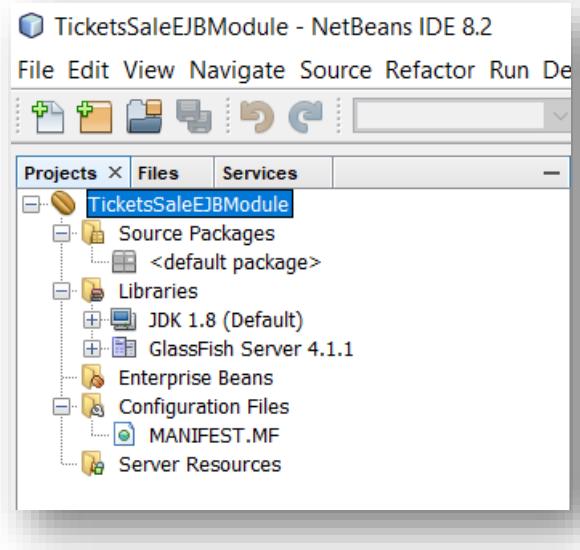
5. Name the project as **TicketsSaleEJBModule**.



6. Click **Next**.

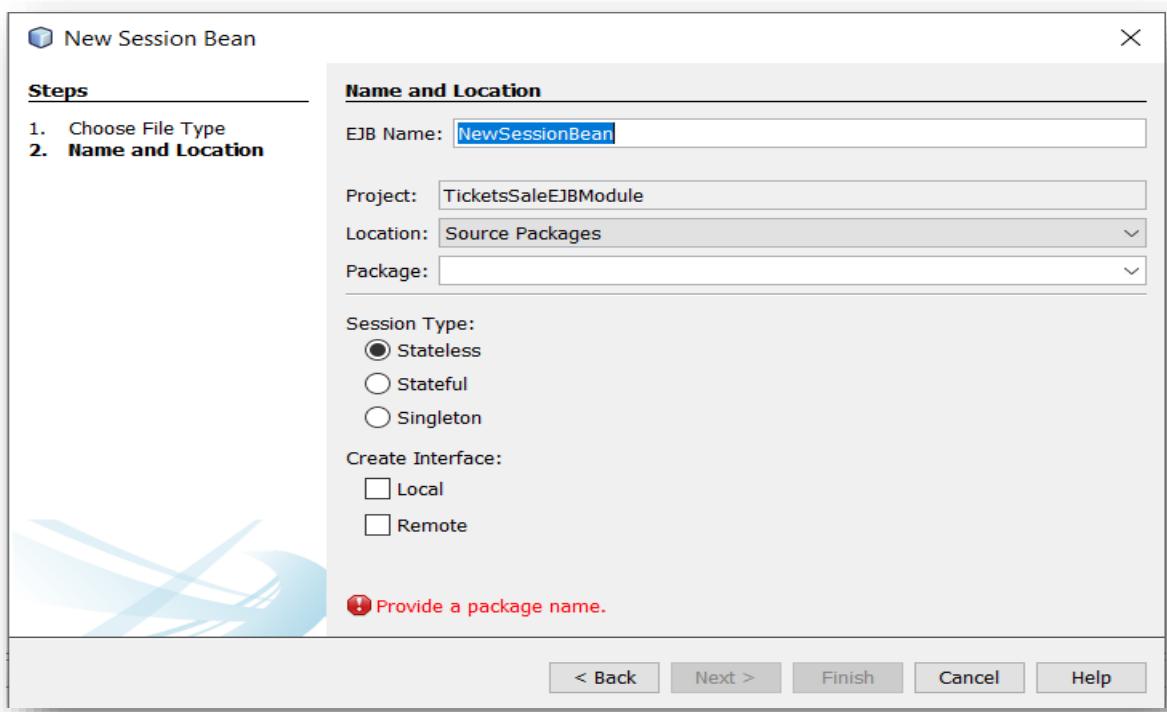


7. Click **Finish**.

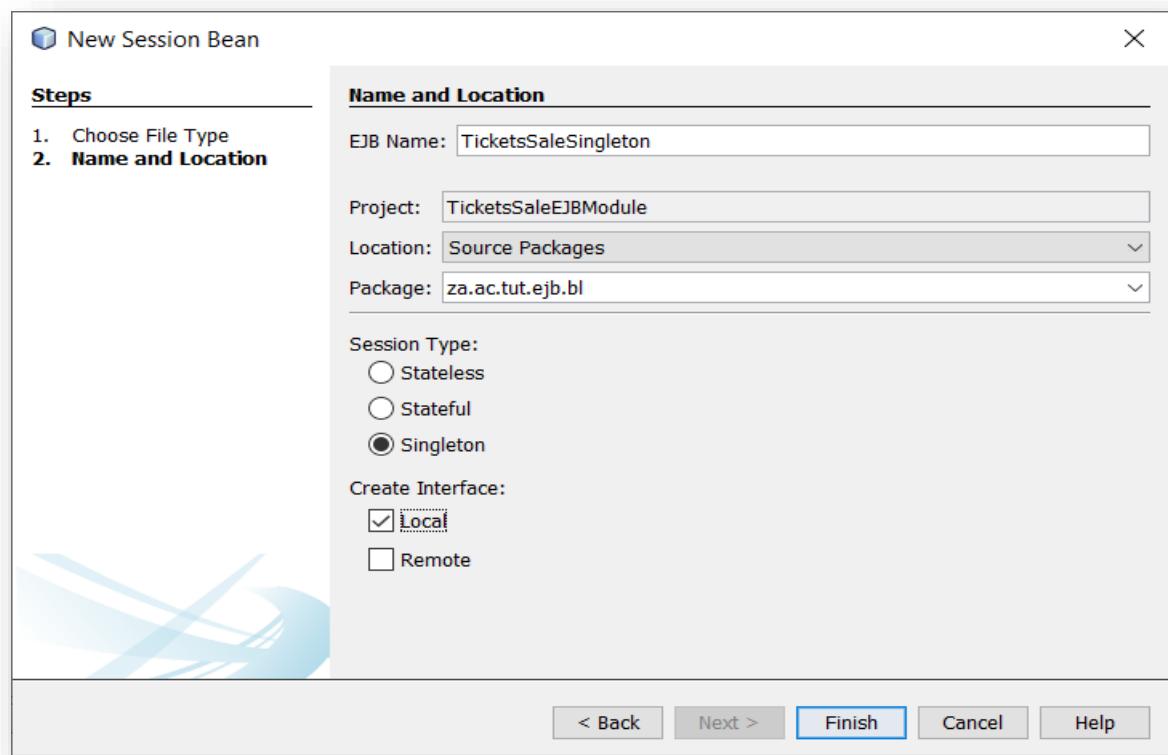


8. Create a singleton session bean. Perform the following steps:

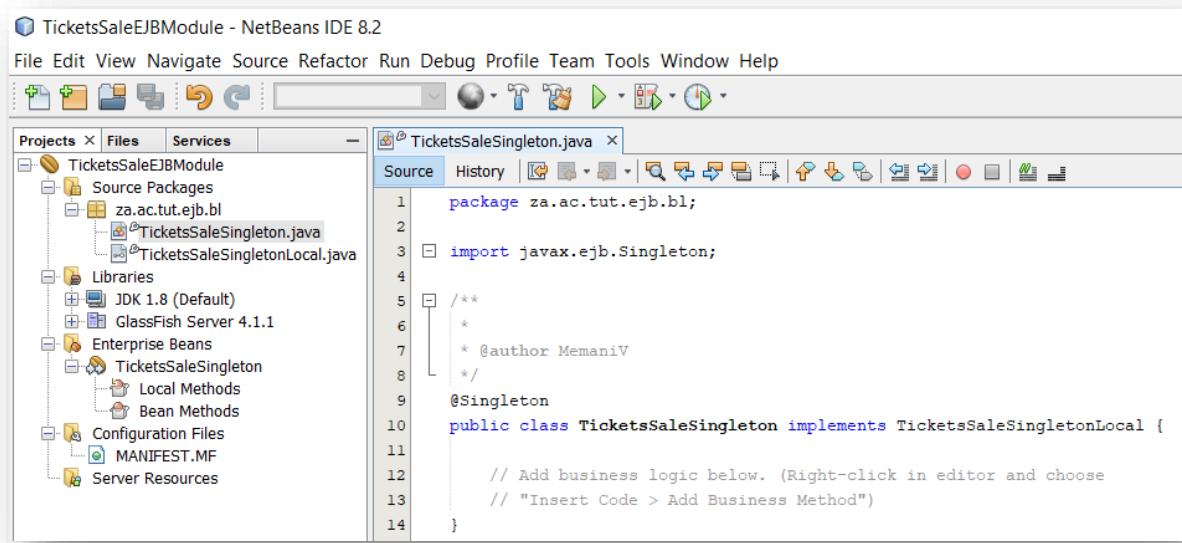
8.1 Right click on the project and select **New | Session Bean**.



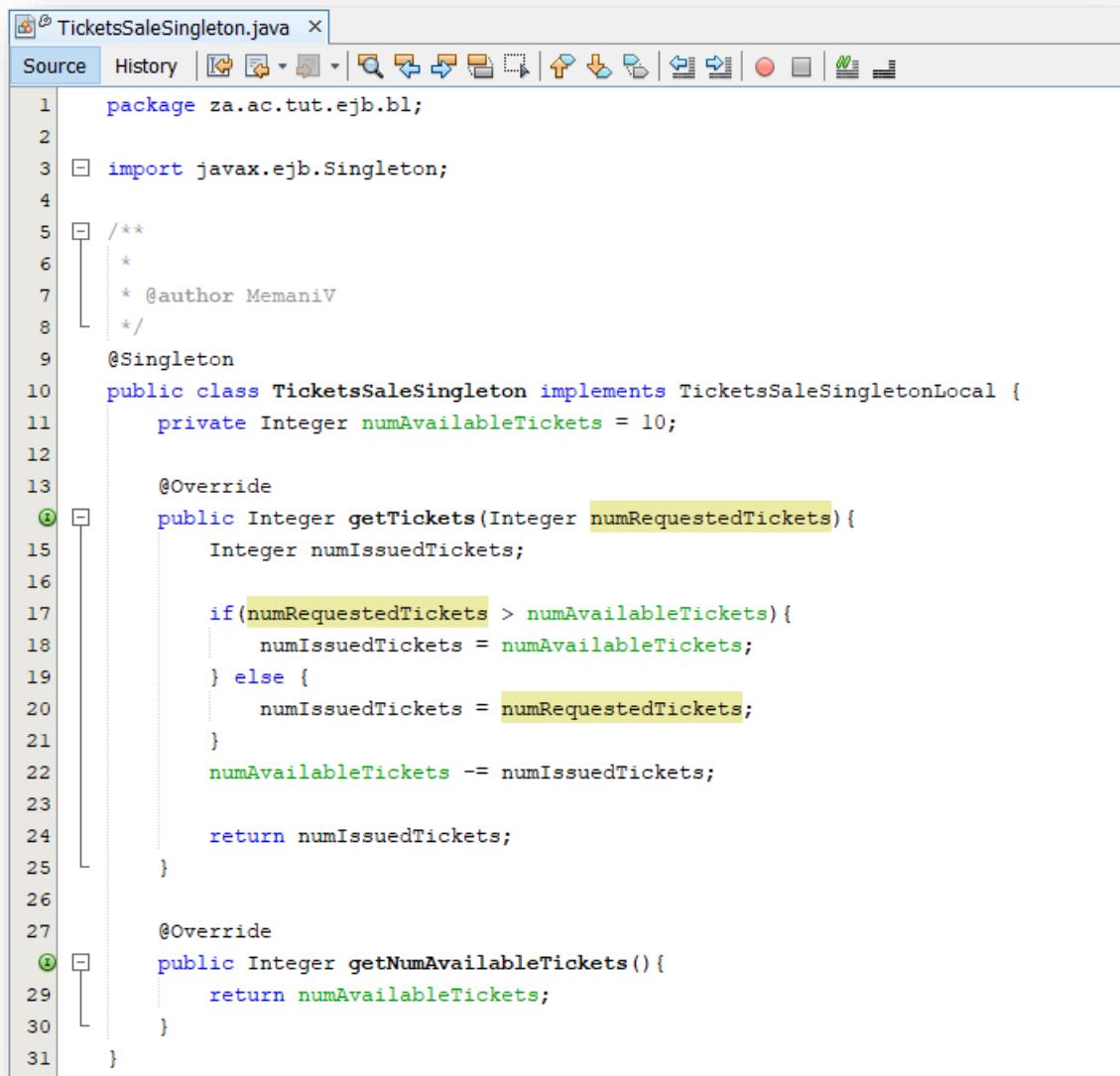
8.2 Name the EJB as **TicketsSaleSingleton**, package it under **za.ac.tut.ejb.bl**, select **Singleton** under Session Type, and select the **Local** interface. The acronym **bl** stands for **business logic**.



8.3 Click **Finish**.



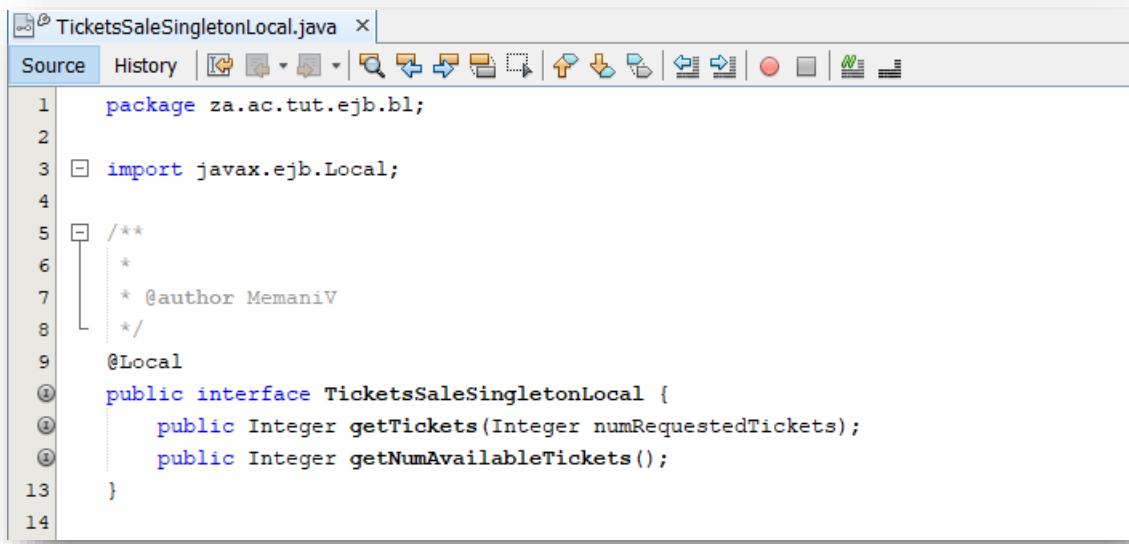
8.4 Add business methods to the singleton.



The screenshot shows the source code for the `TicketsSaleSingleton.java` file. The code defines a singleton bean named `TicketsSaleSingleton` that implements the `TicketsSaleSingletonLocal` interface. It contains two business methods: `getTickets` and `getNumAvailableTickets`. The `getTickets` method checks if the requested number of tickets is greater than the available ones and returns the minimum of the two. The `getNumAvailableTickets` method returns the current number of available tickets. The code uses annotations like `@Singleton`, `@Override`, and `@Local`.

```
1 package za.ac.tut.ejb.bl;
2
3 import javax.ejb.Singleton;
4
5 /**
6  * @author MemaniV
7 */
8
9 @Singleton
10 public class TicketsSaleSingleton implements TicketsSaleSingletonLocal {
11     private Integer numAvailableTickets = 10;
12
13     @Override
14     public Integer getTickets(Integer numRequestedTickets) {
15         Integer numIssuedTickets;
16
17         if (numRequestedTickets > numAvailableTickets) {
18             numIssuedTickets = numAvailableTickets;
19         } else {
20             numIssuedTickets = numRequestedTickets;
21         }
22         numAvailableTickets -= numIssuedTickets;
23
24         return numIssuedTickets;
25     }
26
27     @Override
28     public Integer getNumAvailableTickets() {
29         return numAvailableTickets;
30     }
31 }
```

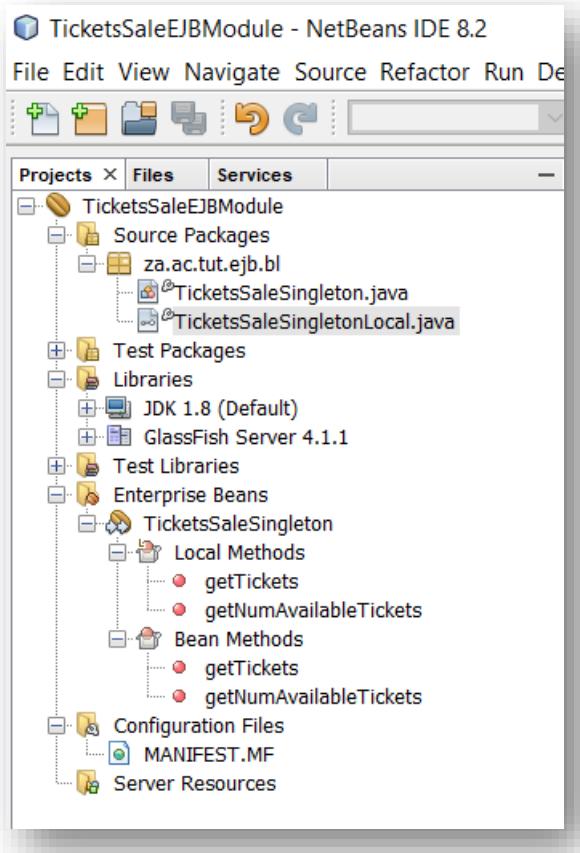
8.5 View the interface.



The screenshot shows the source code for the `TicketsSaleSingletonLocal.java` interface. It defines two methods: `getTickets` which takes an integer parameter and returns an integer, and `getNumAvailableTickets` which returns an integer. The interface uses the `@Local` annotation.

```
1 package za.ac.tut.ejb.bl;
2
3 import javax.ejb.Local;
4
5 /**
6  * @author MemaniV
7 */
8
9 @Local
10 public interface TicketsSaleSingletonLocal {
11     public Integer getTickets(Integer numRequestedTickets);
12     public Integer getNumAvailableTickets();
13 }
14
```

8.6 View the entire project structure.



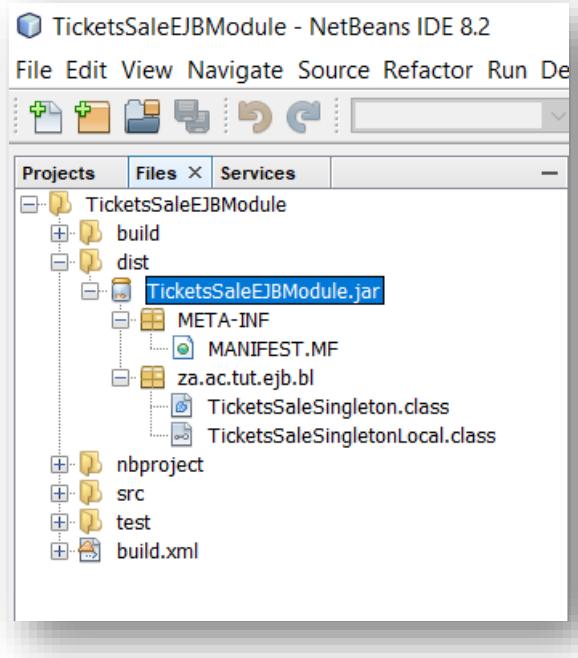
9. Compile the EJB project. Perform the following steps:

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

The screenshot shows the NetBeans Output window titled "Output - TicketsSaleEJBModule (clean, dist) [x]". The window displays the following log output:

```
init:  
deps-jar:  
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\TicketsSaleEJBModule\build\classes  
Copying 1 file to C:\Users\memaniv\Documents\NetBeansProjects\TicketsSaleEJBModule\build\classes\META-INF  
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\TicketsSaleEJBModule\build\empty  
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\TicketsSaleEJBModule\build\generated-sources\ap-source-output  
Compiling 2 source files to C:\Users\memaniv\Documents\NetBeansProjects\TicketsSaleEJBModule\build\classes  
compile:  
library-inclusion-in-archive:  
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\TicketsSaleEJBModule\dist  
Building jar: C:\Users\memaniv\Documents\NetBeansProjects\TicketsSaleEJBModule\dist\TicketsSaleEJBModule.jar  
dist:  
BUILD SUCCESSFUL (total time: 1 second)
```

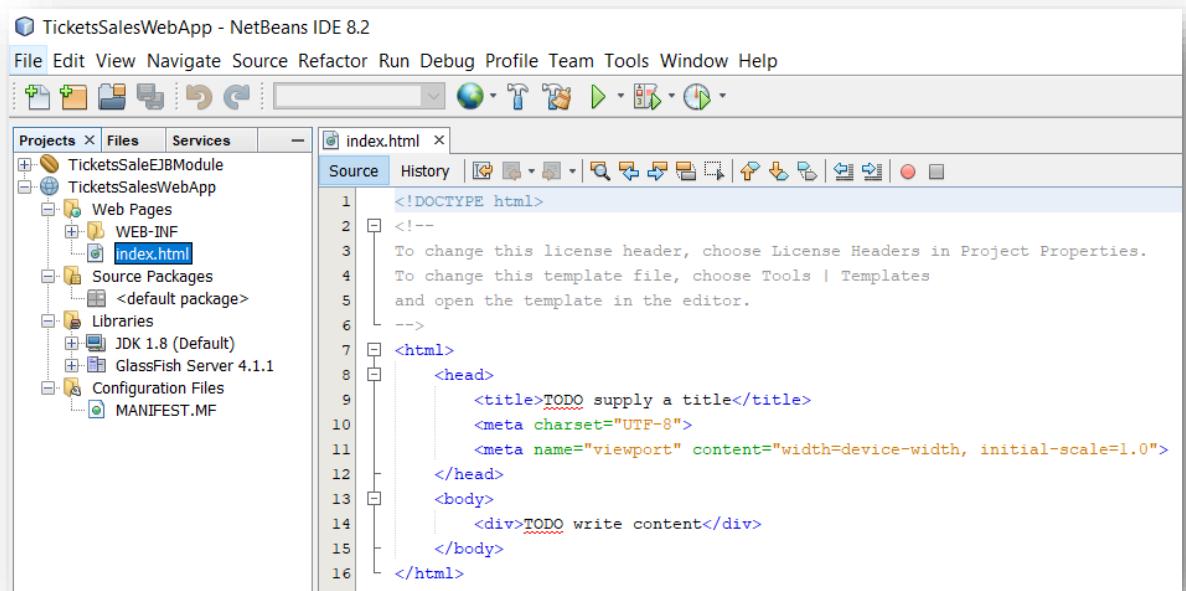
9.2 Click on the **Files** tab and view the contents of the **dist** folder.



Part B - Create web client project

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

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



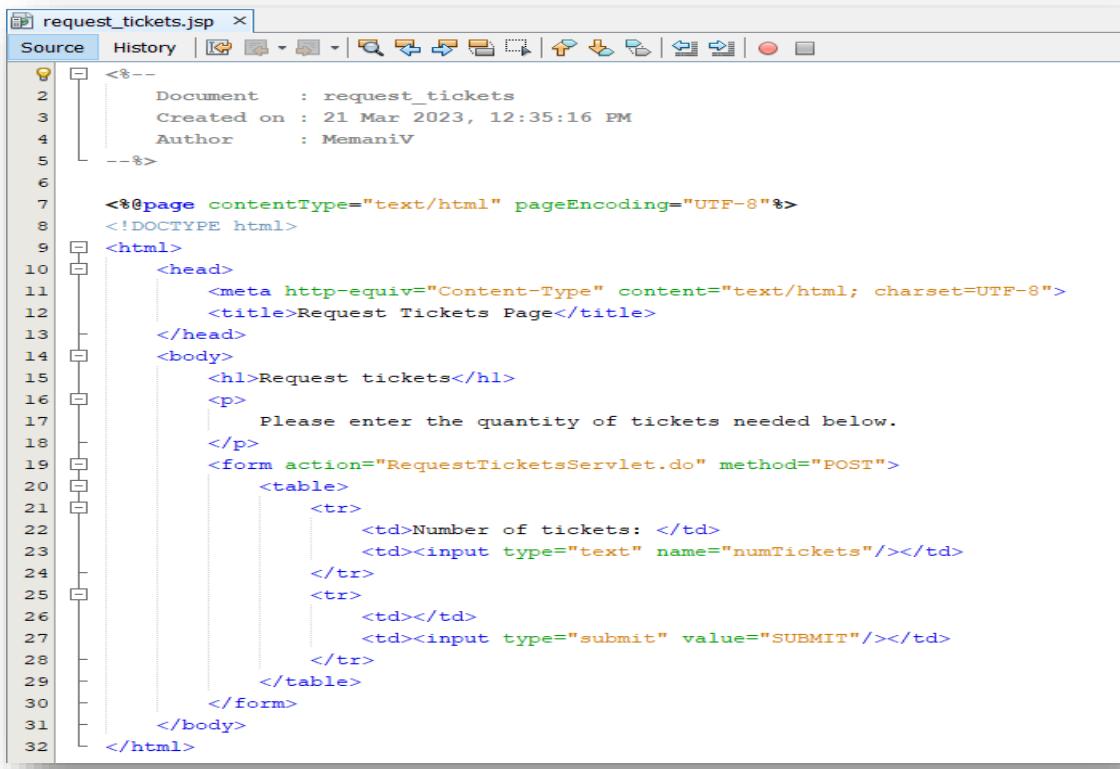
2. Edit the **index.html** file.

```
<!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 online tickets issuing web app. Please click <a href="#menu.jsp">here</a> to start.
        </p>
    </body>
</html>
```

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

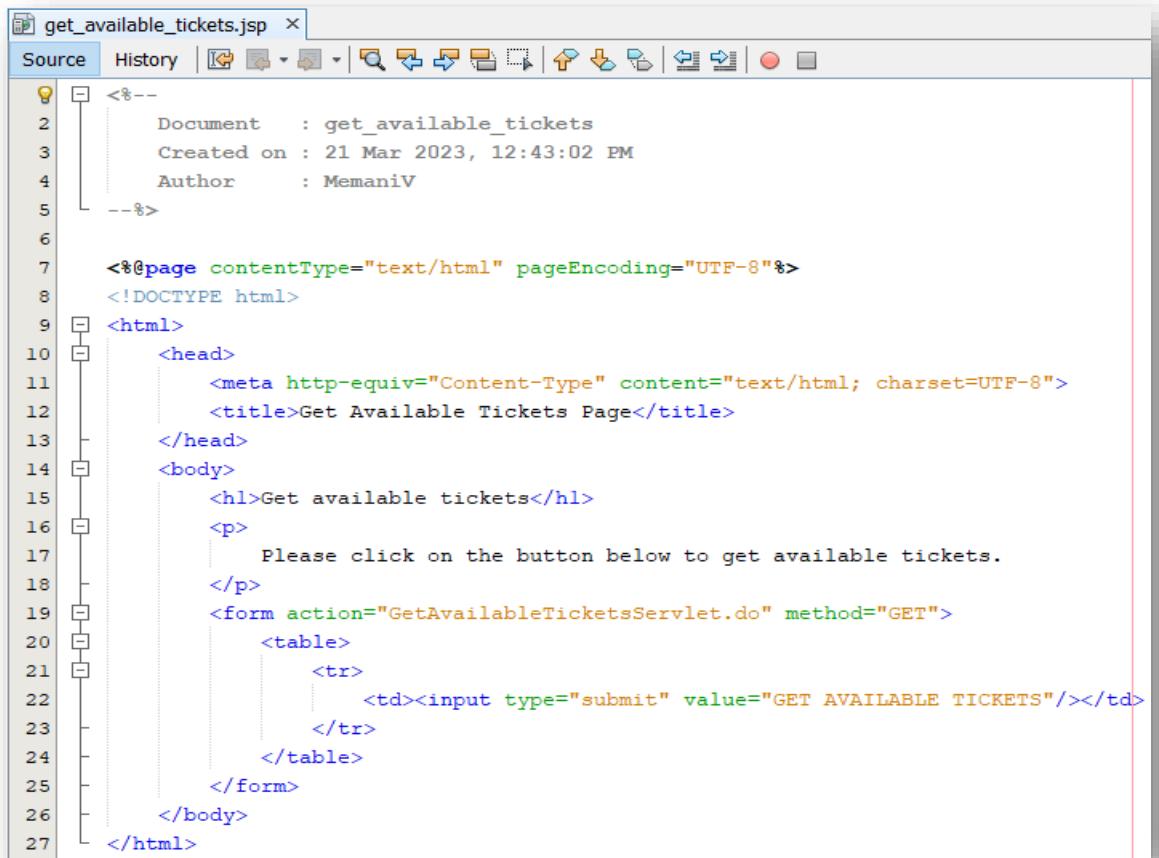
```
<%--  
    Document      : menu  
    Created on   : 21 Mar 2023, 12:30:12 PM  
    Author       : MemaniV  
--%>  
  
<%@page contentType="text/html" pageEncoding="UTF-8"%>  
<!DOCTYPE html>  
<html>  
    <head>  
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
        <title>Menu Page</title>  
    </head>  
    <body>  
        <h1>Menu</h1>  
        <p>  
            Please select one of the following options:  
        </p>  
        <ul>  
            <li>Click <a href="request_tickets.jsp">here</a> to get tickets</li>  
            <li>Click <a href="get_available_tickets.jsp">here</a> to check available tickets.</li>  
        </ul>  
    </body>  
</html>
```

4. Create the **request_tickets.jsp** file.



```
<%--  
    Document      : request_tickets  
    Created on   : 21 Mar 2023, 12:35:16 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>Request Tickets Page</title>  
    </head>  
    <body>  
        <h1>Request tickets</h1>  
        <p>  
            Please enter the quantity of tickets needed below.  
        </p>  
        <form action="RequestTicketsServlet.do" method="POST">  
            <table>  
                <tr>  
                    <td>Number of tickets: </td>  
                    <td><input type="text" name="numTickets"/></td>  
                </tr>  
                <tr>  
                    <td></td>  
                    <td><input type="submit" value="SUBMIT"/></td>  
                </tr>  
            </table>  
        </form>  
    </body>  
</html>
```

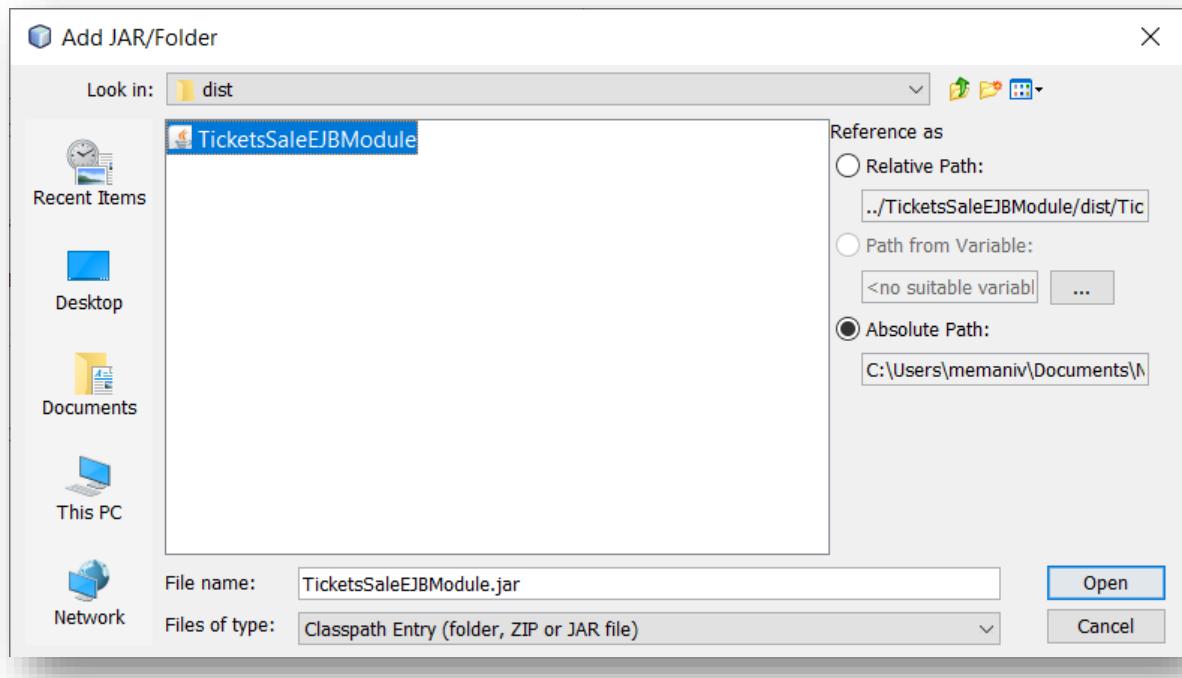
5. Create the **get_available_tickets.jsp** file.



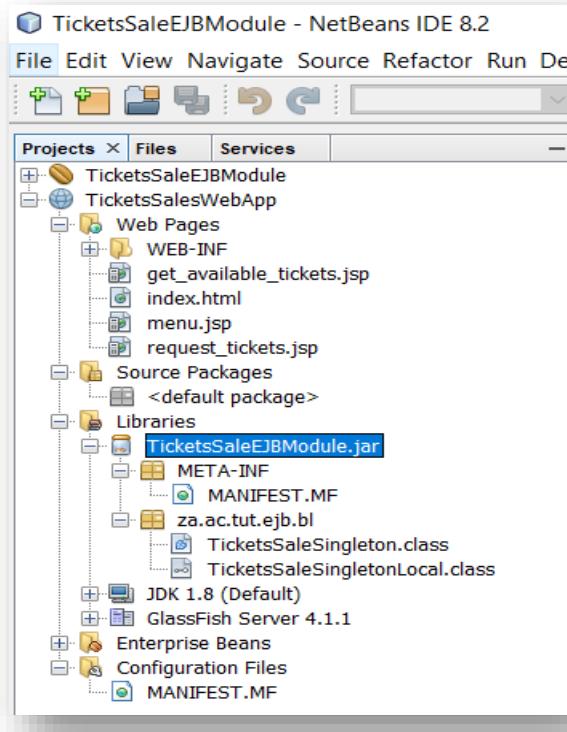
```
<%--  
    Document      : get_available_tickets  
    Created on   : 21 Mar 2023, 12:43:02 PM  
    Author        : MemaniV  
--%>  
  
<%@page contentType="text/html" pageEncoding="UTF-8"%>  
<!DOCTYPE html>  
<html>  
    <head>  
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
        <title>Get Available Tickets Page</title>  
    </head>  
    <body>  
        <h1>Get available tickets</h1>  
        <p>  
            Please click on the button below to get available tickets.  
        </p>  
        <form action="GetAvailableTicketsServlet.do" method="GET">  
            <table>  
                <tr>  
                    <td><input type="submit" value="GET AVAILABLE TICKETS"/></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 **RequestTicketsServlet.java** file.

8. Create the **GetAvailableTicketsServlet.java** file.

9. Create the **request_tickets_outcome.jsp** file.

The screenshot shows the JSP code for the `request_tickets_outcome.jsp` file. The code is as follows:

```
<%--  
1 Document : request_tickets_outcome  
2 Created on : 21 Mar 2023, 1:49:49 PM  
3 Author : MemaniV  
4--%>  
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>Request Tickets Outcome Page</title>  
13 </head>  
14 <body>  
15 <h1>Outcome</h1>  
16 <%  
17 Integer numTicketsRequested = (Integer)request.getAttribute("numTicketsRequested");  
18 Integer numTicketsIssued = (Integer)request.getAttribute("numTicketsIssued");  
19 %>  
20 <p>  
21 You requested <b><%=numTicketsRequested%></b> tickets, and <b><%=numTicketsIssued%></b> were issued to you.  
22 Kindly click <a href="menu.jsp">here</a> to go back to the menu page or <a href="index.html">here</a> to the main page.  
23 </p>  
24 </body>  
25 </html>
```

10. Create the **get_available_tickets_outcome.jsp** file.

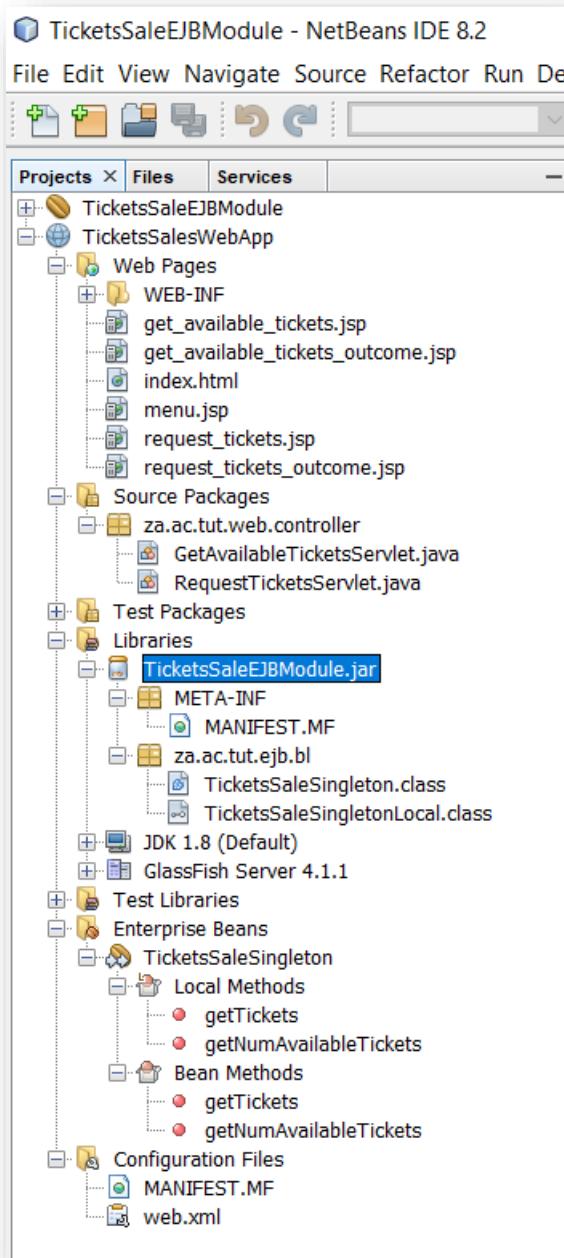
The screenshot shows the JSP code for the `get_available_tickets_outcome.jsp` file. The code is as follows:

```
<%--  
1 Document : get_available_tickets_outcome  
2 Created on : 21 Mar 2023, 2:16:59 PM  
3 Author : MemaniV  
4--%>  
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>Get Available Tickets Outcome Page</title>  
13 </head>  
14 <body>  
15 <h1>Available tickets</h1>  
16 <%  
17 Integer numAvailableTickets = (Integer)request.getAttribute("numAvailableTickets");  
18 %>  
19 <p>  
20 There are <b><%=numAvailableTickets%></b> tickets available.  
21 Kindly click <a href="menu.jsp">here</a> to go back to the menu page or  
22 <a href="index.html">here</a> to the main page.  
23 </p>  
24 </body>  
25 </html>
```

11. Compile the web project.

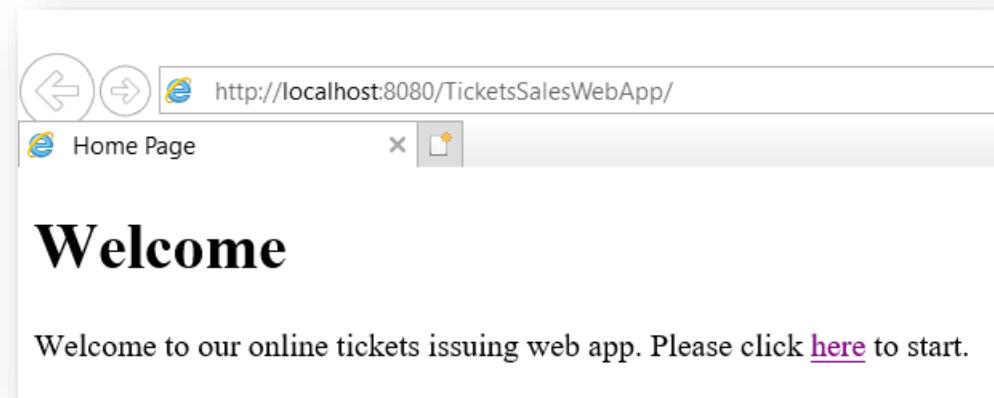
```
Output - TicketsSalesWebApp (clean, dist) x
library-inclusion-in-manifest:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\TicketsSalesWebApp\build\empty
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\TicketsSalesWebApp\build\generated-sources\ap-source-output
Compiling 2 source files to C:\Users\memaniv\Documents\NetBeansProjects\TicketsSalesWebApp\build\web\WEB-INF\classes
compile:
compile-jsp:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\TicketsSalesWebApp\dist
Building jar: C:\Users\memaniv\Documents\NetBeansProjects\TicketsSalesWebApp\dist\TicketsSalesWebApp.war
do-dist:
dist:
BUILD SUCCESSFUL (total time: 0 seconds)
```

12. View entire structure of the web project.

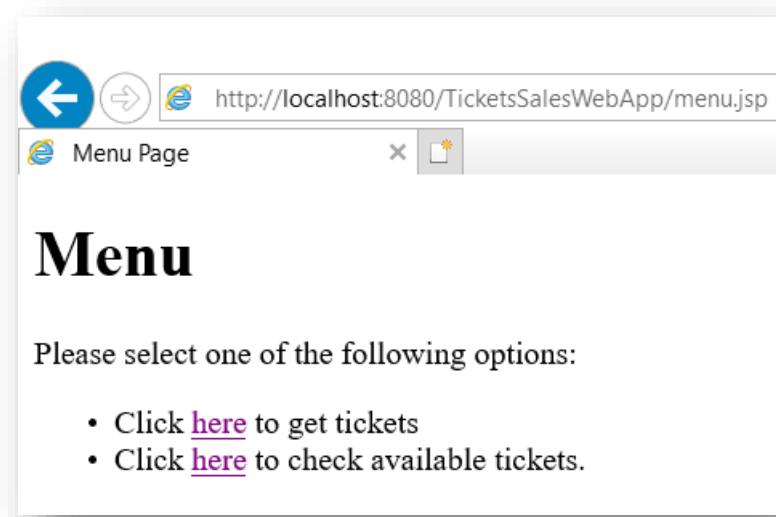


Part C - Run the web client project

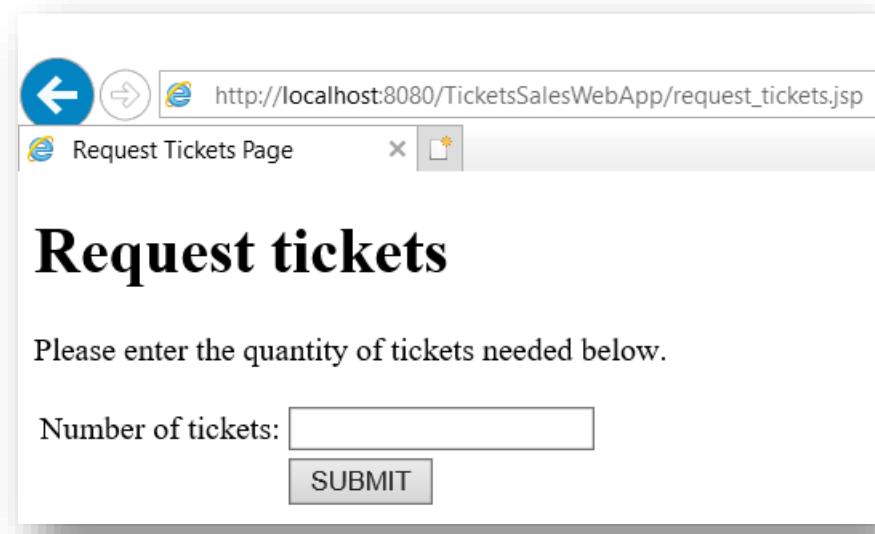
Launch the web app.



Click on the link.



Click on the first link.



Enter the quantity.

The screenshot shows a web browser window with the URL http://localhost:8080/TicketsSalesWebApp/request_tickets.jsp. The title bar says "Request Tickets Page". The main content area has a large heading "Request tickets". Below it is a text instruction "Please enter the quantity of tickets needed below.". A form contains a label "Number of tickets:" followed by a text input field containing the number "6". Below the input field is a "SUBMIT" button.

Click on the button.

The screenshot shows a web browser window with the URL <http://localhost:8080/TicketsSalesWebApp/RequestTicketsServlet.do>. The title bar says "Request Tickets Outcome P...". The main content area has a large heading "Outcome". Below it is a message: "You requested 6 tickets, and 6 were issued to you. Kindly click [here](#) to go back to the menu page or [here](#) to the main page."

Go back to the menu page.

The screenshot shows a web browser window with the URL <http://localhost:8080/TicketsSalesWebApp/menu.jsp>. The title bar says "Menu Page". The main content area has a large heading "Menu". Below it is a text instruction "Please select one of the following options:". A bulleted list follows:

- Click [here](#) to get tickets
- Click [here](#) to check available tickets.

Click on the second link.

The screenshot shows a web browser window with the URL http://localhost:8080/TicketsSalesWebApp/get_available_tickets.jsp. The title bar says "Get Available Tickets Page". The main content area has a large heading "Get available tickets" and a message "Please click on the button below to get available tickets." Below the message is a button labeled "GET AVAILABLE TICKETS".

Click on the button.

The screenshot shows a web browser window with the URL <http://localhost:8080/TicketsSalesWebApp/GetAvailableTicketsServlet.do>. The title bar says "Get Available Tickets Outco...". The main content area has a large heading "Available tickets" and a message "There are 4 tickets available. Kindly click [here](#) to go back to the menu page or [here](#) to the main page."

Go back to the menu page.

The screenshot shows a web browser window with the URL <http://localhost:8080/TicketsSalesWebApp/menu.jsp>. The title bar says "Menu Page". The main content area has a large heading "Menu" and a message "Please select one of the following options:". Below the message is a list:

- Click [here](#) to get tickets
- Click [here](#) to check available tickets.

Click on the first link.

The screenshot shows a web browser window with the URL http://localhost:8080/TicketsSalesWebApp/request_tickets.jsp. The title bar says "Request Tickets Page". The main content area has a large heading "Request tickets" and a message "Please enter the quantity of tickets needed below.". Below this is a form with a label "Number of tickets:" followed by an input field containing "5" and a "SUBMIT" button.

Enter the quantity

The screenshot shows the same web browser window as before, but now the input field contains the number "5". The rest of the page remains the same, with the heading "Request tickets" and the submission button.

Click on the button.

The screenshot shows a web browser window with the URL <http://localhost:8080/TicketsSalesWebApp/RequestTicketsServlet.do>. The title bar says "Request Tickets Outcome P...". The main content area has a large heading "Outcome" and a message "You requested 5 tickets, and 4 were issued to you. Kindly click [here](#) to go back to the menu page or [here](#) to the main page."

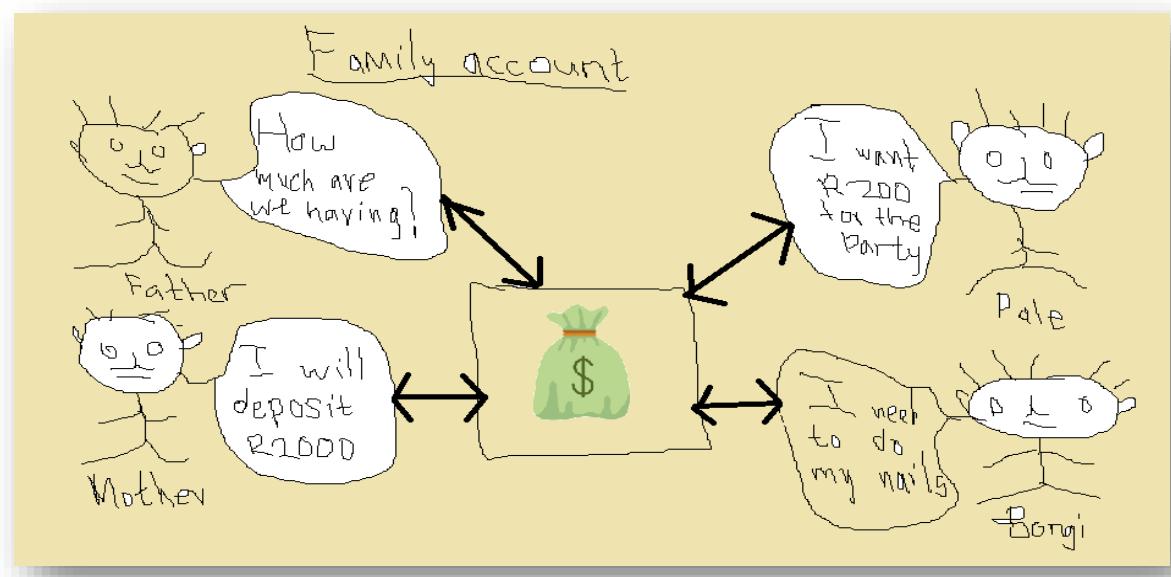
7.3 DIY (Do It Yourself)

In this DIY we want you to develop web applications that use appropriate EJBs for business logic implementation.

Task #1

Mulumba has just secured an internship at bank ABC. He works in the programming section of the bank. His senior, Ms Mudau, has assigned him a task. The task is to create an interactive web application that uses EJBs. The intended application must allow customers, who are sharing the same account, to perform the following tasks on the shared account:

- Check balance;
- Deposit cash; and
- Withdraw cash.



To do

Assuming that you are Mulumba, create such an application for ABC. You may assume that the initial balance is R1000. Ms Mudau gives strict instructions that you must use a **Singleton** to perform the business logic and BMC to manage concurrent access.

-----END OF TASK-----

Task #2

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.

-----END OF TASK-----

Task #3

Sipho is a programming intern at ***WeDoWebApps*** company. The company specializes in developing conversational web applications that are fail-safe for clients. Ms Ntuli, a senior developer at WeDoWebApps, has been assigned the responsibility of mentoring Sipho. As his first task, Ms Ntuli gives Sipho an opportunity of creating a conversational web app that is fail-safe for a client. The client is ABC, an engineering company.

ABC has stringent policy when it comes to hiring new employees at entry level. As an engineering company, they need to make sure that potential employees have the basic numeracy skills. Consequently they have developed a 3-stage recruitment process for potential staff members at entry level. At the first stage, they shortlist promising candidates whom they take to the next stage. During the second stage the numeracy skills of candidates are tested and those who pass are taken to the

next and final stage, which is the interview stage. The company has set a pass mark of 70%. Candidates choose the test they want to write. They can either do an addition, subtraction, division or multiplication test.

The company wants a web application that will handle the competency test part. For each test category (addition, subtraction, division or multiplication), the application must generate five (5) random questions for a candidate. The candidate must be given one question at a time to answer. After a candidate has provided an answer, an outcome must be provided. The outcome must display the question asked; the user's answer; the correct answer; and the actual outcome, that is whether the user got the question right or wrong. A subsequent question must be given for answering.

After all the questions have been asked and answered, a summary report must be generated that shows the name of the candidate; the test chosen by the candidate; the number of questions asked; the number of correct answers; the number of wrong answers; the percentage mark obtained; and the final outcome (whether a candidate qualifies for an interview or not).

The company makes a special request that when it comes to addition; division; and subtraction, the questions must use random numbers in the range of 10 to 90, both numbers inclusive. But for multiplication, the questions must use numbers in the range of 1 to 10, both numbers inclusive.

To do

Assuming that you are Sipho, create such a web application. Use a **stateful** session bean to perform the business logic.

-----END OF TASK-----

Task #4

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 C. 2 D. 0	D
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.

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.

-----END OF TASK-----

7.4 Conclusion

In this chapter we managed to introduce the student to EJBs, the business component of JEE.

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

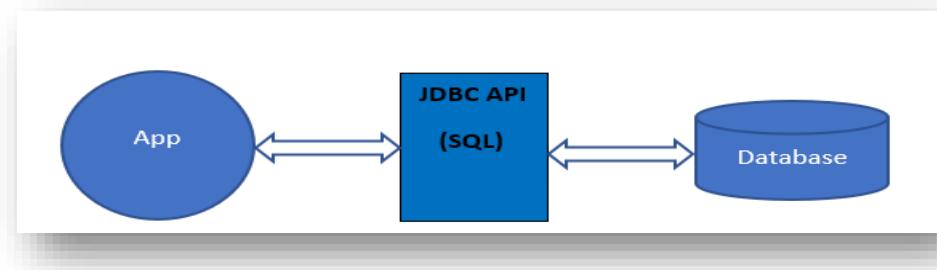
8 Java Persistence API (JPA)

In this chapter, we introduce the student to the concept of JPA, that is Java Persistence API. We also demonstrate the implementation of JPA in a web development environment.

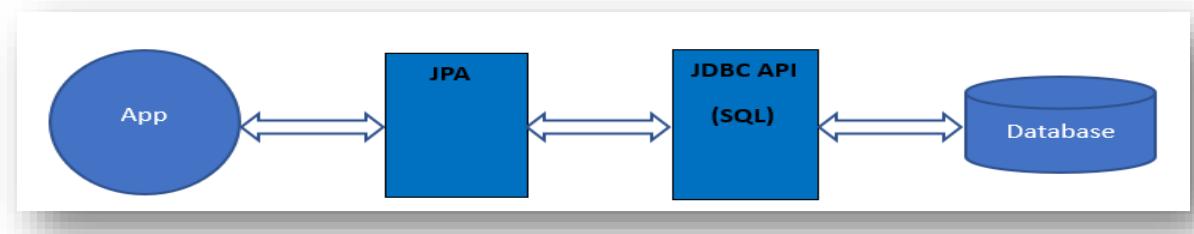
8.1 What is JPA?

JPA is an acronym for Java Persistence API. So this is an API that is used to persist objects in a Relational Database. The API allows programmers to interact with a Relational Database in an Object-Oriented manner. Through the API, objects (instances of classes) are mapped to tables, and their fields to columns of tables. The state of an object is stored as a row in tables.

Previously, when writing databased applications, we used the JDBC (Java Database Connectivity) API. The API required us to use SQL statements in order to interact with the database. The figure below shows the architecture we implemented.



The above architecture required programmers to explicitly write SQL statements in order to interact with relational databases. On the other hand, JPA shields programmers from writing SQL statements. Through JPA, programmers are given an object view of the database, not a relational one. Consequently programmers concern themselves with writing objects and allow the conversion of objects to SQL statements to be done behind the scenes by the persistence provider. The figure below shows the new architecture which includes JPA.



A persistence provider is a piece of code that works similar to a compiler. The provider implements JPA. Part of the task involves conversion of objects into SQL statements for interaction with the database. The provider also performs the reverse conversion where database results needs to be sent back to an application. In this case the provider will convert SQL statements into an object. There are many persistence providers out there, but the one we use is called **Hibernate**.

8.2 Annotations

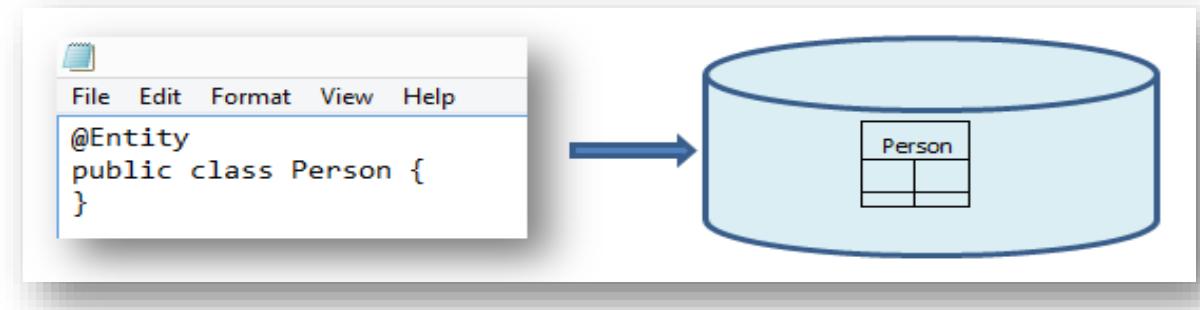
The classes and annotations used in JPA are located in a package called **javax.persistence**. Annotations are special instructions meant for execution by the container. They tell a container what to do with the resource that includes a particular annotation. Let us look at some of the annotations contained in the package.

8.2.1 @Entity

The **@Entity** annotation denotes that a class is an entity. An entity is a persistable object, meaning, an object that can be persisted (stored in a database).

Example

Let us create a Person entity that can be mapped to a Person table in the database.



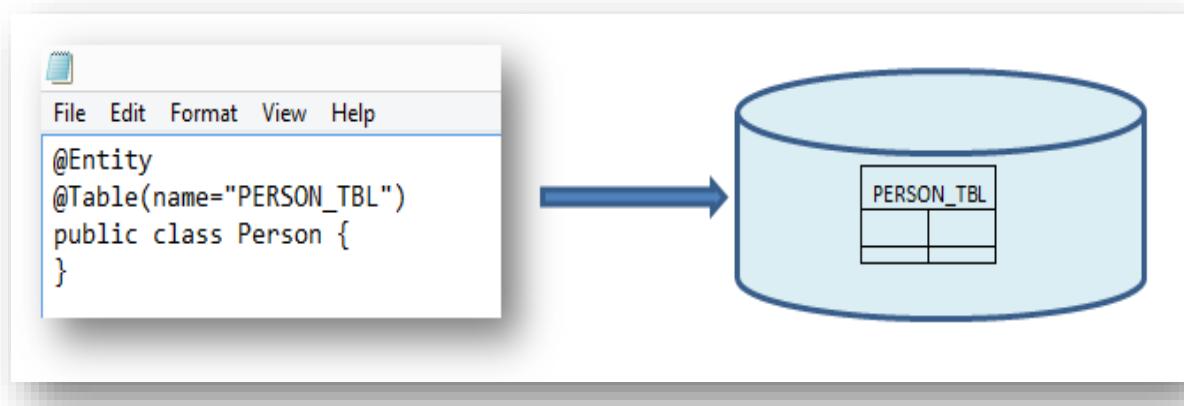
The **Person** class is an entity because of the **@Entity** annotation. The entity is mapped to the **Person** table in the database. This is the default state. A table takes the name of an entity.

8.2.2 @Table

The **@Table** annotation is used to customise a mapped table. The annotation allows a programmer to map an entity to a different table in the database. The annotation has an attribute called **name** that makes this customisation possible.

Example

Let us map the **Person** entity to the **Person_TBL** table in the database.



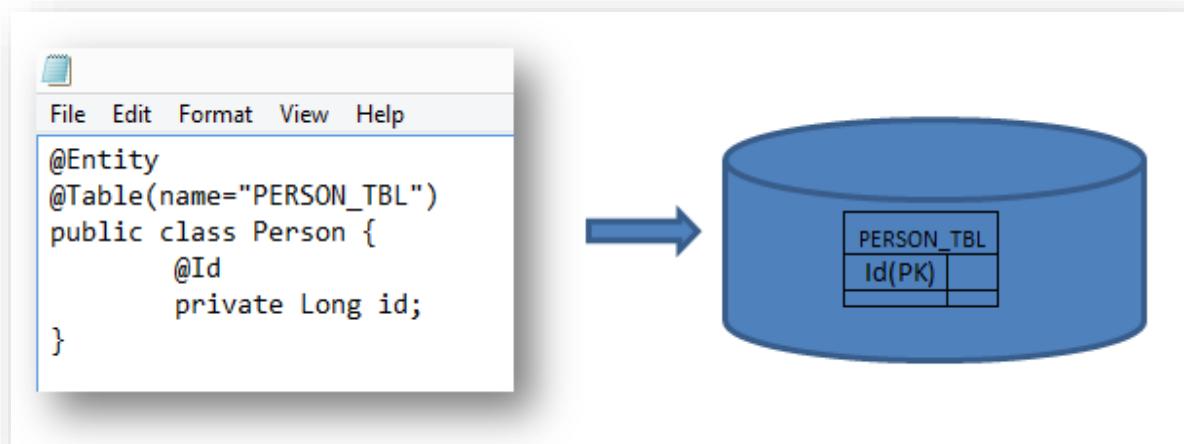
The **name** attribute of the **@Table** annotation allows us to give the name of the table we want to map our entity to in the database.

8.2.3 @Id

The **@Id** annotation is used to denote one of the instance fields of an entity as a primary key. The primary key is used to uniquely identify a record in a table.

Example

Let us create a **Person** entity with a primary key called **id**.



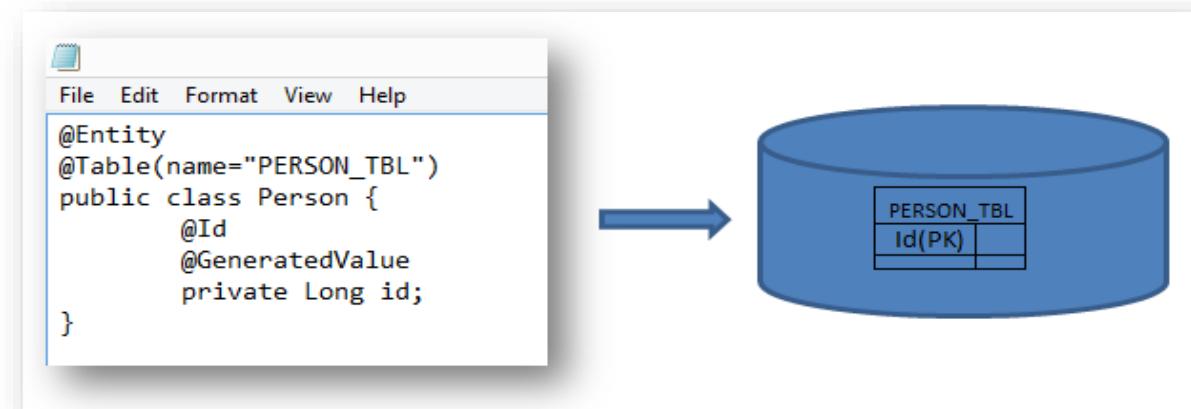
The **PERSON_TBL** table has a **column** called **id**, corresponding to the **id** field of the **Person entity**. The column will store the primary keys of the respective records.

8.2.4 @GeneratedValue

The **@GeneratedValue** annotation denotes that a primary key must be automatically generated by the persistence provider.

Example

Let us create a Person entity whose primary key should be automatically generated by the persistence provider.



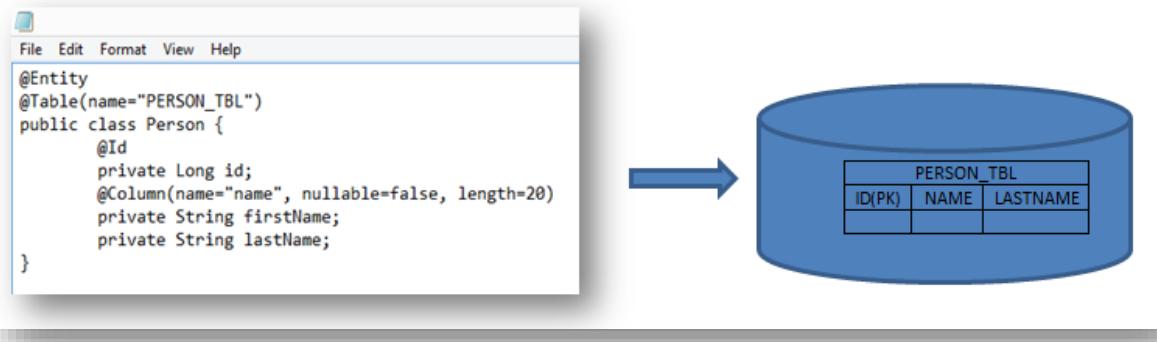
The primary keys in **PERSON_TBL** will be automatically generated. The programmer will not provide the values.

8.2.5 @Column

The **@Column** annotation is used to customise the columns of tables. The **@Column** annotation allows us to map a field name to a different column name. It also allows to state whether columns should accept null values or not, and also the size of columns.

Example

Let us create a **Person** entity that maps to a **PERSON_TBL** table in the database. The entity must have three fields, a primary key field called **id**; **firstName**; and **lastName**. The **firstName** field must be mapped to the **name** column in the table. The field must not be nullable and its maximum length must be 20.



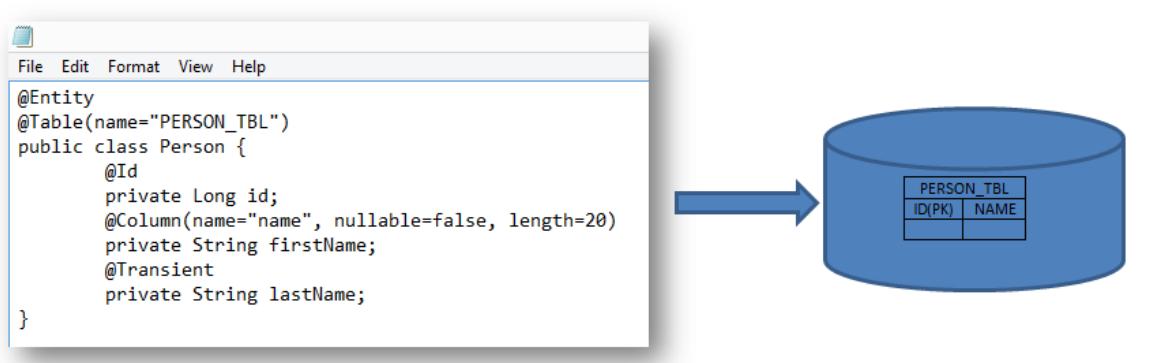
As can be seen, the **id** and **lastName** fields are mapped to the same column names in the table. But the **firstName** field is mapped to the customised **name** column.

8.2.6 @Transient

The **@Transient** annotation denotes that an entity's field must not be persisted in the database.

Example

Let us create a **Person** entity that maps to a **PERSON_TBL** table in the database. The entity must have three fields, a primary key field called **id**; **firstName**; and **lastName**. The **firstName** field must be mapped to the **name** column in the table. The **lastName** field must not be nullable and its maximum length must be 20. The **lastName** must not be persisted in the database.



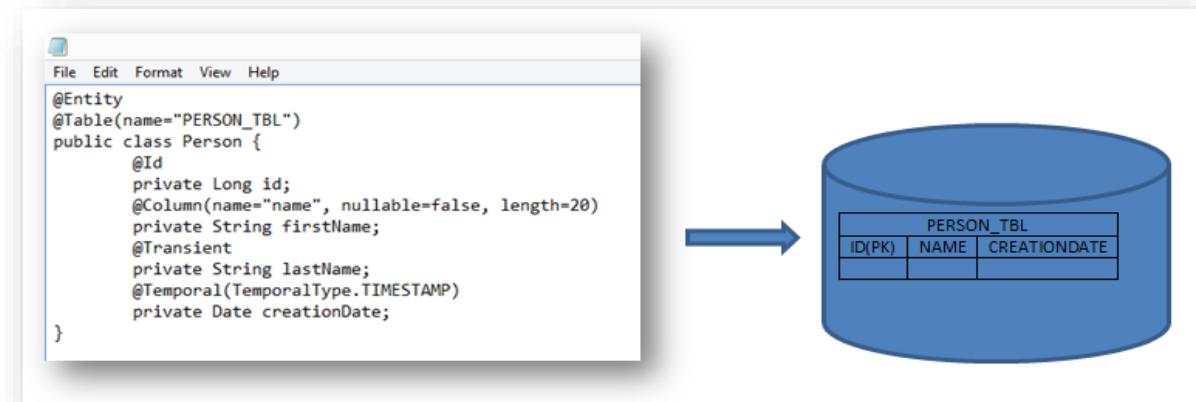
As can be seen, there's no column for the **lastName** field since it is transient.

8.2.7 @Temporal

The **@Temporal** annotation is used to used to persist **date**, **time** and **timestamp**.

Example

Let us create a **Person** entity that maps to a **PERSON_TBL** table in the database. The entity must have four fields, a primary key field called **id**; **firstName**; **lastName**; and **creation date**. The **firstName** field must be mapped to the name column in the table. The field must not be nullable and its maximum length must be 20. The **lastName** must not be persisted into the database. The creation date must be the **timestamp**.



The time stamp is possible through the usage of **TemporalType.TIMESTAMP** as an attribute value of the **@Temporal** annotation. If we want just date, then we would use **TemporalType.DATE**, and for time only we use **TemporalType.TIME**.

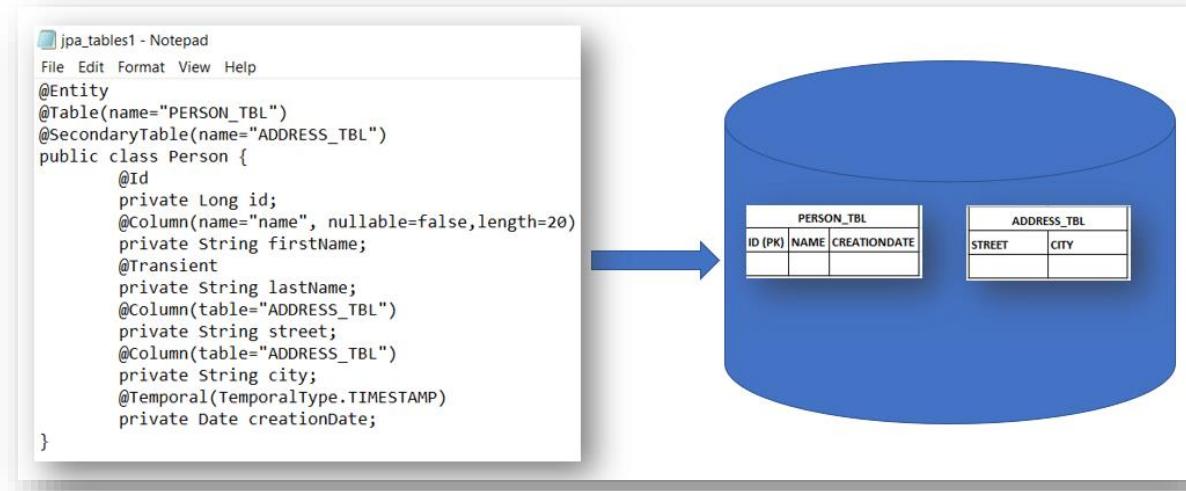
8.2.8 @SecondaryTable

The **@SecondaryTable** annotation is used when we want to map some of the entity's fields to another table.

Example

Let us create a Person entity that maps to a **PERSON_TBL** table in the database. The entity must have five fields, a primary key field called **id**; **firstName**; **lastName**; **creation date**; **street**; and **city**. The **firstName** field must be mapped to the name column in the table. The field must not be nullable and its maximum length must be 20. The **lastName** must not be persisted into the database. The creation date should

be the timestamp. The street and city fields must be persisted to a table called **ADDRESS_TBL**.



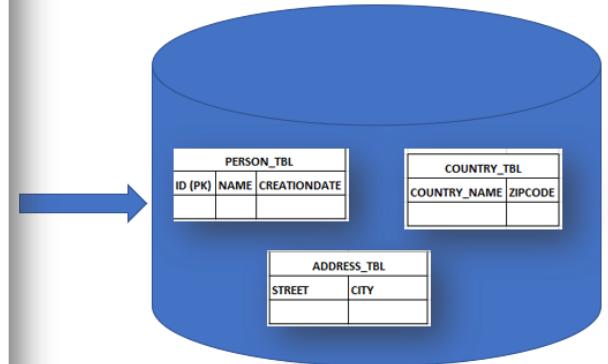
8.2.9 @SecondaryTables

The **@SecondaryTables** annotation is used for mapping some of the entity's fields to other tables.

Example

Let us create a Person entity that maps to a **PERSON_TBL** table in the database. The entity must have seven fields, a primary key field called **id**; **firstName**; **lastName**; **creation date**; **street**; **city**; and **code**. The **firstName** field must be mapped to the **name** column in the table. The field must not be nullable and its maximum length must be 20. The **lastName** must not be persisted into the database. The creation date should be the timestamp. The street, city and code fields must be persisted in a second table called **ADDRESS_TBL**. The **countryName** and **zipCode** fields must be stored in a table called **COUNTRY_TBL**.

```
jpa_tables2 - Notepad
File Edit Format View Help
@Entity
@Table(name="PERSON_TBL")
@SecondaryTables({
    @SecondaryTable(name="ADDRESS_TBL"),
    @SecondaryTable(name="COUNTRY_TBL")
})
public class Person {
    @Id
    private Long id;
    @Column(name="name", nullable=false,length=20)
    private String firstName;
    @Transient
    private String lastName;
    @Column(table="ADDRESS_TBL")
    private String street;
    @Column(table="ADDRESS_TBL")
    private String city;
    @Column(table="COUNTRY_TBL")
    private String countryName;
    @Column(table="COUNTRY_TBL")
    private String zipCode;
    @Temporal(TemporalType.TIMESTAMP)
    private Date creationDate;
}
```



8.3 Implementation of JPA annotations

8.3.1 Using basic annotations

Example

Create a web application that will perform **CRUD** (Create Read Update Delete) operations on a classlist. The classlist contains students. A student has an **id**, **name**, **surname**, and **gender**. The **id** is automatically generated by the persistence provider. The table below shows all the fields of a student and the constraints that must be applied to each field.

Field	Constraint
id	Primary key. Must be automatically generated by the persistence provider.
name	Must be a string. Can be nullable. The maximum length is 20.
surname	Must be a string. Can be nullable. The maximum length is 20.
gender	Must be a string of length 1. Can be nullable. The maximum length is 1.

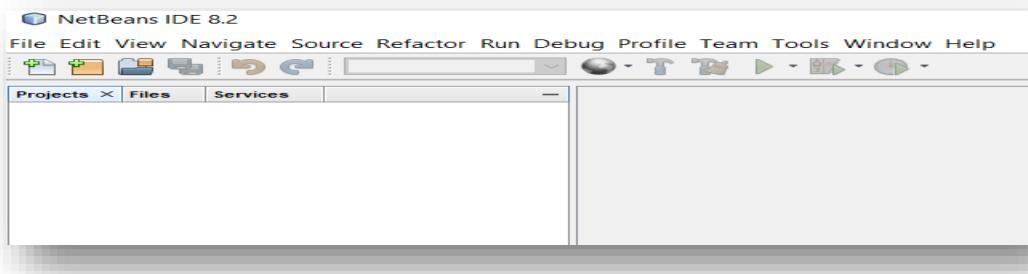
Solution approach

The solution to this problem is going to be done in five parts, namely:

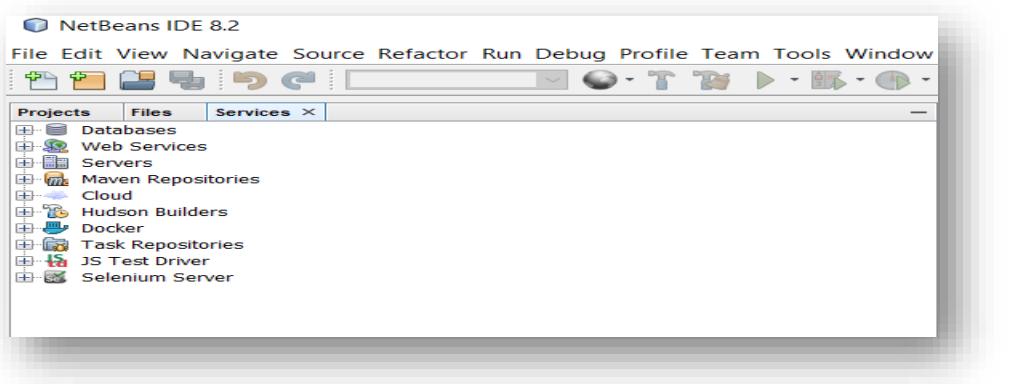
- **Part A:** we create a database.
- **Part B:** we establish connection between the created database and GlassFish.
- **Part C:** we create an EJB module. The module will have an entity with business logic.
- **Part D:** we create a client web application.
- **Part E:** we run the client web application.

Part A - Create a database

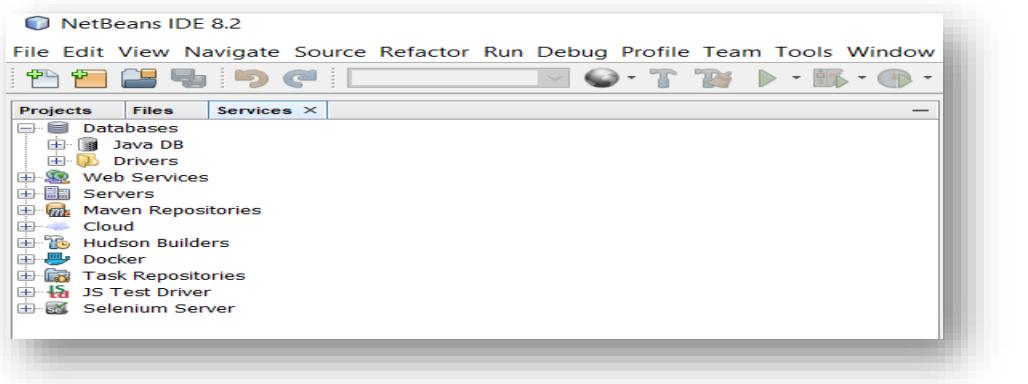
Launch NetBeans.



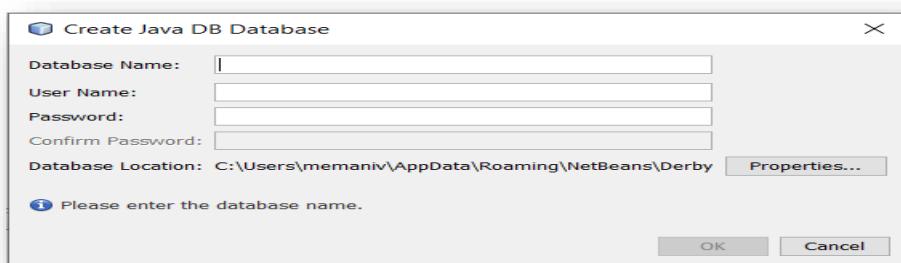
Click on the **Services** tab.



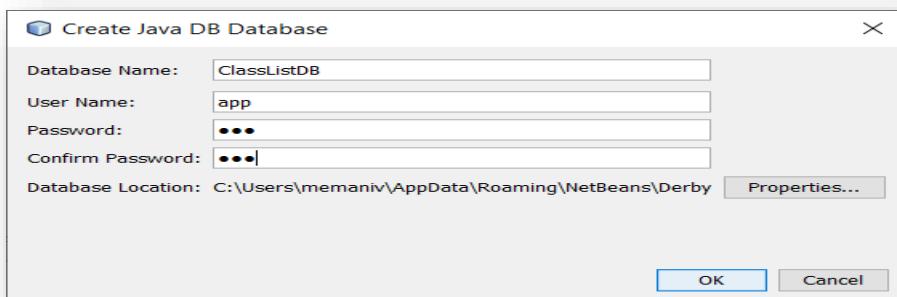
Expand the **Databases** option.



Right-click on **Java DB** and select **Create Database**.

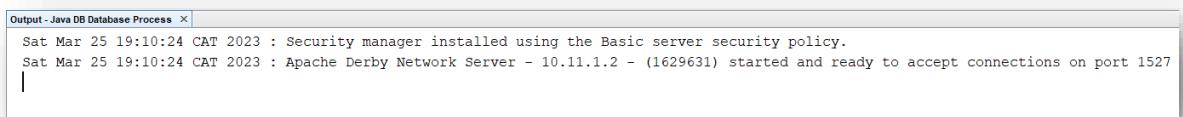


Fill-in the form. I made the password to be **123**.

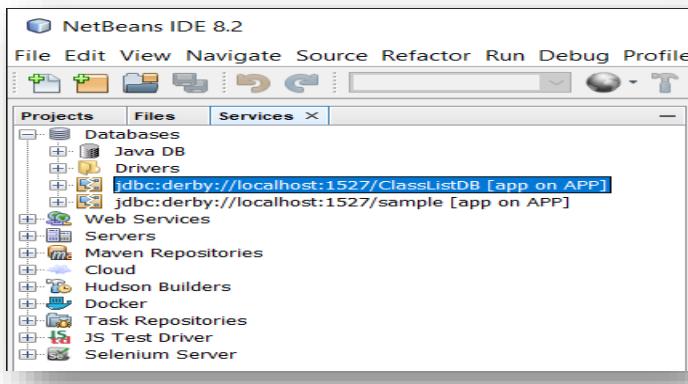


Click **OK**.

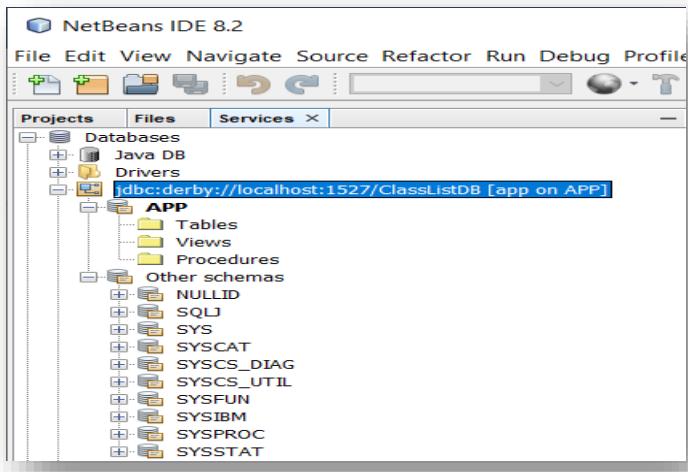
- ✓ An output message confirming that **Derby** has started and ready to accept connections on port **1527** is shown.



- ✓ A database is created.

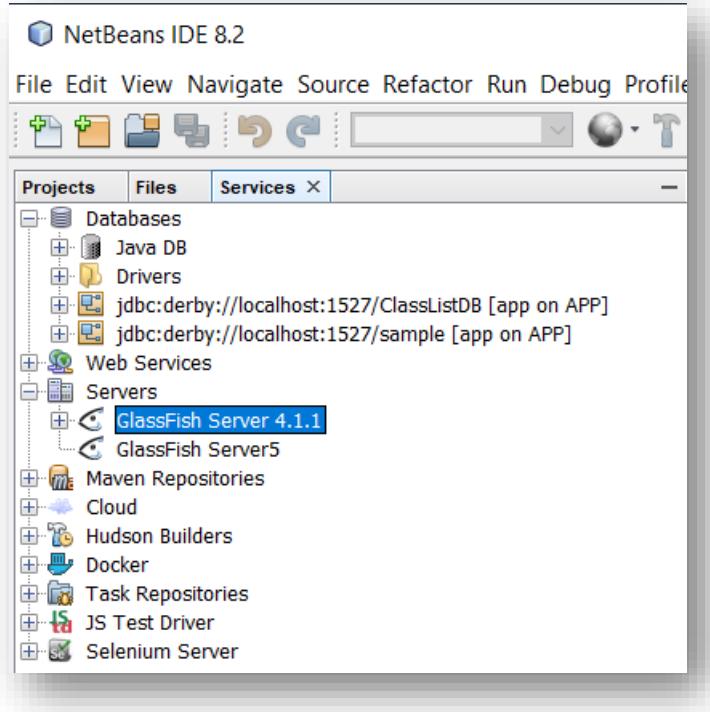


Right-click on the database and select **Connect**.

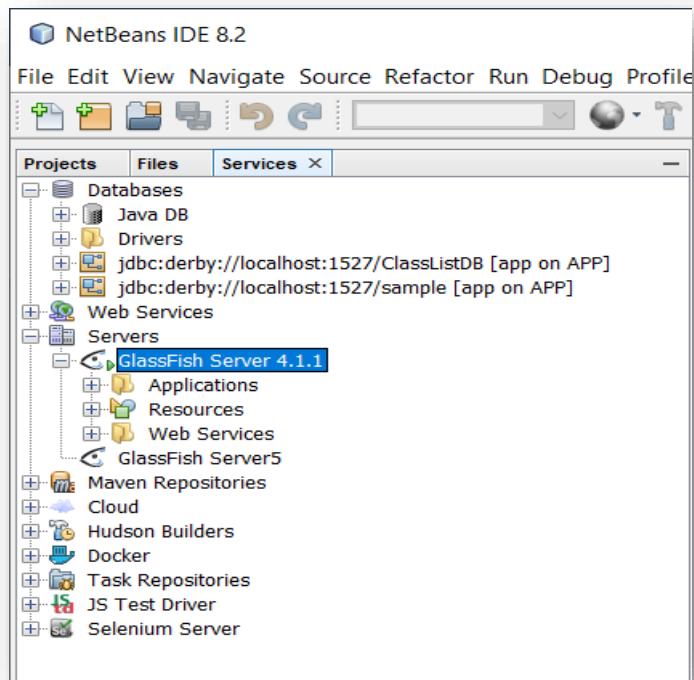


Part B – Connect the database to the application server.

Click on the **Services** tab.



Start the server.



Right-click on the server and select **View Domain Admin Console**.

The screenshot shows the GlassFish Server Open Source Edition Common Tasks console. The URL in the address bar is `http://localhost:4848/common/index.jsf`. The top navigation bar includes links for Home, About..., User: admin | Domain: domain1 | Server: localhost, and the GlassFish Server Open Source Edition logo. The left sidebar, titled "Common Tasks", contains the following categories and sub-items:

- Domain
- server (Admin Server)
- Clusters
- Standalone Instances
- Nodes
- Applications
- Lifecycle Modules
- Monitoring Data
- Resources
 - Concurrent Resources
 - Connectors
 - JDBC
 - JMS Resources
 - JNDI
 - JavaMail Sessions
 - Resource Adapter Configs
- Configurations
 - default-config
 - server-config
- Update Tool

The main content area is titled "GlassFish Console - Common Tasks" and includes sections for GlassFish News, Deployment (List Deployed Applications, Deploy an Application), Administration (Change Administrator Password, List Password Aliases), and Monitoring (Monitoring Data).

Under **Common Tasks** panel, expand **JDBC**.

The screenshot shows the GlassFish Server Open Source Edition JDBC console. The URL in the address bar is `http://localhost:4848/common/index.jsf`. The top navigation bar includes links for Home, About..., User: admin | Domain: domain1 | Server: localhost, and the GlassFish Server Open Source Edition logo. The left sidebar, titled "Common Tasks", contains the following categories and sub-items, with the "JDBC" item selected and highlighted in blue:

- Domain
- server (Admin Server)
- Clusters
- Standalone Instances
- Nodes
- Applications
- Lifecycle Modules
- Monitoring Data
- Resources
 - Concurrent Resources
 - Connectors
 - JDBC
 - JDBC Resources
 - JDBC Connection Pools
 - JMS Resources
 - JNDI
 - JavaMail Sessions
 - Resource Adapter Configs
- Configurations
 - default-config
 - server-config
- Update Tool

The main content area is titled "JDBC" and displays two sub-sections: "JDBC Resources" and "JDBC Connection Pools".

Modify the Connection pool to include the **ClassListDB**.

- ✓ Click on **JDBC Connection Pools**.

The screenshot shows a table titled "Pools (3)" with columns: Select, Pool Name, Resource Type, Classname, and Description. The entries are:

Select	Pool Name	Resource Type	Classname	Description
<input type="checkbox"/>	DerbyPool	javax.sql.DataSource	org.apache.derby.jdbc.ClientDataSource	
<input type="checkbox"/>	SamplePool	javax.sql.DataSource	org.apache.derby.jdbc.ClientDataSource	
<input type="checkbox"/>	TimerPool	javax.sql.XADatasource	org.apache.derby.jdbc.EmbeddedXADataSource	

- ✓ Click on the **DerbyPool** link.

The screenshot shows the "Edit JDBC Connection Pool" page for DerbyPool. It has tabs for General, Advanced, and Additional Properties. The General tab is selected.

General Settings

- Pool Name:** DerbyPool
- Resource Type:** javax.sql.DataSource
- Datasource Classname:** org.apache.derby.jdbc.ClientDataSource
- Driver Classname:** (empty)
- Ping:** Enabled
- Deployment Order:** 100
- Description:** (empty)

Pool Settings

- Initial and Minimum Pool Size:** 8 Connections
- Maximum Pool Size:** 32 Connections
- Pool Resize Quantity:** 2 Connections

- ✓ Click on **Additional Properties**.

The screenshot shows the "Edit JDBC Connection Pool Properties" page for DerbyPool. It has tabs for General, Advanced, and Additional Properties. The Additional Properties tab is selected.

Pool Name: DerbyPool

Additional Properties (6)

Select	Name	Value
<input type="checkbox"/>	PortNumber	1527
<input type="checkbox"/>	Password	APP
<input type="checkbox"/>	User	APP
<input type="checkbox"/>	serverName	localhost
<input type="checkbox"/>	DatabaseName	sun-appserv-samples
<input type="checkbox"/>	connectionAttributes	;create=true

- ✓ Click on **Add Property** and add a **url** property with the following value:

jdbc:derby://localhost:1527/ClassListDB

- ✓ Delete the **connectionAttributes** property.

Select	Name	Value
<input type="checkbox"/>	url	jdbc:derby://localhost:1527/ClassListDB
<input type="checkbox"/>	PortNumber	1527
<input type="checkbox"/>	Password	123
<input type="checkbox"/>	User	app
<input type="checkbox"/>	serverName	localhost
<input type="checkbox"/>	DatabaseName	ClassListDB

- ✓ Save the changes by clicking the **Save** button.

Select	Name	Value
<input type="checkbox"/>	PortNumber	1527
<input type="checkbox"/>	Password	123
<input type="checkbox"/>	User	app
<input type="checkbox"/>	serverName	localhost
<input type="checkbox"/>	DatabaseName	ClassListDB
<input type="checkbox"/>	url	jdbc:derby://localhost:1527/ClassListDB

- ✓ Check if the created connection pool is working. Do this by clicking the **Ping** button.

Confirm that the resource points to the DerbyPool.

- ✓ Still under **JDBC**, click on **JDBC Resources**.

Resources (3)				
Select	JNDI Name	Logical JNDI Name	Enabled	Connection Pool
<input type="checkbox"/>	jdbc/_TimerPool	_TimerPool	<input checked="" type="checkbox"/>	TimerPool
<input type="checkbox"/>	jdbc/_default	java.comp/DefaultDataSource	<input checked="" type="checkbox"/>	DerbyPool
<input type="checkbox"/>	jdbc/sample	SamplePool	<input checked="" type="checkbox"/>	SamplePool

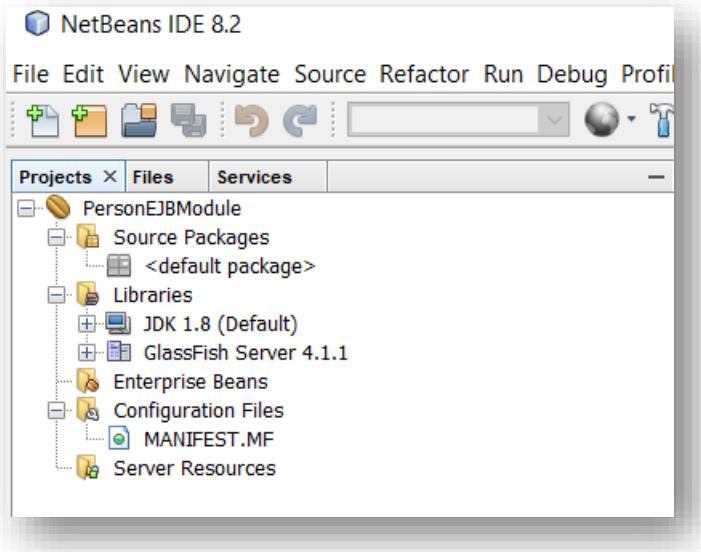
Take note of the purpose of JDBC Resources. It is said:

"JDBC resources provide applications with a means to connect to a database."

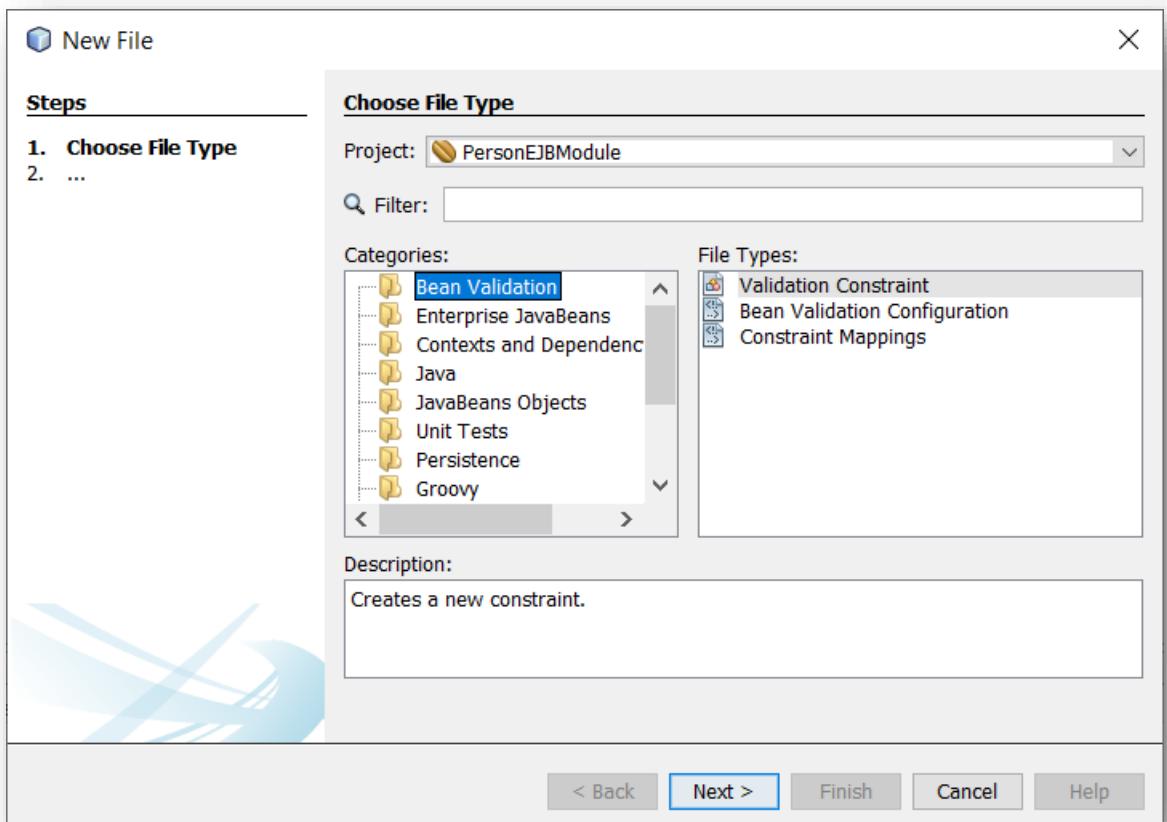
You can also see that the **jdbc/_default** resource is associated with the **DerbyPool** Connection Pool.

Part C – Create an entity.

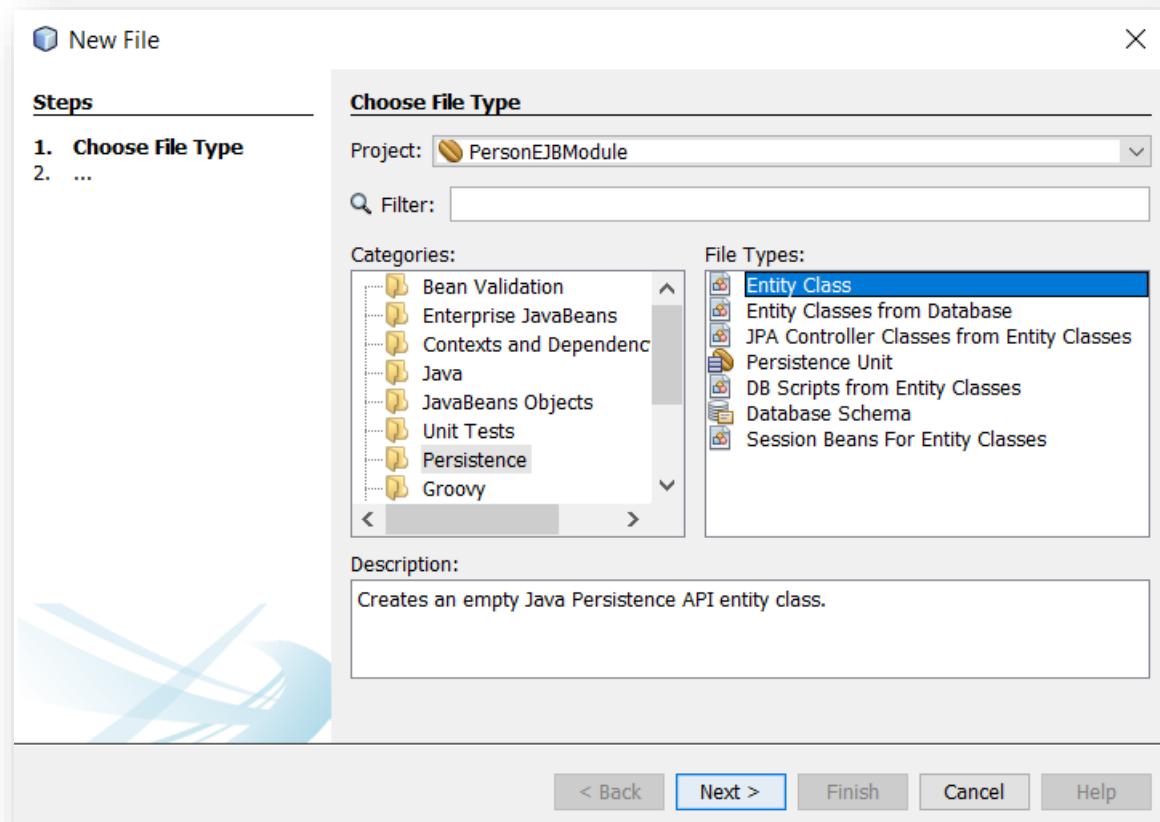
Create an EJB project called **PersonEJBModule**.



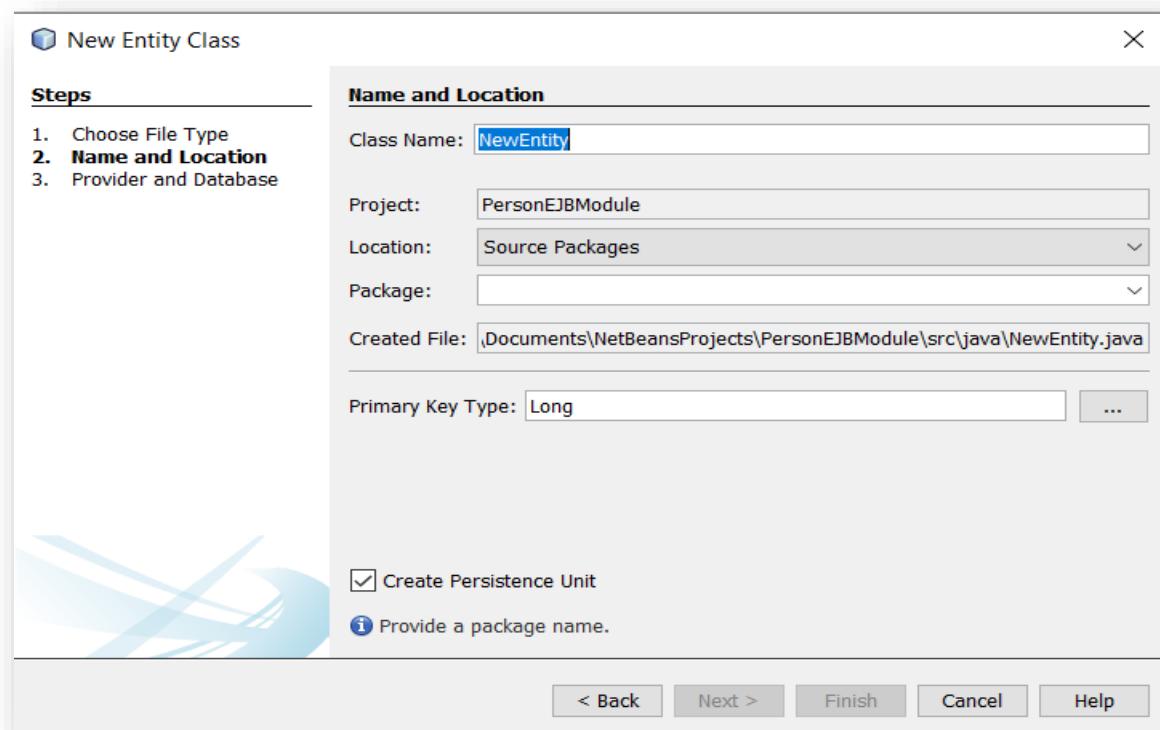
Right-click on the project and select **New | Other**.



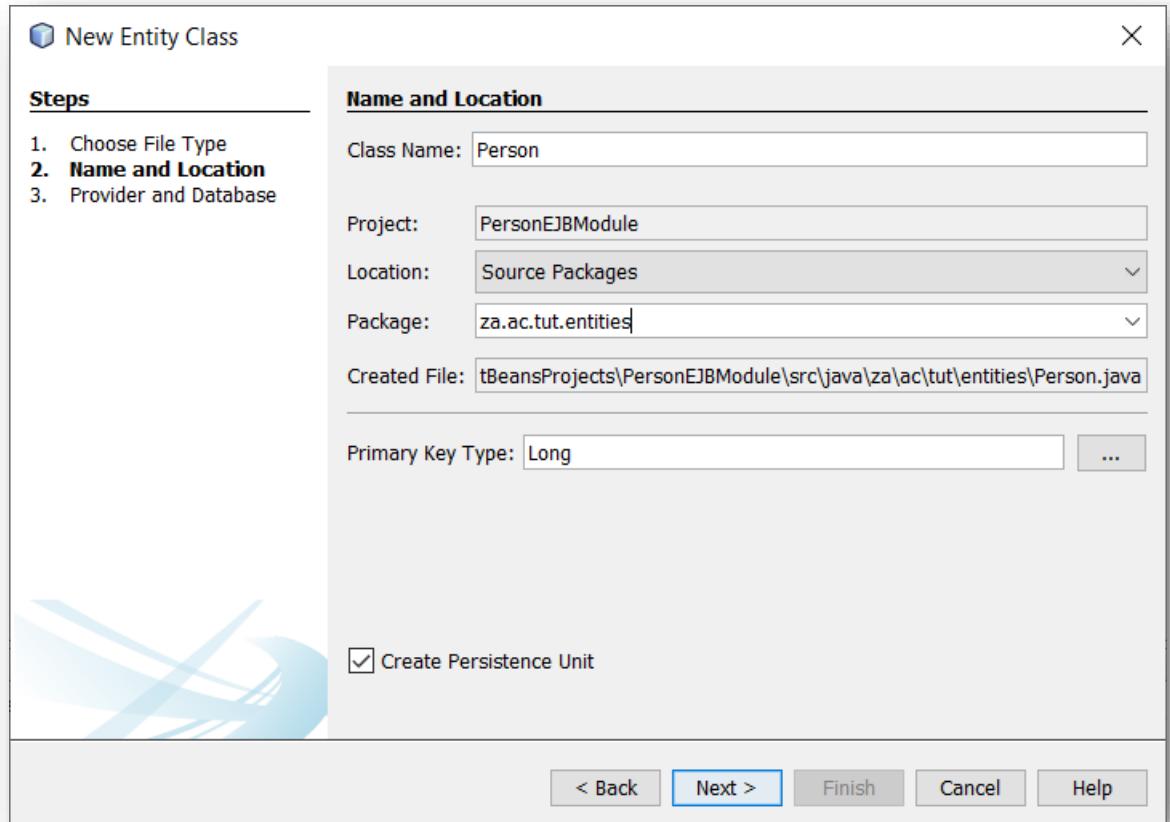
Under **Categories** select **Persistence**, and under **File Types** select **Entity Class**.



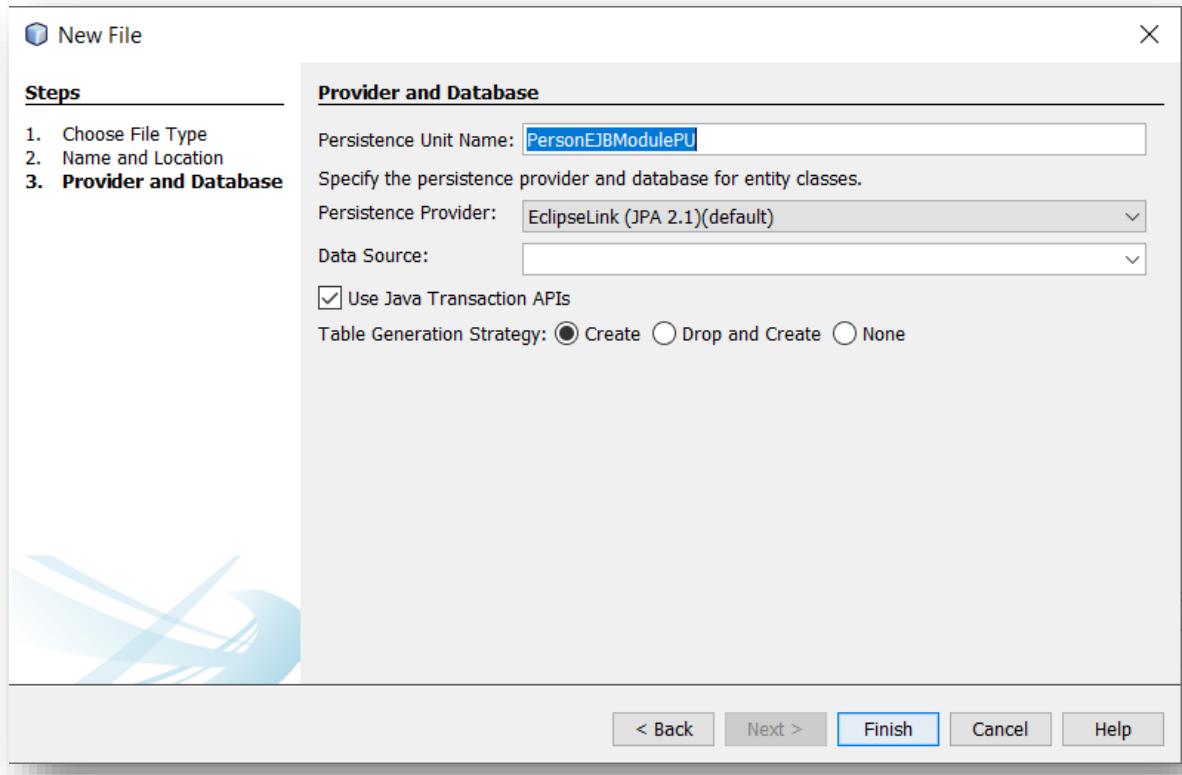
Click **Next**.



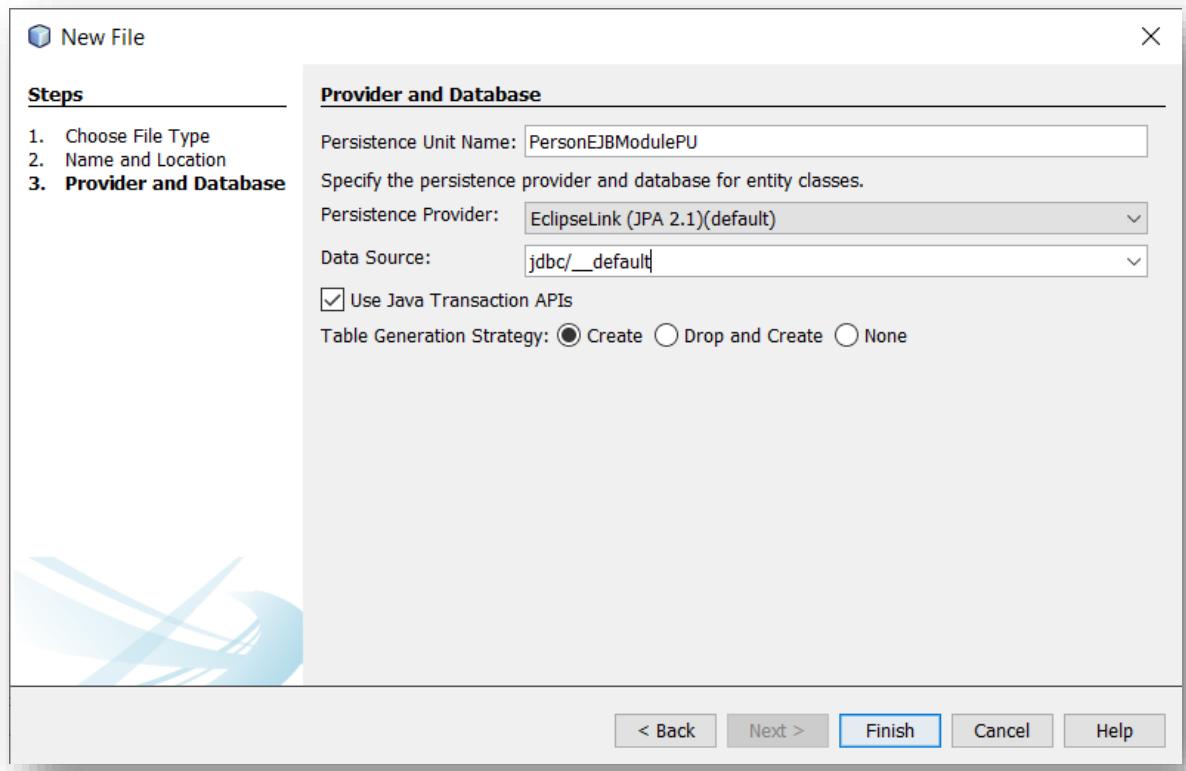
Fill-in the form.



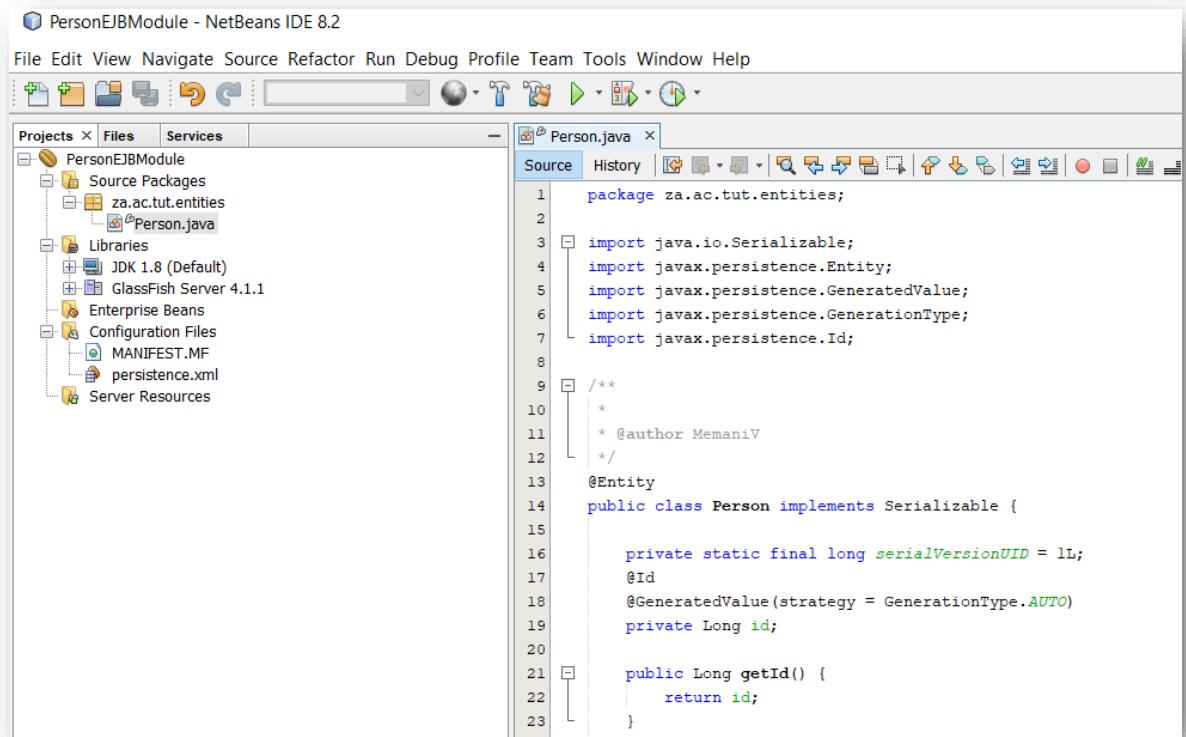
Click **Next**.



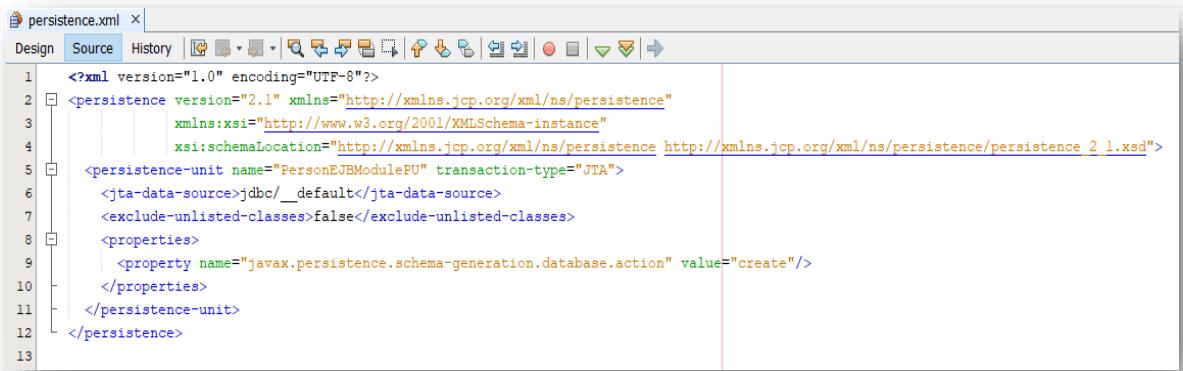
Under **Data Source**, select **jdbc/_default**.



Click **Finish**.



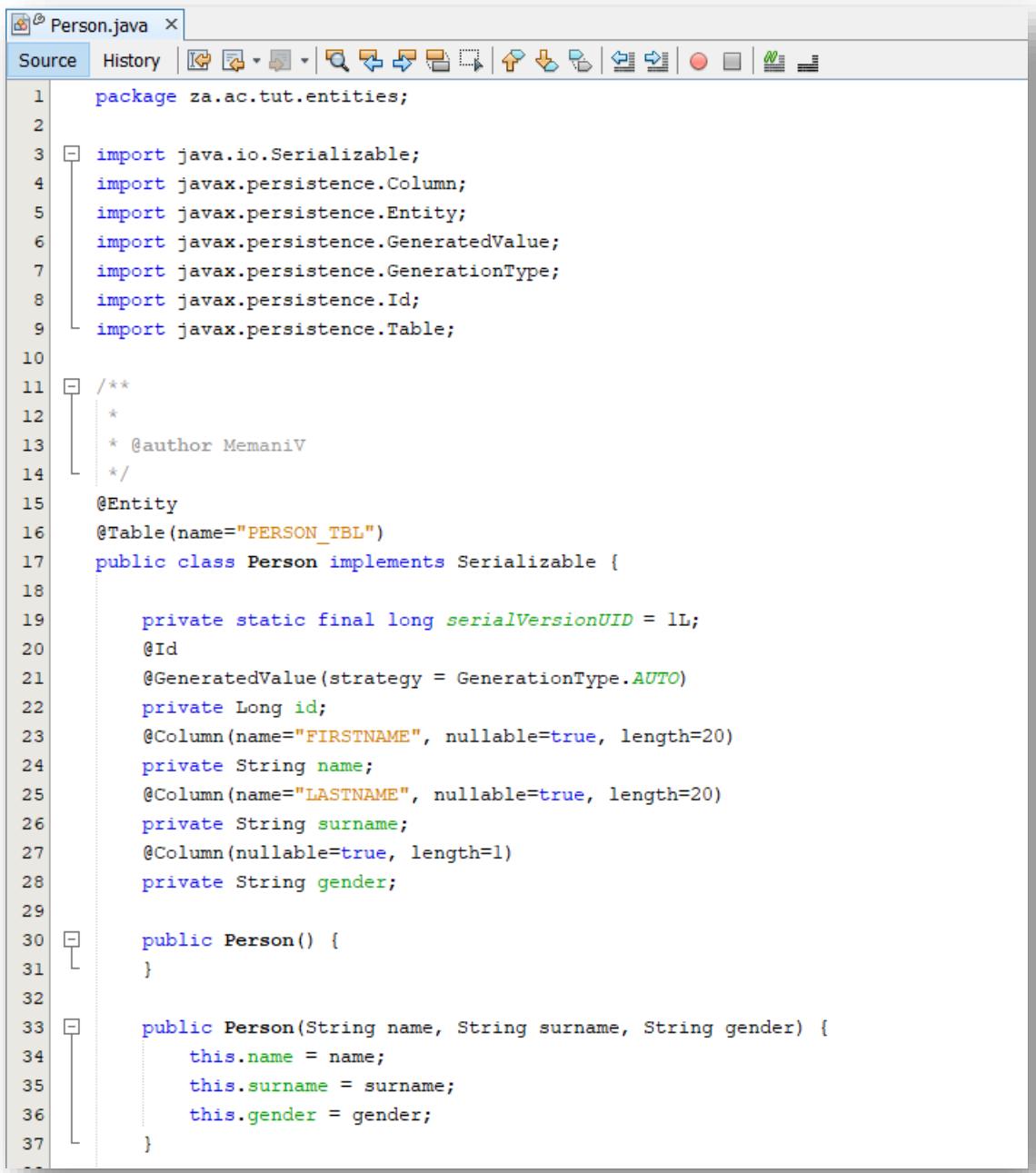
View the **persistence.xml** file.



The screenshot shows the Eclipse IDE interface with the 'persistence.xml' file open in the editor. The tab bar at the top has 'Design', 'Source' (which is selected), and 'History'. Below the tabs is a toolbar with various icons. The code editor displays the XML configuration for the persistence unit:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
    <persistence-unit name="PersonEJBModulePU" transaction-type="JTA">
        <jta-data-source>jdbc/_default</jta-data-source>
        <exclude-unlisted-classes>false</exclude-unlisted-classes>
        <properties>
            <property name="javax.persistence.schema-generation.database.action" value="create"/>
        </properties>
    </persistence-unit>
</persistence>
```

Write complete source code for the entity.



The screenshot shows the Eclipse IDE interface with the 'Person.java' file open in the editor. The tab bar at the top has 'Source' (selected) and 'History'. Below the tabs is a toolbar with various icons. The code editor displays the Java entity class:

```
package za.ac.tut.entities;

import java.io.Serializable;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

/**
 * @author MemaniV
 */
@Entity
@Table(name="PERSON_TBL")
public class Person implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    @Column(name="FIRSTNAME", nullable=true, length=20)
    private String name;
    @Column(name="LASTNAME", nullable=true, length=20)
    private String surname;
    @Column(nullable=true, length=1)
    private String gender;

    public Person() {
    }

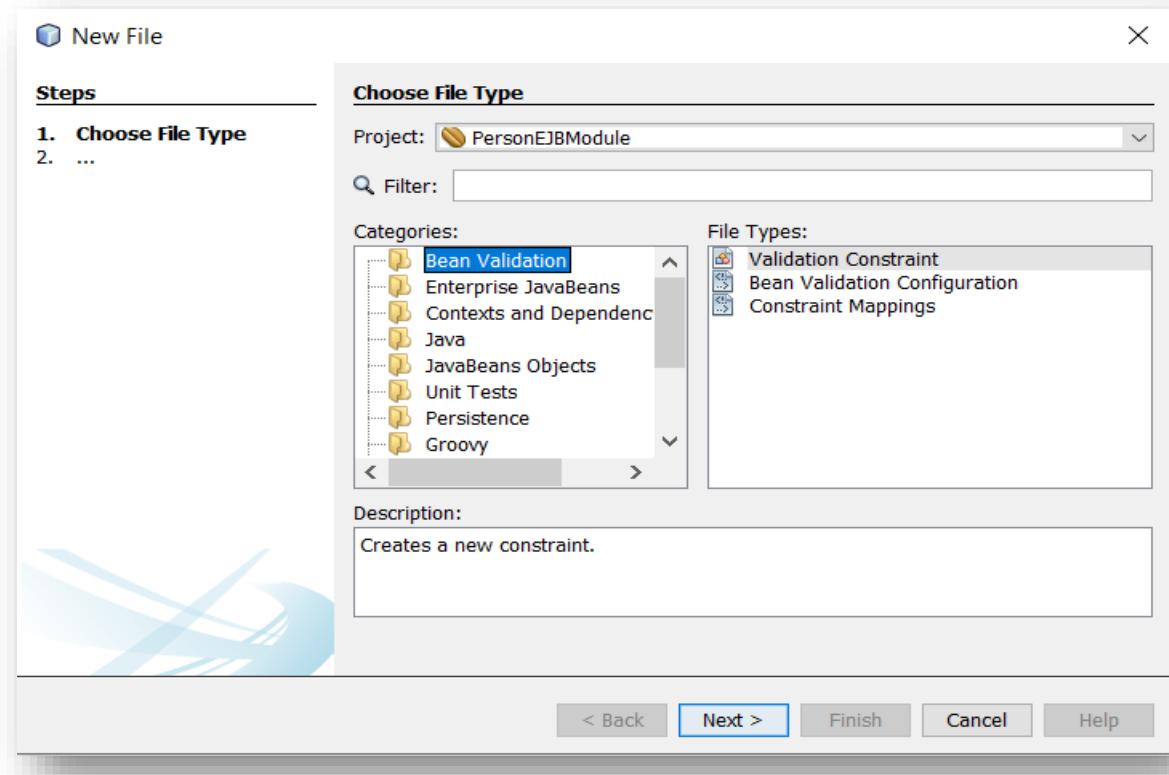
    public Person(String name, String surname, String gender) {
        this.name = name;
        this.surname = surname;
        this.gender = gender;
    }
}
```

```
39  public String getName() {
40      return name;
41  }
42
43  public void setName(String name) {
44      this.name = name;
45  }
46
47  public String getSurname() {
48      return surname;
49  }
50
51  public void setSurname(String surname) {
52      this.surname = surname;
53  }
54
55  public String getGender() {
56      return gender;
57  }
58
59  public void setGender(String gender) {
60      this.gender = gender;
61  }
62
63  public Long getId() {
64      return id;
65  }
66
67  public void setId(Long id) {
68      this.id = id;
69  }
```

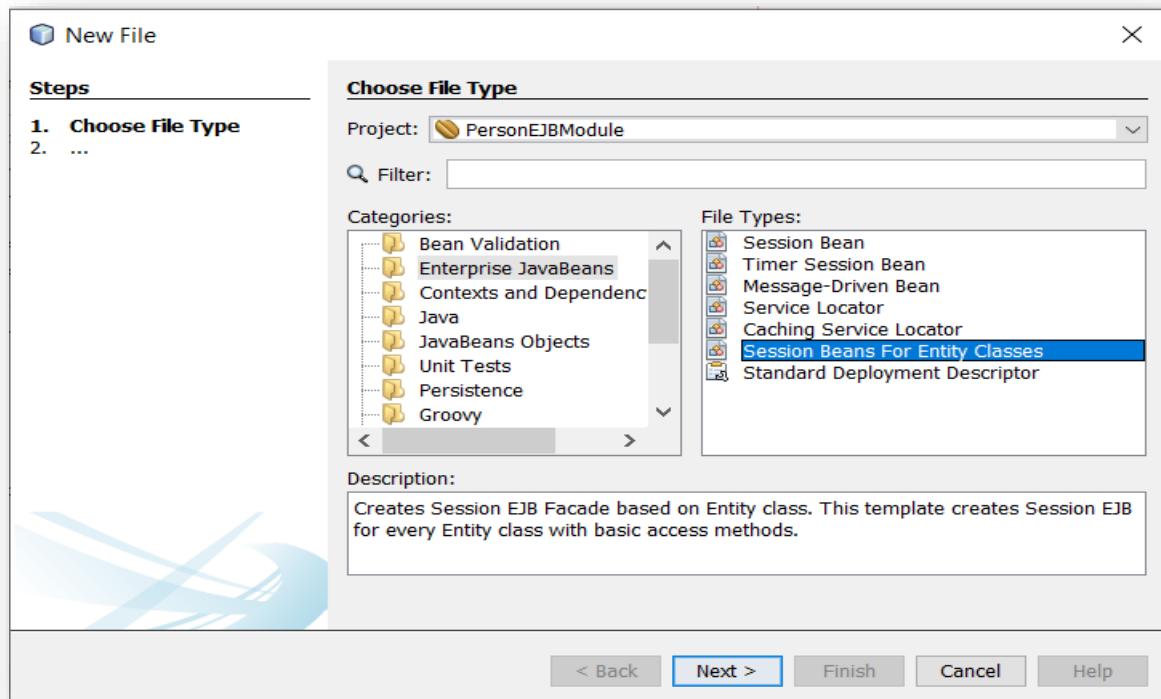
```
71  @Override
72  public int hashCode() {
73      int hash = 0;
74      hash += (id != null ? id.hashCode() : 0);
75      return hash;
76  }
77
78  @Override
79  public boolean equals(Object object) {
80      if (!(object instanceof Person)) {
81          return false;
82      }
83      Person other = (Person) object;
84      if ((this.id == null && other.id != null) ||
85          (this.id != null && !this.id.equals(other.id))) {
86          return false;
87      }
88      return true;
89  }
90
91  @Override
92  public String toString() {
93      return "za.ac.tut.entities.Person[ id=" + id + " ]";
94  }
95
96 }
```

Create business code (CRUD operations) for the entity. Perform the following steps:

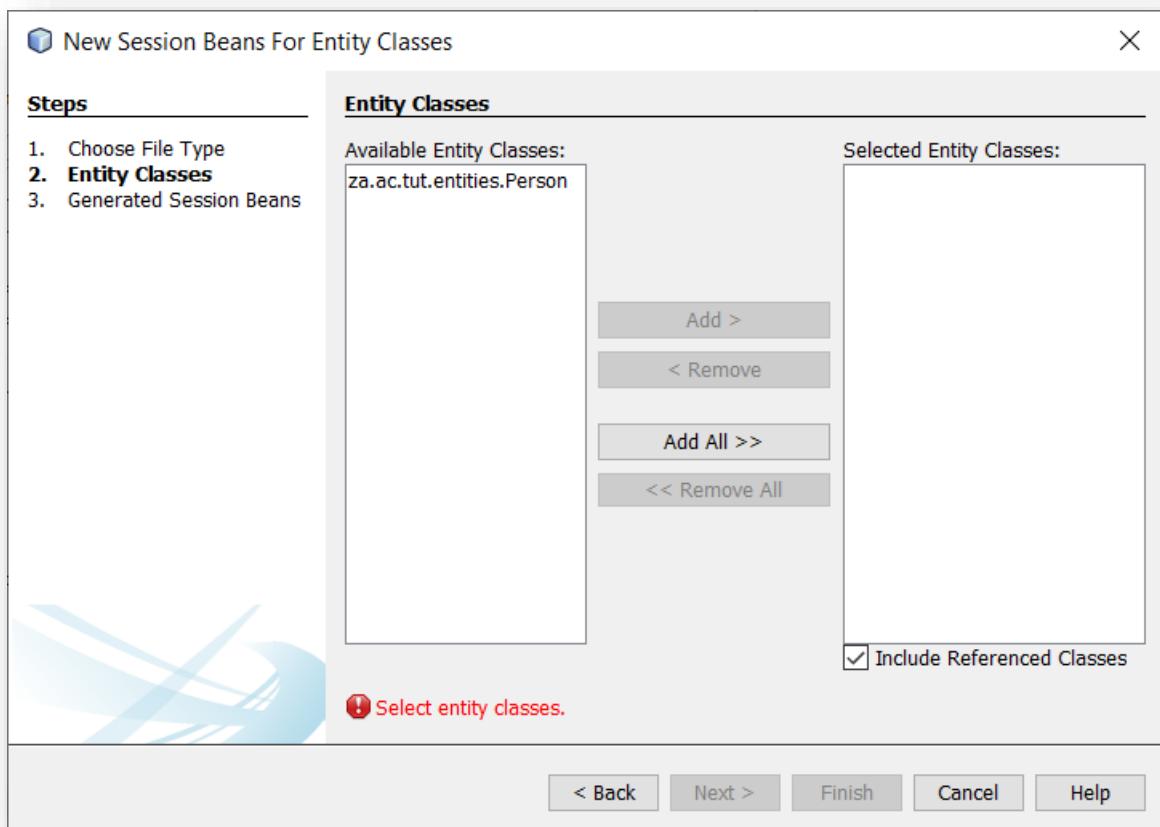
- ✓ Right-click on the project and select **New | Other**.



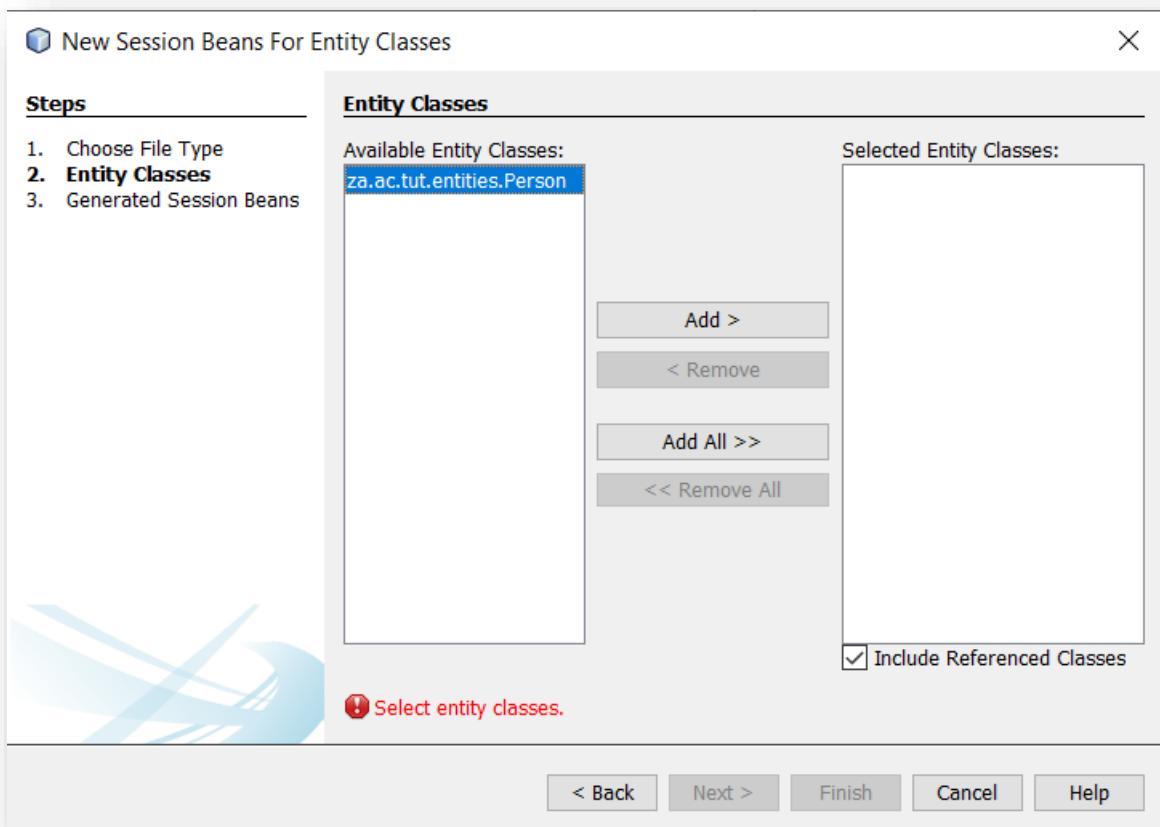
- ✓ Under **Categories**, select **Enterprise JavaBeans**. Under **File Types** select **Session Beans for Entity Classes**.



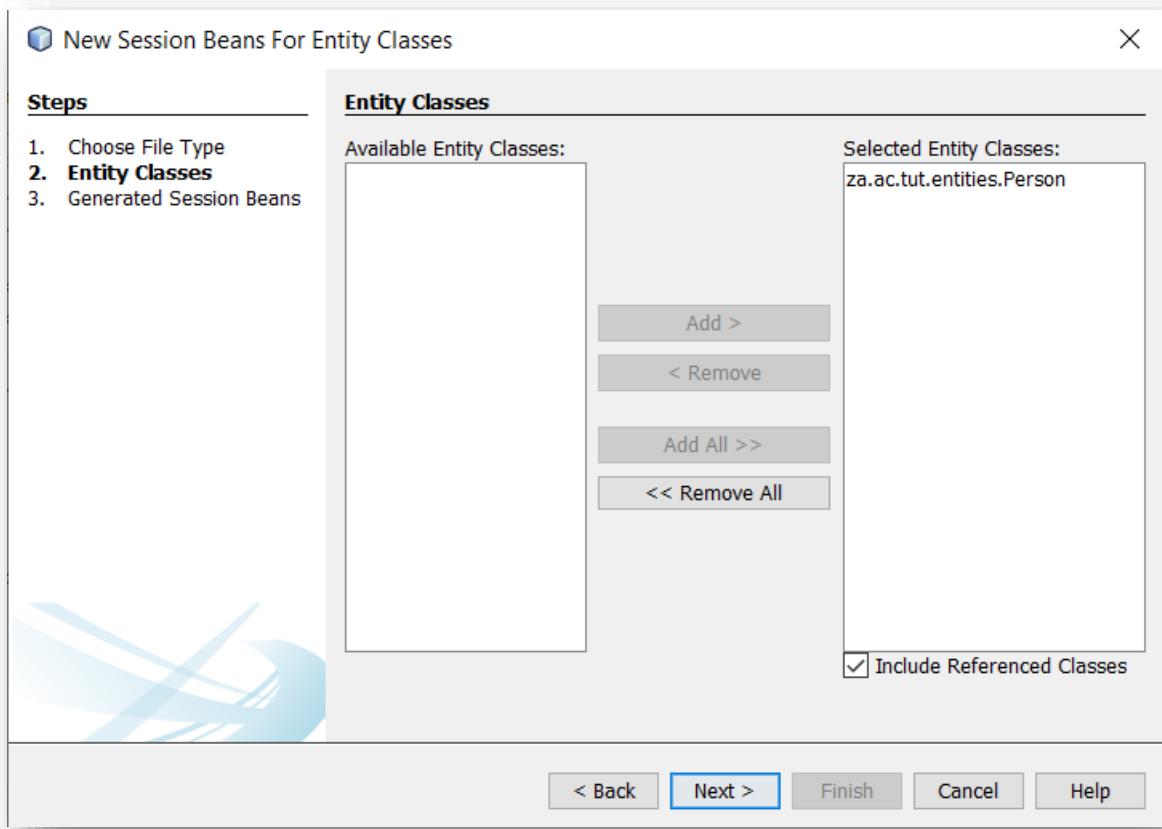
✓ Click **Next**.



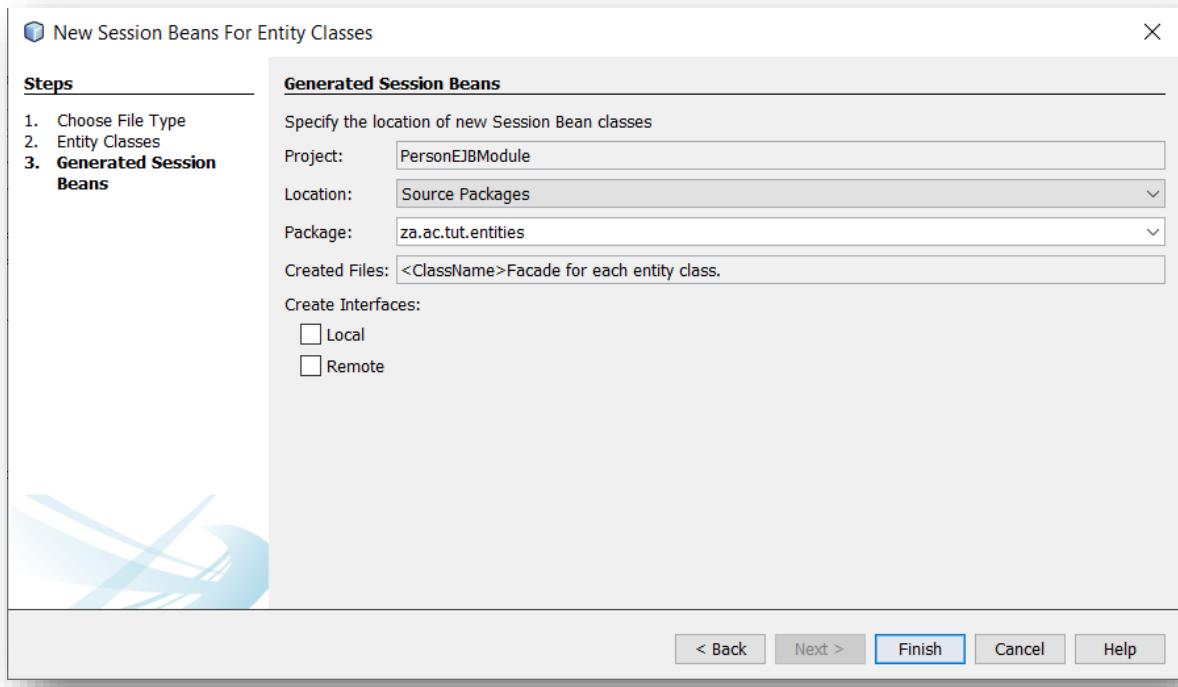
✓ Select **za.ac.tut.entities** under Available Entity Classes.



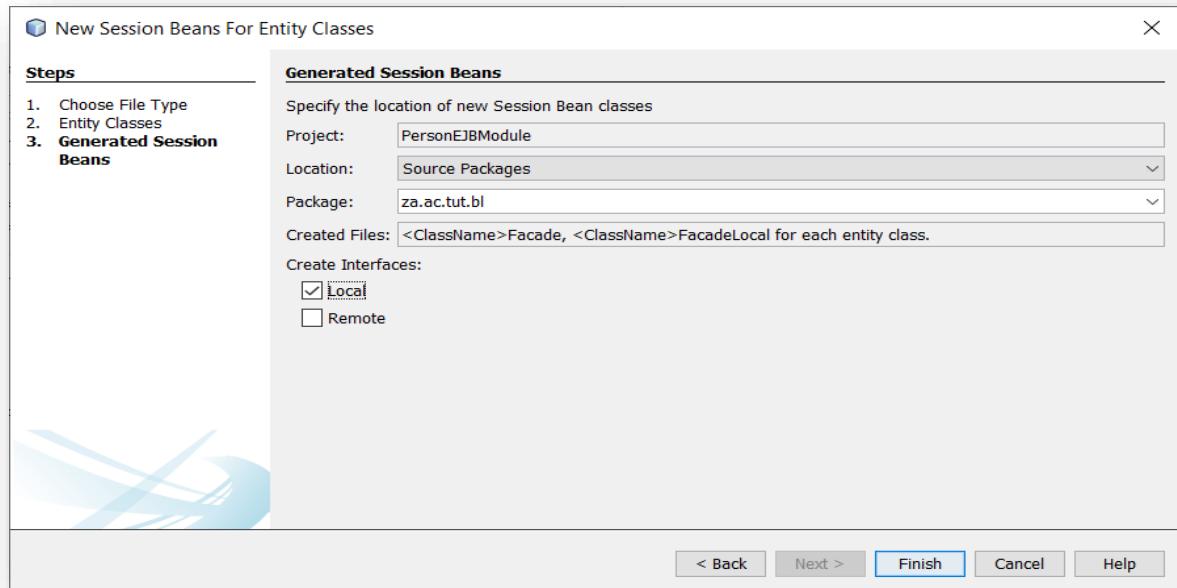
✓ Click Add.



✓ Click Next.



- ✓ Modify the package name to **za.ac.tut.bl**, where **bl** stands for **business logic**. Also select **Local** interface.



- ✓ Click **Finish**. View created files.
 - **PersonFacade**.

```

1 package za.ac.tut.bl;
2
3 import javax.ejb.Stateless;
4 import javax.persistence.EntityManager;
5 import javax.persistence.PersistenceContext;
6 import za.ac.tut.entities.Person;
7
8 /**
9  * 
10 * @author MemaniV
11 */
12 @Stateless
13 public class PersonFacade extends AbstractFacade<Person> implements PersonFacadeLocal {
14
15     @PersistenceContext(unitName = "PersonEJBModulePU")
16     private EntityManager em;
17
18     @Override
19     protected EntityManager getEntityManager() {
20         return em;
21     }
22
23     public PersonFacade() {
24         super(Person.class);
25     }
26
27 }

```

▪ AbstractFacade.

```

1 package za.ac.tut.bl;
2
3 import java.util.List;
4 import javax.persistence.EntityManager;
5
6 /**
7 *
8 * @author MemaniV
9 */
10 public abstract class AbstractFacade<T> {
11
12     private Class<T> entityClass;
13
14     public AbstractFacade(Class<T> entityClass) {
15         this.entityClass = entityClass;
16     }
17
18     protected abstract EntityManager getEntityManager();
19
20     public void create(T entity) {
21         getEntityManager().persist(entity);
22     }
23
24     public void edit(T entity) {
25         getEntityManager().merge(entity);
26     }
27
28     public void remove(T entity) {
29         getEntityManager().remove(getEntityManager().merge(entity));
30     }
31

```

```

32     public T find(Object id) {
33         return getEntityManager().find(entityClass, id);
34     }
35
36     public List<T> findAll() {
37         javax.persistence.criteria.CriteriaQuery cq =
38             getEntityManager().getCriteriaBuilder().createQuery();
39         cq.select(cq.from(entityClass));
40         return getEntityManager().createQuery(cq).getResultList();
41     }
42
43     public List<T> findRange(int[] range) {
44         javax.persistence.criteria.CriteriaQuery cq =
45             getEntityManager().getCriteriaBuilder().createQuery();
46         cq.select(cq.from(entityClass));
47         javax.persistence.Query q = getEntityManager().createQuery(cq);
48         q.setMaxResults(range[1] - range[0] + 1);
49         q.setFirstResult(range[0]);
50         return q.getResultList();
51     }
52
53     public int count() {
54         javax.persistence.criteria.CriteriaQuery cq =
55             getEntityManager().getCriteriaBuilder().createQuery();
56         javax.persistence.criteria.Root<T> rt = cq.from(entityClass);
57         cq.select(getEntityManager().getCriteriaBuilder().count(rt));
58         javax.persistence.Query q = getEntityManager().createQuery(cq);
59         return ((Long) q.getSingleResult()).intValue();
60     }
61
62 }

```

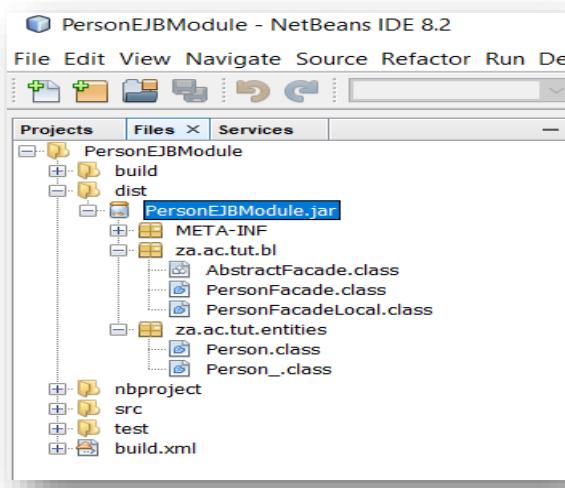
▪ PersonFacadeLocal.

The screenshot shows a Java code editor with the file `PersonFacadeLocal.java` open. The code defines a local interface for a facade. It includes imports for `List`, `Local`, and the `Person` entity. The interface has methods for creating, editing, and removing persons, as well as finding a person by ID, finding all persons, finding persons within a range, and counting the total number of persons.

```
1 package za.ac.tut.bl;
2
3 import java.util.List;
4 import javax.ejb.Local;
5 import za.ac.tut.entities.Person;
6
7 /**
8 * 
9 * @author MemaniV
10 */
11 @Local
12 public interface PersonFacadeLocal {
13
14     void create(Person person);
15
16     void edit(Person person);
17
18     void remove(Person person);
19
20     Person find(Object id);
21
22     List<Person> findAll();
23
24     List<Person> findRange(int[] range);
25
26     int count();
27
28 }
```

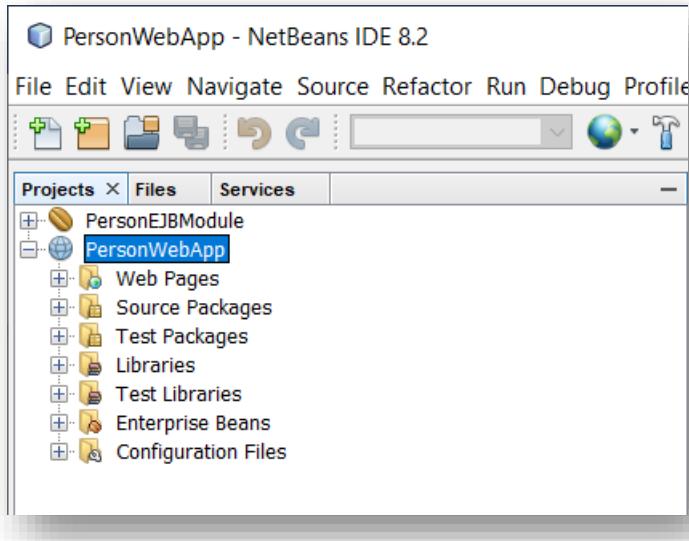
- ✓ Clean and build the EJB project.

```
Output < Java DB Database Process < GlassFish Server 4.1.1 < PersonEJBModule (clean,dist) < |  
INFO: Optional file was not found: META-INF/eclipselink-orm.xml continuing with generation.  
Note: Optional file was not found: META-INF/orm.xml continuing with generation.  
Note: Optional file was not found: META-INF/eclipselink-orm.xml continuing with generation.  
Note: C:\Users\memaniv\Documents\NetBeansProjects\PersonEJBModule\src\java\za\ac\tut\bl\AbstractFacade.java  
Note: Recompile with -Xlint:unchecked for details.  
1 warning  
compile:  
library-inclusion-in-archive:  
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\PersonEJBModule\dist  
Building jar: C:\Users\memaniv\Documents\NetBeansProjects\PersonEJBModule\dist\PersonEJBModule.jar  
dist:  
BUILD SUCCESSFUL (total time: 1 second)
```



Part D – Create a web client project.

Create a web client project called **PersonWebApp**.

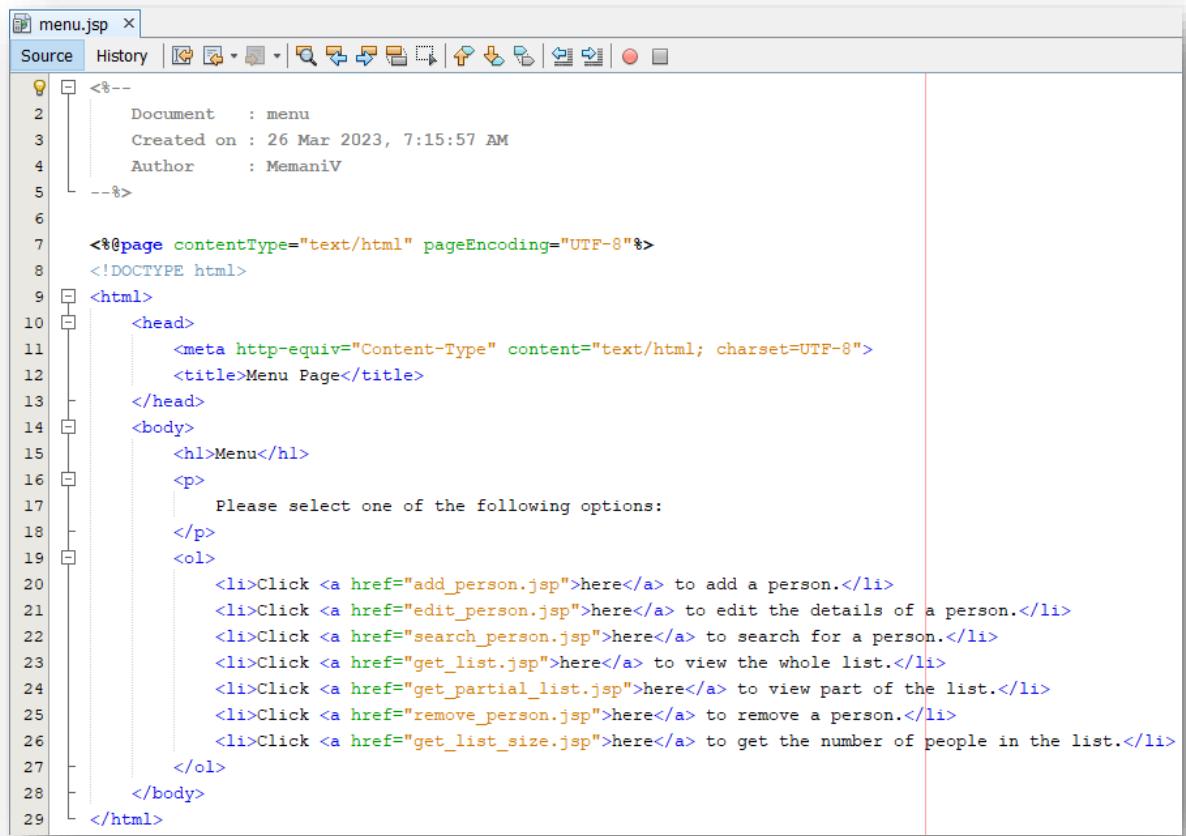


Edit the **index.html** page.

```
<!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 page</h1>
        <p>
            Welcome to our web app. Please click <a href="menu.jsp">here</a> to start.
        </p>
    </body>
</html>
```

The screenshot shows the NetBeans HTML editor with the file "index.html" open. The "Source" tab is selected. The code is displayed with line numbers from 1 to 19. The code itself is a standard HTML5 template with a title, meta tags for character set and viewport, and a body containing an h1 element and a p element with a link to "menu.jsp". The editor has a toolbar above it with various icons for file operations and code navigation.

Create the menu.jsp file.



The screenshot shows a Java IDE interface with a code editor window titled "menu.jsp". The window contains JSP code. The code includes a header section with document information, a page directive, and a scriptlet. It then defines an HTML structure with a head and body section. The body section contains a heading "Menu", a paragraph instructing the user to select an option, and an ordered list (ol) with seven items, each describing a different URL and its purpose. The code uses standard JSP syntax with tags like <%-->, <%@page>, <!DOCTYPE html>, <html>, <head>, <title>, <body>, <h1>, <p>, and .

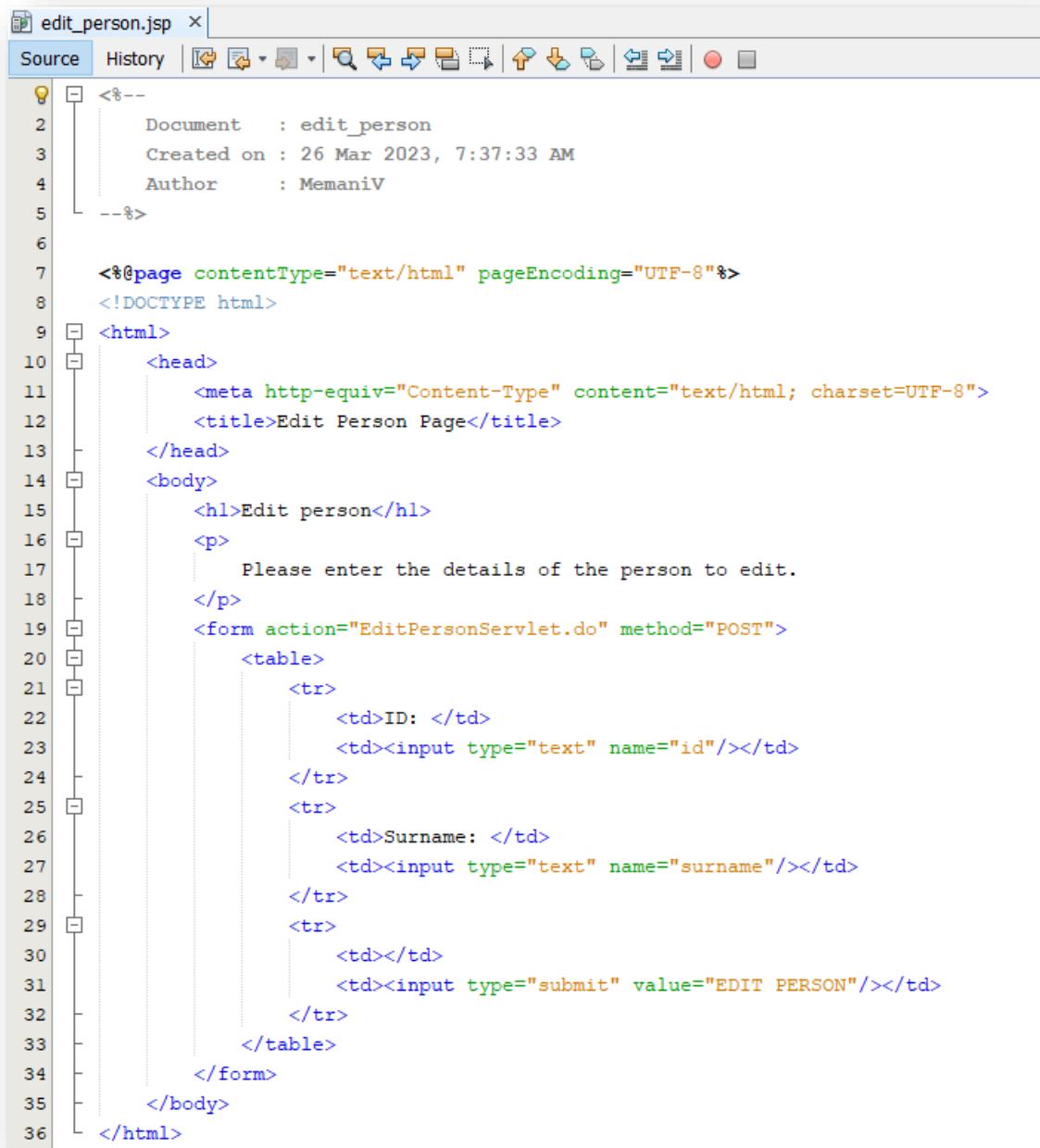
```
<%-->
1 Document : menu
2 Created on : 26 Mar 2023, 7:15:57 AM
3 Author : MemaniV
4 --%>
5
6
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>
8 <!DOCTYPE html>
9 <html>
10 <head>
11     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12     <title>Menu Page</title>
13 </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="add_person.jsp">here</a> to add a person.</li>
21         <li>Click <a href="edit_person.jsp">here</a> to edit the details of a person.</li>
22         <li>Click <a href="search_person.jsp">here</a> to search for a person.</li>
23         <li>Click <a href="get_list.jsp">here</a> to view the whole list.</li>
24         <li>Click <a href="get_partial_list.jsp">here</a> to view part of the list.</li>
25         <li>Click <a href="remove_person.jsp">here</a> to remove a person.</li>
26         <li>Click <a href="get_list_size.jsp">here</a> to get the number of people in the list.</li>
27     </ol>
28 </body>
29 </html>
```

Create add_person.jsp file.



```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Add Person Page</title>
    </head>
    <body>
        <h1>Add person</h1>
        <p>
            Please add person details below:
        </p>
        <form action="AddPersonServlet.do" method="POST">
            <table>
                <tr>
                    <td>Name: </td>
                    <td><input type="text" name="name"/></td>
                </tr>
                <tr>
                    <td>Surname: </td>
                    <td><input type="text" name="surname"/></td>
                </tr>
                <tr>
                    <td>Gender: </td>
                    <td>
                        <select name="gender">
                            <option value="F">Female</option>
                            <option value="M">Male</option>
                        </select>
                    </td>
                </tr>
                <tr>
                    <td></td>
                    <td><input type="submit" value="ADD PERSON"/></td>
                </tr>
            </table>
        </form>
    </body>
</html>
```

Create the **edit_person.jsp** file.



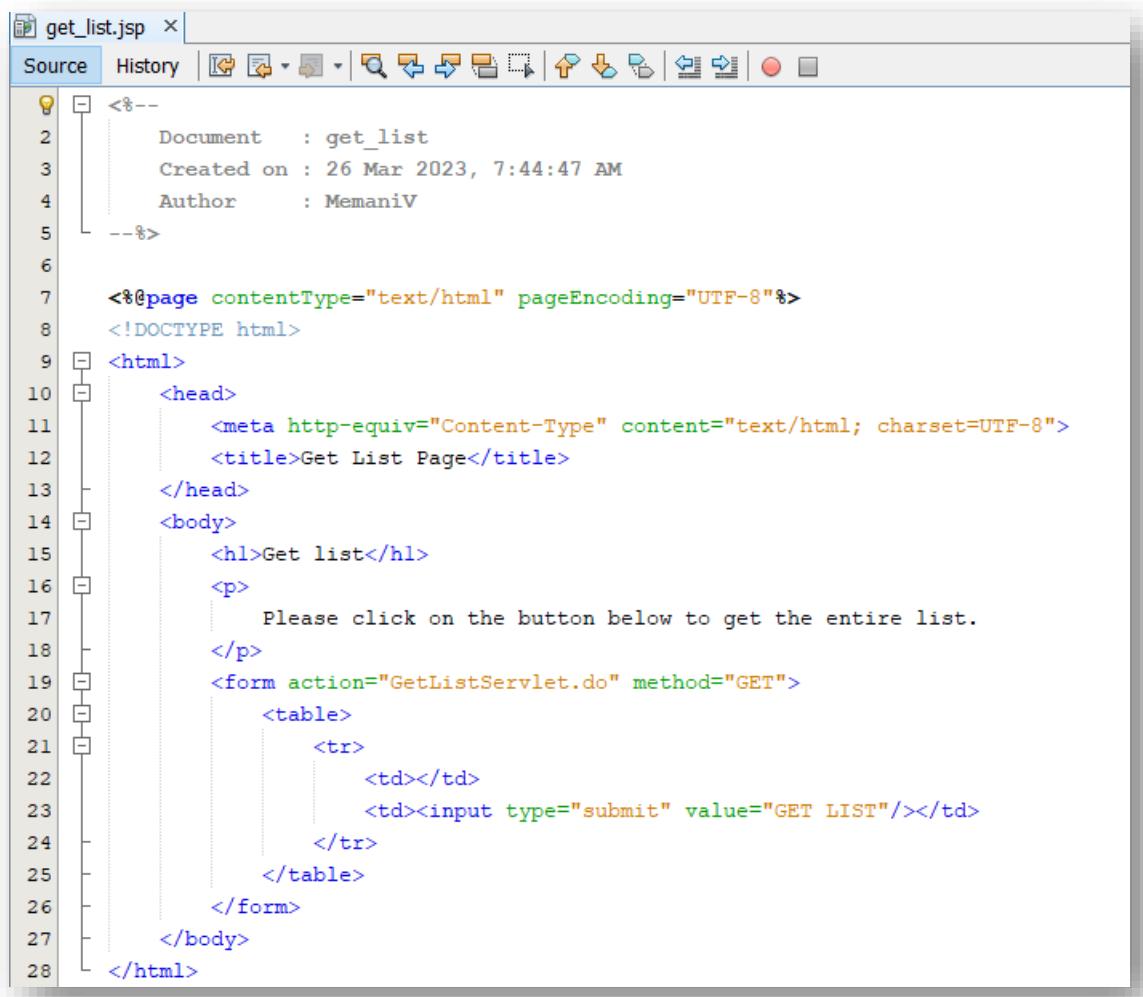
```
<%--  
1 Document : edit_person  
2 Created on : 26 Mar 2023, 7:37:33 AM  
3 Author : MemaniV  
4--%>  
5  
6  
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>  
8 <!DOCTYPE html>  
9 <html>  
10 <head>  
11     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
12     <title>Edit Person Page</title>  
13 </head>  
14 <body>  
15     <h1>Edit person</h1>  
16     <p>  
17         Please enter the details of the person to edit.  
18     </p>  
19     <form action="EditPersonServlet.do" method="POST">  
20         <table>  
21             <tr>  
22                 <td>ID: </td>  
23                 <td><input type="text" name="id"/></td>  
24             </tr>  
25             <tr>  
26                 <td>Surname: </td>  
27                 <td><input type="text" name="surname"/></td>  
28             </tr>  
29             <tr>  
30                 <td></td>  
31                 <td><input type="submit" value="EDIT PERSON"/></td>  
32             </tr>  
33         </table>  
34     </form>  
35 </body>  
36 </html>
```

Create the **search_person.jsp** file.

The screenshot shows a Java code editor with the file `search_person.jsp` open. The code is a JSP page for searching a person. It includes a header with document information, a meta tag for content type, a title, and a form for entering a person's ID and searching.

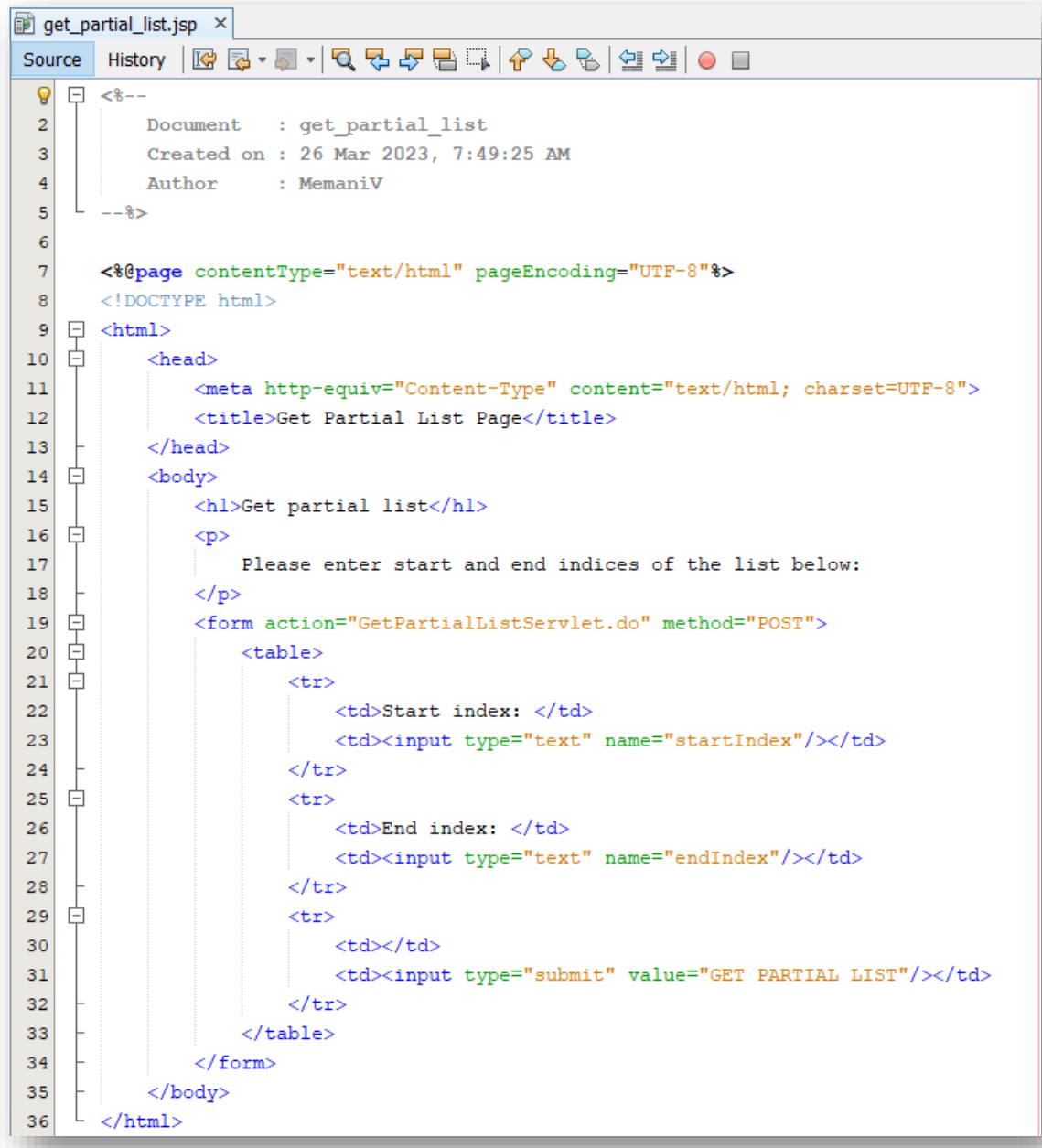
```
<%--  
    Document      : search_person  
    Created on   : 26 Mar 2023, 7:41:58 AM  
    Author        : MemaniV  
--%>  
  
<%@page contentType="text/html" pageEncoding="UTF-8"%>  
<!DOCTYPE html>  
<html>  
    <head>  
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
        <title>Search Person Page</title>  
    </head>  
    <body>  
        <h1>Search person</h1>  
        <p>  
            Please enter the ID of the person to search for.  
        </p>  
        <form action="SearchPersonServlet.do" method="POST">  
            <table>  
                <tr>  
                    <td>ID: </td>  
                    <td><input type="text" name="id"/></td>  
                </tr>  
                <tr>  
                    <td></td>  
                    <td><input type="submit" value="SEARCH"/></td>  
                </tr>  
            </table>  
        </form>  
    </body>  
</html>
```

Create the get_list.jsp file.



```
<%--  
1      Document : get_list  
2      Created on : 26 Mar 2023, 7:44:47 AM  
3      Author    : MemaniV  
4--%>  
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>Get List Page</title>  
13    </head>  
14    <body>  
15      <h1>Get list</h1>  
16      <p>  
17        Please click on the button below to get the entire list.  
18      </p>  
19      <form action="GetListServlet.do" method="GET">  
20        <table>  
21          <tr>  
22            <td></td>  
23            <td><input type="submit" value="GET LIST"/></td>  
24          </tr>  
25        </table>  
26      </form>  
27    </body>  
28  </html>
```

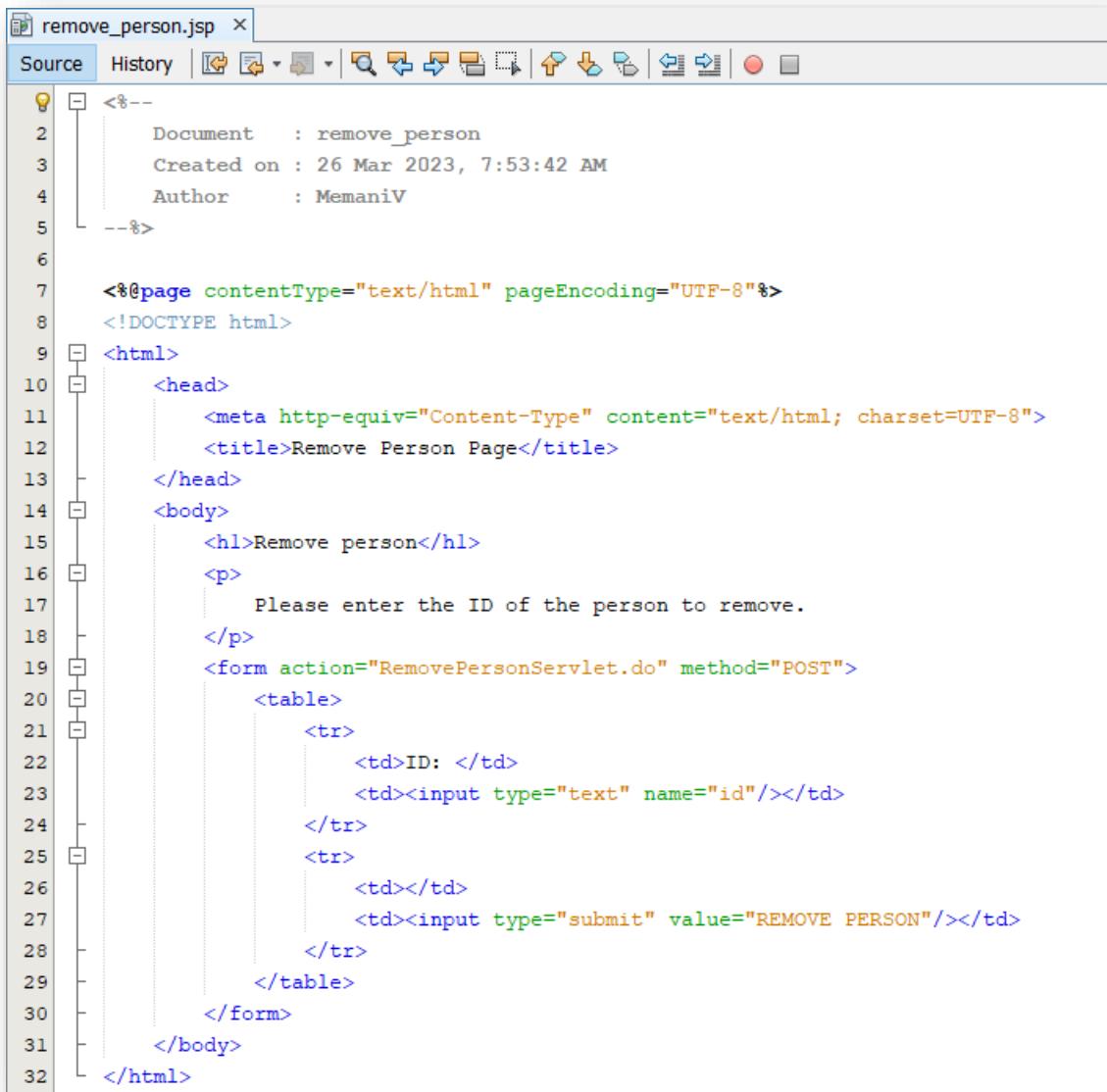
Create the get_partial_list.jsp file.



The screenshot shows a Java IDE interface with the file "get_partial_list.jsp" open. The code is a JSP page that displays a form for getting a partial list. It includes JSP comments for document information, a meta tag for content type, and a title. The form contains fields for start and end indices and a submit button.

```
<%--  
1 Document      : get_partial_list  
2 Created on   : 26 Mar 2023, 7:49:25 AM  
3 Author        : MemaniV  
4--%>  
  
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>Get Partial List Page</title>  
13 </head>  
14 <body>  
15     <h1>Get partial list</h1>  
16     <p>  
17         Please enter start and end indices of the list below:  
18     </p>  
19     <form action="GetPartialListServlet.do" method="POST">  
20         <table>  
21             <tr>  
22                 <td>Start index:</td>  
23                 <td><input type="text" name="startIndex"/></td>  
24             </tr>  
25             <tr>  
26                 <td>End index:</td>  
27                 <td><input type="text" name="endIndex"/></td>  
28             </tr>  
29             <tr>  
30                 <td></td>  
31                 <td><input type="submit" value="GET PARTIAL LIST"/></td>  
32             </tr>  
33         </table>  
34     </form>  
35     </body>  
36 </html>
```

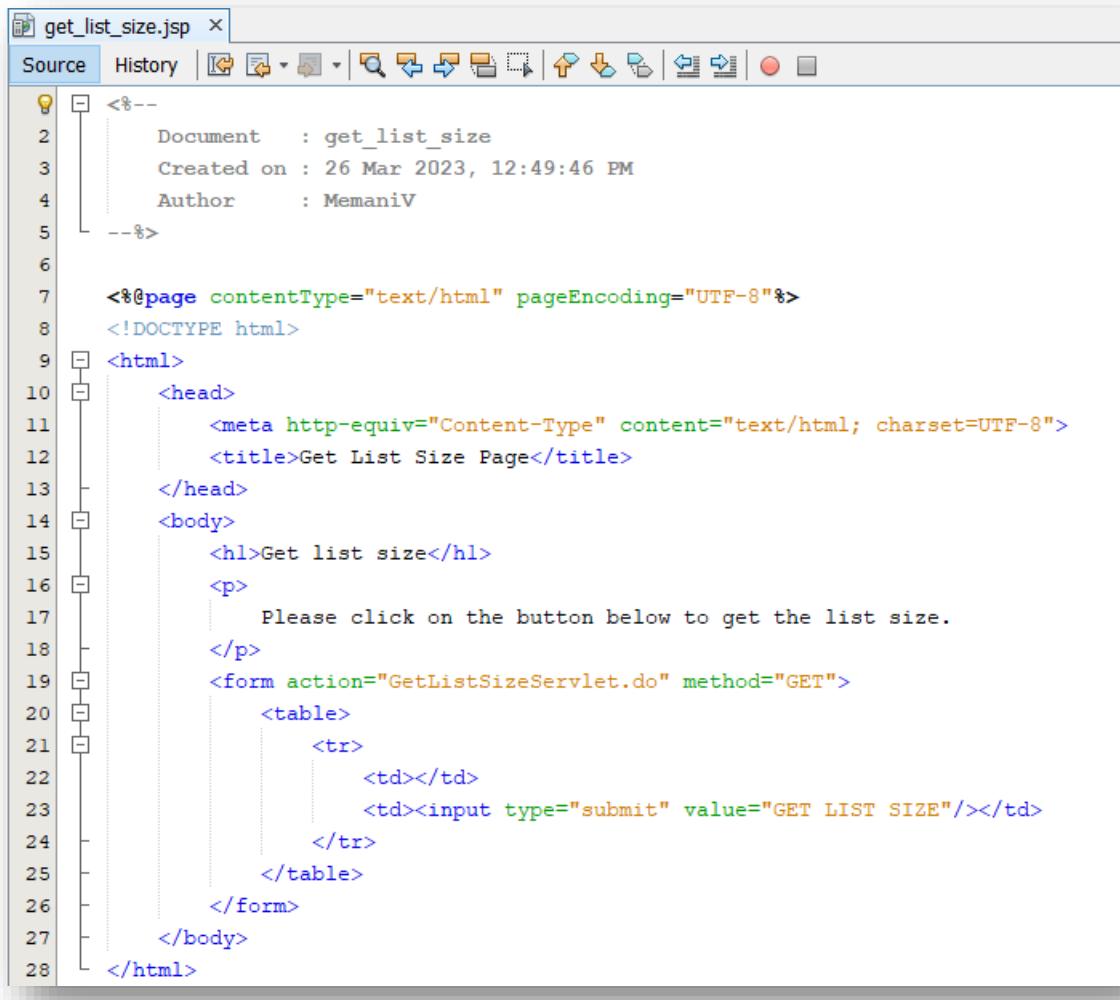
Create the remove_person.jsp file.



The screenshot shows a Java IDE interface with the file 'remove_person.jsp' open. The code is a JSP page for removing a person. It includes a header with document information, a meta tag, and a title. The body contains a heading, a paragraph asking for the ID, and a form with a table having two rows. The first row has a 'td' for the label 'ID:' and another 'td' for an input field named 'id'. The second row has a 'td' for an empty value and another 'td' for a submit button with the value 'REMOVE PERSON'.

```
<%--  
1 Document      : remove_person  
2 Created on   : 26 Mar 2023, 7:53:42 AM  
3 Author        : MemaniV  
4 --%>  
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>Remove Person Page</title>  
13 </head>  
14 <body>  
15     <h1>Remove person</h1>  
16     <p>  
17         Please enter the ID of the person to remove.  
18     </p>  
19     <form action="RemovePersonServlet.do" method="POST">  
20         <table>  
21             <tr>  
22                 <td>ID:</td>  
23                 <td><input type="text" name="id"/></td>  
24             </tr>  
25             <tr>  
26                 <td></td>  
27                 <td><input type="submit" value="REMOVE PERSON"/></td>  
28             </tr>  
29         </table>  
30     </form>  
31 </body>  
32 </html>
```

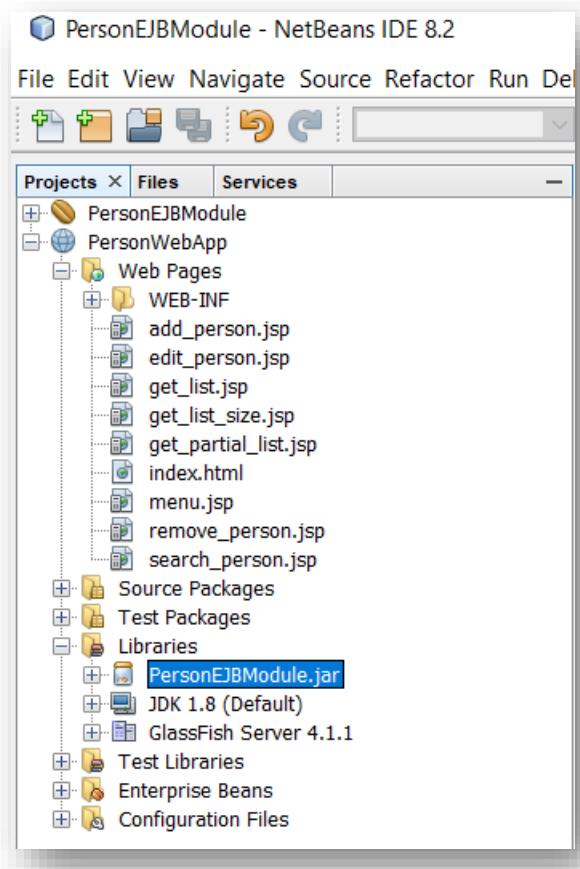
Create the get_list_size.jsp file.



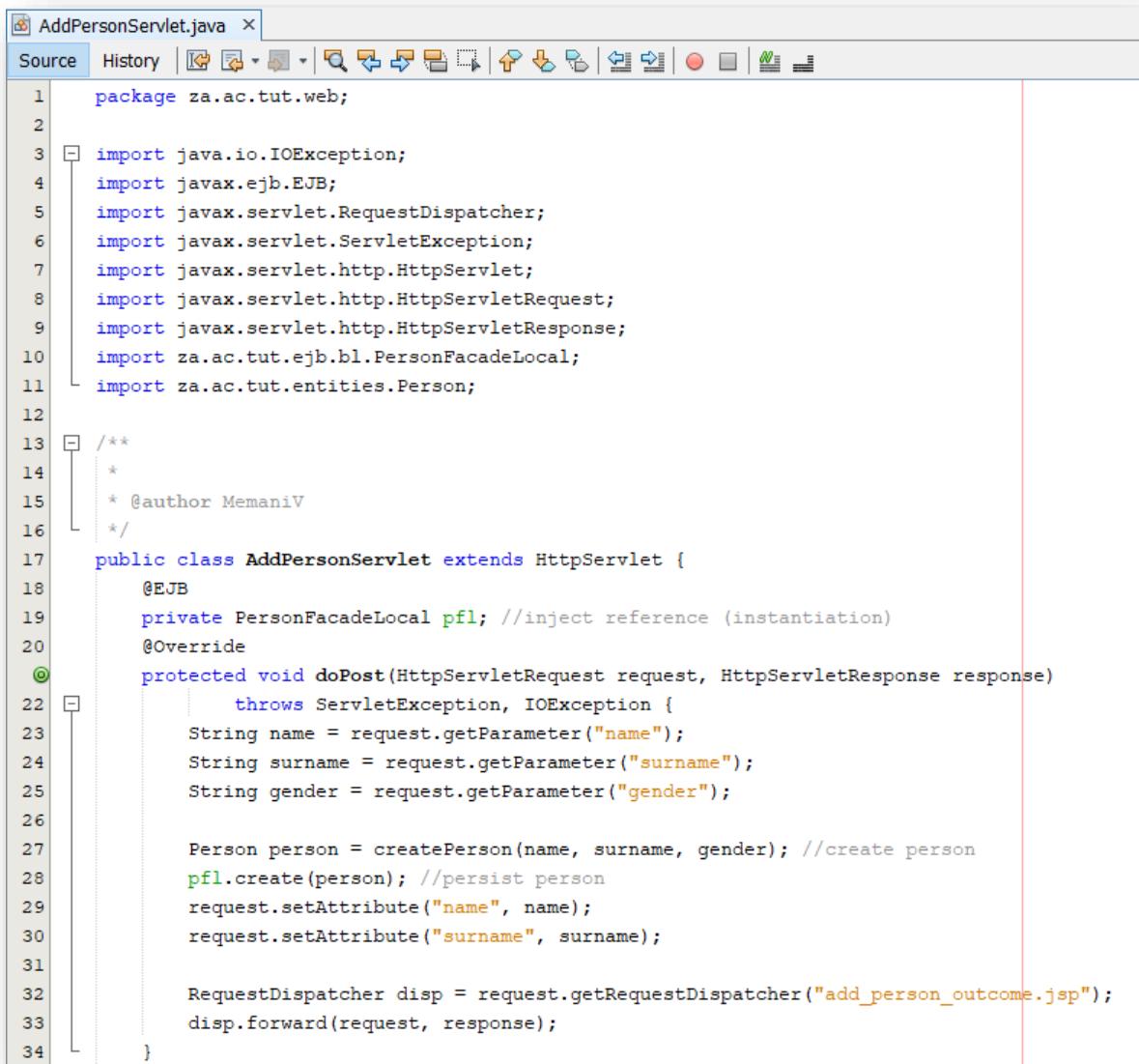
The screenshot shows a Java IDE interface with a code editor window titled "get_list_size.jsp". The window has a toolbar with various icons at the top. The code itself is a JSP page with the following content:

```
<%--  
1 Document      : get_list_size  
2 Created on   : 26 Mar 2023, 12:49:46 PM  
3 Author        : MemaniV  
4 --%>  
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>Get List Size Page</title>  
13 </head>  
14 <body>  
15     <h1>Get list size</h1>  
16     <p>  
17         Please click on the button below to get the list size.  
18     </p>  
19     <form action="GetListSizeServlet.do" method="GET">  
20         <table>  
21             <tr>  
22                 <td></td>  
23                 <td><input type="submit" value="GET LIST SIZE"/></td>  
24             </tr>  
25         </table>  
26     </form>  
27 </body>  
28 </html>
```

Add the **PersonEJBModule.jar** file to the library of **PersonWebApp**.

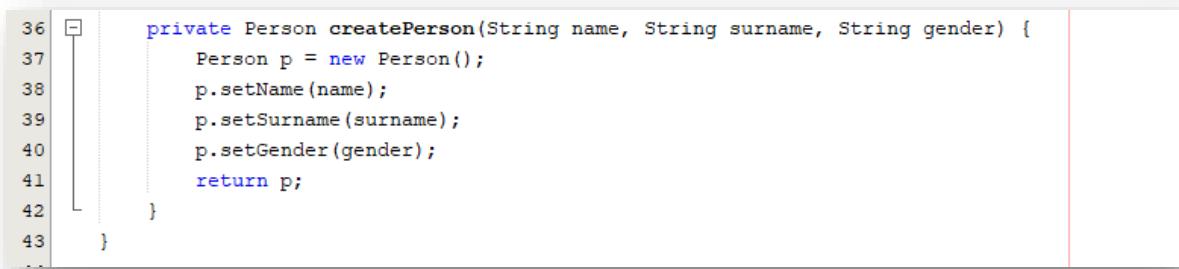


Create the AddPersonServlet.java file.



The screenshot shows a Java code editor window titled "AddPersonServlet.java". The code implements a servlet for adding a person. It imports various Java and EJB packages. The class extends HttpServlet and overrides doPost. It injects a PersonFacadeLocal bean named pfl. The doPost method retrieves name, surname, and gender from the request parameters, creates a new Person object, persist it, and then forward the response to an outcome page.

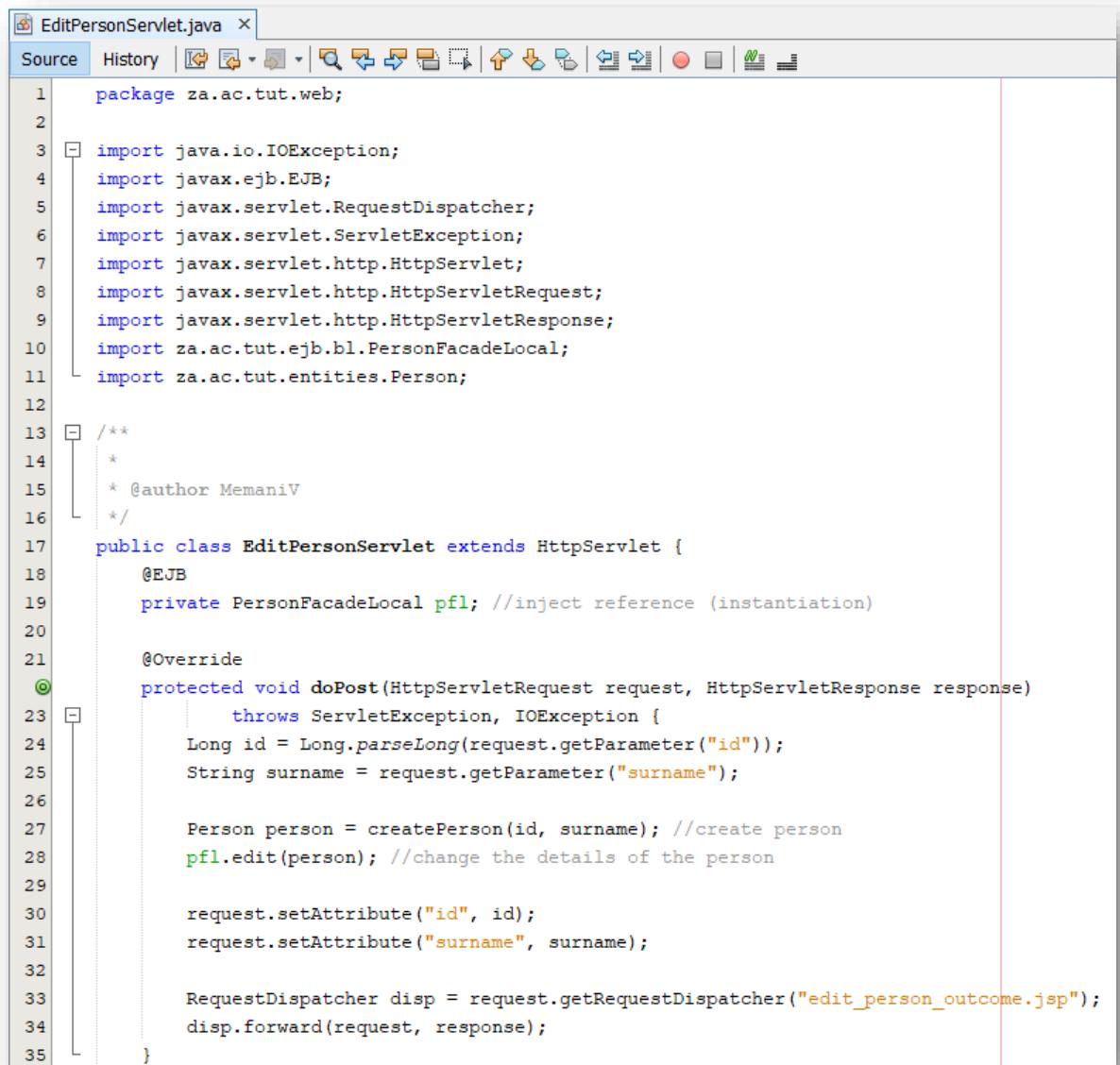
```
1 package za.ac.tut.web;
2
3 import java.io.IOException;
4 import javax.ejb.EJB;
5 import javax.servlet.RequestDispatcher;
6 import javax.servlet.ServletException;
7 import javax.servlet.http.HttpServlet;
8 import javax.servlet.http.HttpServletRequest;
9 import javax.servlet.http.HttpServletResponse;
10 import za.ac.tut.ejb.bl.PersonFacadeLocal;
11 import za.ac.tut.entities.Person;
12
13 /**
14  * 
15  * @author MemaniV
16  */
17 public class AddPersonServlet extends HttpServlet {
18     @EJB
19     private PersonFacadeLocal pfl; //inject reference (instantiation)
20     @Override
21     protected void doPost(HttpServletRequest request, HttpServletResponse response)
22             throws ServletException, IOException {
23         String name = request.getParameter("name");
24         String surname = request.getParameter("surname");
25         String gender = request.getParameter("gender");
26
27         Person person = createPerson(name, surname, gender); //create person
28         pfl.create(person); //persist person
29         request.setAttribute("name", name);
30         request.setAttribute("surname", surname);
31
32         RequestDispatcher disp = request.getRequestDispatcher("add_person_outcome.jsp");
33         disp.forward(request, response);
34     }
35 }
```



The screenshot shows the implementation of the createPerson method. It creates a new Person object, sets its name, surname, and gender, and then returns it.

```
36     private Person createPerson(String name, String surname, String gender) {
37         Person p = new Person();
38         p.setName(name);
39         p.setSurname(surname);
40         p.setGender(gender);
41         return p;
42     }
43 }
```

Create the **EditPersonServlet.java** file.

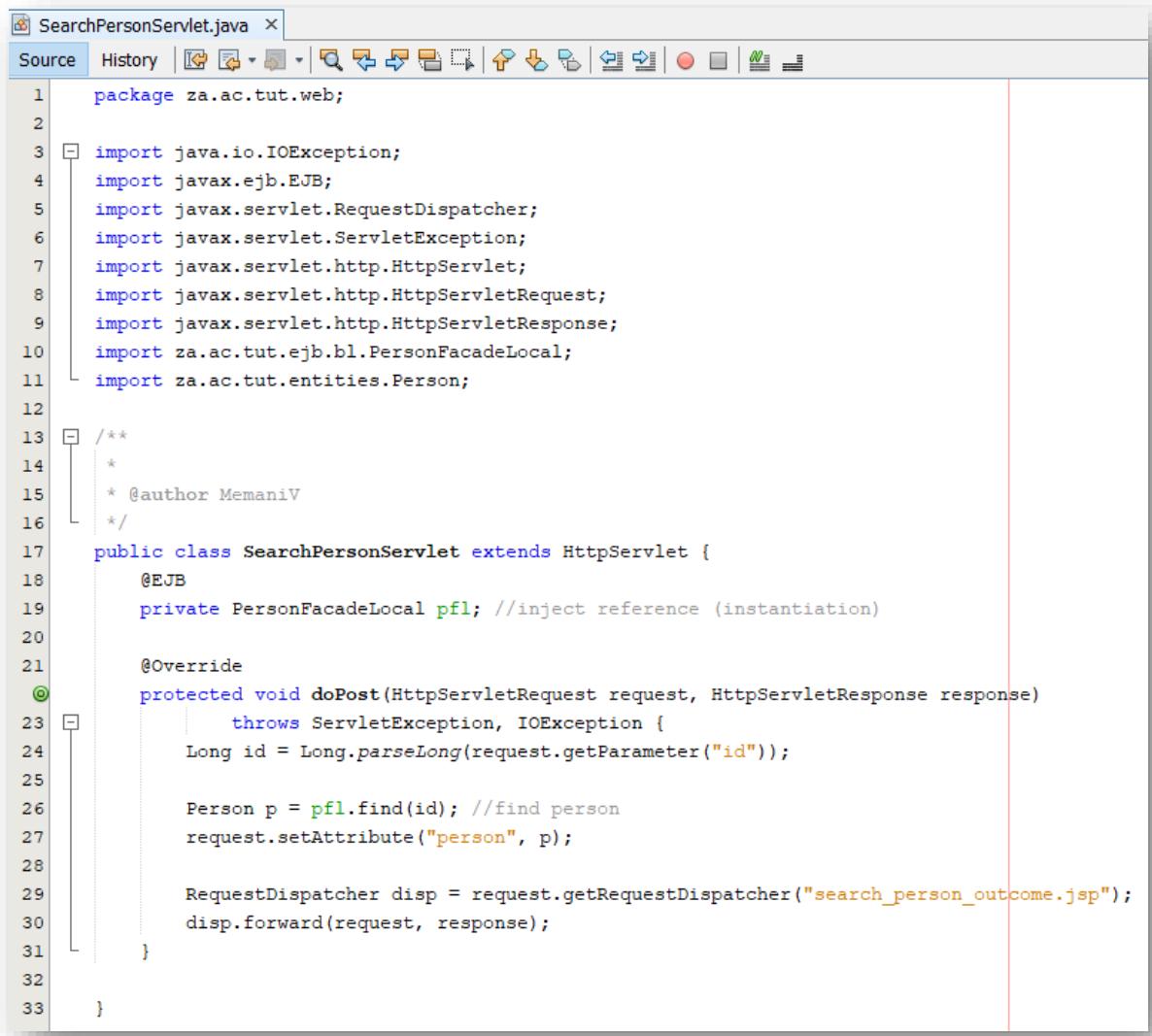


The screenshot shows a Java code editor with the title bar "EditPersonServlet.java". The menu bar includes "Source", "History", and various tool icons. The code is as follows:

```
1 package za.ac.tut.web;
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.PersonFacadeLocal;
11 import za.ac.tut.entities.Person;
12
13 /**
14 * @author MemaniV
15 */
16
17 public class EditPersonServlet extends HttpServlet {
18     @EJB
19     private PersonFacadeLocal pfl; //inject reference (instantiation)
20
21     @Override
22     protected void doPost(HttpServletRequest request, HttpServletResponse response)
23             throws ServletException, IOException {
24         Long id = Long.parseLong(request.getParameter("id"));
25         String surname = request.getParameter("surname");
26
27         Person person = createPerson(id, surname); //create person
28         pfl.edit(person); //change the details of the person
29
30         request.setAttribute("id", id);
31         request.setAttribute("surname", surname);
32
33         RequestDispatcher disp = request.getRequestDispatcher("edit_person_outcome.jsp");
34         disp.forward(request, response);
35     }
36
37     private Person createPerson(Long id, String surname) {
38         Person p = pfl.find(id); //get current person
39         p.setSurname(surname); //change the surname
40         return p;
41     }
42
43 }
```

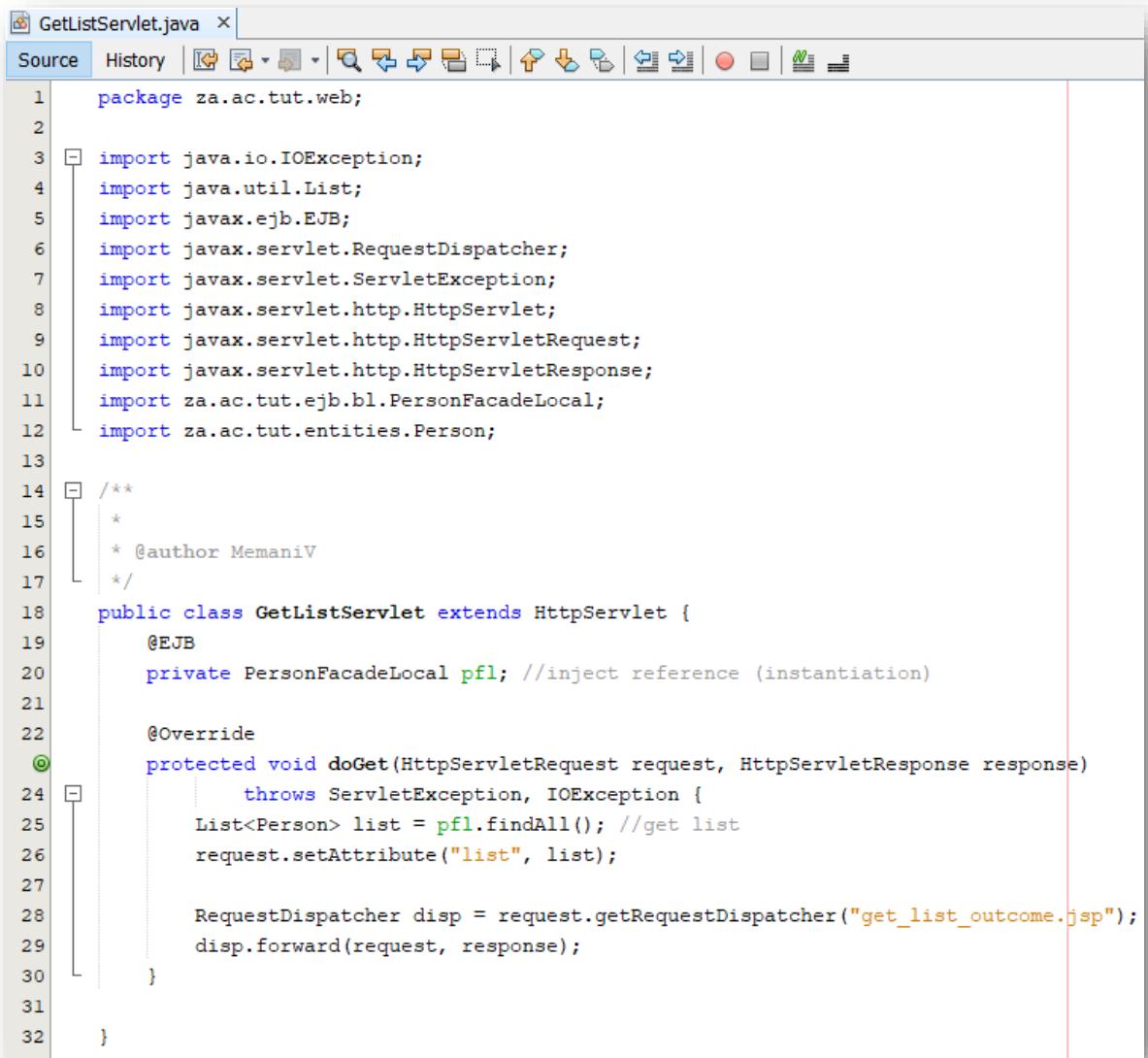
```
37     private Person createPerson(Long id, String surname) {
38         Person p = pfl.find(id); //get current person
39         p.setSurname(surname); //change the surname
40         return p;
41     }
42
43 }
```

Create the **SearchPersonServlet.java** file.



```
1 package za.ac.tut.web;
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.PersonFacadeLocal;
11 import za.ac.tut.entities.Person;
12
13 /**
14 *
15 * @author MemaniV
16 */
17 public class SearchPersonServlet extends HttpServlet {
18     @EJB
19     private PersonFacadeLocal pfl; //inject reference (instantiation)
20
21     @Override
22     protected void doPost(HttpServletRequest request, HttpServletResponse response)
23             throws ServletException, IOException {
24         Long id = Long.parseLong(request.getParameter("id"));
25
26         Person p = pfl.find(id); //find person
27         request.setAttribute("person", p);
28
29         RequestDispatcher disp = request.getRequestDispatcher("search_person_outcome.jsp");
30         disp.forward(request, response);
31     }
32
33 }
```

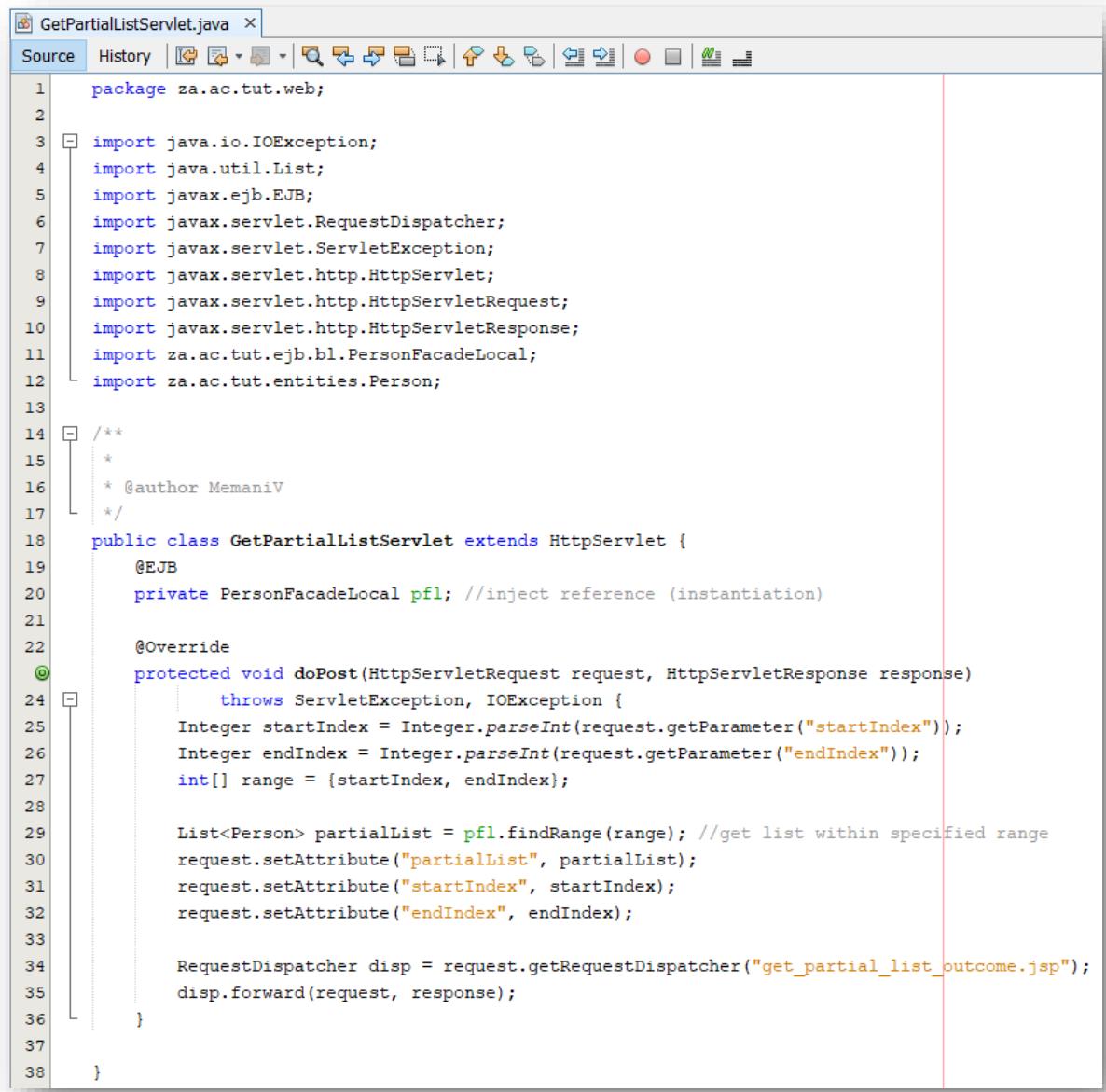
Create the **GetListServlet.java** file.



The screenshot shows a Java code editor window titled "GetListServlet.java". The code is a servlet named "GetListServlet" that extends "HttpServlet". It imports various Java packages and annotations, including "java.io.IOException", "java.util.List", "javax.ejb.EJB", "javax.servlet.RequestDispatcher", "javax.servlet.ServletException", "javax.servlet.http.HttpServlet", "javax.servlet.http.HttpServletRequest", "javax.servlet.http.HttpServletResponse", "za.ac.tut.ejb.bl.PersonFacadeLocal", and "za.ac.tut.entities.Person". The class contains a single method, "doGet", which retrieves all persons from the database using the "PersonFacadeLocal" bean and forwards the request to a JSP page named "get_list_outcome.jsp".

```
1 package za.ac.tut.web;
2
3 import java.io.IOException;
4 import java.util.List;
5 import javax.ejb.EJB;
6 import javax.servlet.RequestDispatcher;
7 import javax.servlet.ServletException;
8 import javax.servlet.http.HttpServlet;
9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11 import za.ac.tut.ejb.bl.PersonFacadeLocal;
12 import za.ac.tut.entities.Person;
13
14 /**
15 * 
16 * @author MemaniV
17 */
18 public class GetListServlet extends HttpServlet {
19     @EJB
20     private PersonFacadeLocal pfl; //inject reference (instantiation)
21
22     @Override
23     protected void doGet(HttpServletRequest request, HttpServletResponse response)
24         throws ServletException, IOException {
25         List<Person> list = pfl.findAll(); //get list
26         request.setAttribute("list", list);
27
28         RequestDispatcher disp = request.getRequestDispatcher("get_list_outcome.jsp");
29         disp.forward(request, response);
30     }
31
32 }
```

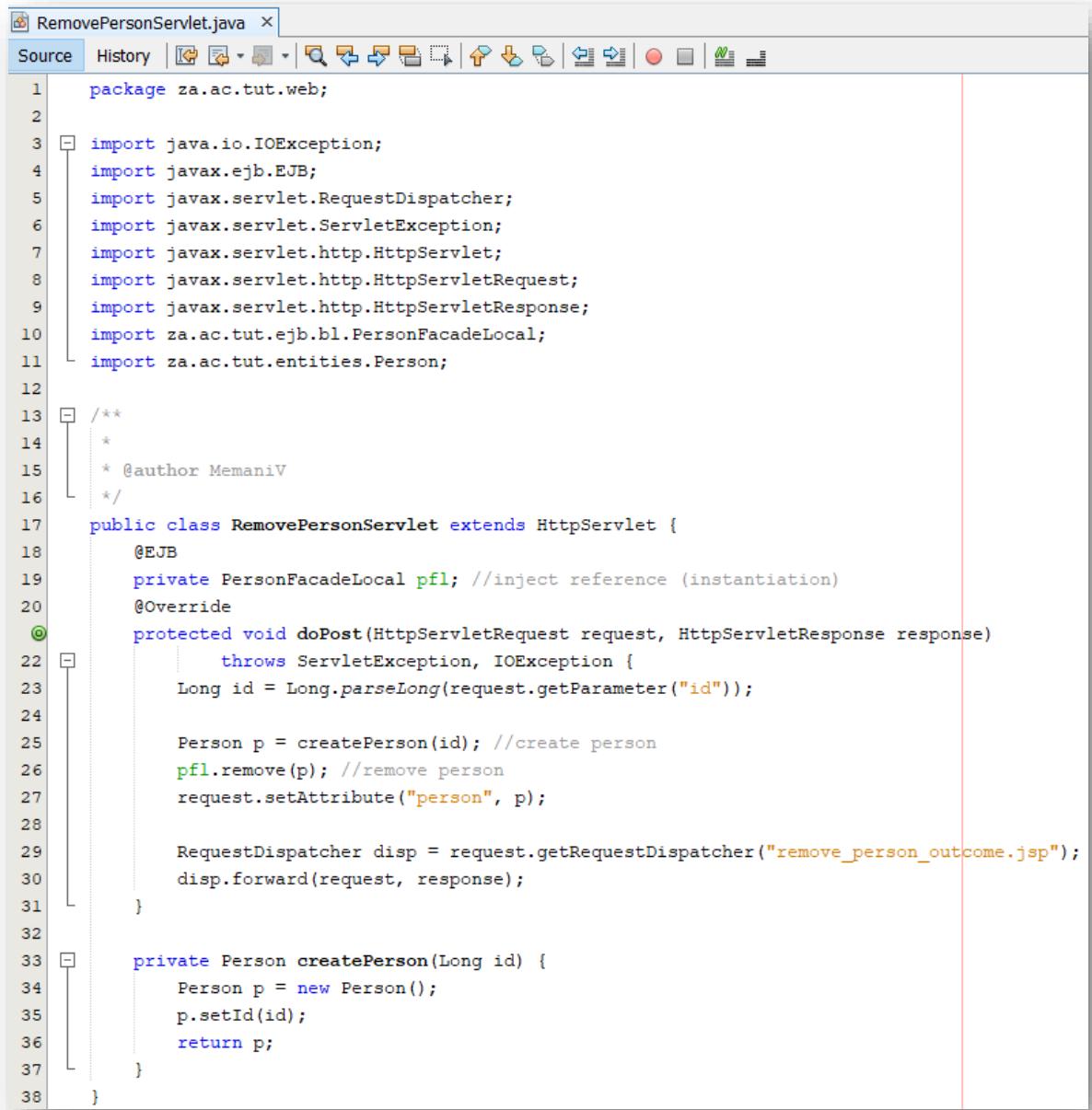
Create the **GetPartialListServlet.java** file.



The screenshot shows a Java code editor window titled "GetPartialListServlet.java". The code implements a HttpServlet that retrieves a partial list of Person objects from an EJB. The code includes imports for various Java and javax packages, a Javadoc comment, and a doPost method that uses the injected PersonFacadeLocal reference to find a range of persons and forwards the request to a JSP page.

```
1 package za.ac.tut.web;
2
3 import java.io.IOException;
4 import java.util.List;
5 import javax.ejb.EJB;
6 import javax.servlet.RequestDispatcher;
7 import javax.servlet.ServletException;
8 import javax.servlet.http.HttpServlet;
9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11 import za.ac.tut.ejb.bl.PersonFacadeLocal;
12 import za.ac.tut.entities.Person;
13
14 /**
15 *
16 * @author MemaniV
17 */
18 public class GetPartialListServlet extends HttpServlet {
19     @EJB
20     private PersonFacadeLocal pfl; //inject reference (instantiation)
21
22     @Override
23     protected void doPost(HttpServletRequest request, HttpServletResponse response)
24             throws ServletException, IOException {
25         Integer startIndex = Integer.parseInt(request.getParameter("startIndex"));
26         Integer endIndex = Integer.parseInt(request.getParameter("endIndex"));
27         int[] range = {startIndex, endIndex};
28
29         List<Person> partialList = pfl.findRange(range); //get list within specified range
30         request.setAttribute("partialList", partialList);
31         request.setAttribute("startIndex", startIndex);
32         request.setAttribute("endIndex", endIndex);
33
34         RequestDispatcher disp = request.getRequestDispatcher("get_partial_list_outcome.jsp");
35         disp.forward(request, response);
36     }
37
38 }
```

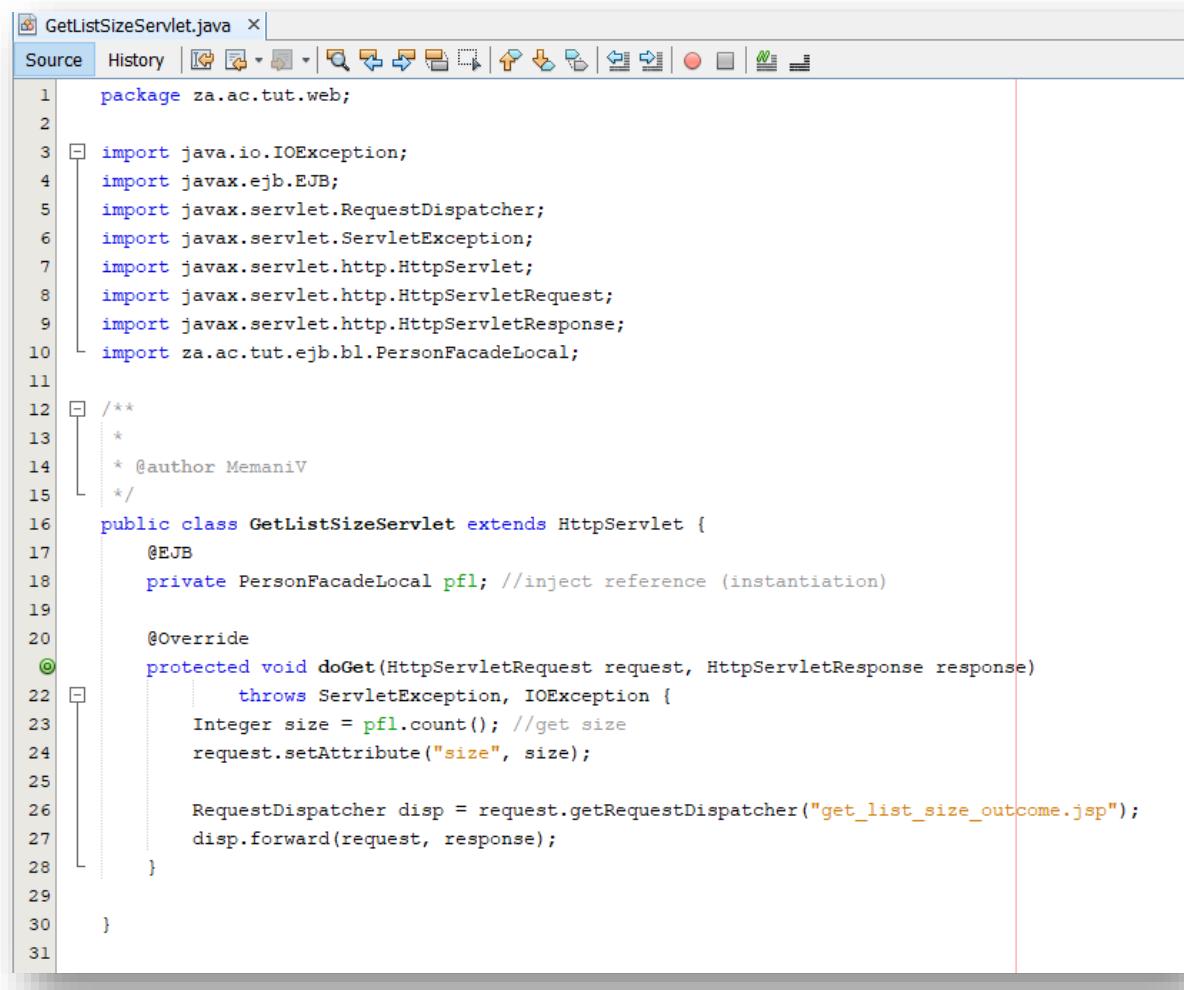
Create the RemovePersonServlet.java file.



The screenshot shows a Java code editor window with the title "RemovePersonServlet.java". The code is a servlet for removing a person from a database. It imports various Java packages and annotations, including `java.io.IOException`, `javax.ejb.EJB`, `javax.servlet.RequestDispatcher`, `javax.servlet.ServletException`, `javax.servlet.http.HttpServlet`, `javax.servlet.http.HttpServletRequest`, `javax.servlet.http.HttpServletResponse`, `za.ac.tut.ejb.bl.PersonFacadeLocal`, and `za.ac.tut.entities.Person`. The class `RemovePersonServlet` extends `HttpServlet` and overrides the `doPost` method. It injects a reference to `PersonFacadeLocal` named `pfl`. The `doPost` method retrieves a person's ID from the request, creates a new person object with that ID, removes it from the facade, and then forwards the request to a JSP page named "remove_person_outcome.jsp". A private helper method `createPerson` is defined to create a new person object with the given ID.

```
1 package za.ac.tut.web;
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.PersonFacadeLocal;
11 import za.ac.tut.entities.Person;
12
13 /**
14 *
15 * @author MemaniV
16 */
17 public class RemovePersonServlet extends HttpServlet {
18     @EJB
19     private PersonFacadeLocal pfl; //inject reference (instantiation)
20     @Override
21     protected void doPost(HttpServletRequest request, HttpServletResponse response)
22             throws ServletException, IOException {
23         Long id = Long.parseLong(request.getParameter("id"));
24
25         Person p = createPerson(id); //create person
26         pfl.remove(p); //remove person
27         request.setAttribute("person", p);
28
29         RequestDispatcher disp = request.getRequestDispatcher("remove_person_outcome.jsp");
30         disp.forward(request, response);
31     }
32
33     private Person createPerson(Long id) {
34         Person p = new Person();
35         p.setId(id);
36         return p;
37     }
38 }
```

Create the **GetListSizeServlet.java** file.



The screenshot shows a Java code editor window with the title "GetListSizeServlet.java". The code is a servlet named "GetListSizeServlet" that extends "HttpServlet". It imports various Java packages including java.io, javax.ejb, javax.servlet, javax.servlet.http, and javax.servlet.jsp. The class contains a constructor with an annotation "@EJB" pointing to "PersonFacadeLocal", a comment block, and an overridden "doGet" method. The "doGet" method retrieves the size of a list from "PersonFacadeLocal" and sets it as an attribute in the request. It then uses "RequestDispatcher" to forward the request to a JSP page named "get_list_size_outcome.jsp". The code is numbered from 1 to 31.

```
1 package za.ac.tut.web;
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.PersonFacadeLocal;
11
12 /**
13 * 
14 * @author MemaniV
15 */
16 public class GetListSizeServlet extends HttpServlet {
17     @EJB
18     private PersonFacadeLocal pfl; //inject reference (instantiation)
19
20     @Override
21     protected void doGet(HttpServletRequest request, HttpServletResponse response)
22             throws ServletException, IOException {
23         Integer size = pfl.count(); //get size
24         request.setAttribute("size", size);
25
26         RequestDispatcher disp = request.getRequestDispatcher("get_list_size_outcome.jsp");
27         disp.forward(request, response);
28     }
29 }
30
31 }
```

Create the add_person_outcome.jsp file.

The screenshot shows a Java IDE interface with the tab "add_person_outcome.jsp" selected. The code editor displays the following JSP page:

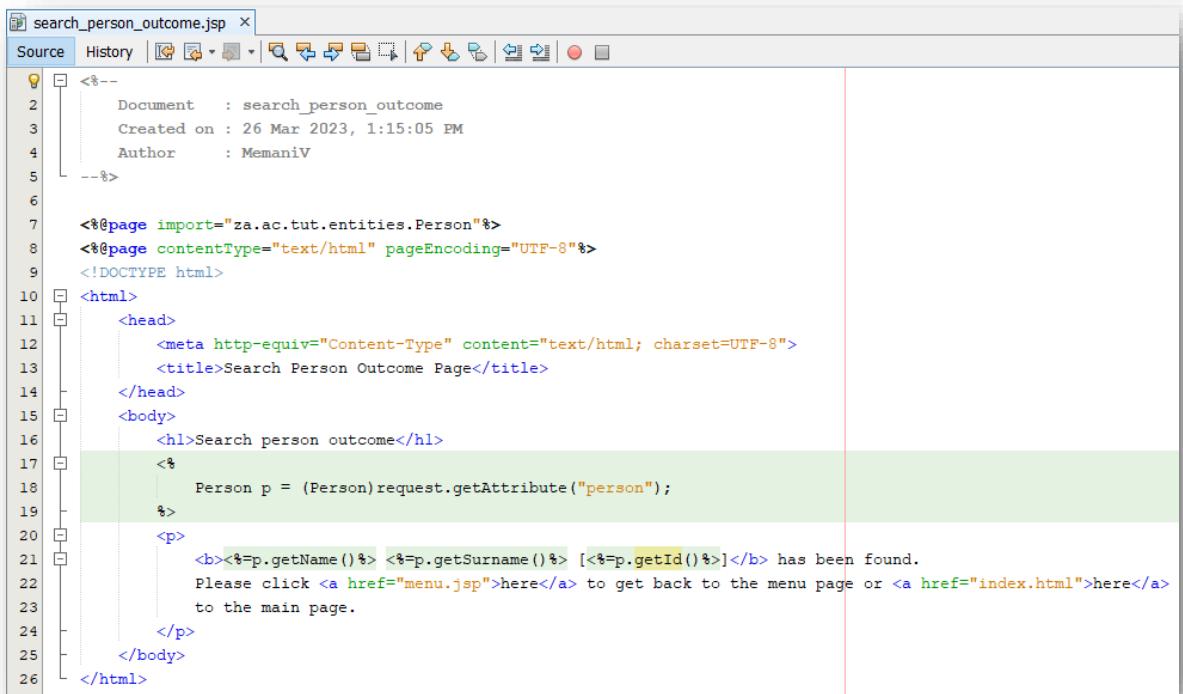
```
<%--  
Document : add_person_outcome  
Created on : 26 Mar 2023, 1:05:24 PM  
Author : MemaniV  
--%>  
  
<%@page contentType="text/html" pageEncoding="UTF-8"%>  
<!DOCTYPE html>  
<html>  
    <head>  
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
        <title>Add Person Outcome Page</title>  
    </head>  
    <body>  
        <h1>Add person outcome</h1>  
        <%  
            String name = (String)request.getAttribute("name");  
            String surname = (String)request.getAttribute("surname");  
        %>  
        <p>  
            <b><%=name%> <%=surname%></b> has been successfully persisted into the database.  
            Please click <a href="menu.jsp">here</a> to get back to the menu page or <a href="index.html">here</a>  
            to the main page.  
        </p>  
    </body>  
</html>
```

Create the edit_person_outcome.jsp file.

The screenshot shows a Java IDE interface with the tab "edit_person_outcome.jsp" selected. The code editor displays the following JSP page:

```
<%--  
Document : edit_person_outcome  
Created on : 26 Mar 2023, 1:11:14 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>Edit Person Outcome Page</title>  
    </head>  
    <body>  
        <h1>Edit person outcome</h1>  
        <%  
            Long id = (Long)request.getAttribute("id");  
            String surname = (String)request.getAttribute("surname");  
        %>  
        <p>  
            <b><%=surname%> [<%=id%>]</b> has been successfully edited in the database.  
            Please click <a href="menu.jsp">here</a> to get back to the menu page or <a href="index.html">here</a>  
            to the main page.  
        </p>  
    </body>  
</html>
```

Create the search_person_outcome.jsp file.



```
<%--  
1      Document   : search_person_outcome  
2      Created on : 26 Mar 2023, 1:15:05 PM  
3      Author     : MemaniV  
4--%>  
5  
6  
7      <%@page import="za.ac.tut.entities.Person"%>  
8      <%@page contentType="text/html" pageEncoding="UTF-8"%>  
9      <!DOCTYPE html>  
10     <html>  
11         <head>  
12             <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
13             <title>Search Person Outcome Page</title>  
14         </head>  
15         <body>  
16             <h1>Search person outcome</h1>  
17             <%  
18                 Person p = (Person)request.getAttribute("person");  
19             %>  
20             <p>  
21                 <b><%=p.getName()%> <%=p.getSurname()%> [<%=p.getId()%>]</b> has been found.  
22                 Please click <a href="menu.jsp">here</a> to get back to the menu page or <a href="index.html">here</a>  
23                 to the main page.  
24             </p>  
25         </body>  
26     </html>
```

Create the search_person_outcome.jsp file.

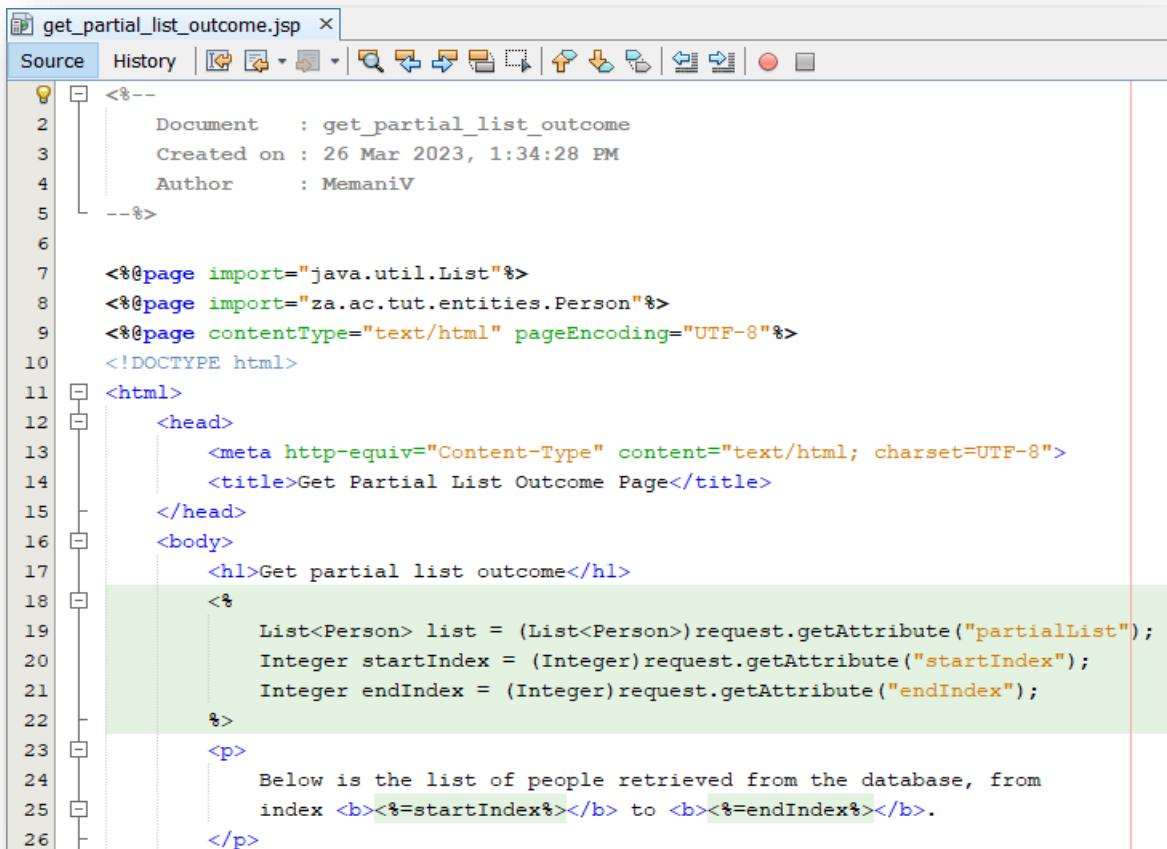
The screenshot shows the code editor interface of a Java IDE. The title bar says "get_list_outcome.jsp". The menu bar includes "Source", "History", and various tool icons. The code itself is a JSP page with the following content:

```
<%--  
1 Document : get_list_outcome  
2 Created on : 26 Mar 2023, 1:21:29 PM  
3 Author : MemaniV  
4--%>  
5  
6  
7 <%@page import="java.util.List"%>  
8 <%@page import="za.ac.tut.entities.Person"%>  
9 <%@page contentType="text/html" pageEncoding="UTF-8"%>  
10 <!DOCTYPE html>  
11 <html>  
12 <head>  
13     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
14     <title>Get List Outcome Page</title>  
15 </head>  
16 <body>  
17     <h1>Get list outcome</h1>  
18     <%  
19         List<Person> list = (List<Person>) request.getAttribute("list");  
20     %>  
21     <p>  
22         Below is the list of people retrieved from the database.  
23     </p>  
24     <table>  
25         <%  
26             for(int i = 0; i < list.size(); i++){  
27                 Person p = list.get(i);  
28                 Long id = p.getId();  
29                 String name = p.getName();  
30                 String surname = p.getSurname();  
31             %>  
32             <tr>  
33                 <td>ID:</td>  
34                 <td><%=id%></td>  
35             </tr>  
36             <tr>  
37                 <td>Name:</td>  
38                 <td><%=name%></td>  
39             </tr>
```

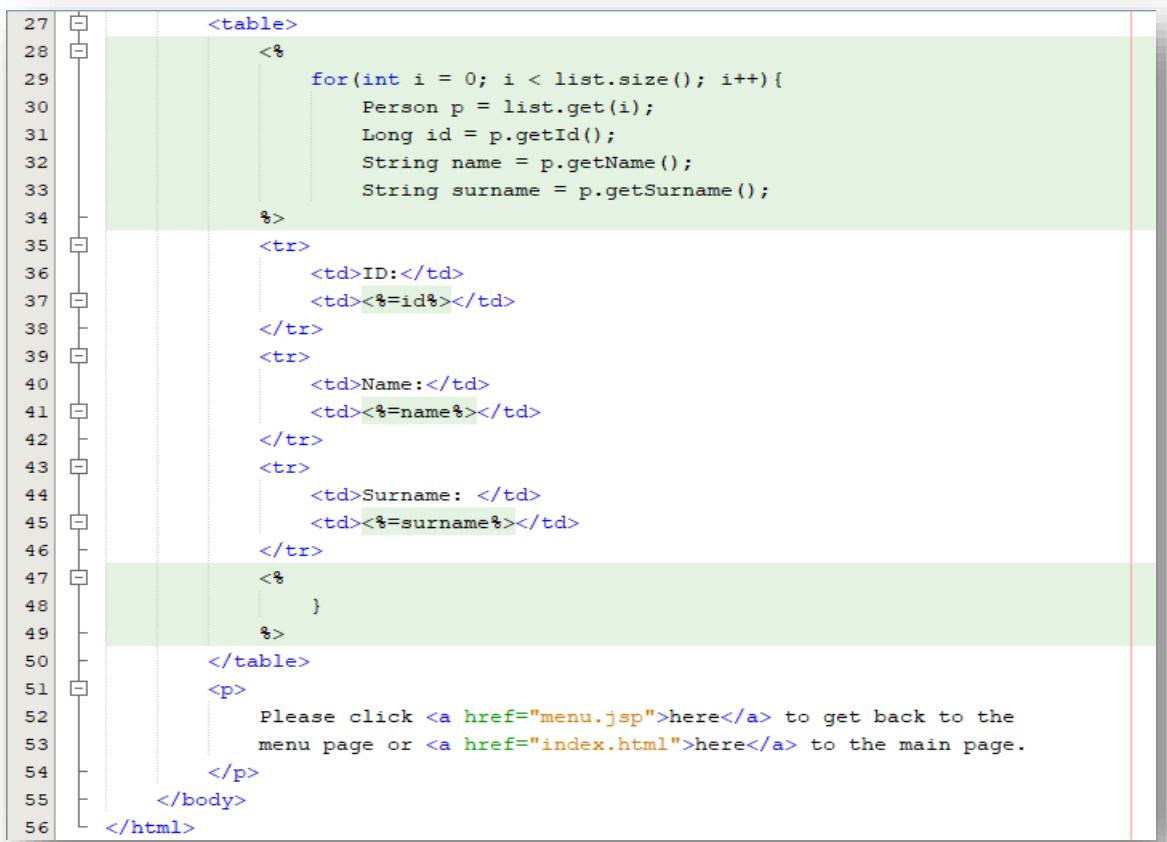
This is a continuation of the code editor screenshot, showing the bottom portion of the JSP page:

```
40         <td>Surname: </td>  
41         <td><%=surname%></td>  
42     </tr>  
43     <br />  
44     <%  
45     %>  
46     </table>  
47     <p>  
48         Please click <a href="menu.jsp">here</a> to get back to the menu page or <a href="index.html">here</a>  
49         to the main page.  
50     </p>  
51     </body>  
52 </html>
```

Create the get_partial_list_outcome.jsp file.



```
<%--  
    Document      : get_partial_list_outcome  
    Created on   : 26 Mar 2023, 1:34:28 PM  
    Author       : MemaniV  
--%>  
  
<%@page import="java.util.List"%>  
<%@page import="za.ac.tut.entities.Person"%>  
<%@page contentType="text/html" pageEncoding="UTF-8"%>  
<!DOCTYPE html>  
<html>  
    <head>  
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
        <title>Get Partial List Outcome Page</title>  
    </head>  
    <body>  
        <h1>Get partial list outcome</h1>  
        <%  
            List<Person> list = (List<Person>)request.getAttribute("partialList");  
            Integer startIndex = (Integer)request.getAttribute("startIndex");  
            Integer endIndex = (Integer)request.getAttribute("endIndex");  
        %>  
        <p>  
            Below is the list of people retrieved from the database, from  
            index <b><%=startIndex%></b> to <b><%=endIndex%></b>.  
        </p>
```



```
<table>  
    <%  
        for(int i = 0; i < list.size(); i++){  
            Person p = list.get(i);  
            Long id = p.getId();  
            String name = p.getName();  
            String surname = p.getSurname();  
        %>  
        <tr>  
            <td>ID:</td>  
            <td><%=id%></td>  
        </tr>  
        <tr>  
            <td>Name:</td>  
            <td><%=name%></td>  
        </tr>  
        <tr>  
            <td>Surname:</td>  
            <td><%=surname%></td>  
        </tr>  
    <%  
    }  
    %>  
    </table>  
    <p>  
        Please click <a href="menu.jsp">here</a> to get back to the  
        menu page or <a href="index.html">here</a> to the main page.  
    </p>  
    </body>  
</html>
```

Create the remove_person_outcome.jsp file.

The screenshot shows a Java IDE interface with the file 'remove_person_outcome.jsp' open. The code is a JSP page that removes a person from a database and displays a success message. The code includes imports for Person, meta tags for content type and charset, and a title 'Remove Person Outcome Page'. It features an h1 header 'Remove person outcome', a scriptlet block to get the person object, and a paragraph displaying a success message with links to menu.jsp and index.html.

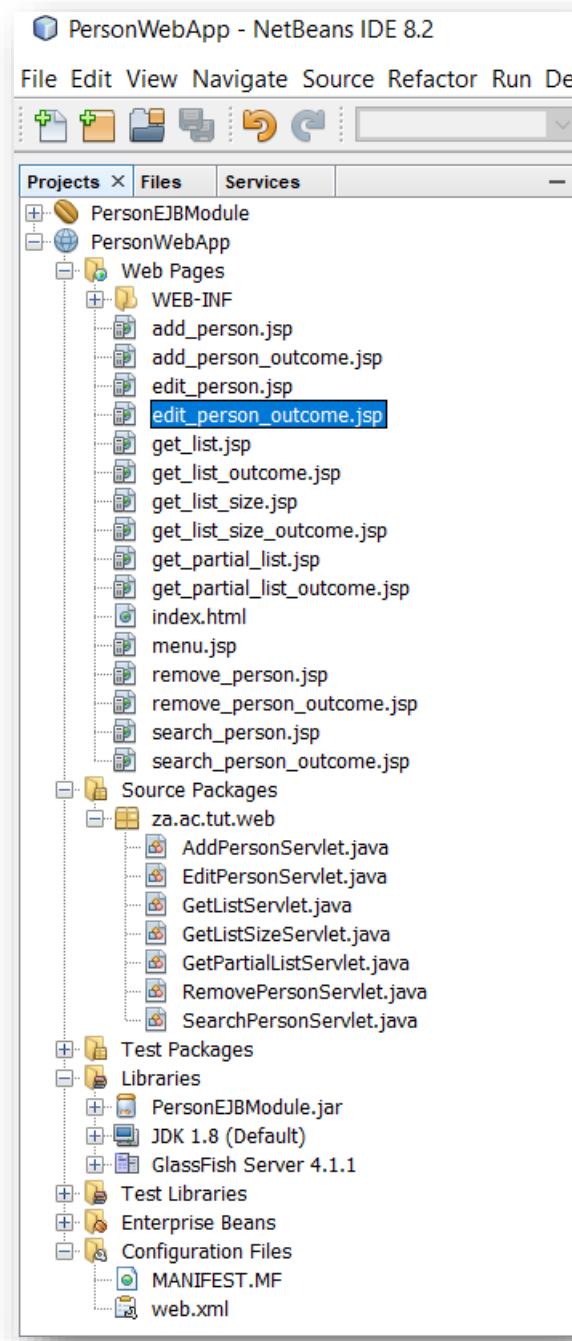
```
<%--  
    Document      : remove_person_outcome  
    Created on   : 26 Mar 2023, 1:40:56 PM  
    Author       : MemaniV  
--%>  
  
<%@page import="za.ac.tut.entities.Person"%>  
<%@page contentType="text/html" pageEncoding="UTF-8"%>  
<!DOCTYPE html>  
<html>  
    <head>  
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
        <title>Remove Person Outcome Page</title>  
    </head>  
    <body>  
        <h1>Remove person outcome</h1>  
        <%  
            Person p = (Person)request.getAttribute("person");  
        %>  
        <p>  
            The person with id number <b><%=p.getId()%></b> has been successfully removed from the database.  
            Please click <a href="menu.jsp">here</a> to get back to the menu page or <a href="index.html">here</a>  
            to the main page.  
        </p>  
    </body>  
</html>
```

Create the get_list_size_outcome.jsp file.

The screenshot shows a Java IDE interface with the file 'get_list_size_outcome.jsp' open. The code is a JSP page that retrieves the size of a database table and displays it. It includes imports for Integer and String, meta tags for content type and charset, and a title 'Get List Size Outcome Page'. It features an h1 header 'Get list size outcome', a scriptlet block to get the size and surname attributes, and a paragraph displaying the size and providing links to menu.jsp and index.html.

```
<%--  
    Document      : get_list_size_outcome  
    Created on   : 26 Mar 2023, 1:45:32 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>Get List Size Outcome Page</title>  
    </head>  
    <body>  
        <h1>Get list size outcome</h1>  
        <%  
            Integer size = (Integer)request.getAttribute("size");  
            String surname = (String)request.getAttribute("surname");  
        %>  
        <p>  
            There are <b><%=size%></b> records in the database.  
            Please click <a href="menu.jsp">here</a> to get back to the menu page or <a href="index.html">here</a>  
            to the main page.  
        </p>  
    </body>  
</html>
```

View the complete project structure of **PersonWebApp**.



Compile the project.

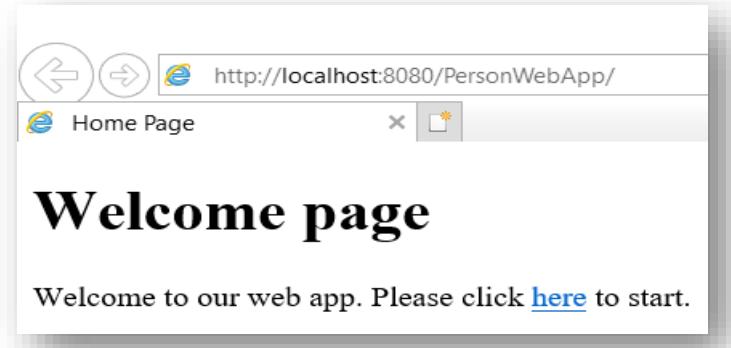
```
Output - PersonWebApp (clean,dist) × | library-inclusion-in-archive:  
Copying 1 file to C:\Users\memaniv\Documents\NetBeansProjects\PersonWebApp\build\web\WEB-INF\lib  
library-inclusion-in-manifest:  
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\PersonWebApp\build\empty  
Created dir: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

Deploy the project.

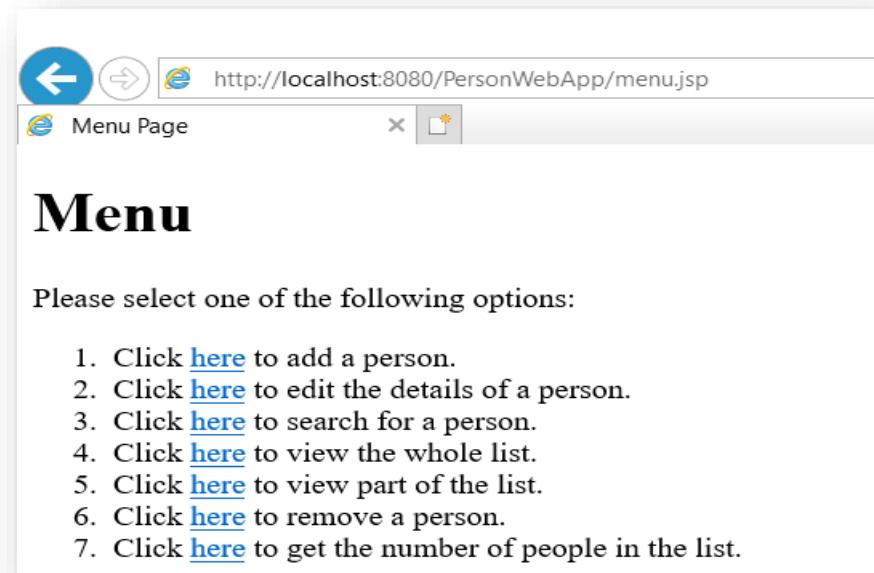
```
Output × | PersonWebApp (run-deploy) × | Java DB Database Process × | Glassfish Server 4.1.1 × |  
Info:   PORTABLE UNDERTOW NAMES FOR EJB PERSONfacade: [ejb:global/PERSONwebApp/PERSONfacade,  
Info:   WELD-000900: 2.2.13 (Final)  
WARN:  WELD-001700: Interceptor annotation class javax.ejb.PostActivate not found, interce...  
WARN:  WELD-001700: Interceptor annotation class javax.ejb.PrePassivate not found, interce...  
WARN:  WELD-000411: Observer method [BackedAnnotatedMethod] org.glassfish.sse.impl.Serve...  
WARN:  WELD-000411: Observer method [BackedAnnotatedMethod] private org.glassfish.jersey...  
WARN:  WELD-000411: Observer method [BackedAnnotatedMethod] public org.glassfish.jms.inje...  
WARN:  WELD-000411: Observer method [BackedAnnotatedMethod] public org.glassfish.jms.inje...  
WARN:  WELD-000411: Observer method [BackedAnnotatedMethod] org.glassfish.sse.impl.Serve...  
WARN:  WELD-000411: Observer method [BackedAnnotatedMethod] private org.glassfish.jersey...  
Info:  Loading application [PersonWebApp] at [/PersonWebApp]  
Info:  PersonWebApp was successfully deployed in 5,594 milliseconds.
```

Part E – Run the web application.

Launch the application.

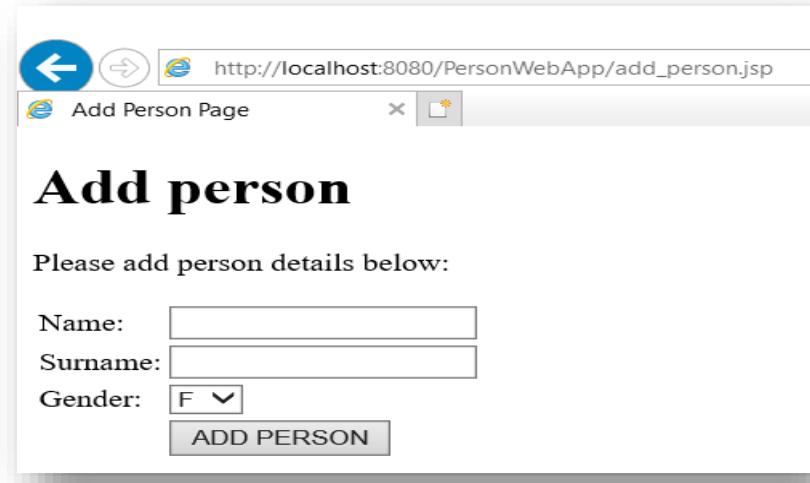


Click on the link



1. Click [here](#) to add a person.
2. Click [here](#) to edit the details of a person.
3. Click [here](#) to search for a person.
4. Click [here](#) to view the whole list.
5. Click [here](#) to view part of the list.
6. Click [here](#) to remove a person.
7. Click [here](#) to get the number of people in the list.

Add a person. Click on the first link.



Fill in the form.

Add person

Please add person details below:

Name:

Surname:

Gender:

Click on the add button.

Add person outcome

Vuyisile Memani has been successfully persisted into the database. Please click [here](#) to get back to the menu page or [here](#) to the main page.

Go back to the menu page.

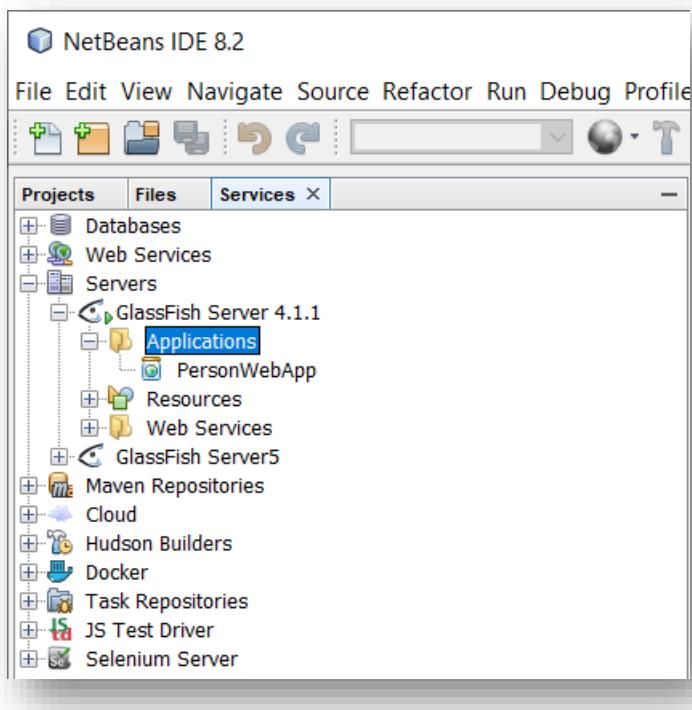
Menu

Please select one of the following options:

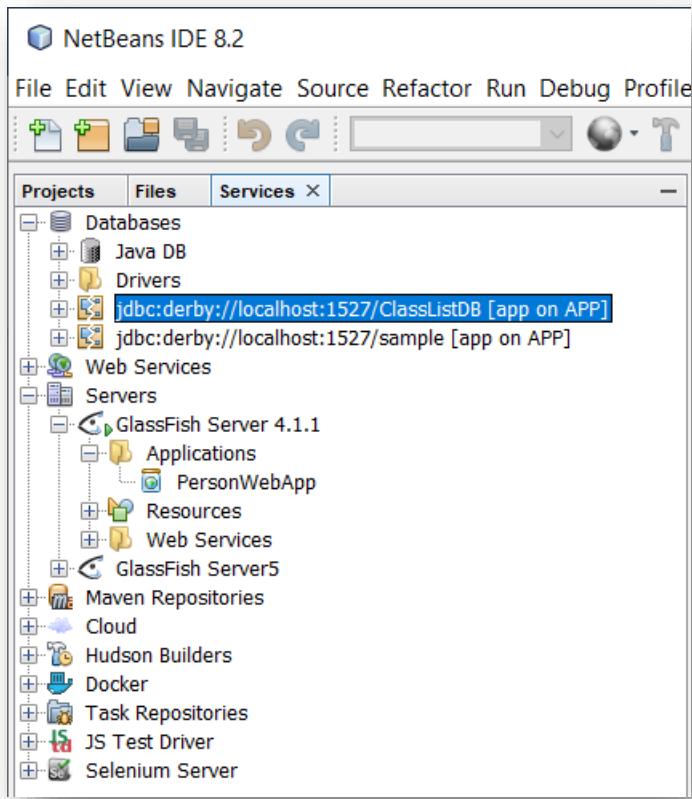
1. Click [here](#) to add a person.
2. Click [here](#) to edit the details of a person.
3. Click [here](#) to search for a person.
4. Click [here](#) to view the whole list.
5. Click [here](#) to view part of the list.
6. Click [here](#) to remove a person.
7. Click [here](#) to get the number of people in the list.

Add four more people into the database. To view the records in the database, do the following:

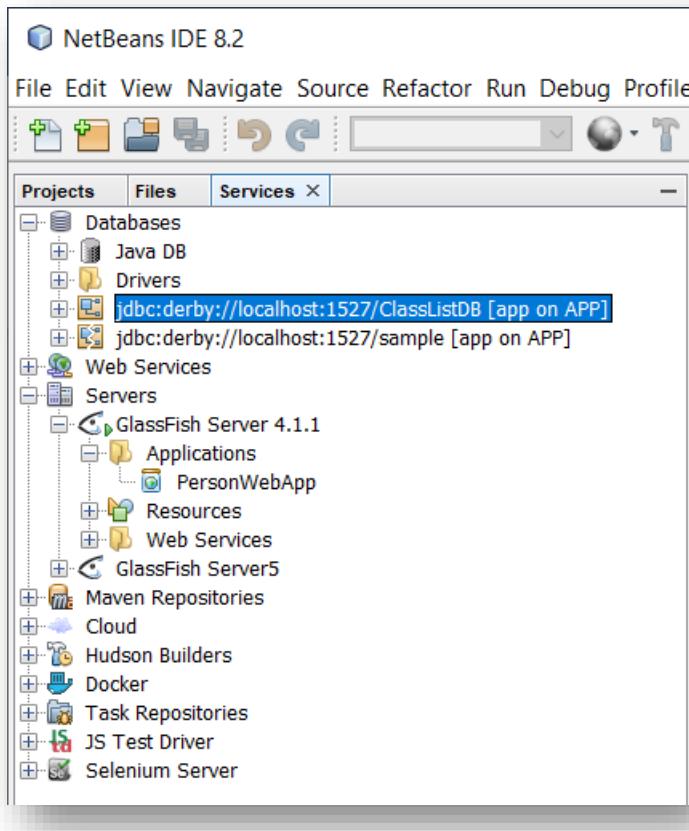
- ✓ Select the **Services** tab.



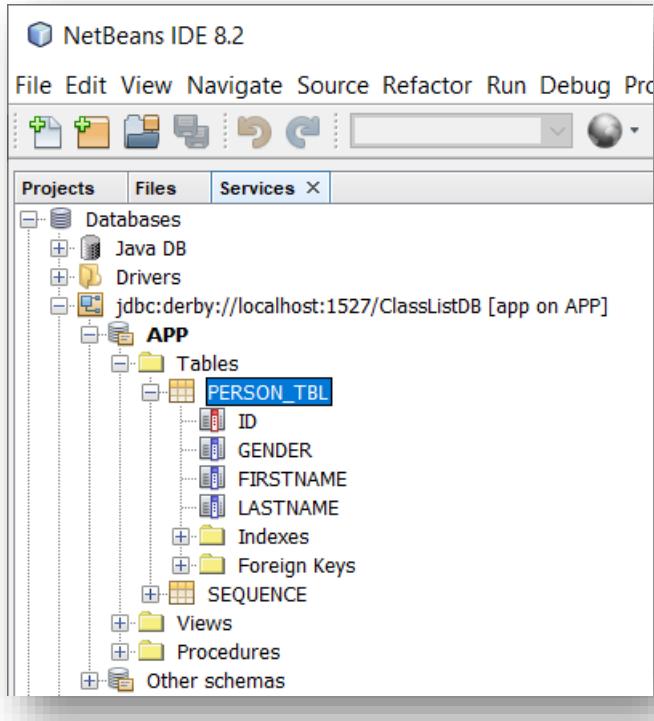
- ✓ Expand **Databases**.



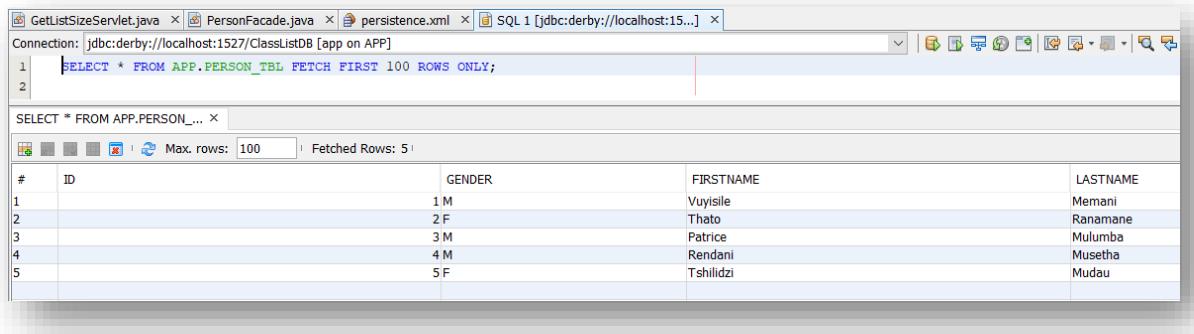
- ✓ Right-click on **ClassListDB** and select **Connect**.



- ✓ Expand the database.



- ✓ Right-click on **Person_TBL** and select **View Data**.



The screenshot shows a Java IDE interface with several tabs at the top: GetListSizeServlet.java, PersonFacade.java, persistence.xml, and SQL 1 [jdbc:derby://localhost:1527/ClassListDB [app on APP]]. The SQL tab contains the following code:

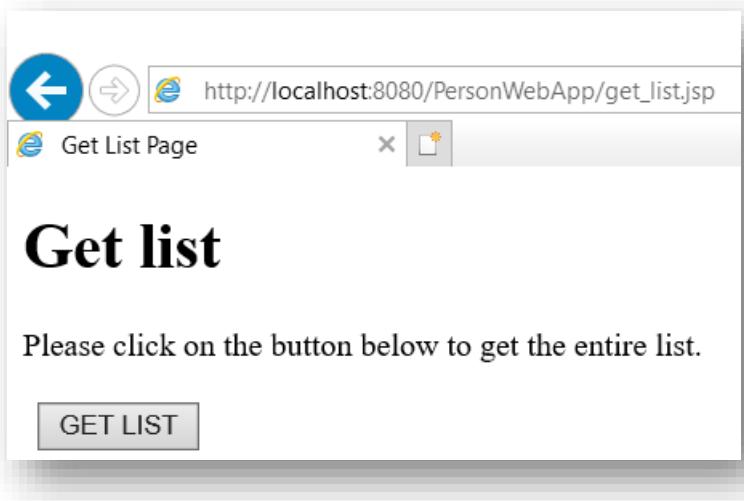
```
1 | SELECT * FROM APP.PERSON_TBL FETCH FIRST 100 ROWS ONLY;
2 |
```

Below the code is a table titled "SELECT * FROM APP.PERSON_...". The table has columns: #, ID, GENDER, FIRSTNAME, and LASTNAME. It displays 5 rows of data:

#	ID	GENDER	FIRSTNAME	LASTNAME
1		1 M	Vuyisile	Memani
2		2 F	Thato	Ranamane
3		3 M	Patrice	Mulumba
4		4 M	Rendani	Musetha
5		5 F	Tshilidzi	Mudau

View all the records.

- ✓ Select option **4** on the menu page.



- ✓ Click on the button.

The screenshot shows a web browser window with the URL <http://localhost:8080/PersonWebApp/GetListServlet.do>. The title bar says "Get List Outcome Page". The main content area has a large heading "Get list outcome". Below it, a message says "Below is the list of people retrieved from the database." followed by a list of five people with their ID, name, and surname:

ID:	1
Name:	Vuyisile
Surname:	Memani
ID:	2
Name:	Thato
Surname:	Ranamane
ID:	3
Name:	Patrice
Surname:	Mulumba
ID:	4
Name:	Rendani
Surname:	Musetha
ID:	5
Name:	Tshilidzi
Surname:	Mudau

Please click [here](#) to get back to the menu page or [here](#) to the main page.

View the number of records.

- ✓ Select option 7 on the menu page.

The screenshot shows a web browser window with the URL http://localhost:8080/PersonWebApp/get_list_size.jsp. The title bar says "Get List Size Page". The main content area has a large heading "Get list size". Below it, a message says "Please click on the button below to get the list size." followed by a single button:

GET LIST SIZE

- ✓ Click on the button.

The screenshot shows a web browser window with the URL <http://localhost:8080/PersonWebApp/GetListSizeServlet.do>. The title bar says "Get List Size Outcome Page". The main content area displays the heading "Get list size outcome" and a message stating "There are 5 records in the database. Please click [here](#) to get back to the menu page or [here](#) to the main page."

Change the surname of Rendani, from **Musetha** to **Baloyi**.

- ✓ Select option **2** on the menu page.

The screenshot shows a web browser window with the URL http://localhost:8080/PersonWebApp/edit_person.jsp. The title bar says "Edit Person Page". The main content area displays the heading "Edit person" and a message "Please enter the details of the person to edit." Below this are three input fields: "Target ID:" with an empty input box, "New surname:" with an empty input box, and a "EDIT PERSON" button.

- ✓ Enter **4** for **ID**, and **Baloyi** as the new surname.

The screenshot shows a web browser window with the URL http://localhost:8080/PersonWebApp/edit_person.jsp. The title bar says "Edit Person Page". The main content area displays the heading "Edit person" and a message "Please enter the details of the person to edit." Below this are three input fields: "Target ID:" with the value "4", "New surname:" with the value "Baloyi", and a "EDIT PERSON" button.

- ✓ Click on the button.

Baloyi [4] has been successfully edited in the database. Please click [here](#) to get back to the menu page or [here](#) to the main page.

- ✓ View the updated list.

Below is the list of people retrieved from the database.

ID:	1
Name:	Vuyisile
Surname:	Memani
ID:	2
Name:	Thato
Surname:	Ranamane
ID:	3
Name:	Patrice
Surname:	Mulumba
ID:	4
Name:	Rendani
Surname:	Baloyi
ID:	5
Name:	Tshilidzi
Surname:	Mudau

Please click [here](#) to get back to the menu page or [here](#) to the main page.

Search for a person.

- ✓ Select option 3 on the menu page.

The screenshot shows a web browser window with the URL http://localhost:8080/PersonWebApp/search_person.jsp. The title bar says "Search Person Page". The main content area has a large heading "Search person" and a message "Please enter the ID of the person to search for.". Below this is a form with a label "ID:" followed by an input field containing "5" and a "SEARCH" button.

- ✓ Enter ID number 5.

The screenshot shows a web browser window with the URL http://localhost:8080/PersonWebApp/search_person.jsp. The title bar says "Search Person Page". The main content area has a large heading "Search person" and a message "Please enter the ID of the person to search for.". Below this is a form with a label "ID:" followed by an input field containing "5" and a "SEARCH" button.

- ✓ Click on the button.

The screenshot shows a web browser window with the URL <http://localhost:8080/PersonWebApp/SearchPersonServlet.do>. The title bar says "Search Person Outcome Page". The main content area has a large heading "Search person outcome" and a message "Tshilidzi Mudau [5] has been found. Please click [here](#) to get back to the menu page or [here](#) to the main page."

View partial list.

- ✓ Select option **5** on the menu page.

The screenshot shows a web browser window with the URL http://localhost:8080/PersonWebApp/get_partial_list.jsp. The title bar says "Get Partial List Page". The main content area has a large heading "Get partial list". Below it is a text instruction: "Please enter start and end indices of the list below:". There are two input fields: "Start index:" containing "1" and "End index:" containing "3". A "GET PARTIAL LIST" button is at the bottom.

- ✓ Enter **1** and **3**.

The screenshot shows a web browser window with the URL http://localhost:8080/PersonWebApp/get_partial_list.jsp. The title bar says "Get Partial List Page". The main content area has a large heading "Get partial list". Below it is a text instruction: "Please enter start and end indices of the list below:". There are two input fields: "Start index:" containing "1" and "End index:" containing "3". A "GET PARTIAL LIST" button is at the bottom.

- ✓ Click on the button.

The screenshot shows a web browser window with the URL <http://localhost:8080/PersonWebApp/GetPartialListServlet.do>. The title bar says "Get Partial List Outcome Page". The main content area displays the heading "Get partial list outcome" and a list of people from index 1 to 3:

ID:	2
Name:	Thato
Surname:	Ranamane
ID:	3
Name:	Patrice
Surname:	Mulumba
ID:	4
Name:	Rendani
Surname:	Baloyi

Please click [here](#) to get back to the menu page or [here](#) to the main page.

Remove person from the database.

- ✓ Select option **6** from the menu page.

The screenshot shows a web browser window with the URL http://localhost:8080/PersonWebApp/remove_person.jsp. The title bar says "Remove Person Page". The main content area displays the heading "Remove person" and a message: "Please enter the ID of the person to remove." Below it is a form with an input field labeled "ID:" and a button labeled "REMOVE PERSON".

- ✓ Enter ID number 3.

Please enter the ID of the person to remove.

ID:

REMOVE PERSON

- ✓ Click on the button.

here to get back to the menu page or [here](#) to the main page.'"/>

Remove person outcome

The person with id number 3 has been successfully removed from the database. Please click [here](#) to get back to the menu page or [here](#) to the main page.

- ✓ View all the records.

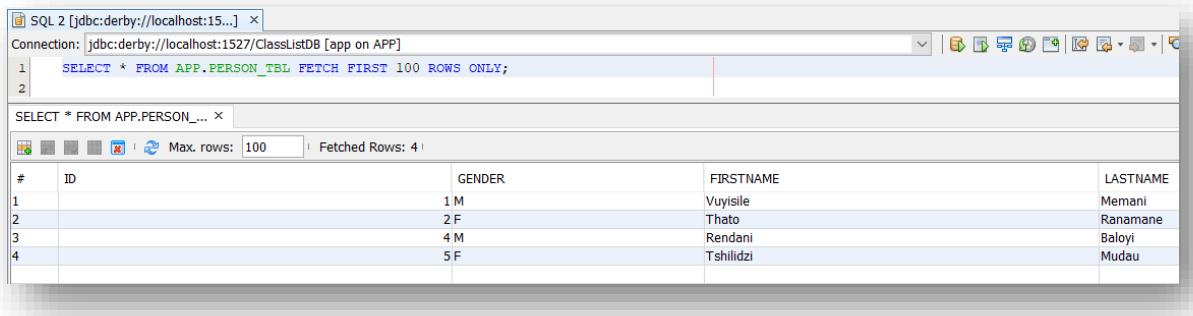
Get list outcome

Below is the list of people retrieved from the database.

ID:	1
Name:	Vuyisile
Surname:	Memani
ID:	2
Name:	Thato
Surname:	Ranamane
ID:	4
Name:	Rendani
Surname:	Baloyi
ID:	5
Name:	Tshilidzi
Surname:	Mudau

Please click [here](#) to get back to the menu page or [here](#) to the main page.

✓ View from the actual database.



The screenshot shows a SQL Server Management Studio window with two panes. The left pane contains a query editor with the following SQL code:

```
1  SELECT * FROM APP.PERSON_TBL FETCH FIRST 100 ROWS ONLY;
2 |
```

The right pane displays the results of the query as a table:

#	ID	GENDER	FIRSTNAME	LASTNAME
1		1 M	Vuyisile	Mamani
2		2 F	Thato	Ranamane
3		4 M	Rendani	Baloyi
4		5 F	Tshilidzi	Mudau

Details: The table has five columns: #, ID, GENDER, FIRSTNAME, and LASTNAME. The data consists of four rows, each with a unique ID (1, 2, 3, 4) and gender (M or F). The first three rows have IDs 1, 2, and 3 respectively, while the fourth row has an empty ID field. The genders are 1 M, 2 F, 4 M, and 5 F. The first three rows have FIRSTNAME values Vuyisile, Thato, and Rendani, and the last row has Tshilidzi. The last three rows have LASTNAME values Mamani, Ranamane, and Baloyi, and the fourth row has Mudau.

8.3.2 Using basic annotations and secondary tables.

Example

Create a web application that will perform part of the **CRUD** (Create Read Update Delete) operations on a list of bank account holders. Specifically we want the application to be able to **create**, **search** and **retrieve** the list. An account holder has an **id**, **full name**, street, **city**, **code**, **cell number**, **email address**, **balance**, and **creation date** attributes. The **street**, **city** and **code** attributes of each account holder must be kept in a table of its own called **Address**. The **cell numbers** and **email addresses** of account holders must be kept another secondary table called **Contacts**. The table below shows all the fields of an account holder and the applicable constraints for each field.

Field	Constraint
id	Primary key.
fullName	Must be a string. Cannot be nullable. The maximum length is 50.
street	Must be a string. Can be nullable. The maximum length is 20.
city	Must be a string. Can be nullable. The maximum length is 20.
code	Must be a string. Can be nullable. The maximum length is 20.
cellNo	Must be a string. Can be nullable. The maximum length is 20.
emailAddress	Must be a string. Can be nullable. The maximum length is 20.
balance	Must be a double. Cannot be nullable.
creationDate	Must be a Date.

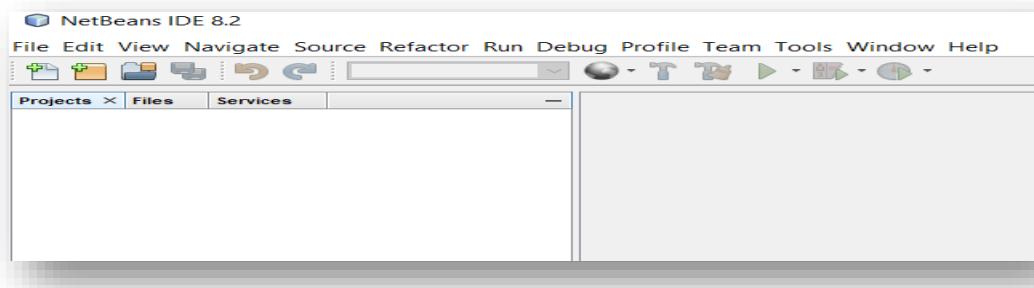
Solution approach

The solution to this problem is going to be done in five parts, namely:

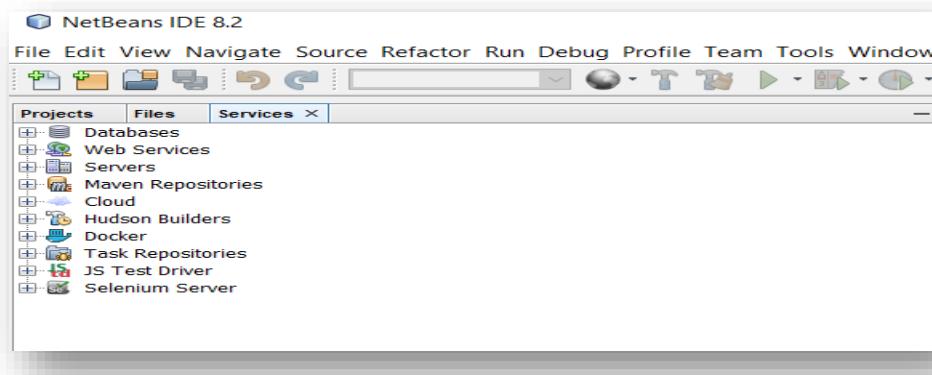
- **Part A:** we create a database.
- **Part B:** we establish connection between the created database and GlassFish.
- **Part C:** we create an EJB module. The module will have an entity with business logic.
- **Part D:** we create a client web application.
- **Part E:** we run the client web application.

Part A - Create a database

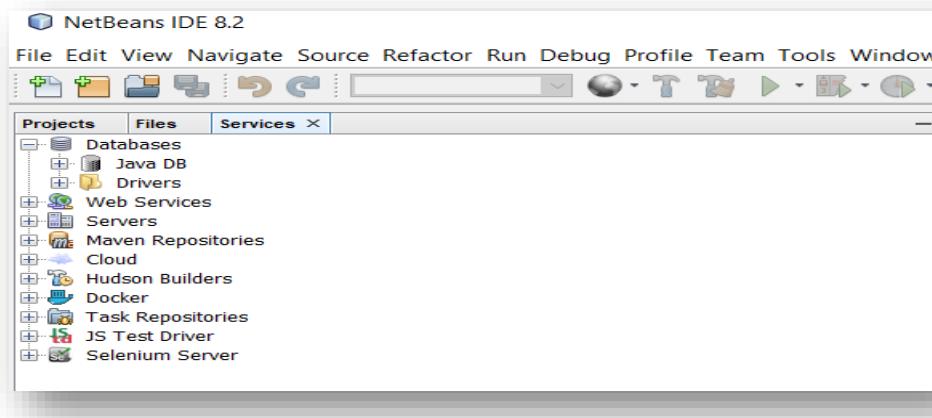
Launch NetBeans.



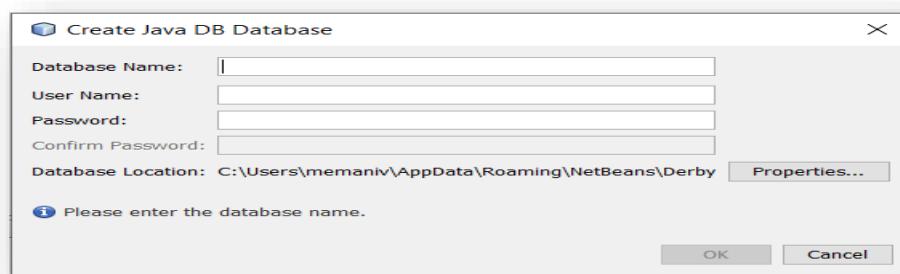
Click on the **Services** tab.



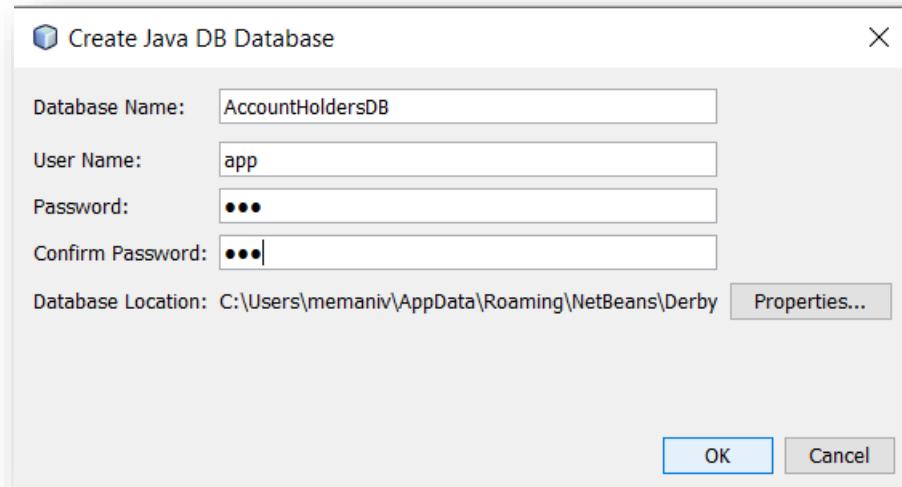
Expand the **Databases** option.



Right-click on **Java DB** and select **Create Database**.



Fill-in the form. I made the password to be **123**.

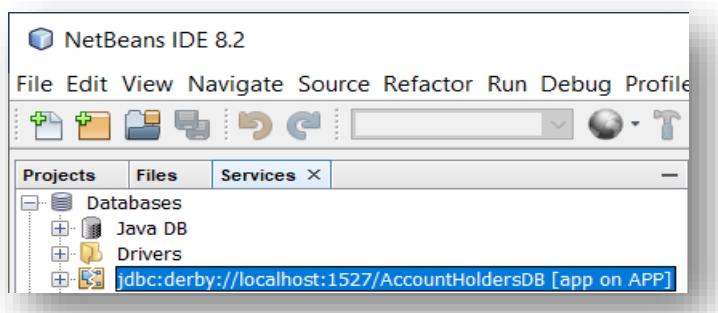


Click **OK**.

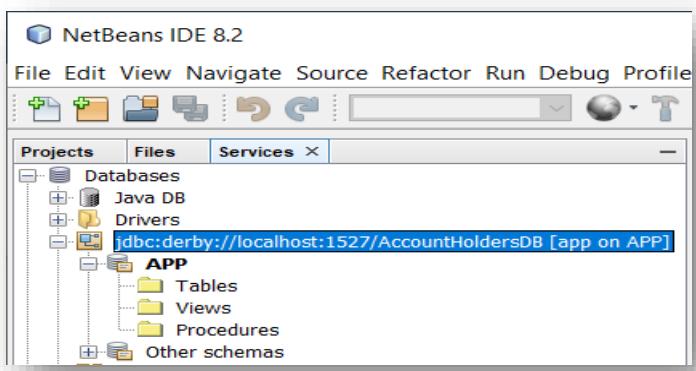
- ✓ An output message confirming that **Derby** has started and ready to accept connections on port 1527 is shown.

```
Output - Java DB Database Process
Tue Mar 28 21:28:25 CAT 2023 : Security manager installed using the Basic server security policy.
Tue Mar 28 21:28:26 CAT 2023 : Apache Derby Network Server - 10.11.1.2 - (1629631) started and ready to accept connections on port 1527
```

- ✓ A database is created.

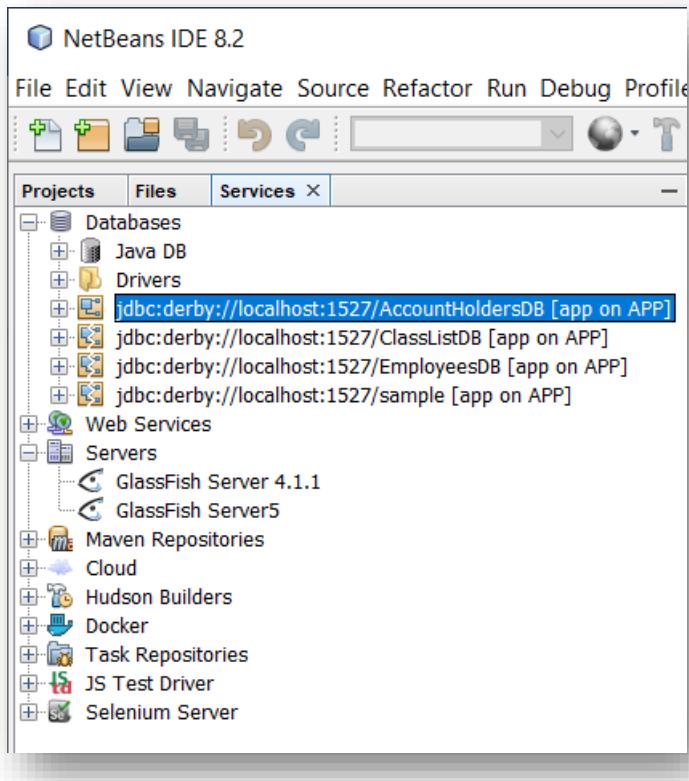


Right-click on the database and select **Connect**.

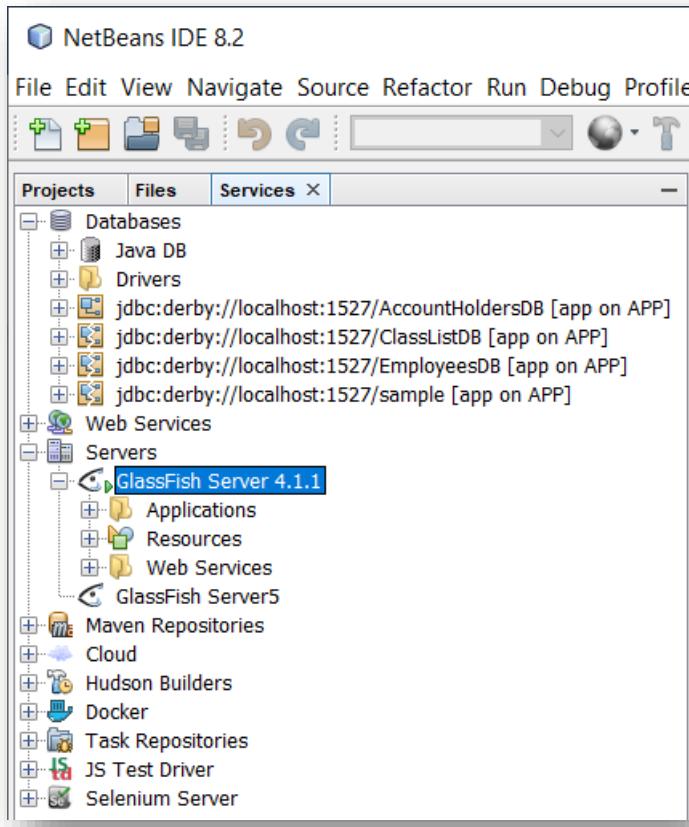


Part B – Connect the database to the application server.

Click on the **Services** tab.



Start the server.



Right-click on the server and select **View Domain Admin Console**.

The screenshot shows the GlassFish Admin Console interface. The top navigation bar displays the URL <http://localhost:4848/common/index.jsf>, the title "GlassFish Console - Comm...", and the user information "User: admin | Domain: domain1 | Server: localhost". Below the title, the header reads "GlassFish™ Server Open Source Edition". The left sidebar, titled "Common Tasks", contains a tree view of server components: Domain, server (Admin Server), Clusters, Standalone Instances, Nodes, Applications, Lifecycle Modules, Monitoring Data, Resources (Concurrent Resources, Connectors, JDBC, JMS Resources, JNDI, JavaMail Sessions, Resource Adapter Configs), Configurations (default-config, server-config), and Update Tool. The main content area is titled "GlassFish Console - Common Tasks" and includes sections for GlassFish News, Deployment (List Deployed Applications, Deploy an Application), Administration (Change Administrator Password, List Password Aliases), and Monitoring (Monitoring Data).

Under **Common Tasks** panel, expand **JDBC**.

The screenshot shows the GlassFish Admin Console with the "JDBC" section expanded in the "Common Tasks" sidebar. The sidebar now highlights the "Resources" node, and under it, the "JDBC" node is selected, revealing its sub-options: JDBC Resources and JDBC Connection Pools. The main content area is titled "JDBC" and displays two items: "JDBC Resources" and "JDBC Connection Pools". The rest of the sidebar and main content area remain the same as in the previous screenshot.

Modify the Connection pool to include the **AccountHoldersDB**.

- ✓ Click on **JDBC Connection Pools**.

JDBC Connection Pools					
To store, organize, and retrieve data, most applications use relational databases. Java EE applications access relational databases through the JDBC API. Before an application can access a database, it must get a connection.					
Pools (3)					
Select	Pool Name	Resource Type	Classname	Description	
<input type="checkbox"/>	DerbyPool	javax.sql.DataSource	org.apache.derby.jdbc.ClientDataSource		
<input type="checkbox"/>	SamplePool	javax.sql.DataSource	org.apache.derby.jdbc.ClientDataSource		
<input type="checkbox"/>	TimerPool	javax.sql.XADataSource	org.apache.derby.jdbc.EmbeddedXADataSource		

- ✓ Click on the **DerbyPool** link.

General	Advanced	Additional Properties
Edit JDBC Connection Pool		
Modify an existing JDBC connection pool. A JDBC connection pool is a group of reusable connections for a particular database.		
Load Defaults Flush Ping		
General Settings		
Pool Name:	DerbyPool	
Resource Type:	javax.sql.DataSource	
Must be specified if the datasource class implements more than 1 of the interface.		
Datasource Classname:	org.apache.derby.jdbc.ClientDataSource	
Vendor-specific classname that implements the DataSource and/or XADataSource APIs		
Driver Classname:		
Vendor-specific classname that implements the java.sql.Driver interface.		
Ping:	<input type="checkbox"/> Enabled	
When enabled, the pool is pinged during creation or reconfiguration to identify and warn of any erroneous values for its attributes		
Deployment Order:	100	
Specifies the loading order of the resource at server startup. Lower numbers are loaded first.		
Description:		
Pool Settings		
Initial and Minimum Pool Size:	8	Connections
Minimum and initial number of connections maintained in the pool		
Maximum Pool Size:	32	Connections
Maximum number of connections that can be created to satisfy client requests		
Pool Resize Quantity:	2	Connections

- ✓ Click on **Additional Properties**.

General	Advanced	Additional Properties
Edit JDBC Connection Pool Properties		
Modify properties of an existing JDBC connection pool.		
Pool Name: DerbyPool		
Additional Properties (6)		
Edit Delete Add Property Delete Properties		
Select	Name	Value
<input type="checkbox"/>	PortNumber	1527
<input type="checkbox"/>	Password	APP
<input type="checkbox"/>	User	APP
<input type="checkbox"/>	serverName	localhost
<input type="checkbox"/>	DatabaseName	sun-appserv-samples
<input type="checkbox"/>	connectionAttributes	,create=true

- ✓ Click on **Add Property** and add a **url** property with the following value:

jdbc:derby://localhost:1527/AccountHoldersDB

- ✓ Delete the **connectionAttributes** property.

General Advanced Additional Properties

Edit JDBC Connection Pool Properties

Modify properties of an existing JDBC connection pool.

Pool Name: DerbyPool

Additional Properties (6)		
Select	Name	Value
<input type="checkbox"/>	PortNumber	1527
<input type="checkbox"/>	Password	123
<input type="checkbox"/>	User	app
<input type="checkbox"/>	serverName	localhost
<input type="checkbox"/>	DatabaseName	AccountHoldersDB
<input type="checkbox"/>	url	jdbc:derby://localhost:1527/AccountHoldersD

- ✓ Save the changes by clicking the **Save** button.

General Advanced Additional Properties

New values successfully saved.

Edit JDBC Connection Pool Properties

Modify properties of an existing JDBC connection pool.

Pool Name: DerbyPool

Additional Properties (6)		
Select	Name	Value
<input type="checkbox"/>	PortNumber	1527
<input type="checkbox"/>	Password	123
<input type="checkbox"/>	User	app
<input type="checkbox"/>	serverName	localhost
<input type="checkbox"/>	DatabaseName	AccountHoldersDB
<input type="checkbox"/>	url	jdbc:derby://localhost:1527/AccountHoldersD

- ✓ Check if the created connection pool is working. Do this by selecting the General tab and then click on the **Ping** button.

The screenshot shows the 'Edit JDBC Connection Pool' dialog with the 'General' tab selected. At the top right, there is a yellow button labeled 'Ping Succeeded' with a green checkmark icon. Below it, the title 'Edit JDBC Connection Pool' is displayed, followed by a brief description: 'Modify an existing JDBC connection pool. A JDBC connection pool is a group of reusable connections for a particular database.' Below the description are several configuration fields:

- Pool Name:** DerbyPool
- Resource Type:** javax.sql.DataSource (selected from a dropdown menu)
- Datasource Classname:** org.apache.derby.jdbc.ClientDataSource
- Driver Classname:** (empty field)
- Ping:** Enabled
- Deployment Order:** 100
- Description:** (empty field)

Confirm that the data source (resource) points to the DerbyPool.

- ✓ Still under **JDBC**, click on **JDBC Resources**.

The screenshot shows the 'JDBC Resources' list with the title 'JDBC Resources' and a sub-instruction: 'JDBC resources provide applications with a means to connect to a database.' Below the title is a table titled 'Resources (3)' with the following data:

Select	JNDI Name	Logical JNDI Name	Enabled	Connection Pool
<input type="checkbox"/>	jdbc/_TimerPool		<input checked="" type="checkbox"/>	_TimerPool
<input type="checkbox"/>	jdbc/_default	java:comp/DefaultDataSource	<input checked="" type="checkbox"/>	DerbyPool
<input type="checkbox"/>	jdbc/sample		<input checked="" type="checkbox"/>	SamplePool

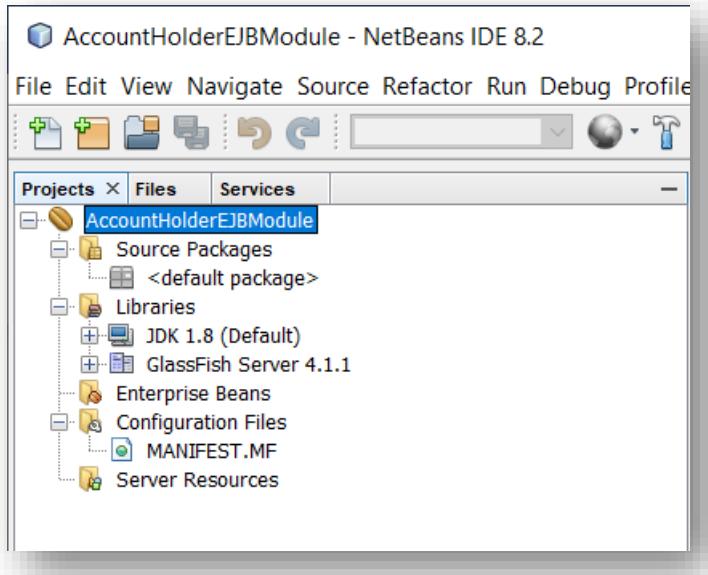
Take note of the purpose of JDBC Resources. It is said:

"JDBC resources provide applications with a means to connect to a database."

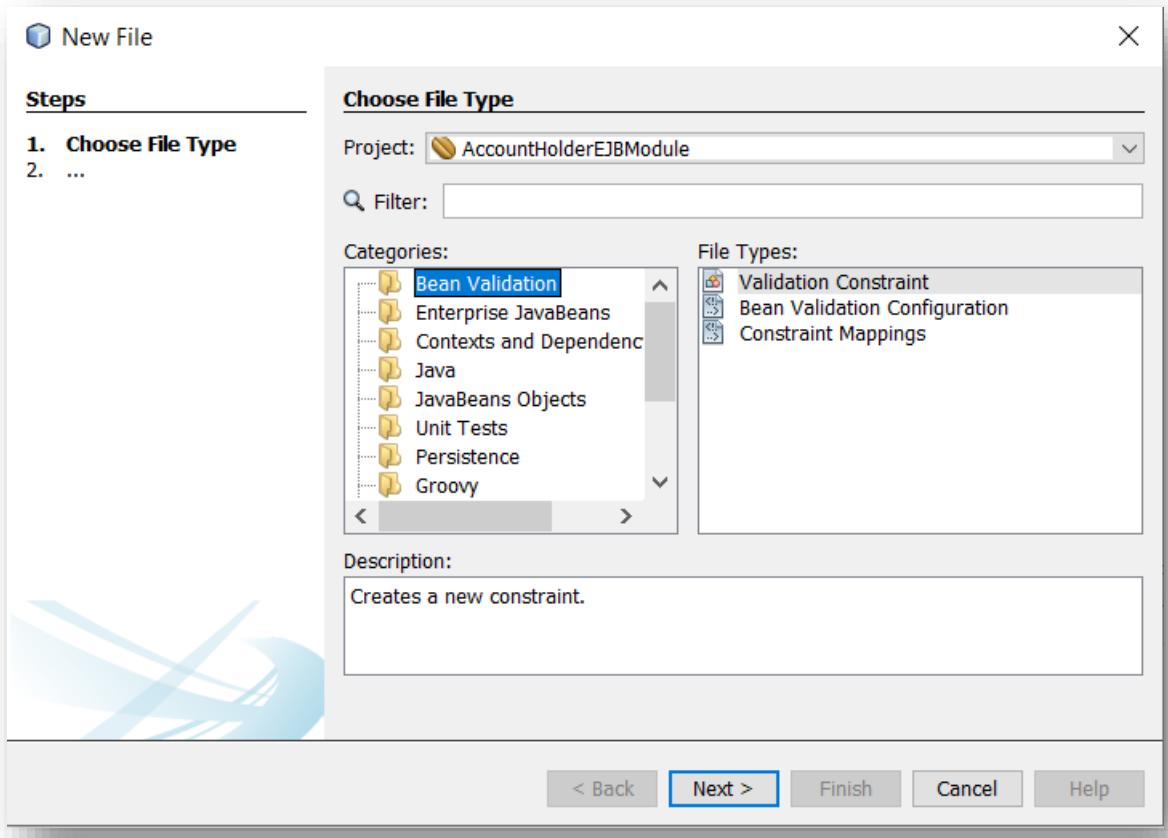
You can also see that the **jdbc/_default** resource is associated with the **DerbyPool** Connection Pool.

Part C – Create an entity.

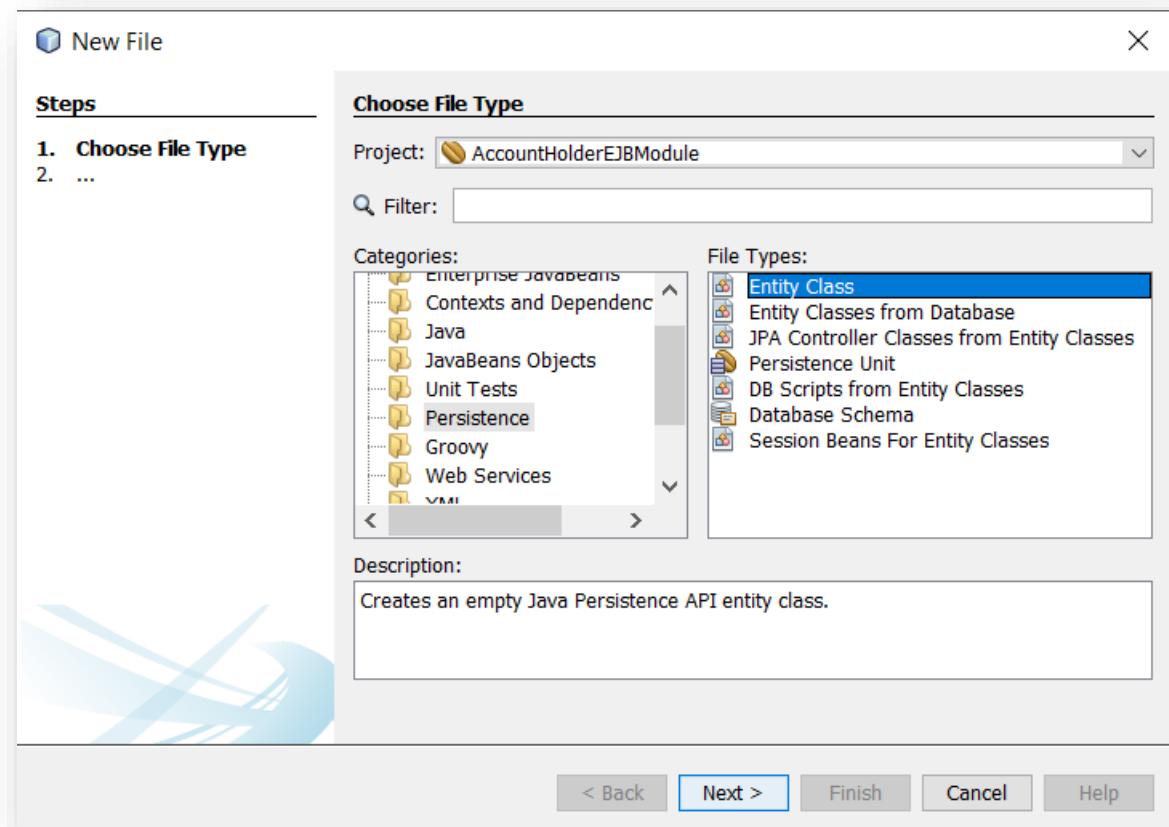
Create an EJB project called **AccountHolderEJBModule**.



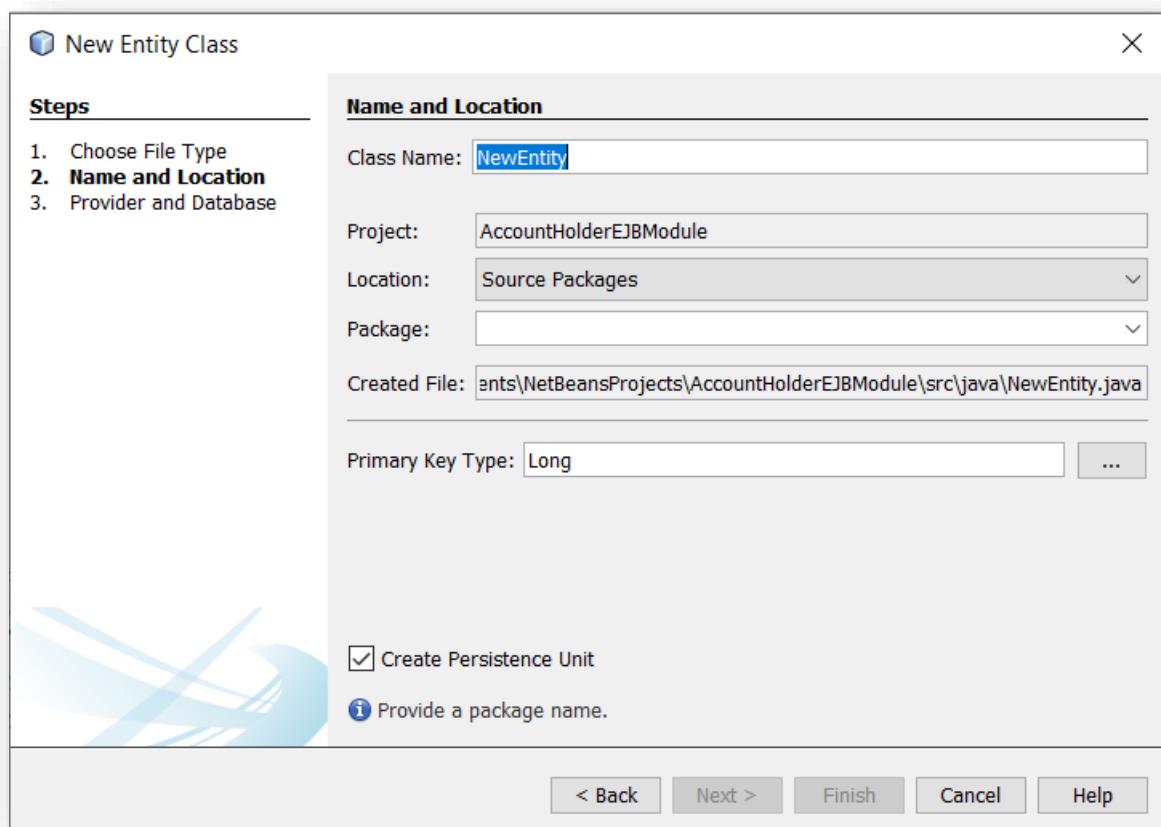
Right-click on the project and select **New | Other**.



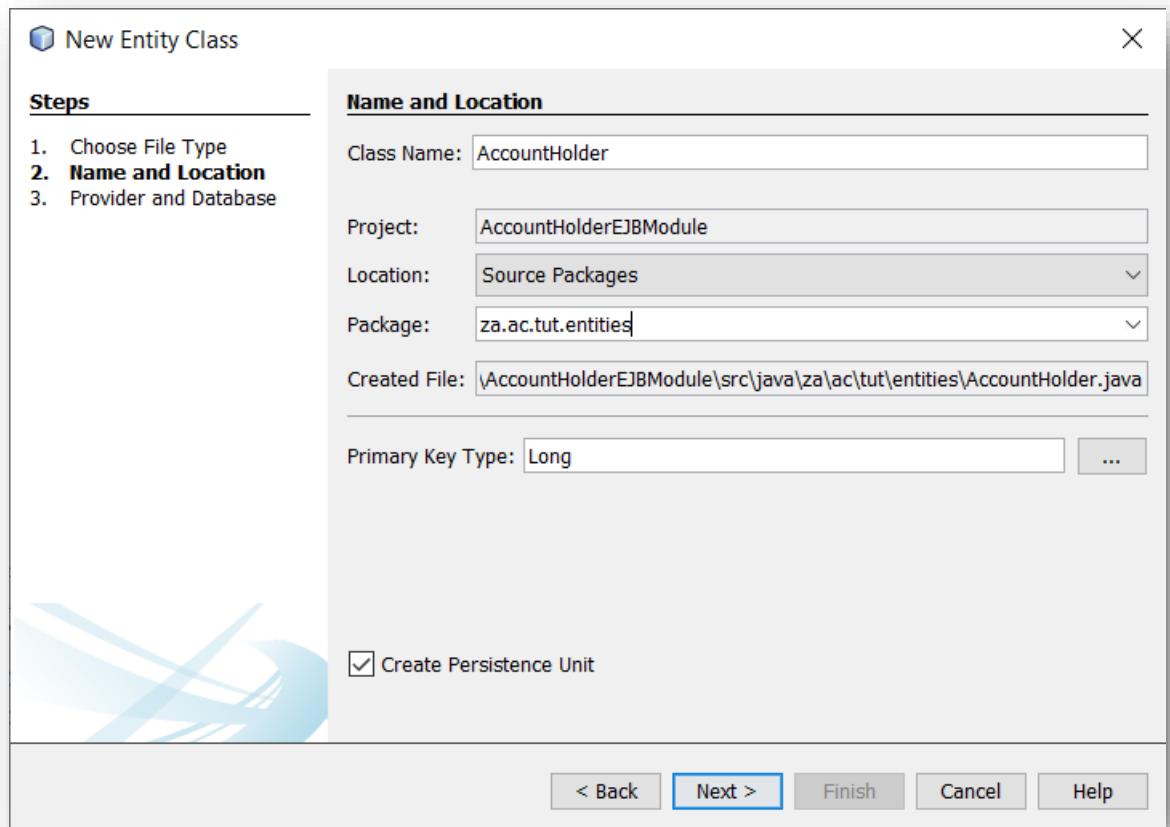
Under **Categories** select **Persistence**, and under **File Types** select **Entity Class**.



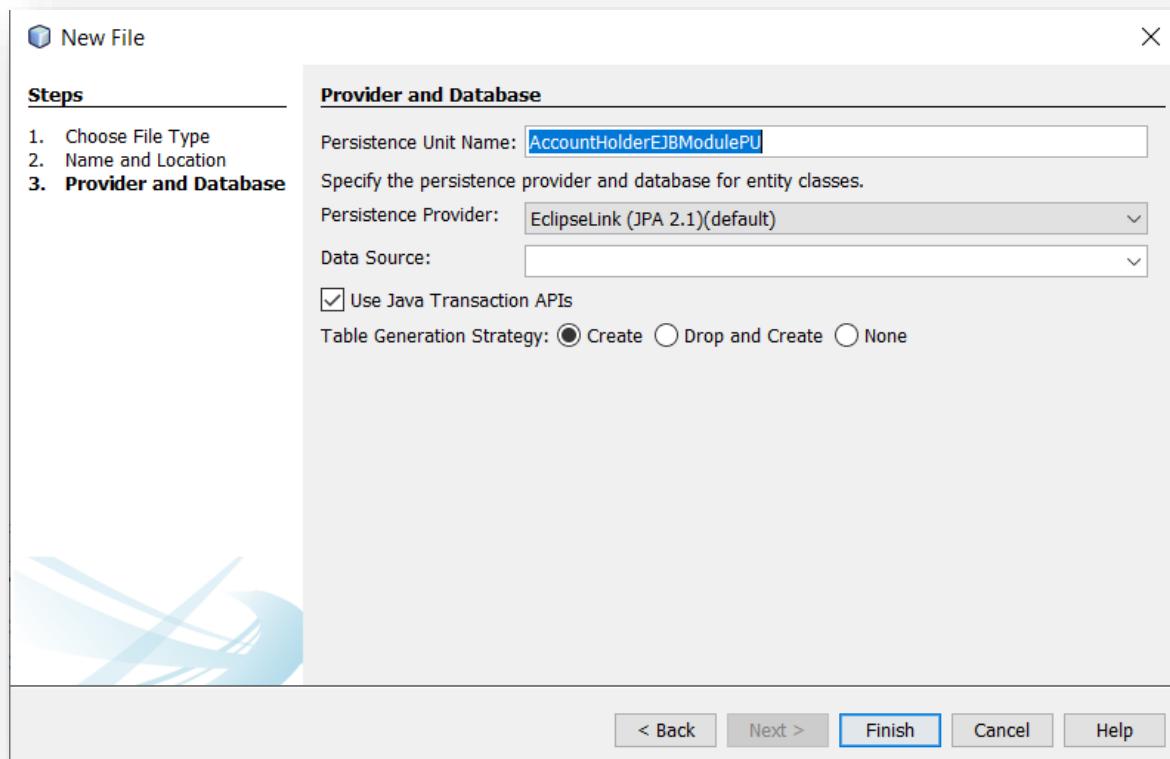
Click **Next**.



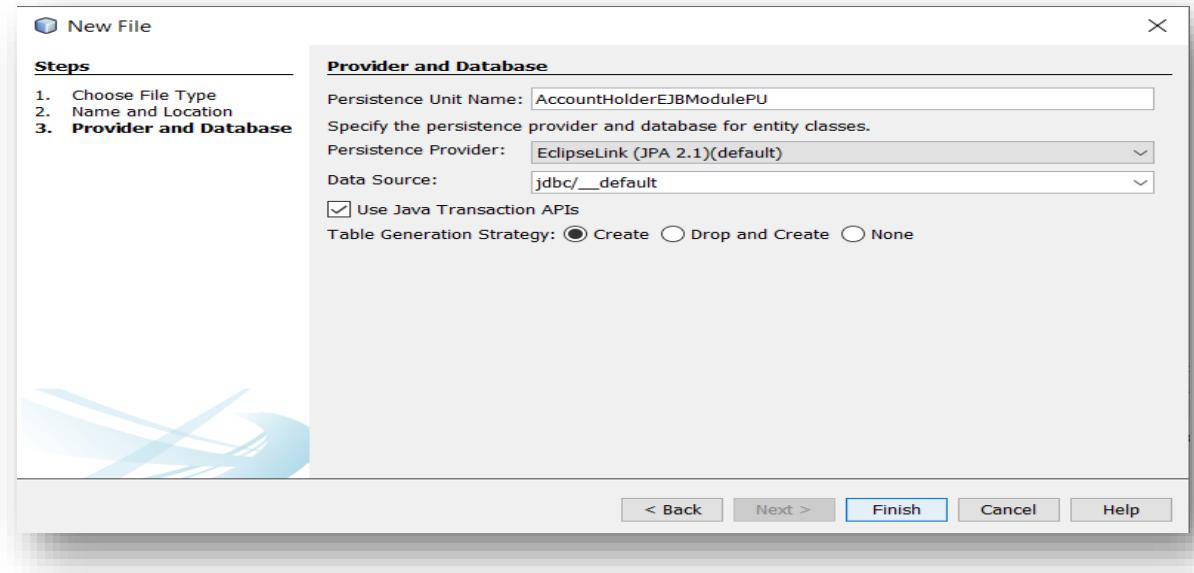
Fill-in the form.



Click **Next**.



Under **Data Source**, select **jdbc/_default**.



Click **Finish**.

The screenshot shows a Java code editor window with the title bar "AccountHolder.java". The menu bar includes "File", "Edit", "Source", "History", and various tool icons. The code itself is a Java class named AccountHolder, which implements Serializable. It imports several Java persistence annotations and classes. The class is annotated with @Entity and @Table(name="ACCOUNTHOLDERS_TBL"). It also has two secondary tables: ADDRESS_TBL and CONTACTS_TBL. The class contains private fields for id, fullName, street, city, code, cellNo, emailAddress, and creationDate. The code is numbered from 1 to 38.

```
1 package za.ac.tut.entities;
2
3 import java.io.Serializable;
4 import java.util.Date;
5 import javax.persistence.Column;
6 import javax.persistence.Entity;
7 import javax.persistence.Id;
8 import javax.persistence.SecondaryTable;
9 import javax.persistence.SecondaryTables;
10 import javax.persistence.Table;
11 import javax.persistence.Temporal;
12
13 /**
14  * 
15  * @author MemaniV
16  */
17 @Entity
18 @Table(name="ACCOUNTHOLDERS_TBL")
19 @SecondaryTables({@SecondaryTable(name="ADDRESS_TBL"), @SecondaryTable(name="CONTACTS_TBL")})
20 public class AccountHolder implements Serializable {
21
22     private static final long serialVersionUID = 1L;
23     @Id
24     private Long id;
25     @Column(nullable=false, length=50)
26     private String fullName;
27     @Column(table="ADDRESS_TBL", nullable=true, length=20)
28     private String street;
29     @Column(table="ADDRESS_TBL", nullable=true, length=20)
30     private String city;
31     @Column(table="ADDRESS_TBL", nullable=true, length=20)
32     private String code;
33     @Column(table="CONTACTS_TBL", nullable=true, length=20)
34     private String cellNo;
35     @Column(table="CONTACTS_TBL", nullable=true, length=20)
36     private String emailAddress;
37     @Temporal(javax.persistence.TemporalType.TIMESTAMP)
38     private Date creationDate;
```

```
40  [ ] public AccountHolder() {
41  [ ] }
42
43  [ ] public AccountHolder(Long id, String fullName, String street, String city, String code,
44  [ ]           String cellNo, String emailAddress, Date creationDate) {
45  [ ]     this.id = id;
46  [ ]     this.fullName = fullName;
47  [ ]     this.street = street;
48  [ ]     this.city = city;
49  [ ]     this.code = code;
50  [ ]     this.cellNo = cellNo;
51  [ ]     this.emailAddress = emailAddress;
52  [ ]     this.creationDate = creationDate;
53  [ ]
54
55  [ ]     public String getFullName() {
56  [ ]         return fullName;
57  [ ]     }
58
59  [ ]     public void setFullName(String fullName) {
60  [ ]         this.fullName = fullName;
61  [ ]     }
62
63  [ ]     public String getStreet() {
64  [ ]         return street;
65  [ ]     }
66
67  [ ]     public void setStreet(String street) {
68  [ ]         this.street = street;
69  [ ]     }
70
71  [ ]     public String getCity() {
72  [ ]         return city;
73  [ ]     }
74
75  [ ]     public void setCity(String city) {
76  [ ]         this.city = city;
77  [ ]     }
```

```

79  public String getCode() {
80      return code;
81  }
82
83  public void setCode(String code) {
84      this.code = code;
85  }
86
87  public String getCellNo() {
88      return cellNo;
89  }
90
91  public void setCellNo(String cellNo) {
92      this.cellNo = cellNo;
93  }
94
95  public String getEmailAddress() {
96      return emailAddress;
97  }
98
99  public void setEmailAddress(String emailAddress) {
100     this.emailAddress = emailAddress;
101 }
102
103  public Date getCreationDate() {
104      return creationDate;
105  }
106
107  public void setCreationDate(Date creationDate) {
108      this.creationDate = creationDate;
109  }
110
111  public Long getId() {
112      return id;
113  }

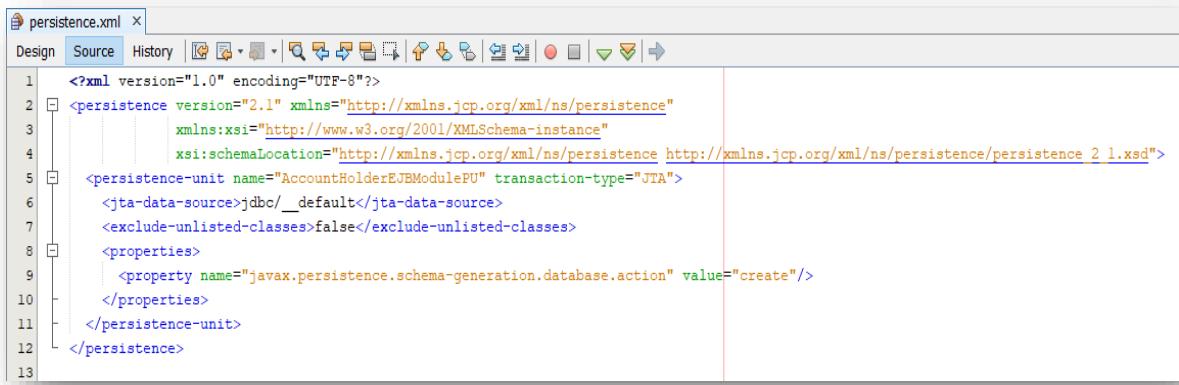
```

```

115  public void setId(Long id) {
116      this.id = id;
117  }
118
119  @Override
120  @Override
121  public int hashCode() {
122      int hash = 0;
123      hash += (id != null ? id.hashCode() : 0);
124      return hash;
125  }
126
127  @Override
128  @Override
129  public boolean equals(Object object) {
130      // TODO: Warning - this method won't work in the case the id fields are not set
131      if (!(object instanceof AccountHolder)) {
132          return false;
133      }
134      AccountHolder other = (AccountHolder) object;
135      if ((this.id == null && other.id != null) || (this.id != null && !this.id.equals(other.id))) {
136          return false;
137      }
138      return true;
139  }
140
141  @Override
142  @Override
143  public String toString() {
144      return "za.ac.tut.entities.AccountHolder[ id=" + id + "]";
145  }

```

View the persistence.xml file.

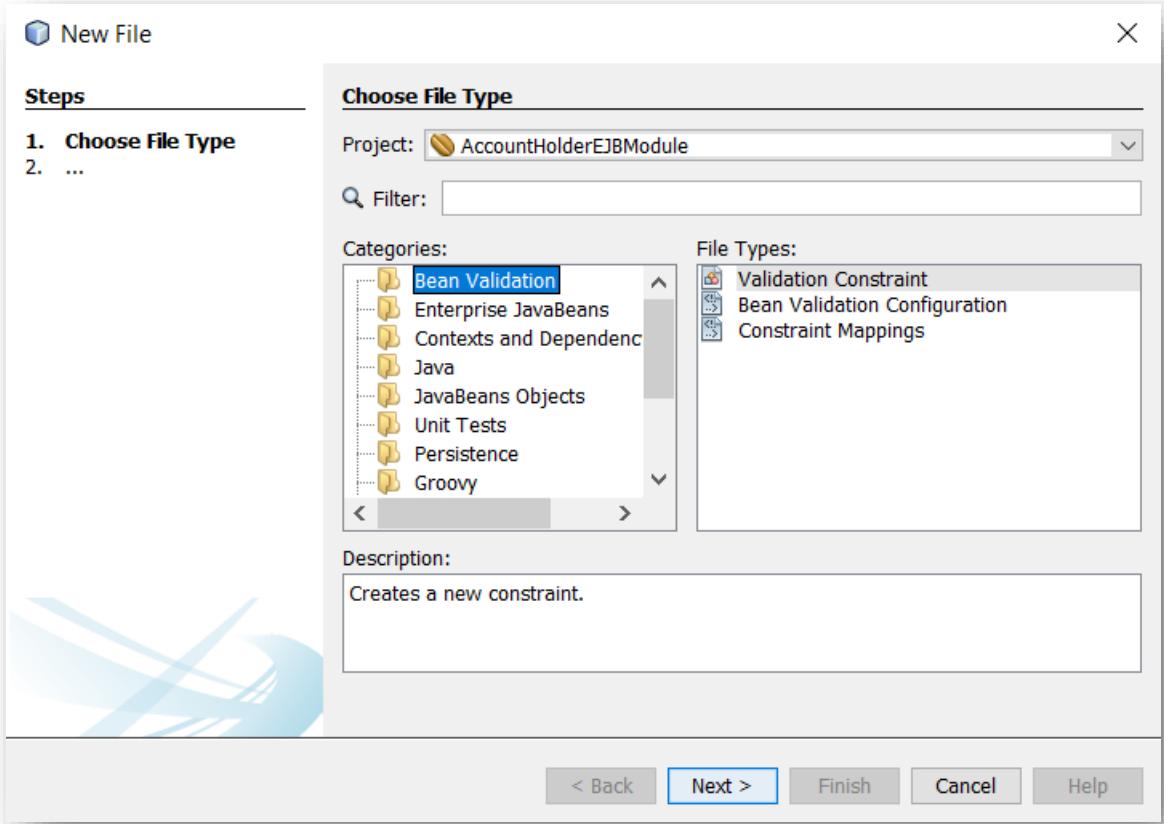


The screenshot shows the Eclipse IDE interface with the 'persistence.xml' file open in the Source tab. The XML code is displayed, defining a persistence unit named 'AccountHolderEJBModulePU' with a transaction type of 'JTA'. It includes a JDBC data source, excludes unlisted classes, and specifies schema generation for the database action.

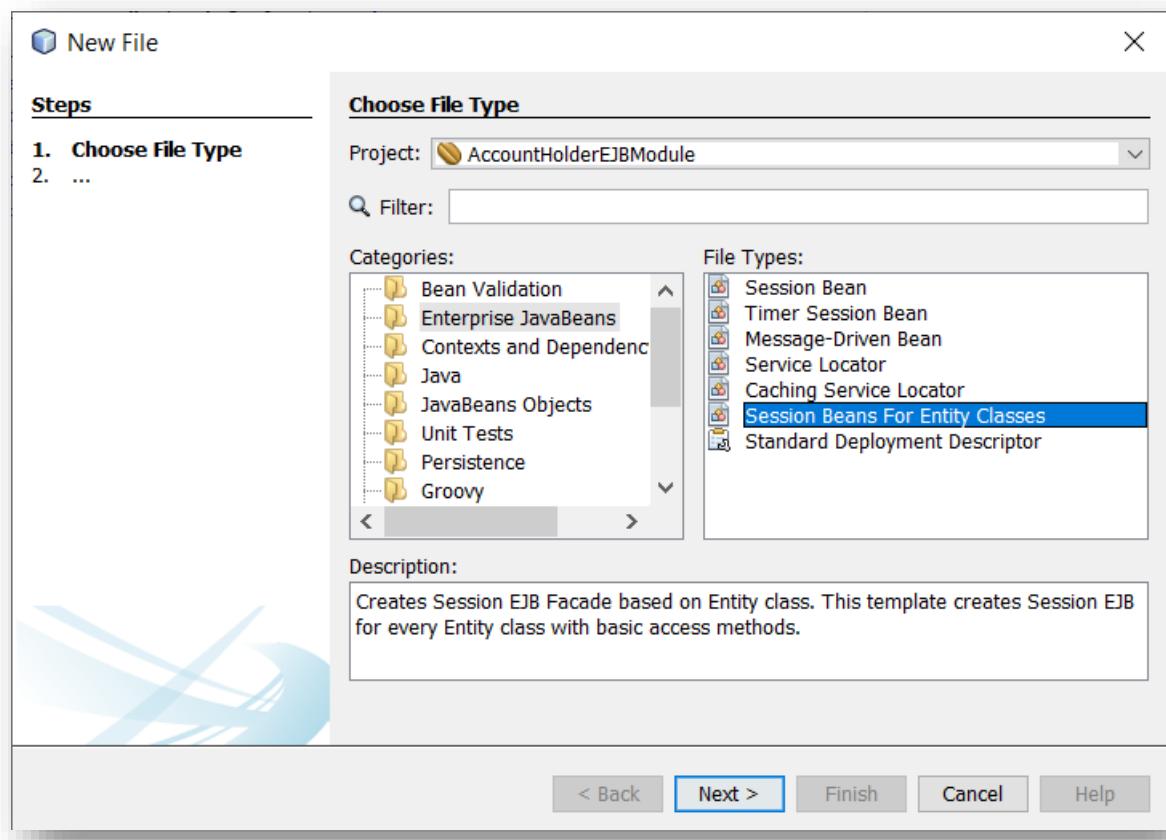
```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
    <persistence-unit name="AccountHolderEJBModulePU" transaction-type="JTA">
        <jta-data-source>jdbc/_default</jta-data-source>
        <exclude-unlisted-classes>false</exclude-unlisted-classes>
        <properties>
            <property name="javax.persistence.schema-generation.database.action" value="create"/>
        </properties>
    </persistence-unit>
</persistence>
```

Create business code (Create account holder, search for account holder and display all account holders) for the entity. Perform the following steps:

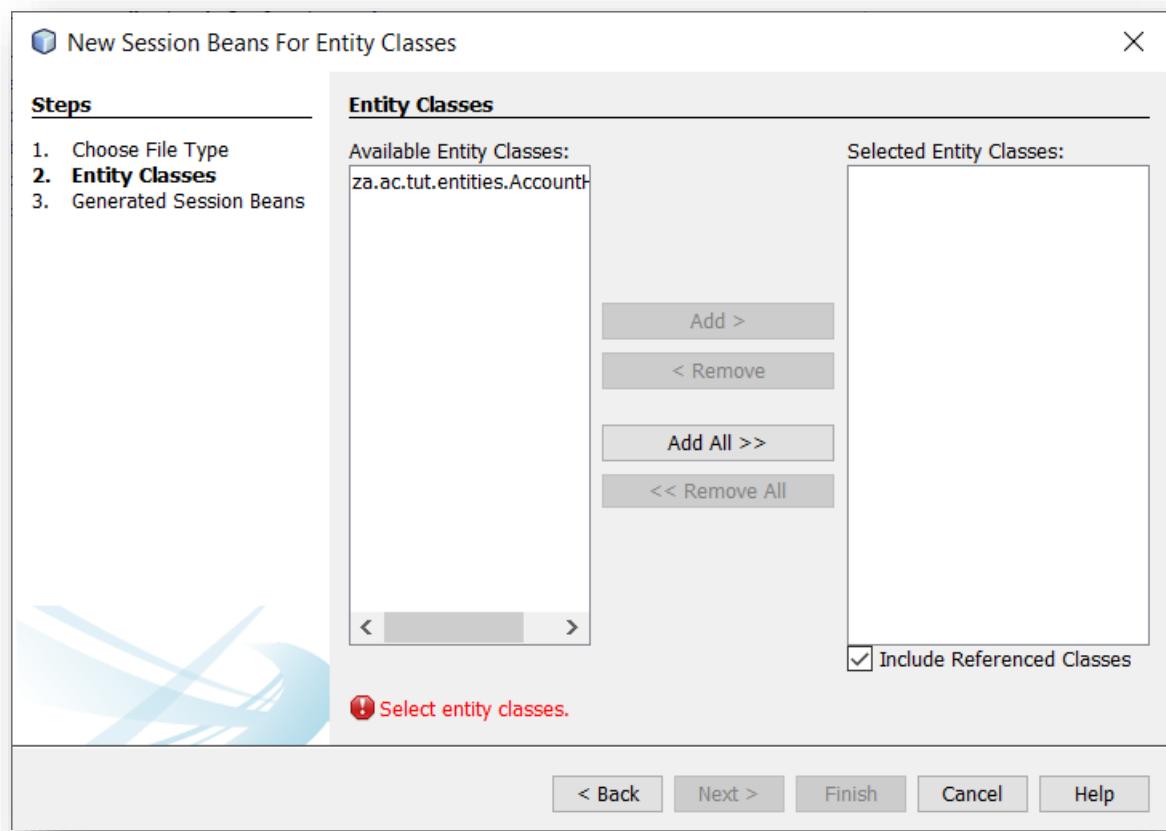
- ✓ Right-click on the project and select **New | Other**.



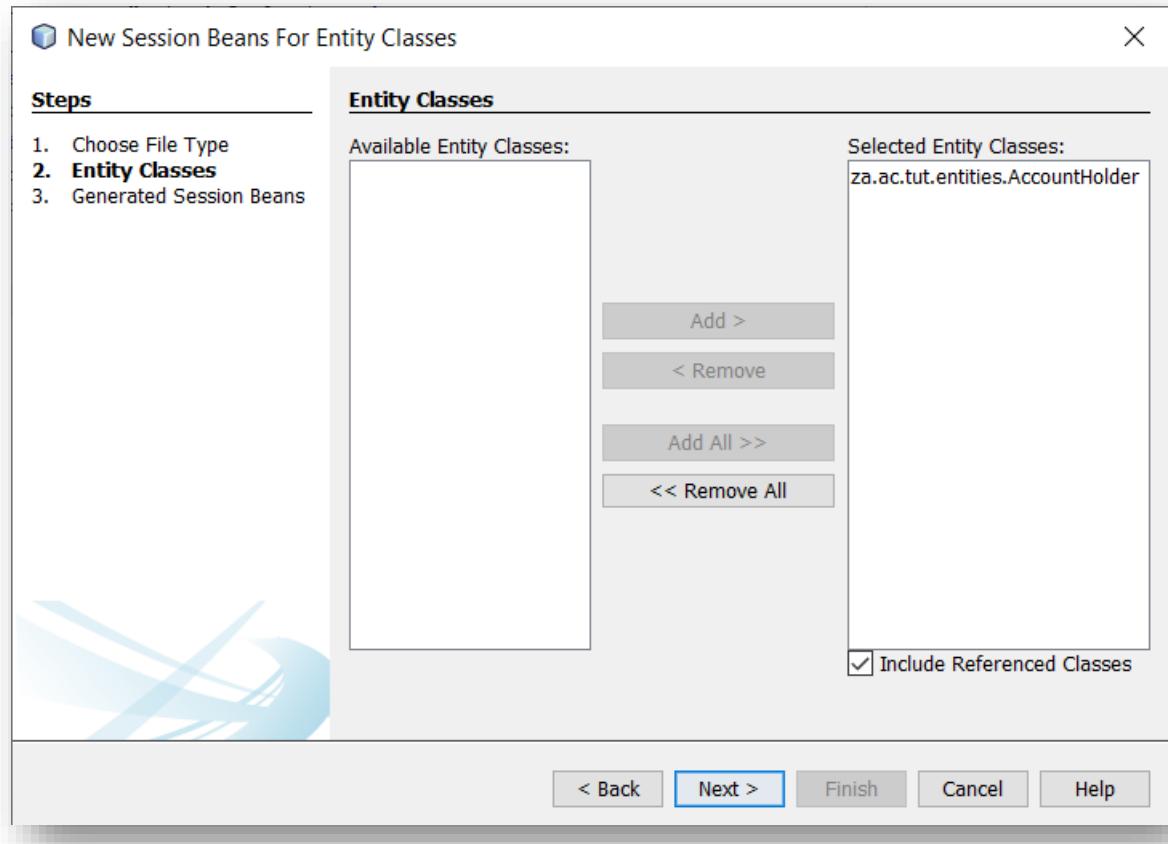
- ✓ Under **Categories**, select **Enterprise JavaBeans**. Under **File Types** select **Session Beans for Entity Classes**.



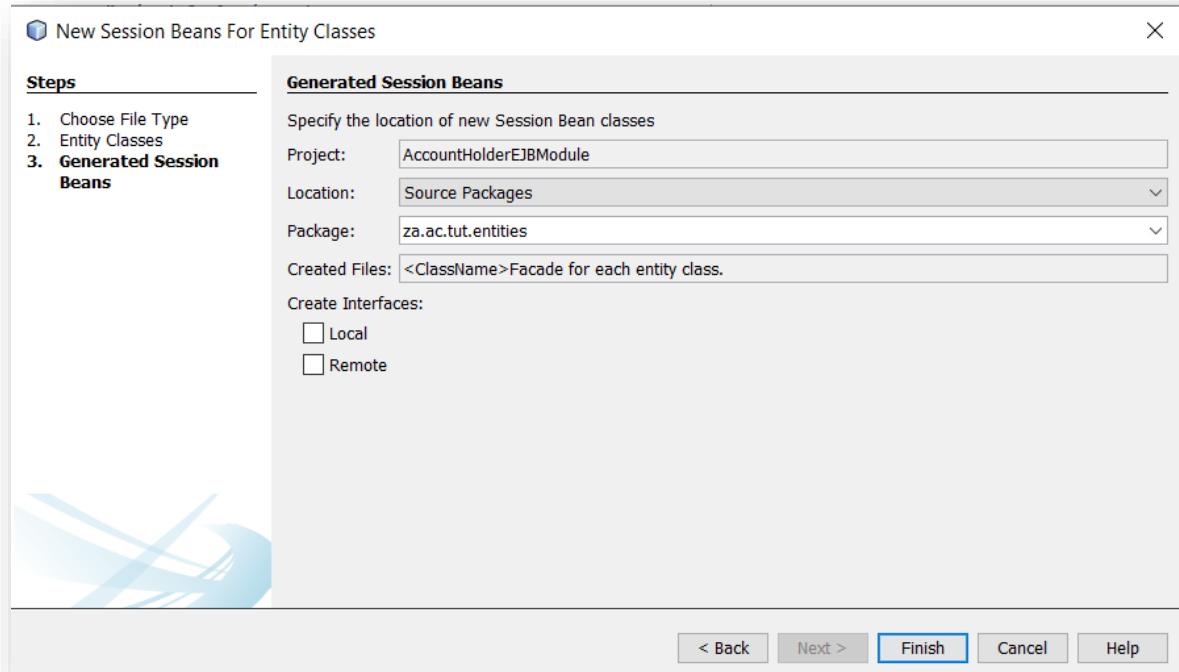
- ✓ Click **Next**.



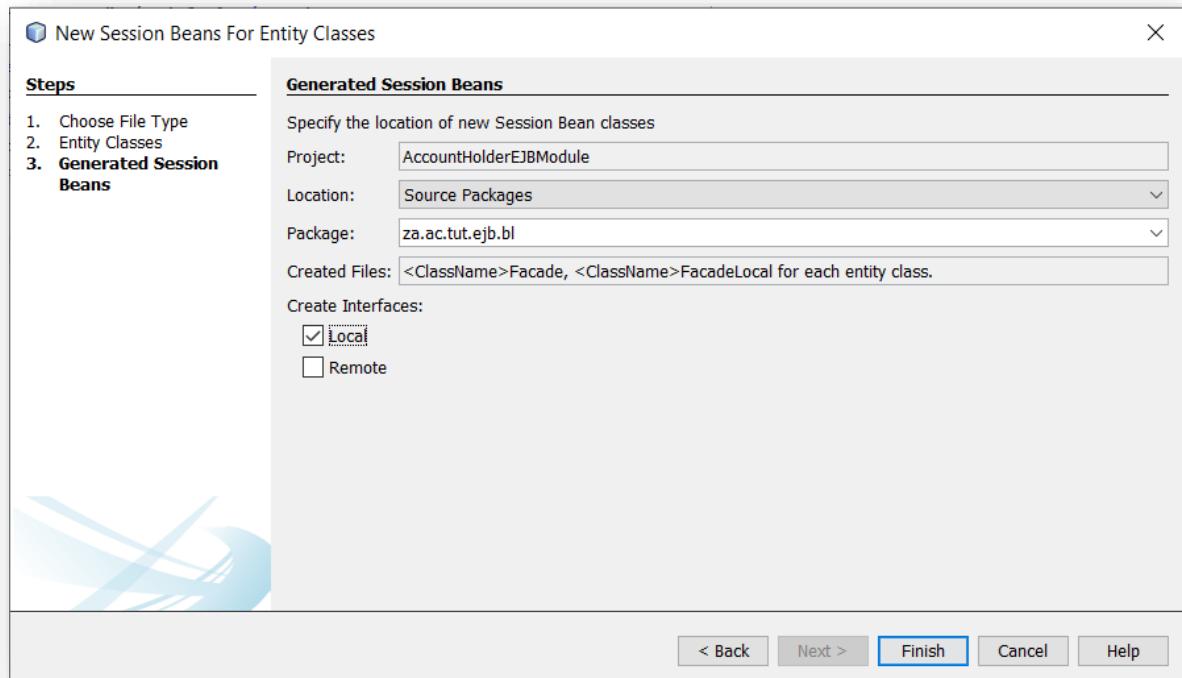
- ✓ Select **za.ac.tut.entities.AccountHolder** under **Available Entity Classes** and click **Add**.



- ✓ Click **Next**.



- ✓ Modify the package name to **za.ac.tut.ejb.bl**, where **bl** stands for **business logic**. Also select **Local** interface.



- ✓ Click **Finish**. View created files.
 - **AccountHolderFacade**.

```

1 package za.ac.tut.ejb.bl;
2
3 import javax.ejb.Stateless;
4 import javax.persistence.EntityManager;
5 import javax.persistence.PersistenceContext;
6 import za.ac.tut.entities.AccountHolder;
7
8 /**
9 * @author MemaniV
10 */
11 @Stateless
12 public class AccountHolderFacade extends AbstractFacade<AccountHolder> implements AccountHolderFacadeLocal {
13
14     @PersistenceContext(unitName = "AccountHolderEJBModulePU")
15     private EntityManager em;
16
17     @Override
18     protected EntityManager getEntityManager() {
19         return em;
20     }
21
22     public AccountHolderFacade() {
23         super(AccountHolder.class);
24     }
25
26 }
27

```

▪ AbstractFacade.

The screenshot shows a Java code editor with the file `AbstractFacade.java` open. The code defines an abstract class `AbstractFacade` that interacts with a database via an `EntityManager`. It includes methods for creating, editing, and removing entities, as well as finding entities by ID or range, and counting them.

```
1 package za.ac.tut.ejb.bl;
2
3 import java.util.List;
4 import javax.persistence.EntityManager;
5
6 /**
7 *
8 * @author MemaniV
9 */
10 public abstract class AbstractFacade<T> {
11
12     private Class<T> entityClass;
13
14     public AbstractFacade(Class<T> entityClass) {
15         this.entityClass = entityClass;
16     }
17
18     protected abstract EntityManager getEntityManager();
19
20     public void create(T entity) {
21         getEntityManager().persist(entity);
22     }
23
24     public void edit(T entity) {
25         getEntityManager().merge(entity);
26     }
27
28     public void remove(T entity) {
29         getEntityManager().remove(getEntityManager().merge(entity));
30     }
31 }
```

The screenshot continues the `AbstractFacade.java` code from the previous part. It adds methods for finding entities by ID, finding all entities, finding entities within a specific range, and counting entities.

```
32     public T find(Object id) {
33         return getEntityManager().find(entityClass, id);
34     }
35
36     public List<T> findAll() {
37         javax.persistence.criteria.CriteriaQuery cq =
38             getEntityManager().getCriteriaBuilder().createQuery();
39         cq.select(cq.from(entityClass));
40         return getEntityManager().createQuery(cq).getResultList();
41     }
42
43     public List<T> findRange(int[] range) {
44         javax.persistence.criteria.CriteriaQuery cq =
45             getEntityManager().getCriteriaBuilder().createQuery();
46         cq.select(cq.from(entityClass));
47         javax.persistence.Query q = getEntityManager().createQuery(cq);
48         q.setMaxResults(range[1] - range[0] + 1);
49         q.setFirstResult(range[0]);
50         return q.getResultList();
51     }
52
53     public int count() {
54         javax.persistence.criteria.CriteriaQuery cq =
55             getEntityManager().getCriteriaBuilder().createQuery();
56         javax.persistence.criteria.Root<T> rt = cq.from(entityClass);
57         cq.select(getEntityManager().getCriteriaBuilder().count(rt));
58         javax.persistence.Query q = getEntityManager().createQuery(cq);
59         return ((Long) q.getSingleResult()).intValue();
60     }
61
62 }
```

- **AccountHolderFacadeLocal.**

The screenshot shows a Java code editor window with the title bar "AccountHolderFacadeLocal.java". The menu bar includes "Source", "History", and various tool icons. The code itself is a Java interface:

```
1 package za.ac.tut.ejb.bl;
2
3 import java.util.List;
4 import javax.ejb.Local;
5 import za.ac.tut.entities.AccountHolder;
6
7 /**
8 * ...
9 * @author MemaniV
10 */
11 @Local
12 public interface AccountHolderFacadeLocal {
13
14     void create(AccountHolder accountHolder);
15
16     void edit(AccountHolder accountHolder);
17
18     void remove(AccountHolder accountHolder);
19
20     AccountHolder find(Object id);
21
22     List<AccountHolder> findAll();
23
24     List<AccountHolder> findRange(int[] range);
25
26     int count();
27
28 }
```

- ✓ Remove all the other interface methods except **create**, **find** and **findAll**.

The screenshot shows the same Java code editor window after modifications. The code now contains only three methods: create, find, and findAll.

```
1 package za.ac.tut.ejb.bl;
2
3 import java.util.List;
4 import javax.ejb.Local;
5 import za.ac.tut.entities.AccountHolder;
6
7 /**
8 * ...
9 * @author MemaniV
10 */
11 @Local
12 public interface AccountHolderFacadeLocal {
13
14     void create(AccountHolder accountHolder);
15
16     AccountHolder find(Object id);
17
18     List<AccountHolder> findAll();
19
20 }
21
```

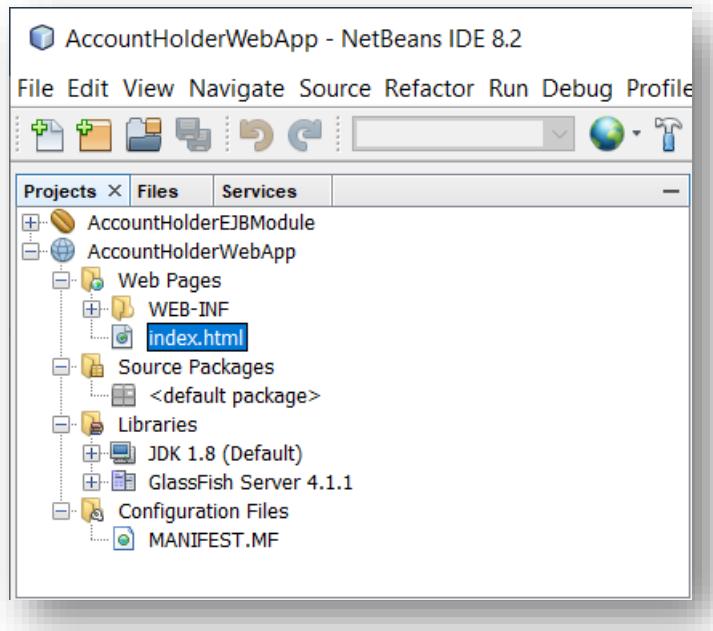
✓ Clean and build the EJB project.

The screenshot shows the NetBeans IDE interface. The top part is the Output window, which displays the build logs for the AccountHolderEJBModule. The logs show several notes about optional files not being found for eclipselink-orm.xml and orm.xml, followed by a warning about recompiling with -Xlint:unchecked. It then shows the creation of a dist directory and the building of a jar file named AccountHolderEJBModule.jar. The message "BUILD SUCCESSFUL (total time: 1 second)" is at the bottom. The bottom part of the screenshot shows the Project Explorer window, which lists the project structure. The dist folder contains the generated jar file and its contents, including META-INF/MANIFEST.MF, persistence.xml, za.ac.tut.ejb.bl (containing AbstractFacade.class, AccountHolderFacade.class, and AccountHolderFacadeLocal.class), and za.ac.tut.entities (containing AccountHolder.class and AccountHolder_.class). Other project folders like build, nbproject, src, and test are also visible.

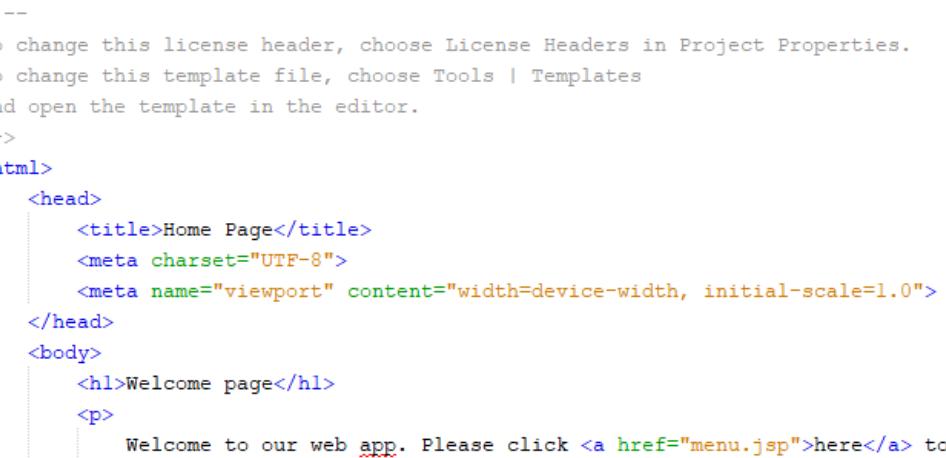
```
Java DB Database Process × GlassFish Server 4.1.1 × AccountHolderEJBModule (clean,dist) ×
Note: Optional file was not found: META-INF/eclipselink-orm.xml continuing with generation.
Note: Optional file was not found: META-INF/orm.xml continuing with generation.
Note: Optional file was not found: META-INF/eclipselink-orm.xml continuing with generation.
Note: C:\Users\memaniv\Documents\NetBeansProjects\AccountHolderEJBModule\src\java\za\ac\tut\ejb\bl\AbstractFacade.java
Note: Recompile with -Xlint:unchecked for details.
1 warning
compile:
library-inclusion-in-archive:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\AccountHolderEJBModule\dist
Building jar: C:\Users\memaniv\Documents\NetBeansProjects\AccountHolderEJBModule\dist\AccountHolderEJBModule.jar
dist:
BUILD SUCCESSFUL (total time: 1 second)
```

Part D – Create a web client project.

Create a web client project called **AccountHolderWebApp**.



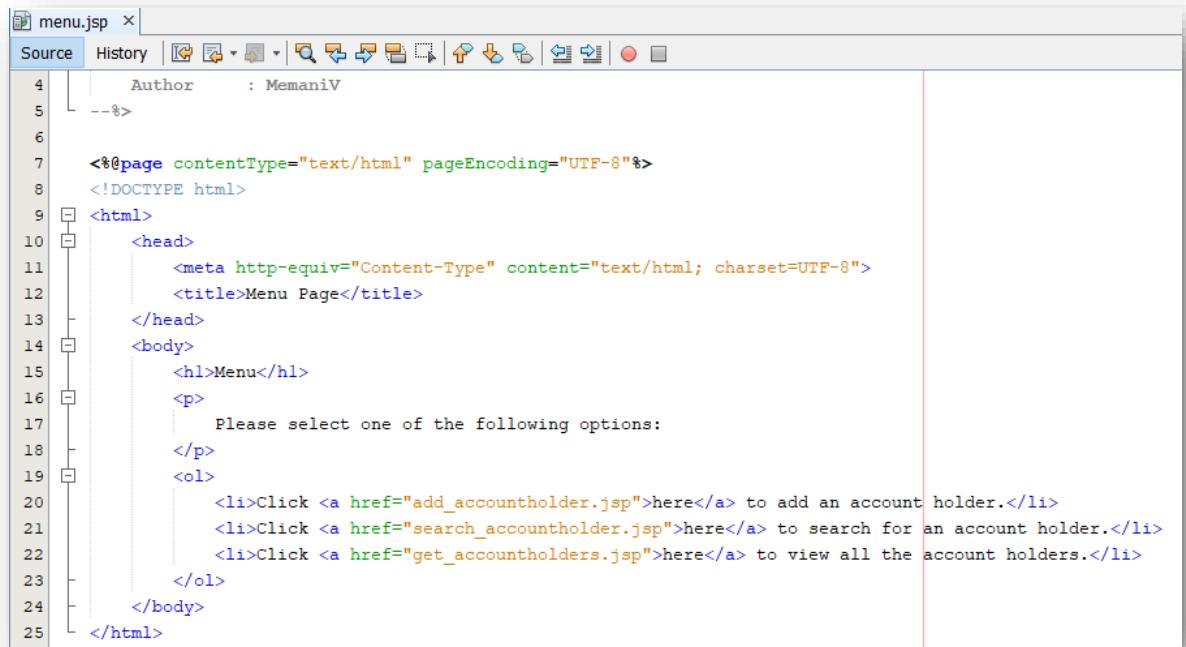
Edit the **index.html** page.



The screenshot shows a browser window with the title "index.html". The "Source" tab is selected, displaying the raw HTML code. The code includes a license header, an HTML structure with a head section containing meta tags for charset and viewport, and a body section with a heading and a paragraph. A red box highlights the word "here" in the paragraph text.

```
<!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 page</h1>
        <p>
            Welcome to our web app. Please click <a href="menu.jsp">here</a> to start.
        </p>
    </body>
</html>
```

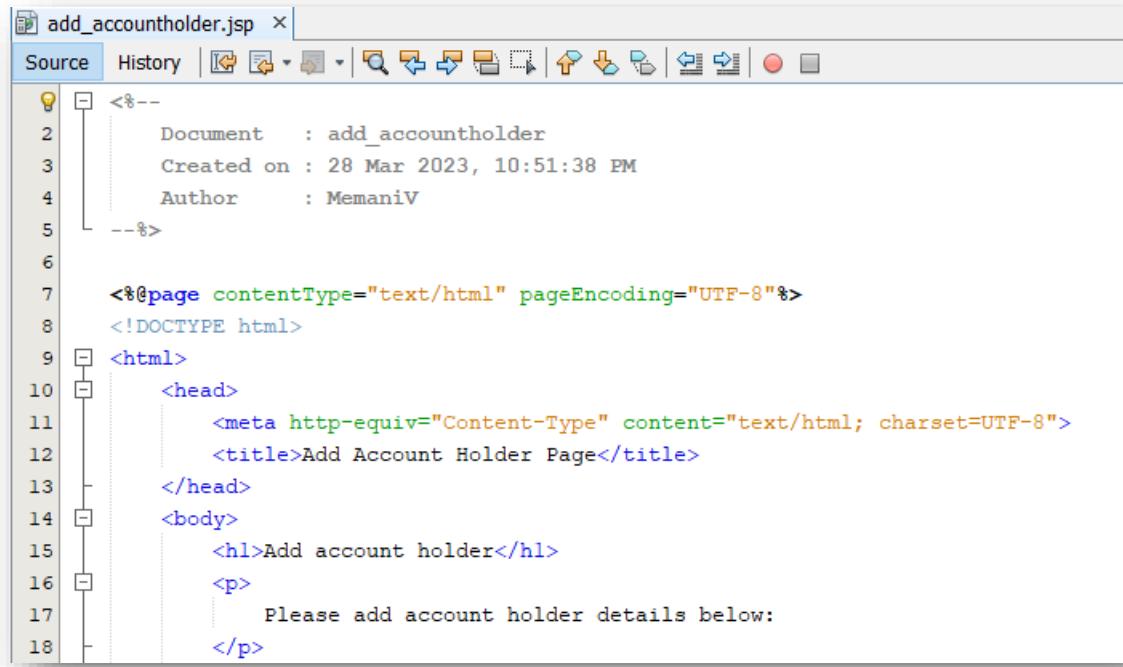
Create the menu.jsp file.



The screenshot shows the code editor of a Java IDE with the file "menu.jsp" open. The code is a JSP page that displays a menu. It includes meta tags, a title, and a body section with an h1 header and a list of three options for adding, searching, or viewing account holders.

```
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="add_acountholder.jsp">here</a> to add an account holder.</li>
21         <li>Click <a href="search_acountholder.jsp">here</a> to search for an account holder.</li>
22         <li>Click <a href="get_acountholders.jsp">here</a> to view all the account holders.</li>
23     </ol>
24 </body>
25 </html>
```

Create the add_acountholder.jsp file.

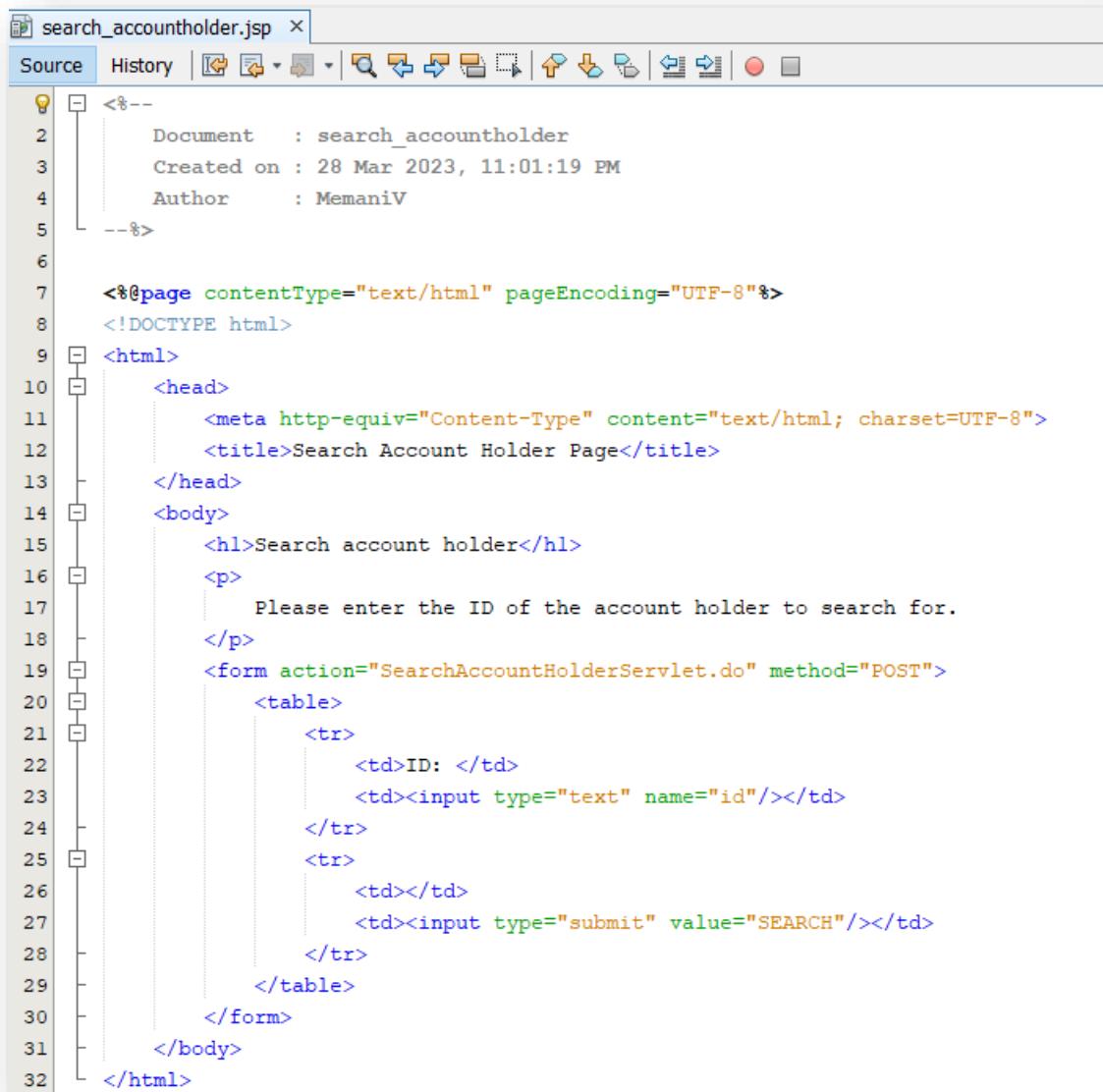


The screenshot shows the code editor of a Java IDE with the file "add_acountholder.jsp" open. The code is a JSP page for adding an account holder. It includes meta tags, a title, and a body section with an h1 header and a p tag asking for account holder details.

```
1 <%--
2     Document : add_acountholder
3     Created on : 28 Mar 2023, 10:51:38 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>Add Account Holder Page</title>
13 </head>
14 <body>
15     <h1>Add account holder</h1>
16     <p>
17         Please add account holder details below:
18     </p>
```

```
19 | 19 | <form action="AddAccountHolderServlet.do" method="POST">
20 | 20 |   <table>
21 | 21 |     <tr>
22 | 22 |       <td>ID: </td>
23 | 23 |       <td><input type="text" name="id"/></td>
24 | 24 |     </tr>
25 | 25 |     <tr>
26 | 26 |       <td>Full name: </td>
27 | 27 |       <td><input type="text" name="fullname"/></td>
28 | 28 |     </tr>
29 | 29 |     <tr>
30 | 30 |       <td>Street: </td>
31 | 31 |       <td><input type="text" name="street"/></td>
32 | 32 |     </tr>
33 | 33 |     <tr>
34 | 34 |       <td>City: </td>
35 | 35 |       <td><input type="text" name="city"/></td>
36 | 36 |     </tr>
37 | 37 |     <tr>
38 | 38 |       <td>Code: </td>
39 | 39 |       <td><input type="text" name="code"/></td>
40 | 40 |     </tr>
41 | 41 |     <tr>
42 | 42 |       <td>Cell number: </td>
43 | 43 |       <td><input type="text" name="cellNo"/></td>
44 | 44 |     </tr>
45 | 45 |     <tr>
46 | 46 |       <td>Balance: R</td>
47 | 47 |       <td><input type="text" name="balance"/></td>
48 | 48 |     </tr>
49 | 49 |     <tr>
50 | 50 |       <td></td>
51 | 51 |       <td><input type="submit" value="ADD ACCOUNT HOLDER"/></td>
52 | 52 |     </tr>
53 | 53 |   </table>
54 | 54 | </form>
55 | 55 | </body>
56 | 56 | </html>
```

Create the **search_acountholder.jsp** file.

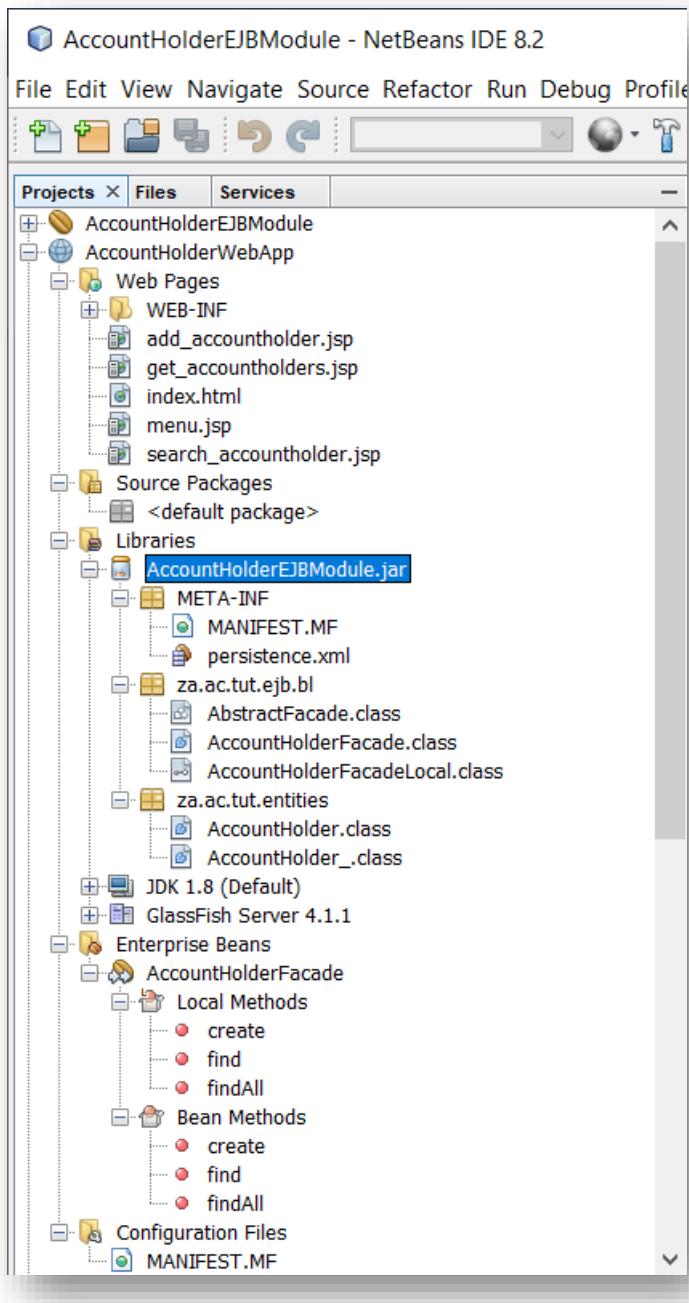


```
<%--  
1 Document : search_acountholder  
2 Created on : 28 Mar 2023, 11:01:19 PM  
3 Author : MemaniV  
4 --%>  
5  
6  
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>  
8 <!DOCTYPE html>  
9 <html>  
10 <head>  
11     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
12     <title>Search Account Holder Page</title>  
13 </head>  
14 <body>  
15     <h1>Search account holder</h1>  
16     <p>  
17         Please enter the ID of the account holder to search for.  
18     </p>  
19     <form action="SearchAccountHolderServlet.do" method="POST">  
20         <table>  
21             <tr>  
22                 <td>ID: </td>  
23                 <td><input type="text" name="id"/></td>  
24             </tr>  
25             <tr>  
26                 <td></td>  
27                 <td><input type="submit" value="SEARCH"/></td>  
28             </tr>  
29         </table>  
30     </form>  
31 </body>  
32 </html>
```

Create the get_acountholders.jsp file.

```
<%--  
1 Document : search_acountholder  
2 Created on : 28 Mar 2023, 11:01:19 PM  
3 Author : MemaniV  
4 --%>  
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>Search Account Holder Page</title>  
13 </head>  
14 <body>  
15     <h1>Search account holder</h1>  
16     <p>  
17         Please enter the ID of the account holder to search for.  
18     </p>  
19     <form action="SearchAccountHolderServlet.do" method="POST">  
20         <table>  
21             <tr>  
22                 <td>ID: </td>  
23                 <td><input type="text" name="id"/></td>  
24             </tr>  
25             <tr>  
26                 <td></td>  
27                 <td><input type="submit" value="SEARCH"/></td>  
28             </tr>  
29         </table>  
30     </form>  
31 </body>  
32 </html>
```

Add the **AccountHolderEJBModule.jar** file to the library of **AccountHolderWebApp**.



Create the AddAccountHolderServlet.java file.

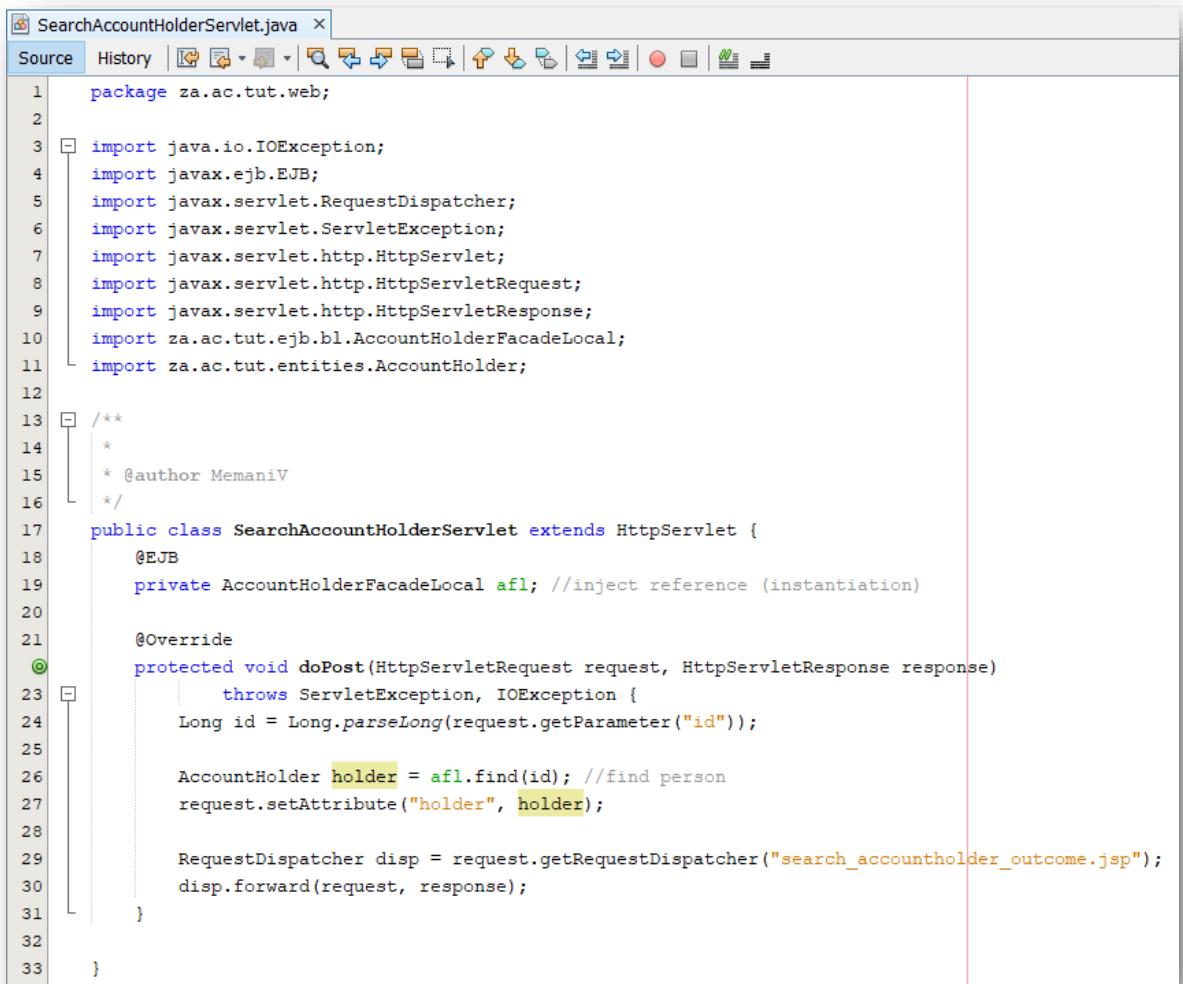
The screenshot shows the source code for the `AddAccountHolderServlet.java` file. The code is annotated with Javadoc comments and annotations like `@EJB` and `@Override`. It uses `RequestDispatcher` to forward the request to a JSP page. The code is part of a Java web application project.

```
1 package za.ac.tut.web;
2
3 import java.io.IOException;
4 import java.util.Date;
5 import javax.ejb.EJB;
6 import javax.servlet.RequestDispatcher;
7 import javax.servlet.ServletException;
8 import javax.servlet.http.HttpServlet;
9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11 import za.ac.tut.bl.AccountHolderFacadeLocal;
12 import za.ac.tut.entities.AccountHolder;
13
14 /**
15 * 
16 * @author MemaniV
17 */
18 public class AddAccountHolderServlet extends HttpServlet {
19     @EJB
20     private AccountHolderFacadeLocal afl; //inject reference (instantiation)
21     @Override
22     protected void doPost(HttpServletRequest request, HttpServletResponse response)
23             throws ServletException, IOException {
24         Long id = Long.parseLong(request.getParameter("id"));
25         String fullname = request.getParameter("fullname");
26         String street = request.getParameter("street");
27         String city = request.getParameter("city");
28         String code = request.getParameter("code");
29         String cellNo = request.getParameter("cellNo");
30         String emailAddress = request.getParameter("emailAddress");
31
32         AccountHolder holder = createAccountHolder(id, fullname, street, city, code, cellNo, emailAddress); //create account holder
33         afl.create(holder); //persist account holder
34         request.setAttribute("fullname", fullname);
35
36         RequestDispatcher disp = request.getRequestDispatcher("add_person_outcome.jsp");
37         disp.forward(request, response);
38     }
39 }
```

The screenshot shows the implementation of the `createAccountHolder` method. This is a private method that creates a new `AccountHolder` object and sets its properties based on the input parameters.

```
40     private AccountHolder createAccountHolder(Long id, String fullname, String street, String city, String code, String cellNo, String emailAddress) {
41         AccountHolder holder = new AccountHolder();
42         holder.setId(id);
43         holder.setFullName(fullname);
44         holder.setStreet(street);
45         holder.setCity(city);
46         holder.setCode(code);
47         holder.setCellNo(cellNo);
48         holder.setEmailAddress(emailAddress);
49         holder.setCreationDate(new Date());
50         return holder;
51     }
52 }
```

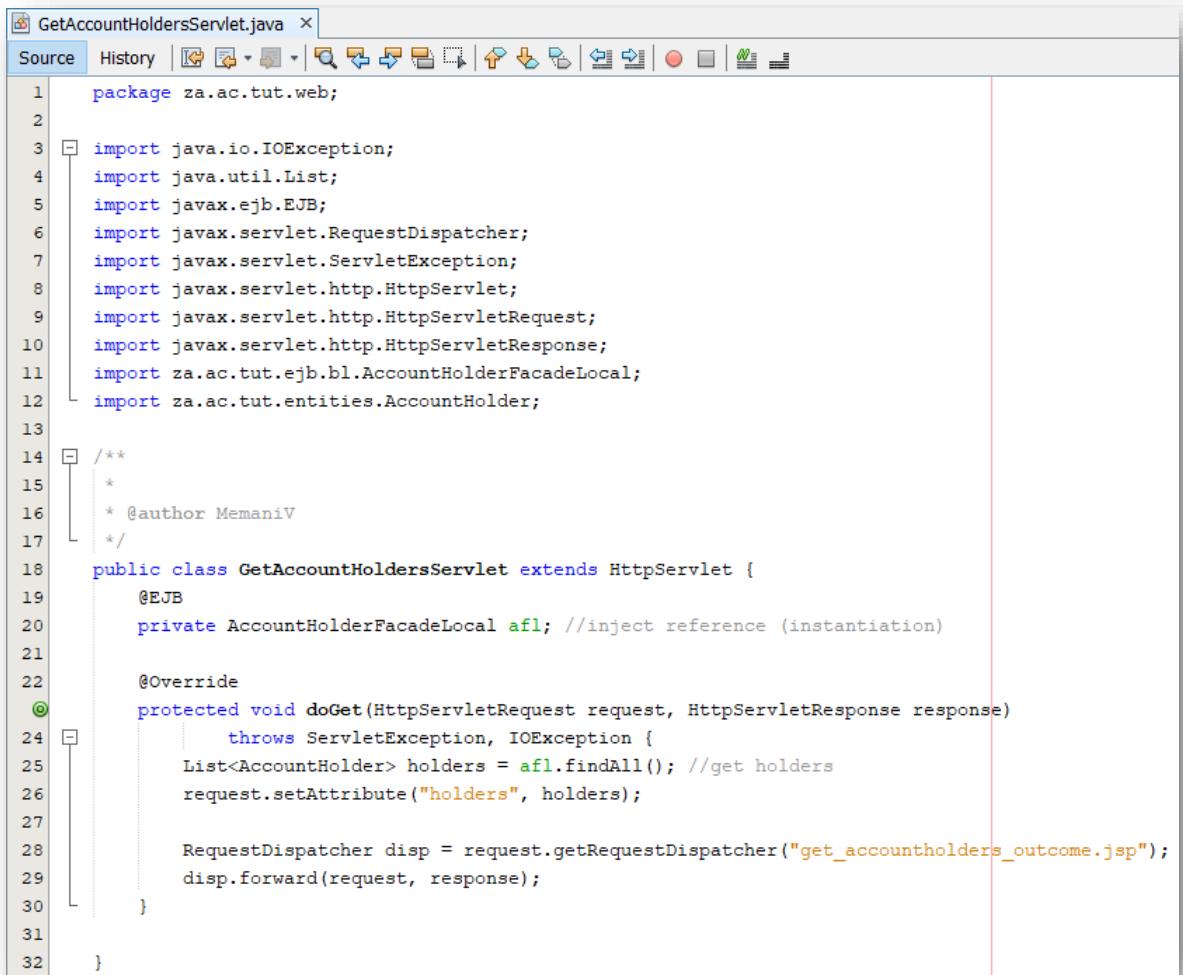
Create the **SearchAccountHolderServlet.java** file.



The screenshot shows a Java code editor window titled "SearchAccountHolderServlet.java". The code implements a HttpServlet that injects an AccountHolderFacadeLocal reference and performs a search based on an ID parameter.

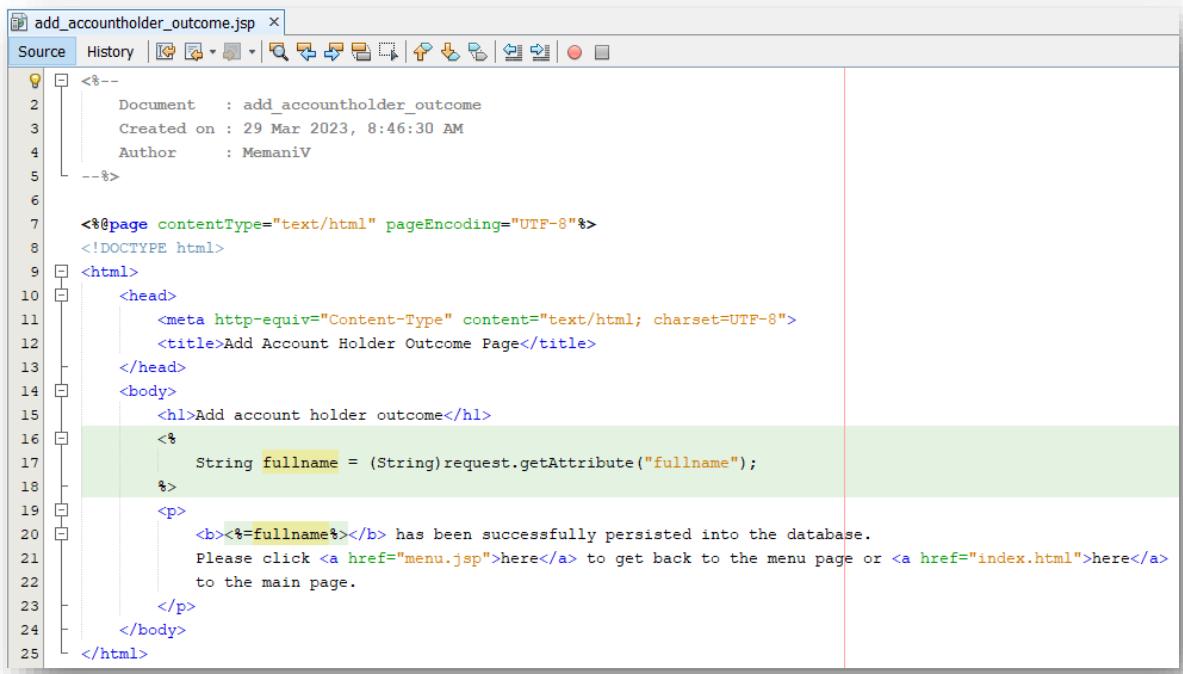
```
1 package za.ac.tut.web;
2
3 import java.io.IOException;
4 import javax.ejb.EJB;
5 import javax.servlet.RequestDispatcher;
6 import javax.servlet.ServletException;
7 import javax.servlet.http.HttpServlet;
8 import javax.servlet.http.HttpServletRequest;
9 import javax.servlet.http.HttpServletResponse;
10 import za.ac.tut.ejb.bl.AccountHolderFacadeLocal;
11 import za.ac.tut.entities.AccountHolder;
12
13 /**
14 * @author MemaniV
15 */
16 public class SearchAccountHolderServlet extends HttpServlet {
17     @EJB
18     private AccountHolderFacadeLocal afl; //inject reference (instantiation)
19
20     @Override
21     protected void doPost(HttpServletRequest request, HttpServletResponse response)
22             throws ServletException, IOException {
23         Long id = Long.parseLong(request.getParameter("id"));
24
25         AccountHolder holder = afl.find(id); //find person
26         request.setAttribute("holder", holder);
27
28         RequestDispatcher disp = request.getRequestDispatcher("search_accountholder_outcome.jsp");
29         disp.forward(request, response);
30     }
31 }
32
33 }
```

Create the **GetAccountHoldersServlet.java** file.



```
1 package za.ac.tut.web;
2
3 import java.io.IOException;
4 import java.util.List;
5 import javax.ejb.EJB;
6 import javax.servlet.RequestDispatcher;
7 import javax.servlet.ServletException;
8 import javax.servlet.http.HttpServlet;
9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11 import za.ac.tut.ejb.bl.AccountHolderFacadeLocal;
12 import za.ac.tut.entities.AccountHolder;
13
14 /**
15 *
16 * @author MemaniV
17 */
18 public class GetAccountHoldersServlet extends HttpServlet {
19     @EJB
20     private AccountHolderFacadeLocal afl; //inject reference (instantiation)
21
22     @Override
23     protected void doGet(HttpServletRequest request, HttpServletResponse response)
24             throws ServletException, IOException {
25         List<AccountHolder> holders = afl.findAll(); //get holders
26         request.setAttribute("holders", holders);
27
28         RequestDispatcher disp = request.getRequestDispatcher("get_accountholders_outcome.jsp");
29         disp.forward(request, response);
30     }
31
32 }
```

Create the **add_accountholder_outcome.jsp** file.



```
<%-->
1 Document : add_accountholder_outcome
2 Created on : 29 Mar 2023, 8:46:30 AM
3 Author : MemaniV
4 --%>
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>Add Account Holder Outcome Page</title>
13    </head>
14    <body>
15        <h1>Add account holder outcome</h1>
16        <%
17            String fullname = (String)request.getAttribute("fullname");
18        %>
19        <p>
20            <b><%=fullname%></b> has been successfully persisted into the database.
21            Please click <a href="menu.jsp">here</a> to get back to the menu page or <a href="index.html">here</a>
22            to the main page.
23        </p>
24    </body>
25 </html>
```

Create the search_acountholder_outcome.jsp file.

```
<%--  
1 Document      : search_acountholder_outcome  
2 Created on   : 29 Mar 2023, 8:49:05 AM  
3 Author        : MemaniV  
4 --%>  
5  
6  
7 <%@page import="za.ac.tut.entities.AccountHolder"%>  
8 <%@page contentType="text/html" pageEncoding="UTF-8"%>  
9 <!DOCTYPE html>  
10 <html>  
11 <head>  
12     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
13     <title>Search Account Holder Outcome Page</title>  
14 </head>  
15 <body>  
16     <h1>Search account holder outcome</h1>  
17     <%  
18         AccountHolder holder = (AccountHolder)request.getAttribute("holder");  
19     %>  
20     <p>  
21         Below are the details of the account holder.  
22     </p>
```

```
23 <table>  
24     <tr>  
25         <td><b>ID:</b> </td>  
26         <td><%=holder.getId()%></td>  
27     </tr>  
28     <tr>  
29         <td><b>Full name:</b> </td>  
30         <td><%=holder.getFullName()%></td>  
31     </tr>  
32     <tr>  
33         <td><b>Street:</b> </td>  
34         <td><%=holder.getStreet()%></td>  
35     </tr>  
36     <tr>  
37         <td><b>City:</b> </td>  
38         <td><%=holder.getCity()%></td>  
39     </tr>  
40     <tr>  
41         <td><a href="#"><b>Code:</b></a> </td>  
42         <td><%=holder.getCode()%></td>  
43     </tr>  
44     <tr>  
45         <td><b>Cell number:</b> </td>  
46         <td><%=holder.getCellNo()%></td>  
47     </tr>  
48     <tr>  
49         <td><b>Email address:</b> </td>  
50         <td><%=holder.getEmailAddress()%></td>  
51     </tr>  
52 </table>  
53 <p>  
54     Please click <a href="menu.jsp">here</a> to get back to the menu  
55     page or <a href="index.html">here</a> to the main page.  
56 </p>  
57 </body>  
58 </html>
```

Create the **search_acountholder_outcome.jsp** file.

The screenshot shows a Java IDE interface with a code editor. The title bar says "get_acountholders_outcome.jsp". The code is a JSP page with the following content:

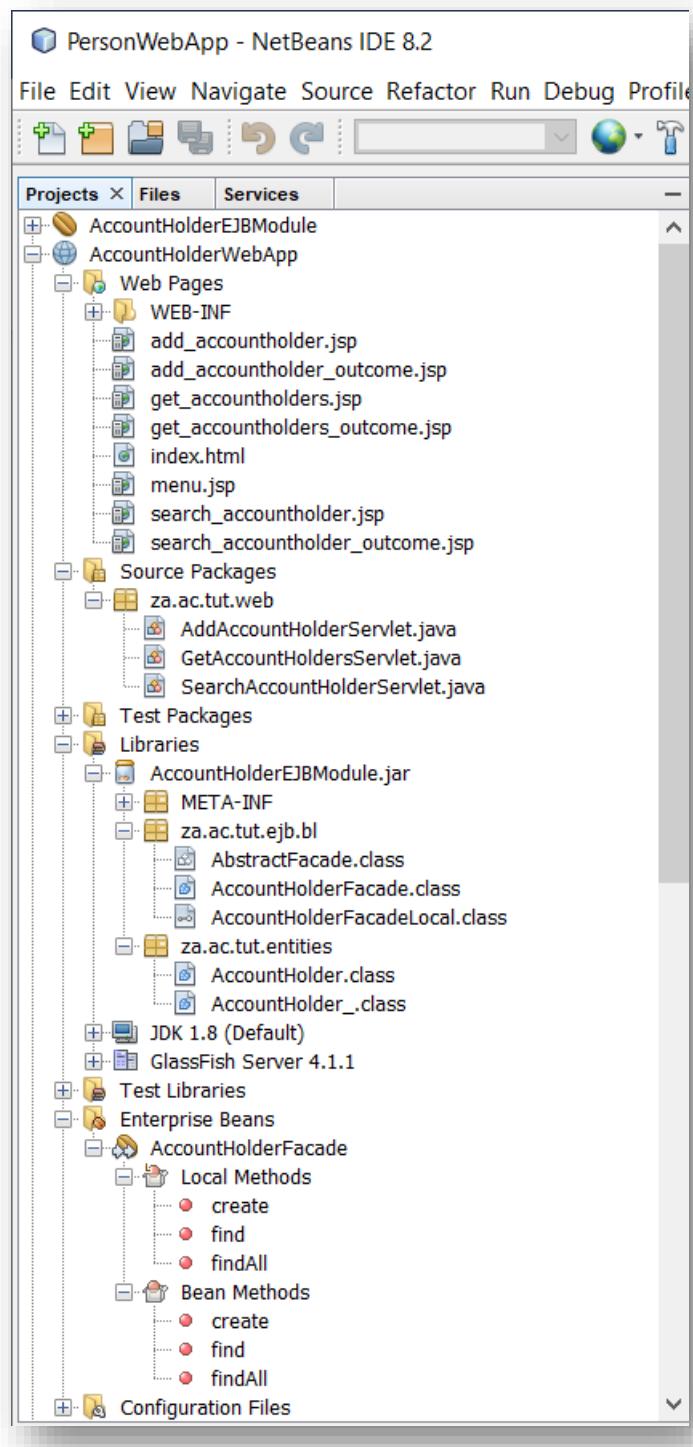
```
<%--  
Document : get_acountholders_outcome  
Created on : 29 Mar 2023, 9:02:17 AM  
Author : MemaniV  
--%>  
  
<%@page import="za.ac.tut.entities.AccountHolder"%>  
<%@page import="java.util.List"%>  
<%@page contentType="text/html" pageEncoding="UTF-8"%>  
<!DOCTYPE html>  
<html>  
<head>  
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
    <title>Get Account Holders Outcome Page</title>  
</head>  
<body>  
    <h1>Get account holders outcome</h1>  
    <%  
        List<AccountHolder> list = (List<AccountHolder>) request.getAttribute("list");  
    %>  
    <p>  
        Below are the account holders retrieved from the database.  
    </p>
```

The screenshot shows a Java IDE interface with a code editor, continuing from the previous code block. The code is a JSP page with the following content:

```
<table>  
    <%  
        for(int i = 0; i < list.size(); i++){  
            AccountHolder p = list.get(i);  
            Long id = p.getId();  
            String fullName = p.getFullName();  
            String street = p.getStreet();  
            String city = p.getCity();  
            String code = p.getCode();  
            String cellNo = p.getCellNo();  
            String emailAddress = p.getEmailAddress();  
        %>  
        <tr>  
            <td><b>ID:</b></td>  
            <td><%=id%></td>  
        </tr>  
        <tr>  
            <td><b>Full name:</b></td>  
            <td><%=fullName%></td>  
        </tr>  
        <tr>  
            <td><b>Street:</b></td>  
            <td><%=street%></td>  
        </tr>  
        <tr>  
            <td><b>City:</b></td>  
            <td><%=city%></td>  
        </tr>  
        <tr>  
            <td><b>Code:</b></td>  
            <td><%=code%></td>  
        </tr>  
        <tr>  
            <td><b>Cell number:</b></td>  
            <td><%=cellNo%></td>  
        </tr>
```

```
60 |     <tr>
61 |       <td><b>Email address:</b> </td>
62 |       <td><%=emailAddress%></td>
63 |     </tr>
64 |     <tr></tr>
65 |     <%
66 |       |
67 |       %
68 |     </table>
69 |   <p>
70 |     Please click <a href="menu.jsp">here</a> to get back to the menu
71 |     page or <a href="index.html">here</a> to the main page.
72 |   </p>
73 |   </body>
74 | </html>
```

View the complete project structure of **AccountHolderWebApp**.



Compile the project.

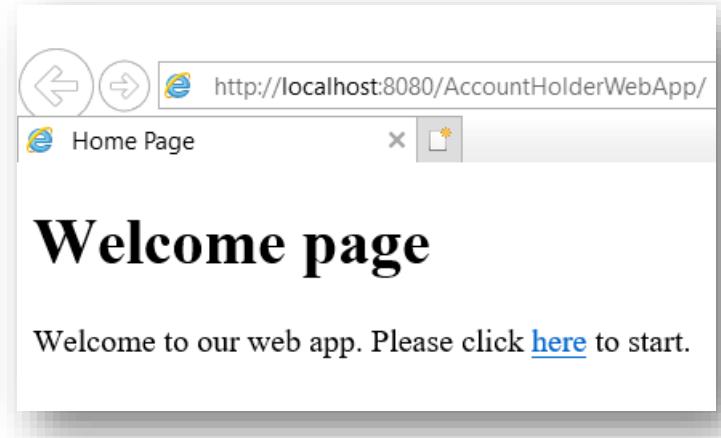
```
Output - AccountHolderWebApp (clean,dist) ×
check-clean:
clean:
init:
deps-module-jar:
deps-ear-jar:
deps-jar:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\AccountHolderWebApp\build\web\WEB-INF\classes
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\AccountHolderWebApp\build\web\META-INF
Copying 1 file to C:\Users\memaniv\Documents\NetBeansProjects\AccountHolderWebApp\build\web\META-INF
Copying 9 files to C:\Users\memaniv\Documents\NetBeansProjects\AccountHolderWebApp\build\web\WEB-INF\lib
library-inclusion-in-archive:
Copying 1 file to C:\Users\memaniv\Documents\NetBeansProjects\AccountHolderWebApp\build\web\WEB-INF\lib
library-inclusion-in-manifest:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\AccountHolderWebApp\build\empty
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\AccountHolderWebApp\build\generated-sources\ap-source-output
Compiling 3 source files to C:\Users\memaniv\Documents\NetBeansProjects\AccountHolderWebApp\build\web\WEB-INF\classes
compile:
compile-jsp:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\AccountHolderWebApp\dist
Building jar: C:\Users\memaniv\Documents\NetBeansProjects\AccountHolderWebApp\dist\AccountHolderWebApp.war
do-dist:
dist:
BUILD SUCCESSFUL (total time: 0 seconds)
```

Deploy the project.

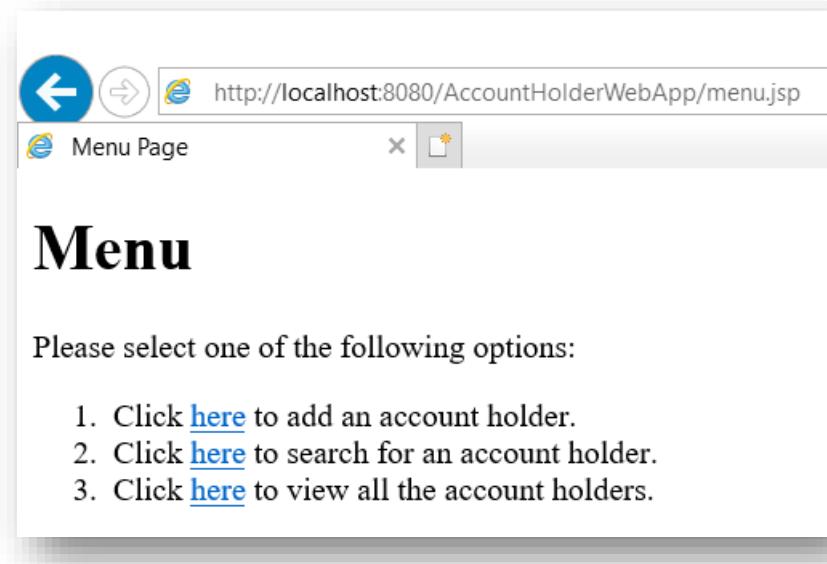
```
Output ×
AccountHolderWebApp (run-deploy) × | Java DB Database Process × | GlassFish Server 4.1.1 ×
Info: /file:/C:/Users/memaniv/Documents/NetBeansProjects/AccountHolderWebApp/build/web/
Info: visiting unvisited references
Info: EclipseLink, version: Eclipse Persistence Services - 2.6.1.v20150605-31e8258
Info: /file:/C:/Users/memaniv/Documents/NetBeansProjects/AccountHolderWebApp/build/web/
Info: Portable JNDI names for EJB AccountHolderFacade: [java:global/AccountHolderWebApp/
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] org.glassfish.sse.impl.Server
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] public org.glassfish.jms.inje
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] private org.glassfish.jersey.
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] org.glassfish.sse.impl.Server
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] private org.glassfish.jersey.
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] public org.glassfish.jms.inje
Info: Loading application [AccountHolderWebApp] at [/AccountHolderWebApp]
Info: AccountHolderWebApp was successfully deployed in 434 milliseconds.
```

Part E – Run the web application.

Launch the application.



Click on the link.



1. Click [here](#) to add an account holder.
2. Click [here](#) to search for an account holder.
3. Click [here](#) to view all the account holders.

Add an account holder.

- ✓ Click on the first link.

The screenshot shows a web browser window with the URL http://localhost:8080/AccountHolderWebApp/add_accountholder.jsp. The title bar says "Add Account Holder Page". The main content area has a large heading "Add account holder". Below it, a message says "Please add account holder details below:". There are seven input fields for "ID", "Full name", "Street", "City", "Code", "Cell number", and "Email address". A large "ADD ACCOUNT HOLDER" button is at the bottom.

- ✓ Fill in the form.

The screenshot shows a web browser window with the same URL and title as the previous one. The main content area has a large heading "Add account holder". Below it, a message says "Please add account holder details below:". The input fields now contain the following data:
ID: 222
Full name: Thato Ranamane
Street: 13 Biko
City: Mabopane
Code: 0142
Cell number: 089
Email address: thato@gmail.com
A large "ADD ACCOUNT HOLDER" button is at the bottom.

- ✓ Click on the add button.

Add account holder outcome

Thato Ranamane has been successfully persisted into the database. Please click [here](#) to get back to the menu page or [here](#) to the main page.

- ✓ Go back to the menu page.

Menu

Please select one of the following options:

1. Click [here](#) to add an account holder.
2. Click [here](#) to search for an account holder.
3. Click [here](#) to view all the account holders.

- ✓ Add four more account holders into the database. View the records in the database.

#	ID	CREATIONDATE	FULLNAME
1	111	2023-03-29 09:22:39.947	Vuyisile Memani
2	222	2023-03-29 09:27:38.95	Thato Ranamane
3	333	2023-03-29 09:29:48.71	Glogza Nthane
4	444	2023-03-29 09:30:38.989	Onkgopotse Tiro
5	555	2023-03-29 09:31:38.745	Beyers Naude

Connection: jdbc:derby://localhost:1527/AccountHoldersDB [app on APP]

```

1 | SELECT * FROM APP.ADDRESS_TBL FETCH FIRST 100 ROWS ONLY;
2 |

```

SELECT * FROM APP.ADDRESS... ×

#	ID	CITY	CODE	STREET
1		111 Garankuwa	0208	12 Maditse
2		222 Mabopane	0142	13 Biko
3		333 Atteridgeville	0154	14 Nkomo
4		444 Mankweng	031	16 Turf
5		555 Pretoria	0100	15 Mandela

Connection: jdbc:derby://localhost:1527/AccountHoldersDB [app on APP]

```

1 | SELECT * FROM APP.CONTACTS_TBL FETCH FIRST 100 ROWS ONLY;
2 |

```

SELECT * FROM APP.CONTACT... ×

#	ID	CELLNO	EMAILADDRESS
1		111 0111	me@gmail.com
2		222 089	thato@gmail.com
3		333 081	nthane@gmail.com
4		444 072	tiro@gmail.com
5		555 061	naude@gmail.com

Search for an account holder.

- ✓ Click on the second link.

The screenshot shows a web browser window with the following details:

- Address Bar:** http://localhost:8080/AccountHolderWebApp/search_accountholder.jsp
- Title Bar:** Search Account Holder Page
- Content Area:**

Search account holder

Please enter the ID of the account holder to search for.

ID:

- ✓ Enter ID number 111.

The screenshot shows a web browser window with the following details:

- Address bar: http://localhost:8080/AccountHolderWebApp/search_accountholder.jsp
- Title bar: Search Account Holder Page
- Content area:
 - Search account holder**
 - Please enter the ID of the account holder to search for.
 - ID:
 - SEARCH**

- ✓ Click on the button.

The screenshot shows a web browser window with the following details:

- Address bar: http://localhost:8080/AccountHolderWebApp/SearchAccountHolderServlet.do
- Title bar: Search Account Holder Out...
- Content area:
 - Search account holder outcome**
 - Below are the details of the account holder.
 - ID:** 111
 - Full name:** Vuyisile Memani
 - Street:** 12 Maditse
 - City:** Garankuwa
 - Code:** 0208
 - Cell number:** 0111
 - Email address:** me@gmail.com
- Message at the bottom: Please click [here](#) to get back to the menu page or [here](#) to the main page.

Display all the account holders.

- ✓ Click on the third link.

The screenshot shows a web browser window with the URL http://localhost:8080/AccountHolderWebApp/get_accountholders.jsp. The title bar says "Get Account Holders Page". The main content area has a large heading "Get account holders" and a sub-instruction "Please click on the button below to get all account holders." Below this is a button labeled "GET AACOUNT HOLDERS".

- ✓ Click on the button.

The screenshot shows a web browser window with the URL <http://localhost:8080/AccountHolderWebApp/GetAccountHoldersServlet.do>. The title bar says "Get Account Holders Outco...". The main content area has a large heading "Get account holders outcome". Below this is a sub-instruction "Below are the account holders retrieved from the database." followed by a list of account holder details for three users:

ID:	111
Full name:	Vuyisile Memani
Street:	12 Maditse
City:	Garankuwa
Code:	0208
Cell number:	0111
Email address:	me@gmail.com
ID:	222
Full name:	Thato Ranamane
Street:	13 Biko
City:	Mabopane
Code:	0142
Cell number:	089
Email address:	thato@gmail.com
ID:	333
Full name:	Glogza Nthane
Street:	14 Nkomo
City:	Atteridgeville
Code:	0154
Cell number:	081
Email address:	nthane@gmail.com

ID: 444
Full name: Onkgopotse Tiro
Street: 16 Turf
City: Mankweng
Code: 031
Cell number: 072
Email address: tiro@gmail.com

ID: 555
Full name: Beyers Naude
Street: 15 Mandela
City: Pretoria
Code: 0100
Cell number: 061
Email address: naude@gmail.com

Please click [here](#) to get back to the menu page or [here](#) to the main page.

8.3.3 Using basic annotations and collections.

Example

Create a web application that will perform part of the **CRUD** (Create Read Update Delete) operations on a list of employees. Specifically we want the application to be able to create and read the list. An employee has an id, two contact numbers which is the employee's cell number and that of a close relative. The table below shows all the fields of an employee and the applicable constraints to each field.

Field	Constraint
id	Primary key.
contactNos	Must be a list of strings.
creationDate	Must be a Date. Can be nullable. The maximum length is 20.

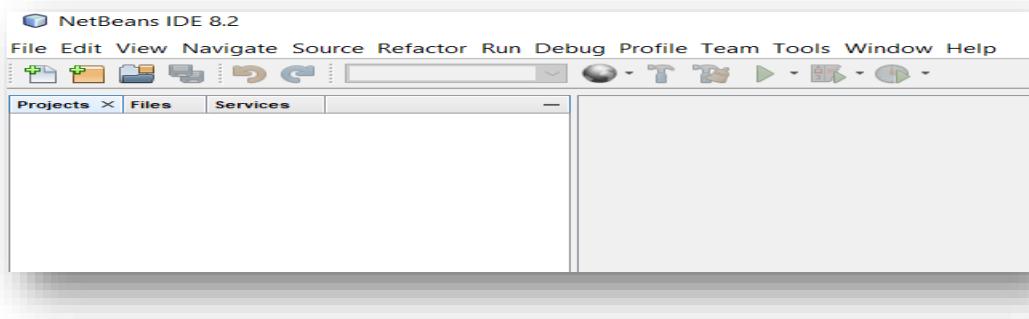
Solution approach

The solution to this problem is going to be done in five parts, namely:

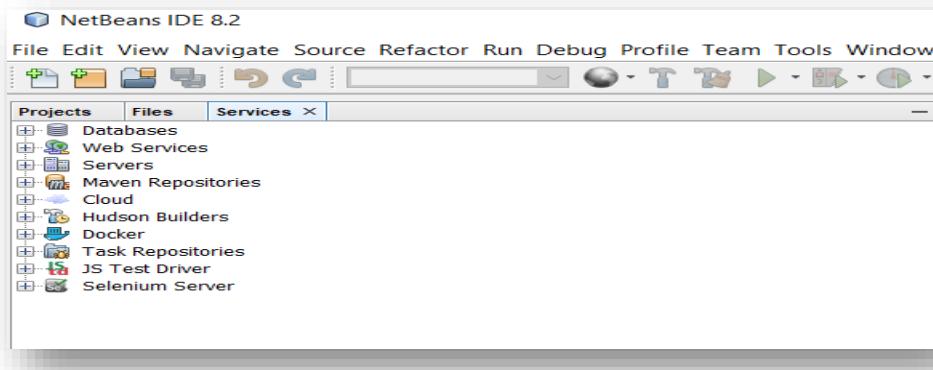
- **Part A:** we create a database.
- **Part B:** we establish connection between the created database and GlassFish.
- **Part C:** we create an EJB module. The module will have an entity with business logic.
- **Part D:** we create a client web application.
- **Part E:** we run the client web application.

Part A - Create a database

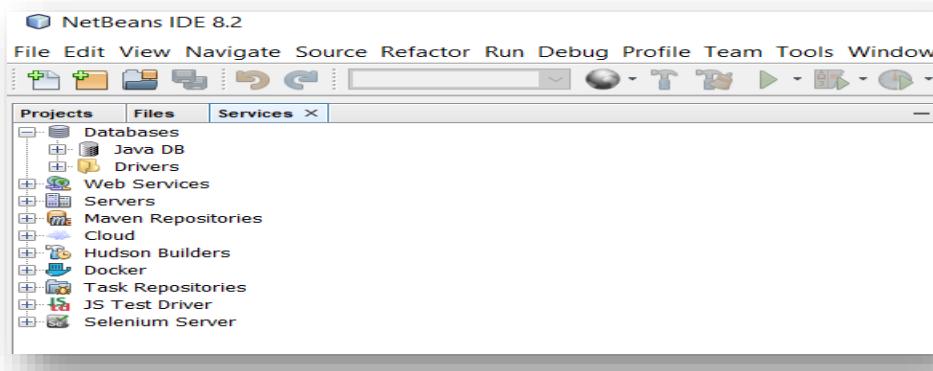
Launch NetBeans.



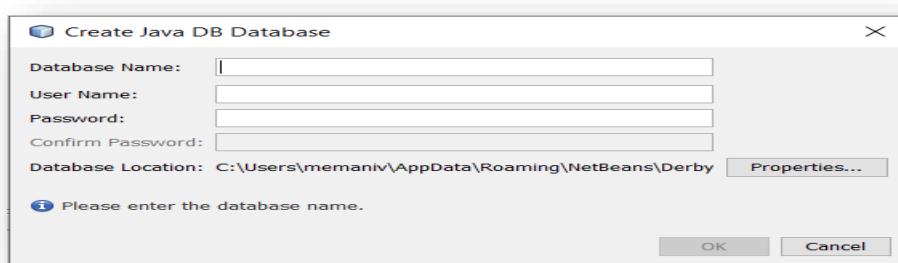
Click on the **Services** tab.



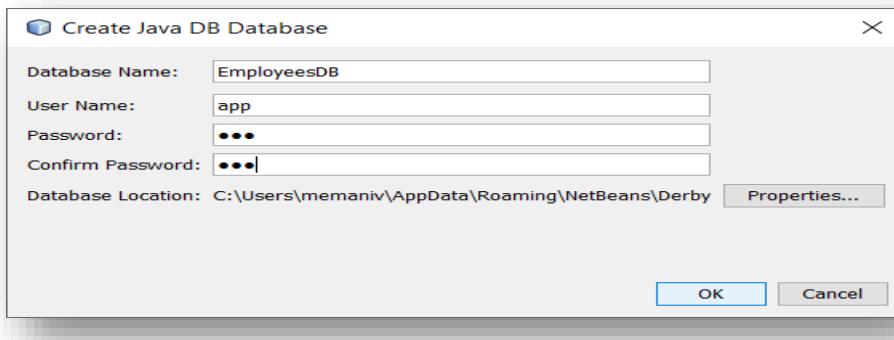
Expand the **Databases** option.



Right-click on **Java DB** and select **Create Database**.

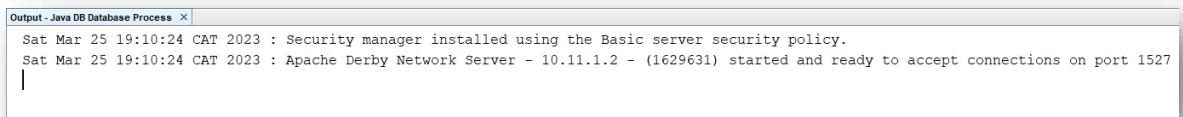


Fill-in the form. I made the password to be 123.

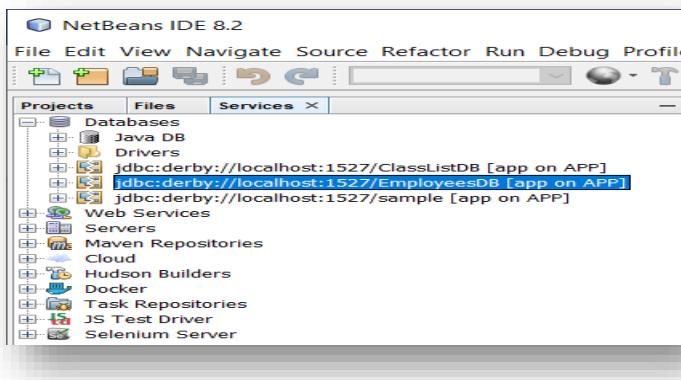


Click **OK**.

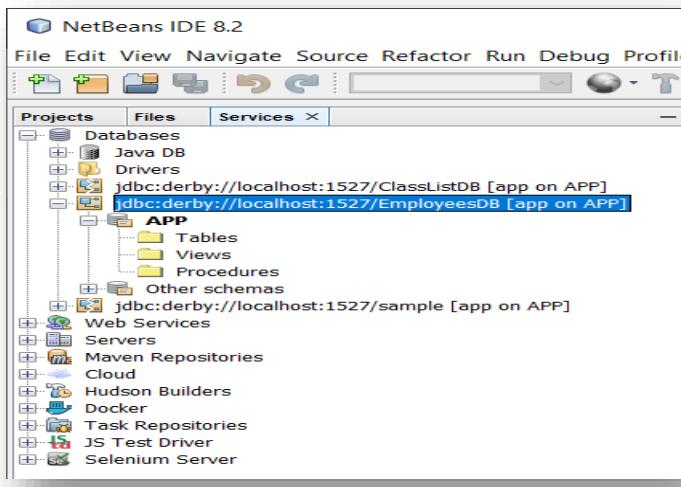
- ✓ An output message confirming that **Derby** has started and ready to accept connections on port 1527 is shown.



- ✓ A database is created.

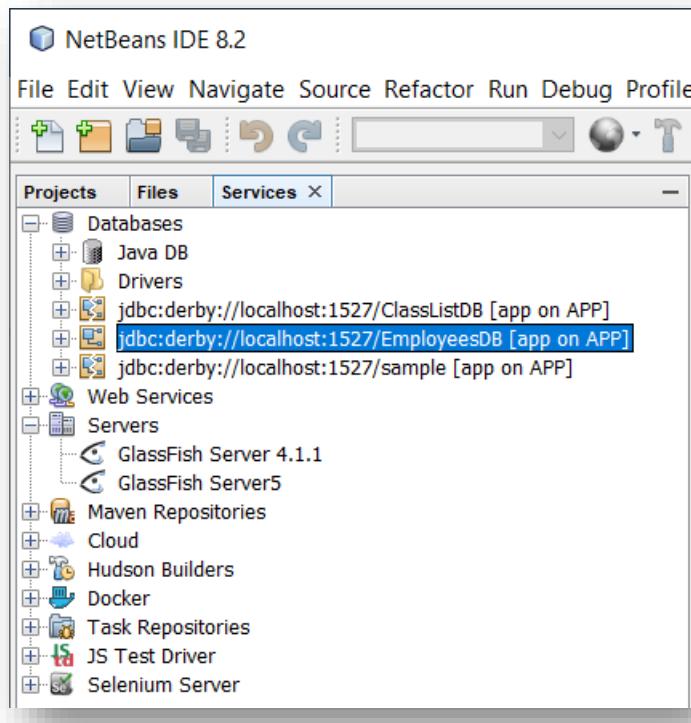


Right-click on the database and select **Connect**.

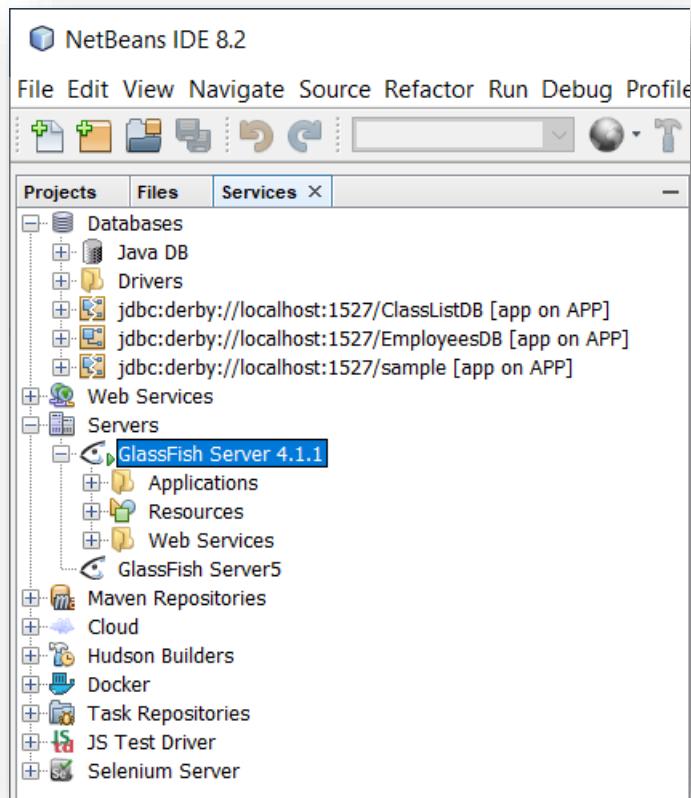


Part B – Connect the database to the application server.

Click on the **Services** tab.



Start the server.



Right-click on the server and select **View Domain Admin Console**.

The screenshot shows the GlassFish Server Open Source Edition Common Tasks console. The URL in the address bar is <http://localhost:4848/common/index.jsf>. The top navigation bar includes links for Home and About... with the current user information: User: admin | Domain: domain1 | Server: localhost. The title bar reads "GlassFish™ Server Open Source Edition". The left sidebar, titled "Common Tasks", contains the following categories: Domain, server (Admin Server), Clusters, Standalone Instances, Nodes, Applications, Lifecycle Modules, Monitoring Data, Resources (expanded to show Concurrent Resources, Connectors, JDBC, JMS Resources, JNDI, JavaMail Sessions, Resource Adapter Configs), Configurations (expanded to show default-config, server-config, Update Tool). The main content area is titled "GlassFish Console - Common Tasks" and lists sections for GlassFish News, Deployment (List Deployed Applications, Deploy an Application), Administration (Change Administrator Password, List Password Aliases), and Monitoring (Monitoring Data).

Under **Common Tasks** panel, expand **JDBC**.

The screenshot shows the GlassFish Server Open Source Edition JDBC console. The URL in the address bar is <http://localhost:4848/common/index.jsf>. The top navigation bar includes links for Home and About... with the current user information: User: admin | Domain: domain1 | Server: localhost. The title bar reads "GlassFish™ Server Open Source Edition". The left sidebar, titled "Common Tasks", contains the following categories: Domain, server (Admin Server), Clusters, Standalone Instances, Nodes, Applications, Lifecycle Modules, Monitoring Data, Resources (expanded to show Concurrent Resources, Connectors, JDBC (selected), JMS Resources, JNDI, JavaMail Sessions, Resource Adapter Configs), Configurations (expanded to show default-config, server-config, Update Tool). The main content area is titled "JDBC" and displays two items: JDBC Resources and JDBC Connection Pools.

Modify the Connection pool to include the **ClassListDB**.

- ✓ Click on **JDBC Connection Pools**.

JDBC Connection Pools

To store, organize, and retrieve data, most applications use relational databases. Java EE applications access relational databases through the JDBC API. Before an application can access a database, it must get a connection.

Select	Pool Name	Resource Type	Classname	Description
<input type="checkbox"/>	DerbyPool	javax.sql.DataSource	org.apache.derby.jdbc.ClientDataSource	
<input type="checkbox"/>	SamplePool	javax.sql.DataSource	org.apache.derby.jdbc.ClientDataSource	
<input type="checkbox"/>	TimerPool	javax.sql.XADatasource	org.apache.derby.jdbc.EmbeddedXADatasource	

- ✓ Click on the **DerbyPool** link.

Edit JDBC Connection Pool

Modify an existing JDBC connection pool. A JDBC connection pool is a group of reusable connections for a particular database.

General Settings

Pool Name: DerbyPool
Resource Type: javax.sql.DataSource
Must be specified if the datasource class implements more than 1 of the interface.
Datasource Classname: org.apache.derby.jdbc.ClientDataSource
Vendor-specific classname that implements the DataSource and/or XADatasource APIs
Driver Classname:
Vendor-specific classname that implements the java.sql.Driver interface.
Ping: Enabled
When enabled, the pool is pinged during creation or reconfiguration to identify and warn of any erroneous values for its attributes
Deployment Order: 100
Specifies the loading order of the resource at server startup. Lower numbers are loaded first.
Description:

Pool Settings

Initial and Minimum Pool Size: 8 Connections
Minimum and initial number of connections maintained in the pool
Maximum Pool Size: 32 Connections
Maximum number of connections that can be created to satisfy client requests
Pool Resize Quantity: 2 Connections

- ✓ Click on **Additional Properties**.

Edit JDBC Connection Pool Properties

Modify properties of an existing JDBC connection pool.

Pool Name: DerbyPool

Additional Properties (6)

Select	Name	Value
<input type="checkbox"/>	PortNumber	1527
<input type="checkbox"/>	Password	APP
<input type="checkbox"/>	User	APP
<input type="checkbox"/>	serverName	localhost
<input type="checkbox"/>	DatabaseName	sun-appserv-samples
<input type="checkbox"/>	connectionAttributes	;create=true

- ✓ Click on **Add Property** and add a **url** property with the following value:

jdbc:derby://localhost:1527/EmployeesDB

- ✓ Delete the **connectionAttributes** property.

General Advanced Additional Properties

Edit JDBC Connection Pool Properties

Modify properties of an existing JDBC connection pool.

Pool Name: DerbyPool

Additional Properties (6)			
		Add Property	Delete Properties
Select	Name	Value	
<input type="checkbox"/>	PortNumber	1527	
<input type="checkbox"/>	Password	123	
<input type="checkbox"/>	User	app	
<input type="checkbox"/>	serverName	localhost	
<input type="checkbox"/>	DatabaseName	EmployeesDB	
<input type="checkbox"/>	url	jdbc:derby://localhost:1527/EmployeesDB	

- ✓ Save the changes by clicking the **Save** button.

General Advanced Additional Properties

New values successfully saved.

Edit JDBC Connection Pool Properties

Modify properties of an existing JDBC connection pool.

Pool Name: DerbyPool

Additional Properties (6)			
		Add Property	Delete Properties
Select	Name	Value	
<input type="checkbox"/>	PortNumber	1527	
<input type="checkbox"/>	Password	123	
<input type="checkbox"/>	User	app	
<input type="checkbox"/>	serverName	localhost	
<input type="checkbox"/>	DatabaseName	EmployeesDB	
<input type="checkbox"/>	url	jdbc:derby://localhost:1527/EmployeesDB	

- ✓ Check if the created connection pool is working. Do this by selecting the General tab and then click on the **Ping** button.

The screenshot shows the 'Edit JDBC Connection Pool' dialog with the 'General' tab selected. At the top right, there is a yellow button labeled 'Ping Succeeded' with a green checkmark icon. Below it, the title 'Edit JDBC Connection Pool' is displayed, followed by a brief description: 'Modify an existing JDBC connection pool. A JDBC connection pool is a group of reusable connections for a particular database.' Below the description are three buttons: 'Load Defaults', 'Flush', and 'Ping'. The main configuration area is titled 'General Settings' and contains the following fields:

Pool Name:	DerbyPool
Resource Type:	javax.sql.DataSource
Datasource Classname:	org.apache.derby.jdbc.ClientDataSource
Driver Classname:	
Ping:	<input checked="" type="checkbox"/> Enabled
Deployment Order:	100
Description:	

Below the table, a note states: 'Must be specified if the datasource class implements more than 1 of the interface.'

Confirm that the data source (resource) points to the DerbyPool.

- ✓ Still under **JDBC**, click on **JDBC Resources**.

The screenshot shows the 'JDBC Resources' table with the heading 'Resources (3)'. The table has columns: Select, JNDI Name, Logical JNDI Name, Enabled, and Connection Pool. The data is as follows:

Select	JNDI Name	Logical JNDI Name	Enabled	Connection Pool
<input type="checkbox"/>	jdbc/_TimerPool		<input checked="" type="checkbox"/>	_TimerPool
<input type="checkbox"/>	jdbc/_default	java.comp/DefaultDataSource	<input checked="" type="checkbox"/>	DerbyPool
<input type="checkbox"/>	jdbc/sample		<input checked="" type="checkbox"/>	SamplePool

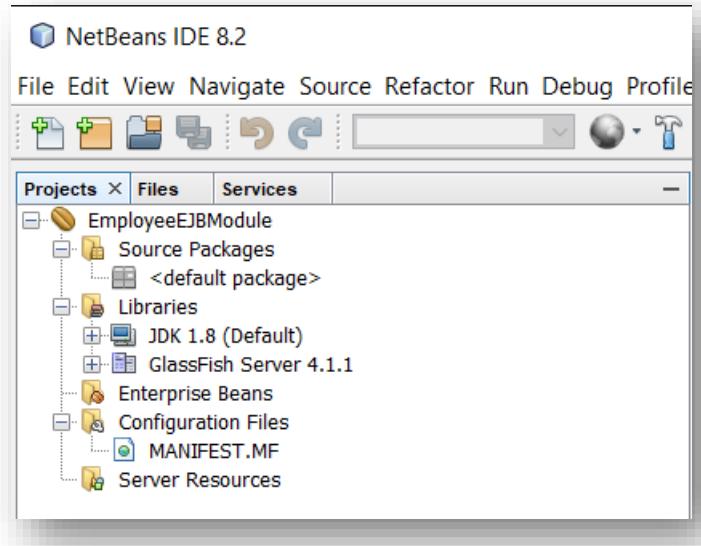
Take note of the purpose of JDBC Resources. It is said:

"JDBC resources provide applications with a means to connect to a database."

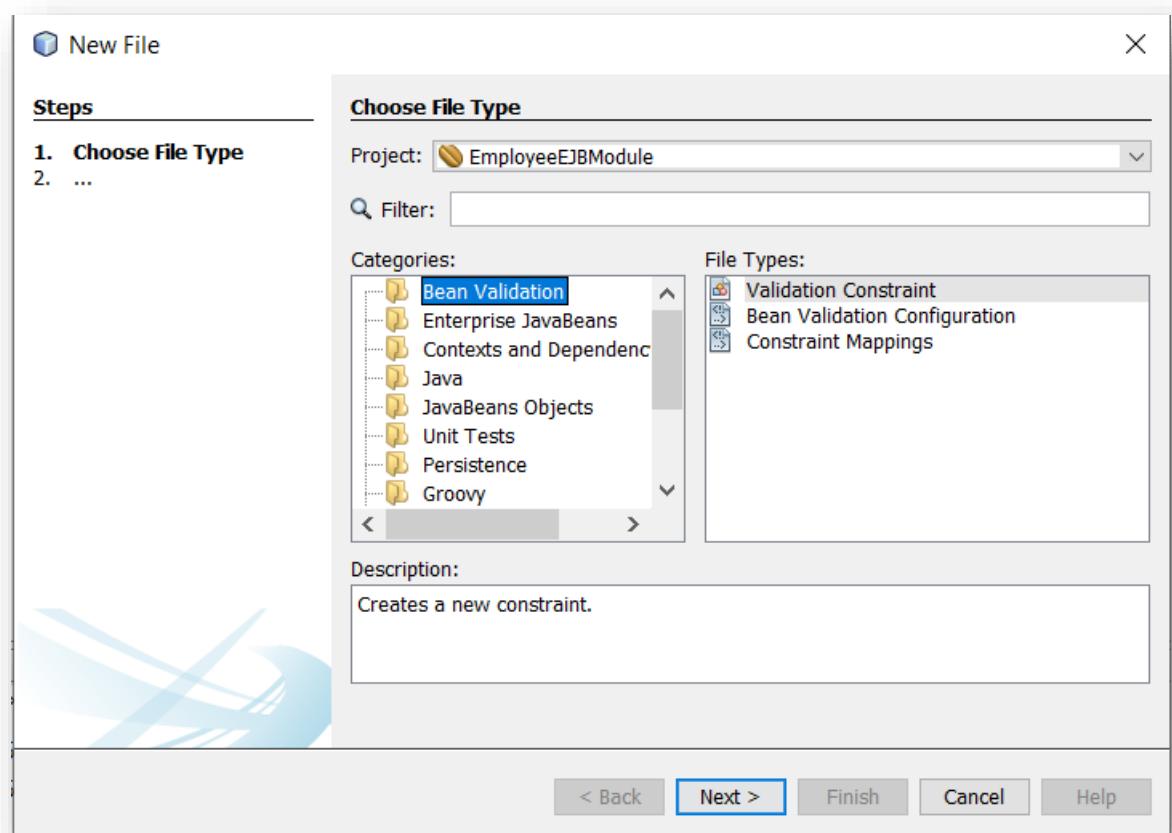
You can also see that the **jdbc/_default** resource is associated with the **DerbyPool** Connection Pool.

Part C – Create an entity.

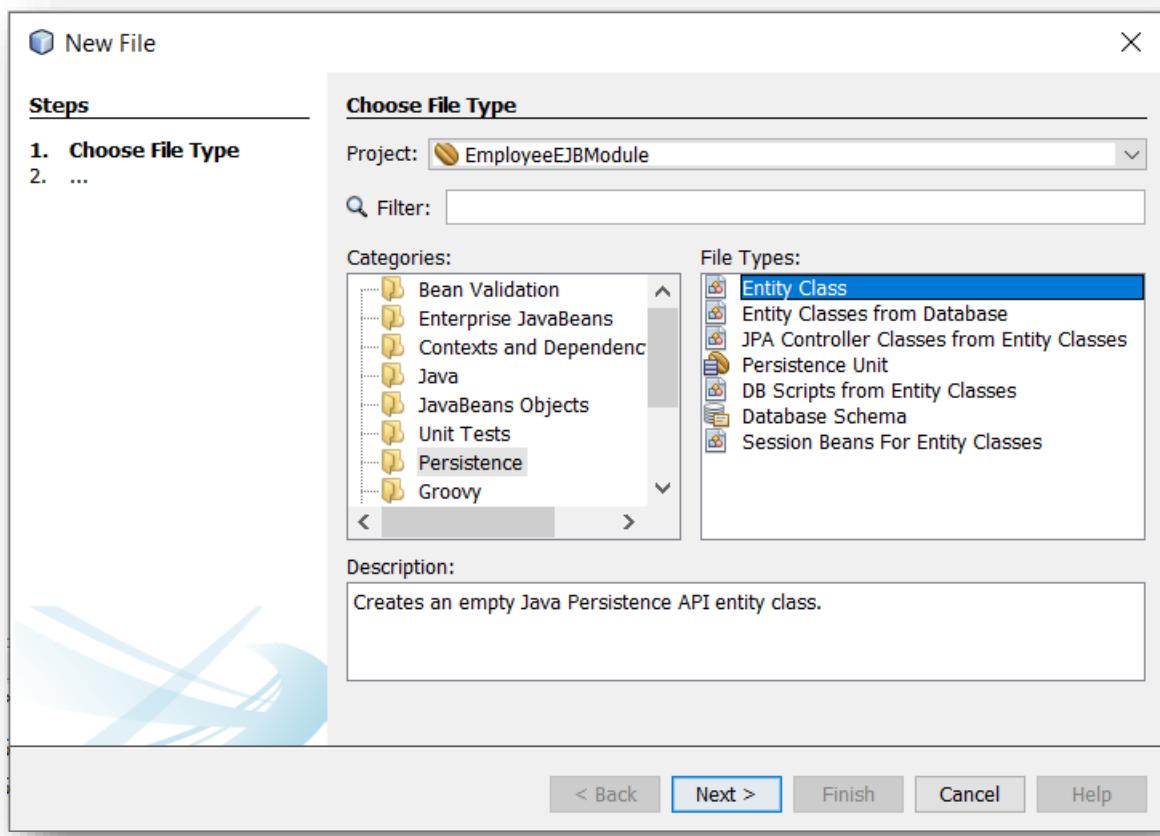
Create an EJB project called **EmployeeEJBModule**.



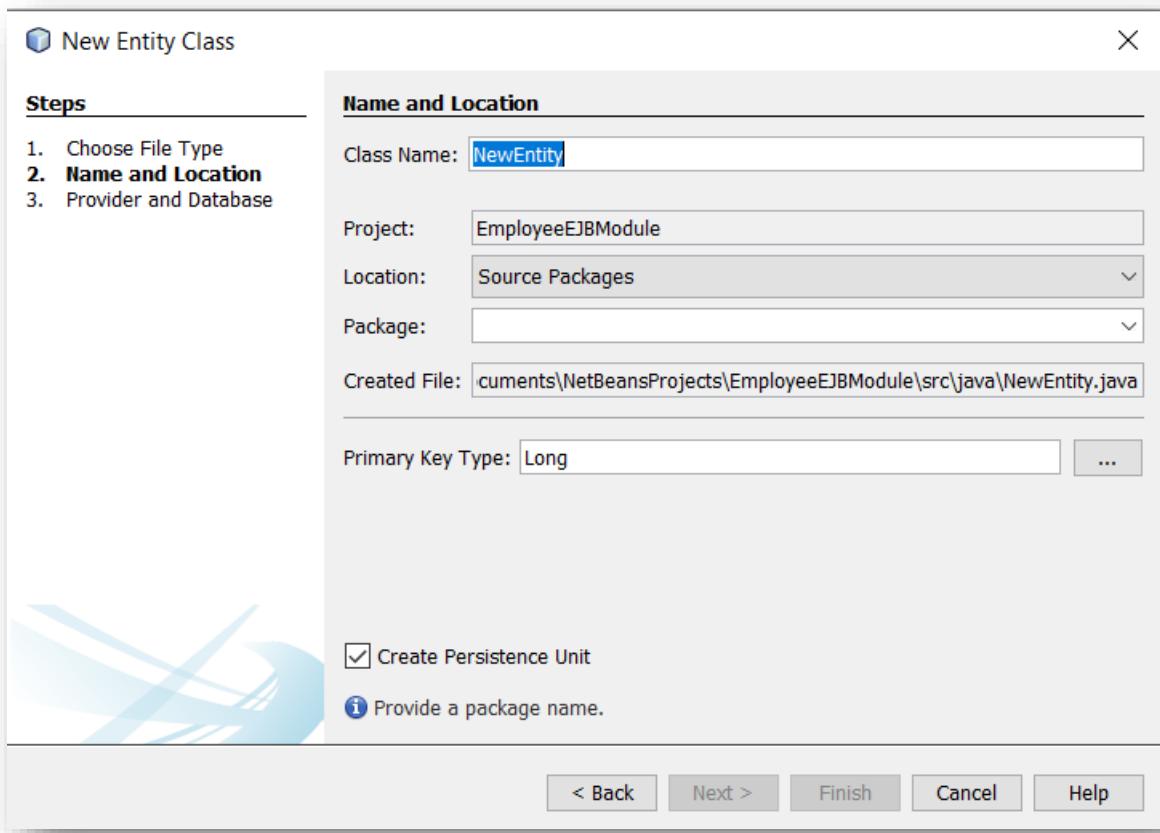
Right-click on the project and select **New | Other**.



Under **Categories** select **Persistence**, and under **File Types** select **Entity Class**.



Click **Next**.



Fill-in the form.

New Entity Class

Steps

1. Choose File Type
2. **Name and Location**
3. Provider and Database

Name and Location

Class Name: Employee

Project: EmployeeEJBModule

Location: Source Packages

Package: za.ac.tut.entities

Created File: sProjects\EmployeeEJBModule\src\java\za\ac\tut\entities\Employee.java

Primary Key Type: Long

Create Persistence Unit

< Back **Next >** Finish Cancel Help

Click **Next**.

New File

Steps

1. Choose File Type
2. Name and Location
3. **Provider and Database**

Provider and Database

Persistence Unit Name: EmployeeEJBModulePU

Specify the persistence provider and database for entity classes.

Persistence Provider: EclipseLink (JPA 2.1)(default)

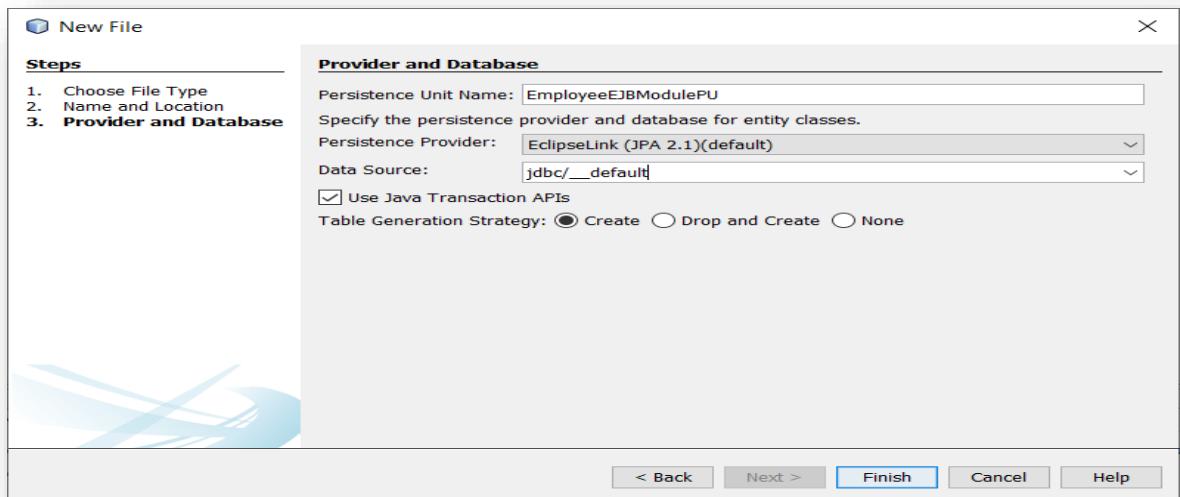
Data Source:

Use Java Transaction APIs

Table Generation Strategy: Create Drop and Create None

< Back **Next >** Finish Cancel Help

Under **Data Source**, select **jdbc/_default**.



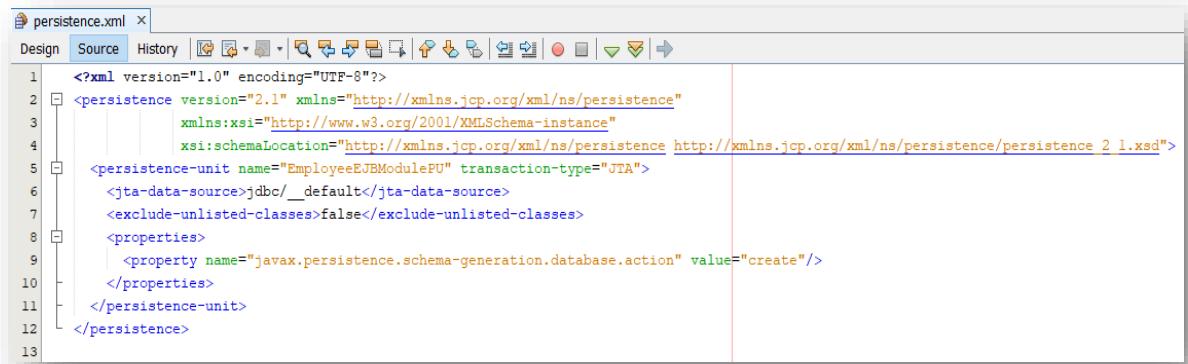
Click **Finish**.

```
1 package za.ac.tut.entities;
2
3 import java.io.Serializable;
4 import java.util.ArrayList;
5 import java.util.Date;
6 import java.util.List;
7 import javax.persistence.CollectionTable;
8 import javax.persistence.Column;
9 import javax.persistence.ElementCollection;
10 import javax.persistence.Entity;
11 import javax.persistence.FetchType;
12 import javax.persistence.Id;
13 import javax.persistence.Temporal;
14
15 /**
16 * 
17 * @author MemaniV
18 */
19 @Entity
20 public class Employee implements Serializable {
21
22     private static final long serialVersionUID = 1L;
23     @Id
24     private Long id;
25     @ElementCollection(fetch=FetchType.LAZY)
26     @CollectionTable(name="Contacts")
27     @Column(name="Contact")
28     private List<String> contactNos = new ArrayList<>();
29     @Temporal(javax.persistence.TemporalType.TIMESTAMP)
30     private Date creationDate;
```

```
32 |     public Employee() {
33 |     }
34 |
35 |     public Employee(Long id, Date creationDate) {
36 |         this.id = id;
37 |         this.creationDate = creationDate;
38 |     }
39 |
40 |     public List<String> getContactNos() {
41 |         return contactNos;
42 |     }
43 |
44 |     public void setContactNos(List<String> contactNos) {
45 |         this.contactNos = contactNos;
46 |     }
47 |
48 |     public Date getCreationDate() {
49 |         return creationDate;
50 |     }
51 |
52 |     public void setCreationDate(Date creationDate) {
53 |         this.creationDate = creationDate;
54 |     }
55 |
56 |     public Long getId() {
57 |         return id;
58 |     }
59 |
60 |     public void setId(Long id) {
61 |         this.id = id;
62 |     }
63 |
64 |     @Override
65 |     @Override
66 |     public int hashCode() {
67 |         int hash = 0;
68 |         hash += (id != null ? id.hashCode() : 0);
69 |         return hash;
    }
```

```
71 |     @Override
72 |     @Override
73 |     public boolean equals(Object object) {
74 |         if (!(object instanceof Employee)) {
75 |             return false;
76 |         }
77 |         Employee other = (Employee) object;
78 |         if ((this.id == null && other.id != null) ||
79 |             (this.id != null && !this.id.equals(other.id))) {
80 |             return false;
81 |         }
82 |         return true;
83 |     }
84 |
85 |     @Override
86 |     @Override
87 |     public String toString() {
88 |         return "za.ac.tut.entities.Employee[ id=" + id + " ]";
89 |     }
90 | }
```

View the persistence.xml file.

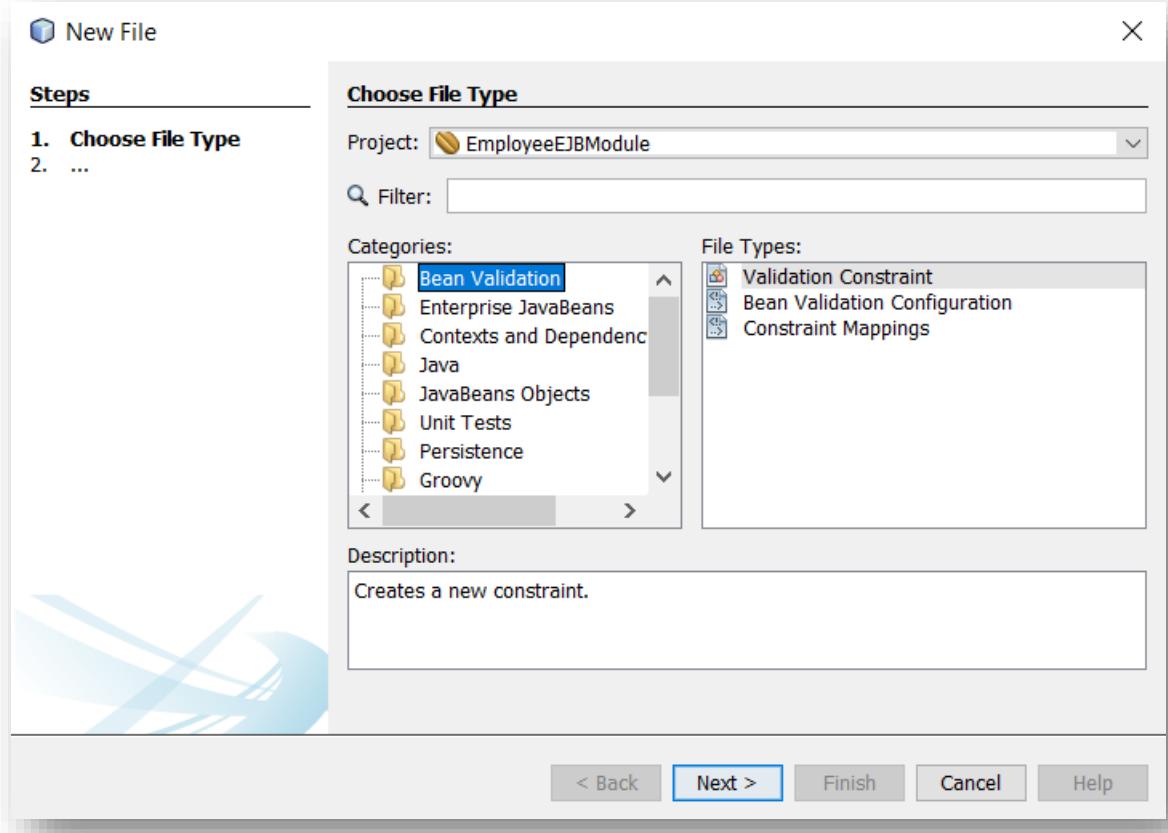


```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
    <persistence-unit name="EmployeeEJBModulePU" transaction-type="JTA">
        <jta-data-source>jdbc/_default</jta-data-source>
        <exclude-unlisted-classes>false</exclude-unlisted-classes>
        <properties>
            <property name="javax.persistence.schema-generation.database.action" value="create"/>
        </properties>
    </persistence-unit>
</persistence>
```

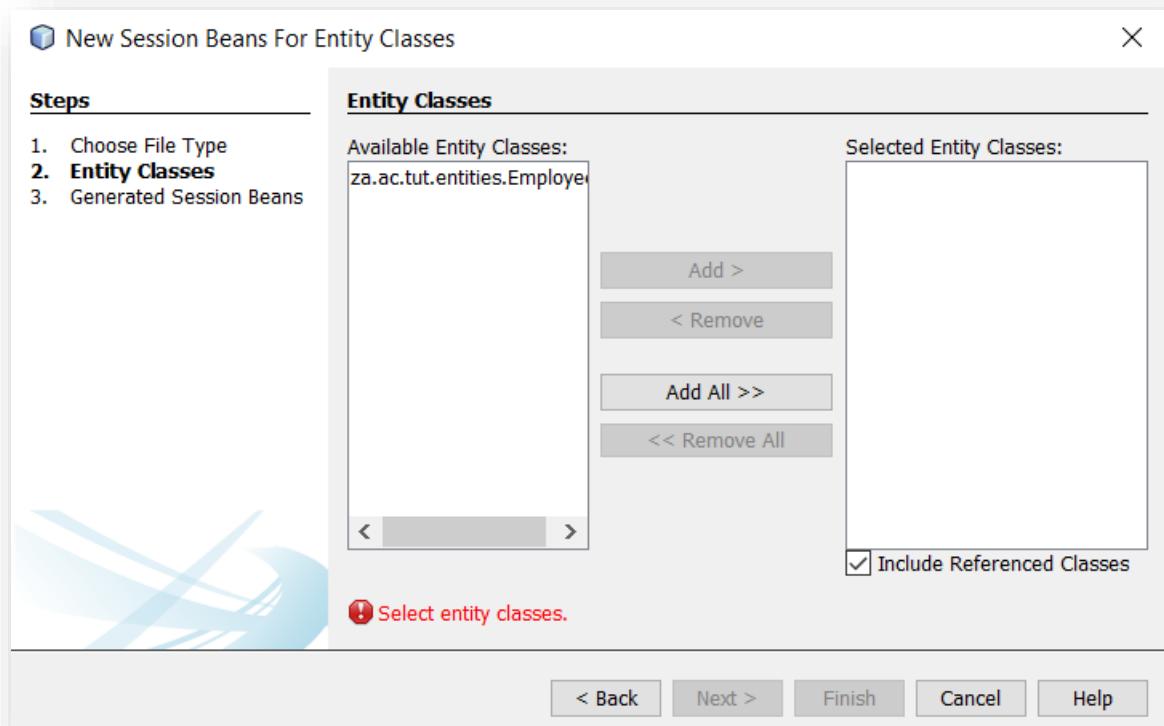
Create business code (Create employee and search for employee) for the entity.

Perform the following steps:

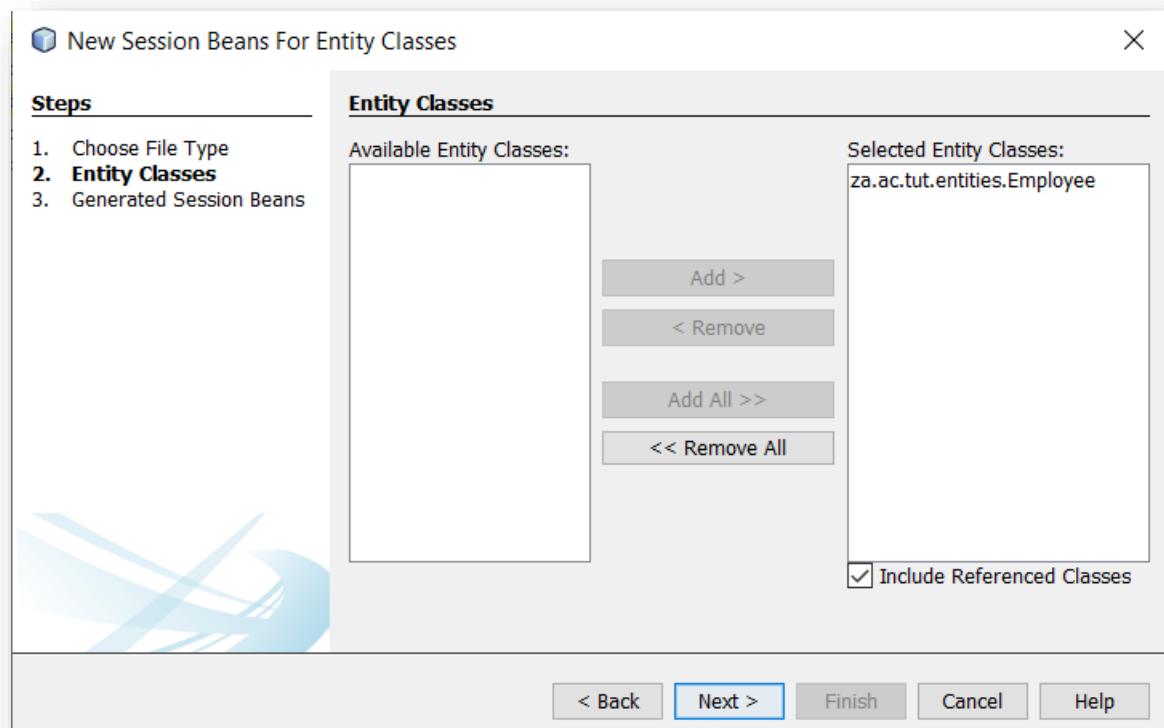
- ✓ Right-click on the project and select **New | Other**. Under **Categories**, select **Enterprise JavaBeans**. Under **File Types** select **Session Beans for Entity Classes**.



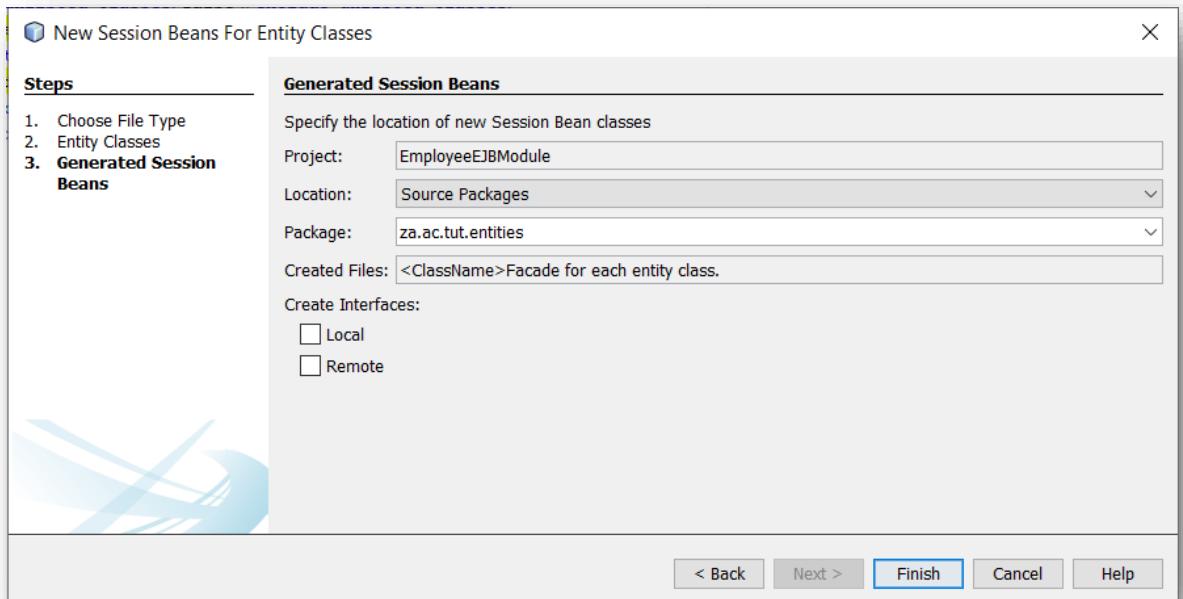
✓ Click **Next**.



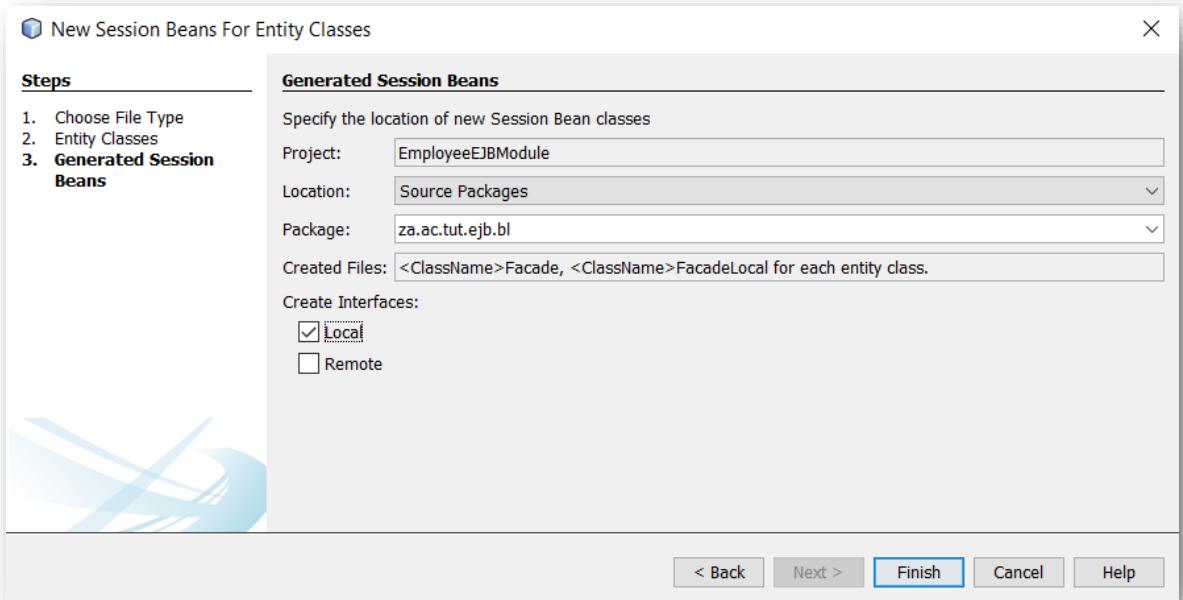
✓ Select **za.ac.tut.entities.Employee** under **Available Entity Classes** and click **Add**.



✓ Click **Next**.



✓ Modify the package name to **za.ac.tut.ejb.bl**, where **bl** stands for **business logic**. Also select **Local** interface.



- ✓ Click **Finish**. View created files.
 - **EmployeeFacade.**

The screenshot shows a Java code editor with the file `EmployeeFacade.java` open. The code implements a stateless EJB facade for the `Employee` entity. It includes imports for `Stateless`, `EntityManager`, `PersistenceContext`, and the `Employee` entity. The class is annotated with `@Stateless` and `@PersistenceContext(unitName = "EmployeeEJBModulePU")`. It overrides the `getEntityManager` method and provides a constructor that calls the superclass constructor with `Employee.class`.

```
1 package za.ac.tut.ejb.bl;
2
3 import javax.ejb.Stateless;
4 import javax.persistence.EntityManager;
5 import javax.persistence.PersistenceContext;
6 import za.ac.tut.entities.Employee;
7
8 /**
9  * @author MemaniV
10 */
11
12 @Stateless
13 public class EmployeeFacade extends AbstractFacade<Employee> implements EmployeeFacadeLocal {
14
15     @PersistenceContext(unitName = "EmployeeEJBModulePU")
16     private EntityManager em;
17
18     @Override
19     protected EntityManager getEntityManager() {
20         return em;
21     }
22
23     public EmployeeFacade() {
24         super(Employee.class);
25     }
26
27 }
```

▪ AbstractFacade.

```

1 package za.ac.tut.bl;
2
3 import java.util.List;
4 import javax.persistence.EntityManager;
5
6 /**
7 *
8 * @author MemaniV
9 */
10 public abstract class AbstractFacade<T> {
11
12     private Class<T> entityClass;
13
14     public AbstractFacade(Class<T> entityClass) {
15         this.entityClass = entityClass;
16     }
17
18     protected abstract EntityManager getEntityManager();
19
20     public void create(T entity) {
21         getEntityManager().persist(entity);
22     }
23
24     public void edit(T entity) {
25         getEntityManager().merge(entity);
26     }
27
28     public void remove(T entity) {
29         getEntityManager().remove(getEntityManager().merge(entity));
30     }
31

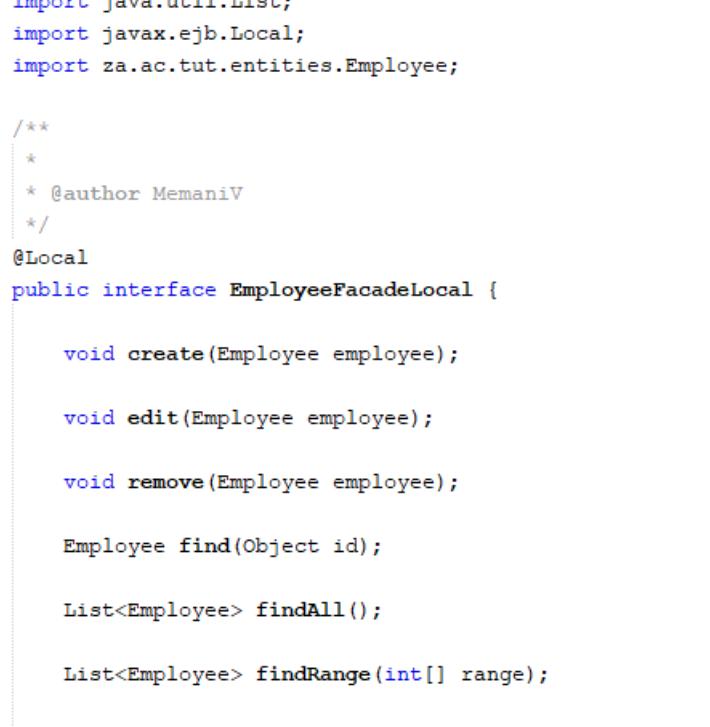
```

```

32     public T find(Object id) {
33         return getEntityManager().find(entityClass, id);
34     }
35
36     public List<T> findAll() {
37         javax.persistence.criteria.CriteriaQuery cq =
38             getEntityManager().getCriteriaBuilder().createQuery();
39         cq.select(cq.from(entityClass));
40         return getEntityManager().createQuery(cq).getResultList();
41     }
42
43     public List<T> findRange(int[] range) {
44         javax.persistence.criteria.CriteriaQuery cq =
45             getEntityManager().getCriteriaBuilder().createQuery();
46         cq.select(cq.from(entityClass));
47         javax.persistence.Query q = getEntityManager().createQuery(cq);
48         q.setMaxResults(range[1] - range[0] + 1);
49         q.setFirstResult(range[0]);
50         return q.getResultList();
51     }
52
53     public int count() {
54         javax.persistence.criteria.CriteriaQuery cq =
55             getEntityManager().getCriteriaBuilder().createQuery();
56         javax.persistence.criteria.Root<T> rt = cq.from(entityClass);
57         cq.select(getEntityManager().getCriteriaBuilder().count(rt));
58         javax.persistence.Query q = getEntityManager().createQuery(cq);
59         return ((Long) q.getSingleResult()).intValue();
60     }
61
62 }

```

▪ EmployeeFacadeLocal.



The screenshot shows a Java code editor with the file `EmployeeFacadeLocal.java` open. The code defines a local interface for managing employees. It includes imports for `List`, `javax.ejb.Local`, and the `Employee` entity. The interface has methods for creating, editing, removing, finding by ID, finding all, finding by range, and counting employees.

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

import java.util.List;
import javax.ejb.Local;
import za.ac.tut.entities.Employee;

/**
 * 
 * @author MemaniV
 */

@Local
public interface EmployeeFacadeLocal {

    void create(Employee employee);

    void edit(Employee employee);

    void remove(Employee employee);

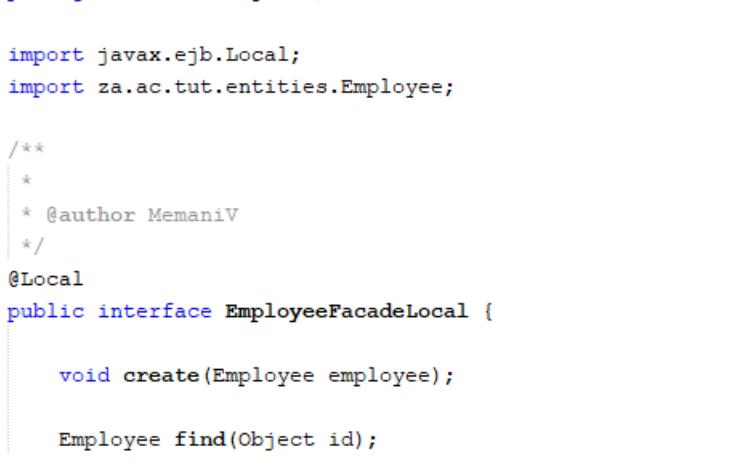
    Employee find(Object id);

    List<Employee> findAll();

    List<Employee> findRange(int[] range);

    int count();
}
```

- ✓ Remove all the other interface methods except create and find.



The screenshot shows a Java code editor with the file `EmployeeFacadeLocal.java` open. The code defines a local interface for an employee facade. It includes imports for `javax.ejb.Local` and `za.ac.tut.entities.Employee`. The interface has a single method `create` which takes an `Employee` object as a parameter, and another method `find` which takes an `Object` representing an ID.

```
1 package za.ac.tut.ejb.bl;
2
3 import javax.ejb.Local;
4 import za.ac.tut.entities.Employee;
5
6 /**
7 * @author MemaniV
8 */
9
10 @Local
11 public interface EmployeeFacadeLocal {
12
13     void create(Employee employee);
14
15     Employee find(Object id);
16
17 }
18
```

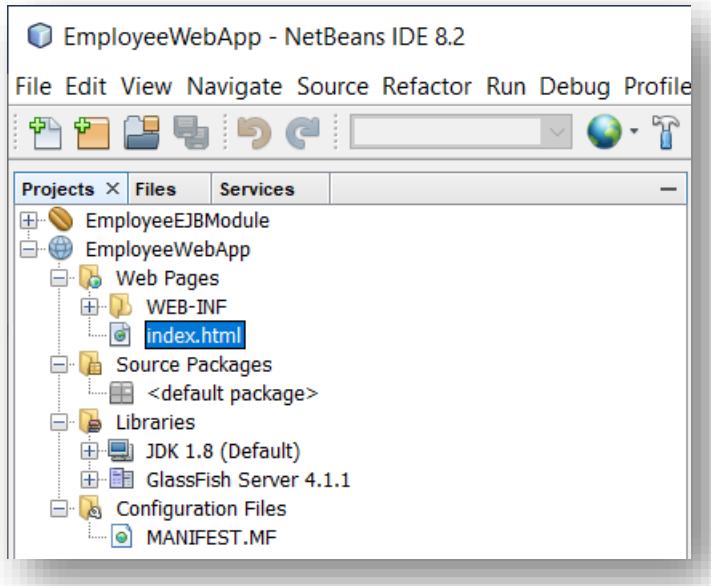
- ✓ Clean and build the EJB project.

The screenshot shows the NetBeans IDE interface. The top part is the Output window, which displays the build logs for the EmployeeEJBModule. The logs show the process of copying files, creating directories, compiling source files, and building the jar file. It concludes with a "BUILD SUCCESSFUL" message. The bottom part is the Project Explorer, showing the structure of the EmployeeEJBModule project. It includes sub-folders like build, dist, nbproject, src, and test, and contains files such as EmployeeEJBModule.jar, MANIFEST.MF, persistence.xml, AbstractFacade.class, EmployeeFacade.class, EmployeeFacadeLocal.class, Employee.class, Employee_.class, and build.xml.

```
Copying 2 files to C:\Users\memaniv\Documents\NetBeansProjects\EmployeeEJBModule\build\classes\META-INF
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\EmployeeEJBModule\build\empty
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\EmployeeEJBModule\build\generated-sources\ap-source
Compiling 4 source files to C:\Users\memaniv\Documents\NetBeansProjects\EmployeeEJBModule\build\classes
warning: Supported source version 'RELEASE_6' from annotation processor 'org.eclipse.persistence.internal.jp
Note: Creating static metadata factory ...
Note: Optional file was not found: META-INF/orm.xml continuing with generation.
Note: Optional file was not found: META-INF/eclipselink-orm.xml continuing with generation.
Note: Optional file was not found: META-INF/orm.xml continuing with generation.
Note: Optional file was not found: META-INF/eclipselink-orm.xml continuing with generation.
Note: C:\Users\memaniv\Documents\NetBeansProjects\EmployeeEJBModule\src\java\za\ac\tut\ejb\bl\AbstractFacade
Note: Recompile with -Xlint:unchecked for details.
1 warning
compile:
library-inclusion-in-archive:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\EmployeeEJBModule\dist
Building jar: C:\Users\memaniv\Documents\NetBeansProjects\EmployeeEJBModule\dist\EmployeeEJBModule.jar
dist:
BUILD SUCCESSFUL (total time: 1 second)
```

Part D – Create a web client project.

Create a web client project called **EmployeeWebApp**.

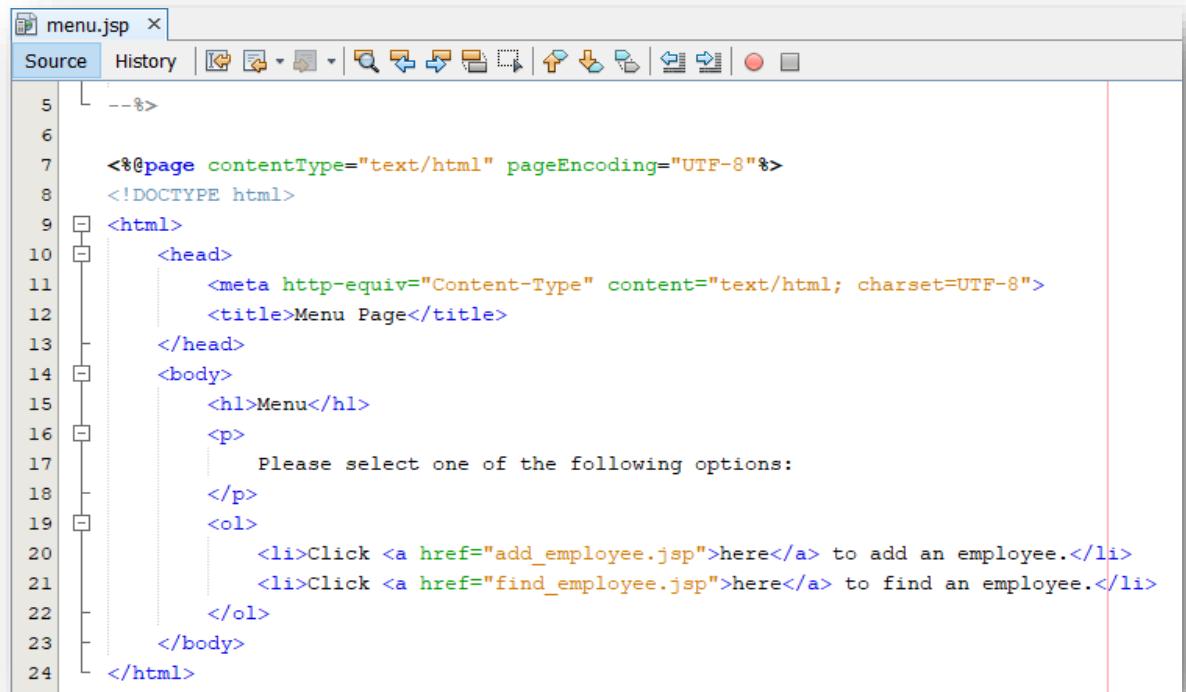


Edit the **index.html** page.

The screenshot shows the NetBeans code editor with the tab "index.html" selected. The "Source" tab is active. The code editor displays the following HTML content:

```
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 <html>
7   <head>
8     <title>Home Page</title>
9     <meta charset="UTF-8">
10    <meta name="viewport" content="width=device-width, initial-scale=1.0">
11  </head>
12  <body>
13    <h1>Welcome</h1>
14    <p>
15      Welcome to our web app. Please click <a href="menu.jsp">here</a> to start.
16    </p>
17  </body>
18</html>
19
20
```

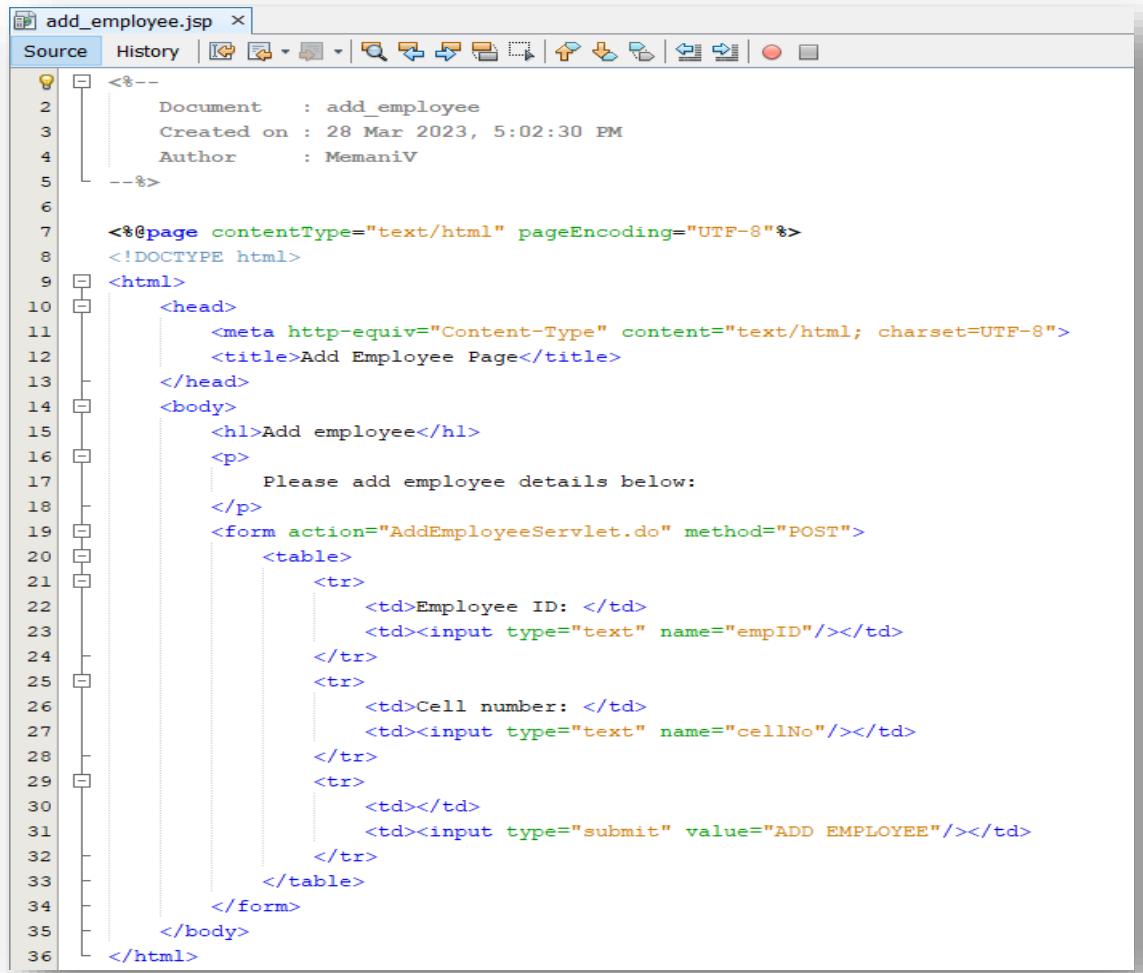
Create the menu.jsp file.



The screenshot shows the code editor of a Java IDE with the file "menu.jsp" open. The code is a simple JSP page that displays a menu. It includes meta tags for content type and character encoding, a title, and a body containing an H1 tag and a list of options for adding or finding employees.

```
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="add_employee.jsp">here</a> to add an employee.</li>
21          <li>Click <a href="find_employee.jsp">here</a> to find an employee.</li>
22      </ol>
23  </body>
24 </html>
```

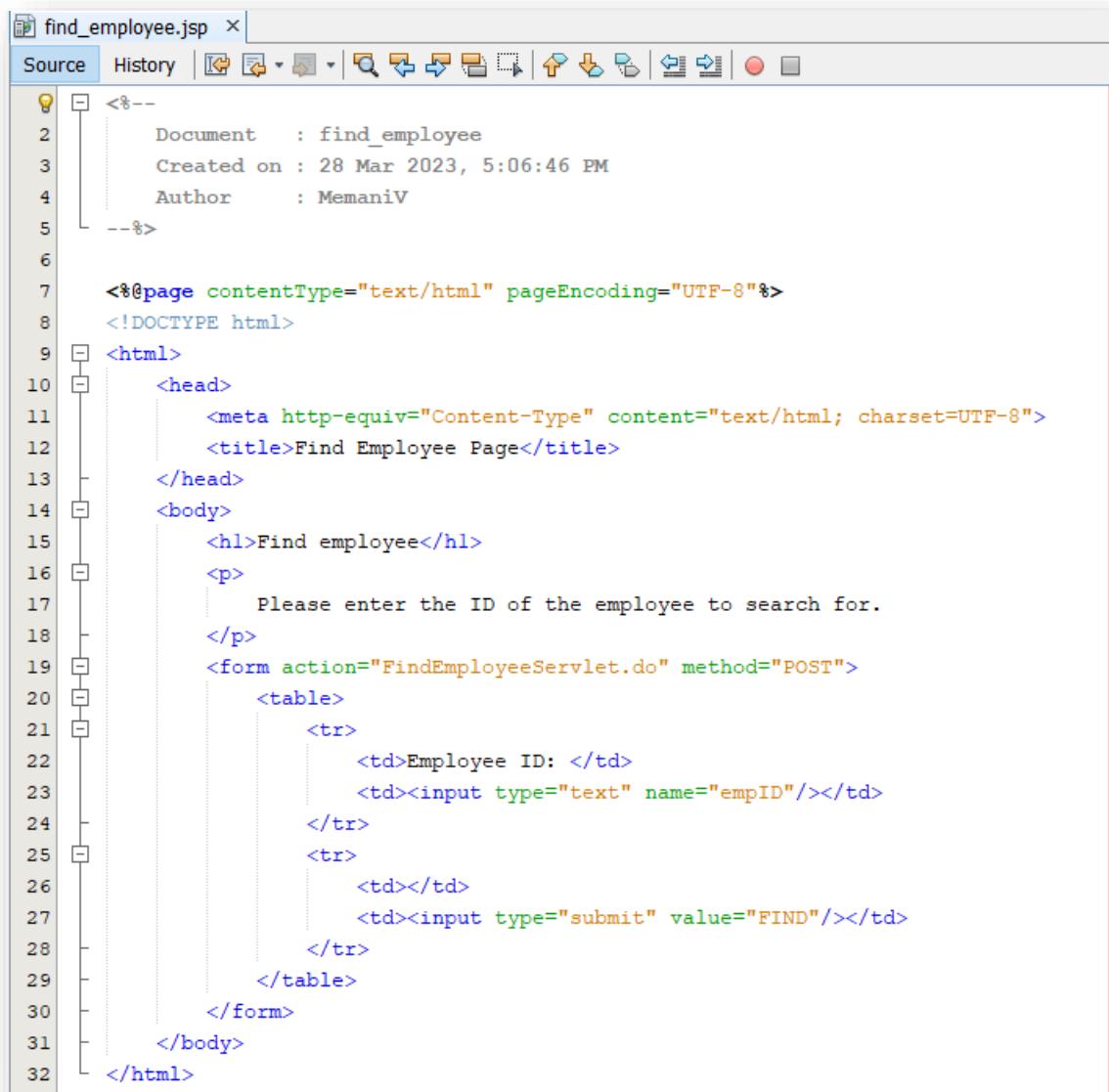
Create add_employee.jsp file.



The screenshot shows the code editor of a Java IDE with the file "add_employee.jsp" open. The code is a JSP page for adding an employee. It includes document metadata, a title, and a form for entering employee details. The form uses a POST method and has three input fields: Employee ID, Cell number, and a submit button.

```
1<%--
2     Document      : add_employee
3     Created on   : 28 Mar 2023, 5:02:30 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>Add Employee Page</title>
13  </head>
14  <body>
15      <h1>Add employee</h1>
16      <p>
17          Please add employee details below:
18      </p>
19      <form action="AddEmployeeServlet.do" method="POST">
20          <table>
21              <tr>
22                  <td>Employee ID:</td>
23                  <td><input type="text" name="empID"/></td>
24              </tr>
25              <tr>
26                  <td>Cell number:</td>
27                  <td><input type="text" name="cellNo"/></td>
28              </tr>
29              <tr>
30                  <td></td>
31                  <td><input type="submit" value="ADD EMPLOYEE"/></td>
32              </tr>
33          </table>
34      </form>
35  </body>
36 </html>
```

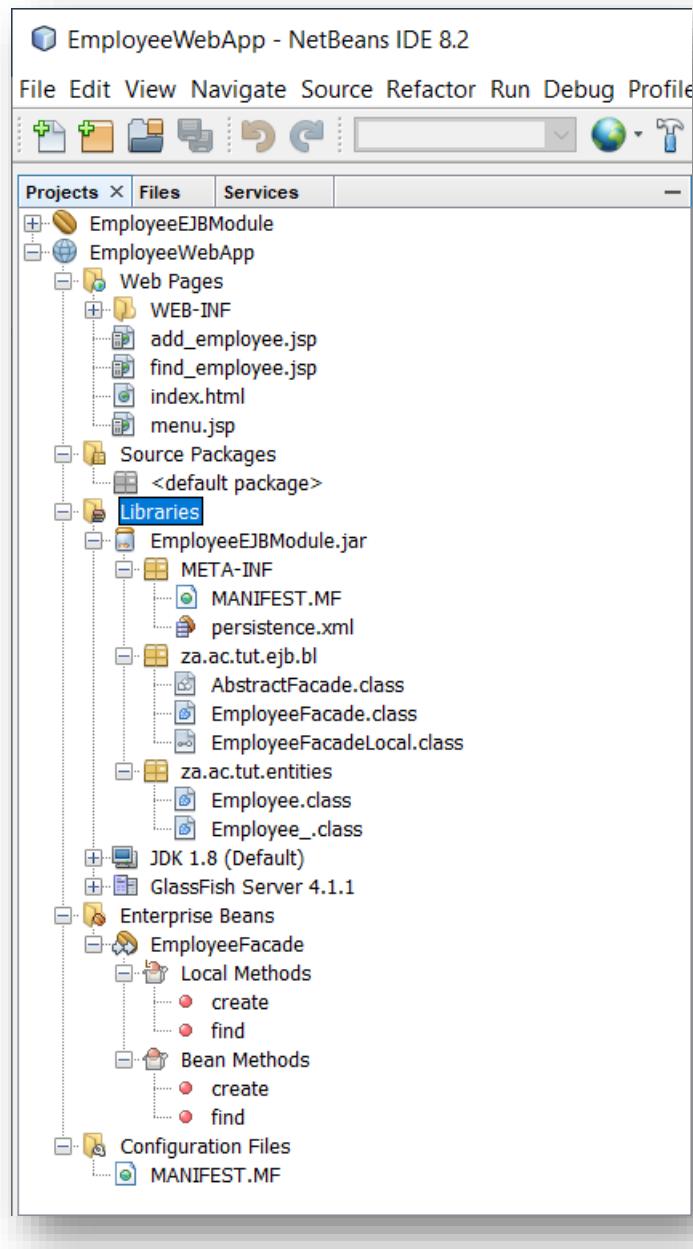
Create the **find_employee.jsp** file.



The screenshot shows a Java IDE interface with a code editor window titled "find_employee.jsp". The window has a toolbar with various icons at the top. The code itself is a JSP page with the following content:

```
<%--  
1 Document      : find_employee  
2 Created on   : 28 Mar 2023, 5:06:46 PM  
3 Author        : MemaniV  
4 --%>  
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>Find Employee Page</title>  
13 </head>  
14 <body>  
15     <h1>Find employee</h1>  
16     <p>  
17         Please enter the ID of the employee to search for.  
18     </p>  
19     <form action="FindEmployeeServlet.do" method="POST">  
20         <table>  
21             <tr>  
22                 <td>Employee ID: </td>  
23                 <td><input type="text" name="empID"/></td>  
24             </tr>  
25             <tr>  
26                 <td></td>  
27                 <td><input type="submit" value="FIND"/></td>  
28             </tr>  
29         </table>  
30     </form>  
31 </body>  
32 </html>
```

Add the **EmployeeEJBModule.jar** file to the library of **EmployeeWebApp**.



Create the AddEmployeeServlet.java file.

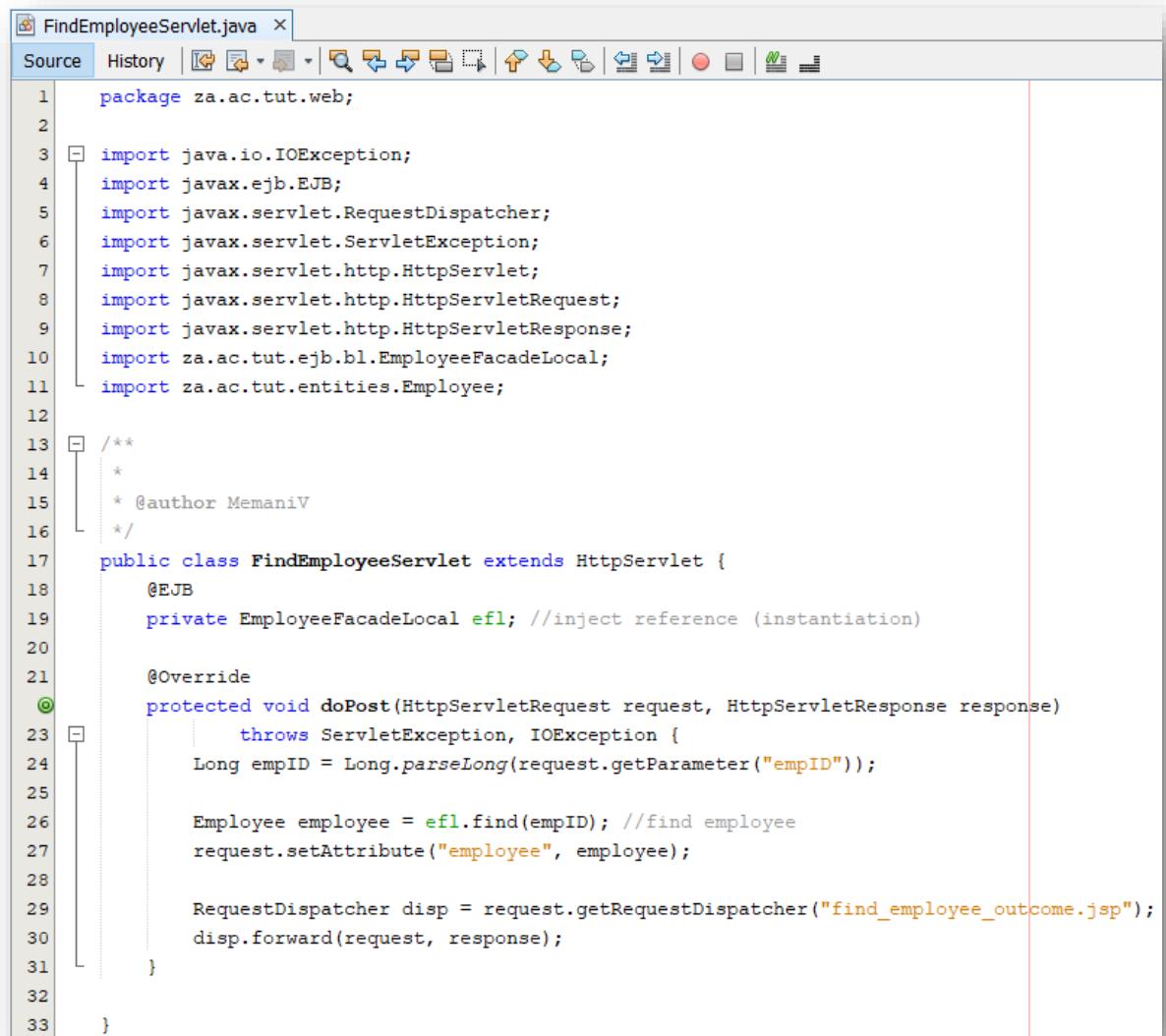
The screenshot shows a Java code editor with the title bar "AddEmployeeServlet.java". The code is a servlet that injects an EJB reference and processes POST requests to create an employee. The code includes imports for various Java and JBoss annotations, and logic to parse request parameters and call the EJB's create method. The code is annotated with Javadoc and includes a copyright notice.

```
1 package za.ac.tut.web;
2
3 import java.io.IOException;
4 import java.util.ArrayList;
5 import java.util.Date;
6 import java.util.List;
7 import javax.ejb.EJB;
8 import javax.servlet.RequestDispatcher;
9 import javax.servlet.ServletException;
10 import javax.servlet.http.HttpServlet;
11 import javax.servlet.http.HttpServletRequest;
12 import javax.servlet.http.HttpServletResponse;
13 import za.ac.tut.ejb.bl.EmployeeFacadeLocal;
14 import za.ac.tut.entities.Employee;
15
16 /**
17 *
18 * @author MemaniV
19 */
20 public class AddEmployeeServlet extends HttpServlet {
21     @EJB
22     private EmployeeFacadeLocal efl; //inject reference (instantiation)
23     @Override
24     protected void doPost(HttpServletRequest request, HttpServletResponse response)
25             throws ServletException, IOException {
26         Long empID = Long.parseLong(request.getParameter("empID"));
27         String empCellNo = request.getParameter("empCellNo");
28         String nextKinCellNo = request.getParameter("nextKinCellNo");
29
30         Employee employee = createEmployee(empID, empCellNo, nextKinCellNo); //create employee
31         efl.create(employee); //persist employee
32
33         RequestDispatcher disp = request.getRequestDispatcher("add_employee_outcome.jsp");
34         disp.forward(request, response);
35     }
36
37     private Employee createEmployee(Long empID, String empCellNo, String nextKinCellNo) {
38         Employee emp = new Employee();
39         List<String> list = new ArrayList<>();
40         list.add(empCellNo);
41         list.add(nextKinCellNo);
42         emp.setContactNos(list);
43         emp.setId(empID);
44         emp.setCreationDate(new Date());
45         return emp;
46     }
47 }
```

This screenshot shows the implementation of the `createEmployee` method from the previous code. It creates a new `Employee` object, initializes its contact numbers, sets its ID and creation date, and returns it.

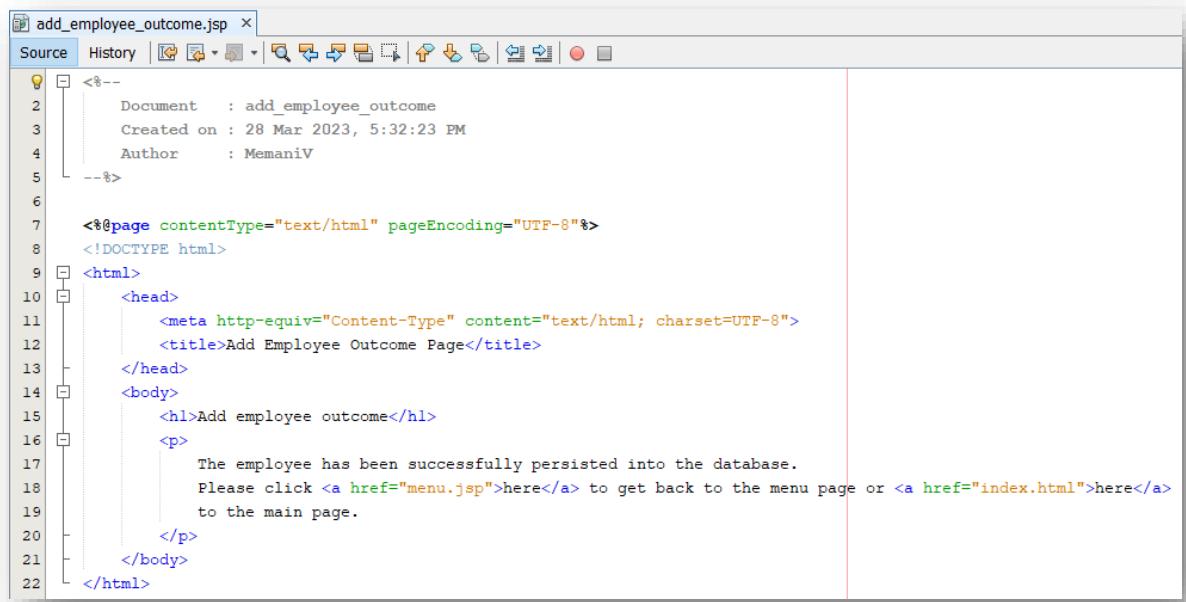
```
37     private Employee createEmployee(Long empID, String empCellNo, String nextKinCellNo) {
38         Employee emp = new Employee();
39         List<String> list = new ArrayList<>();
40         list.add(empCellNo);
41         list.add(nextKinCellNo);
42         emp.setContactNos(list);
43         emp.setId(empID);
44         emp.setCreationDate(new Date());
45         return emp;
46     }
47 }
```

Create the **FindEmployeeServlet.java** file.



```
1 package za.ac.tut.web;
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.EmployeeFacadeLocal;
11 import za.ac.tut.entities.Employee;
12
13 /**
14 *
15 * @author MemaniV
16 */
17 public class FindEmployeeServlet extends HttpServlet {
18     @EJB
19     private EmployeeFacadeLocal efl; //inject reference (instantiation)
20
21     @Override
22     protected void doPost(HttpServletRequest request, HttpServletResponse response)
23             throws ServletException, IOException {
24         Long empID = Long.parseLong(request.getParameter("empID"));
25
26         Employee employee = efl.find(empID); //find employee
27         request.setAttribute("employee", employee);
28
29         RequestDispatcher disp = request.getRequestDispatcher("find_employee_outcome.jsp");
30         disp.forward(request, response);
31     }
32
33 }
```

Create the **add_employee_outcome.jsp** file.



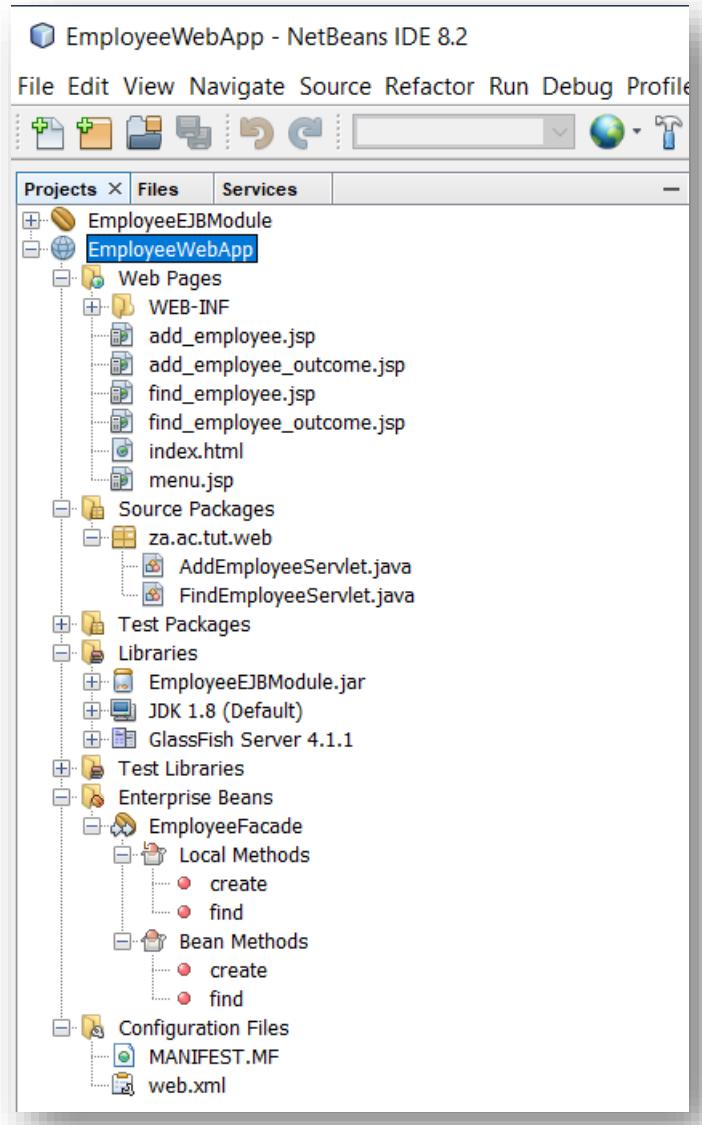
```
1 <%-- Document : add_employee_outcome
2 Created on : 28 Mar 2023, 5:32:23 PM
3 Author : MemaniV
4 --%>
5
6
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>
8 <!DOCTYPE html>
9 <html>
10    <head>
11        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12        <title>Add Employee Outcome Page</title>
13    </head>
14    <body>
15        <h1>Add employee outcome</h1>
16        <p>
17            The employee has been successfully persisted into the database.
18            Please click <a href="menu.jsp">here</a> to get back to the menu page or <a href="index.html">here</a>
19            to the main page.
20        </p>
21    </body>
22 </html>
```

Create the find_employee_outcome.jsp file.

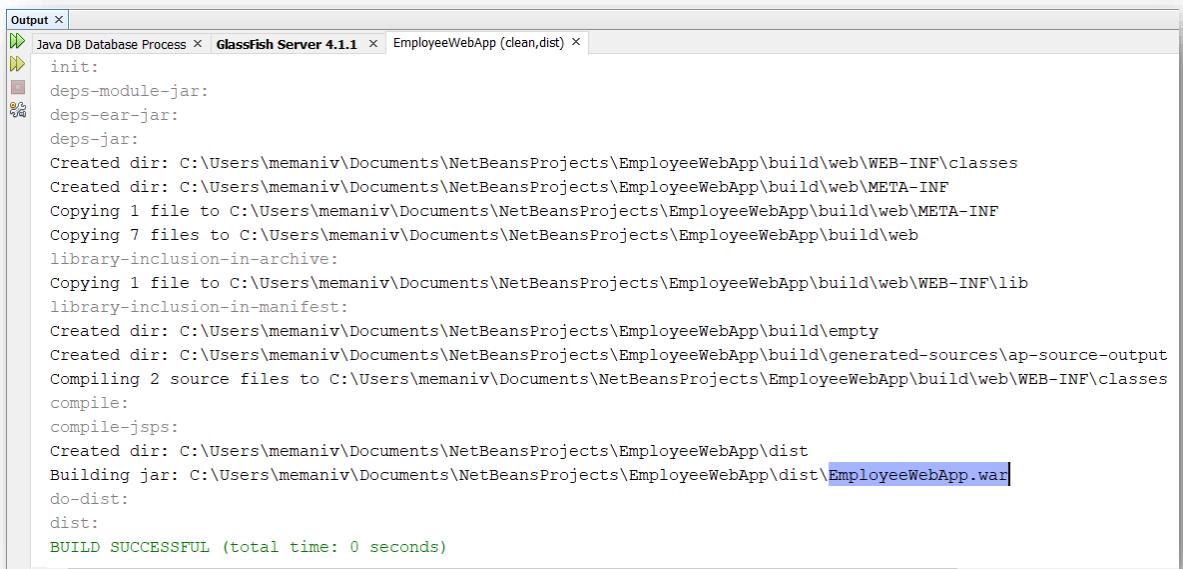
```
<%--  
1 Document      : find_employee_outcome  
2 Created on   : 28 Mar 2023, 5:34:14 PM  
3 Author        : MemaniV  
4--%>  
5  
6  
7 <%@page import="za.ac.tut.entities.Employee"%>  
8 <%@page contentType="text/html" pageEncoding="UTF-8"%>  
9 <!DOCTYPE html>  
10 <html>  
11 <head>  
12     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
13     <title>Find Employee Outcome Page</title>  
14 </head>  
15 <body>  
16     <h1>Find employee outcome</h1>  
17     <%  
18         Employee emp = (Employee)request.getAttribute("employee");  
19         Long empID = emp.getId();  
20         String empCellNo = emp.getContactNos().get(0);  
21         String nextKinCellNo = emp.getContactNos().get(1);  
22     %>  
23     <p>  
24         Below are the details of the employee.  
25     </p>
```

```
26 <table>  
27     <tr>  
28         <td><b>Employee ID:</b> </td>  
29         <td><%=empID%></td>  
30     </tr>  
31     <tr>  
32         <td><b>Cell number:</b> </td>  
33         <td><%=empCellNo%></td>  
34     </tr>  
35     <tr>  
36         <td><b>Next of kin cell number:</b> </td>  
37         <td><%=nextKinCellNo%></td>  
38     </tr>  
39 </table>  
40 <p>  
41     Please click <a href="menu.jsp">here</a> to get back to the menu  
42     page or <a href="index.html">here</a> to the main page.  
43 </p>  
44 </body>  
45 </html>
```

View the complete project structure of **EmployeeWebApp**.

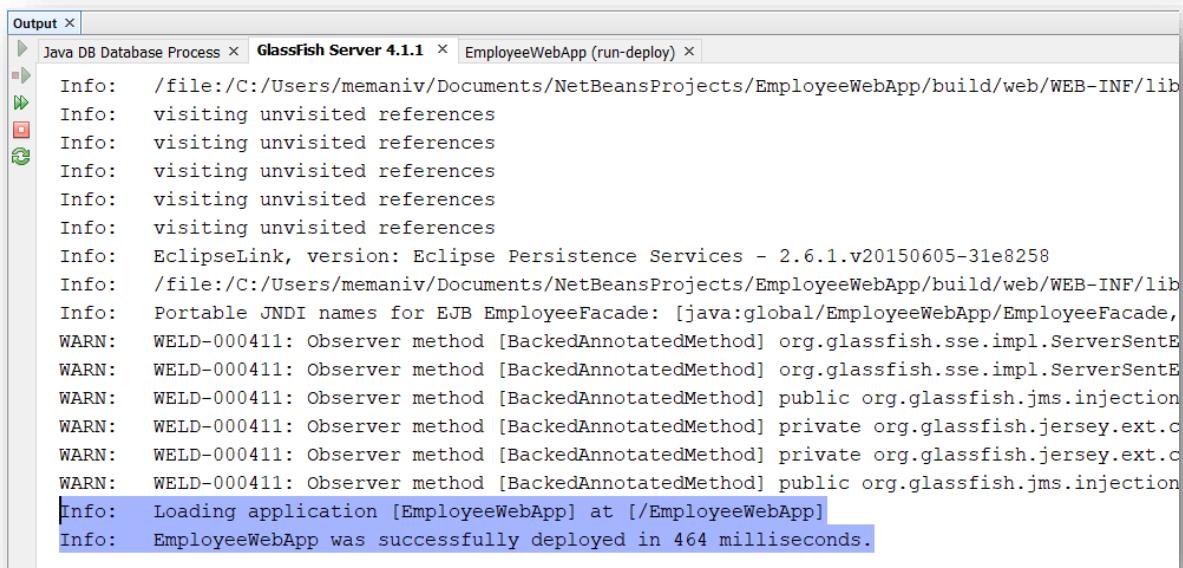


Compile the project.



```
Output ×
Java DB Database Process × GlassFish Server 4.1.1 × EmployeeWebApp (clean,dist) ×
init:
deps-module-jar:
deps-ear-jar:
deps-jar:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\EmployeeWebApp\build\web\WEB-INF\classes
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\EmployeeWebApp\build\web\META-INF
Copying 1 file to C:\Users\memaniv\Documents\NetBeansProjects\EmployeeWebApp\build\web\META-INF
Copying 7 files to C:\Users\memaniv\Documents\NetBeansProjects\EmployeeWebApp\build\web
library-inclusion-in-archive:
Copying 1 file to C:\Users\memaniv\Documents\NetBeansProjects\EmployeeWebApp\build\web\WEB-INF\lib
library-inclusion-in-manifest:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\EmployeeWebApp\build\empty
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\EmployeeWebApp\build\generated-sources\ap-source-output
Compiling 2 source files to C:\Users\memaniv\Documents\NetBeansProjects\EmployeeWebApp\build\web\WEB-INF\classes
compile:
compile-jsps:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\EmployeeWebApp\dist
Building jar: C:\Users\memaniv\Documents\NetBeansProjects\EmployeeWebApp\dist\EmployeeWebApp.war
do-dist:
dist:
BUILD SUCCESSFUL (total time: 0 seconds)
```

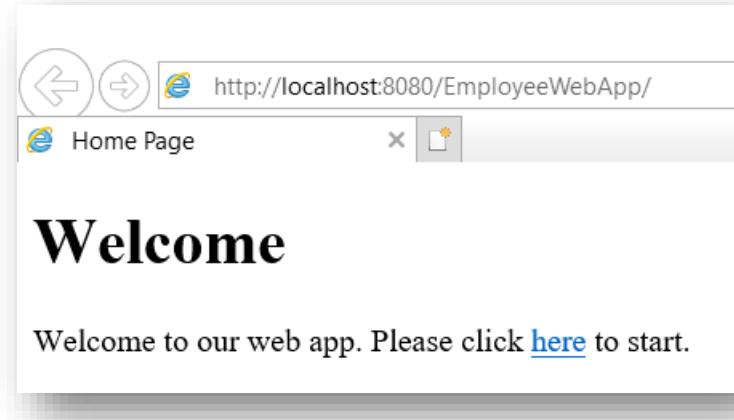
Deploy the project.



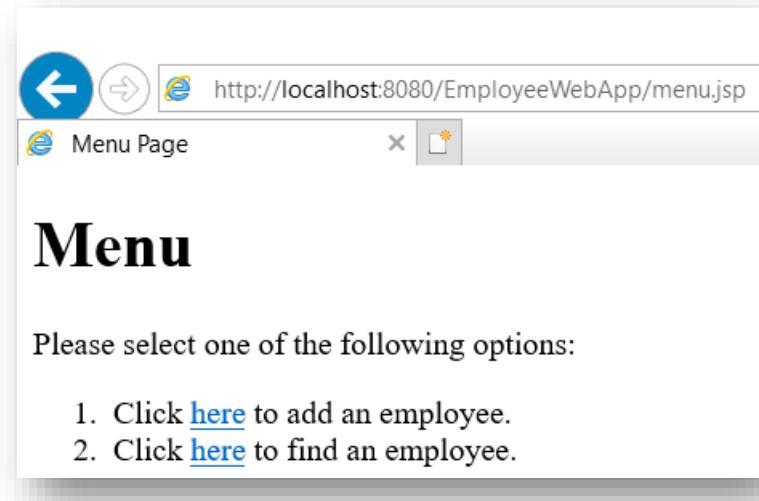
```
Output ×
Java DB Database Process × GlassFish Server 4.1.1 × EmployeeWebApp (run-deploy) ×
Info: /file:/C:/Users/memaniv/Documents/NetBeansProjects/EmployeeWebApp/build/web/WEB-INF/lib
Info: visiting unvisited references
Info: EclipseLink, version: Eclipse Persistence Services - 2.6.1.v20150605-31e8258
Info: /file:/C:/Users/memaniv/Documents/NetBeansProjects/EmployeeWebApp/build/web/WEB-INF/lib
Info: Portable JNDI names for EJB EmployeeFacade: [java:global/EmployeeWebApp/EmployeeFacade,
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] org.glassfish.sse.impl.ServerSentE
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] org.glassfish.sse.impl.ServerSentE
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] public org.glassfish.jms.injection
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] private org.glassfish.jersey.ext.c
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] private org.glassfish.jersey.ext.c
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] public org.glassfish.jms.injection
Info: Loading application [EmployeeWebApp] at [/EmployeeWebApp]
Info: EmployeeWebApp was successfully deployed in 464 milliseconds.
```

Part E – Run the web application.

Launch the application.



Click on the link



Add an employee.

- ✓ Click on the first link.

The screenshot shows a web browser window with the URL http://localhost:8080/EmployeeWebApp/add_employee.jsp. The title bar says "Add Employee Page". The main content area has a large heading "Add employee" and a sub-instruction "Please add employee details below:". Below this are three input fields: "Employee ID:", "Cell number:", and "Next of kin cell number:", each with a corresponding empty text box. At the bottom is a "ADD EMPLOYEE" button.

- ✓ Fill in the form.

The screenshot shows the same web browser window as the previous one, but now with filled form fields. The "Employee ID:" field contains "111", the "Cell number:" field contains "01234", and the "Next of kin cell number:" field contains "56789". The "ADD EMPLOYEE" button is still at the bottom.

- ✓ Click on the add button.

Add employee outcome

The employee has been successfully persisted into the database. Please click [here](#) to get back to the menu page or [here](#) to the main page.

- ✓ Go back to the menu page.

Menu

Please select one of the following options:

1. Click [here](#) to add an employee.
2. Click [here](#) to find an employee.

- ✓ Add four more employees into the database. View the records in the database.

#	ID	CREATIONDATE
1	111	2023-03-28 17:53:05.967
2	222	2023-03-28 17:55:50.501
3	333	2023-03-28 17:56:10.846
4	444	2023-03-28 17:56:28.007
5	555	2023-03-28 17:56:45.662

#	EMPLOYEE_ID	CONTACT
1	111	01234
2	111	56789
3	222	12456
4	222	23478
5	333	23900
6	333	11156
7	444	56832
8	444	61230
9	555	10297
10	555	20296

Search for an employee.

- ✓ Click on the second link.

Please enter the ID of the employee to search for.

Employee ID:

FIND

- ✓ Enter ID number **333**.

Please enter the ID of the employee to search for.

Employee ID:

FIND

- ✓ Click on the button.

Below are the details of the employee.

Employee ID: 333
Cell number: 23900
Next of kin cell number: 11156

Please click [here](#) to get back to the menu page or [here](#) to the main page.

8.3.4 Using basic annotations and maps

Example

Create a web application that will perform part of the **CRUD** (Create Read Update Delete) operations on a list of subjects. Specifically we want the application to be able to create and read the list. A subject has a **code** and a name. The table below shows all the fields of an employee and the applicable constraints to each field.

Field	Description	Constraint
id	The primary key.	Primary key. Automatically generated by the persistence provider.
subjectCode	The code of a subject.	This must be stored in as a key . So the Map will be Map< subjectCode , subjectName>.
subjectName	The name of a subject.	This must be stored in as a value . So the Map will be Map<subjectCode, subjectName >.
creationDate	The timestamp when subject details were persisted in a database.	Must be a Date.

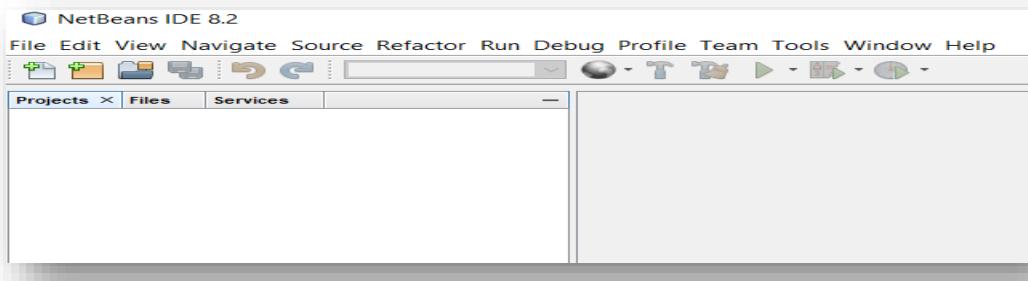
Solution approach

The solution to this problem is going to be done in five parts, namely:

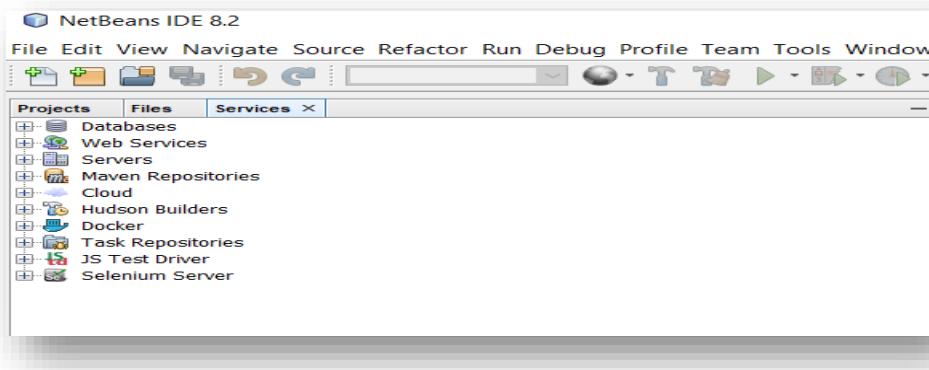
- **Part A:** we create a database.
- **Part B:** we establish connection between the created database and GlassFish.
- **Part C:** we create an EJB module. The module will have an entity with business logic.
- **Part D:** we create a client web application.
- **Part E:** we run the client web application.

Part A - Create a database

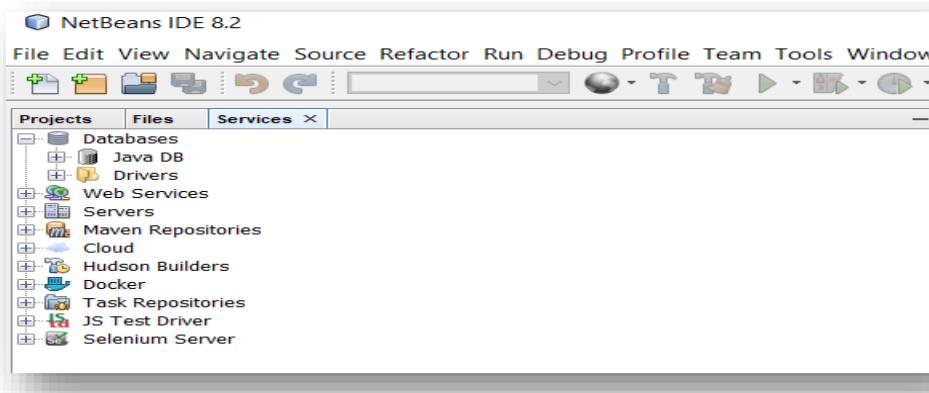
Launch NetBeans.



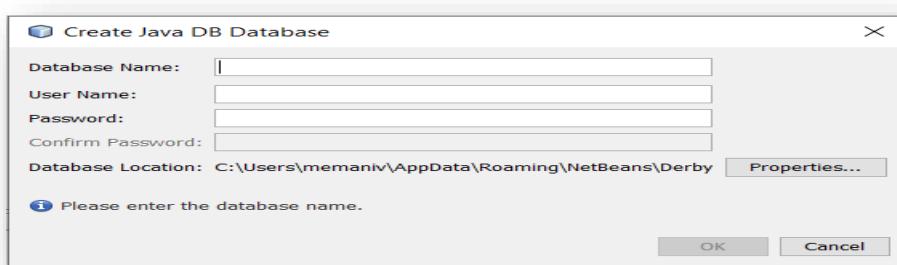
Click on the **Services** tab.



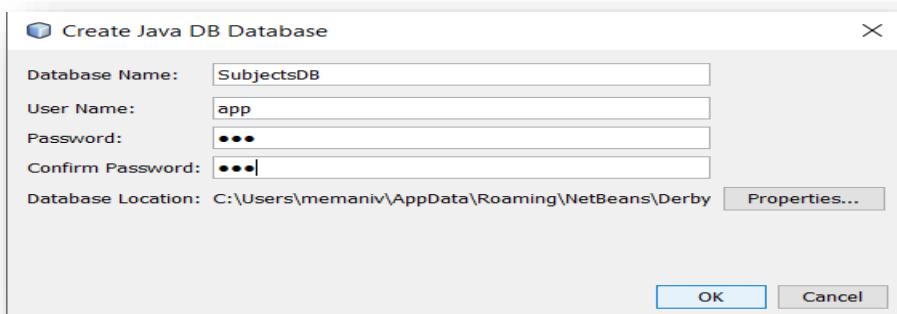
Expand the **Databases** option.



Right-click on **Java DB** and select **Create Database**.

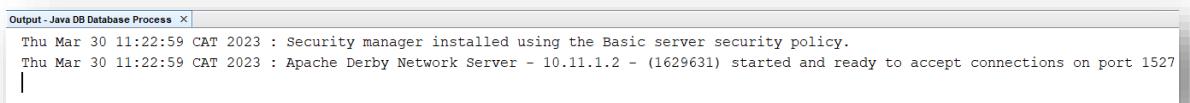


Fill-in the form. I made the password to be **123**.

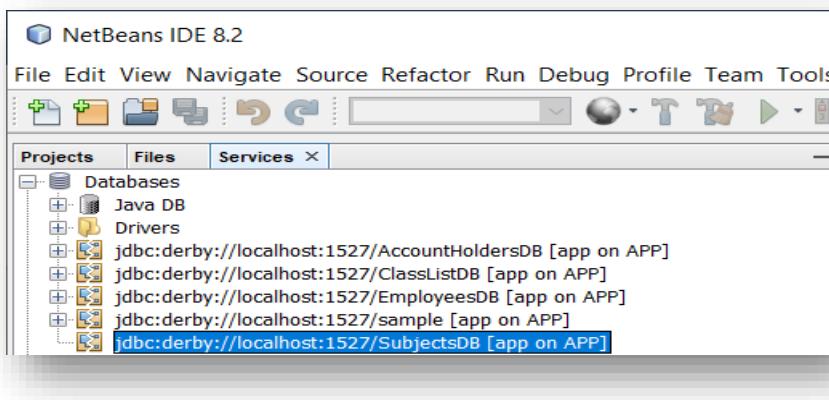


Click **OK**.

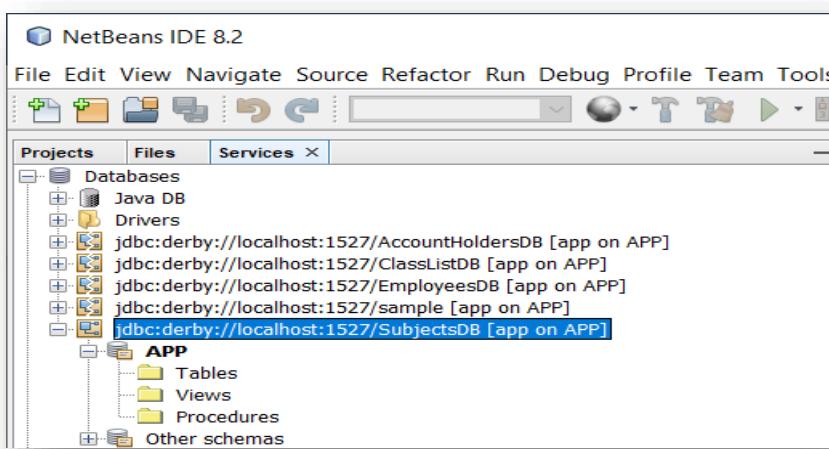
- ✓ An output message confirming that **Derby** has started and ready to accept connections on port 1527 is shown.



- ✓ A database is created.

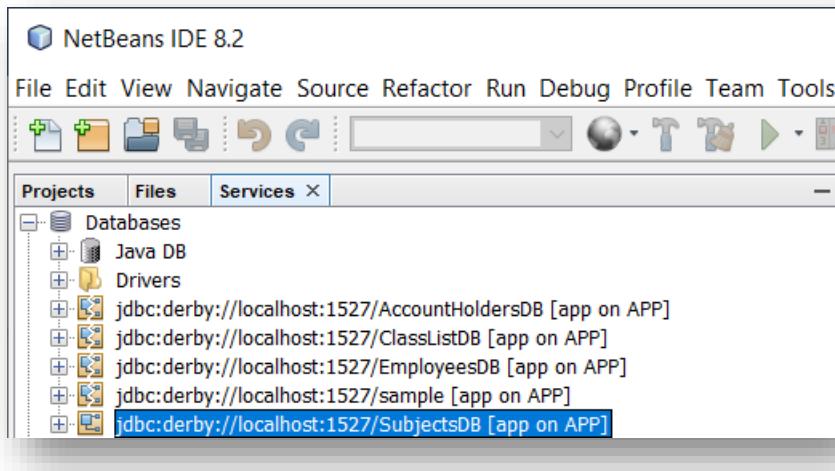


Right-click on the database and select **Connect**.

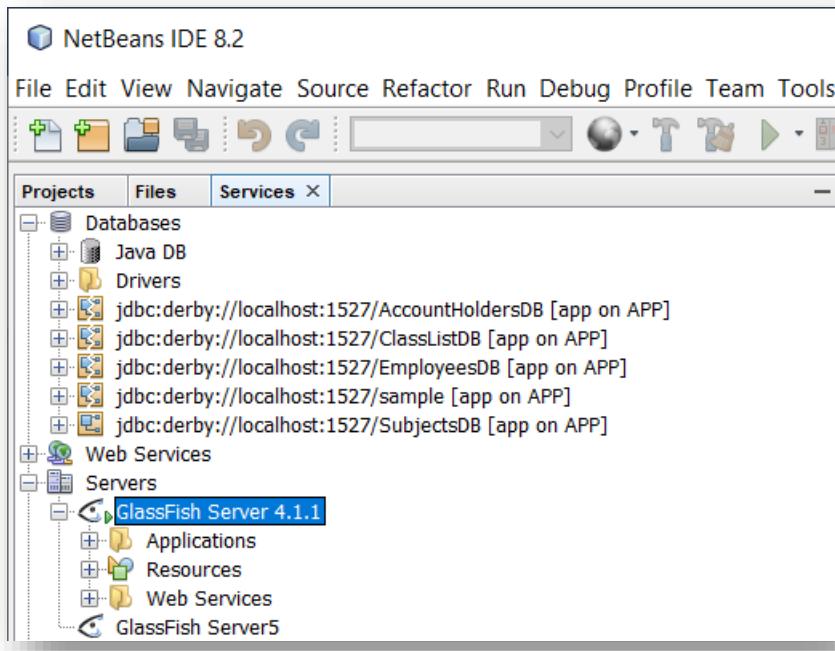


Part B – Connect the database to the application server.

Click on the **Services** tab.



Start the server.



Right-click on the server and select **View Domain Admin Console**.

The screenshot shows the GlassFish Server Open Source Edition Common Tasks console. The URL in the address bar is <http://localhost:4848/common/index.jsf>. The top navigation bar includes links for Home and About..., and displays the user as admin, domain as domain1, and server as localhost. The main title is "GlassFish™ Server Open Source Edition". The left sidebar, titled "Common Tasks", contains a tree view of server components: Domain, server (Admin Server), Clusters, Standalone Instances, Nodes, Applications, Lifecycle Modules, Monitoring Data, Resources (with sub-options like Concurrent Resources, Connectors, JDBC, JMS Resources, JNDI, JavaMail Sessions, and Resource Adapter Configs), Configurations (with sub-options for default-config and server-config), and an Update Tool. The right panel, titled "GlassFish Console - Common Tasks", is divided into sections: GlassFish News (with a "GlassFish News" link), Deployment (with "List Deployed Applications" and "Deploy an Application" links), Administration (with "Change Administrator Password" and "List Password Aliases" links), and Monitoring (with a "Monitoring Data" link).

Under **Common Tasks** panel, expand **JDBC**.

The screenshot shows the GlassFish Server Open Source Edition JDBC console. The URL in the address bar is <http://localhost:4848/common/index.jsf>. The top navigation bar includes links for Home and About..., and displays the user as admin, domain as domain1, and server as localhost. The main title is "GlassFish™ Server Open Source Edition". The left sidebar, titled "Common Tasks", is identical to the one in the previous screenshot. The right panel, titled "JDBC", contains two main categories: "JDBC Resources" and "JDBC Connection Pools". The "Resources" section in the sidebar is currently expanded, showing options like Concurrent Resources, Connectors, JDBC (which is selected and highlighted in blue), JMS Resources, JNDI, JavaMail Sessions, and Resource Adapter Configs.

Modify the Connection pool to include the **ClassListDB**.

- ✓ Click on **JDBC Connection Pools**.

JDBC Connection Pools					
To store, organize, and retrieve data, most applications use relational databases. Java EE applications access relational databases through the JDBC API. Before an application can access a database, it must get a connection.					
Pools (3)					
Select	Pool Name	Resource Type	Classname	Description	
<input type="checkbox"/>	DerbyPool	javax.sql.DataSource	org.apache.derby.jdbc.ClientDataSource		
<input type="checkbox"/>	SamplePool	javax.sql.DataSource	org.apache.derby.jdbc.ClientDataSource		
<input type="checkbox"/>	TimerPool	javax.sql.XADataSource	org.apache.derby.jdbc.EmbeddedXADataSource		

- ✓ Click on the **DerbyPool** link.

General	Advanced	Additional Properties
Edit JDBC Connection Pool		
Modify an existing JDBC connection pool. A JDBC connection pool is a group of reusable connections for a particular database.		
Load Defaults Flush Ping		
General Settings		
Pool Name:	DerbyPool	
Resource Type:	javax.sql.DataSource	
Must be specified if the datasource class implements more than 1 of the interface.		
Datasource Classname:	org.apache.derby.jdbc.ClientDataSource	
Vendor-specific classname that implements the DataSource and/or XADataSource APIs		
Driver Classname:		
Vendor-specific classname that implements the java.sql.Driver interface.		
Ping:	<input type="checkbox"/> Enabled	
When enabled, the pool is pinged during creation or reconfiguration to identify and warn of any erroneous values for its attributes		
Deployment Order:	100	
Specifies the loading order of the resource at server startup. Lower numbers are loaded first.		
Description:		
Pool Settings		
Initial and Minimum Pool Size:	8	Connections
Minimum and initial number of connections maintained in the pool		
Maximum Pool Size:	32	Connections
Maximum number of connections that can be created to satisfy client requests		
Pool Resize Quantity:	2	Connections

- ✓ Click on **Additional Properties**.

General	Advanced	Additional Properties
Edit JDBC Connection Pool Properties		
Modify properties of an existing JDBC connection pool.		
Pool Name: DerbyPool		
Additional Properties (6)		
Edit Delete Add Property Delete Properties		
Select	Name	Value
<input type="checkbox"/>	PortNumber	1527
<input type="checkbox"/>	Password	APP
<input type="checkbox"/>	User	APP
<input type="checkbox"/>	serverName	localhost
<input type="checkbox"/>	DatabaseName	sun-appserv-samples
<input type="checkbox"/>	connectionAttributes	,create=true

- ✓ Click on **Add Property** and add a **url** property with the following value:

jdbc:derby://localhost:1527/SubjectsDB

- ✓ Delete the **connectionAttributes** property.

Select	Name	Value
<input type="checkbox"/>	PortNumber	1527
<input type="checkbox"/>	Password	123
<input type="checkbox"/>	User	app
<input type="checkbox"/>	serverName	localhost
<input type="checkbox"/>	DatabaseName	SubjectsDB
<input type="checkbox"/>	url	jdbc:derby://localhost:1527/SubjectsDB

- ✓ Save the changes by clicking the **Save** button.

Select	Name	Value
<input type="checkbox"/>	PortNumber	1527
<input type="checkbox"/>	Password	123
<input type="checkbox"/>	User	app
<input type="checkbox"/>	serverName	localhost
<input type="checkbox"/>	DatabaseName	SubjectsDB
<input type="checkbox"/>	url	jdbc:derby://localhost:1527/SubjectsDB

- ✓ Check if the created connection pool is working. Do this by selecting the General tab and then click on the **Ping** button.

The screenshot shows the 'Edit JDBC Connection Pool' dialog with the 'General' tab selected. At the top right, a yellow button displays a green checkmark and the text 'Ping Succeeded'. Below this, the title 'Edit JDBC Connection Pool' is followed by a brief description: 'Modify an existing JDBC connection pool. A JDBC connection pool is a group of reusable connections for a particular database.' Three buttons are visible: 'Load Defaults', 'Flush', and 'Ping'. The main configuration area includes fields for 'Pool Name' (DerbyPool), 'Resource Type' (selected as 'javax.sql.DataSource'), 'Datasource Classname' (org.apache.derby.jdbc.ClientDataSource), 'Driver Classname' (left empty), 'Ping' (checkbox checked, labeled 'Enabled'), 'Deployment Order' (set to 100), and 'Description' (empty). The entire dialog has a light gray gradient background.

Confirm that the data source (resource) points to the DerbyPool.

- ✓ Still under **JDBC**, click on **JDBC Resources**.

The screenshot shows the 'JDBC Resources' list. The title 'JDBC Resources' is at the top, followed by a sub-section 'Resources (3)'. Below is a table with columns: Select, JNDI Name, Logical JNDI Name, Enabled, and Connection Pool. The data is as follows:

Select	JNDI Name	Logical JNDI Name	Enabled	Connection Pool
<input type="checkbox"/>	jdbc/_TimerPool		<input checked="" type="checkbox"/>	_TimerPool
<input type="checkbox"/>	jdbc/_default	java:comp/DefaultDataSource	<input checked="" type="checkbox"/>	DerbyPool
<input type="checkbox"/>	jdbc/sample		<input checked="" type="checkbox"/>	SamplePool

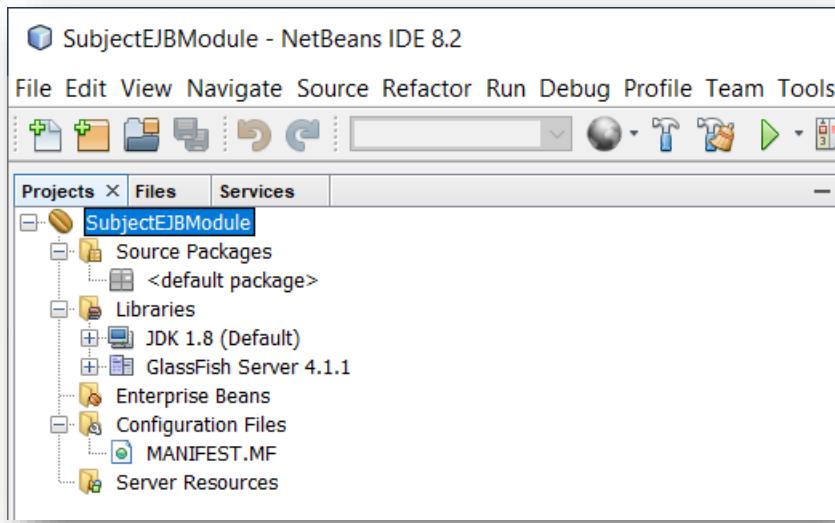
Take note of the purpose of JDBC Resources. It is said:

"JDBC resources provide applications with a means to connect to a database."

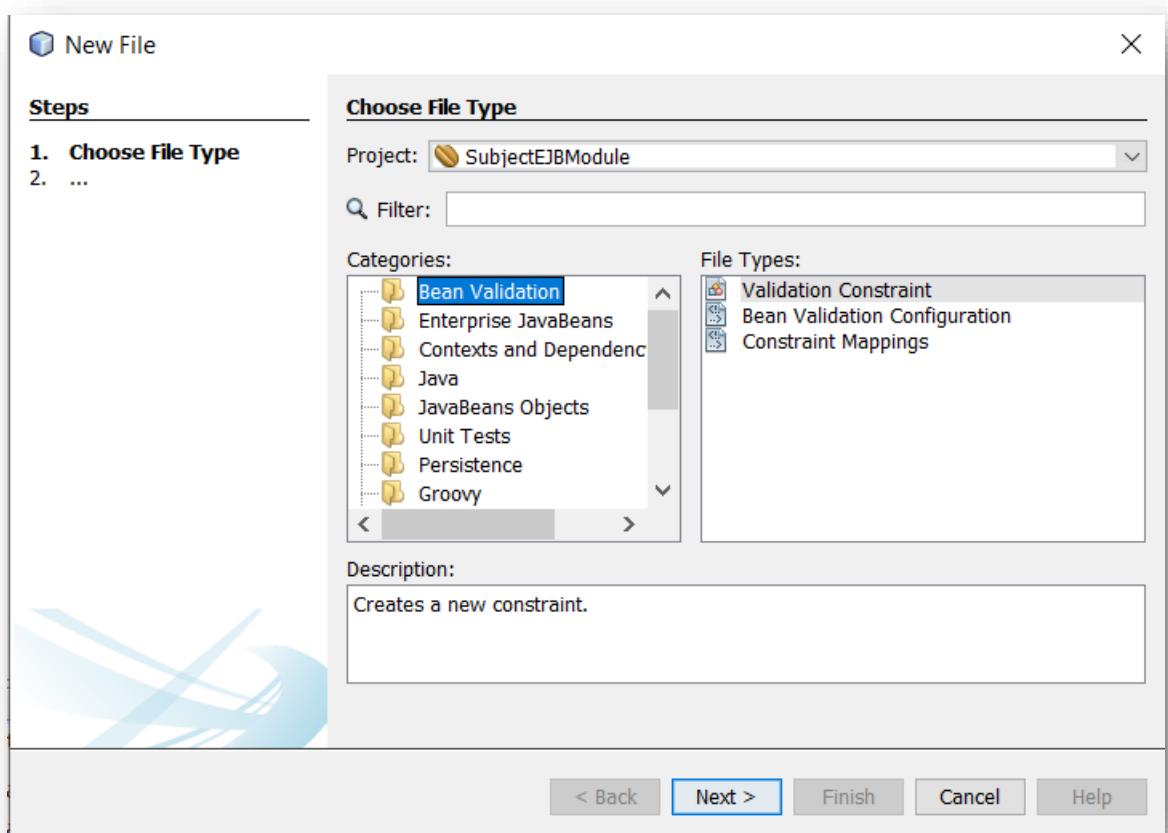
You can also see that the **jdbc/_default** resource is associated with the **DerbyPool** Connection Pool.

Part C – Create an entity.

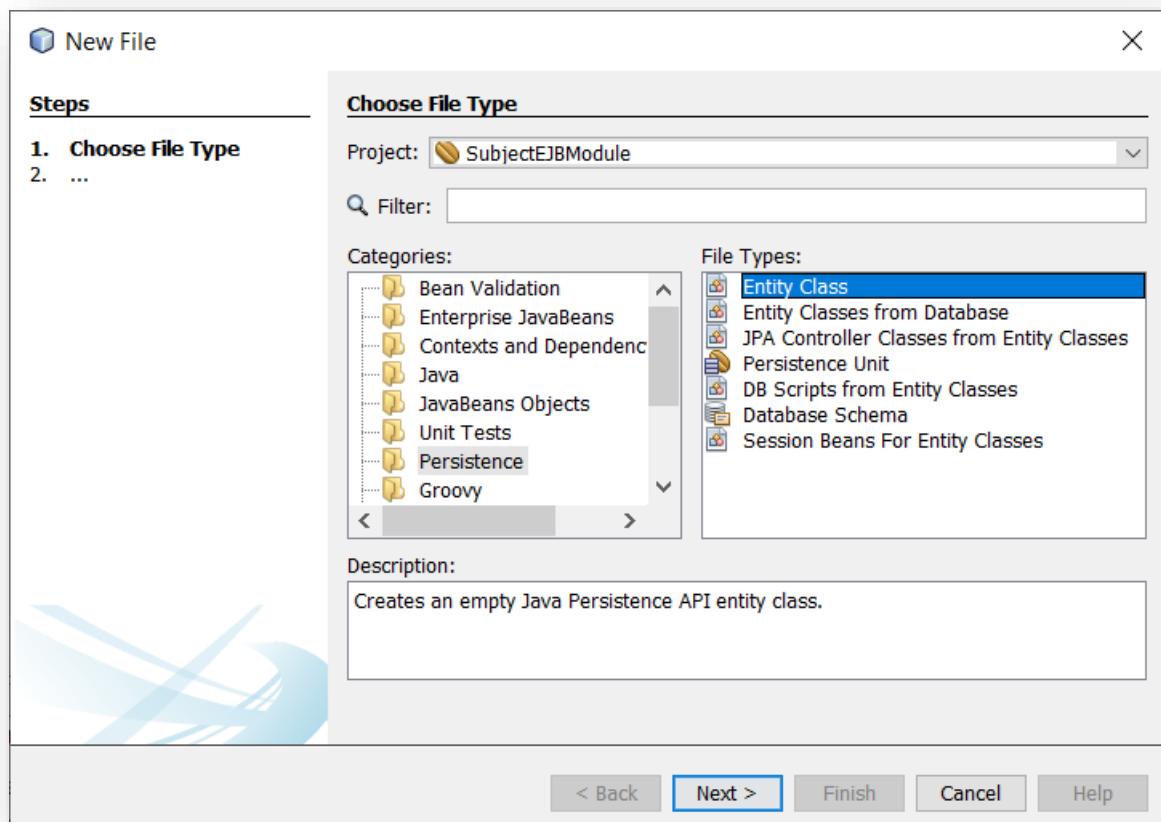
Create an EJB project called **SubjectEJBModule**.



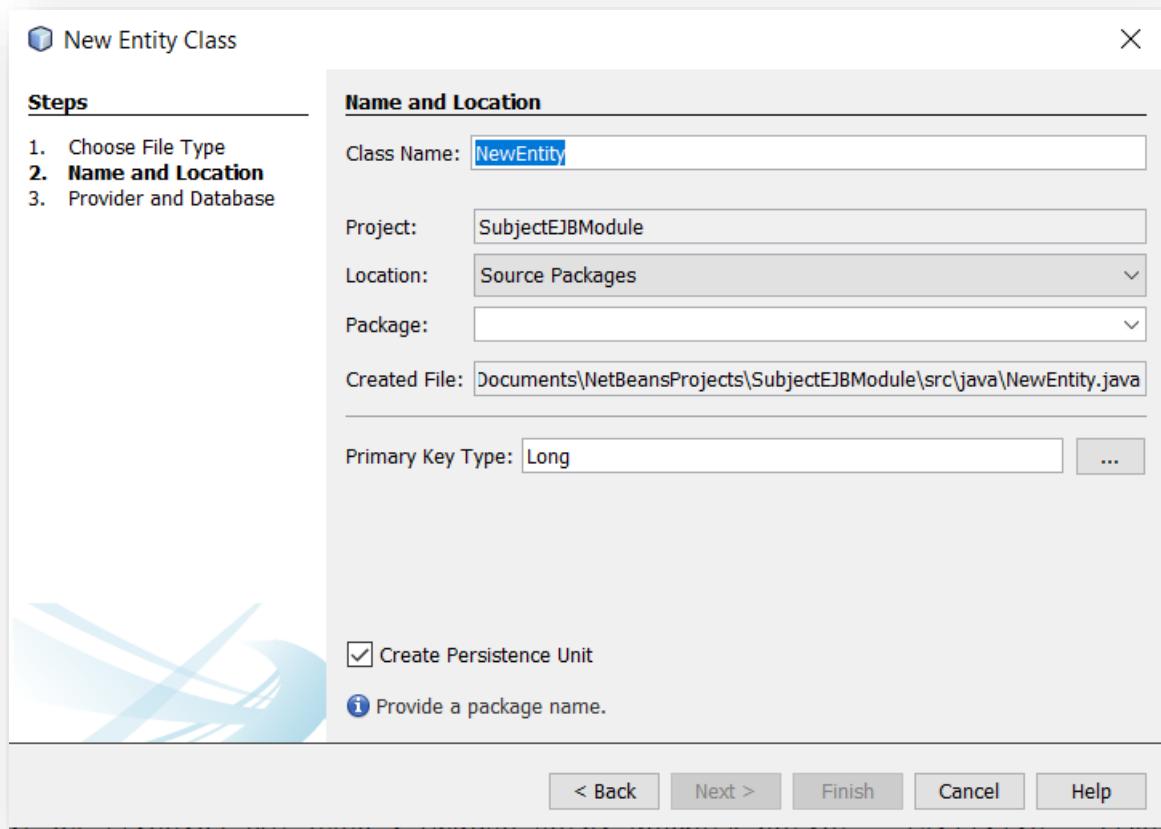
Right-click on the project and select **New | Other**.



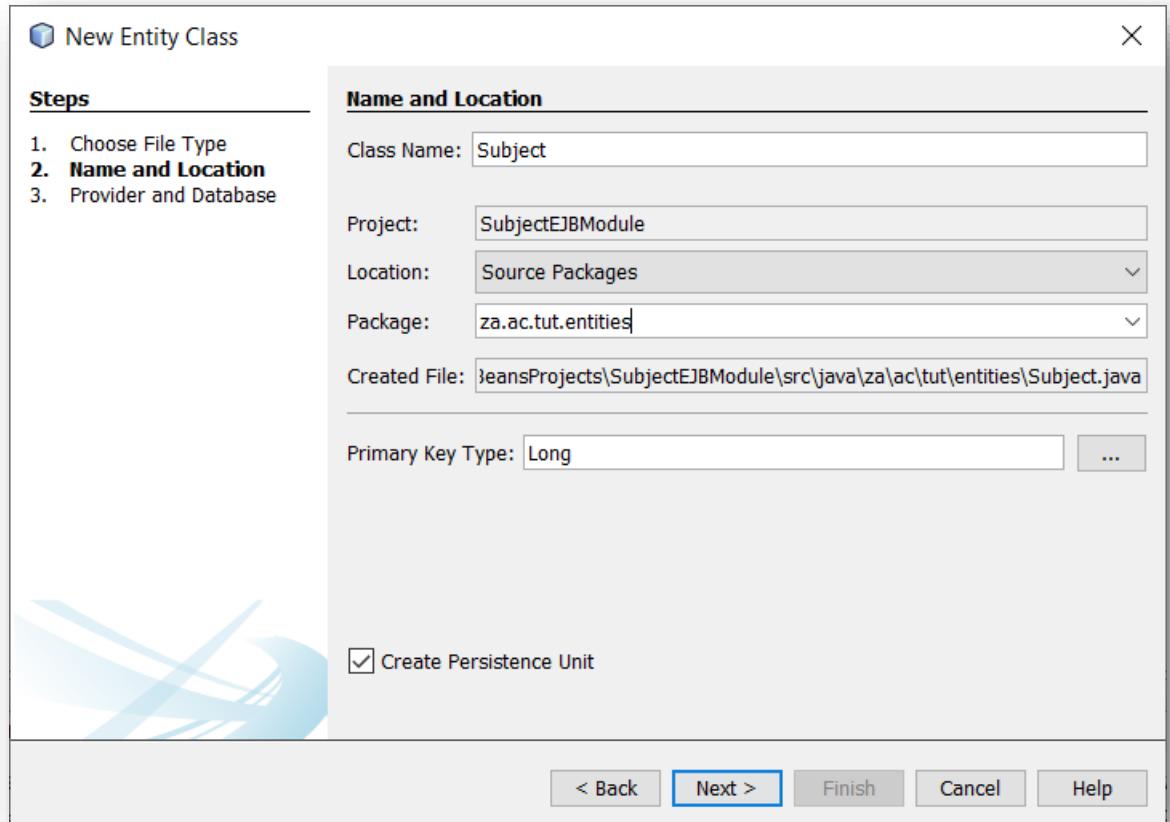
Under **Categories** select **Persistence**, and under **File Types** select **Entity Class**.



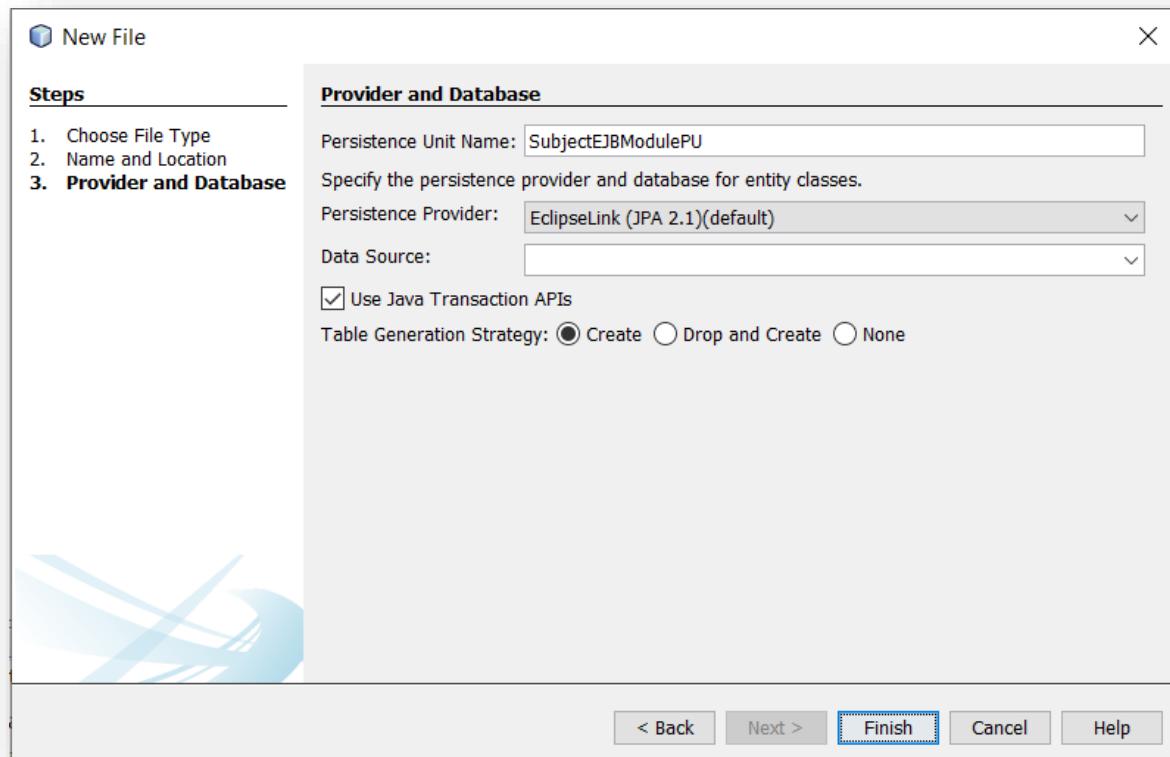
Click **Next**.



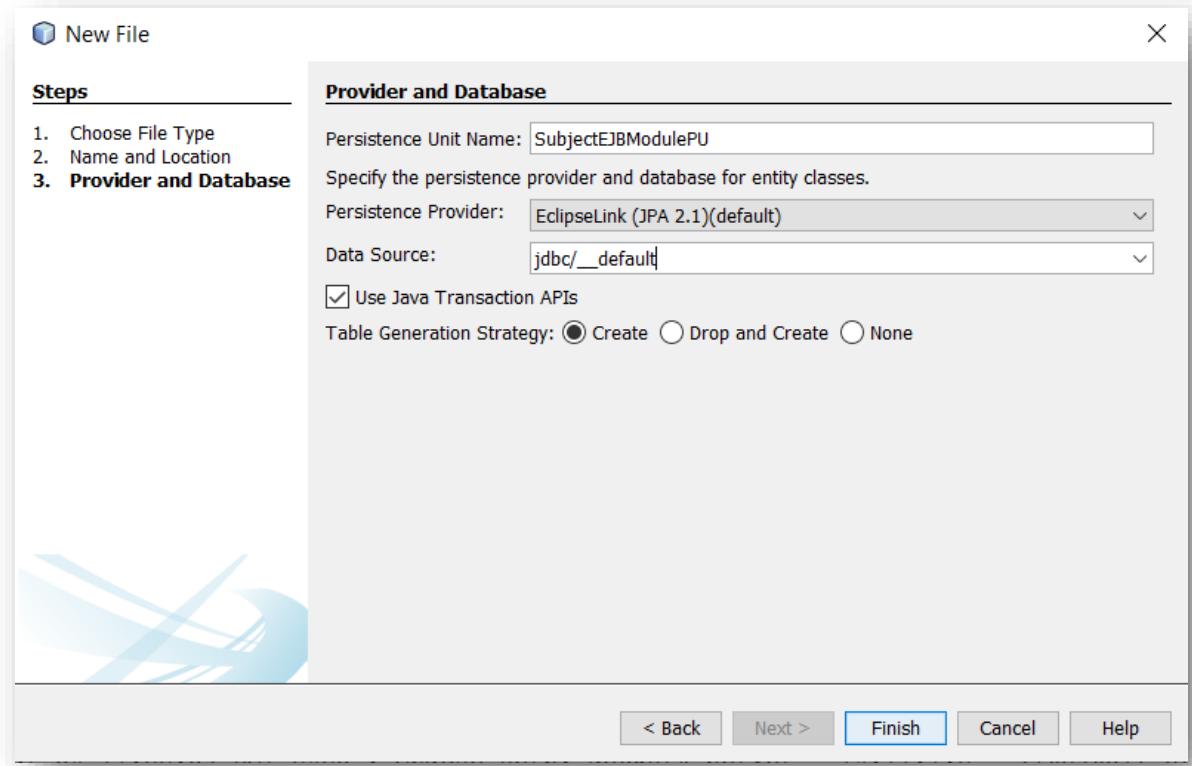
Fill-in the form.



Click **Next**.



Under **Data Source**, select **jdbc/_default**.



Click **Finish**.

The screenshot shows a Java code editor window titled "Subject.java". The code is annotated with several yellow highlights:

- Line 7: `import javax.persistence.ElementCollection;`
- Line 25: `@ElementCollection`
- Line 34: The closing brace of the `getMap()` method.
- Line 38: The opening brace of the `setMap(Map<String, String> map)` method.

```
1 package za.ac.tut.entities;
2
3 import java.io.Serializable;
4 import java.util.HashMap;
5 import java.util.Map;
6 import javax.persistence.CollectionTable;
7 import javax.persistence.ElementCollection;
8 import javax.persistence.Entity;
9 import javax.persistence.GeneratedValue;
10 import javax.persistence.GenerationType;
11 import javax.persistence.Id;
12 import javax.persistence.MapKeyColumn;
13
14 /**
15 *
16 * @author MemaniV
17 */
18 @Entity
19 public class Subject implements Serializable {
20
21     private static final long serialVersionUID = 1L;
22     @Id
23     @GeneratedValue(strategy = GenerationType.AUTO)
24     private Long id;
25     @ElementCollection
26     @CollectionTable(name="subject")
27     @MapKeyColumn(name="key")
28     private Map<String, String> map = new HashMap<>();
29
30     public Subject() {
31     }
32
33     public Map<String, String> getMap() {
34         return map;
35     }
36
37     public void setMap(Map<String, String> map) {
38         this.map = map;
39     }
}
```

```

41     public Long getId() {
42         return id;
43     }
44
45     public void setId(Long id) {
46         this.id = id;
47     }
48
49     @Override
50     public int hashCode() {
51         int hash = 0;
52         hash += (id != null ? id.hashCode() : 0);
53         return hash;
54     }
55
56     @Override
57     public boolean equals(Object object) {
58         if (!(object instanceof Subject)) {
59             return false;
60         }
61         Subject other = (Subject) object;
62         if ((this.id == null &amp; other.id != null) ||
63             (this.id != null &amp; !this.id.equals(other.id))) {
64             return false;
65         }
66         return true;
67     }
68
69     @Override
70     public String toString() {
71         return "za.ac.tut.entities.Subject[ id=" + id + " ]";
72     }
73
74 }

```

View the **persistence.xml** file.

```

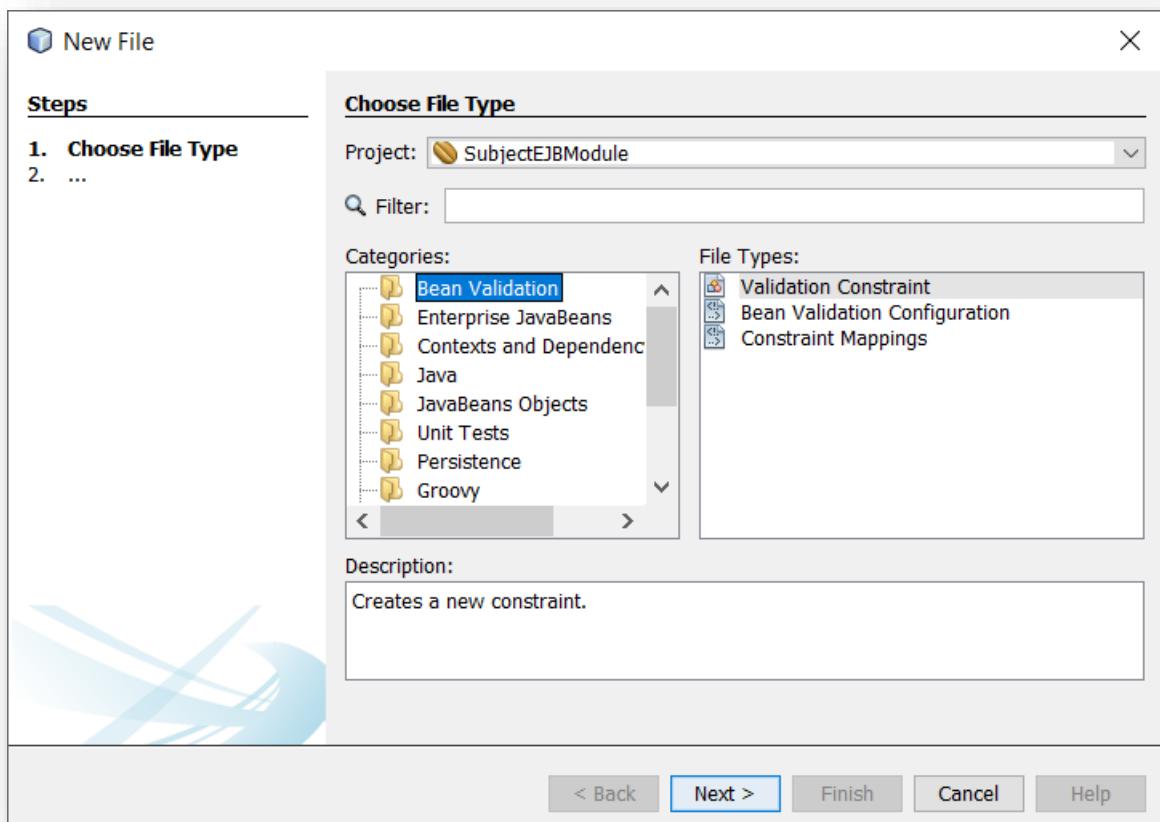
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
    <persistence-unit name="SubjectEJBModulePU" transaction-type="JTA">
        <jta-data-source>jdbc/__default</jta-data-source>
        <exclude-unlisted-classes>false</exclude-unlisted-classes>
        <properties>
            <property name="javax.persistence.schema-generation.database.action" value="create"/>
        </properties>
    </persistence-unit>
</persistence>

```

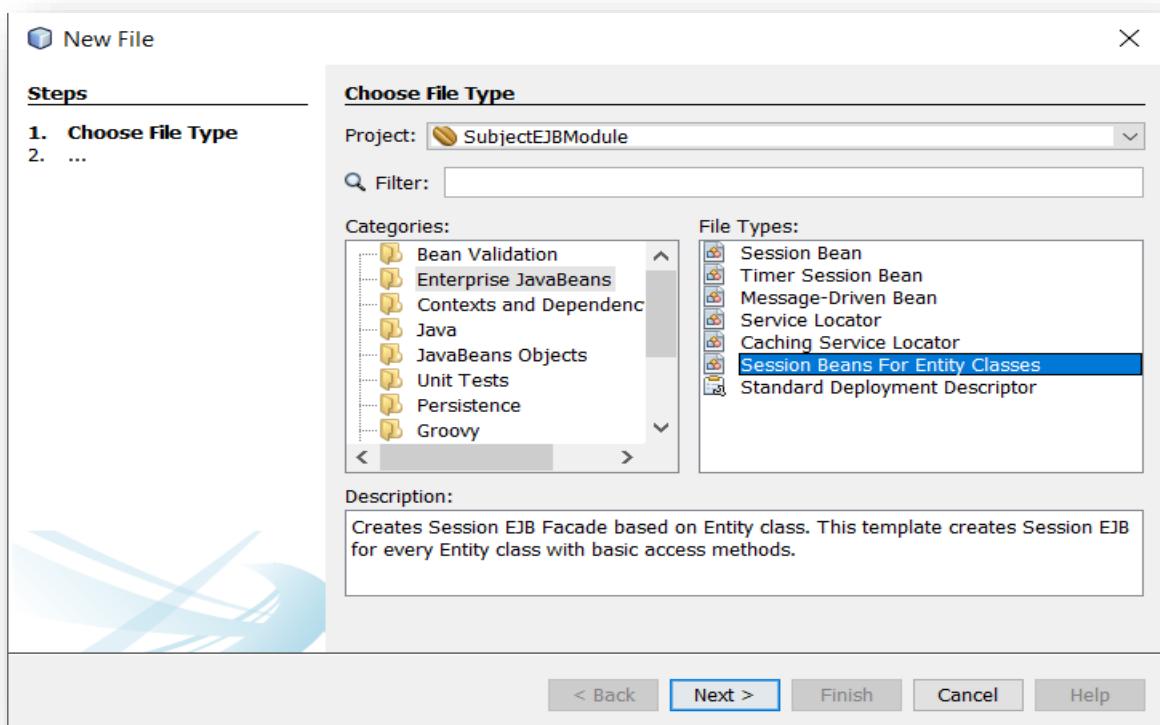
Create business code (Create employee and search for employee) for the entity.

Perform the following steps:

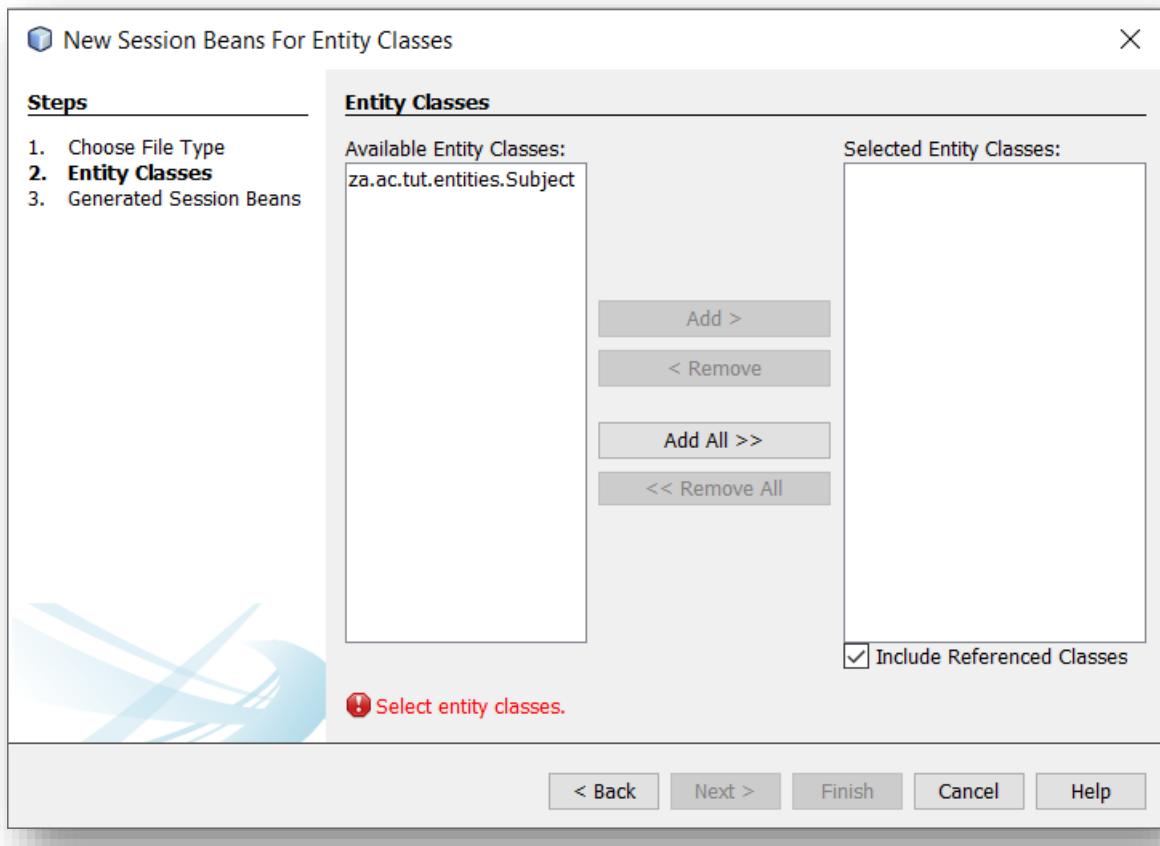
- ✓ Right-click on the project and select **New | Other**.



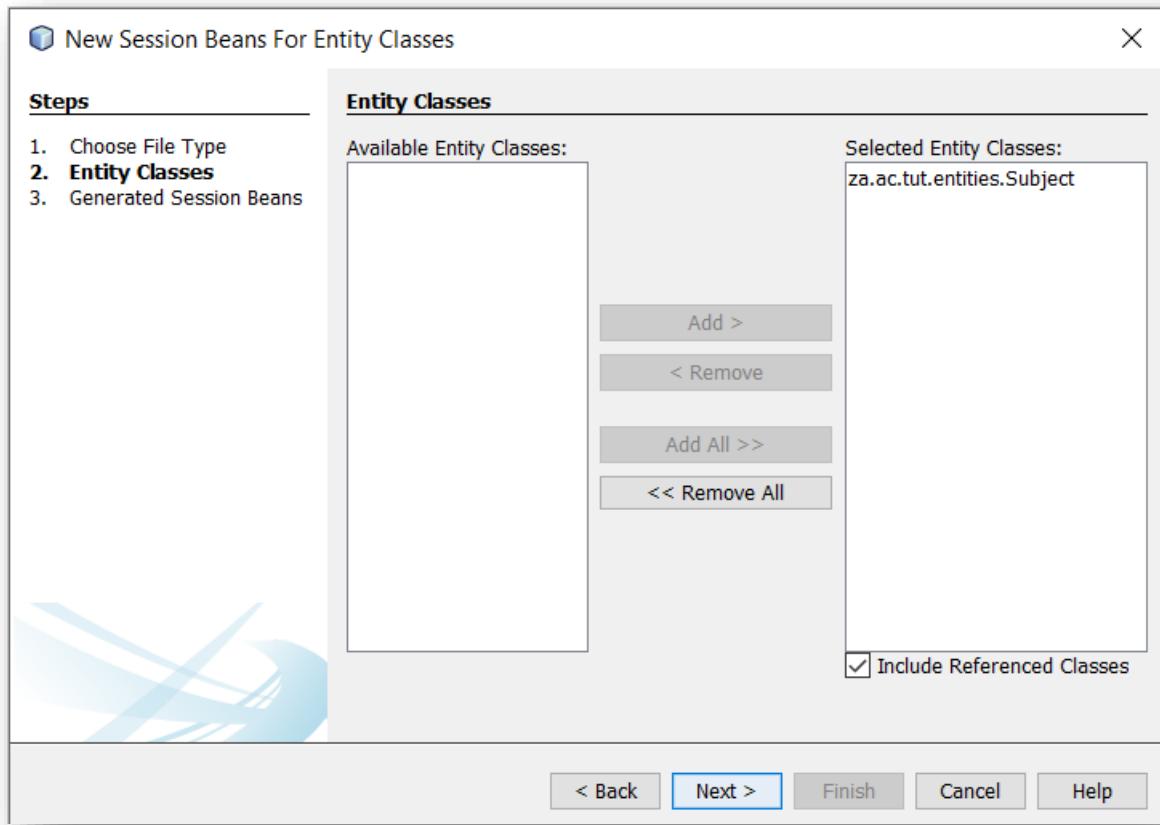
- ✓ Under **Categories**, select **Enterprise JavaBeans**. Under **File Types** select **Session Beans for Entity Classes**.



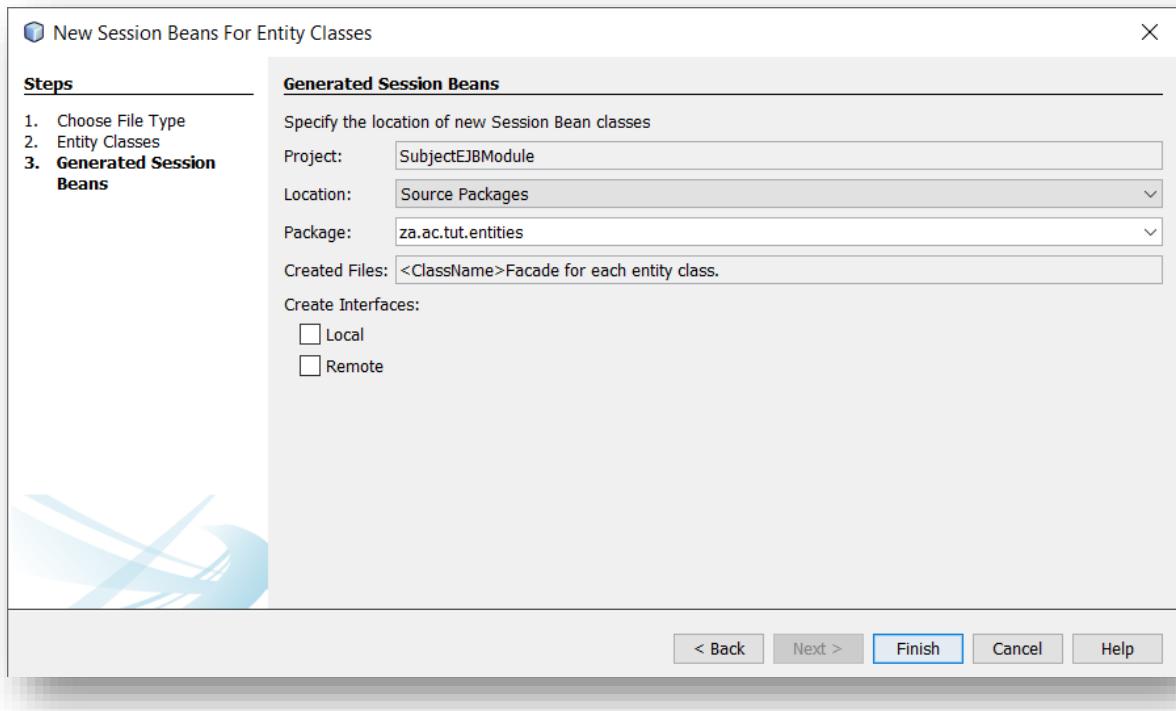
✓ Click **Next**.



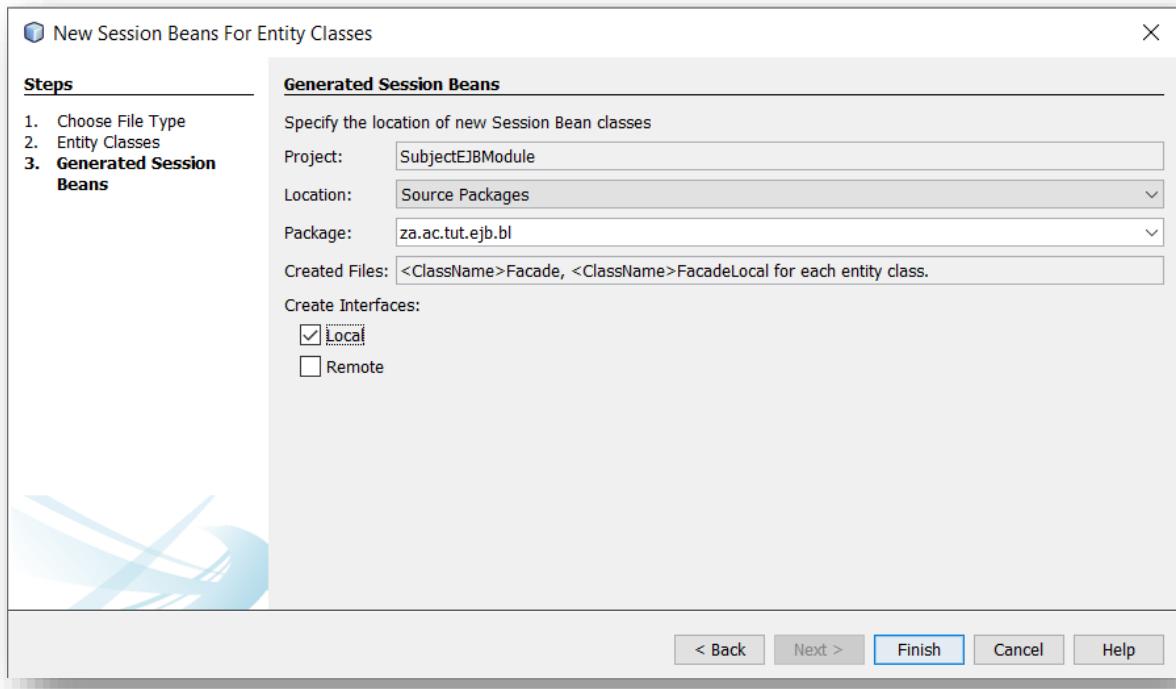
✓ Select **za.ac.tut.entities.Subject** under **Available Entity Classes** and click **Add**.



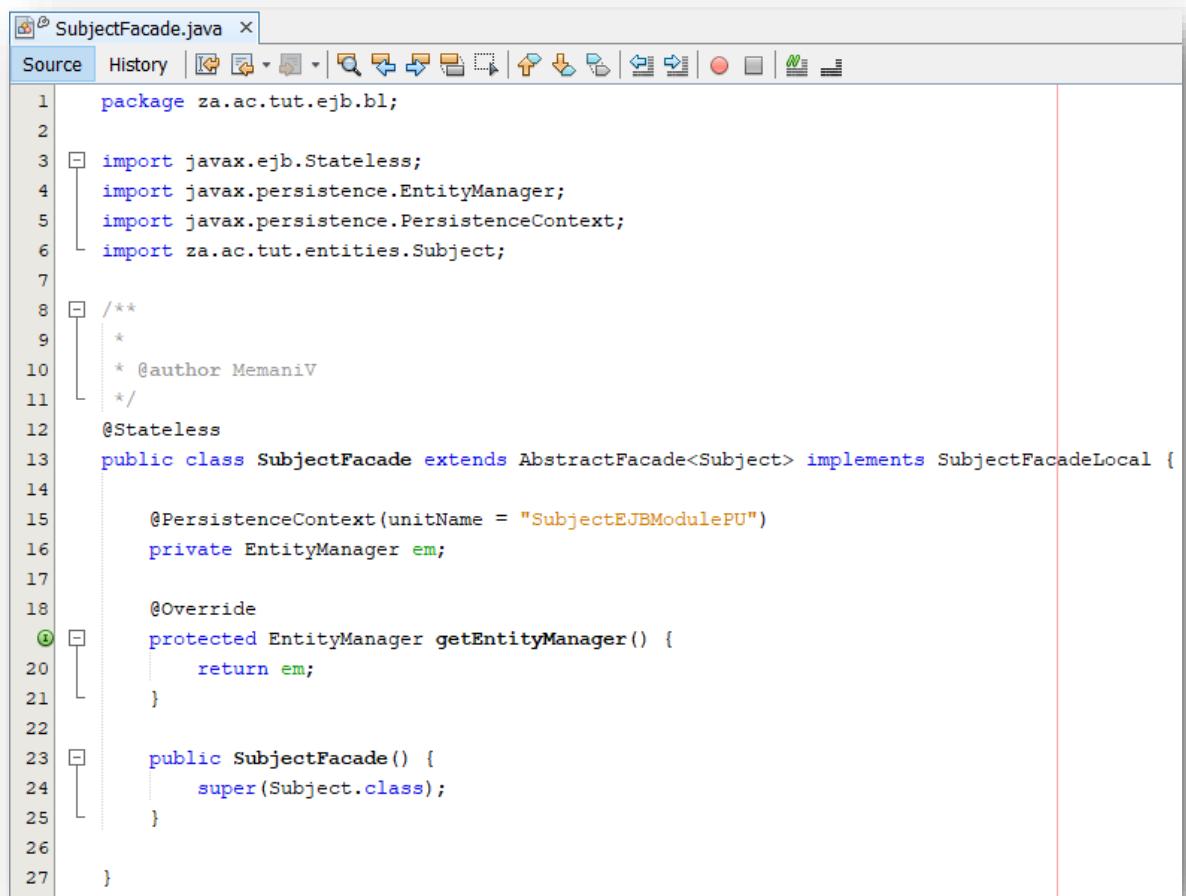
✓ Click **Next**.



✓ Modify the package name to **za.ac.tut.ejb.bl**, where **bl** stands for **business logic**. Also select **Local** interface.



- ✓ Click **Finish**. View created files.
 - **SubjectFacade**.



```
1 package za.ac.tut.ejb.bl;
2
3 import javax.ejb.Stateless;
4 import javax.persistence.EntityManager;
5 import javax.persistence.PersistenceContext;
6 import za.ac.tut.entities.Subject;
7
8 /**
9  * 
10 * @author MemaniV
11 */
12 @Stateless
13 public class SubjectFacade extends AbstractFacade<Subject> implements SubjectFacadeLocal {
14
15     @PersistenceContext(unitName = "SubjectEJBModulePU")
16     private EntityManager em;
17
18     @Override
19     protected EntityManager getEntityManager() {
20         return em;
21     }
22
23     public SubjectFacade() {
24         super(Subject.class);
25     }
26
27 }
```

▪ AbstractFacade.

```

1 package za.ac.tut.bl;
2
3 import java.util.List;
4 import javax.persistence.EntityManager;
5
6 /**
7 *
8 * @author MemaniV
9 */
10 public abstract class AbstractFacade<T> {
11
12     private Class<T> entityClass;
13
14     public AbstractFacade(Class<T> entityClass) {
15         this.entityClass = entityClass;
16     }
17
18     protected abstract EntityManager getEntityManager();
19
20     public void create(T entity) {
21         getEntityManager().persist(entity);
22     }
23
24     public void edit(T entity) {
25         getEntityManager().merge(entity);
26     }
27
28     public void remove(T entity) {
29         getEntityManager().remove(getEntityManager().merge(entity));
30     }
31

```

```

32     public T find(Object id) {
33         return getEntityManager().find(entityClass, id);
34     }
35
36     public List<T> findAll() {
37         javax.persistence.criteria.CriteriaQuery cq =
38             getEntityManager().getCriteriaBuilder().createQuery();
39         cq.select(cq.from(entityClass));
40         return getEntityManager().createQuery(cq).getResultList();
41     }
42
43     public List<T> findRange(int[] range) {
44         javax.persistence.criteria.CriteriaQuery cq =
45             getEntityManager().getCriteriaBuilder().createQuery();
46         cq.select(cq.from(entityClass));
47         javax.persistence.Query q = getEntityManager().createQuery(cq);
48         q.setMaxResults(range[1] - range[0] + 1);
49         q.setFirstResult(range[0]);
50         return q.getResultList();
51     }
52
53     public int count() {
54         javax.persistence.criteria.CriteriaQuery cq =
55             getEntityManager().getCriteriaBuilder().createQuery();
56         javax.persistence.criteria.Root<T> rt = cq.from(entityClass);
57         cq.select(getEntityManager().getCriteriaBuilder().count(rt));
58         javax.persistence.Query q = getEntityManager().createQuery(cq);
59         return ((Long) q.getSingleResult()).intValue();
60     }
61
62 }

```

- **SubjectFacadeLocal.**

The screenshot shows a Java code editor window titled "SubjectFacadeLocal.java". The code defines a local interface for managing subjects. It includes imports for java.util.List, javax.ejb.Local, and za.ac.tut.entities.Subject. The interface is annotated with @Local. It contains methods for creating, editing, removing, finding by ID, finding all, finding by range, and counting subjects.

```
1 package za.ac.tut.ejb.bl;
2
3 import java.util.List;
4 import javax.ejb.Local;
5 import za.ac.tut.entities.Subject;
6
7 /**
8 *
9 * @author MemaniV
10 */
11 @Local
12 public interface SubjectFacadeLocal {
13
14     void create(Subject subject);
15
16     void edit(Subject subject);
17
18     void remove(Subject subject);
19
20     Subject find(Object id);
21
22     List<Subject> findAll();
23
24     List<Subject> findRange(int[] range);
25
26     int count();
27
28 }
```

- ✓ Remove all the other interface methods except **create** and **findAll**.

The screenshot shows the same Java code editor window after modifications. The interface now only contains the create method and the findAll method, while the other methods have been removed.

```
1 package za.ac.tut.ejb.bl;
2
3 import java.util.List;
4 import javax.ejb.Local;
5 import za.ac.tut.entities.Subject;
6
7 /**
8 *
9 * @author MemaniV
10 */
11 @Local
12 public interface SubjectFacadeLocal {
13
14     void create(Subject subject);
15
16     List<Subject> findAll();
17
18 }
```

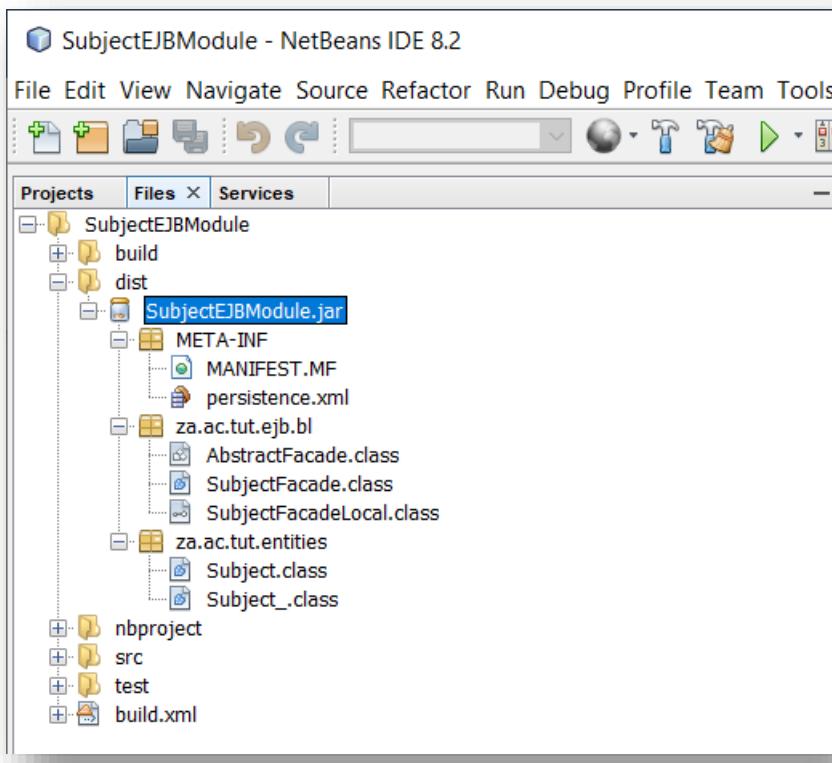
- ✓ Clean and build the EJB project.

Output X

```

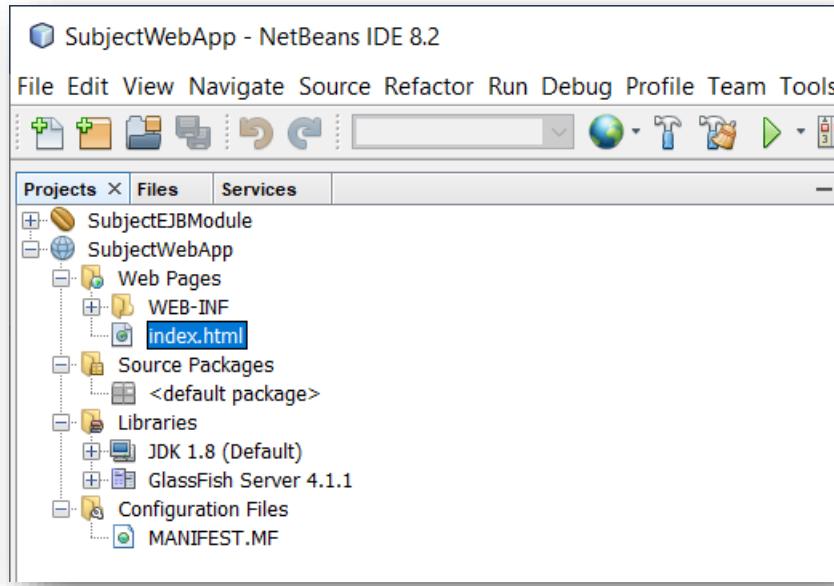
Java DB Database Process X GlassFish Server 4.1.1 x SubjectEJBModule (clean,dist) x
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\SubjectEJBModule\build\empty
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\SubjectEJBModule\build\generated-sources\ap-so
Compiling 4 source files to C:\Users\memaniv\Documents\NetBeansProjects\SubjectEJBModule\build\classes
warning: Supported source version 'RELEASE_6' from annotation processor 'org.eclipse.persistence.intern
Note: Creating static metadata factory ...
Note: Optional file was not found: META-INF/orm.xml continuing with generation.
Note: Optional file was not found: META-INF/eclipselink-orm.xml continuing with generation.
Note: Optional file was not found: META-INF/orm.xml continuing with generation.
Note: Optional file was not found: META-INF/eclipselink-orm.xml continuing with generation.
Note: C:\Users\memaniv\Documents\NetBeansProjects\SubjectEJBModule\src\java\za\ac\tut\ejb\bl\AbstractFa
Note: Recompile with -Xlint:unchecked for details.
1 warning
compile:
library-inclusion-in-archive:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\SubjectEJBModule\dist
Building jar: C:\Users\memaniv\Documents\NetBeansProjects\SubjectEJBModule\dist\SubjectEJBModule.jar
dist:
BUILD SUCCESSFUL (total time: 1 second)

```



Part D – Create a web client project.

Create a web client project called **SubjectWebApp**.

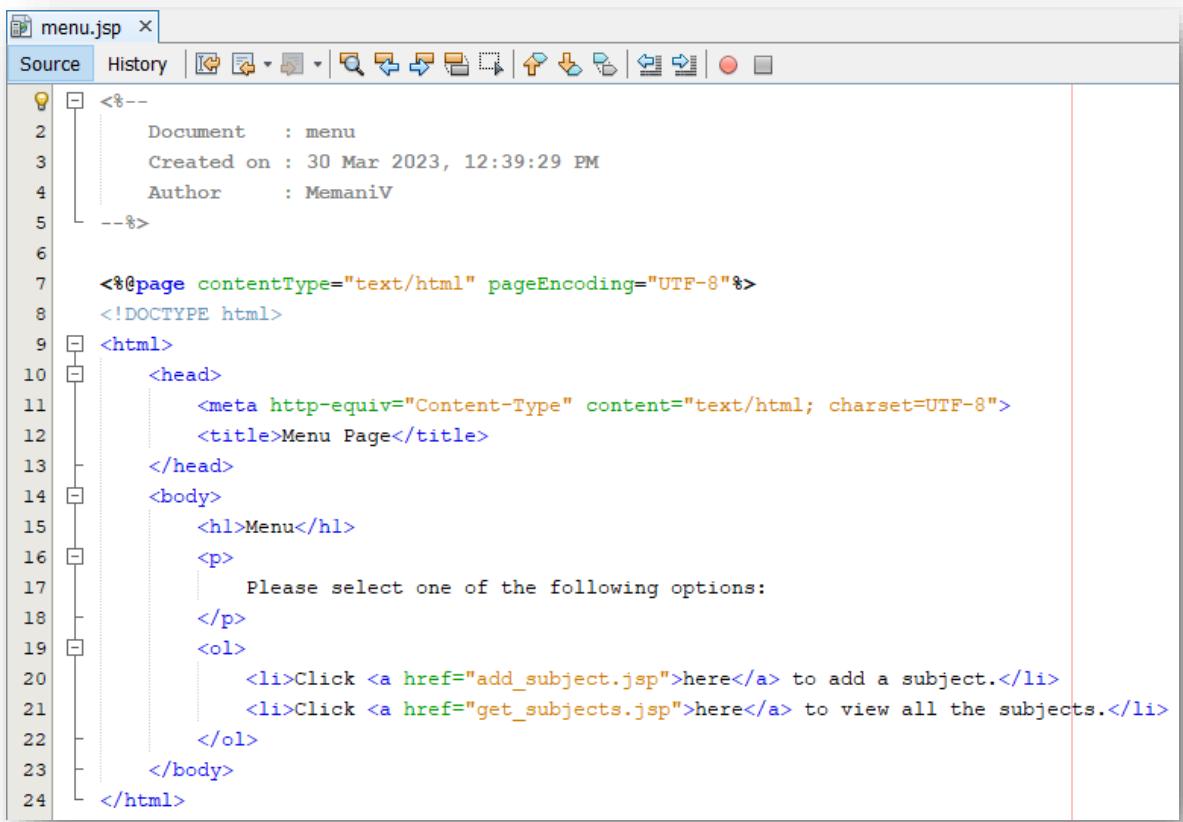


Edit the **index.html** page.

The screenshot shows a Java IDE interface with the title bar "index.html x". The main area is the "Source" tab, displaying the following HTML code:

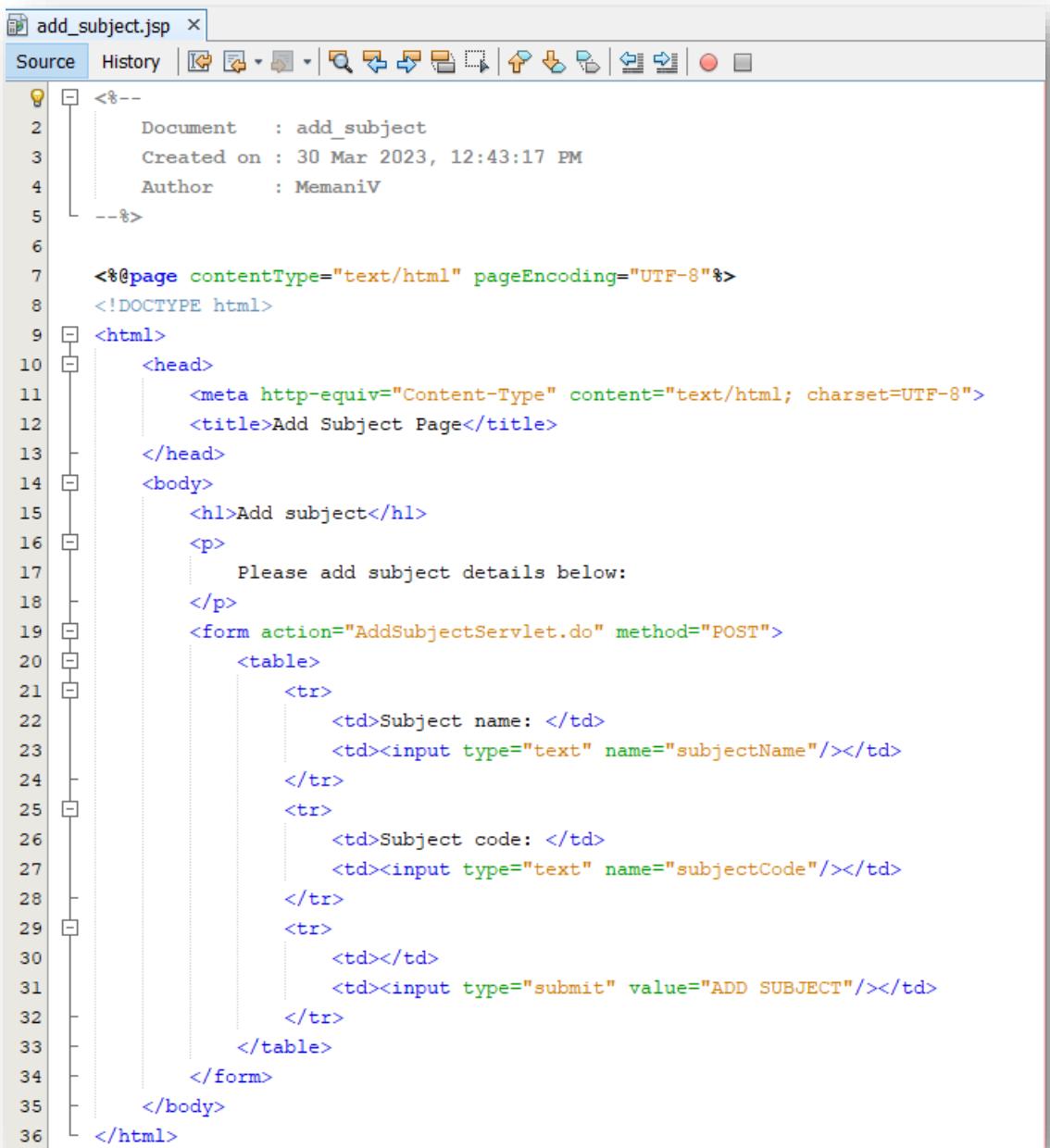
```
<!DOCTYPE html>
<!--
To change this license header, choose License Headers in Project Properties.
To change this template file, choose Tools | Templates
and open the template in the editor.
-->
<html>
    <head>
        <title>Home Page</title>
        <meta charset="UTF-8">
        <meta name="viewport"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

Create the menu.jsp file.



```
<%--  
1 Document : menu  
2 Created on : 30 Mar 2023, 12:39:29 PM  
3 Author : MemaniV  
4 --%>  
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="add_subject.jsp">here</a> to add a subject.</li>  
21         <li>Click <a href="get_subjects.jsp">here</a> to view all the subjects.</li>  
22     </ol>  
23 </body>  
24 </html>
```

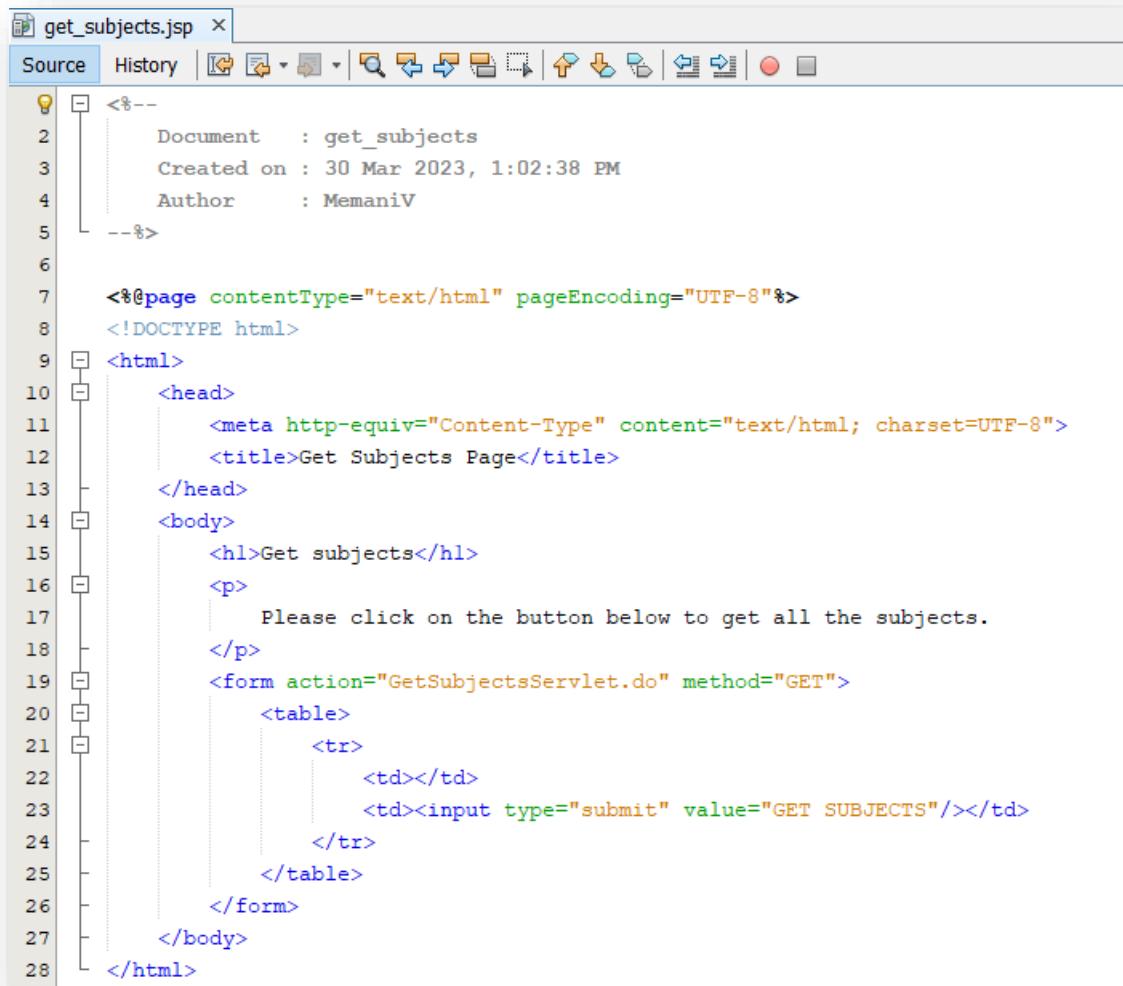
Create add_subject.jsp file.



The screenshot shows a Java IDE interface with the tab "add_subject.jsp" selected. The code editor displays the JSP page source code. The code includes a header block with document information, a head section with meta tags and a title, and a body section containing an H1 header, a paragraph instructing the user to add subject details, and a form for inputting subject name and code. The form uses a table with three rows and two columns each, and it includes a submit button.

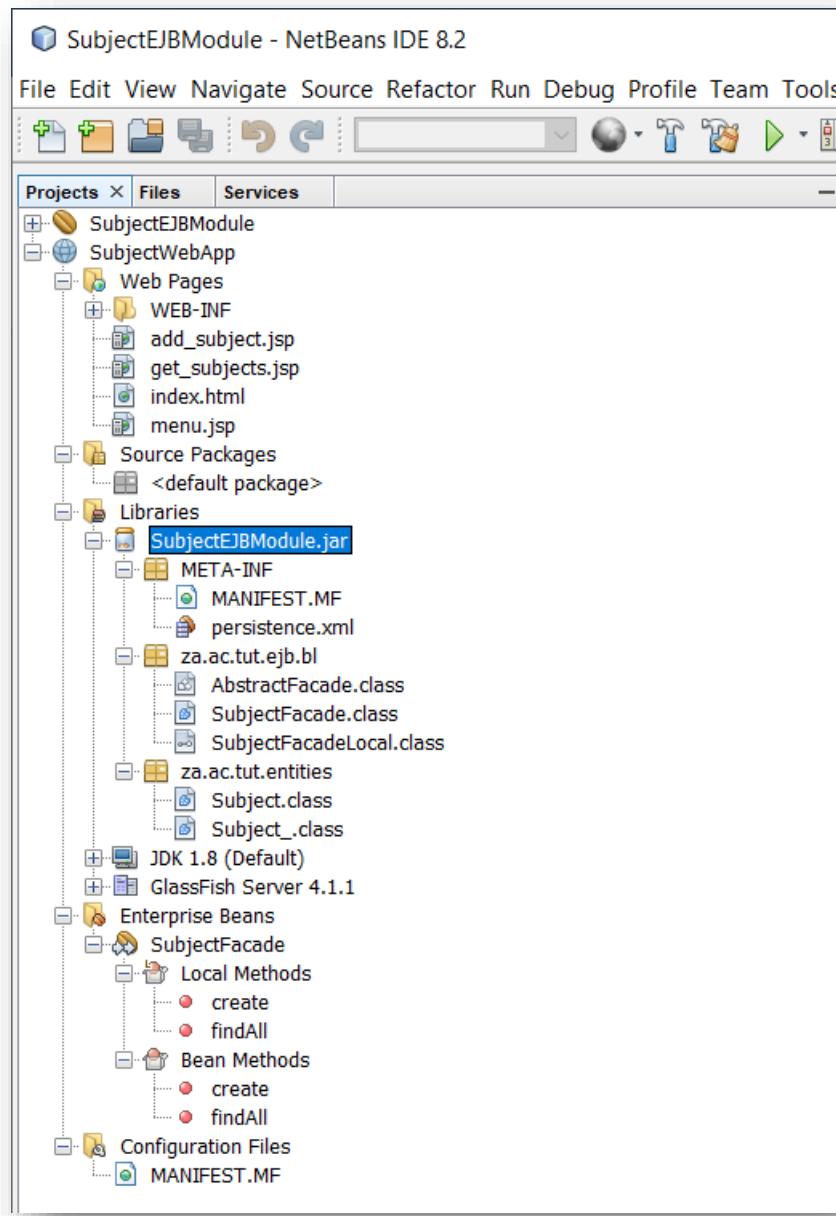
```
<%--  
2 Document : add_subject  
3 Created on : 30 Mar 2023, 12:43:17 PM  
4 Author : MemaniV  
5 --%>  
6  
<%@page contentType="text/html" pageEncoding="UTF-8"%>  
<!DOCTYPE html>  
<html>  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
<title>Add Subject Page</title>  
</head>  
<body>  
<h1>Add subject</h1>  
<p>  
Please add subject details below:  
</p>  
<form action="AddSubjectServlet.do" method="POST">  
<table>  
<tr>  
<td>Subject name:</td>  
<td><input type="text" name="subjectName"/></td>  
</tr>  
<tr>  
<td>Subject code:</td>  
<td><input type="text" name="subjectCode"/></td>  
</tr>  
<tr>  
<td></td>  
<td><input type="submit" value="ADD SUBJECT"/></td>  
</tr>  
</table>  
</form>  
</body>  
</html>
```

Create the `get_subjects.jsp` file.

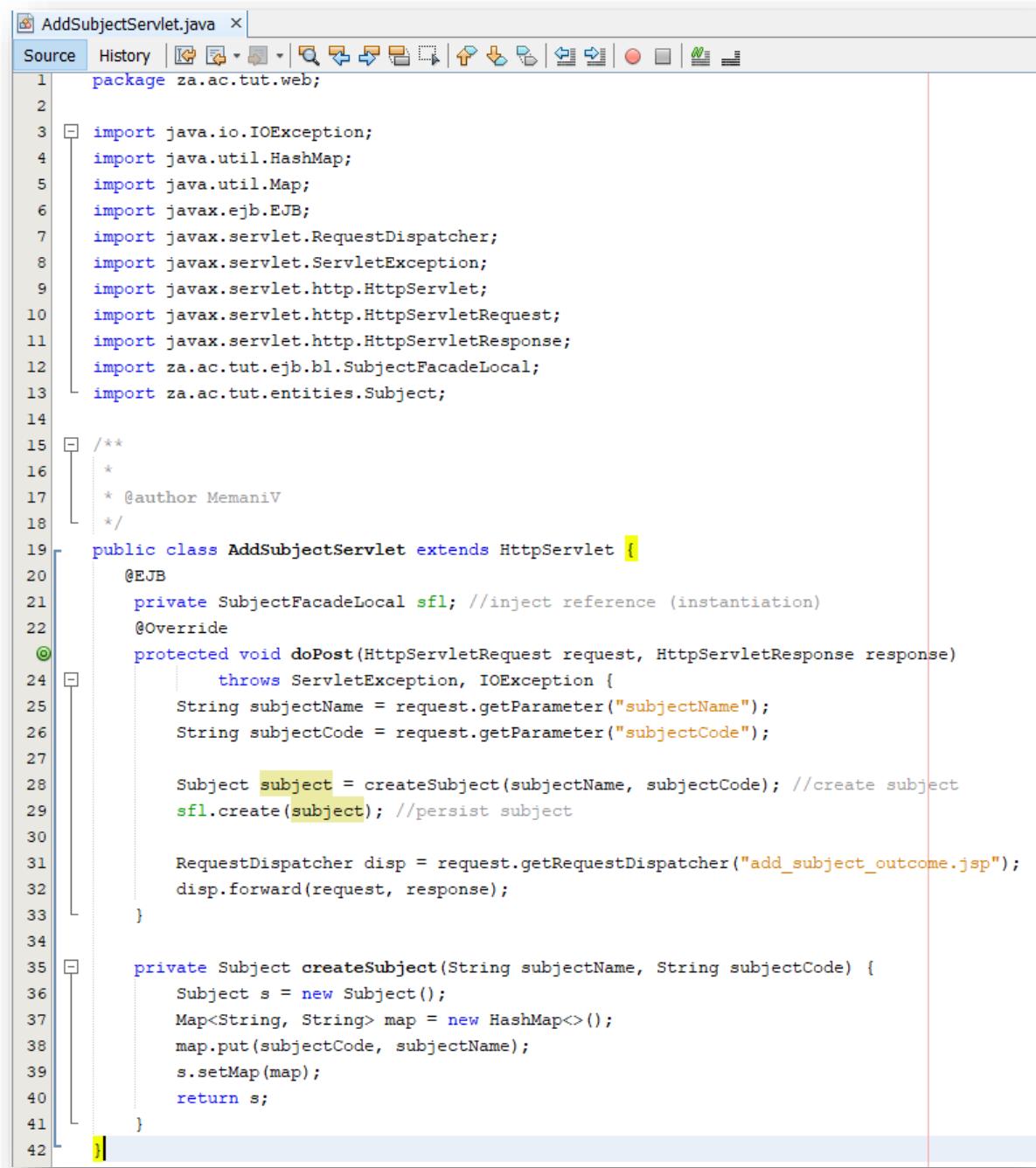


```
<%--  
1 Document      : get_subjects  
2 Created on   : 30 Mar 2023, 1:02:38 PM  
3 Author        : MemaniV  
4 --%>  
5  
6  
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>  
8 <!DOCTYPE html>  
9 <html>  
10 <head>  
11     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
12     <title>Get Subjects Page</title>  
13 </head>  
14 <body>  
15     <h1>Get subjects</h1>  
16     <p>  
17         Please click on the button below to get all the subjects.  
18     </p>  
19     <form action="GetSubjectsServlet.do" method="GET">  
20         <table>  
21             <tr>  
22                 <td></td>  
23                 <td><input type="submit" value="GET SUBJECTS"/></td>  
24             </tr>  
25         </table>  
26     </form>  
27 </body>  
28 </html>
```

Add the **SubjectEJBModule.jar** file to the library of **SubjectWebApp**.



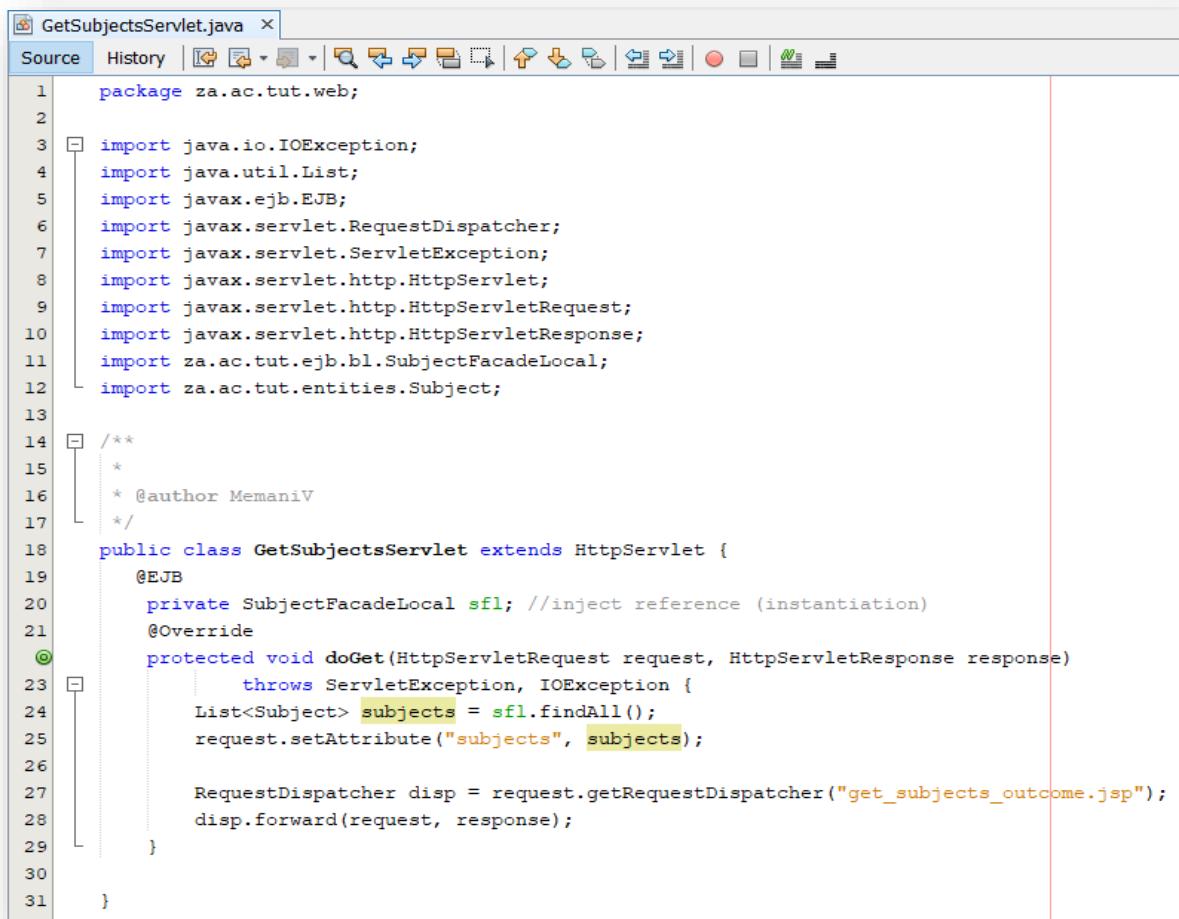
Create the AddSubjectServlet.java file.



The screenshot shows a Java code editor window with the title bar "AddSubjectServlet.java". The menu bar includes "Source", "History", and various tool icons. The code itself is a Java servlet named "AddSubjectServlet" that extends "HttpServlet". It imports several Java packages and classes, including "java.io.IOException", "java.util.HashMap", "java.util.Map", "javax.ejb.EJB", "javax.servlet.RequestDispatcher", "javax.servlet.ServletException", "javax.servlet.http.HttpServlet", "javax.servlet.http.HttpServletRequest", "javax.servlet.http.HttpServletResponse", "za.ac.tut.ejb.bl.SubjectFacadeLocal", and "za.ac.tut.entities.Subject". The class contains annotations "@EJB" and "@Override". The "doPost" method retrieves subject name and code from the request, creates a new "Subject" object, and persists it using the injected "SubjectFacadeLocal" reference. Finally, it forwards the request to a JSP page. A private helper method "createSubject" is also defined to handle the creation of the "Subject" object.

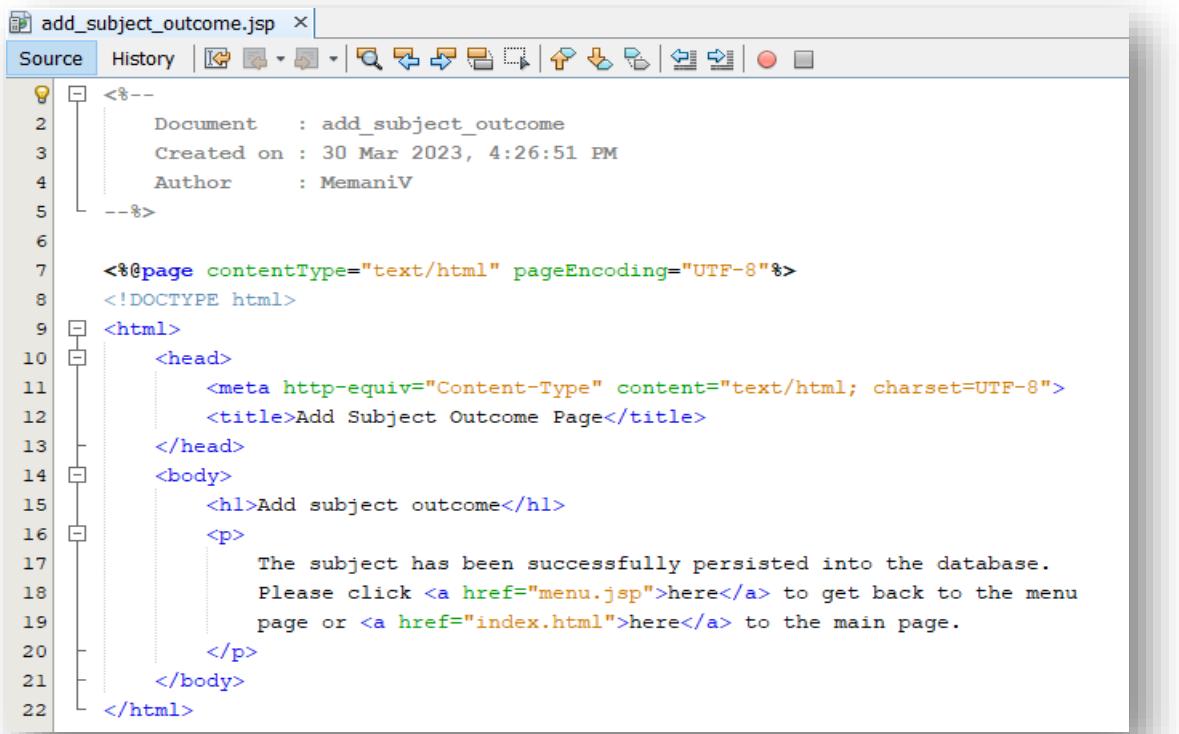
```
1 package za.ac.tut.web;
2
3 import java.io.IOException;
4 import java.util.HashMap;
5 import java.util.Map;
6 import javax.ejb.EJB;
7 import javax.servlet.RequestDispatcher;
8 import javax.servlet.ServletException;
9 import javax.servlet.http.HttpServlet;
10 import javax.servlet.http.HttpServletRequest;
11 import javax.servlet.http.HttpServletResponse;
12 import za.ac.tut.ejb.bl.SubjectFacadeLocal;
13 import za.ac.tut.entities.Subject;
14
15 /**
16 * @author MemaniV
17 */
18 public class AddSubjectServlet extends HttpServlet {
19     @EJB
20     private SubjectFacadeLocal sfl; //inject reference (instantiation)
21     @Override
22     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
23         String subjectName = request.getParameter("subjectName");
24         String subjectCode = request.getParameter("subjectCode");
25
26         Subject subject = createSubject(subjectName, subjectCode); //create subject
27         sfl.create(subject); //persist subject
28
29         RequestDispatcher disp = request.getRequestDispatcher("add_subject_outcome.jsp");
30         disp.forward(request, response);
31     }
32
33     private Subject createSubject(String subjectName, String subjectCode) {
34         Subject s = new Subject();
35         Map<String, String> map = new HashMap<>();
36         map.put(subjectCode, subjectName);
37         s.setMap(map);
38         return s;
39     }
40 }
41
42 }
```

Create the **GetSubjectsServlet.java** file.



```
1 package za.ac.tut.web;
2
3 import java.io.IOException;
4 import java.util.List;
5 import javax.ejb.EJB;
6 import javax.servlet.RequestDispatcher;
7 import javax.servlet.ServletException;
8 import javax.servlet.http.HttpServlet;
9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11 import za.ac.tut.ejb.bl.SubjectFacadeLocal;
12 import za.ac.tut.entities.Subject;
13
14 /**
15 *
16 * @author MemaniV
17 */
18 public class GetSubjectsServlet extends HttpServlet {
19     @EJB
20     private SubjectFacadeLocal sfl; //inject reference (instantiation)
21     @Override
22     protected void doGet(HttpServletRequest request, HttpServletResponse response)
23             throws ServletException, IOException {
24         List<Subject> subjects = sfl.findAll();
25         request.setAttribute("subjects", subjects);
26
27         RequestDispatcher disp = request.getRequestDispatcher("get_subjects_outcome.jsp");
28         disp.forward(request, response);
29     }
30 }
31 }
```

Create the **add_subject_outcome.jsp** file.



```
<%-->
1 Document      : add_subject_outcome
2 Created on   : 30 Mar 2023, 4:26:51 PM
3 Author        : MemaniV
4 --%>
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>Add Subject Outcome Page</title>
13    </head>
14    <body>
15        <h1>Add subject outcome</h1>
16        <p>
17            The subject has been successfully persisted into the database.
18            Please click <a href="menu.jsp">here</a> to get back to the menu
19            page or <a href="index.html">here</a> to the main page.
20        </p>
21    </body>
22 </html>
```

Create the get_subjects_outcome.jsp file.

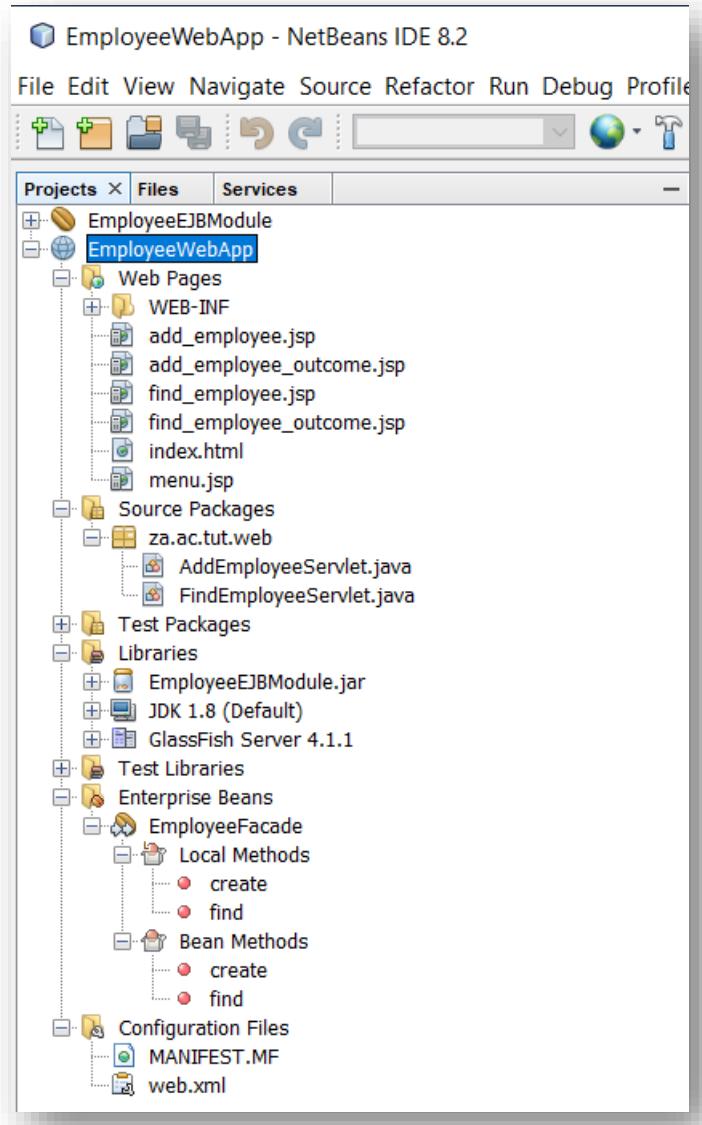
The screenshot shows a JSP file named "find_employee_outcome.jsp" in an IDE. The code is as follows:

```
<%--  
Document : find_employee_outcome  
Created on : 28 Mar 2023, 5:34:14 PM  
Author : MemaniV  
--%>  
  
<%@page import="za.ac.tut.entities.Employee"%>  
<%@page contentType="text/html" pageEncoding="UTF-8"%>  
<!DOCTYPE html>  
<html>  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
<title>Find Employee Outcome Page</title>  
</head>  
<body>  
<h1>Find employee outcome</h1>  
<%  
Employee emp = (Employee)request.getAttribute("employee");  
Long empID = emp.getId();  
String empCellNo = emp.getContactNos().get(0);  
String nextKinCellNo = emp.getContactNos().get(1);  
%>  
<p>  
Below are the details of the employee.  
</p>
```

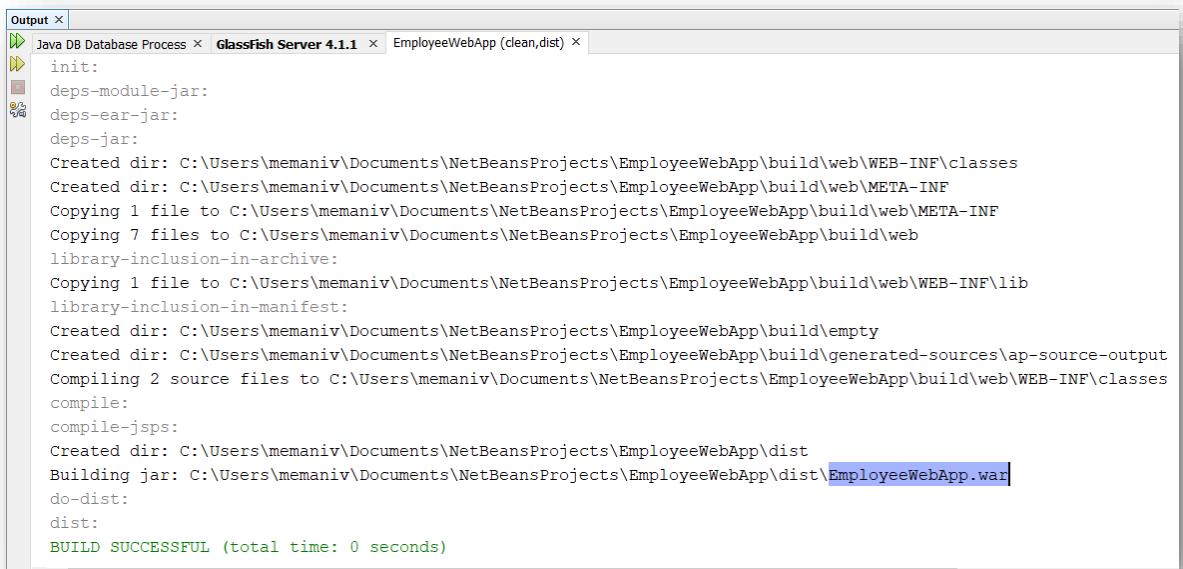
The screenshot shows the continuation of the JSP file "find_employee_outcome.jsp". The code is as follows:

```
<table>  
<tr>  
<td><b>Employee ID:</b> </td>  
<td><%=empID%></td>  
</tr>  
<tr>  
<td><b>Cell number:</b> </td>  
<td><%=empCellNo%></td>  
</tr>  
<tr>  
<td><b>Next of kin cell number:</b> </td>  
<td><%=nextKinCellNo%></td>  
</tr>  
</table>  
<p>  
Please click <a href="menu.jsp">here</a> to get back to the menu  
page or <a href="index.html">here</a> to the main page.  
</p>  
</body>  
</html>
```

View the complete project structure of **EmployeeWebApp**.

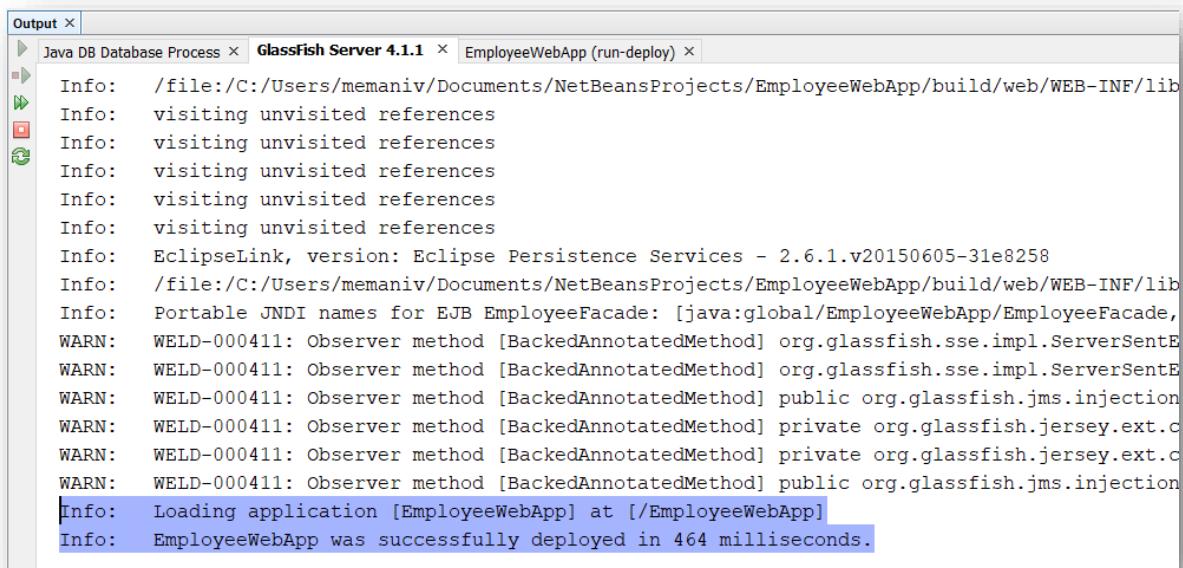


Compile the project.



```
Output ×
Java DB Database Process × GlassFish Server 4.1.1 × EmployeeWebApp (clean,dist) ×
init:
deps-module-jar:
deps-ear-jar:
deps-jar:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\EmployeeWebApp\build\web\WEB-INF\classes
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\EmployeeWebApp\build\web\META-INF
Copying 1 file to C:\Users\memaniv\Documents\NetBeansProjects\EmployeeWebApp\build\web\META-INF
Copying 7 files to C:\Users\memaniv\Documents\NetBeansProjects\EmployeeWebApp\build\web
library-inclusion-in-archive:
Copying 1 file to C:\Users\memaniv\Documents\NetBeansProjects\EmployeeWebApp\build\web\WEB-INF\lib
library-inclusion-in-manifest:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\EmployeeWebApp\build\empty
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\EmployeeWebApp\build\generated-sources\ap-source-output
Compiling 2 source files to C:\Users\memaniv\Documents\NetBeansProjects\EmployeeWebApp\build\web\WEB-INF\classes
compile:
compile-jsps:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\EmployeeWebApp\dist
Building jar: C:\Users\memaniv\Documents\NetBeansProjects\EmployeeWebApp\dist\EmployeeWebApp.war
do-dist:
dist:
BUILD SUCCESSFUL (total time: 0 seconds)
```

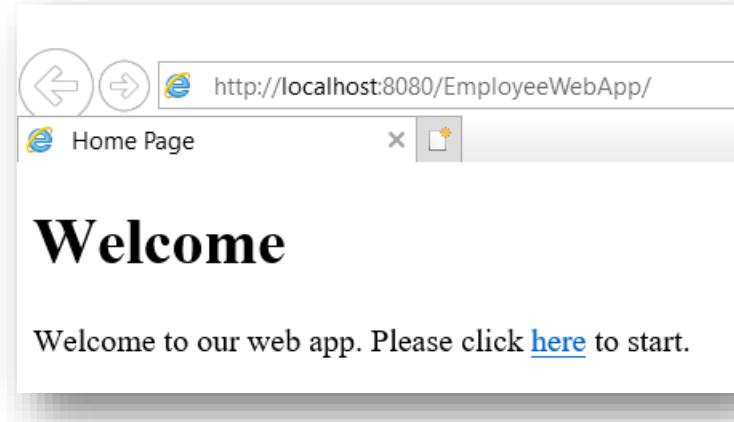
Deploy the project.



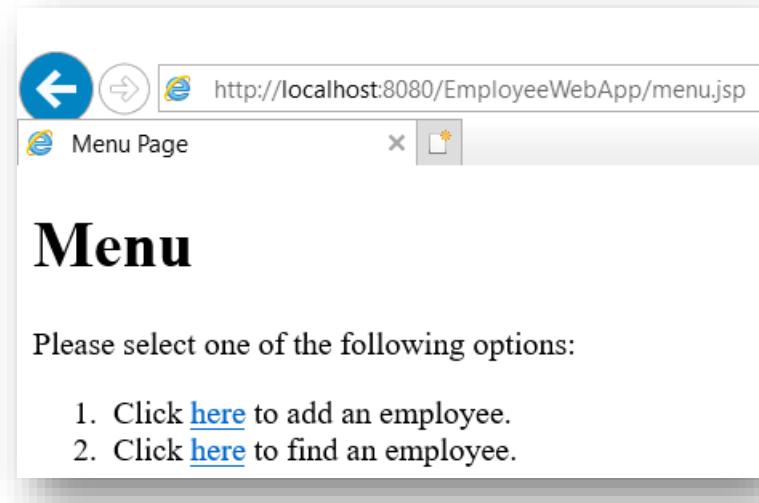
```
Output ×
Java DB Database Process × GlassFish Server 4.1.1 × EmployeeWebApp (run-deploy) ×
Info: /file:/C:/Users/memaniv/Documents/NetBeansProjects/EmployeeWebApp/build/web/WEB-INF/lib
Info: visiting unvisited references
Info: EclipseLink, version: Eclipse Persistence Services - 2.6.1.v20150605-31e8258
Info: /file:/C:/Users/memaniv/Documents/NetBeansProjects/EmployeeWebApp/build/web/WEB-INF/lib
Info: Portable JNDI names for EJB EmployeeFacade: [java:global/EmployeeWebApp/EmployeeFacade,
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] org.glassfish.sse.impl.ServerSentE
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] org.glassfish.sse.impl.ServerSentE
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] public org.glassfish.jms.injection
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] private org.glassfish.jersey.ext.c
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] private org.glassfish.jersey.ext.c
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] public org.glassfish.jms.injection
Info: Loading application [EmployeeWebApp] at [/EmployeeWebApp]
Info: EmployeeWebApp was successfully deployed in 464 milliseconds.
```

Part E – Run the web application.

Launch the application.



Click on the link



Add an employee.

- ✓ Click on the first link.

The screenshot shows a web browser window with the URL http://localhost:8080/EmployeeWebApp/add_employee.jsp. The title bar says "Add Employee Page". The main content area has a large heading "Add employee" and a sub-instruction "Please add employee details below:". Below this are three input fields: "Employee ID:", "Cell number:", and "Next of kin cell number:", each with a corresponding empty text box. At the bottom is a grey "ADD EMPLOYEE" button.

- ✓ Fill in the form.

The screenshot shows the same web browser window as the previous one, but now with filled form fields. The "Employee ID:" field contains "111", the "Cell number:" field contains "01234", and the "Next of kin cell number:" field contains "56789". The "ADD EMPLOYEE" button is still at the bottom.

- ✓ Click on the add button.

Add employee outcome

The employee has been successfully persisted into the database. Please click [here](#) to get back to the menu page or [here](#) to the main page.

- ✓ Go back to the menu page.

Menu

Please select one of the following options:

1. Click [here](#) to add an employee.
2. Click [here](#) to find an employee.

- ✓ Add four more employees into the database. View the records in the database.

#	ID	CREATIONDATE
1	111	2023-03-28 17:53:05.967
2	222	2023-03-28 17:55:50.501
3	333	2023-03-28 17:56:10.846
4	444	2023-03-28 17:56:28.007
5	555	2023-03-28 17:56:45.662

#	EMPLOYEE_ID	CONTACT
1	111	01234
2	111	56789
3	222	12456
4	222	23478
5	333	23900
6	333	11156
7	444	56832
8	444	61230
9	555	10297
10	555	20296

Search for an employee.

- ✓ Click on the second link.

http://localhost:8080/EmployeeWebApp/find_employee.jsp

Find Employee Page

Find employee

Please enter the ID of the employee to search for.

Employee ID:

FIND

- ✓ Enter ID number **333**.

http://localhost:8080/EmployeeWebApp/find_employee.jsp

Find Employee Page

Find employee

Please enter the ID of the employee to search for.

Employee ID:

FIND

- ✓ Click on the button.

http://localhost:8080/EmployeeWebApp/FindEmployeeServlet.do

Find Employee Outcome Pa... ×

Find employee outcome

Below are the details of the employee.

Employee ID: 333

Cell number: 23900

Next of kin cell number: 11156

Please click [here](#) to get back to the menu page or [here](#) to the main page.

8.4 Mapping relationships

JPA also supports the mapping of relationships between tables. There four relationships that are normally used, namely;

- OneToOne;
- OneToMany;
- ManyToOne; and
- ManyToMany.

JPA uses annotations to map these relationships. In our case, we are going to confine ourselves to two annotations, **@OneToOne** and **@OneToMany**.

8.4.1 OneToOne

The **@OneToOne** annotation is used to denote a **one-to-one** relationship existing between two tables. This means one table has one reference to another table. In relational databases, this reference is a foreign key. So we can say we have a mother table and a child table. The foreign key points to a record in the child table associated with a record in the mother table.

Talking objects wise, you have one class containing an instance of another class. This means you have a container class which serves as the parent and the contained class which serves as a child.

Example

Create a web application that will perform some of the **CRUD** (**C**reate **R**ead **U**pdate **D**elete) operations on books. A book has an **isbn**, **title**, **author**, **publication date** and **price**. An author has a **name**, **surname** and **email address**.

BOOK_TBL	
Field	Constraint
isbn	Primary key.
title	Must be a string. Cannot be nullable. The maximum length is 20.
author	Must be a contained instance. Must have a reference to the child class called author.
publicationDate	Must be a Date. Cannot be nullable.
price	Must be a Double. Cannot be nullable.
creationDate	The timestamp.

AUTHOR_TBL	
Field	Constraint
id	Primary key. Must be automatically generated by the persistence provider.
name	Must be a string. Cannot be nullable. The maximum length is 50.
surname	Must be a string. Cannot be nullable. The maximum length is 50.

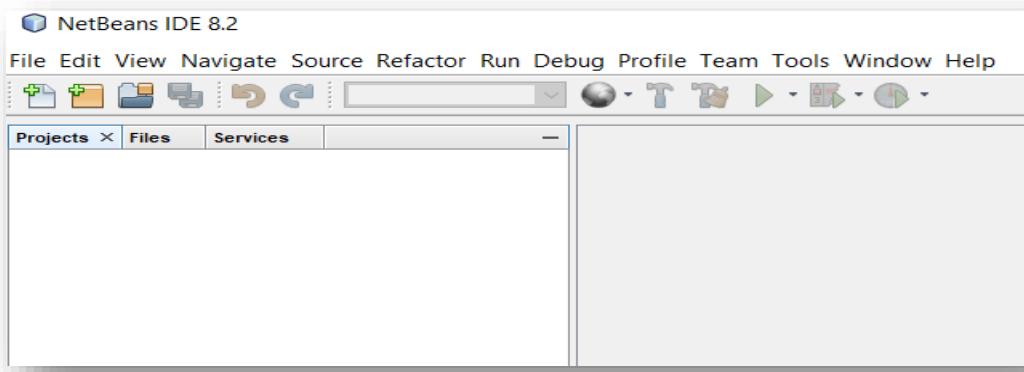
Solution approach

The solution to this problem is going to be done in five parts, namely:

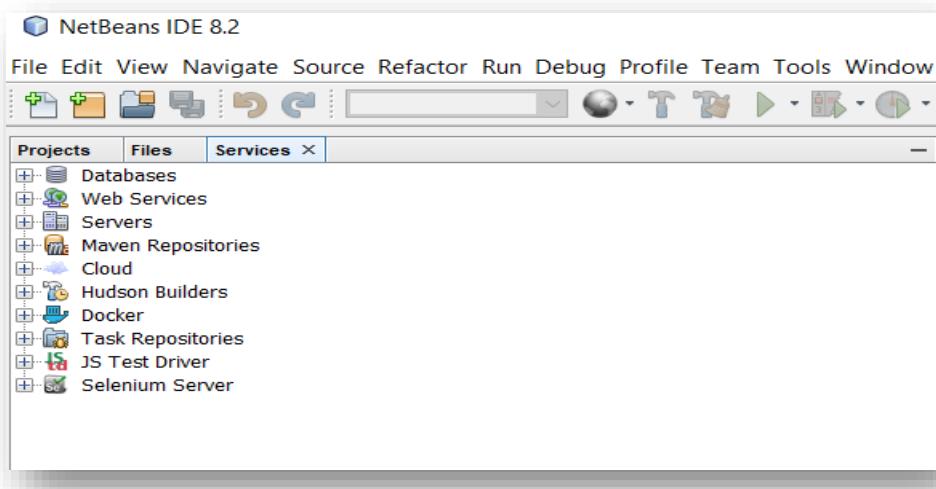
- **Part A:** we create a database.
- **Part B:** we establish connection between the created database and GlassFish.
- **Part C:** we create an EJB module. The module will have an entity with business logic.
- **Part D:** we create a client web application.
- **Part E:** we run the client web application.

Part A - Create a database

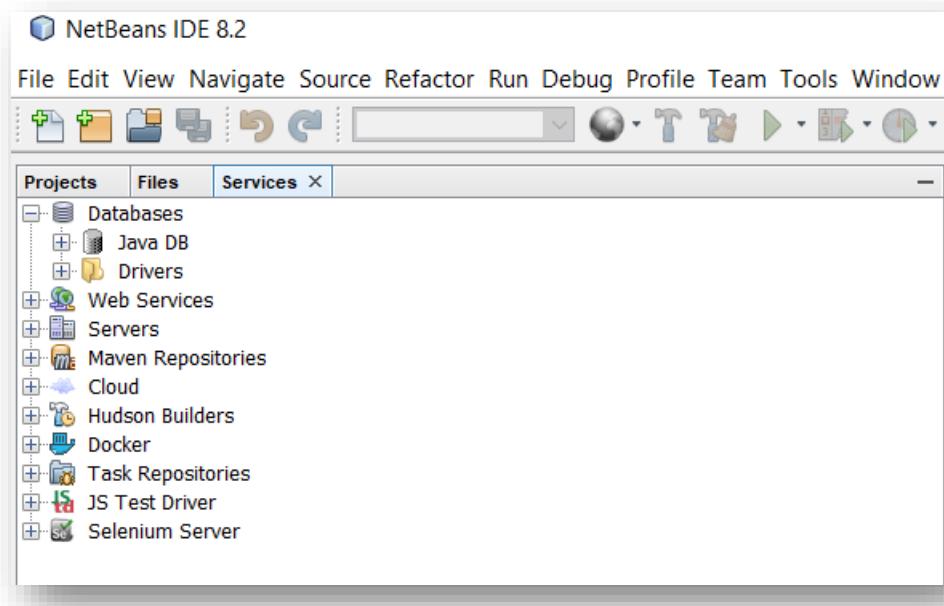
Launch NetBeans.



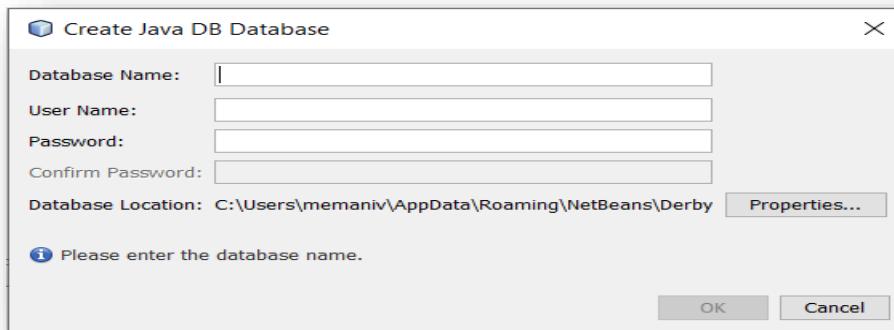
Click on the **Services** tab.



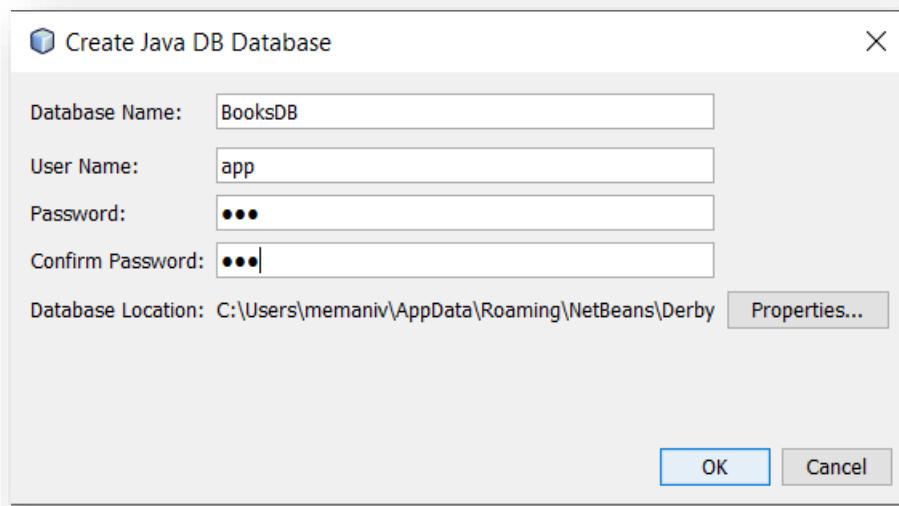
Expand the **Databases** option.



Right-click on **Java DB** and select **Create Database**.

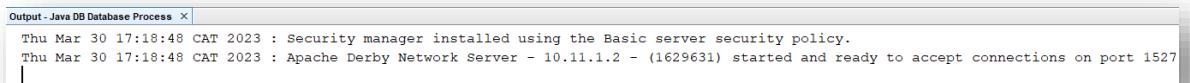


Fill-in the form. I made the password to be **123**.



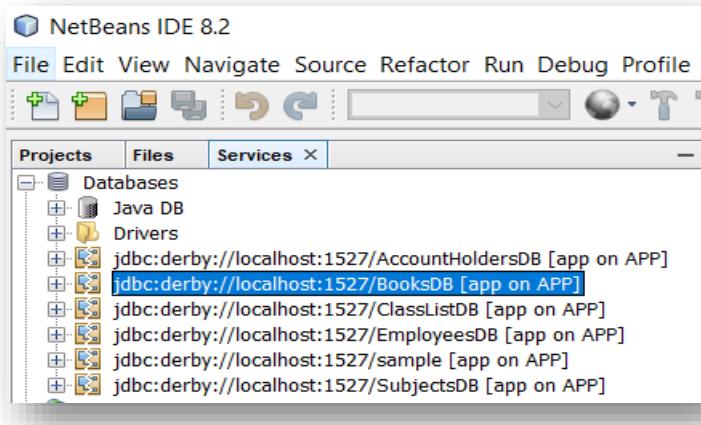
Click **OK**.

- ✓ An output message confirming that **Derby** has started and ready to accept connections on port 1527 is shown.

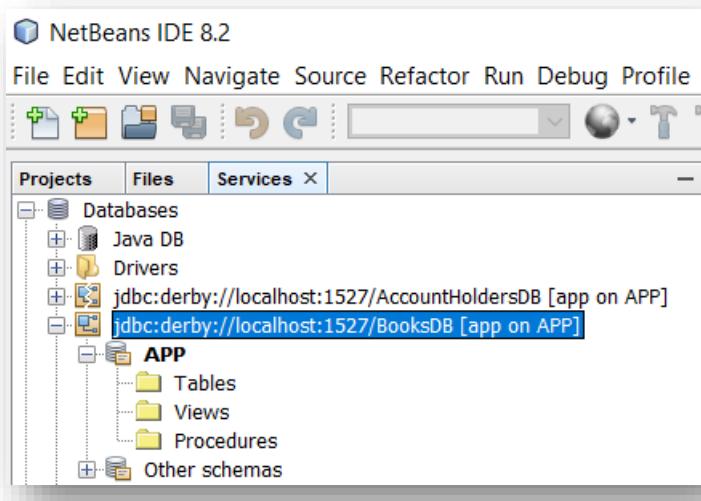


```
Output - Java DB Database Process ×
Thu Mar 30 17:18:48 CAT 2023 : Security manager installed using the Basic server security policy.
Thu Mar 30 17:18:48 CAT 2023 : Apache Derby Network Server - 10.11.1.2 - (1629631) started and ready to accept connections on port 1527
```

- ✓ A database is created.

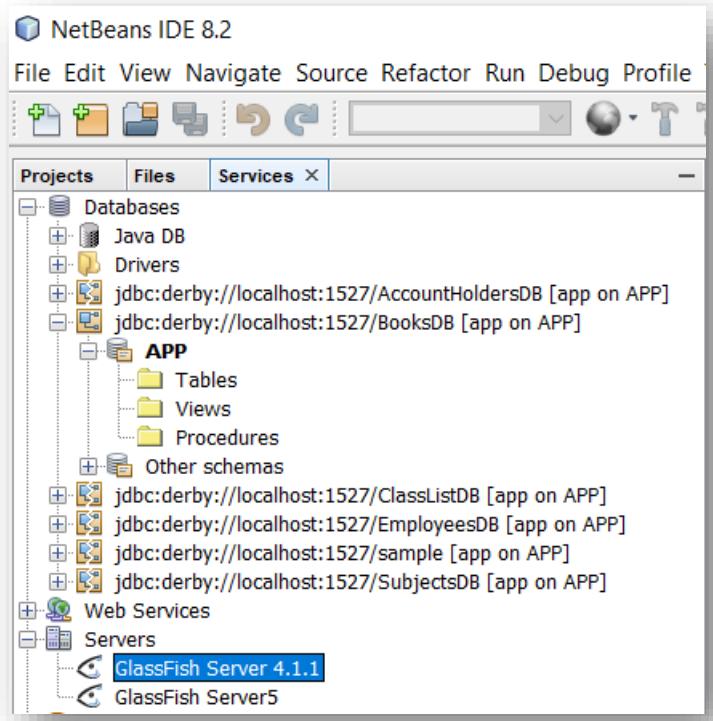


Right-click on the database and select **Connect**.

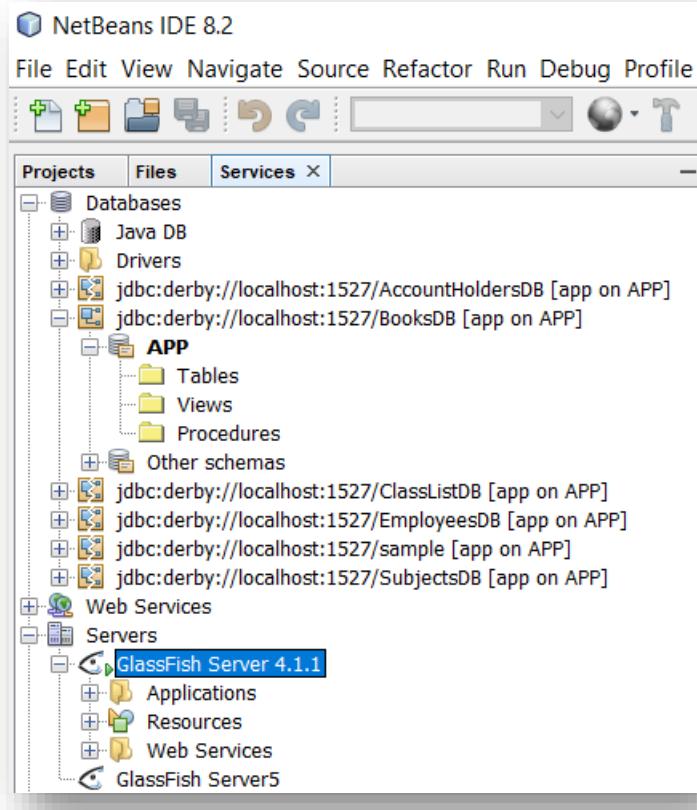


Part B – Connect the database to the application server.

Click on the **Services** tab.



Start the server.



Right-click on the server and select **View Domain Admin Console**.

The screenshot shows the GlassFish Server Open Source Edition Common Tasks console. The URL in the browser is <http://localhost:4848/common/index.jsf>. The title bar says "GlassFish Console - Comm...". The top menu has "Home" and "About...". Below it, the user information is "User: admin | Domain: domain1 | Server: localhost". The main title is "GlassFish™ Server Open Source Edition". On the left, there's a sidebar titled "Common Tasks" with the following items: Domain, server (Admin Server), Clusters, Standalone Instances, Nodes, Applications, Lifecycle Modules, Monitoring Data, Resources (which is expanded to show Concurrent Resources, Connectors, JDBC, JMS Resources, JNDI, JavaMail Sessions, and Resource Adapter Configs), Configurations (with default-config and server-config), and Update Tool. The right panel is titled "GlassFish Console - Common Tasks" and contains sections for GlassFish News, Deployment (List Deployed Applications, Deploy an Application), Administration (Change Administrator Password, List Password Aliases), and Monitoring (Monitoring Data).

Under **Common Tasks** panel, expand **JDBC**.

The screenshot shows the GlassFish Server Open Source Edition JDBC console. The URL in the browser is <http://localhost:4848/common/index.jsf>. The title bar says "JDBC". The top menu has "Home" and "About...". Below it, the user information is "User: admin | Domain: domain1 | Server: localhost". The main title is "GlassFish™ Server Open Source Edition". On the left, the "Common Tasks" sidebar is identical to the previous screenshot. The right panel is titled "JDBC" and contains two main categories: "JDBC Resources" and "JDBC Connection Pools".

Modify the Connection pool to include the **ClassListDB**.

- ✓ Click on **JDBC Connection Pools**.

JDBC Connection Pools					
To store, organize, and retrieve data, most applications use relational databases. Java EE applications access relational databases through the JDBC API. Before an application can access a database, it must get a connection.					
Pools (3)					
Select	Pool Name	Resource Type	Classname	Description	
<input type="checkbox"/>	DerbyPool	javax.sql.DataSource	org.apache.derby.jdbc.ClientDataSource		
<input type="checkbox"/>	SamplePool	javax.sql.DataSource	org.apache.derby.jdbc.ClientDataSource		
<input type="checkbox"/>	TimerPool	javax.sql.XADatasource	org.apache.derby.jdbc.EmbeddedXADataSource		

- ✓ Click on the **DerbyPool** link.

General	Advanced	Additional Properties
Edit JDBC Connection Pool Modify an existing JDBC connection pool. A JDBC connection pool is a group of reusable connections for a particular database. <input type="button" value="Load Defaults"/> <input type="button" value="Flush"/> <input type="button" value="Ping"/>		
General Settings		
Pool Name:	DerbyPool	
Resource Type:	<input type="text" value="javax.sql.DataSource"/> <input type="button" value="▼"/>	
Must be specified if the datasource class implements more than 1 of the interface.		
Datasource Classname:	<input type="text" value="org.apache.derby.jdbc.ClientDataSource"/>	
Vendor-specific classname that implements the DataSource and/or XADatasource APIs		
Driver Classname:	<input type="text"/>	
Vendor-specific classname that implements the java.sql.Driver interface.		
Ping:	<input type="checkbox"/> Enabled	
When enabled, the pool is pinged during creation or reconfiguration to identify and warn of any erroneous values for its attributes		
Deployment Order:	<input type="text" value="100"/>	
Specifies the loading order of the resource at server startup. Lower numbers are loaded first.		
Description:	<input type="text"/>	
Pool Settings		
Initial and Minimum Pool Size:	<input type="text" value="8"/>	Connections Minimum and initial number of connections maintained in the pool
Maximum Pool Size:	<input type="text" value="32"/>	Connections Maximum number of connections that can be created to satisfy client requests
Pool Resize Quantity:	<input type="text" value="2"/>	Connections

- ✓ Click on **Additional Properties**.

General	Advanced	Additional Properties
Edit JDBC Connection Pool Properties Modify properties of an existing JDBC connection pool.		
Pool Name: DerbyPool		
Additional Properties (6) <input type="button" value="Add Property"/> <input type="button" value="Delete Properties"/>		
Select	Name	Value
<input type="checkbox"/>	PortNumber	<input type="text" value="1527"/>
<input type="checkbox"/>	Password	<input type="text" value="APP"/>
<input type="checkbox"/>	User	<input type="text" value="APP"/>
<input type="checkbox"/>	serverName	<input type="text" value="localhost"/>
<input type="checkbox"/>	DatabaseName	<input type="text" value="sun-appserv-samples"/>
<input type="checkbox"/>	connectionAttributes	<input type="text" value=";create=true"/>

- ✓ Click on **Add Property** and add a **url** property with the following value:

jdbc:derby://localhost:1527/BooksDB

- ✓ Delete the **connectionAttributes** property.

General	Advanced	Additional Properties
---------	----------	-----------------------

Edit JDBC Connection Pool Properties

Modify properties of an existing JDBC connection pool.

Pool Name: DerbyPool

Additional Properties (6)

Select	Name	Value
<input type="checkbox"/>	PortNumber	1527
<input type="checkbox"/>	Password	123
<input type="checkbox"/>	User	app
<input type="checkbox"/>	serverName	localhost
<input type="checkbox"/>	DatabaseName	BooksDB
<input type="checkbox"/>	url	jdbc:derby://localhost:1527/BooksDB

- ✓ Save the changes by clicking the **Save** button.

General	Advanced	Additional Properties
---------	----------	-----------------------

✓ New values successfully saved.

Edit JDBC Connection Pool Properties

Modify properties of an existing JDBC connection pool.

Pool Name: DerbyPool

Additional Properties (6)

Select	Name	Value
<input type="checkbox"/>	PortNumber	1527
<input type="checkbox"/>	Password	123
<input type="checkbox"/>	User	app
<input type="checkbox"/>	serverName	localhost
<input type="checkbox"/>	DatabaseName	BooksDB
<input type="checkbox"/>	url	jdbc:derby://localhost:1527/BooksDB

- ✓ Check if the created connection pool is working. Do this by clicking the **Ping** button.

Edit JDBC Connection Pool

Modify an existing JDBC connection pool. A JDBC connection pool is a group of reusable connections for a particular database.

General Settings

Pool Name: DerbyPool

Resource Type: javax.sql.DataSource

Must be specified if the datasource class implements more than 1 of the interface.

Datasource Classname: org.apache.derby.jdbc.ClientDataSource

Vendor-specific classname that implements the DataSource and/or XDataSource APIs

Driver Classname:

Vendor-specific classname that implements the java.sql.Driver interface.

Ping: Enabled
When enabled, the pool is pinged during creation or reconfiguration to identify and warn of any erroneous values for its attributes

Deployment Order: 100
Specifies the loading order of the resource at server startup. Lower numbers are loaded first.

Description:

Confirm that the resource points to the DerbyPool.

- ✓ Still under **JDBC**, click on **JDBC Resources**.

JDBC Resources

JDBC resources provide applications with a means to connect to a database.

Resources (3)

Select	JNDI Name	Logical JNDI Name	Enabled	Connection Pool
<input type="checkbox"/>	jdbc/_TimerPool		<input checked="" type="checkbox"/>	_TimerPool
<input type="checkbox"/>	jdbc/_default	java.comp/DefaultDataSource	<input checked="" type="checkbox"/>	DerbyPool
<input type="checkbox"/>	jdbc/sample		<input checked="" type="checkbox"/>	SamplePool

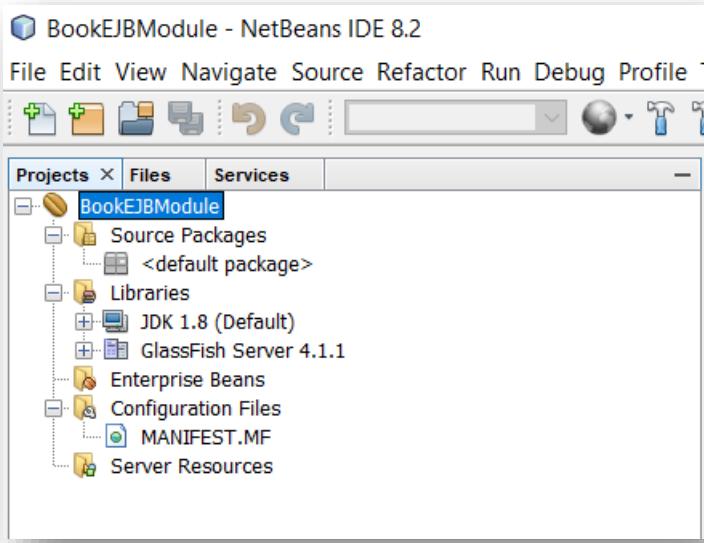
Take note of the purpose of JDBC Resources. It is said:

"JDBC resources provide applications with a means to connect to a database."

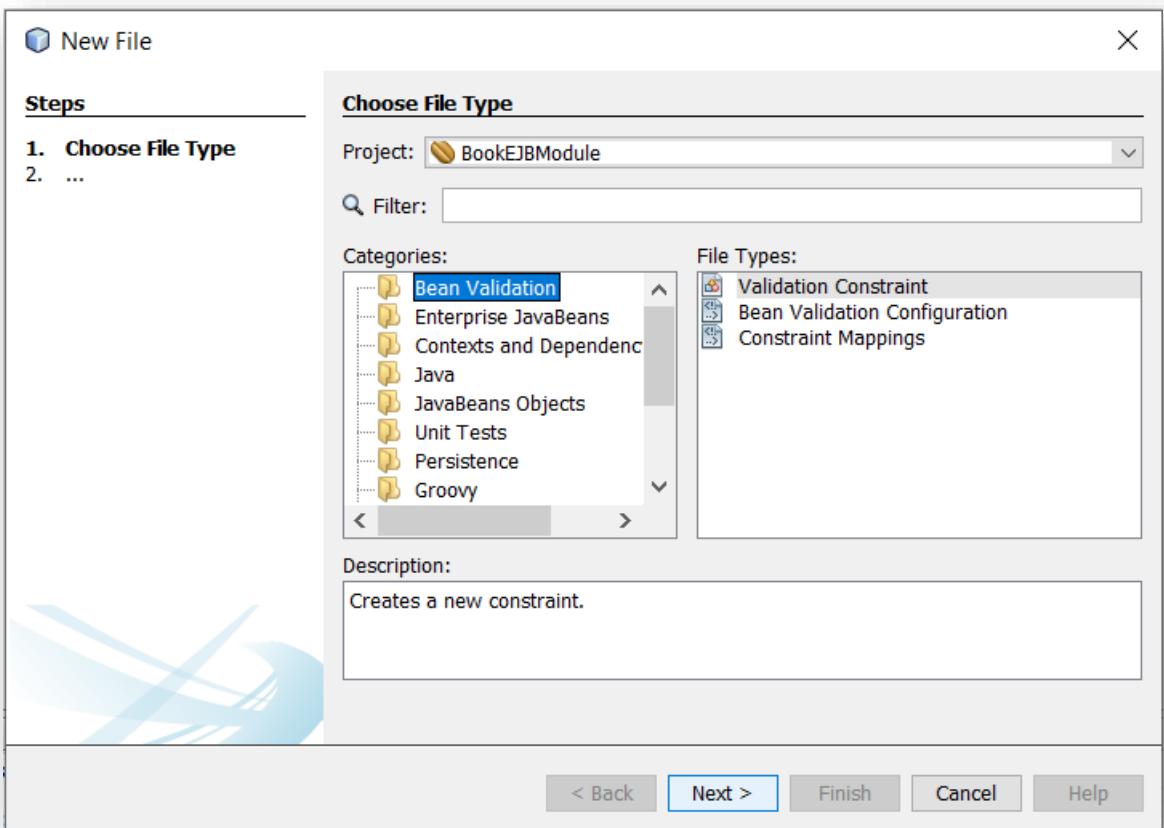
You can also see that the **jdbc/_default** resource is associated with the **DerbyPool** Connection Pool.

Part C – Create an entity.

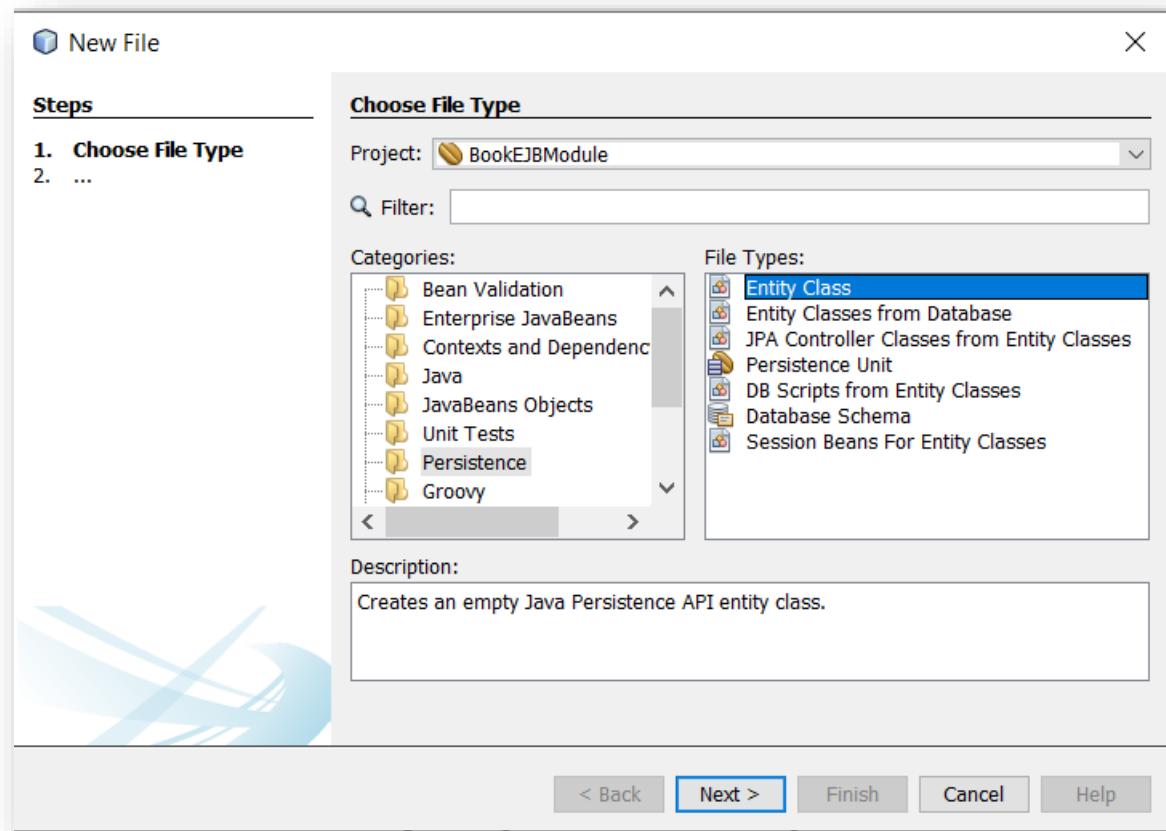
Create an EJB project called **BooksEJBModule**.



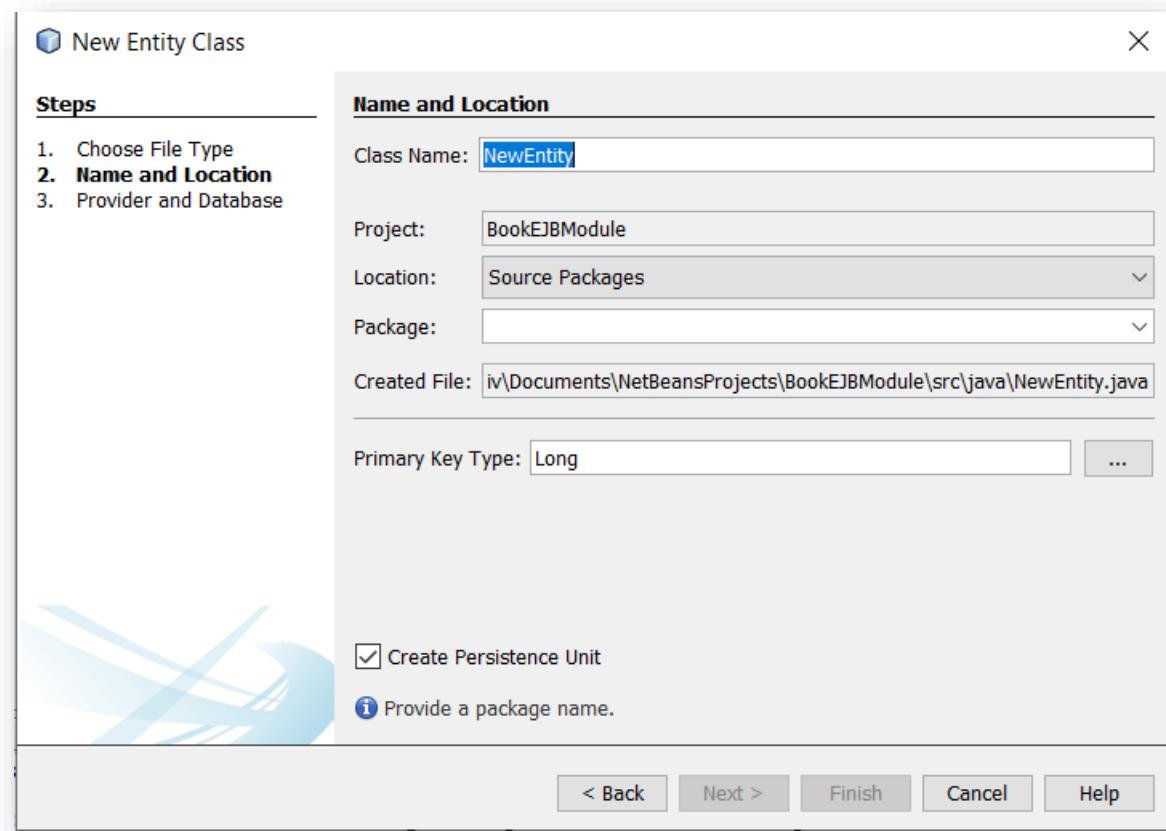
Right-click on the project and select **New | Other**.



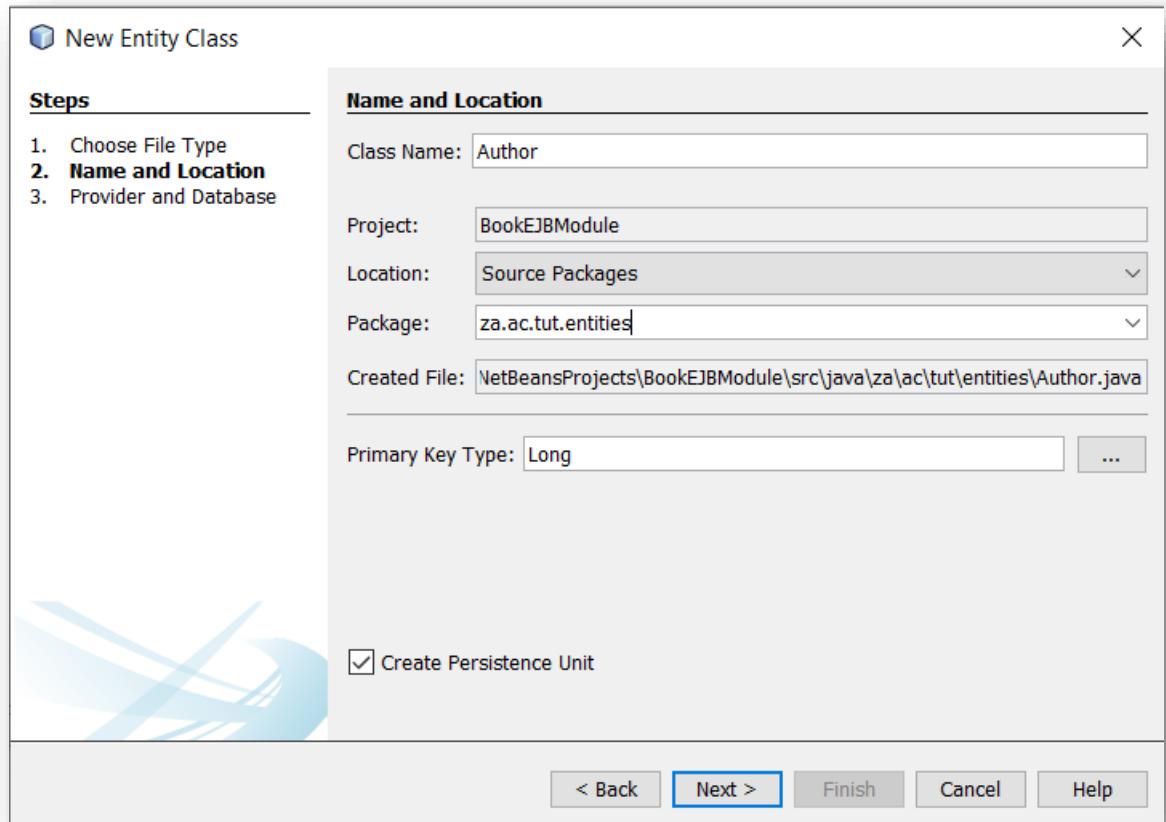
Under **Categories** select **Persistence**, and under **File Types** select **Entity Class**.



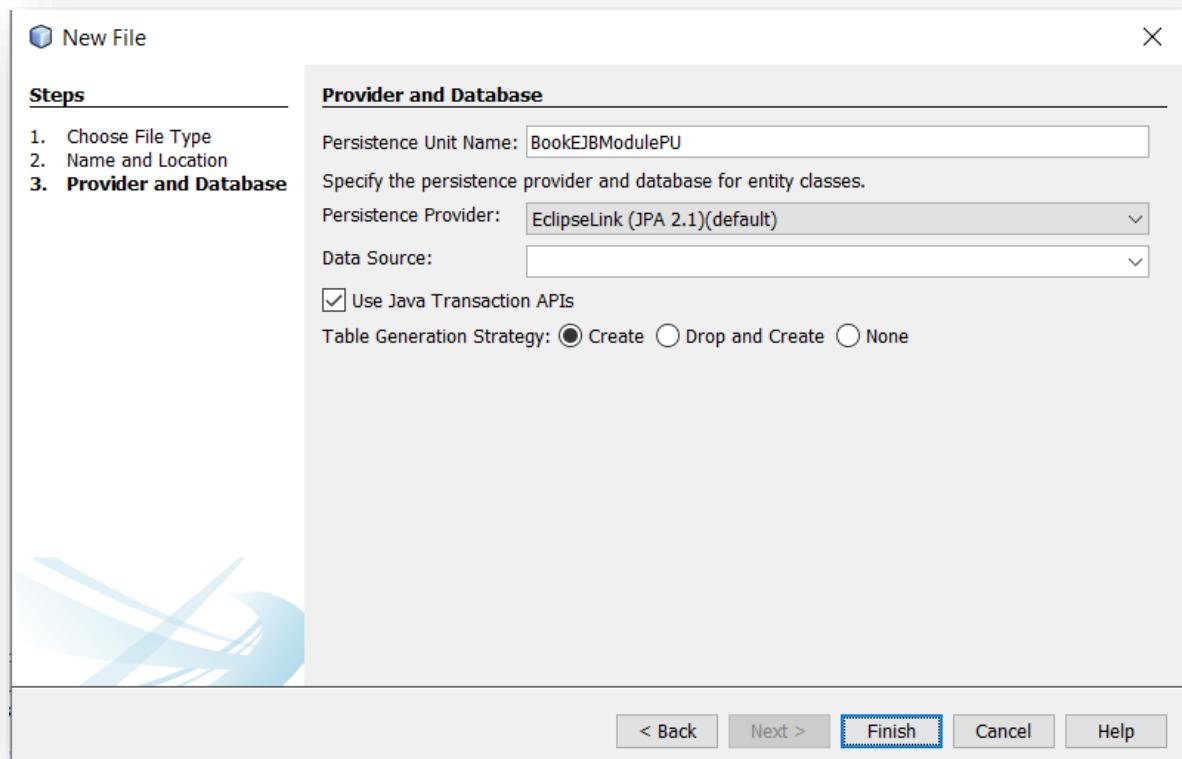
Click **Next**.



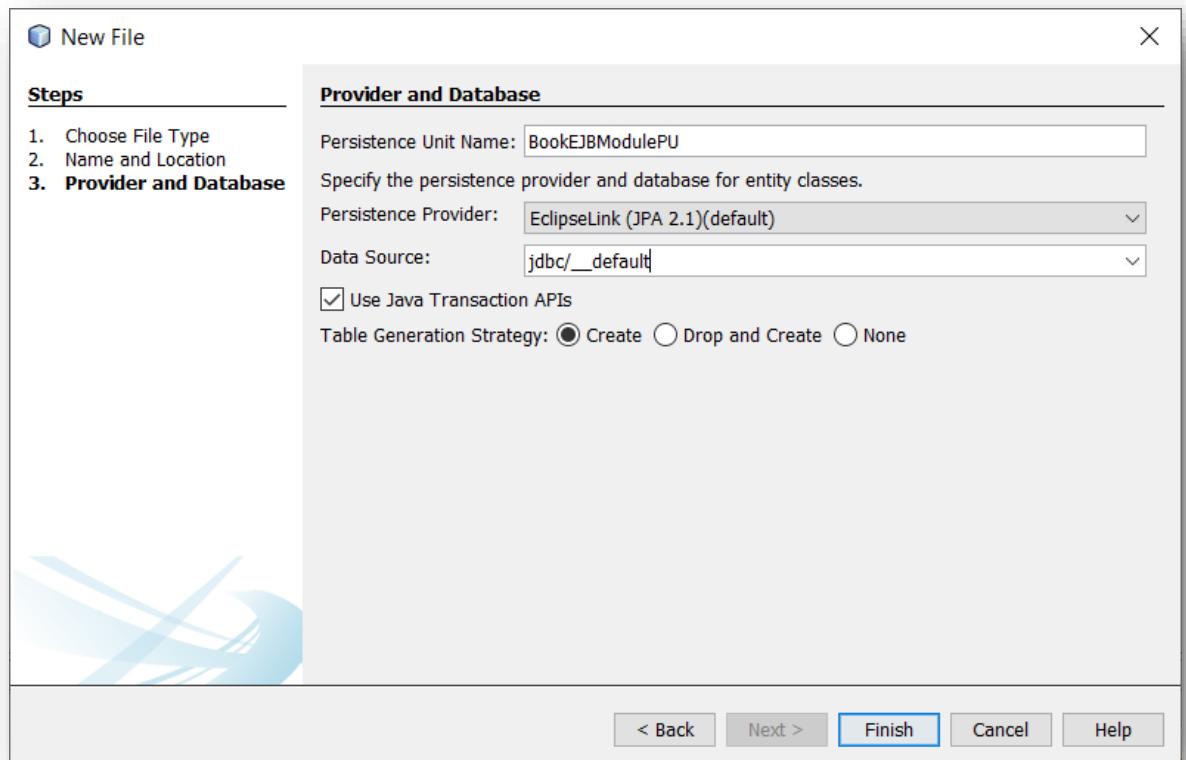
Fill-in the form.



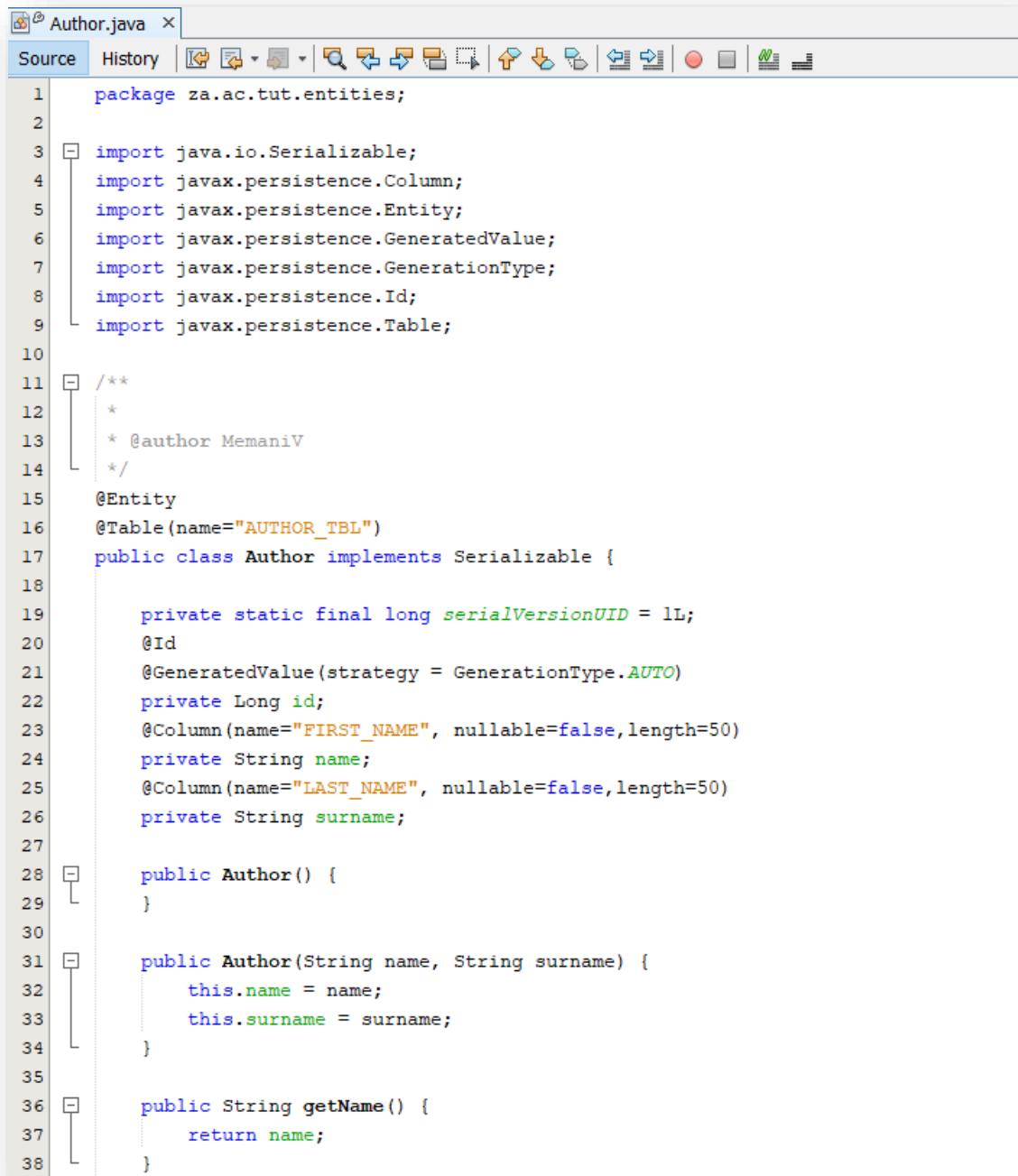
Click **Next**.



Under **Data Source**, select **jdbc/_default**.



Click **Finish**.



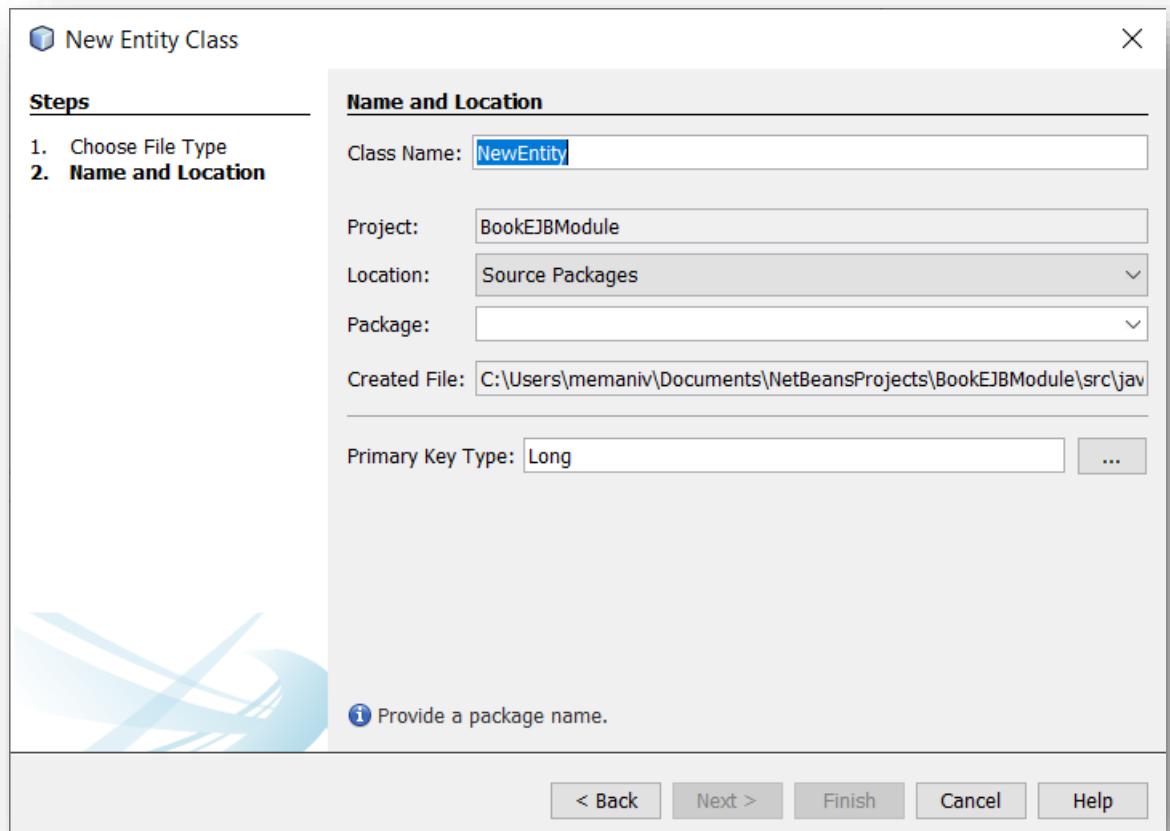
The screenshot shows a Java code editor window titled "Author.java". The "Source" tab is selected. The code implements the Serializable interface and defines an Entity named "Author" with two columns: "FIRST_NAME" and "LAST_NAME". It includes constructors and a getter for the name.

```
1 package za.ac.tut.entities;
2
3 import java.io.Serializable;
4 import javax.persistence.Column;
5 import javax.persistence.Entity;
6 import javax.persistence.GeneratedValue;
7 import javax.persistence.GenerationType;
8 import javax.persistence.Id;
9 import javax.persistence.Table;
10
11 /**
12 * @author MemaniV
13 */
14
15 @Entity
16 @Table(name="AUTHOR_TBL")
17 public class Author implements Serializable {
18
19     private static final long serialVersionUID = 1L;
20     @Id
21     @GeneratedValue(strategy = GenerationType.AUTO)
22     private Long id;
23     @Column(name="FIRST_NAME", nullable=false,length=50)
24     private String name;
25     @Column(name="LAST_NAME", nullable=false,length=50)
26     private String surname;
27
28     public Author() {
29     }
30
31     public Author(String name, String surname) {
32         this.name = name;
33         this.surname = surname;
34     }
35
36     public String getName() {
37         return name;
38     }
}
```

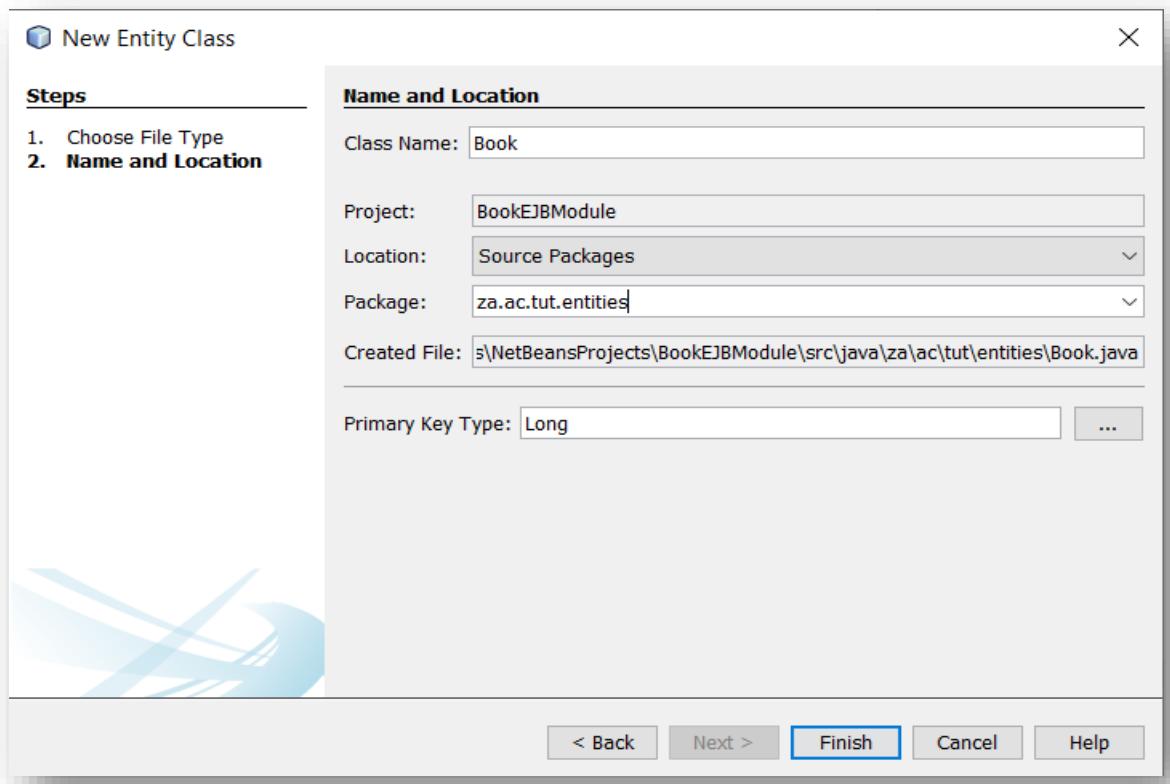
```
36     public String getName() {
37         return name;
38     }
39
40     public void setName(String name) {
41         this.name = name;
42     }
43
44     public String getSurname() {
45         return surname;
46     }
47
48     public void setSurname(String surname) {
49         this.surname = surname;
50     }
51
52     public Long getId() {
53         return id;
54     }
55
56     public void setId(Long id) {
57         this.id = id;
58     }
59
60     @Override
61     public int hashCode() {
62         int hash = 0;
63         hash += (id != null ? id.hashCode() : 0);
64         return hash;
65     }
```

```
67     @Override
68     public boolean equals(Object object) {
69         if (!(object instanceof Author)) {
70             return false;
71         }
72         Author other = (Author) object;
73         if ((this.id == null && other.id != null) ||
74             (this.id != null && !this.id.equals(other.id))) {
75             return false;
76         }
77         return true;
78     }
79
80     @Override
81     public String toString() {
82         return "za.ac.tut.entities.Author[ id=" + id + " ]";
83     }
84
85 }
```

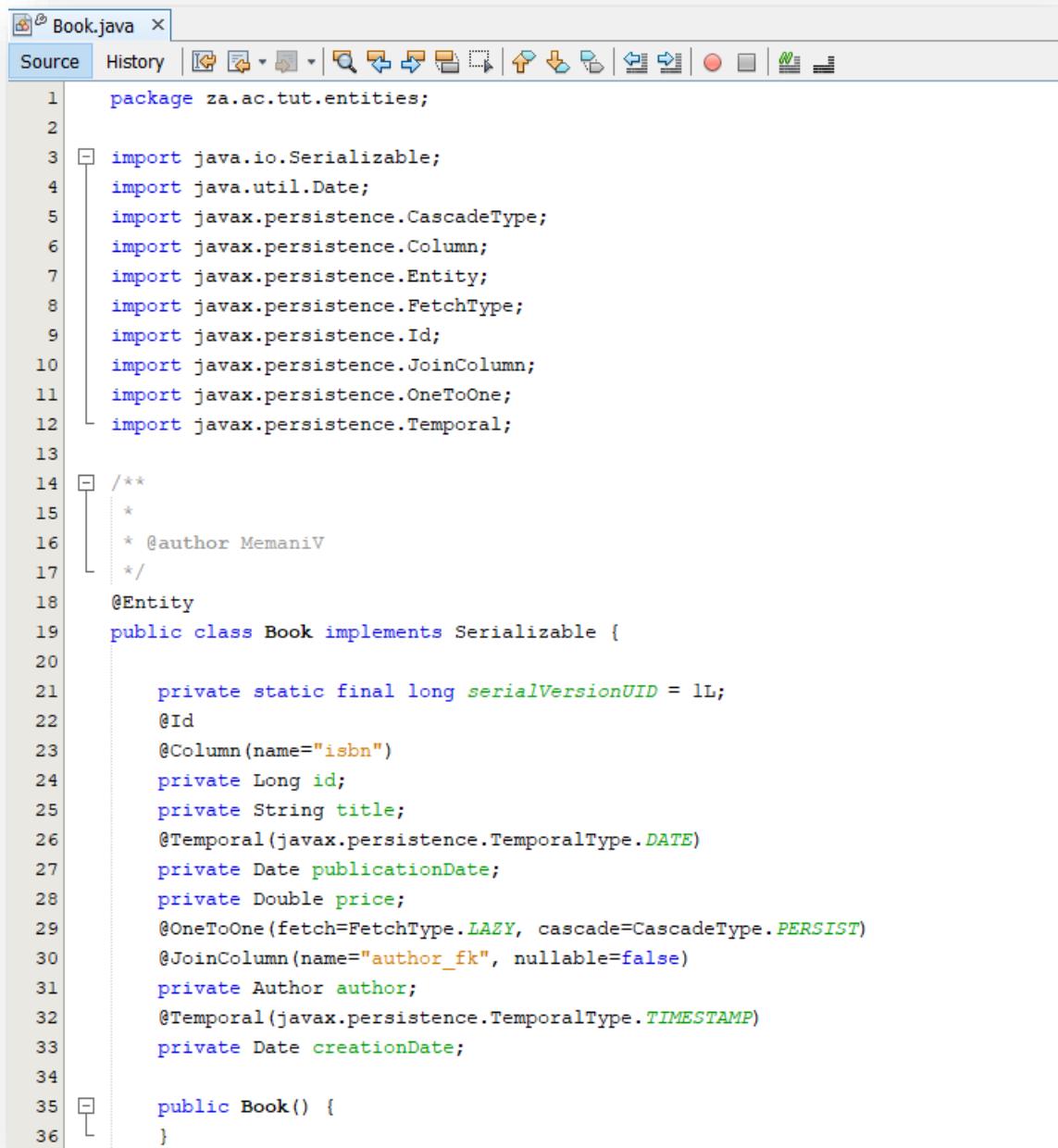
Right-click on the project and select New | Entity Class



Fill-in the form.



Click **Finish**.



The screenshot shows a Java code editor window with the tab 'Book.java' selected. The code is a Java class named 'Book' that implements the 'Serializable' interface. It includes annotations for persistence and relationships. The code is numbered from 1 to 36.

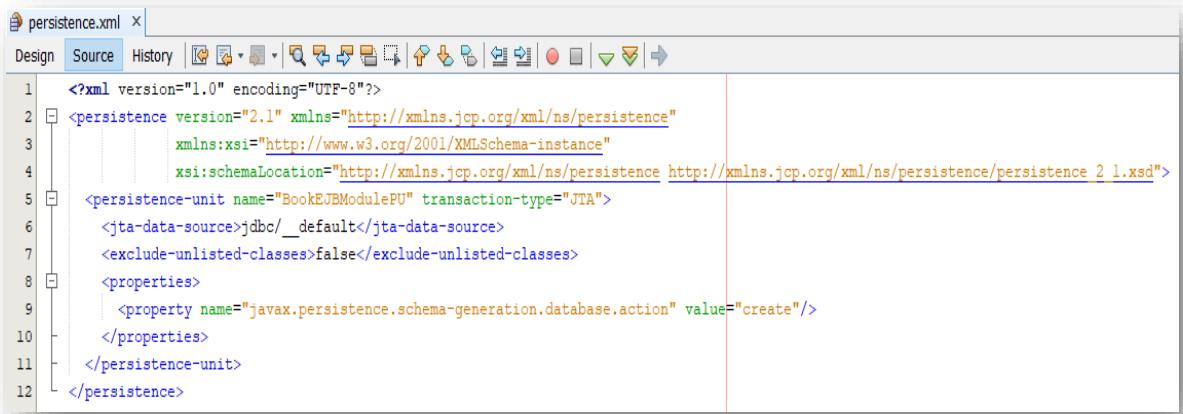
```
1 package za.ac.tut.entities;
2
3 import java.io.Serializable;
4 import java.util.Date;
5 import javax.persistence.CascadeType;
6 import javax.persistence.Column;
7 import javax.persistence.Entity;
8 import javax.persistence.FetchType;
9 import javax.persistence.Id;
10 import javax.persistence.JoinColumn;
11 import javax.persistence.OneToOne;
12 import javax.persistence.Temporal;
13
14 /**
15 * 
16 * @author MemaniV
17 */
18 @Entity
19 public class Book implements Serializable {
20
21     private static final long serialVersionUID = 1L;
22     @Id
23     @Column(name="isbn")
24     private Long id;
25     private String title;
26     @Temporal(javax.persistence.TemporalType.DATE)
27     private Date publicationDate;
28     private Double price;
29     @OneToOne(fetch=FetchType.LAZY, cascade=CascadeType.PERSIST)
30     @JoinColumn(name="author_fk", nullable=false)
31     private Author author;
32     @Temporal(javax.persistence.TemporalType.TIMESTAMP)
33     private Date creationDate;
34
35     public Book() {
36 }
```

```
38     public Book(Long id, String title, Date publicationDate, Double price,
39     Author author, Date creationDate) {
40         this.id = id;
41         this.title = title;
42         this.publicationDate = publicationDate;
43         this.price = price;
44         this.author = author;
45         this.creationDate = creationDate;
46     }
47
48     public String getTitle() {
49         return title;
50     }
51
52     public void setTitle(String title) {
53         this.title = title;
54     }
55
56     public Date getPublicationDate() {
57         return publicationDate;
58     }
59
60     public void setPublicationDate(Date publicationDate) {
61         this.publicationDate = publicationDate;
62     }
63
64     public Double getPrice() {
65         return price;
66     }
67
68     public void setPrice(Double price) {
69         this.price = price;
70     }
71
72     public Author getAuthor() {
73         return author;
74     }
```

```
76  public void setAuthor(Author author) {  
77      this.author = author;  
78  }  
79  
80  public Date getCreationDate() {  
81      return creationDate;  
82  }  
83  
84  public void setCreationDate(Date creationDate) {  
85      this.creationDate = creationDate;  
86  }  
87  
88  public Long getId() {  
89      return id;  
90  }  
91  
92  public void setId(Long id) {  
93      this.id = id;  
94  }  
95  
96  @Override  
97  public int hashCode() {  
98      int hash = 0;  
99      hash += (id != null ? id.hashCode() : 0);  
100     return hash;  
101 }
```

```
103  @Override  
104  public boolean equals(Object object) {  
105      if (!(object instanceof Book)) {  
106          return false;  
107      }  
108      Book other = (Book) object;  
109      if ((this.id == null && other.id != null) ||  
110          (this.id != null && !this.id.equals(other.id))) {  
111          return false;  
112      }  
113      return true;  
114  }  
115  
116  @Override  
117  public String toString() {  
118      return "za.ac.tut.entities.Book[ id=" + id + " ]";  
119  }  
120  
121 }
```

View the **persistence.xml** file.

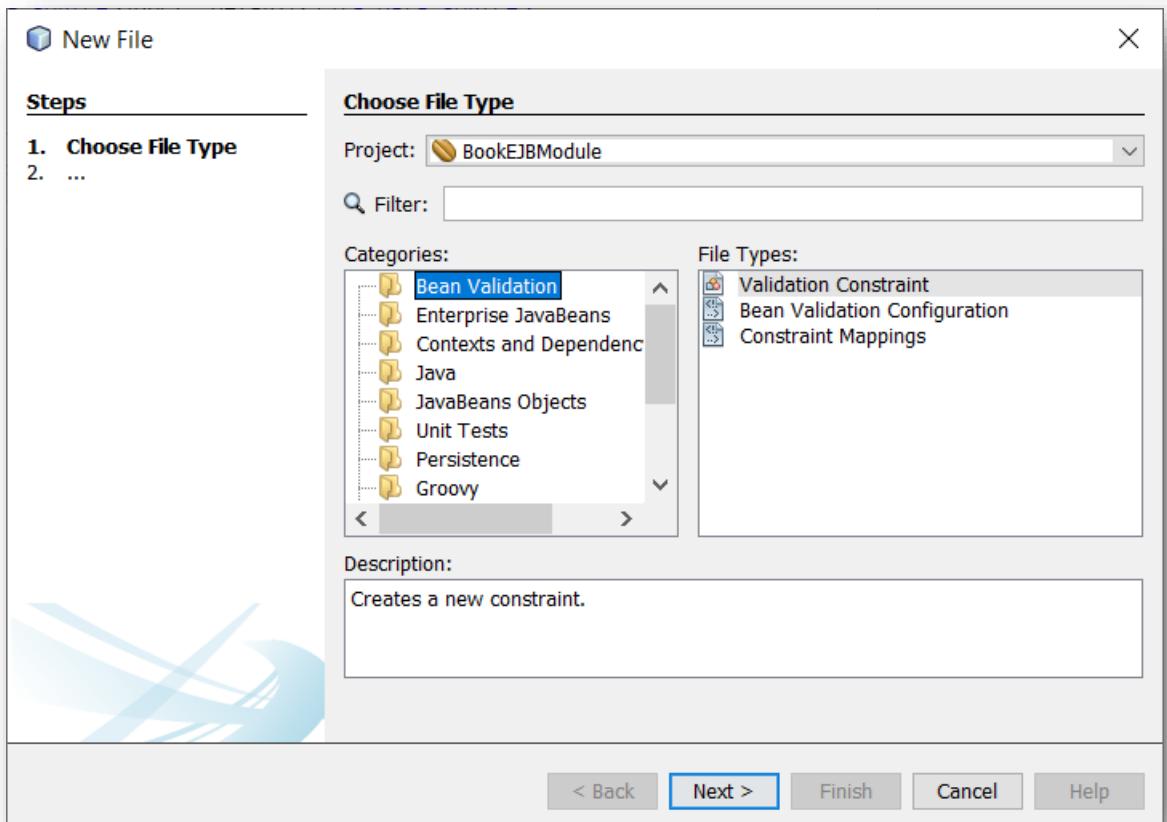


The screenshot shows the Eclipse IDE interface with the title bar "persistence.xml". The "Source" tab is selected. The code editor displays the XML configuration for a persistence unit:

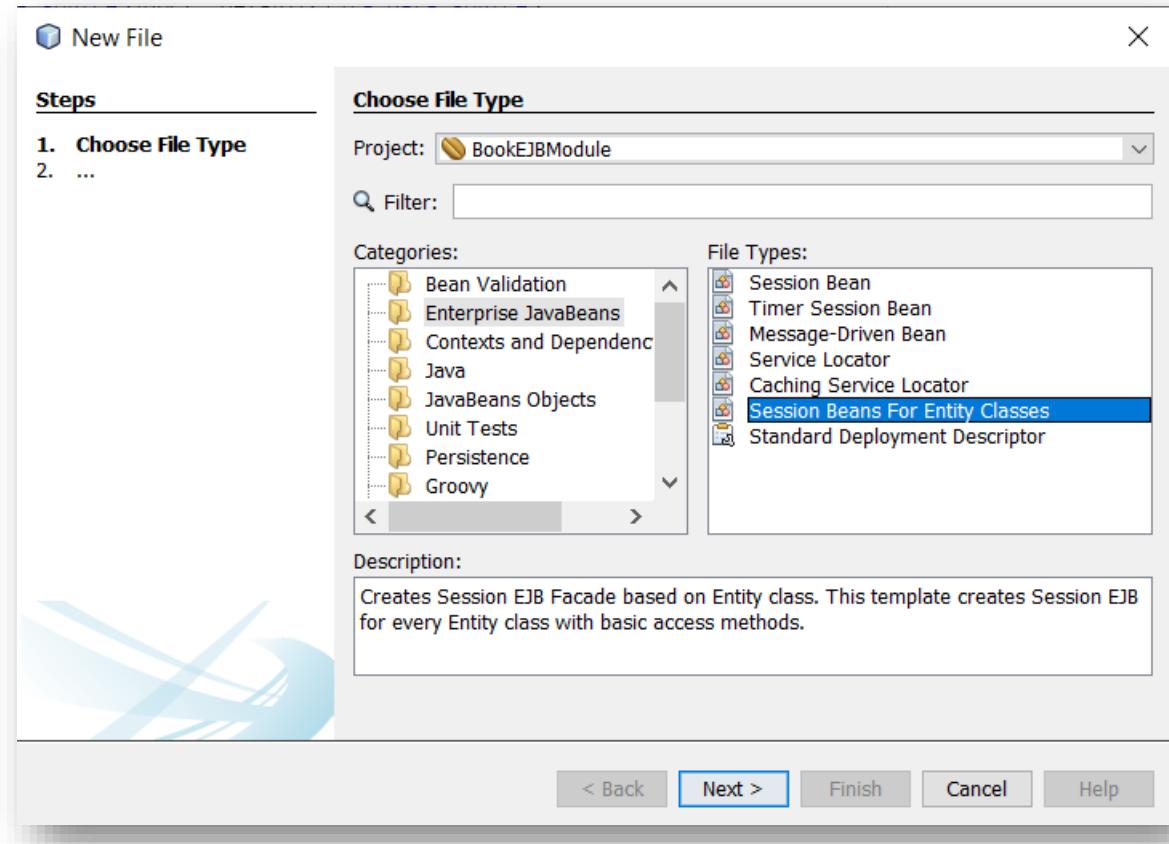
```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
    <persistence-unit name="BookEJBModulePU" transaction-type="JTA">
        <jta-data-source>jdbc/__default</jta-data-source>
        <exclude-unlisted-classes>false</exclude-unlisted-classes>
        <properties>
            <property name="javax.persistence.schema-generation.database.action" value="create"/>
        </properties>
    </persistence-unit>
</persistence>
```

Create business code (CRUD operations) for the entity. Perform the following steps:

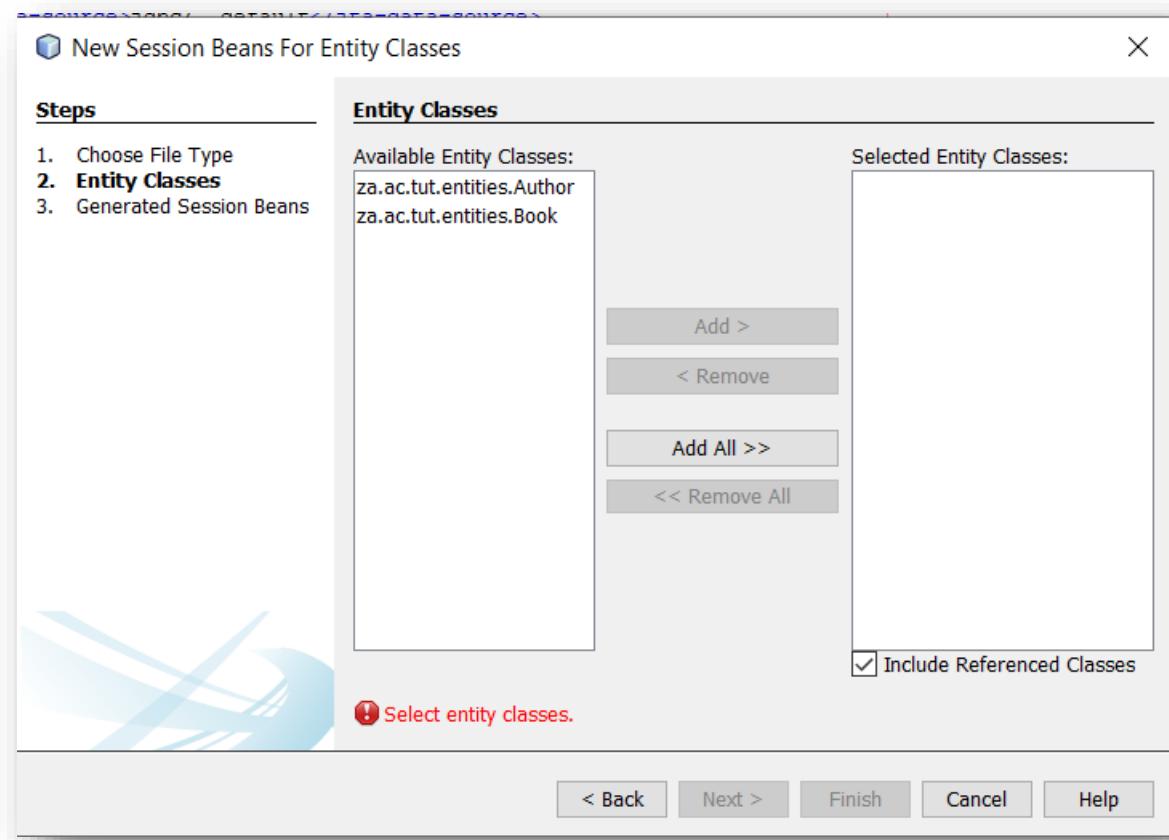
- ✓ Right-click on the project and select **New | Other**.



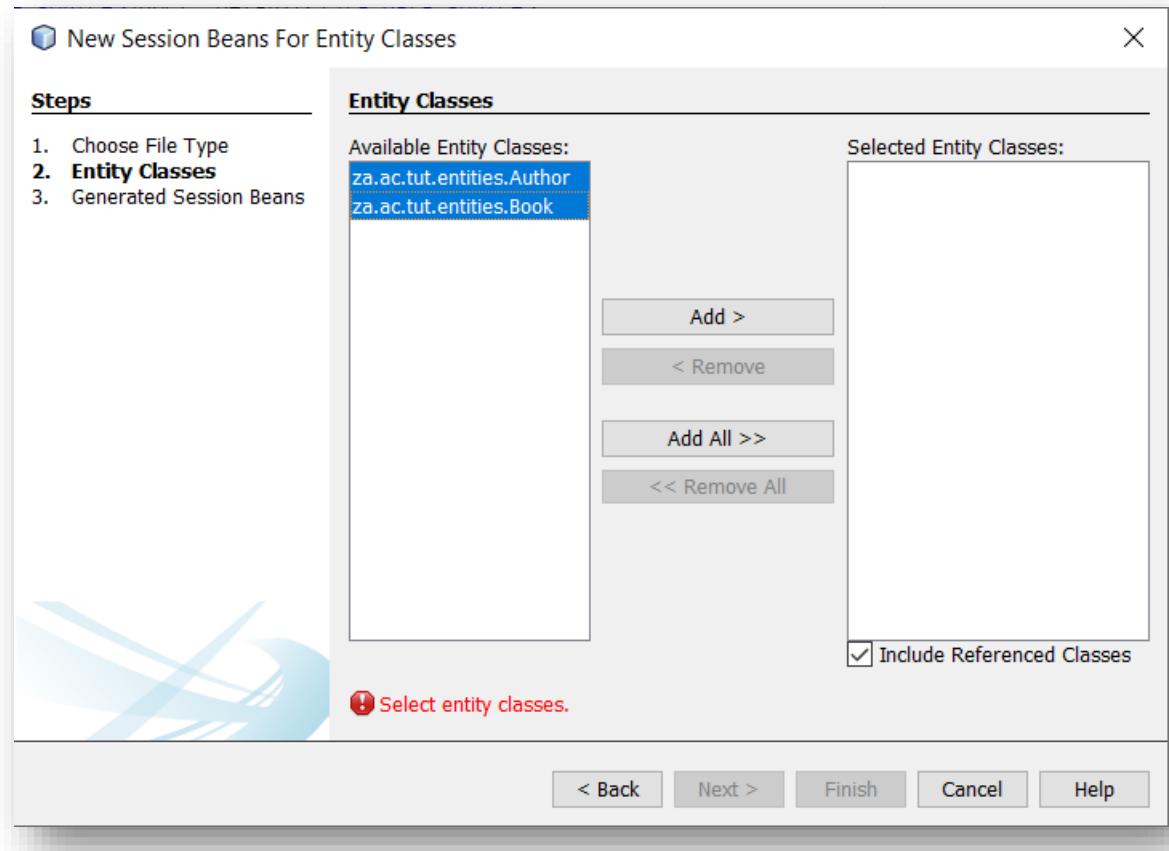
- ✓ Under **Categories**, select **Enterprise JavaBeans**. Under **File Types** select **Session Beans for Entity Classes**.



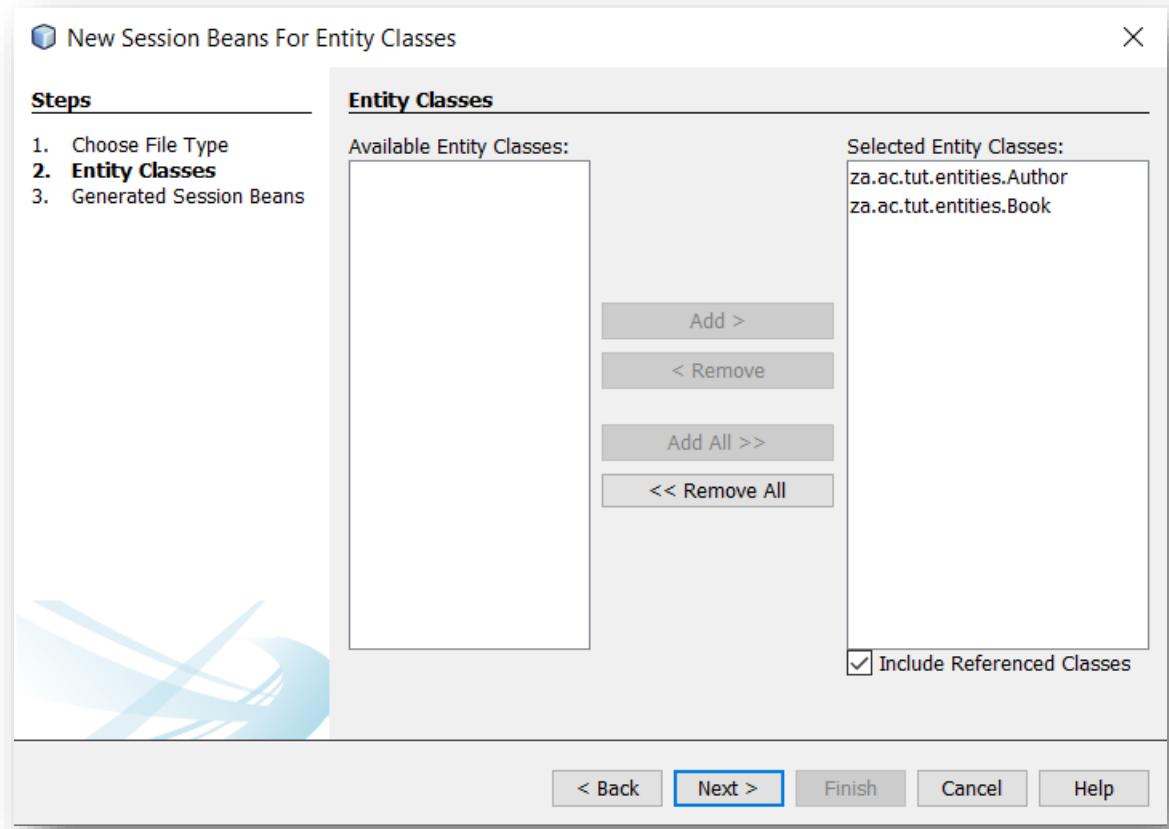
- ✓ Click **Next**.



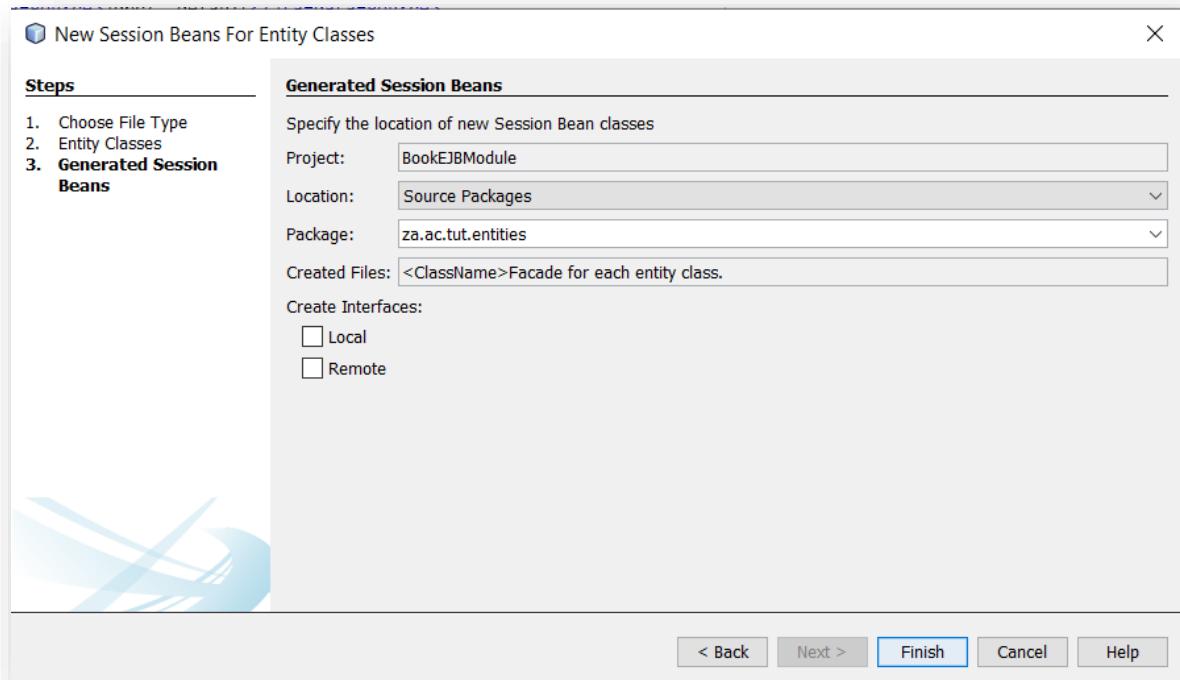
- ✓ Select **all** the entities under Available Entity Classes.



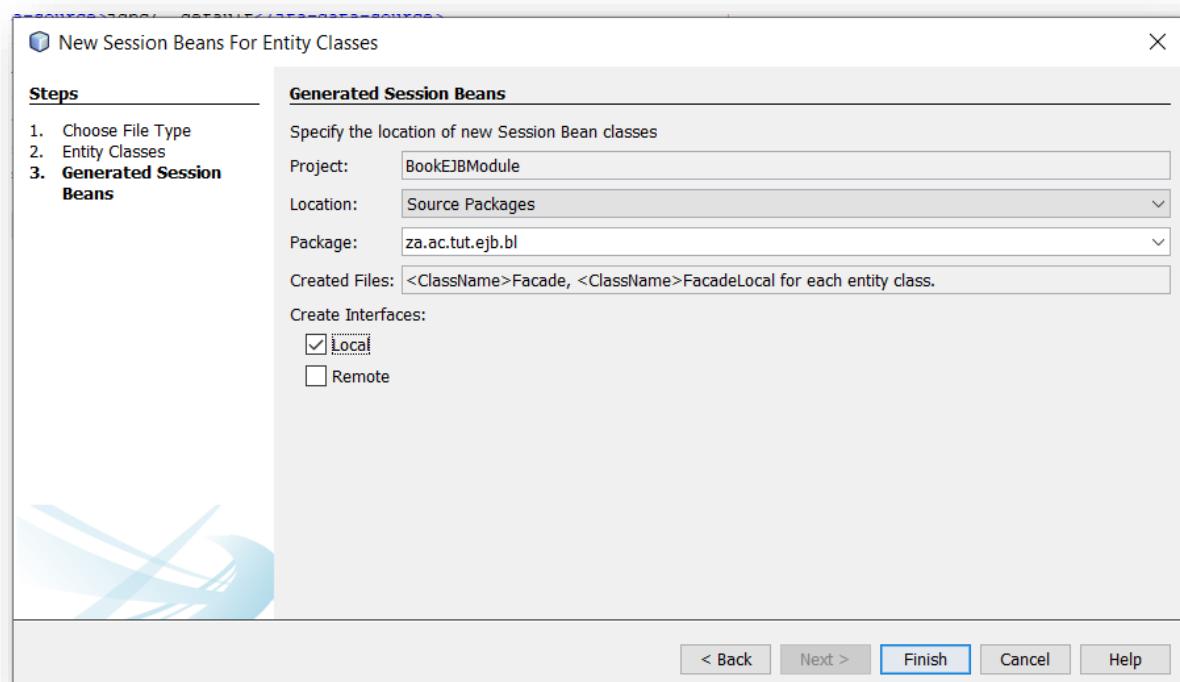
- ✓ Click **Add**.



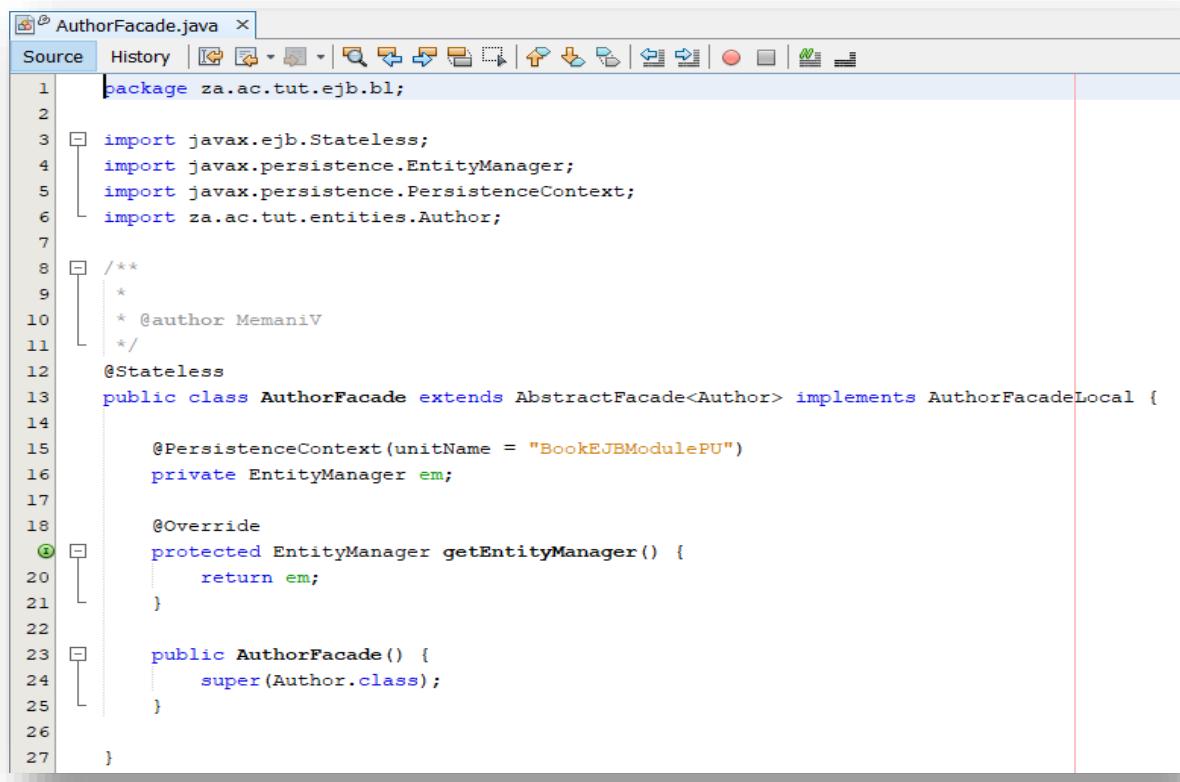
✓ Click **Next**.



✓ Modify the package name to **za.ac.tut.ejb.bl**, where **bl** stands for **business logic**. Also select **Local** interface.



- ✓ Click **Finish**. View created files.
- **AuthorFacade**.

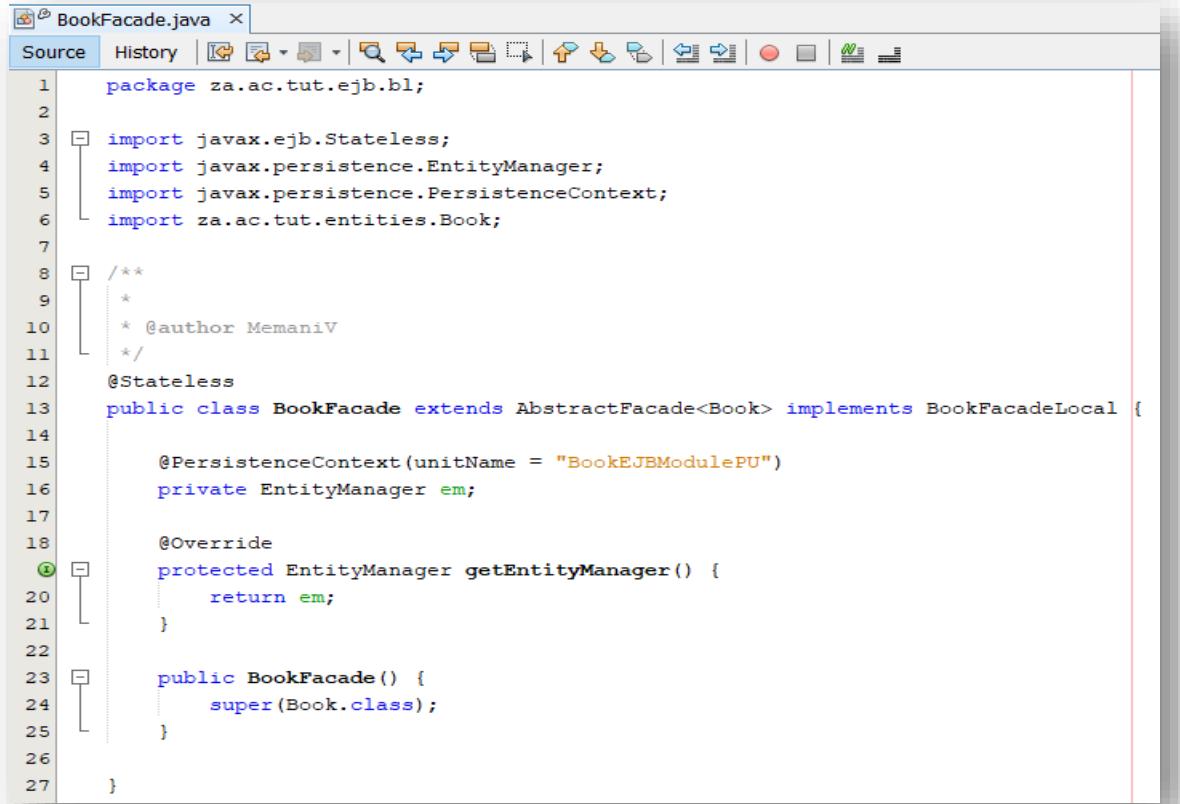


```

1 package za.ac.tut.ejb.bl;
2
3 import javax.ejb.Stateless;
4 import javax.persistence.EntityManager;
5 import javax.persistence.PersistenceContext;
6 import za.ac.tut.entities.Author;
7
8 /**
9  * @author MemaniV
10 */
11 @Stateless
12 public class AuthorFacade extends AbstractFacade<Author> implements AuthorFacadeLocal {
13
14     @PersistenceContext(unitName = "BookEJBModulePU")
15     private EntityManager em;
16
17     @Override
18     protected EntityManager getEntityManager() {
19         return em;
20     }
21
22     public AuthorFacade() {
23         super(Author.class);
24     }
25
26 }
27

```

- **BookFacade**



```

1 package za.ac.tut.ejb.bl;
2
3 import javax.ejb.Stateless;
4 import javax.persistence.EntityManager;
5 import javax.persistence.PersistenceContext;
6 import za.ac.tut.entities.Book;
7
8 /**
9  * @author MemaniV
10 */
11 @Stateless
12 public class BookFacade extends AbstractFacade<Book> implements BookFacadeLocal {
13
14     @PersistenceContext(unitName = "BookEJBModulePU")
15     private EntityManager em;
16
17     @Override
18     protected EntityManager getEntityManager() {
19         return em;
20     }
21
22     public BookFacade() {
23         super(Book.class);
24     }
25
26 }
27

```

▪ AbstractFacade.

```

1 package za.ac.tut.bl;
2
3 import java.util.List;
4 import javax.persistence.EntityManager;
5
6 /**
7 *
8 * @author MemaniV
9 */
10 public abstract class AbstractFacade<T> {
11
12     private Class<T> entityClass;
13
14     public AbstractFacade(Class<T> entityClass) {
15         this.entityClass = entityClass;
16     }
17
18     protected abstract EntityManager getEntityManager();
19
20     public void create(T entity) {
21         getEntityManager().persist(entity);
22     }
23
24     public void edit(T entity) {
25         getEntityManager().merge(entity);
26     }
27
28     public void remove(T entity) {
29         getEntityManager().remove(getEntityManager().merge(entity));
30     }
31

```

```

32     public T find(Object id) {
33         return getEntityManager().find(entityClass, id);
34     }
35
36     public List<T> findAll() {
37         javax.persistence.criteria.CriteriaQuery cq =
38             getEntityManager().getCriteriaBuilder().createQuery();
39         cq.select(cq.from(entityClass));
40         return getEntityManager().createQuery(cq).getResultList();
41     }
42
43     public List<T> findRange(int[] range) {
44         javax.persistence.criteria.CriteriaQuery cq =
45             getEntityManager().getCriteriaBuilder().createQuery();
46         cq.select(cq.from(entityClass));
47         javax.persistence.Query q = getEntityManager().createQuery(cq);
48         q.setMaxResults(range[1] - range[0] + 1);
49         q.setFirstResult(range[0]);
50         return q.getResultList();
51     }
52
53     public int count() {
54         javax.persistence.criteria.CriteriaQuery cq =
55             getEntityManager().getCriteriaBuilder().createQuery();
56         javax.persistence.criteria.Root<T> rt = cq.from(entityClass);
57         cq.select(getEntityManager().getCriteriaBuilder().count(rt));
58         javax.persistence.Query q = getEntityManager().createQuery(cq);
59         return ((Long) q.getSingleResult()).intValue();
60     }
61
62 }

```

▪ **AuthorPersonFacadeLocal**.

The screenshot shows a Java code editor with the following details:

- Title Bar:** The title bar displays "AuthorFacadeLocal.java" with a file icon.
- Toolbar:** A standard toolbar with icons for file operations like Open, Save, Print, and Find.
- Code Area:** The main area contains the Java code for the `AuthorFacadeLocal` interface. The code includes imports for `List`, `javax.ejb.Local`, and `Author`. It features a Javadoc block with a copyright notice and author information. The code defines methods for creating, editing, removing authors, finding an author by ID, finding all authors, finding authors within a range, and counting authors.

```
1 package za.ac.tut.ejb.bl;
2
3 import java.util.List;
4 import javax.ejb.Local;
5 import za.ac.tut.entities.Author;
6
7 /**
8 *
9 * @author MemaniV
10 */
11 @Local
12 public interface AuthorFacadeLocal {
13
14     void create(Author author);
15
16     void edit(Author author);
17
18     void remove(Author author);
19
20     Author find(Object id);
21
22     List<Author> findAll();
23
24     List<Author> findRange(int[] range);
25
26     int count();
27
28 }
```

- **BookFacadeLocal.**

The screenshot shows a Java code editor with the following details:

- Title Bar:** The title bar displays "BookFacadeLocal.java" and a close button.
- Toolbar:** A standard toolbar with icons for file operations like Open, Save, Print, and a magnifying glass for search.
- Source Tab:** The "Source" tab is selected, indicating the current view of the code.
- Code Area:** The main area contains the Java code for the `BookFacadeLocal` interface. The code includes imports for `List`, `Local`, and `Book`. It features Javadoc comments, annotations (`@Local`), and several methods: `create`, `edit`, `remove`, `find`, `findAll`, `findRange`, and `count`.

```
1 package za.ac.tut.ejb.bl;
2
3 import java.util.List;
4 import javax.ejb.Local;
5 import za.ac.tut.entities.Book;
6
7 /**
8 * @author MemaniV
9 */
10
11 @Local
12 public interface BookFacadeLocal {
13
14     void create(Book book);
15
16     void edit(Book book);
17
18     void remove(Book book);
19
20     Book find(Object id);
21
22     List<Book> findAll();
23
24     List<Book> findRange(int[] range);
25
26     int count();
27
28 }
```

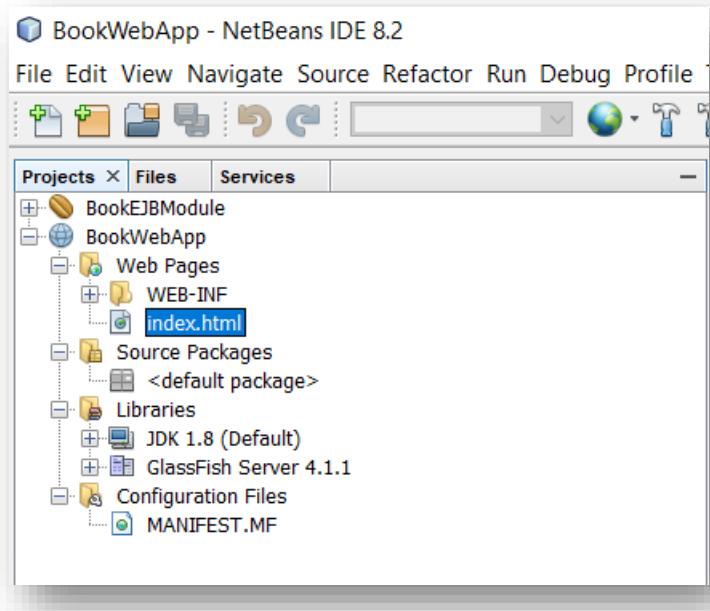
✓ Clean and build the EJB project.

The screenshot shows the NetBeans IDE interface. The top part is the Output window, which displays the build logs for the BookEJBModule. The logs show the compilation process, including warnings and notes about static metadata factories and optional files. It also indicates the creation of the dist directory and the final jar file, BookEJBModule.jar, located at C:\Users\memaniv\Documents\NetBeansProjects\BookEJBModule\dist\BookEJBModule.jar. The message "BUILD SUCCESSFUL (total time: 5 seconds)" is displayed at the end. Below the Output window is the main NetBeans window, titled "BookEJBModule - NetBeans IDE 8.2". The Projects tab in the left-hand navigation bar is selected, showing the structure of the BookEJBModule project. The "dist" folder contains the generated jar file "BookEJBModule.jar", which is expanded to show its contents: META-INF (containing MANIFEST.MF and persistence.xml), za.ac.tut.ejb.bl (containing AbstractFacade.class, AuthorFacade.class, AuthorFacadeLocal.class, BookFacade.class, BookFacadeLocal.class), and za.ac.tut.entities (containing Author.class, Author_.class, Book.class, Book_.class). Other project components shown include nbproject, src, test, and build.xml.

```
Java DB Database Process × GlassFish Server 4.1.1 × BookEJBModule (clean,dist) ×
warning: Supported source version RELEASE_8 from annotation processor org.eclipse.persistence...
Note: Creating static metadata factory ...
Note: Optional file was not found: META-INF/orm.xml continuing with generation.
Note: Optional file was not found: META-INF/eclipselink-orm.xml continuing with generation.
Note: Optional file was not found: META-INF/orm.xml continuing with generation.
Note: Optional file was not found: META-INF/eclipselink-orm.xml continuing with generation.
Note: C:\Users\memaniv\Documents\NetBeansProjects\BookEJBModule\src\java\za\ac\tut\ejb\bl\AbstractFacade.java
Note: Recompile with -Xlint:unchecked for details.
1 warning
compile:
library-inclusion-in-archive:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\BookEJBModule\dist
Building jar: C:\Users\memaniv\Documents\NetBeansProjects\BookEJBModule\dist\BookEJBModule.jar
dist:
BUILD SUCCESSFUL (total time: 5 seconds)
```

Part D – Create a web client project.

Create a web client project called **BookWebApp**.

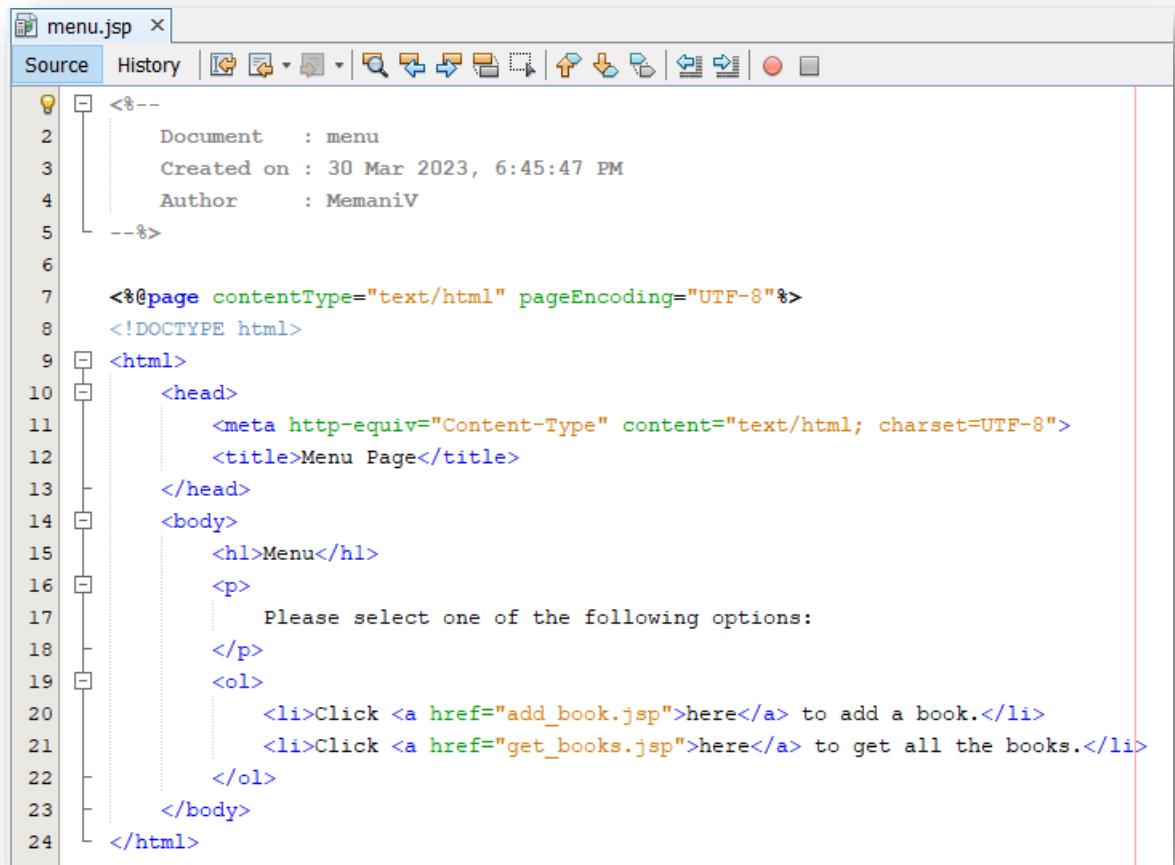


Edit the **index.html** page.

The screenshot shows the NetBeans code editor with the tab "index.html" selected. The toolbar above the editor includes tabs for Source, History, and several icons for file operations. The code editor displays the following HTML code:

```
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 page</h1>
15         <p>
16             Welcome to our web app. Please click <a href="menu.jsp">here</a> to start.
17         </p>
18     </body>
19 </html>
```

Create the menu.jsp file.



The screenshot shows a Java IDE interface with a tab labeled "menu.jsp". The "Source" tab is selected, displaying the following JSP code:

```
<%--  
1 Document      : menu  
2 Created on   : 30 Mar 2023, 6:45:47 PM  
3 Author        : MemaniV  
4 --%>  
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="add_book.jsp">here</a> to add a book.</li>  
21     <li>Click <a href="get_books.jsp">here</a> to get all the books.</li>  
22   </ol>  
23 </body>  
24 </html>
```

Create add_book.jsp file.

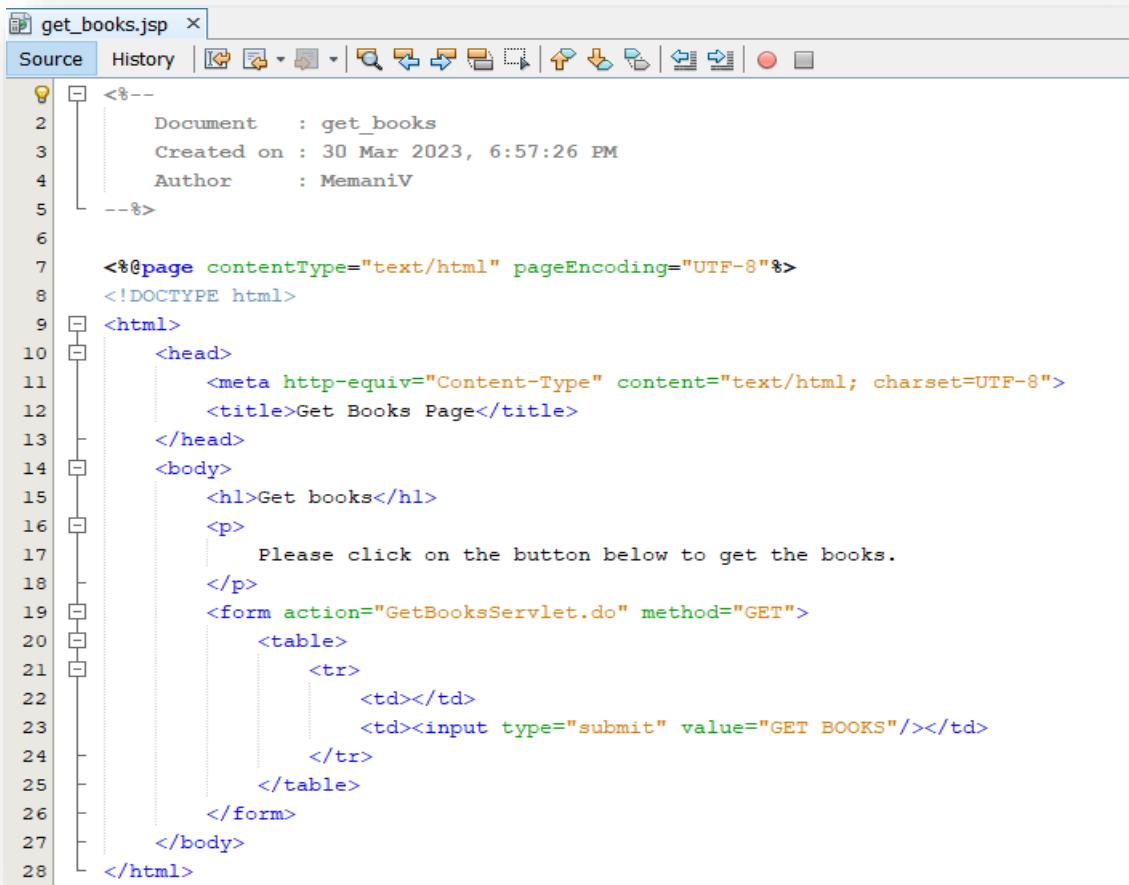
The screenshot shows the source code for a JSP file named 'add_book.jsp'. The code includes JSP declarations, page directives, and an HTML structure for displaying a form to add book details. The code is color-coded for syntax highlighting.

```
<%--  
Document      : add_book  
Created on   : 30 Mar 2023, 6:48:34 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>Add Book Page</title>  
    </head>  
    <body>  
        <h1>Add book</h1>  
        <p>  
            Please add book details below:  
        </p>
```

The screenshot shows the continuation of the 'add_book.jsp' source code, specifically the form definition. It uses a table structure with multiple rows to collect book information such as ISBN, Title, Price, Publication date, Author name, Author surname, and a submit button.

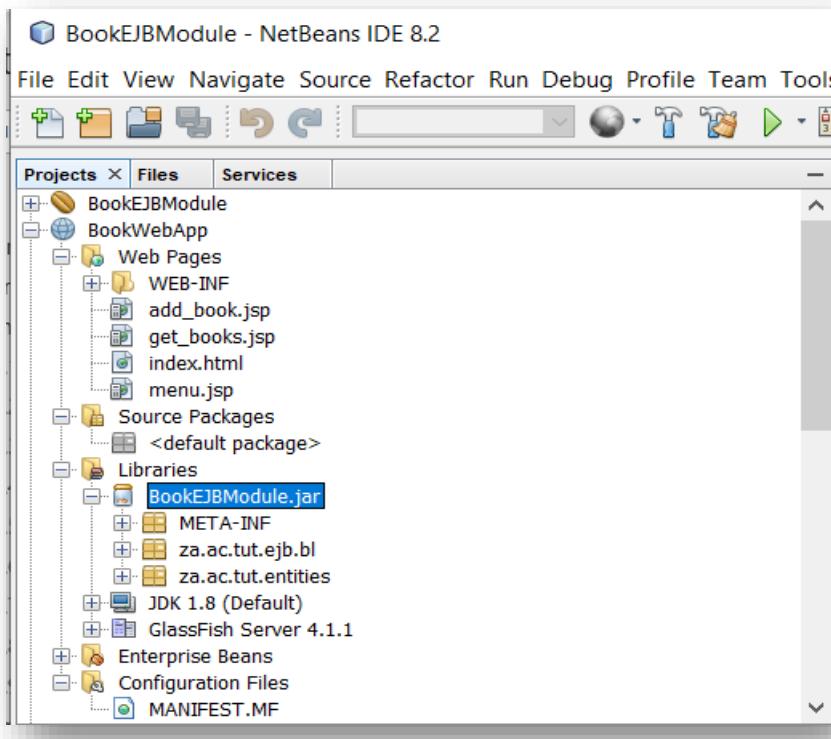
```
19 |     <form action="AddBookServlet.do" method="POST">  
20 |         <table>  
21 |             <tr>  
22 |                 <td>ISBN: </td>  
23 |                 <td><input type="text" name="isbn"/></td>  
24 |             </tr>  
25 |             <tr>  
26 |                 <td>Title: </td>  
27 |                 <td><input type="text" name="title"/></td>  
28 |             </tr>  
29 |             <tr>  
30 |                 <td>Price: </td>  
31 |                 <td><input type="text" name="price"/></td>  
32 |             </tr>  
33 |             <tr>  
34 |                 <td>Publication date: </td>  
35 |                 <td><input type="date" name="publicationDate"/></td>  
36 |             </tr>  
37 |             <tr>  
38 |                 <td>Author name: </td>  
39 |                 <td><input type="text" name="authorName"/></td>  
40 |             </tr>  
41 |             <tr>  
42 |                 <td>Author surname: </td>  
43 |                 <td><input type="text" name="authorSurname"/></td>  
44 |             </tr>  
45 |             <tr>  
46 |                 <td></td>  
47 |                 <td><input type="submit" value="ADD BOOK"/></td>  
48 |             </tr>  
49 |         </table>  
50 |     </form>  
51 | </body>  
52 | </html>
```

Create the get_books.jsp file.

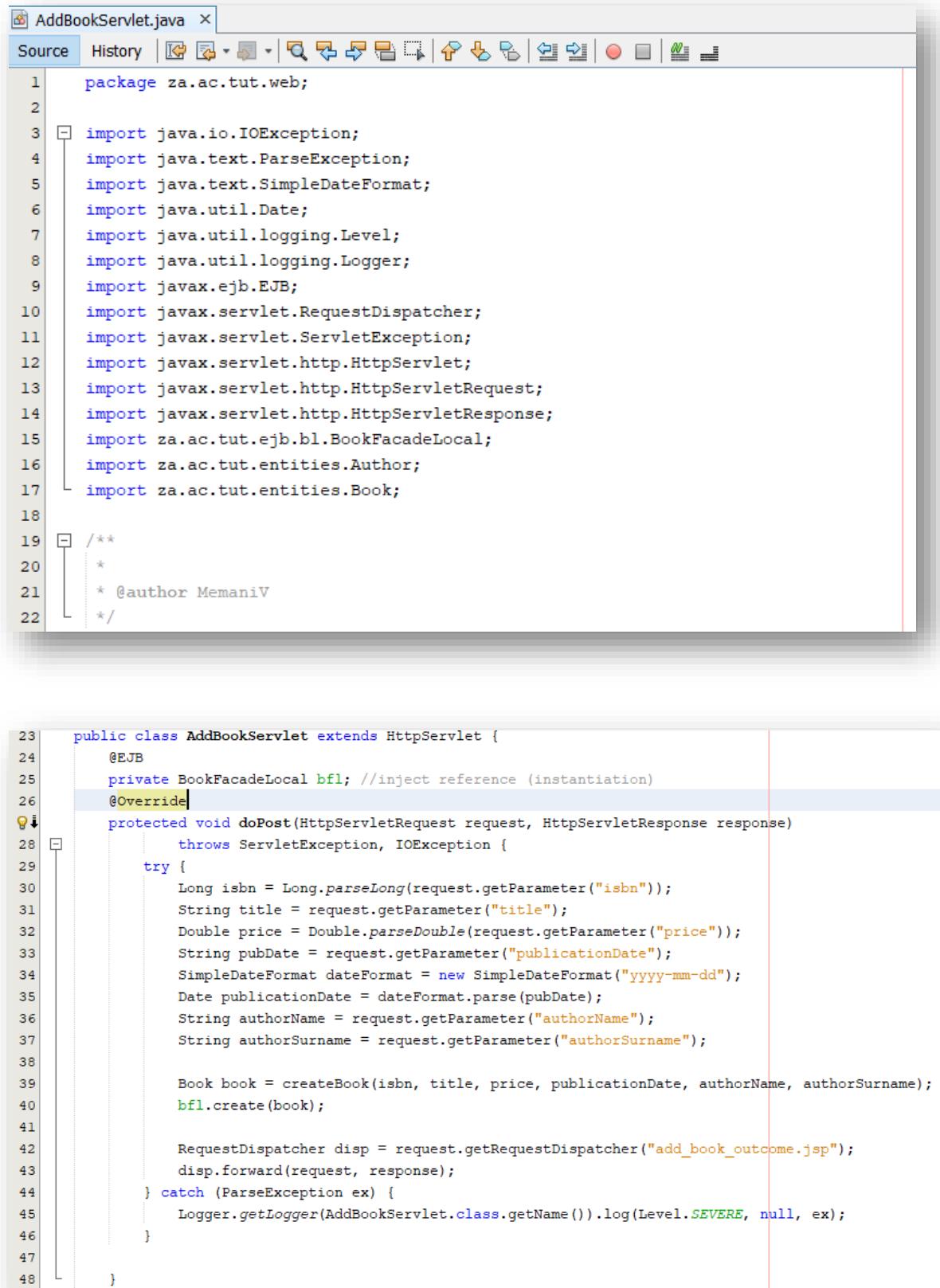


```
<%--  
1 Document      : get_books  
2 Created on   : 30 Mar 2023, 6:57:26 PM  
3 Author        : MemaniV  
4--%>  
  
5  
6  
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>  
8 <!DOCTYPE html>  
9 <html>  
10 <head>  
11     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
12     <title>Get Books Page</title>  
13 </head>  
14 <body>  
15     <h1>Get books</h1>  
16     <p>  
17         Please click on the button below to get the books.  
18     </p>  
19     <form action="GetBooksServlet.do" method="GET">  
20         <table>  
21             <tr>  
22                 <td></td>  
23                 <td><input type="submit" value="GET BOOKS"/></td>  
24             </tr>  
25         </table>  
26     </form>  
27 </body>  
28 </html>
```

Add the **BookEJBModule.jar** file to the library of **BookWebApp**.



Create the **AddBookServlet.java** file.

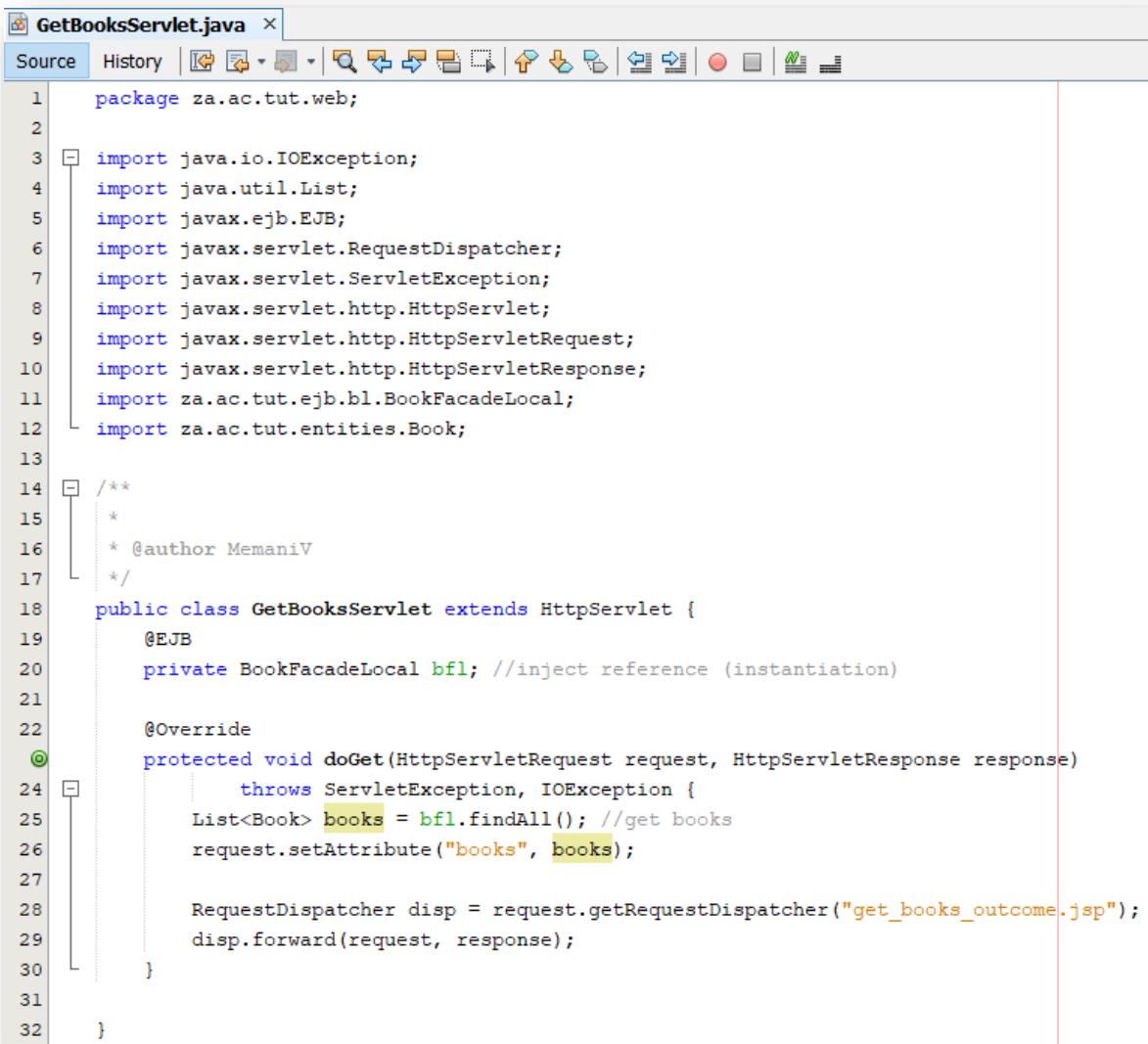


The screenshot shows a Java code editor with the title bar "AddBookServlet.java". The menu bar includes "Source", "History", and various tool icons. The code is divided into two sections:

```
1 package za.ac.tut.web;
2
3 import java.io.IOException;
4 import java.text.ParseException;
5 import java.text.SimpleDateFormat;
6 import java.util.Date;
7 import java.util.logging.Level;
8 import java.util.logging.Logger;
9 import javax.ejb.EJB;
10 import javax.servlet.RequestDispatcher;
11 import javax.servlet.ServletException;
12 import javax.servlet.http.HttpServlet;
13 import javax.servlet.http.HttpServletRequest;
14 import javax.servlet.http.HttpServletResponse;
15 import za.ac.tut.ejb.bl.BookFacadeLocal;
16 import za.ac.tut.entities.Author;
17 import za.ac.tut.entities.Book;
18
19 /**
20 *
21 * @author MemaniV
22 */
23
24 public class AddBookServlet extends HttpServlet {
25     @EJB
26     private BookFacadeLocal bfl; //inject reference (instantiation)
27     @Override
28     protected void doPost(HttpServletRequest request, HttpServletResponse response)
29             throws ServletException, IOException {
30         try {
31             Long isbn = Long.parseLong(request.getParameter("isbn"));
32             String title = request.getParameter("title");
33             Double price = Double.parseDouble(request.getParameter("price"));
34             String pubDate = request.getParameter("publicationDate");
35             SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-mm-dd");
36             Date publicationDate = dateFormat.parse(pubDate);
37             String authorName = request.getParameter("authorName");
38             String authorSurname = request.getParameter("authorSurname");
39
40             Book book = createBook(isbn, title, price, publicationDate, authorName, authorSurname);
41             bfl.create(book);
42
43             RequestDispatcher disp = request.getRequestDispatcher("add_book_outcome.jsp");
44             disp.forward(request, response);
45         } catch (ParseException ex) {
46             Logger.getLogger(AddBookServlet.class.getName()).log(Level.SEVERE, null, ex);
47         }
48     }
}
```

```
50     }
51     private Book createBook(Long isbn, String title, Double price, Date publicationDate, String authorName, String authorSurname) {
52         Book book = new Book();
53         Author author = new Author();
54         author.setName(authorName);
55         author.setSurname(authorSurname);
56         book.setId(isbn);
57         book.setTitle(title);
58         book.setPrice(price);
59         book.setPublicationDate(publicationDate);
60         book.setAuthor(author);
61         book.setCreationDate(new Date());
62         return book;
63     }
```

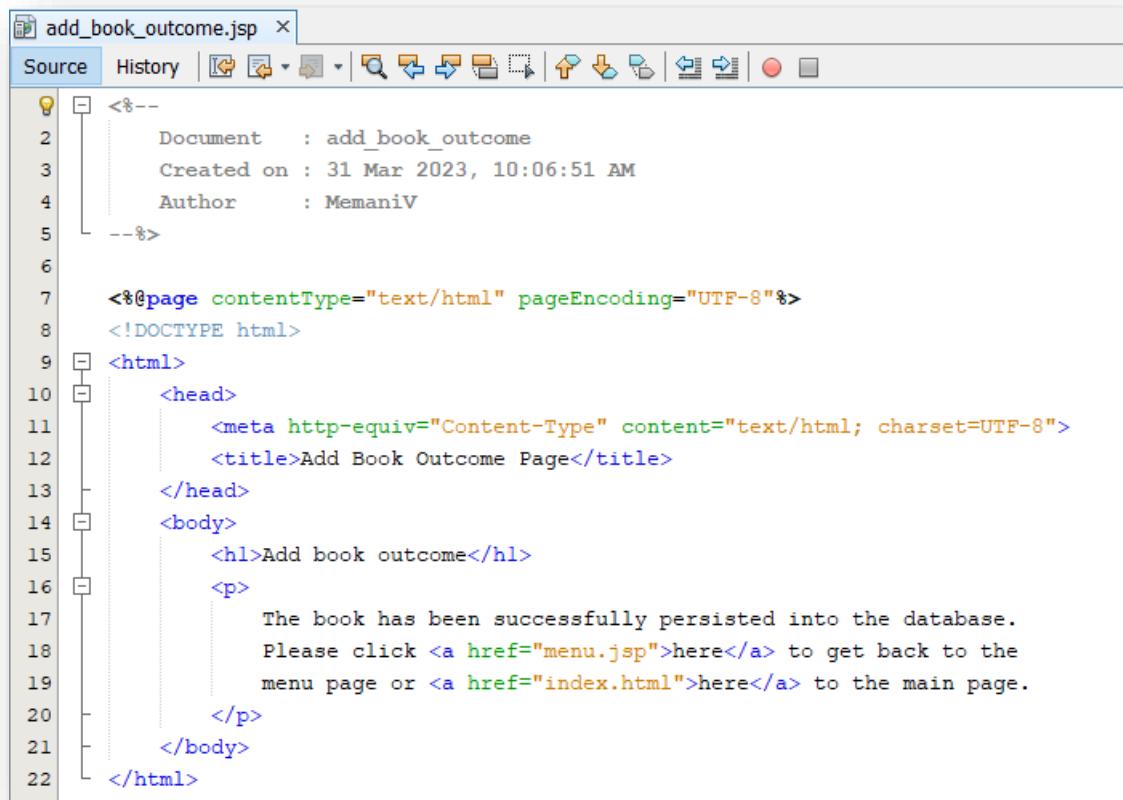
Create the **GetBooksServlet.java** file.



The screenshot shows an IDE interface with the tab "GetBooksServlet.java" selected. The code editor displays Java code for a servlet. The code includes imports for various Java and javax packages, a Javadoc comment, and an implementation of the doGet method. The variable 'books' and the call to 'request.setAttribute' are highlighted in yellow, indicating they are selected or being edited.

```
1 package za.ac.tut.web;
2
3 import java.io.IOException;
4 import java.util.List;
5 import javax.ejb.EJB;
6 import javax.servlet.RequestDispatcher;
7 import javax.servlet.ServletException;
8 import javax.servlet.http.HttpServlet;
9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11 import za.ac.tut.ejb.bl.BookFacadeLocal;
12 import za.ac.tut.entities.Book;
13
14 /**
15 * @author MemaniV
16 */
17 public class GetBooksServlet extends HttpServlet {
18     @EJB
19     private BookFacadeLocal bfl; //inject reference (instantiation)
20
21     @Override
22     protected void doGet(HttpServletRequest request, HttpServletResponse response)
23             throws ServletException, IOException {
24         List<Book> books = bfl.findAll(); //get books
25         request.setAttribute("books", books);
26
27         RequestDispatcher disp = request.getRequestDispatcher("get_books_outcome.jsp");
28         disp.forward(request, response);
29     }
30 }
31
32 }
```

Create the **add_book_outcome.jsp** file.



```
<%--  
1 Document      : add_book_outcome  
2 Created on   : 31 Mar 2023, 10:06:51 AM  
3 Author        : MemaniV  
4--%>  
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>Add Book Outcome Page</title>  
13 </head>  
14 <body>  
15     <h1>Add book outcome</h1>  
16     <p>  
17         The book has been successfully persisted into the database.  
18         Please click <a href="menu.jsp">here</a> to get back to the  
19         menu page or <a href="index.html">here</a> to the main page.  
20     </p>  
21 </body>  
22 </html>
```

Create the **get_books_outcome.jsp** file.

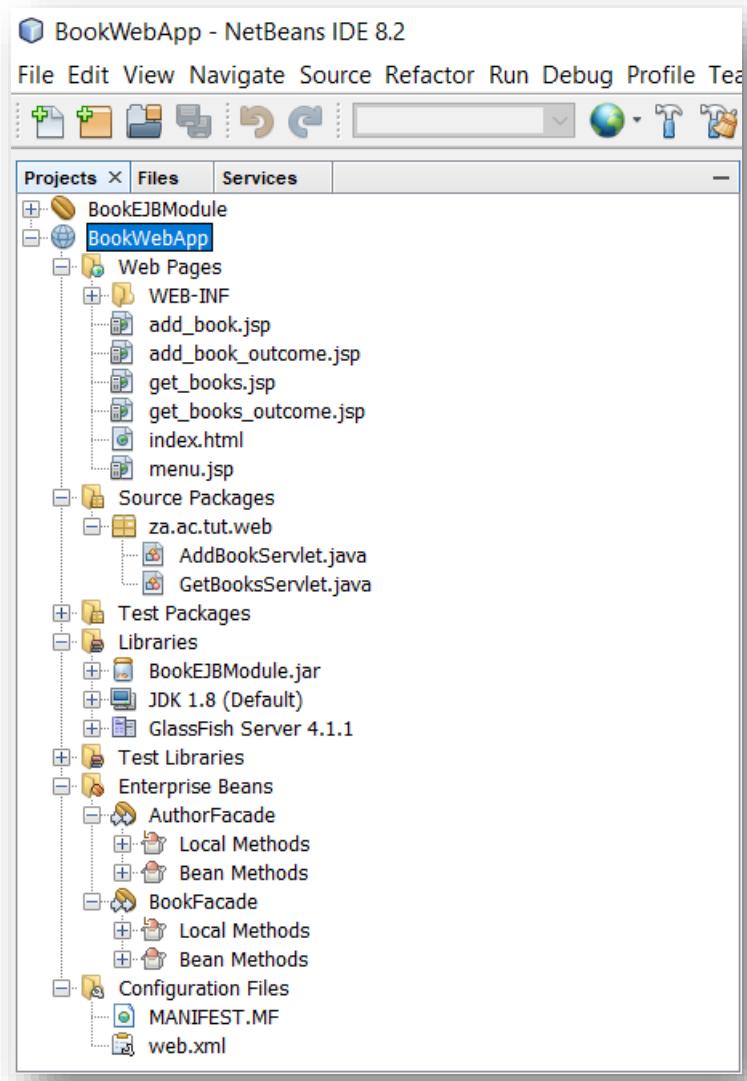
The screenshot shows a Java IDE interface with the title bar "get_books_outcome.jsp". The menu bar includes "Source", "History", and various tool icons. The code editor displays the JSP file content:

```
<%--  
1 Document      : get_books_outcome  
2 Created on   : 31 Mar 2023, 10:12:17 AM  
3 Author        : MemaniV  
4 --%>  
5  
6  
7 <%@page import="java.util.Date"%>  
8 <%@page import="java.util.List"%>  
9 <%@page import="za.ac.tut.entities.Book"%>  
10 <%@page contentType="text/html" pageEncoding="UTF-8"%>  
11 <!DOCTYPE html>  
12 <html>  
13     <head>  
14         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
15         <title>Get Books Outcome Page</title>  
16     </head>  
17     <body>  
18         <h1>Get books outcome</h1>  
19         <%  
20             List<Book> books = (List<Book>) request.getAttribute("books");  
21         %>  
22         <p>  
23             Below are the books retrieved from the database.  
24         </p>
```

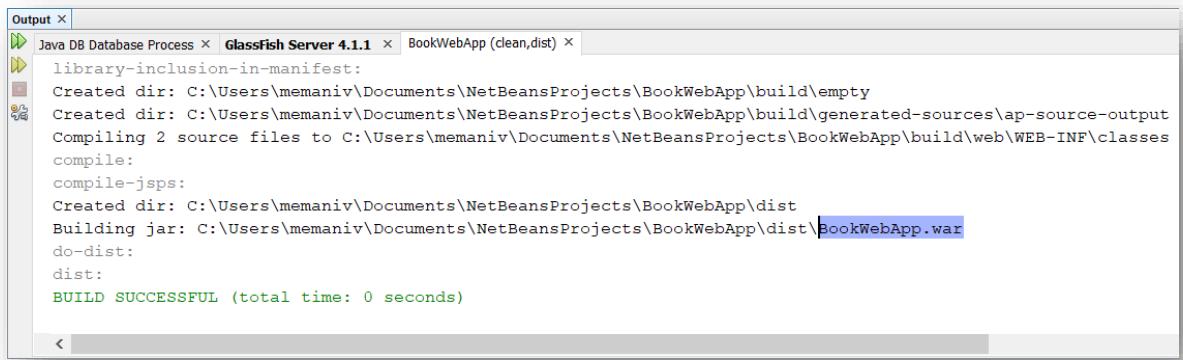
```
26 |     <table>
27 |         <%
28 |             for(int i = 0; i < books.size(); i++){
29 |                 Book p = books.get(i);
30 |                 Long id = p.getId();
31 |                 String title = p.getTitle();
32 |                 String name = p.getAuthor().getName();
33 |                 String surname = p.getAuthor().getSurname();
34 |                 Double price = p.getPrice();
35 |                 SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
36 |                 Date pubDate = p.getPublicationDate();
37 |                 String date = dateFormat.format(pubDate);
38 |             %>
39 |             <tr>
40 |                 <td>ISBN:</td>
41 |                 <td><%=id%></td>
42 |             </tr>
43 |             <tr>
44 |                 <td>Title:</td>
45 |                 <td><%=title%></td>
46 |             </tr>
47 |             <tr>
48 |                 <td>Author name:</td>
49 |                 <td><%=name%></td>
50 |             </tr>
51 |             <tr>
52 |                 <td>Author surname: </td>
53 |                 <td><%=surname%></td>
54 |             </tr>
55 |             <tr>
56 |                 <td>Price: </td>
57 |                 <td><%=price%></td>
58 |             </tr>
```

```
59 |             <tr>
60 |                 <td>Publication date: </td>
61 |                 <td><%=date%></td>
62 |             </tr>
63 |             <%
64 |             }
65 |             %>
66 |         </table>
67 |         <p>
68 |             Please click <a href="menu.jsp">here</a> to get back to the
69 |             menu page or <a href="index.html">here</a> to the main page.
70 |         </p>
71 |     </body>
72 | </html>
```

View the complete project structure of **BookWebApp**.

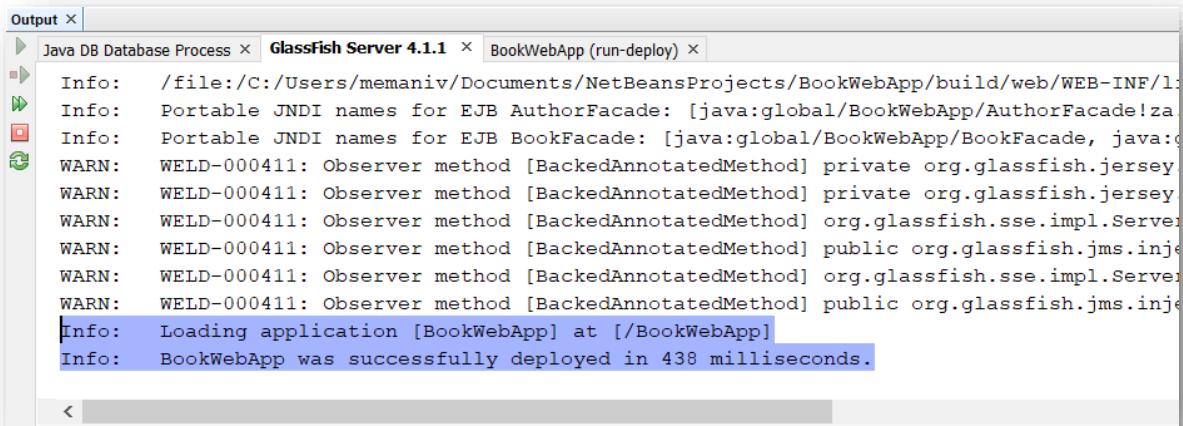


Compile the project.



```
Output X
Java DB Database Process X | GlassFish Server 4.1.1 X | BookWebApp (clean,dist) X
library-inclusion-in-manifest:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\BookWebApp\build\empty
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\BookWebApp\build\generated-sources\ap-source-output
Compiling 2 source files to C:\Users\memaniv\Documents\NetBeansProjects\BookWebApp\build\web\WEB-INF\classes
compile:
compile-jsp:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\BookWebApp\dist
Building jar: C:\Users\memaniv\Documents\NetBeansProjects\BookWebApp\dist\BookWebApp.war
do-dist:
dist:
BUILD SUCCESSFUL (total time: 0 seconds)
```

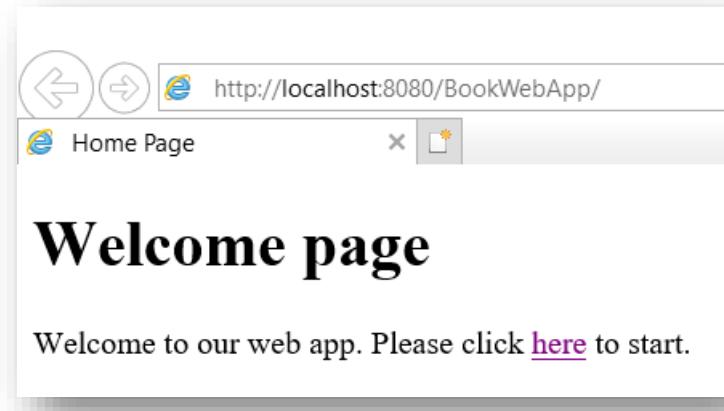
Deploy the project.



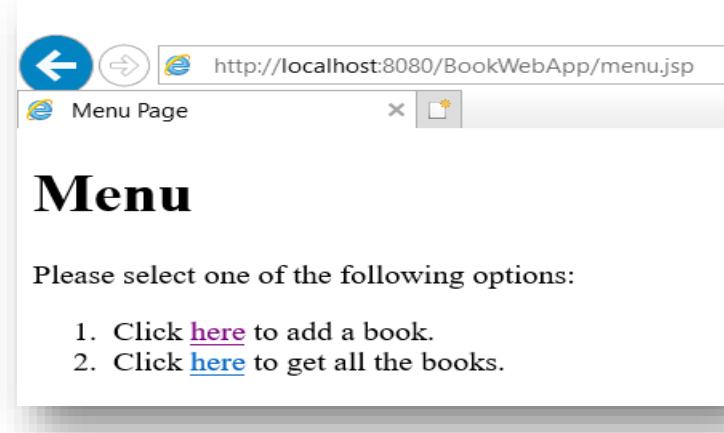
```
Output X
Java DB Database Process X | GlassFish Server 4.1.1 X | BookWebApp (run-deploy) X
Info: /file:/C:/Users/memaniv/Documents/NetBeansProjects/BookWebApp/build/web/WEB-INF/lib/
Info: Portable JNDI names for EJB AuthorFacade: [java:global/BookWebApp/AuthorFacade!za
Info: Portable JNDI names for EJB BookFacade: [java:global/BookWebApp/BookFacade, java:global/BookWebApp/BookFacade!za
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] private org.glassfish.jersey.media.multipart.providers.FormDataBodyWriter$FormDataProvider org.glassfish.jersey.media.multipart.providers.FormDataBodyWriter$FormDataProvider
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] private org.glassfish.jersey.media.multipart.providers.FormDataBodyWriter$FormDataProvider org.glassfish.jersey.media.multipart.providers.FormDataBodyWriter$FormDataProvider
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] org.glassfish.sse.impl.ServerSessionImpl org.glassfish.sse.impl.ServerSessionImpl
WARN: WELD-000411: observer method [BackedAnnotatedMethod] public org.glassfish.jms.injection.JMSContainer org.glassfish.jms.injection.JMSContainer
WARN: WELD-000411: observer method [BackedAnnotatedMethod] org.glassfish.sse.impl.ServerSessionImpl org.glassfish.sse.impl.ServerSessionImpl
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] public org.glassfish.jms.injection.JMSContainer org.glassfish.jms.injection.JMSContainer
Info: Loading application [BookWebApp] at [/BookWebApp]
Info: BookWebApp was successfully deployed in 438 milliseconds.
```

Part E – Run the web application.

Launch the application.



Click on the link



Add a book.

- ✓ Click on the first link.

A screenshot of a web browser window. The address bar shows the URL "http://localhost:8080/BookWebApp/add_book.jsp". The title bar says "Add Book Page". The main content area displays the text "Add book" in large bold letters, followed by "Please add book details below:". Below this, there are six input fields labeled "ISBN:", "Title:", "Price:", "Publication date:", "Author name:", and "Author surname:". At the bottom right is a "ADD BOOK" button.

- ✓ Fill in the form.

Please add book details below:

ISBN:	111
Title:	Java
Price:	123
Publication date:	2003-10-23
Author name:	Vuyisile
Author surname:	Memani

ADD BOOK

Click on the add button.

here to get back to the menu page or [here](#) to the main page.'"/>

Add book outcome

The book has been successfully persisted into the database. Please click [here](#) to get back to the menu page or [here](#) to the main page.

Add four more books. View all the books.

- ✓ Click on the second link.

Get books

Please click on the button below to get the books.

GET BOOKS

Click on the button.

Get Books Outcome Page

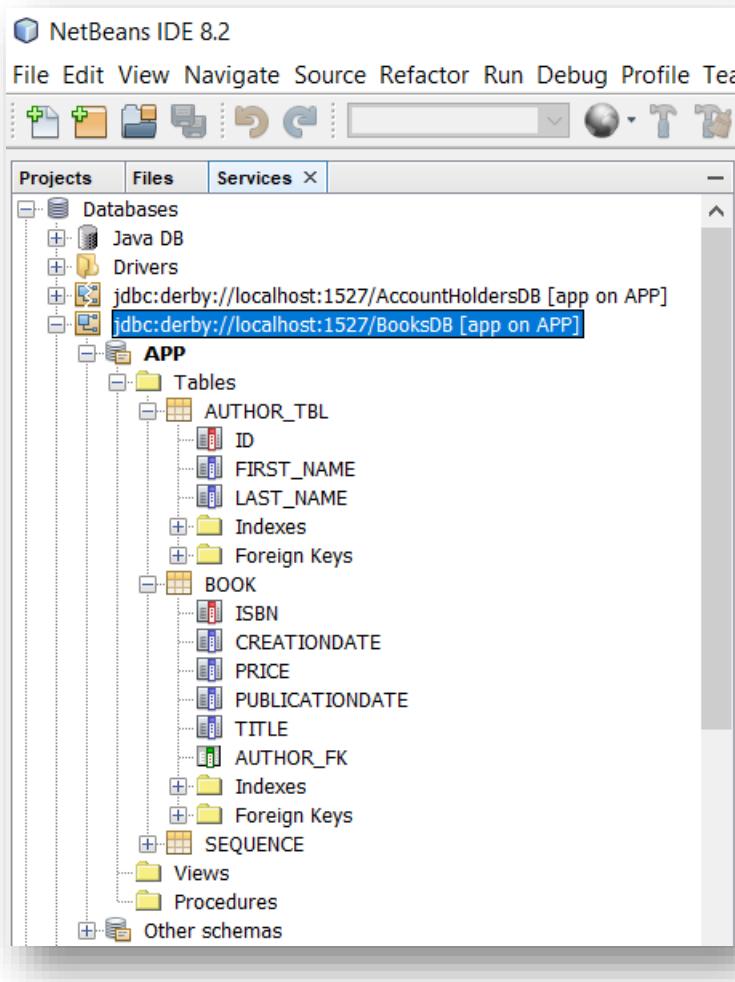
Get books outcome

Below are the books retrieved from the database.

ISBN:	111
Title:	Java
Author name:	Vuyisile
Author surname:	Memani
Price:	123.0
Publication date:	2003-01-23
ISBN:	222
Title:	C++
Author name:	Onkopotse
Author surname:	Tiro
Price:	100.0
Publication date:	2010-01-15
ISBN:	333
Title:	VB
Author name:	Patrice
Author surname:	Mulumba
Price:	200.0
Publication date:	2015-01-17
ISBN:	444
Title:	JEE
Author name:	Zakes
Author surname:	Mda
Price:	400.0
Publication date:	2018-01-22
ISBN:	555
Title:	Assembly
Author name:	Ojo
Author surname:	Mujinga
Price:	600.0
Publication date:	2020-01-27

Please click [here](#) to get back to the menu page or [here](#) to the main page.

View the records in the database.



Contents of the BOOK table.

#	ISBN	CREATIONDATE	PRICE	PUBLICATIONDATE	TITLE	AUTHOR_FK
1	111 2023-03-31 10:44:06.677			123.0 2003-01-23	Java	1
2	222 2023-03-31 11:07:23.787			100.0 2010-01-15	C++	2
3	333 2023-03-31 11:08:37.447			200.0 2015-01-17	VB	3
4	444 2023-03-31 11:09:21.226			400.0 2018-01-22	JEE	4
5	555 2023-03-31 11:10:50.763			600.0 2020-01-27	Assembly	5

Contents of the AUTHOR_TBL

#	ID	FIRST_NAME	LAST_NAME
1	1	Vuyisile	Memani
2	2	Onkopotshe	Tiro
3	3	Patrice	Mulumba
4	4	Zakes	Mda
5	5	Ojo	Mujinga

8.4.2 One-to-many

The **@OneToMany** annotation is used to denote a one-to-many relationship between tables. This means one table has many records in another table. Talking objects wise, you have one class containing many instances of another class. That is, a single class contains a list of objects.

Example

Create a web application that will perform some of the **CRUD** (Create Read Update Delete) operations on customers. A customer has a **name**, **surname**, and **many contact numbers**. A contact has a value.

CUSTOMER_TBL		
Field	Description	Constraint
id	The customer id.	Primary key. The customer id.
name	The name of a customer.	Must be a string. Cannot be nullable. The maximum length is 20.
surname	The surname of a customer.	Must be a contained instance. Must have a reference to the child class called author.
contacts	The list of contacts.	Must be a Date. Cannot be nullable.
creationDate	The creation date.	The timestamp.

CONTACT_TBL		
Field	Description	Constraint
id	The customer account number.	Primary key. The customer id.
cellNo	The cell number.	Must be a string. Cannot be nullable. The maximum length is 20.

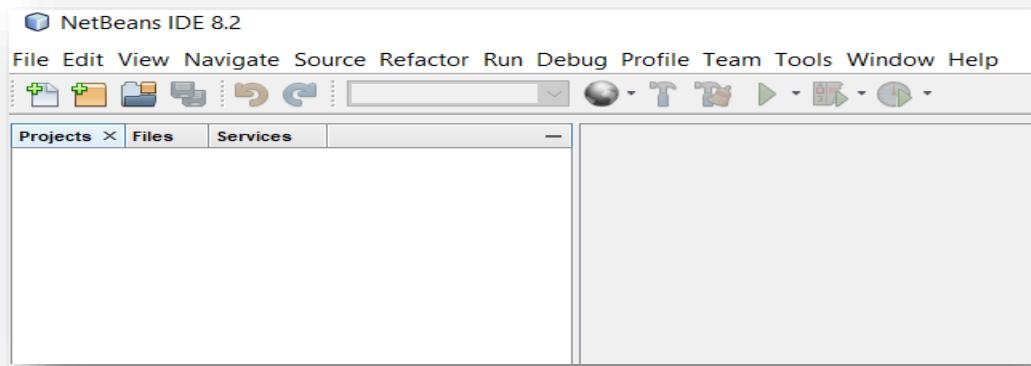
Solution approach

The solution to this problem is going to be done in five parts, namely:

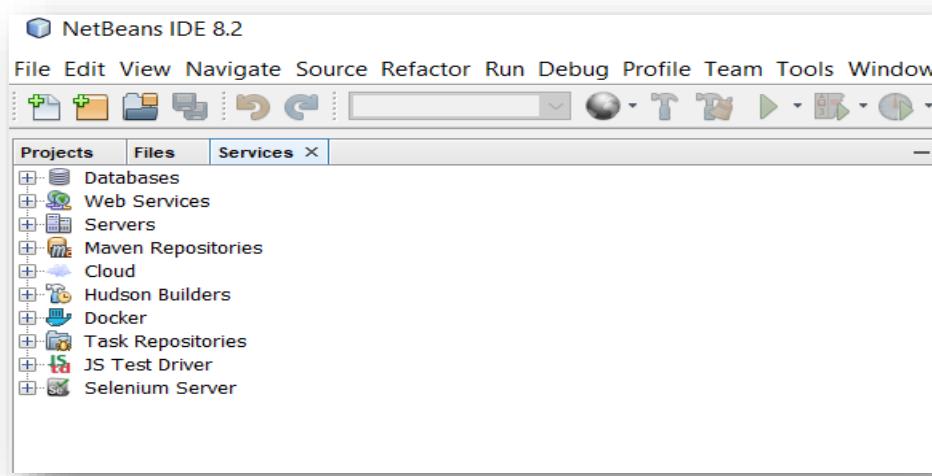
- **Part A:** we create a database.
- **Part B:** we establish connection between the created database and GlassFish.
- **Part C:** we create an EJB module. The module will have an entity with business logic.
- **Part D:** we create a client web application.
- **Part E:** we run the client web application.

Part A - Create a database

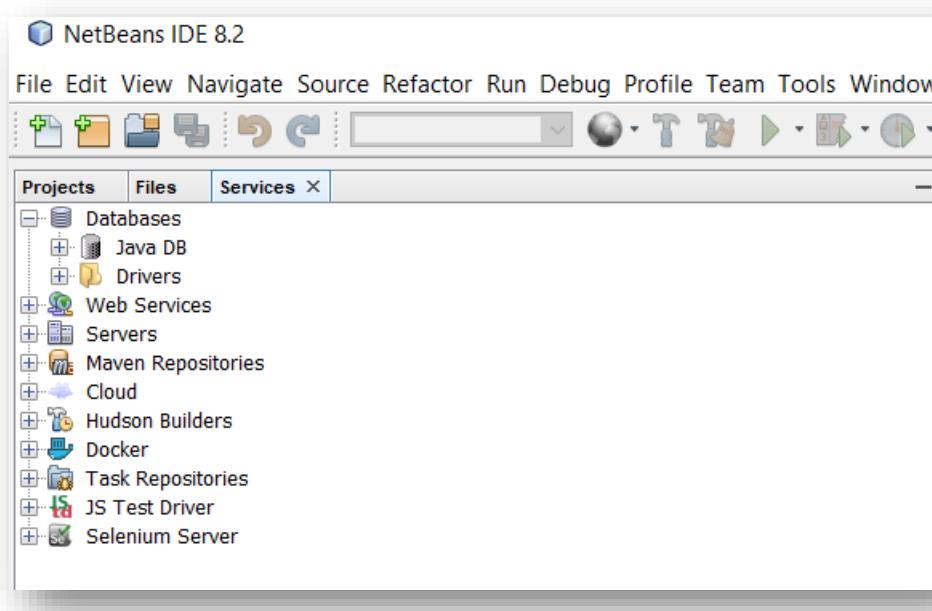
Launch NetBeans.



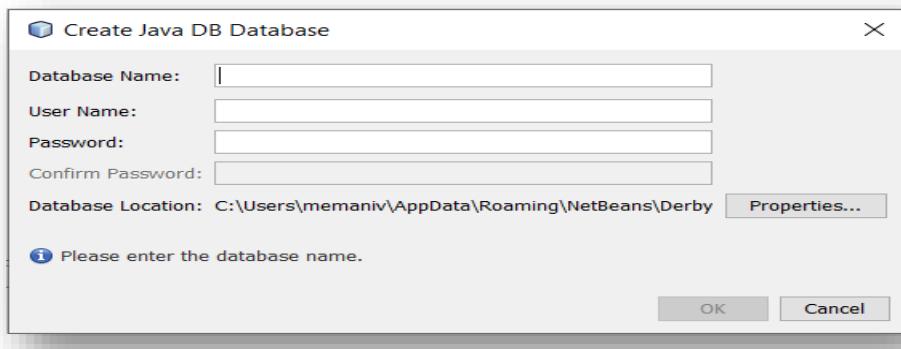
Click on the **Services** tab.



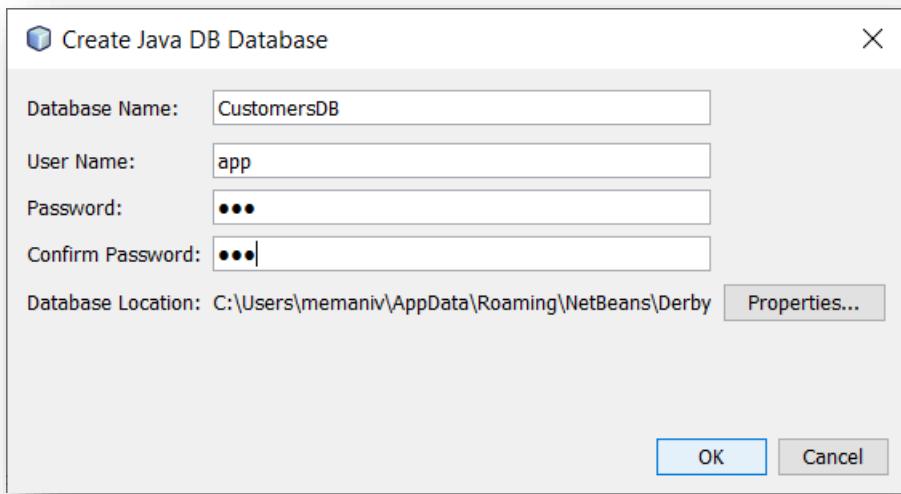
Expand the **Databases** option.



Right-click on **Java DB** and select **Create Database**.



Fill-in the form. I made the password to be **123**.

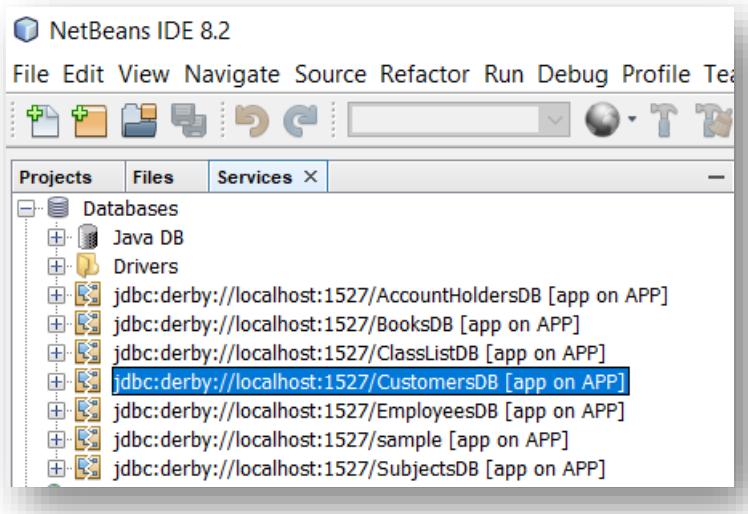


Click **OK**.

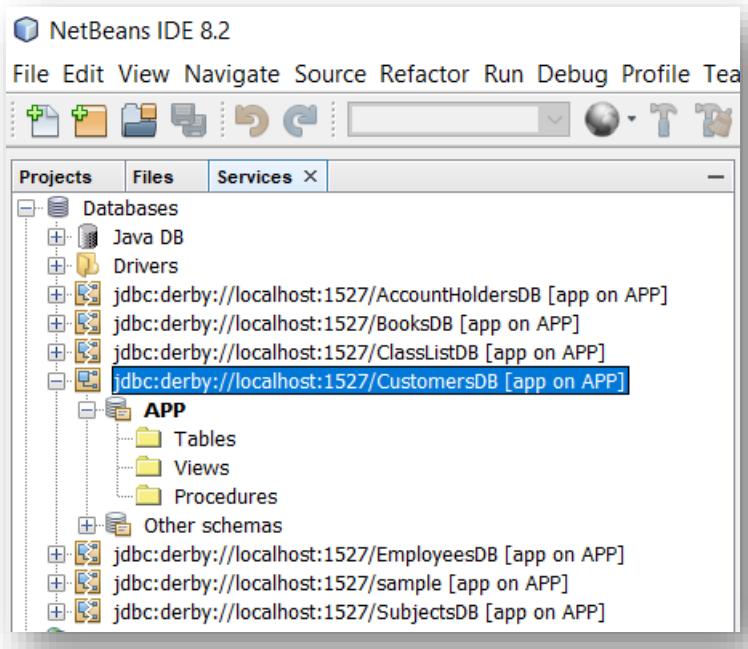
- ✓ An output message confirming that **Derby** has started and ready to accept connections on port 1527 is shown.

```
Output - Java DB Database Process
Thu Mar 30 17:18:48 CAT 2023 : Security manager installed using the Basic server security policy.
Thu Mar 30 17:18:48 CAT 2023 : Apache Derby Network Server - 10.11.1.2 - (1629631) started and ready to accept connections on port 1527
```

- ✓ A database is created.

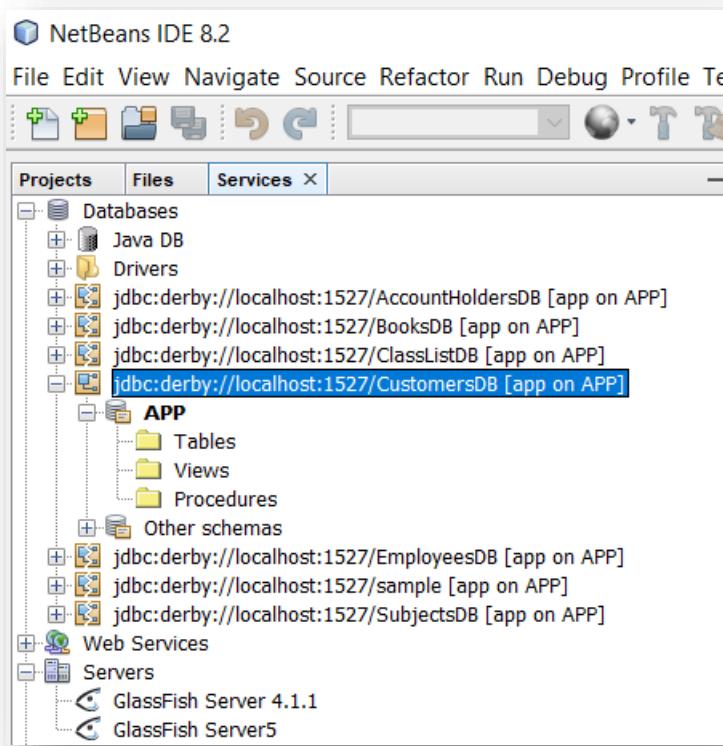


Right-click on the database and select **Connect**.

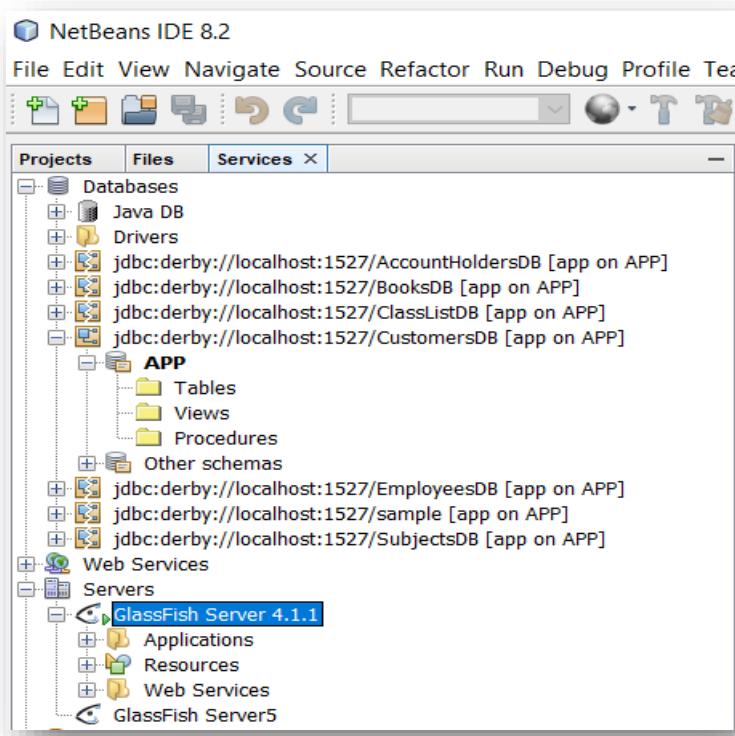


Part B – Connect the database to the application server.

Click on the **Services** tab.



Start the server.



Right-click on the server and select **View Domain Admin Console**.

The screenshot shows the GlassFish Server Open Source Edition Common Tasks console. The URL in the address bar is <http://localhost:4848/common/index.jsf>. The top navigation bar includes links for Home and About... The status bar at the top right shows User: admin | Domain: domain1 | Server: localhost. The main content area is titled "GlassFish Console - Common Tasks". On the left, there is a sidebar titled "Common Tasks" with the following menu items: Domain, server (Admin Server), Clusters, Standalone Instances, Nodes, Applications, Lifecycle Modules, Monitoring Data, Resources (which is expanded to show Concurrent Resources, Connectors, JDBC, JMS Resources, JNDI, JavaMail Sessions, and Resource Adapter Configs), Configurations (which is expanded to show default-config, server-config, and Update Tool). To the right of the sidebar, there are several sections: "GlassFish News" (with a link to GlassFish News), "Deployment" (with links to List Deployed Applications and Deploy an Application), "Administration" (with links to Change Administrator Password and List Password Aliases), and "Monitoring" (with a link to Monitoring Data).

Under **Common Tasks** panel, expand **JDBC**.

The screenshot shows the GlassFish Server Open Source Edition JDBC console. The URL in the address bar is <http://localhost:4848/common/index.jsf>. The top navigation bar includes links for Home and About... The status bar at the top right shows User: admin | Domain: domain1 | Server: localhost. The main content area is titled "JDBC". On the left, there is a sidebar titled "Common Tasks" with the same menu items as the previous screenshot, but the "Resources" section is expanded to show JDBC (which is selected and highlighted in blue). Under JDBC, there are links for JDBC Resources and JDBC Connection Pools. To the right of the sidebar, there are two main sections: "JDBC Resources" and "JDBC Connection Pools".

Modify the Connection pool to include the **ClassListDB**.

- ✓ Click on **JDBC Connection Pools**.

JDBC Connection Pools

To store, organize, and retrieve data, most applications use relational databases. Java EE applications access relational databases through the JDBC API. Before an application can access a database, it must get a connection.

Pools (3)					
Select	Pool Name	Resource Type	Classname	Description	
<input type="checkbox"/>	DerbyPool	javax.sql.DataSource	org.apache.derby.jdbc.ClientDataSource		
<input type="checkbox"/>	SamplePool	javax.sql.DataSource	org.apache.derby.jdbc.ClientDataSource		
<input type="checkbox"/>	TimerPool	javax.sql.XADatasource	org.apache.derby.jdbc.EmbeddedXADataSource		

- ✓ Click on the **DerbyPool** link.

Edit JDBC Connection Pool

Modify an existing JDBC connection pool. A JDBC connection pool is a group of reusable connections for a particular database.

Load Defaults Flush Ping

General Settings

Pool Name: DerbyPool
Resource Type: javax.sql.DataSource
Must be specified if the datasource class implements more than 1 of the interface.

Datasource Classname: org.apache.derby.jdbc.ClientDataSource
Vendor-specific classname that implements the DataSource and/or XADatasource APIs

Driver Classname:
Vendor-specific classname that implements the java.sql.Driver interface.

Ping: Enabled
When enabled, the pool is pinged during creation or reconfiguration to identify and warn of any erroneous values for its attributes

Deployment Order: 100
Specifies the loading order of the resource at server startup. Lower numbers are loaded first.

Description:

Pool Settings

Initial and Minimum Pool Size: 8 Connections
Minimum and initial number of connections maintained in the pool

Maximum Pool Size: 32 Connections
Maximum number of connections that can be created to satisfy client requests

Pool Resize Quantity: 2 Connections

- ✓ Click on **Additional Properties**.

Edit JDBC Connection Pool Properties

Modify properties of an existing JDBC connection pool.

Pool Name: DerbyPool

Additional Properties (6)

Add Property Delete Properties

Select	Name	Value
<input type="checkbox"/>	PortNumber	1527
<input type="checkbox"/>	Password	APP
<input type="checkbox"/>	User	APP
<input type="checkbox"/>	serverName	localhost
<input type="checkbox"/>	DatabaseName	sun-appserv-samples
<input type="checkbox"/>	connectionAttributes	;create=true

- ✓ Click on **Add Property** and add a **url** property with the following value:

jdbc:derby://localhost:1527/CustomersDB

- ✓ Delete the **connectionAttributes** property.

General Advanced Additional Properties

Edit JDBC Connection Pool Properties

Modify properties of an existing JDBC connection pool.

Pool Name: DerbyPool

Additional Properties (6)		
Select	Name	Value
<input type="checkbox"/>	PortNumber	1527
<input type="checkbox"/>	Password	123
<input type="checkbox"/>	User	app
<input type="checkbox"/>	serverName	localhost
<input type="checkbox"/>	DatabaseName	CustomersDB
<input type="checkbox"/>	url	jdbc:derby://localhost:1527/CustomersDB

- ✓ Save the changes by clicking the **Save** button.

General Advanced Additional Properties

New values successfully saved.

Edit JDBC Connection Pool Properties

Modify properties of an existing JDBC connection pool.

Pool Name: DerbyPool

Additional Properties (6)		
Select	Name	Value
<input type="checkbox"/>	PortNumber	1527
<input type="checkbox"/>	Password	123
<input type="checkbox"/>	User	app
<input type="checkbox"/>	serverName	localhost
<input type="checkbox"/>	DatabaseName	CustomersDB
<input type="checkbox"/>	url	jdbc:derby://localhost:1527/CustomersDB

- ✓ Check if the created connection pool is working. Do this by clicking the **Ping** button.

Edit JDBC Connection Pool

Modify an existing JDBC connection pool. A JDBC connection pool is a group of reusable connections for a particular database.

General Settings

Pool Name:	DerbyPool
Resource Type:	javax.sql.DataSource
Must be specified if the datasource class implements more than 1 of the interface.	
Datasource Classname:	org.apache.derby.jdbc.ClientDataSource
Vendor-specific classname that implements the DataSource and/or XDataSource APIs	
Driver Classname:	
Vendor-specific classname that implements the java.sql.Driver interface.	
Ping:	<input checked="" type="checkbox"/> Enabled
When enabled, the pool is pinged during creation or reconfiguration to identify and warn of any erroneous values for its attributes	
Deployment Order:	100
Specifies the loading order of the resource at server startup. Lower numbers are loaded first.	
Description:	

Confirm that the resource points to the DerbyPool.

- ✓ Still under **JDBC**, click on **JDBC Resources**.

Resources (3)					
Select	JNDI Name	Logical JNDI Name	Enabled	Connection Pool	
<input type="checkbox"/>	jdbc/_TimerPool		<input checked="" type="checkbox"/>	_TimerPool	
<input type="checkbox"/>	jdbc/_default	java.comp/DefaultDataSource	<input checked="" type="checkbox"/>	DerbyPool	
<input type="checkbox"/>	jdbc/sample		<input checked="" type="checkbox"/>	SamplePool	

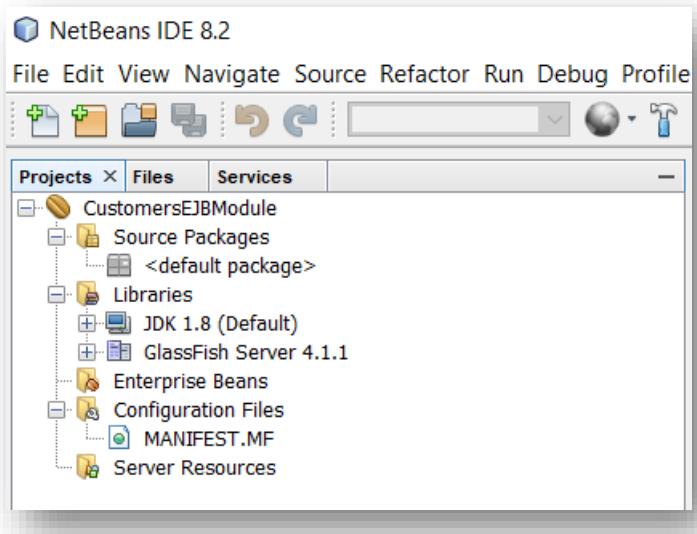
Take note of the purpose of JDBC Resources. It is said:

"JDBC resources provide applications with a means to connect to a database."

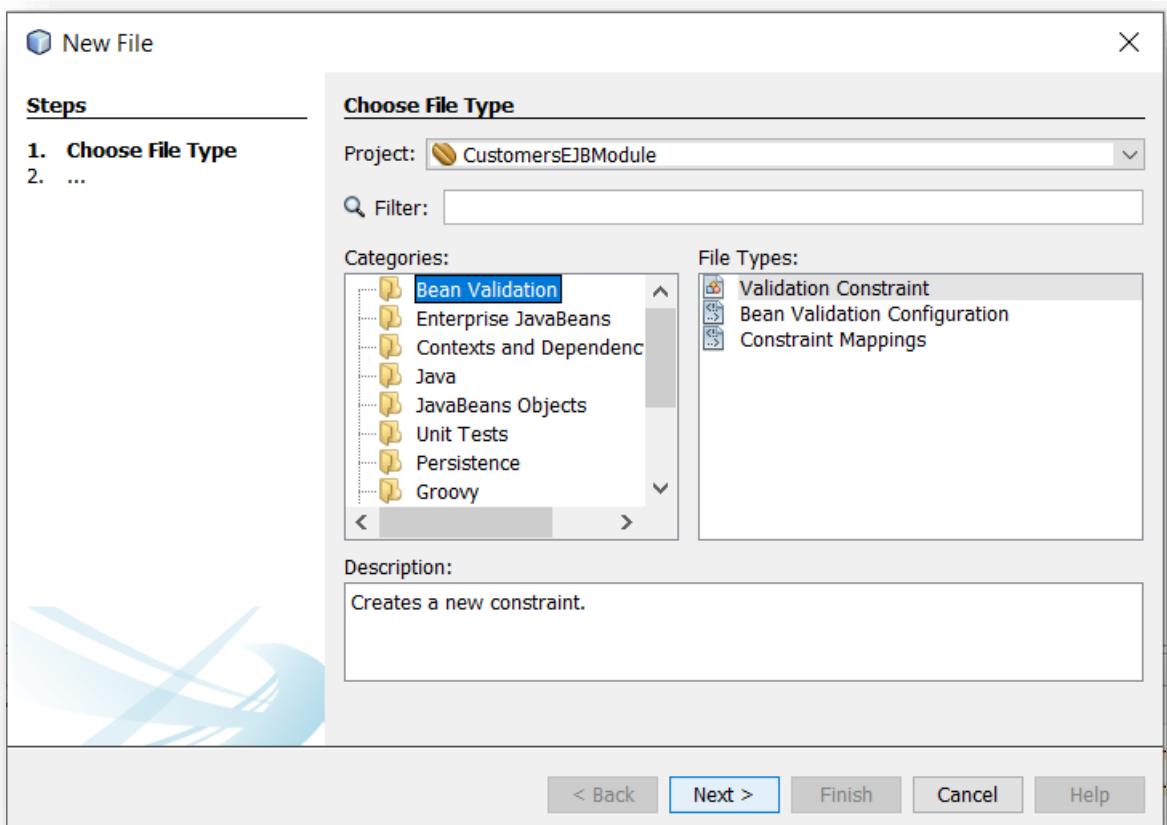
You can also see that the **jdbc/_default** resource is associated with the **DerbyPool** Connection Pool.

Part C – Create an entity.

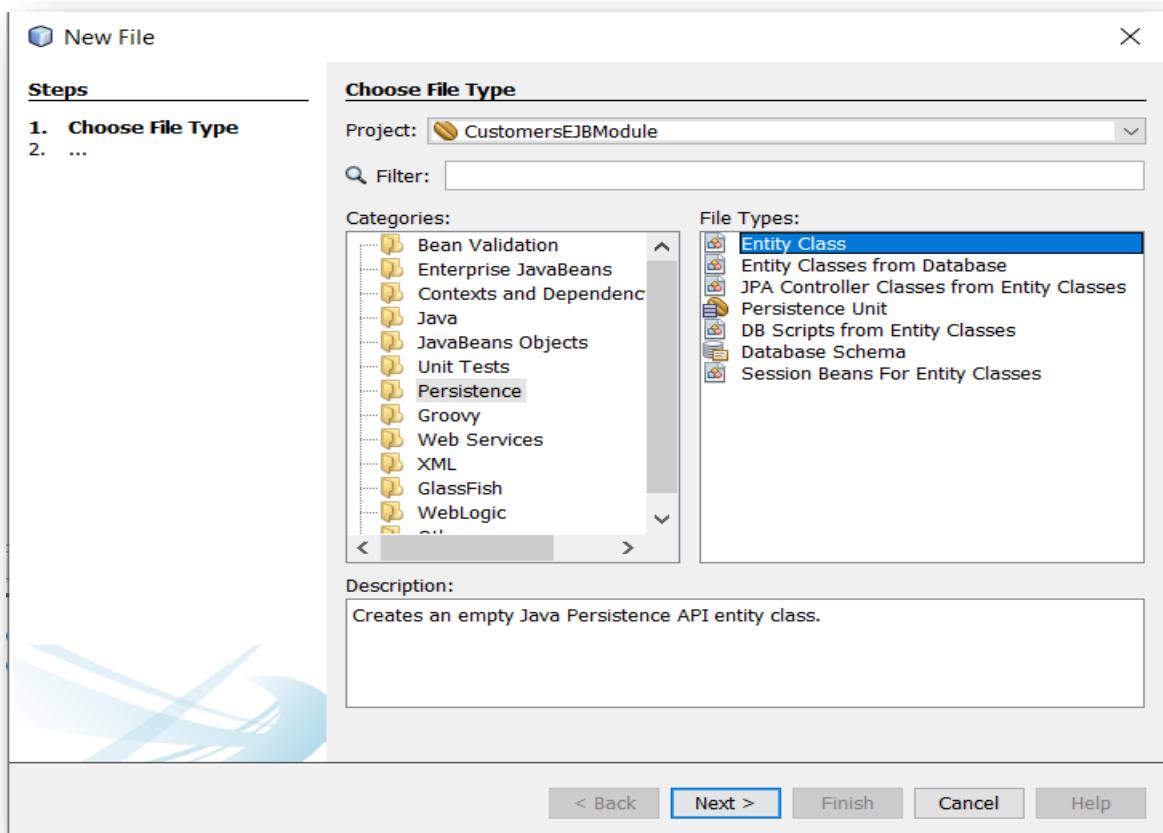
Create an EJB project called **CustomersEJBModule**.



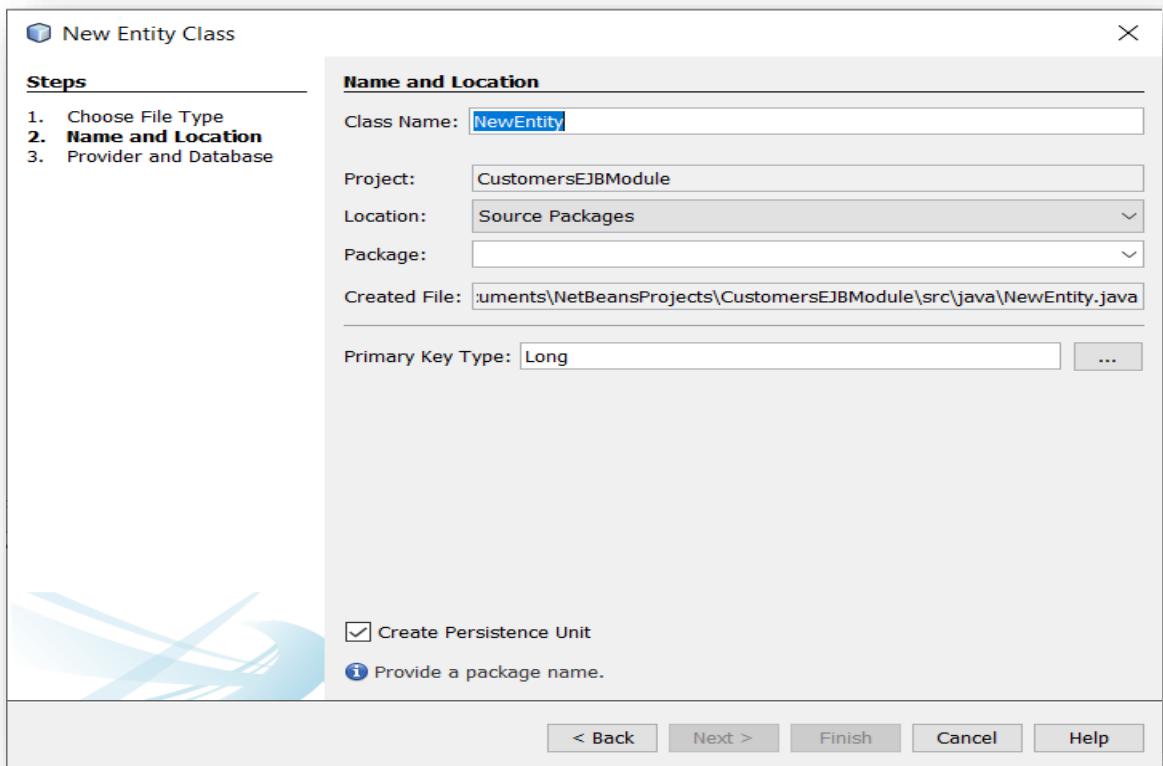
Right-click on the project and select **New | Other**.



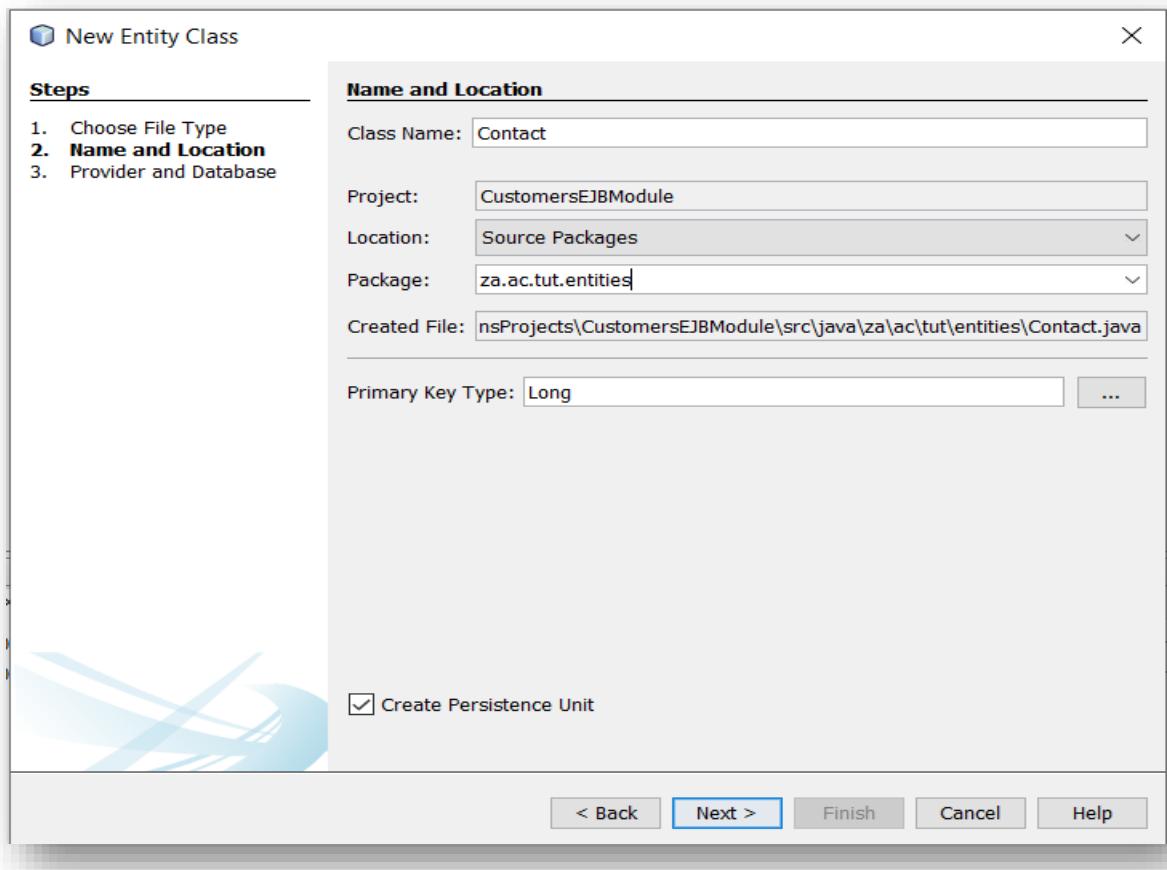
Under **Categories** select **Persistence**, and under **File Types** select **Entity Class**.



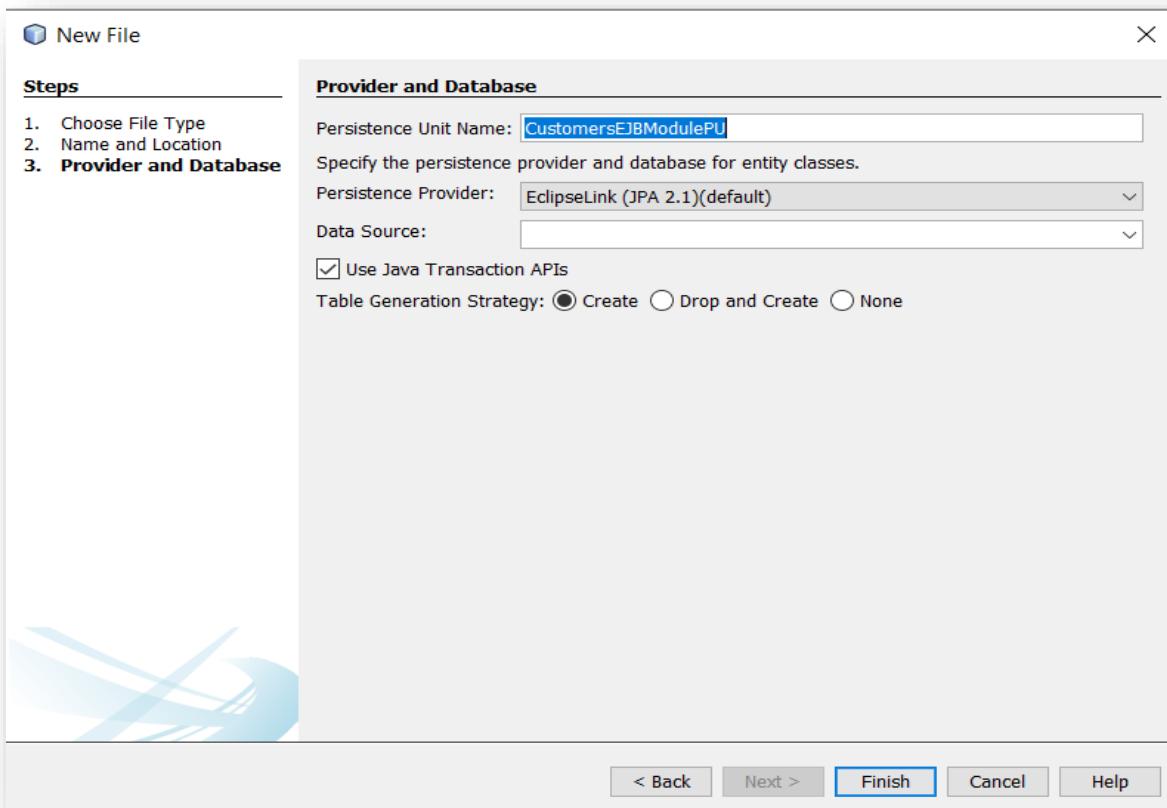
Click **Next**.



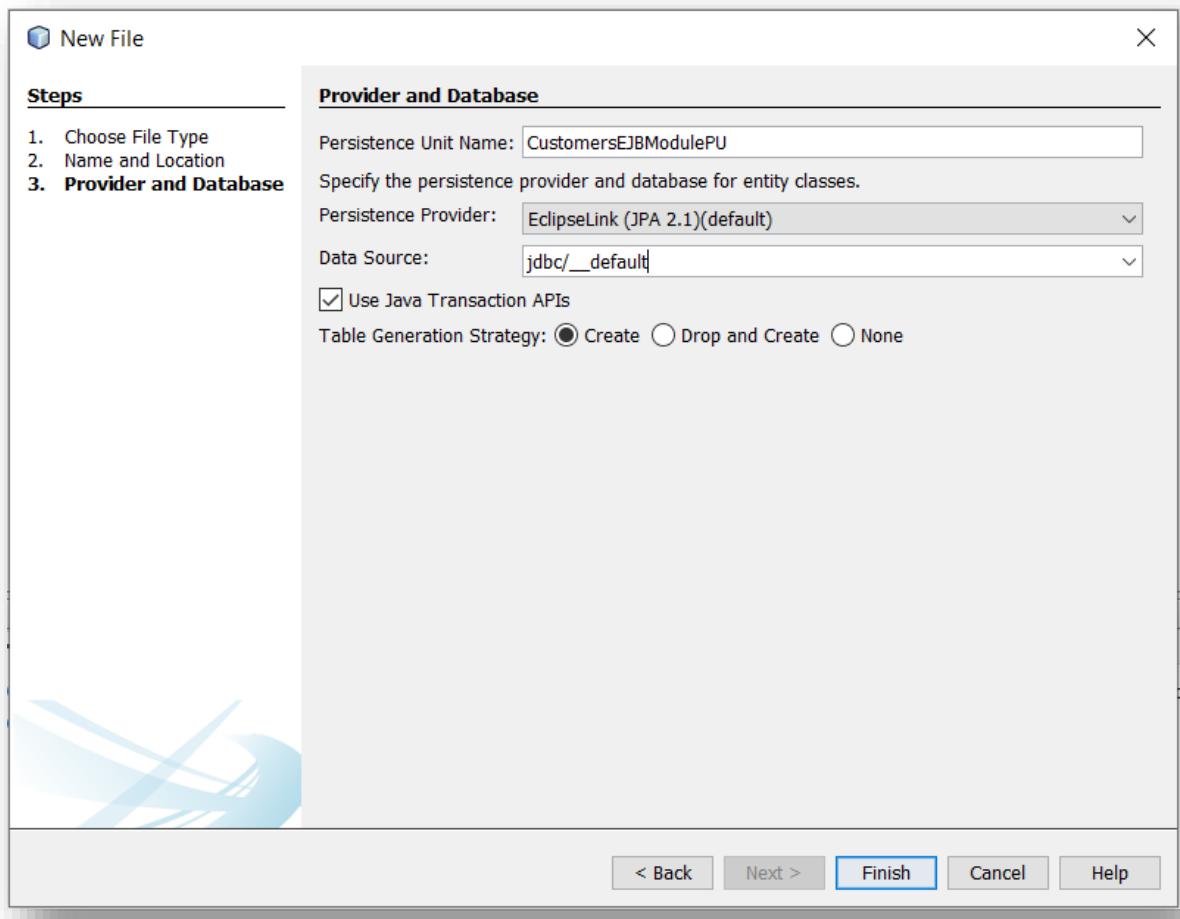
Fill-in the form.



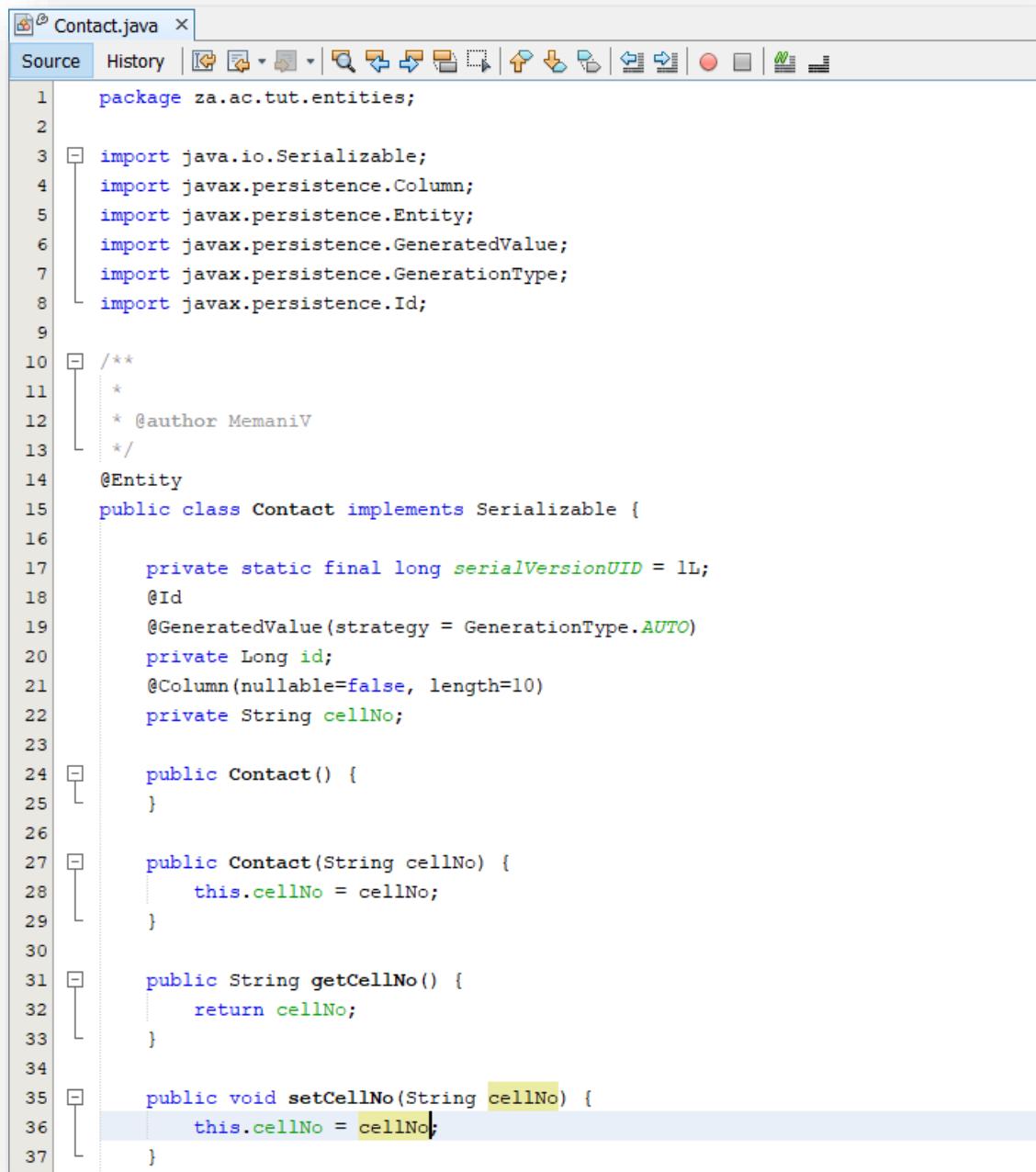
Click **Next**.



Under **Data Source**, select **jdbc/_default**.



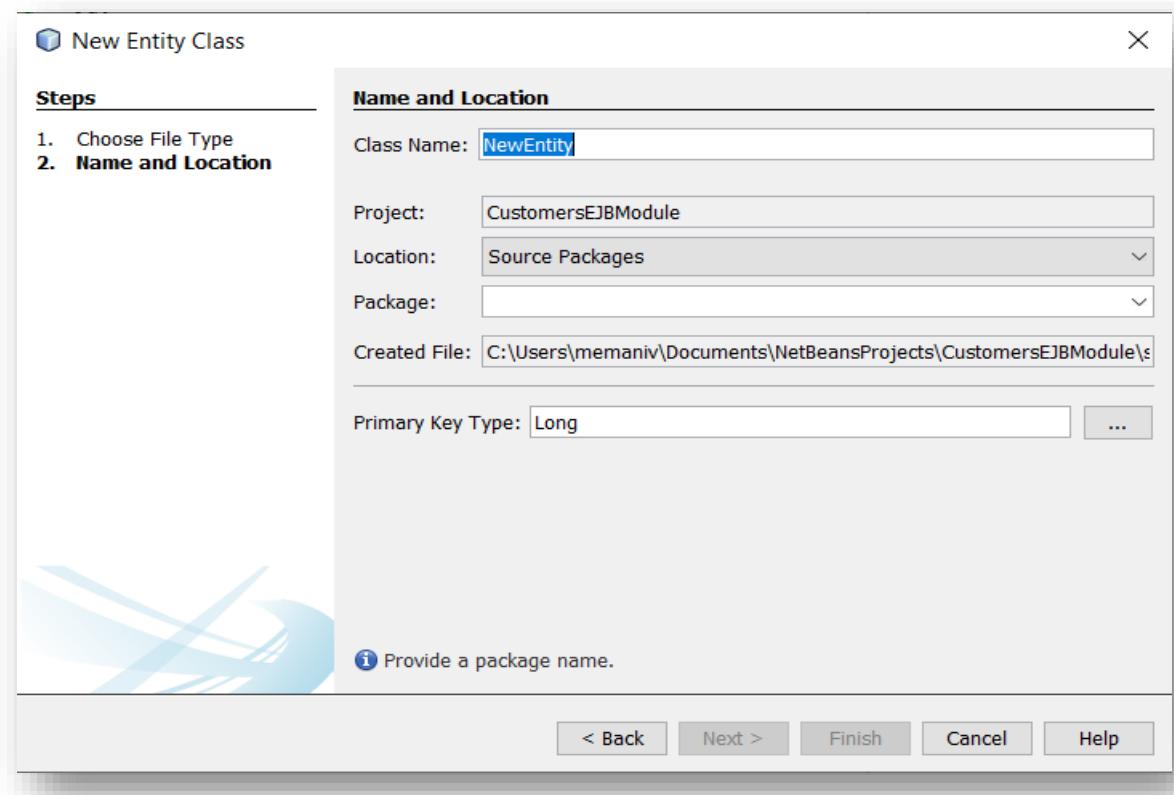
Click **Finish**.



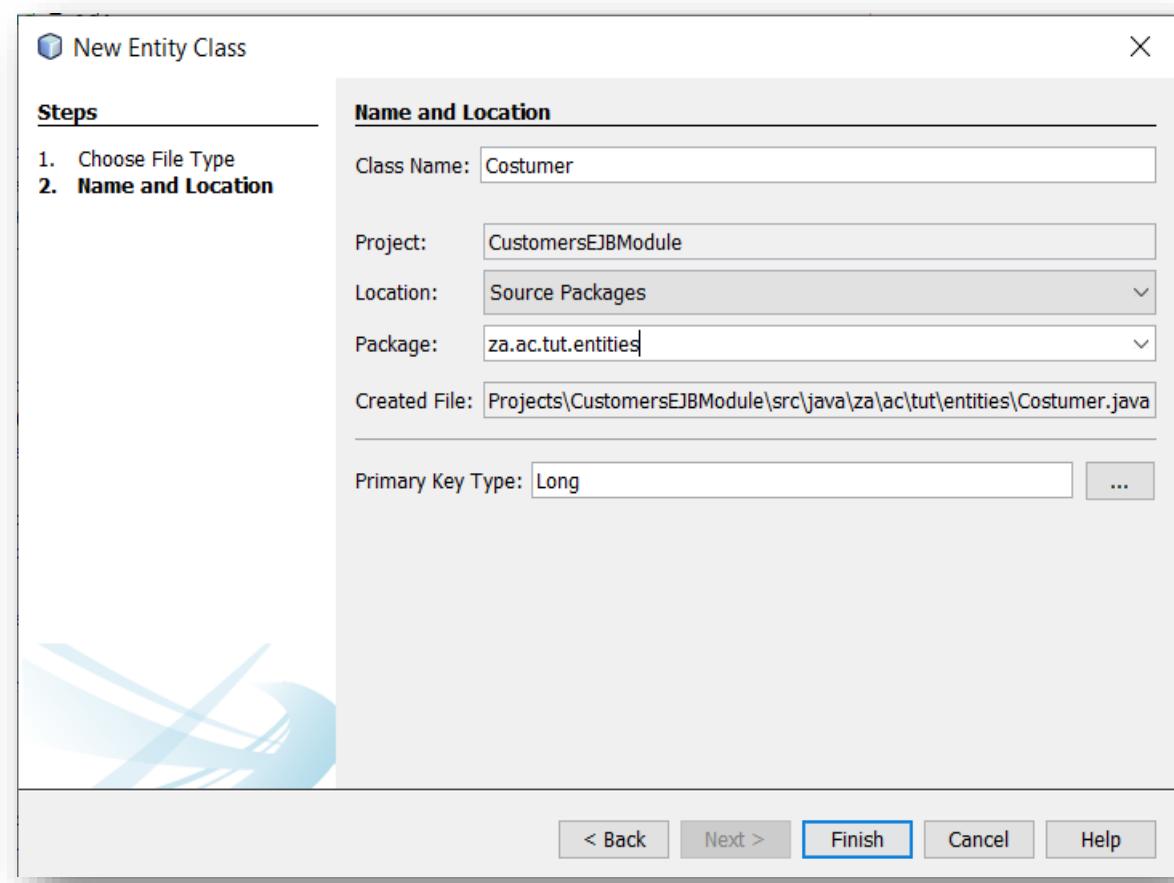
```
1 package za.ac.tut.entities;
2
3 import java.io.Serializable;
4 import javax.persistence.Column;
5 import javax.persistence.Entity;
6 import javax.persistence.GeneratedValue;
7 import javax.persistence.GenerationType;
8 import javax.persistence.Id;
9
10 /**
11 *
12 * @author MemaniV
13 */
14 @Entity
15 public class Contact implements Serializable {
16
17     private static final long serialVersionUID = 1L;
18     @Id
19     @GeneratedValue(strategy = GenerationType.AUTO)
20     private Long id;
21     @Column(nullable=false, length=10)
22     private String cellNo;
23
24     public Contact() {
25     }
26
27     public Contact(String cellNo) {
28         this.cellNo = cellNo;
29     }
30
31     public String getCellNo() {
32         return cellNo;
33     }
34
35     public void setCellNo(String cellNo) {
36         this.cellNo = cellNo;
37     }
}
```

```
39  [-]     public Long getId() {
40      [ ]         return id;
41  }
42
43  [-]     public void setId(Long id) {
44      [ ]         this.id = id;
45  }
46
47  [-]     @Override
48  @ [-]     public int hashCode() {
49      [ ]         int hash = 0;
50      [ ]         hash += (id != null ? id.hashCode() : 0);
51      [ ]         return hash;
52  }
53
54  [-]     @Override
55  @ [-]     public boolean equals(Object object) {
56      [ ]         if (!(object instanceof Contact)) {
57      [ ]             return false;
58  }
59      [ ]         Contact other = (Contact) object;
60      [ ]         if ((this.id == null &amp; other.id != null) ||
61      [ ]             (this.id != null &amp; !this.id.equals(other.id))) {
62      [ ]             return false;
63  }
64      [ ]         return true;
65  }
66
67  [-]     @Override
68  @ [-]     public String toString() {
69      [ ]         return "za.ac.tut.entities.Contact[ id=" + id + " ]";
70  }
71
72  }
```

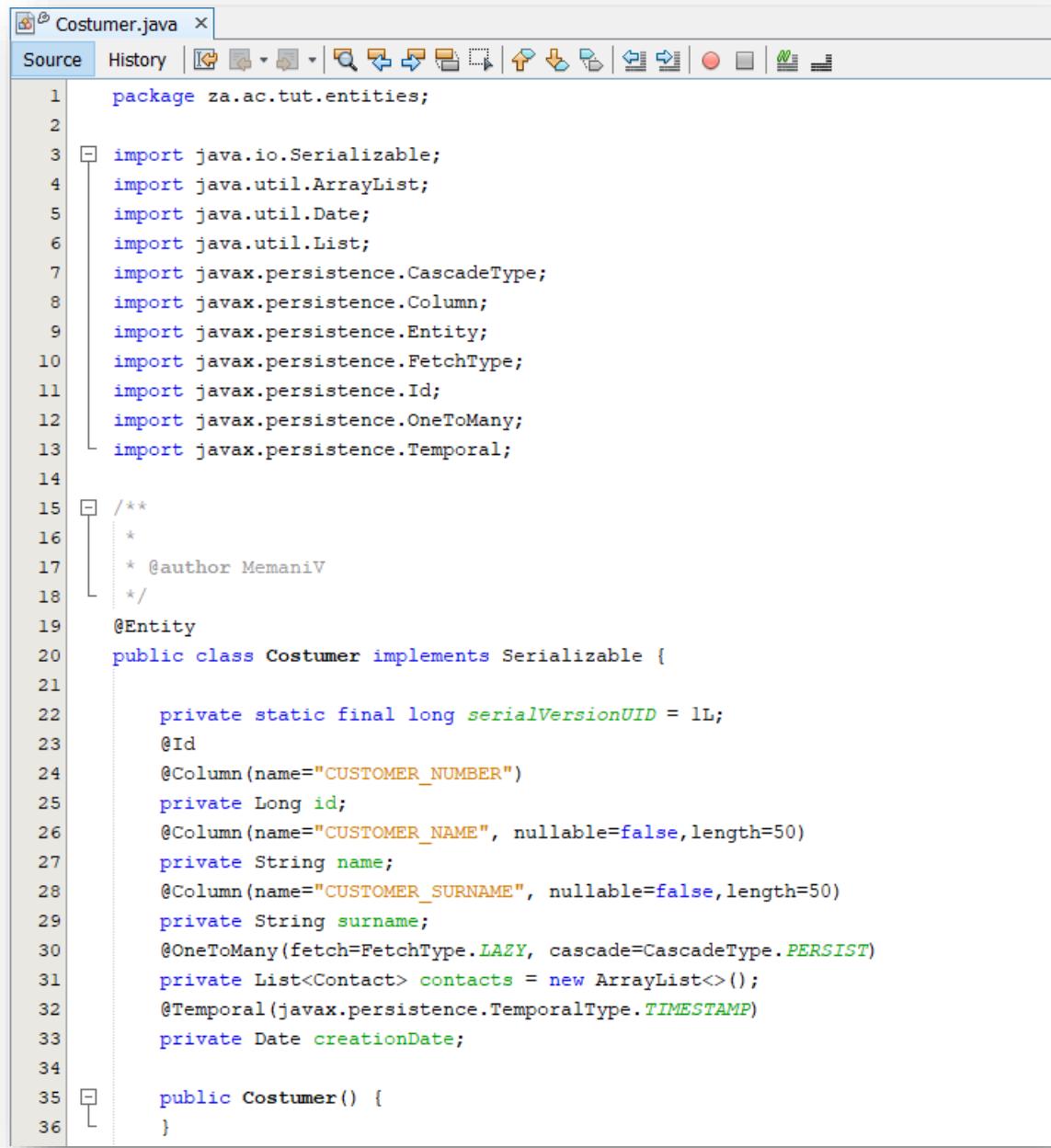
Right-click on the project and select **New | Entity Class**



Fill-in the form.



Click **Finish**.



The screenshot shows a Java code editor window with the following details:

- Title Bar:** The title bar displays "Costumer.java X".
- Toolbar:** The toolbar contains various icons for file operations like Open, Save, Print, Copy, Paste, Find, and Delete.
- Menu Bar:** The menu bar has "Source" selected, followed by "History" and other options.
- Code Area:** The main area contains the Java code for the `Costumer` class. The code includes imports for `Serializable`, `ArrayList`, `Date`, `List`, `CascadeType`, `Column`, `Entity`, `FetchType`, `Id`, `OneToMany`, and `Temporal`. It also includes a Javadoc comment block and annotations for `@Entity`, `@Id`, `@Column`, `@OneToMany`, and `@Temporal`. The constructor is also defined.

```
1 package za.ac.tut.entities;
2
3 import java.io.Serializable;
4 import java.util.ArrayList;
5 import java.util.Date;
6 import java.util.List;
7 import javax.persistence.CascadeType;
8 import javax.persistence.Column;
9 import javax.persistence.Entity;
10 import javax.persistence.FetchType;
11 import javax.persistence.Id;
12 import javax.persistence.OneToMany;
13 import javax.persistence.Temporal;
14
15 /**
16 * 
17 * @author MemaniV
18 */
19 @Entity
20 public class Costumer implements Serializable {
21
22     private static final long serialVersionUID = 1L;
23     @Id
24     @Column(name="CUSTOMER_NUMBER")
25     private Long id;
26     @Column(name="CUSTOMER_NAME", nullable=false,length=50)
27     private String name;
28     @Column(name="CUSTOMER_SURNAME", nullable=false,length=50)
29     private String surname;
30     @OneToMany(fetch=FetchType.LAZY, cascade=CascadeType.PERSIST)
31     private List<Contact> contacts = new ArrayList<>();
32     @Temporal(javax.persistence.TemporalType.TIMESTAMP)
33     private Date creationDate;
34
35     public Costumer() {
36     }
```

```
38     public Costumer(Long id, String name, String surname, Date creationDate) {
39         this.id = id;
40         this.name = name;
41         this.surname = surname;
42         this.creationDate = creationDate;
43     }
44
45     public String getName() {
46         return name;
47     }
48
49     public void setName(String name) {
50         this.name = name;
51     }
52
53     public String getSurname() {
54         return surname;
55     }
56
57     public void setSurname(String surname) {
58         this.surname = surname;
59     }
60
61     public List<Contact> getContacts() {
62         return contacts;
63     }
64
65     public void setContacts(List<Contact> contacts) {
66         this.contacts = contacts;
67     }
68
69     public Date getCreationDate() {
70         return creationDate;
71     }
72
73     public void setCreationDate(Date creationDate) {
74         this.creationDate = creationDate;
75     }
```

```

77     public Long getId() {
78         return id;
79     }
80
81     public void setId(Long id) {
82         this.id = id;
83     }
84
85     @Override
86     public int hashCode() {
87         int hash = 0;
88         hash += (id != null ? id.hashCode() : 0);
89         return hash;
90     }
91
92     @Override
93     public boolean equals(Object object) {
94         if (!(object instanceof Costumer)) {
95             return false;
96         }
97         Costumer other = (Costumer) object;
98         if ((this.id == null & other.id != null) ||
99             (this.id != null & !this.id.equals(other.id))) {
100            return false;
101        }
102        return true;
103    }
104
105    @Override
106    public String toString() {
107        return "za.ac.tut.entities.Costumer[ id=" + id + " ]";
108    }
109
110 }

```

View the **persistence.xml** file.

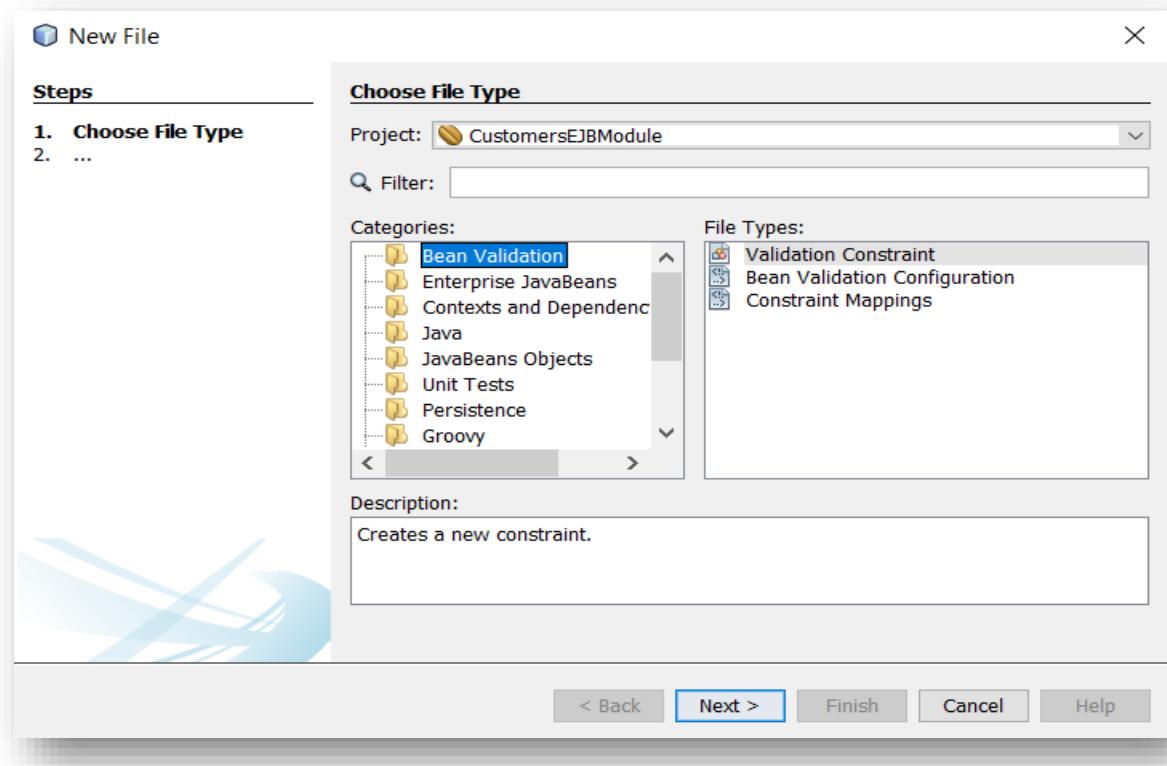
```

<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence_2_1.xsd">
    <persistence-unit name="CustomersEJBModulePU" transaction-type="JTA">
        <jta-data-source>jdbc/_default</jta-data-source>
        <exclude-unlisted-classes>false</exclude-unlisted-classes>
        <properties>
            <property name="javax.persistence.schema-generation.database.action" value="create"/>
        </properties>
    </persistence-unit>
</persistence>

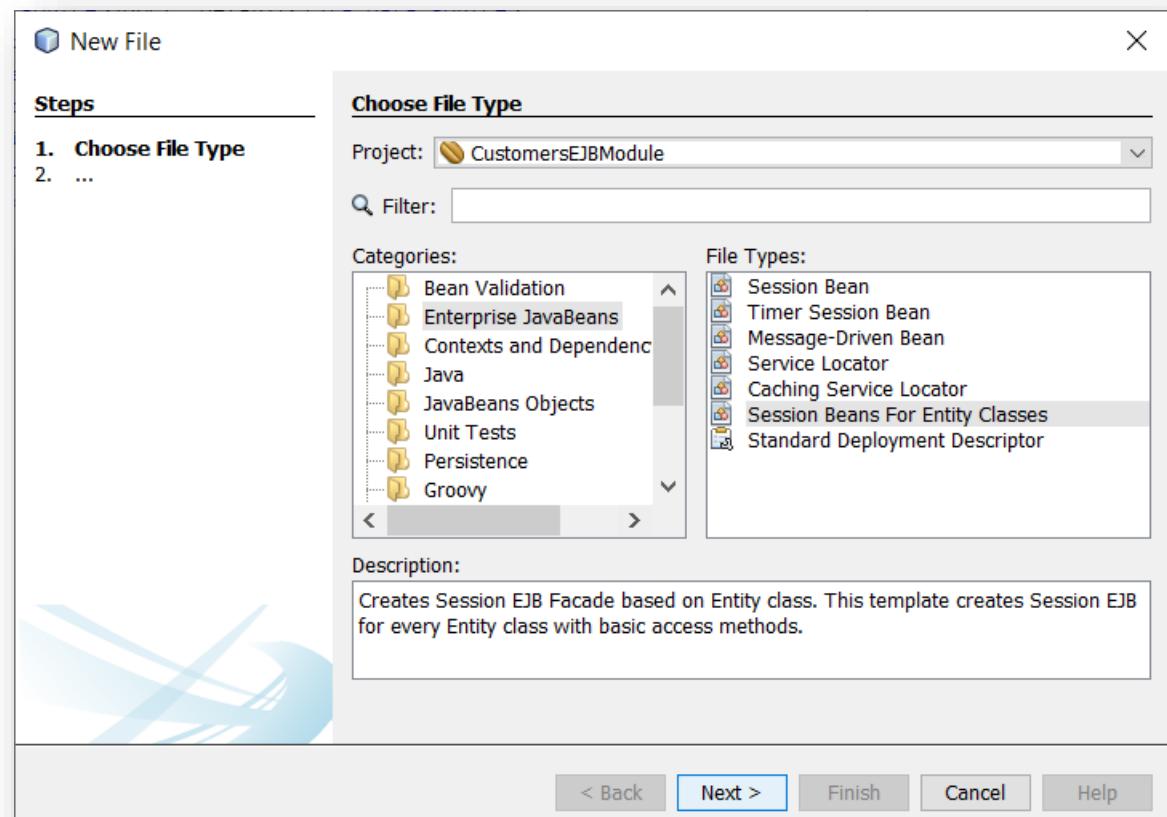
```

Create business code (CRUD operations) for the entity. Perform the following steps:

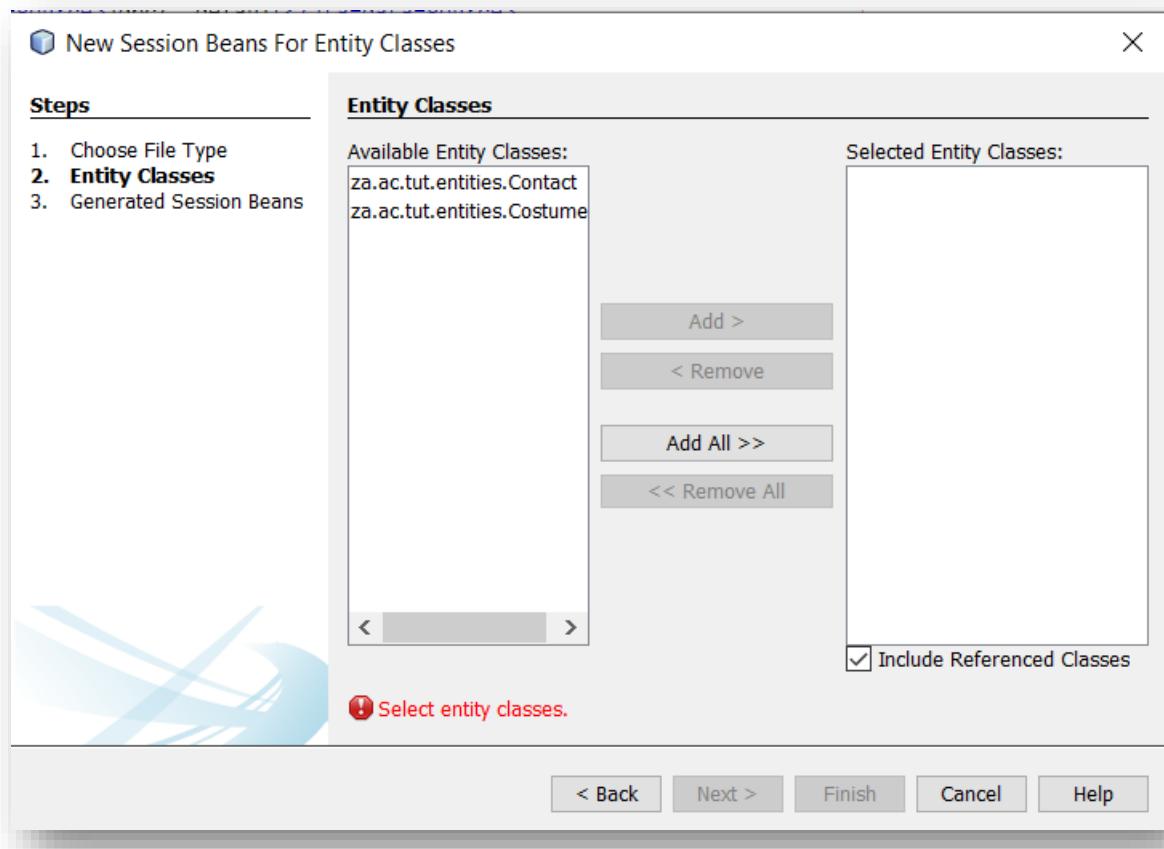
- ✓ Right-click on the project and select **New | Other**.



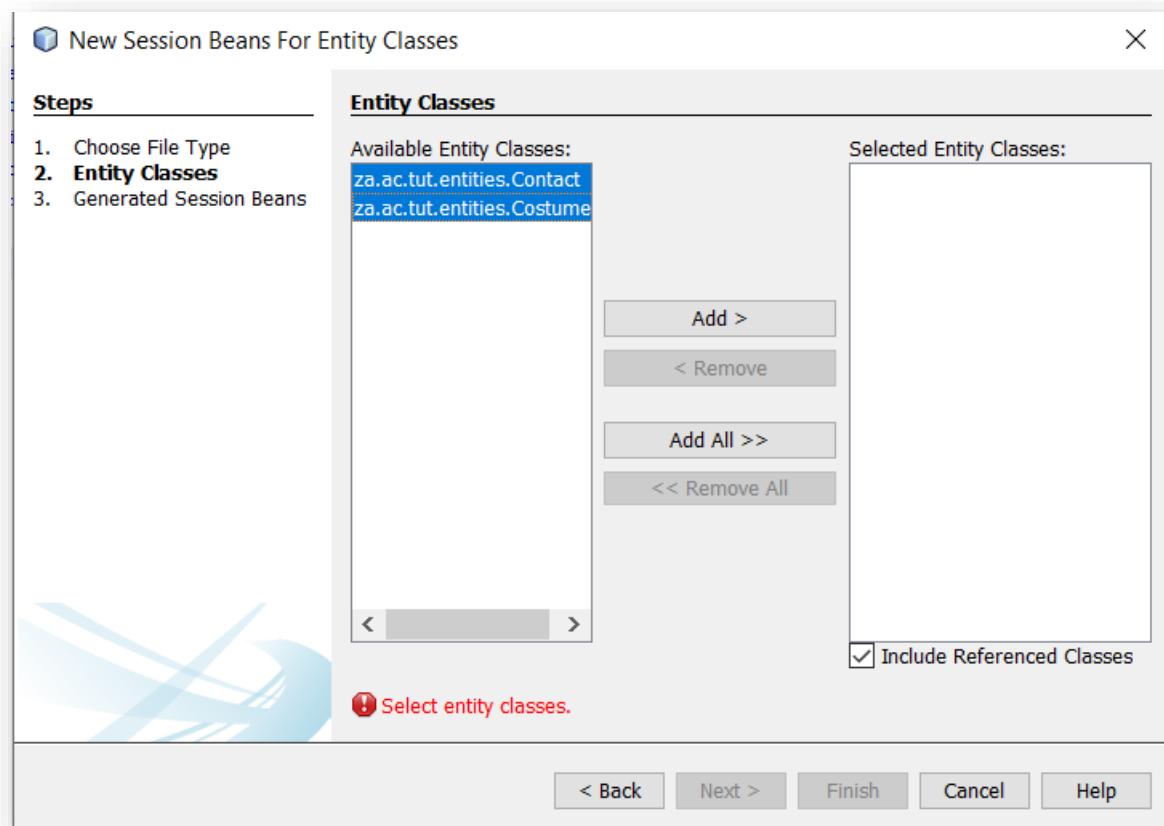
- ✓ Under **Categories**, select **Enterprise JavaBeans**. Under **File Types** select **Session Beans for Entity Classes**.



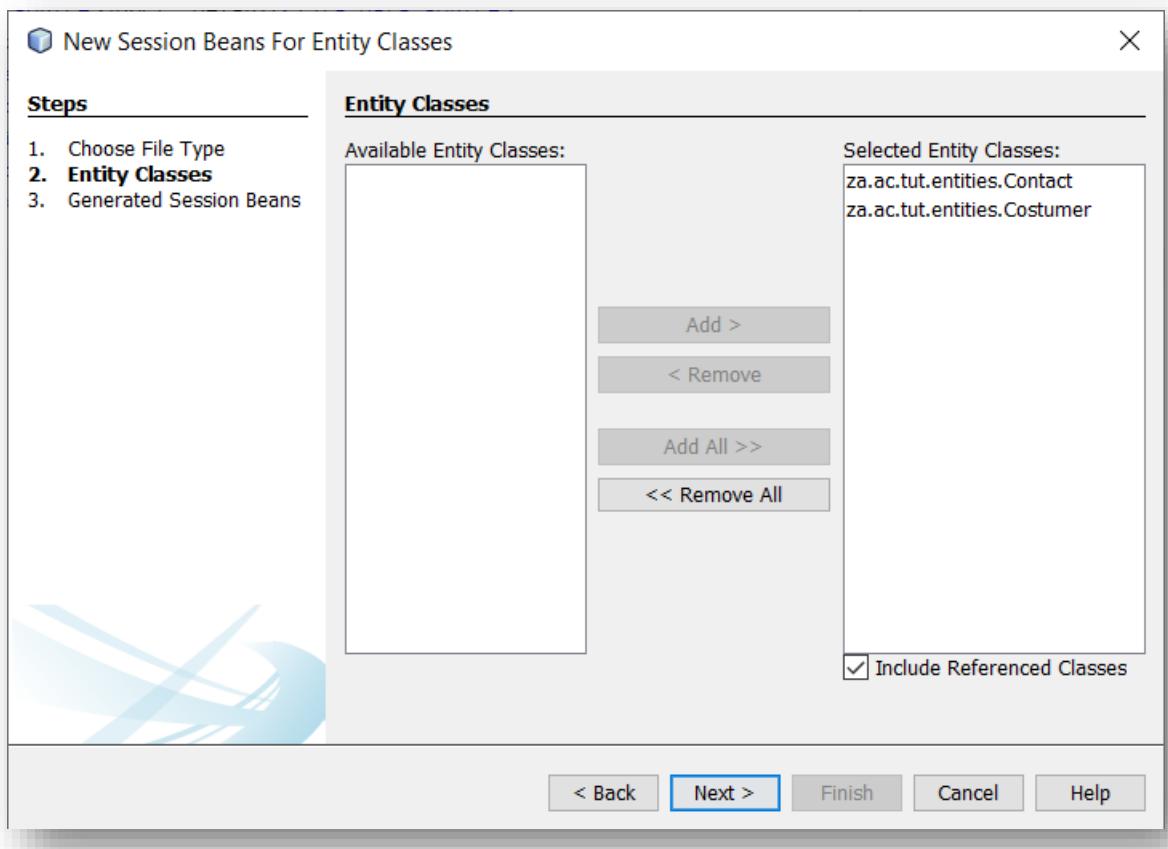
✓ Click **Next**.



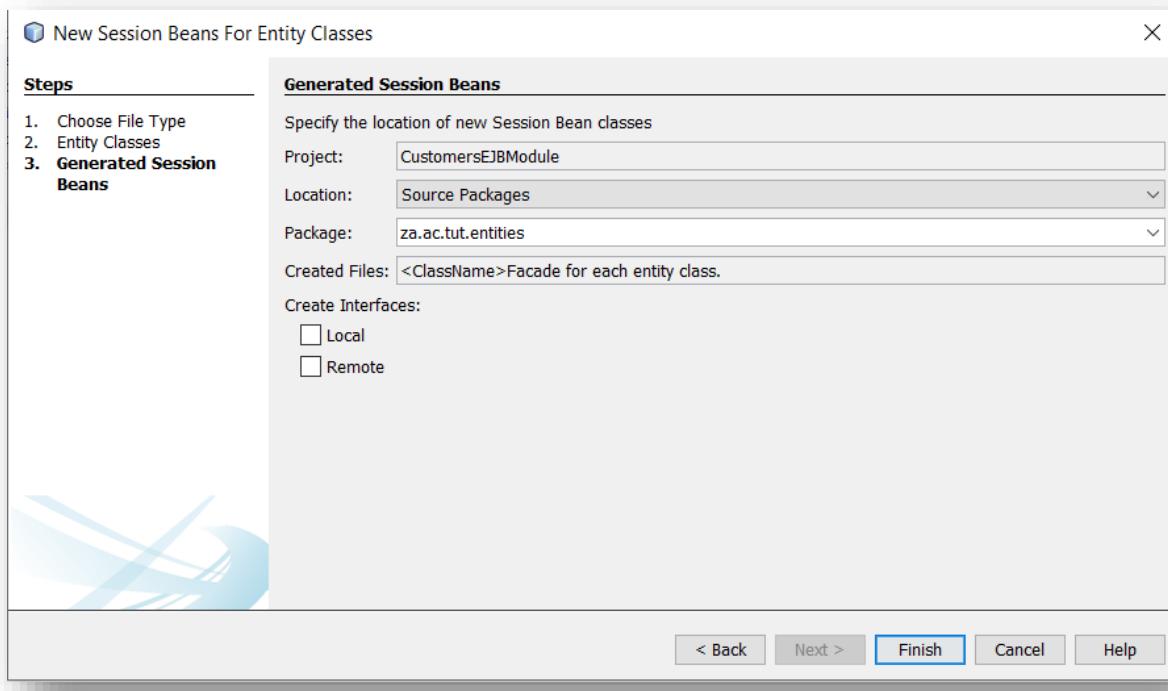
✓ Select **all** the entities under Available Entity Classes.



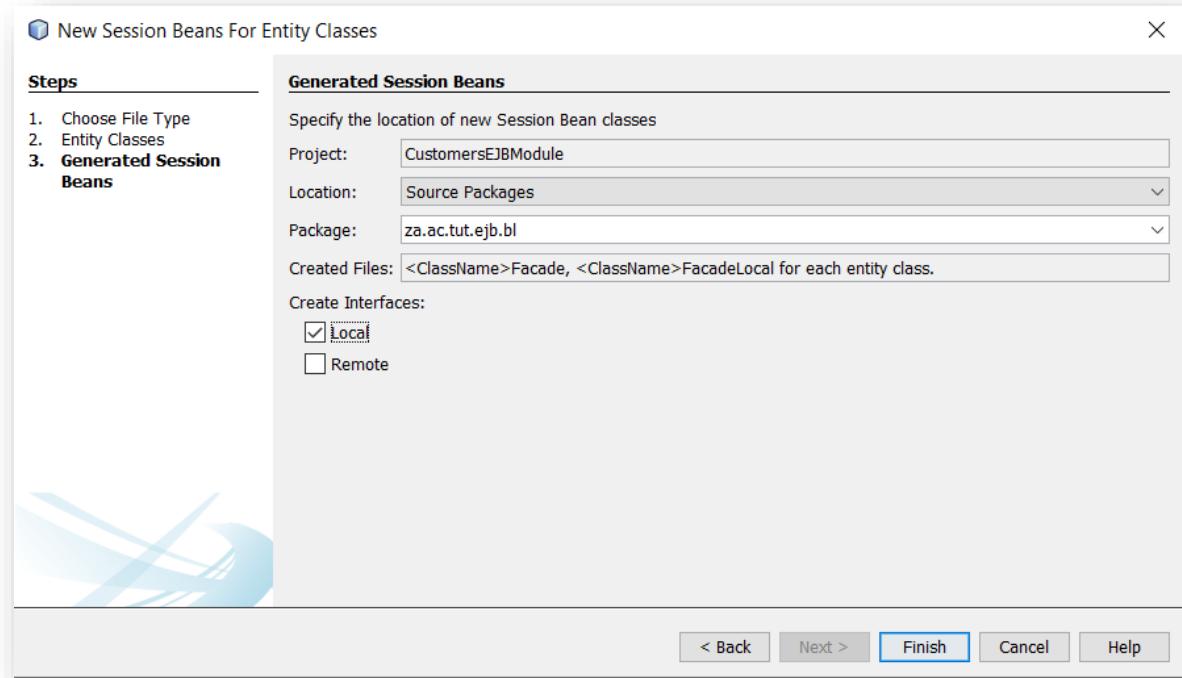
✓ Click Add.



✓ Click Next.



- ✓ Modify the package name to **za.ac.tut.ejb.bl**, where **bl** stands for **business logic**. Also select **Local** interface.



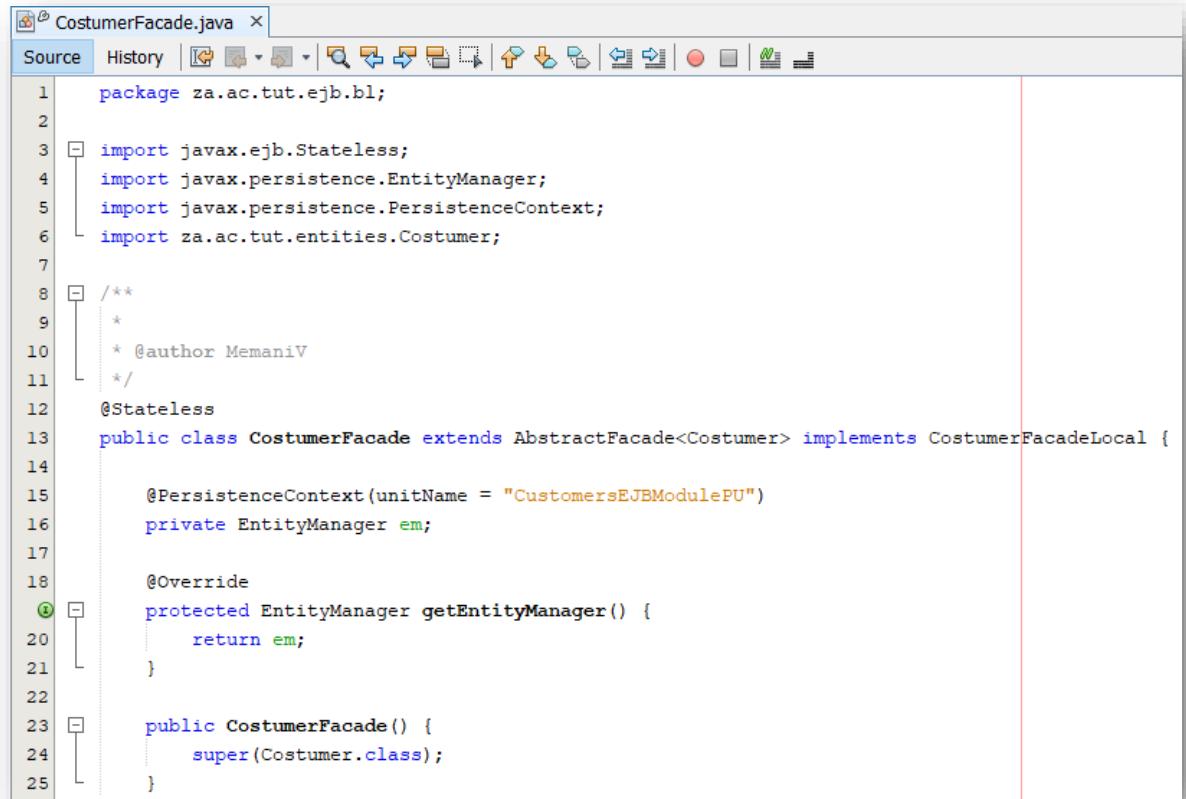
- ✓ Click **Finish**. View created files.
▪ **ContactFacade**.

```

1 package za.ac.tut.ejb.bl;
2
3 import javax.ejb.Stateless;
4 import javax.persistence.EntityManager;
5 import javax.persistence.PersistenceContext;
6 import za.ac.tut.entities.Contact;
7
8 /**
9 *
10 * @author MemaniV
11 */
12 @Stateless
13 public class ContactFacade extends AbstractFacade<Contact> implements ContactFacadeLocal {
14
15     @PersistenceContext(unitName = "CustomersEJBMModulePU")
16     private EntityManager em;
17
18     @Override
19     protected EntityManager getEntityManager() {
20         return em;
21     }
22
23     public ContactFacade() {
24         super(Contact.class);
25     }
26
27 }

```

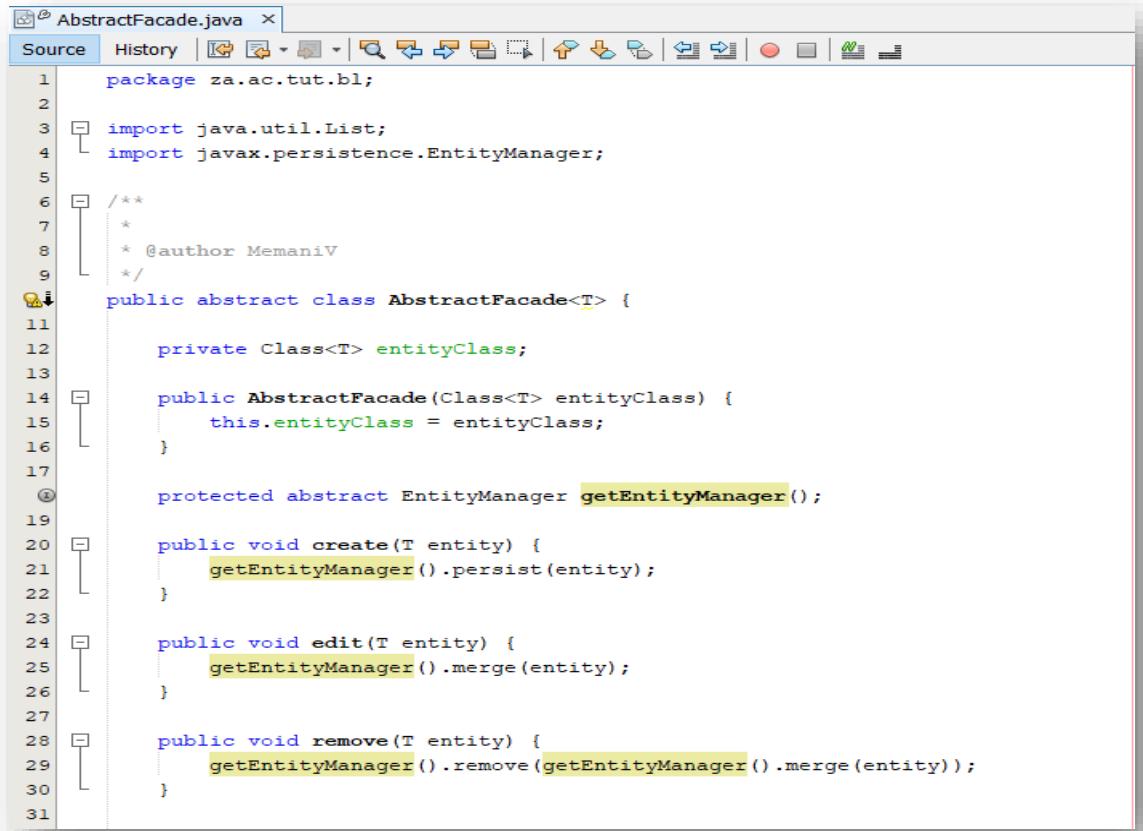
▪ CustomerFacade



The screenshot shows a Java code editor with the file 'CostumerFacade.java' open. The code implements a stateless EJB facade for a 'Costumer' entity. It imports javax.ejb.Stateless, javax.persistence.EntityManager, javax.persistence.PersistenceContext, and za.ac.tut.entities.Costumer. The class is annotated with @Stateless and has a constructor that calls super(Costumer.class). It overrides the getEntityManager method to return the private EntityManager em. The code also includes persistence context annotations (@PersistenceContext(unitName = "CustomersEJBModulePU")) and JBoss Seam annotations (@author MemaniV).

```
1 package za.ac.tut.ejb.bl;
2
3 import javax.ejb.Stateless;
4 import javax.persistence.EntityManager;
5 import javax.persistence.PersistenceContext;
6 import za.ac.tut.entities.Costumer;
7
8 /**
9  * @author MemaniV
10 */
11 @Stateless
12 public class CostumerFacade extends AbstractFacade<Costumer> implements CostumerFacadeLocal {
13
14     @PersistenceContext(unitName = "CustomersEJBModulePU")
15     private EntityManager em;
16
17
18     @Override
19     protected EntityManager getEntityManager() {
20         return em;
21     }
22
23     public CostumerFacade() {
24         super(Costumer.class);
25     }
26 }
```

▪ AbstractFacade.



The screenshot shows a Java code editor with the file 'AbstractFacade.java' open. It defines an abstract class 'AbstractFacade' that takes a type parameter T. The class has a private field 'entityClass' of type Class<T>. It has a constructor that takes an entityClass and initializes the private field. It also has protected abstract methods for EntityManager and void methods for create, edit, and remove operations. The EntityManager methods are implemented using the EntityManager's persist, merge, and remove methods respectively.

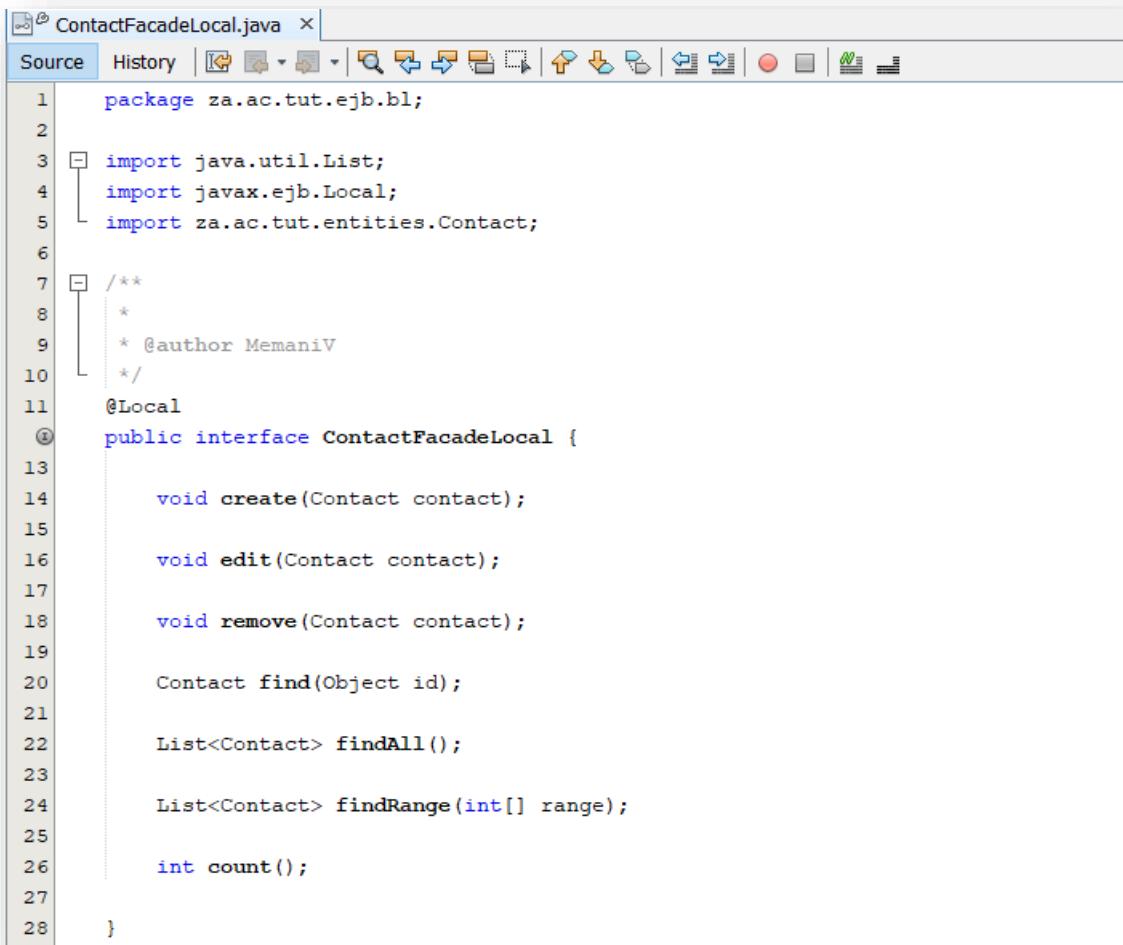
```
1 package za.ac.tut.bl;
2
3 import java.util.List;
4 import javax.persistence.EntityManager;
5
6 /**
7  * @author MemaniV
8 */
9 public abstract class AbstractFacade<T> {
10
11     private Class<T> entityClass;
12
13     public AbstractFacade(Class<T> entityClass) {
14         this.entityClass = entityClass;
15     }
16
17     protected abstract EntityManager getEntityManager();
18
19     public void create(T entity) {
20         getEntityManager().persist(entity);
21     }
22
23     public void edit(T entity) {
24         getEntityManager().merge(entity);
25     }
26
27     public void remove(T entity) {
28         getEntityManager().remove(getEntityManager().merge(entity));
29     }
30 }
```

```

32     public T find(Object id) {
33         return getEntityManager().find(entityClass, id);
34     }
35
36     public List<T> findAll() {
37         javax.persistence.criteria.CriteriaQuery cq =
38             getEntityManager().getCriteriaBuilder().createQuery();
39         cq.select(cq.from(entityClass));
40         return getEntityManager().createQuery(cq).getResultList();
41     }
42
43     public List<T> findRange(int[] range) {
44         javax.persistence.criteria.CriteriaQuery cq =
45             getEntityManager().getCriteriaBuilder().createQuery();
46         cq.select(cq.from(entityClass));
47         javax.persistence.Query q = getEntityManager().createQuery(cq);
48         q.setMaxResults(range[1] - range[0] + 1);
49         q.setFirstResult(range[0]);
50         return q.getResultList();
51     }
52
53     public int count() {
54         javax.persistence.criteria.CriteriaQuery cq =
55             getEntityManager().getCriteriaBuilder().createQuery();
56         javax.persistence.criteria.Root<T> rt = cq.from(entityClass);
57         cq.select(getEntityManager().getCriteriaBuilder().count(rt));
58         javax.persistence.Query q = getEntityManager().createQuery(cq);
59         return ((Long) q.getSingleResult()).intValue();
60     }
61 }
62

```

▪ ContactFacadeLocal.



The screenshot shows the Java code for the `ContactFacadeLocal` interface in an IDE. The code defines a local interface for managing contacts. It includes imports for `List`, `java.util.List`, `javax.ejb.Local`, and `za.ac.tut.entities.Contact`. The interface is annotated with `@Local`. It contains methods for creating, editing, removing, finding by ID, finding all, finding a range, and counting contacts.

```

1 package za.ac.tut.ejb.bl;
2
3 import java.util.List;
4 import javax.ejb.Local;
5 import za.ac.tut.entities.Contact;
6
7 /**
8 * @author MemaniV
9 */
10 @Local
11 public interface ContactFacadeLocal {
12
13     void create(Contact contact);
14
15     void edit(Contact contact);
16
17     void remove(Contact contact);
18
19     Contact find(Object id);
20
21     List<Contact> findAll();
22
23     List<Contact> findRange(int[] range);
24
25     int count();
26 }

```

- **CustomerFacadeLocal.**

The screenshot shows a Java code editor window titled "CostumerFacadeLocal.java". The code is a Java interface for a local EJB. It includes imports for java.util.List, javax.ejb.Local, and za.ac.tut.entities.Costumer. The interface contains methods for creating, editing, removing, finding, and finding all Costumer entities, along with a count method. A copyright notice and author information are present at the top of the file.

```
1 package za.ac.tut.ejb.bl;
2
3 import java.util.List;
4 import javax.ejb.Local;
5 import za.ac.tut.entities.Costumer;
6
7 /**
8 * 
9 * @author MemaniV
10 */
11 @Local
12 public interface CostumerFacadeLocal {
13
14     void create(Costumer costumer);
15
16     void edit(Costumer costumer);
17
18     void remove(Costumer costumer);
19
20     Costumer find(Object id);
21
22     List<Costumer> findAll();
23
24     List<Costumer> findRange(int[] range);
25
26     int count();
27
28 }
```

✓ Clean and build the EJB project.



The screenshot shows the NetBeans IDE interface. At the top, there is a notifications window titled "Output - CustomersEJBModule (clean, dist) x". The window displays the build log:

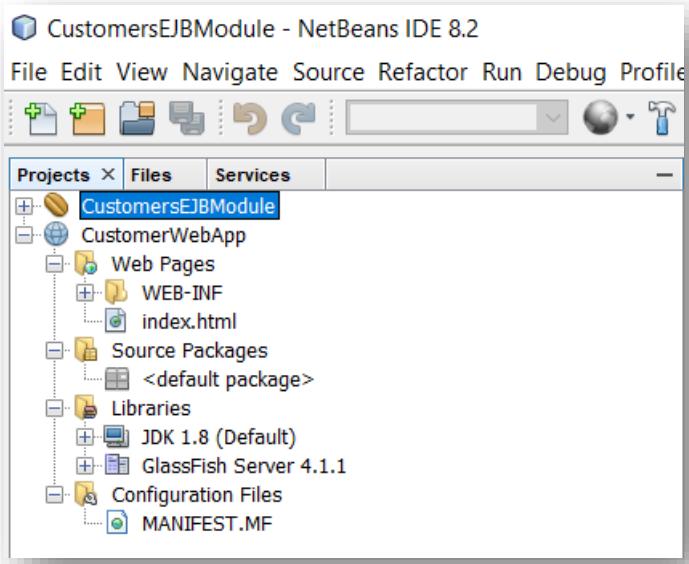
```
Compiling 7 source files to C:\Users\memaniv\Documents\NetBeansProjects\CustomersEJBModule\build\classes
warning: Supported source version 'RELEASE_6' from annotation processor 'org.eclipse.persistence.internal.jpa.modelgen.processor.JPAParticipant'.
Note: Creating static metadata factory ...
Note: Optional file was not found: META-INF/orm.xml continuing with generation.
Note: Optional file was not found: META-INF/eclipselink-orm.xml continuing with generation.
Note: Optional file was not found: META-INF/orm.xml continuing with generation.
Note: Optional file was not found: META-INF/eclipselink-orm.xml continuing with generation.
Note: C:\Users\memaniv\Documents\NetBeansProjects\CustomersEJBModule\src\java\za\ac\tut\ejb\bl\AbstractFacade.java
Note: Recompile with -Xlint:unchecked for details.
1 warning
compile:
library-inclusion-in-archive:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\CustomersEJBModule\dist
Building jar: C:\Users\memaniv\Documents\NetBeansProjects\CustomersEJBModule\dist\CustomersEJBModule.jar
dist:
BUILD SUCCESSFUL (total time: 1 second)
```

Below the notifications window is the main NetBeans window. The title bar says "CustomersEJBModule - NetBeans IDE 8.2". The menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, and Profile. The toolbar has icons for new file, open file, save, and others. The Projects tab is selected in the left sidebar, showing the project structure:

- CustomersEJBModule
 - build
 - dist
 - CustomersEJBModule.jar
 - META-INF
 - MANIFEST.MF
 - persistence.xml
 - za.ac.tut.ejb.bl
 - AbstractFacade.class
 - ContactFacade.class
 - ContactFacadeLocal.class
 - CostumerFacade.class
 - CostumerFacadeLocal.class
 - za.ac.tut.entities
 - Contact.class
 - Contact_.class
 - Costumer.class
 - Costumer_.class
 - nbproject
 - src
 - test
 - build.xml

Part D – Create a web client project.

Create a web client project called **CustomerWebApp**.

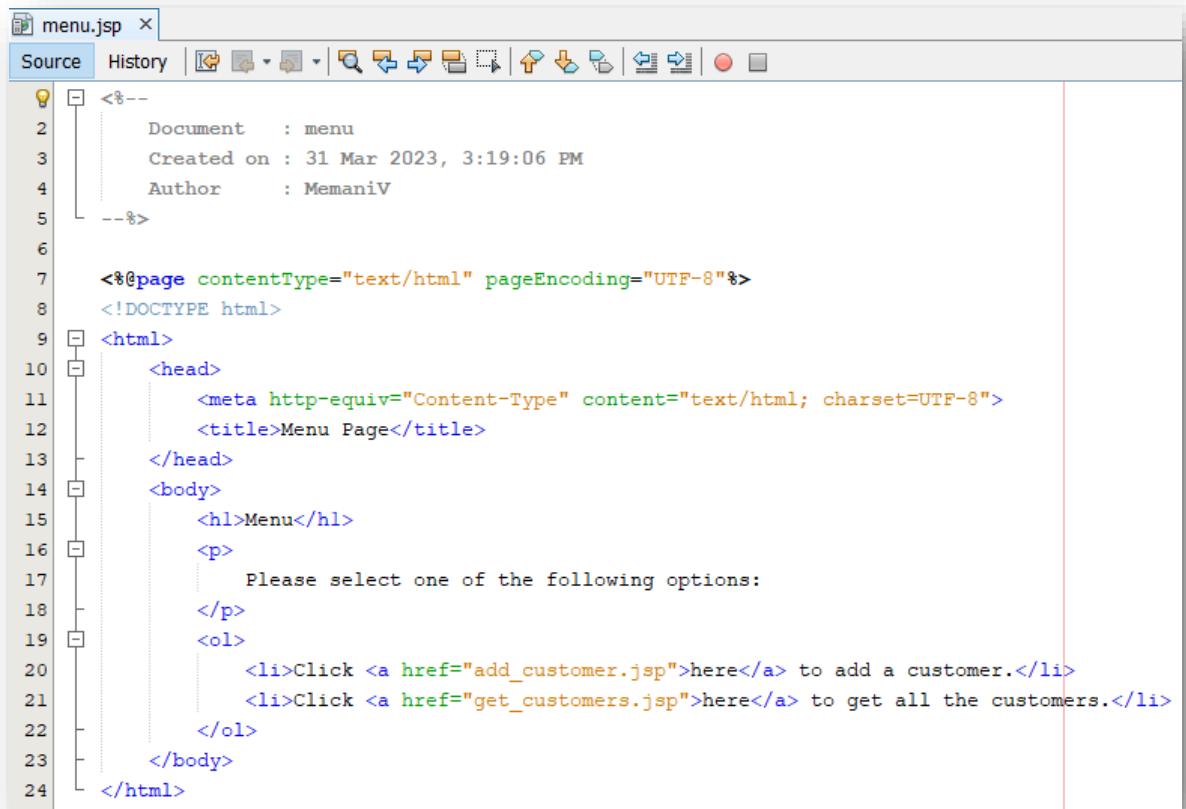


Edit the **index.html** page.

The screenshot shows the NetBeans code editor with the tab "index.html" selected. The toolbar above the editor includes tabs for Source, History, and several icons for file operations. The code editor displays the following HTML content:

```
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 page</h1>
15    <p>
16      Welcome to our web app. Please click <a href="menu.jsp">here</a> to start.
17    </p>
18  </body>
19 </html>
```

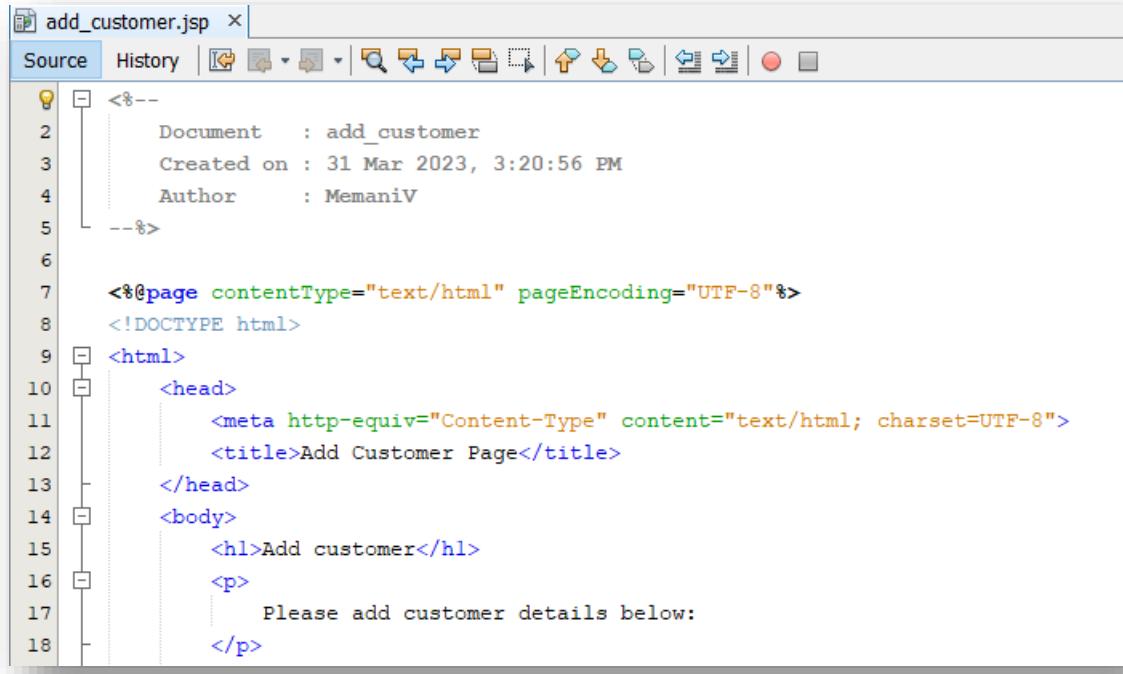
Create the menu.jsp file.



The screenshot shows the code editor of a Java IDE with the file "menu.jsp" open. The code is a JSP page that displays a menu. It includes comments at the top, meta tags, and a list of links for adding and getting customers.

```
<%--  
Document : menu  
Created on : 31 Mar 2023, 3:19:06 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="add_customer.jsp">here</a> to add a customer.</li>  
            <li>Click <a href="get_customers.jsp">here</a> to get all the customers.</li>  
        </ol>  
    </body>  
</html>
```

Create add_customer.jsp file.



The screenshot shows the code editor of a Java IDE with the file "add_customer.jsp" open. The code is a JSP page that adds a new customer. It includes meta tags and a form for inputting customer details.

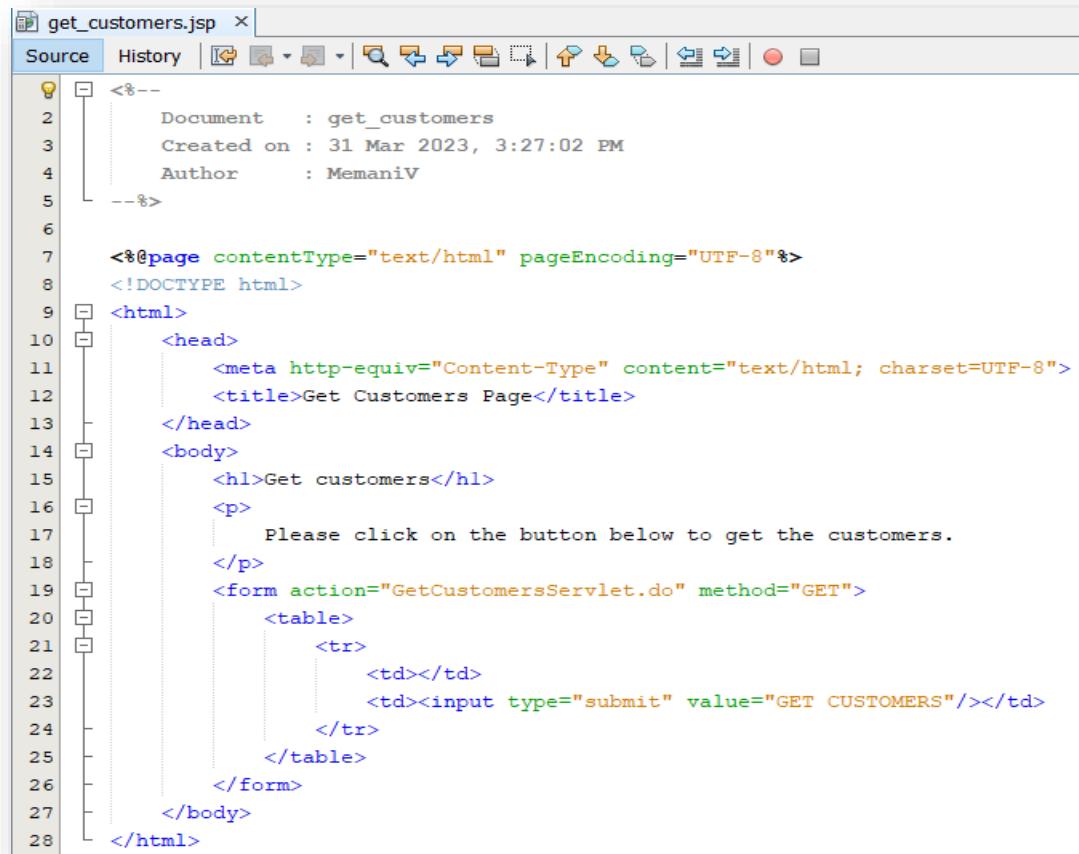
```
<%--  
Document : add_customer  
Created on : 31 Mar 2023, 3:20:56 PM  
Author : MemaniV  
--%>  
  
<%@page contentType="text/html" pageEncoding="UTF-8"%>  
<!DOCTYPE html>  
<html>  
    <head>  
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
        <title>Add Customer Page</title>  
    </head>  
    <body>  
        <h1>Add customer</h1>  
        <p>  
            Please add customer details below:  
        </p>
```

```

19 <form action="AddCustomerServlet.do" method="POST">
20   <table>
21     <tr>
22       <td>Customer number: </td>
23       <td><input type="text" name="custID"/></td>
24     </tr>
25     <tr>
26       <td>Name: </td>
27       <td><input type="text" name="name"/></td>
28     </tr>
29     <tr>
30       <td>Surname: </td>
31       <td><input type="text" name="surname"/></td>
32     </tr>
33     <tr>
34       <td>First cell number: </td>
35       <td><input type="text" name="cellNo1"/></td>
36     </tr>
37     <tr>
38       <td>Second cell number: </td>
39       <td><input type="text" name="cellNo2"/></td>
40     </tr>
41     <tr>
42       <td></td>
43       <td><input type="submit" value="ADD CUSTOMER"/></td>
44     </tr>
45   </table>
46 </form>
47 </body>
48 </html>

```

Create the **get_customers.jsp** file.



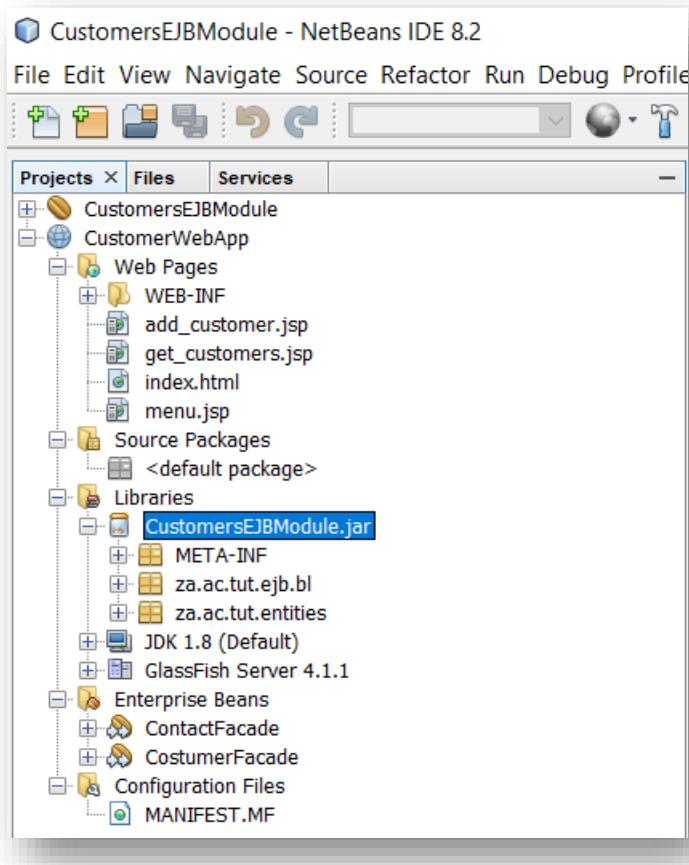
```

<%@--%
Document      : get_customers
Created on   : 31 Mar 2023, 3:27:02 PM
Author        : MemaniV
--%>

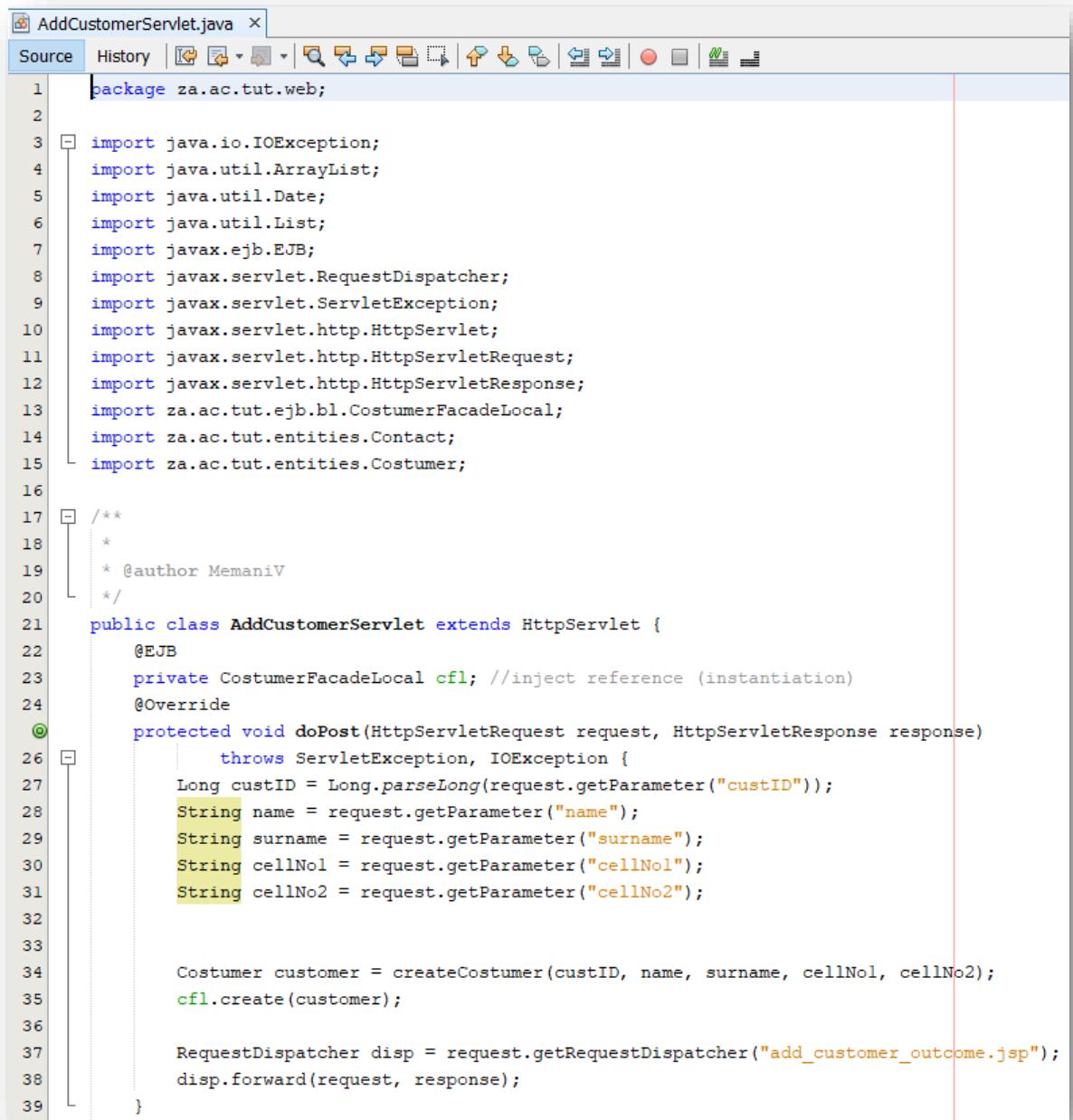
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Get Customers Page</title>
  </head>
  <body>
    <h1>Get customers</h1>
    <p>
      Please click on the button below to get the customers.
    </p>
    <form action="GetCustomersServlet.do" method="GET">
      <table>
        <tr>
          <td></td>
          <td><input type="submit" value="GET CUSTOMERS"/></td>
        </tr>
      </table>
    </form>
  </body>
</html>

```

Add the **CustomersEJBModule.jar** file to the library of **CustomerWebApp**.



Create the AddCustomerServlet.java file.



```
1 package za.ac.tut.web;
2
3 import java.io.IOException;
4 import java.util.ArrayList;
5 import java.util.Date;
6 import java.util.List;
7 import javax.ejb.EJB;
8 import javax.servlet.RequestDispatcher;
9 import javax.servlet.ServletException;
10 import javax.servlet.http.HttpServlet;
11 import javax.servlet.http.HttpServletRequest;
12 import javax.servlet.http.HttpServletResponse;
13 import za.ac.tut.ejb.bl.CostumerFacadeLocal;
14 import za.ac.tut.entities.Contact;
15 import za.ac.tut.entities.Costumer;
16
17 /**
18 *
19 * @author MemaniV
20 */
21 public class AddCustomerServlet extends HttpServlet {
22     @EJB
23     private CostumerFacadeLocal cfl; //inject reference (instantiation)
24     @Override
25     protected void doPost(HttpServletRequest request, HttpServletResponse response)
26             throws ServletException, IOException {
27         Long custID = Long.parseLong(request.getParameter("custID"));
28         String name = request.getParameter("name");
29         String surname = request.getParameter("surname");
30         String cellNol = request.getParameter("cellNol");
31         String cellNo2 = request.getParameter("cellNo2");
32
33
34         Costumer customer = createCostumer(custID, name, surname, cellNol, cellNo2);
35         cfl.create(customer);
36
37         RequestDispatcher disp = request.getRequestDispatcher("add_customer_outcome.jsp");
38         disp.forward(request, response);
39     }
40 }
```

```
41     private Costumer createCostumer(Long custID, String name, String surname, String cellNol, String cellNo2) {
42         Costumer customer = new Costumer();
43         List<Contact> contacts = new ArrayList<>();
44         Contact contact1 = new Contact();
45         Contact contact2 = new Contact();
46
47         contact1.setCellNo(cellNol);
48         contact2.setCellNo(cellNo2);
49         contacts.add(contact1);
50         contacts.add(contact2);
51
52         customer.setId(custID);
53         customer.setName(name);
54         customer.setSurname(surname);
55         customer.setContacts(contacts);
56         customer.setCreationDate(new Date());
57         return customer;
58     }
59 }
```

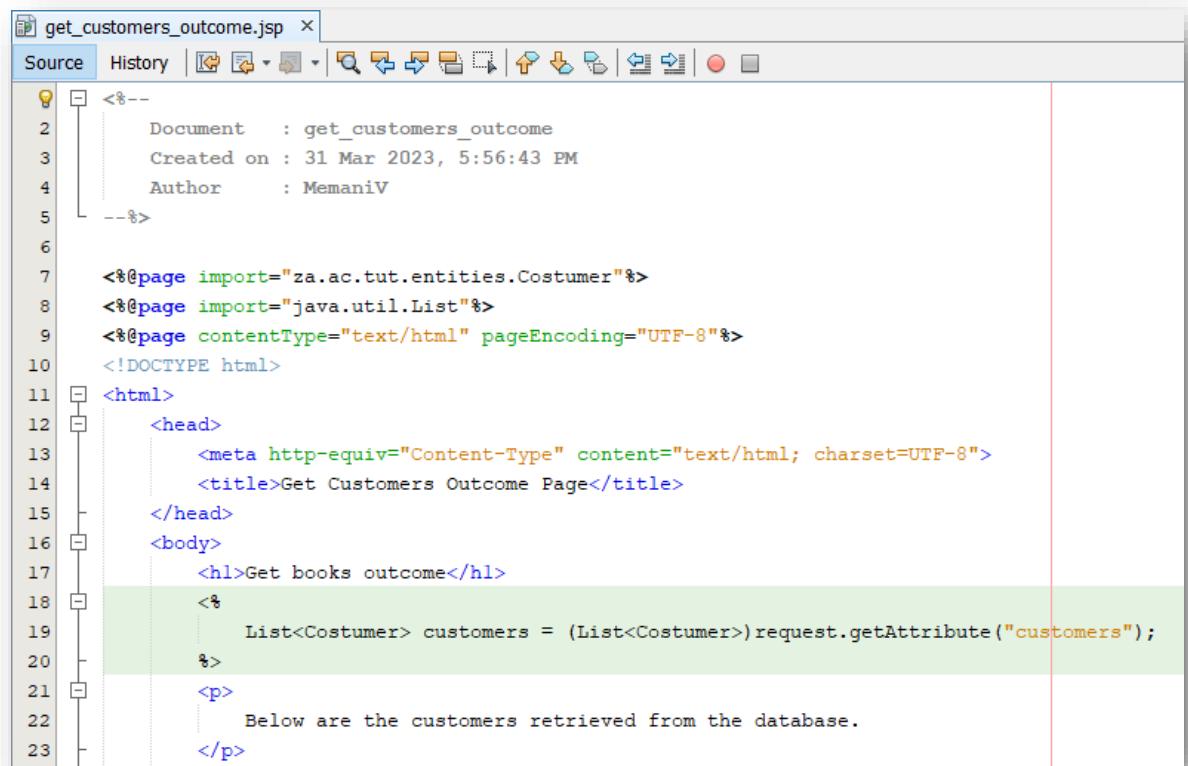
Create the **GetCustomersServlet.java** file.

The screenshot shows a Java code editor with the file `GetCustomersServlet.java` open. The code implements a servlet for retrieving customer data from an EJB. It includes imports for various Java and JEE packages, a Javadoc comment, and a doGet method that retrieves all customers from the database and forwards the request to a JSP page.

```
1 package za.ac.tut.web;
2
3 import java.io.IOException;
4 import java.util.List;
5 import javax.ejb.EJB;
6 import javax.servlet.RequestDispatcher;
7 import javax.servlet.ServletException;
8 import javax.servlet.http.HttpServlet;
9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11 import za.ac.tut.ejb.bl.CostumerFacadeLocal;
12 import za.ac.tut.entities.Costumer;
13
14 /**
15 *
16 * @author MemaniV
17 */
18 public class GetCustomersServlet extends HttpServlet {
19     @EJB
20     private CostumerFacadeLocal cfl; //inject reference (instantiation)
21
22     @Override
23     @Override
24     protected void doGet(HttpServletRequest request, HttpServletResponse response)
25             throws ServletException, IOException {
26         List<Costumer> customers = cfl.findAll(); //get customers
27         request.setAttribute("customers", customers);
28
29         RequestDispatcher disp = request.getRequestDispatcher("get_customers_outcome.jsp");
30         disp.forward(request, response);
31     }
32 }
```

Create the **add_customer_outcome.jsp** file.

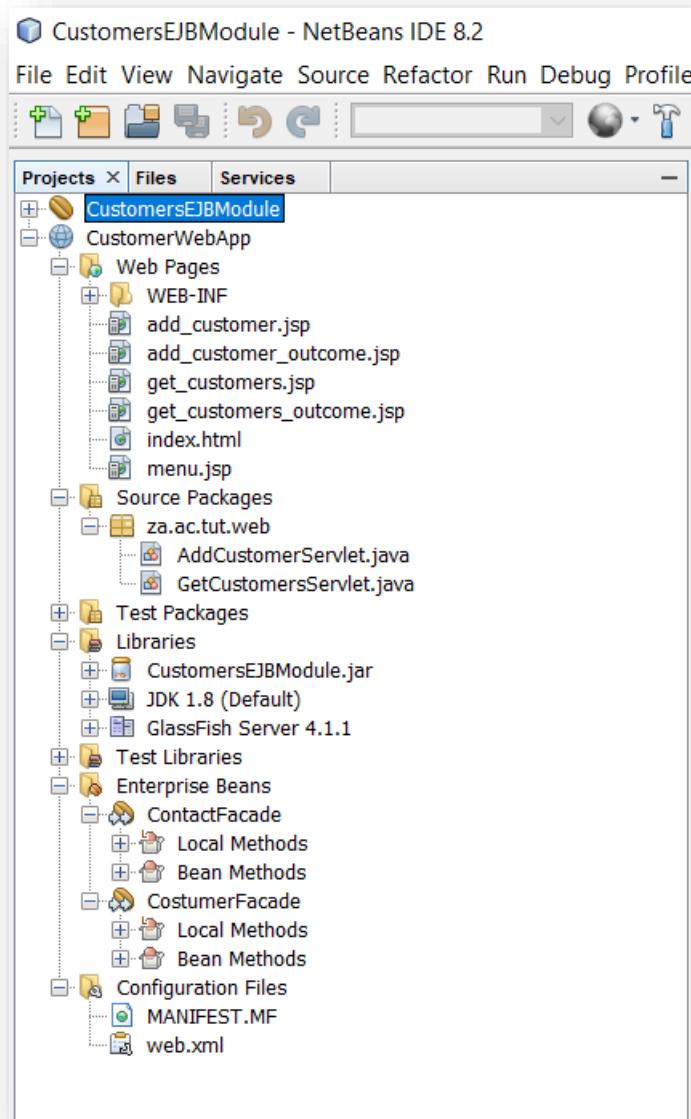
Create the **get_customers_outcome.jsp** file.



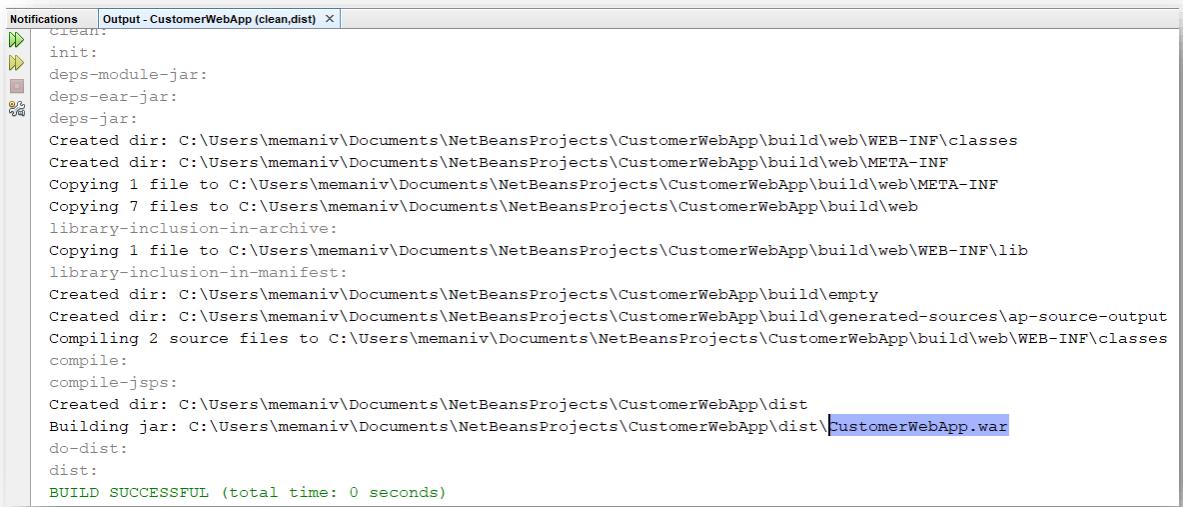
```
<%--  
1      Document      : get_customers_outcome  
2      Created on   : 31 Mar 2023, 5:56:43 PM  
3      Author        : MemaniV  
4--%>  
5  
6  
7  <%@page import="za.ac.tut.entities.Costumer"%>  
8  <%@page import="java.util.List"%>  
9  <%@page contentType="text/html" pageEncoding="UTF-8"%>  
10 <!DOCTYPE html>  
11 <html>  
12  <head>  
13    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
14    <title>Get Customers Outcome Page</title>  
15  </head>  
16  <body>  
17    <h1>Get books outcome</h1>  
18    <%  
19      List<Costumer> customers = (List<Costumer>)request.getAttribute("customers");  
20    %>  
21    <p>  
22      Below are the customers retrieved from the database.  
23    </p>
```

```
24 <table>
25     <%
26         for(int i = 0; i < customers.size(); i++){
27             Costumer p = customers.get(i);
28             Long id = p.getId();
29             String name = p.getName();
30             String surname = p.getSurname();
31             String cellNol = p.getContacts().get(0).getCellNo();
32             String cellNo2 = p.getContacts().get(1).getCellNo();
33         %}
34     <tr>
35         <td>Customer number:</td>
36         <td><%=id%></td>
37     </tr>
38     <tr>
39         <td>Name:</td>
40         <td><%=name%></td>
41     </tr>
42     <tr>
43         <td>Surname:</td>
44         <td><%=surname%></td>
45     </tr>
46     <tr>
47         <td>Cell number 1:</td>
48         <td><%=cellNol%></td>
49     </tr>
50     <tr>
51         <td>Cell number 2: </td>
52         <td><%=cellNo2%></td>
53     </tr>
54     <%
55     }
56     %>
57 </table>
58 <p>
59     Please click <a href="menu.jsp">here</a> to get back to the
60     menu page or <a href="index.html">here</a> to the main page.
61 </p>
62 </body>
63 </html>
```

View the complete project structure of **CustomerWebApp**.

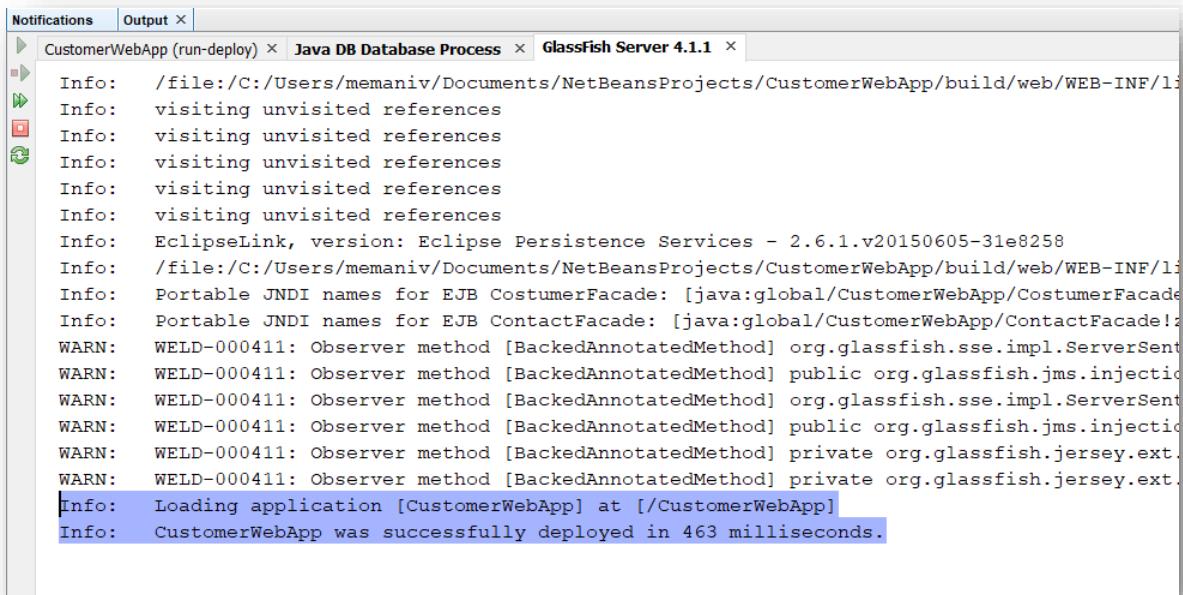


Compile the project.



```
Notifications Output - CustomerWebApp (clean,dist) ×
Clean:
init:
deps-module-jar:
deps-ear-jar:
deps-jar:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\CustomerWebApp\build\web\WEB-INF\classes
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\CustomerWebApp\build\web\META-INF
Copying 1 file to C:\Users\memaniv\Documents\NetBeansProjects\CustomerWebApp\build\web\META-INF
Copying 7 files to C:\Users\memaniv\Documents\NetBeansProjects\CustomerWebApp\build\web
library-inclusion-in-archive:
Copying 1 file to C:\Users\memaniv\Documents\NetBeansProjects\CustomerWebApp\build\web\WEB-INF\lib
library-inclusion-in-manifest:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\CustomerWebApp\build\empty
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\CustomerWebApp\build\generated-sources\ap-source-output
Compiling 2 source files to C:\Users\memaniv\Documents\NetBeansProjects\CustomerWebApp\build\web\WEB-INF\classes
compile:
compile-jsp:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\CustomerWebApp\dist
Building jar: C:\Users\memaniv\Documents\NetBeansProjects\CustomerWebApp\dist\CustomerWebApp.war
do-dist:
dist:
BUILD SUCCESSFUL (total time: 0 seconds)
```

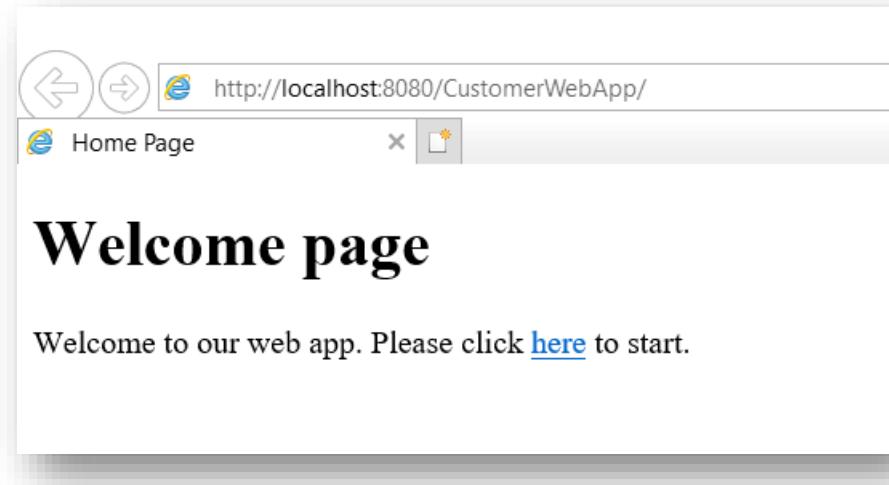
Deploy the project.



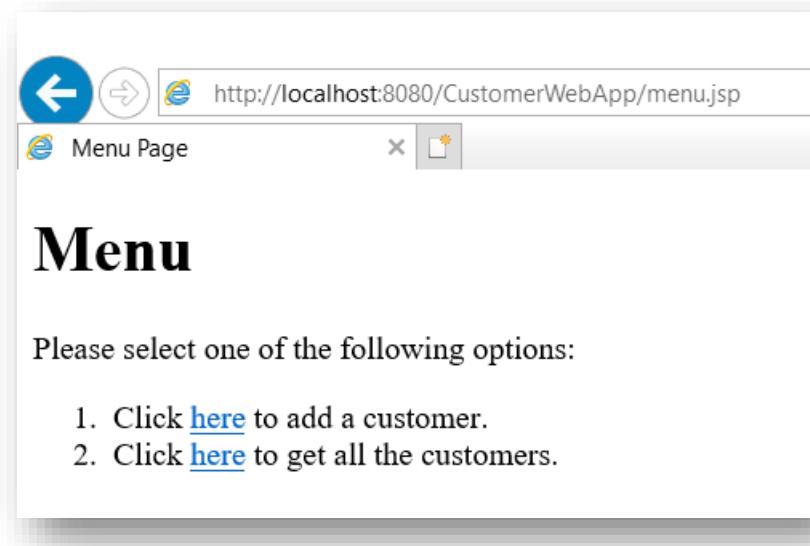
```
Notifications Output × Java DB Database Process × GlassFish Server 4.1.1 ×
CustomerWebApp (run-deploy) ×
Info: /file:/C:/Users/memaniv/Documents/NetBeansProjects/CustomerWebApp/build/web/WEB-INF/lib
Info: visiting unvisited references
Info: EclipseLink, version: Eclipse Persistence Services - 2.6.1.v20150605-31e8258
Info: /file:/C:/Users/memaniv/Documents/NetBeansProjects/CustomerWebApp/build/web/WEB-INF/lib
Info: Portable JNDI names for EJB CostumerFacade: [java:global/CustomerWebApp/CostumerFacade]
Info: Portable JNDI names for EJB ContactFacade: [java:global/CustomerWebApp/ContactFacade]
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] org.glassfish.sse.impl.ServerSent
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] public org.glassfish.jms.injection
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] org.glassfish.sse.impl.ServerSent
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] public org.glassfish.jms.injection
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] private org.glassfish.jersey.ext.
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] private org.glassfish.jersey.ext.
Info: Loading application [CustomerWebApp] at [/CustomerWebApp]
Info: CustomerWebApp was successfully deployed in 463 milliseconds.
```

Part E – Run the web application.

Launch the application.



Click on the link



1. Click [here](#) to add a customer.
2. Click [here](#) to get all the customers.

Add a customer.

- ✓ Click on the first link.

Customer number:

Name:

Surname:

First cell number:

Second cell number:

ADD CUSTOMER

- ✓ Fill in the form.

Customer number:

Name:

Surname:

First cell number:

Second cell number:

ADD CUSTOMER

Click on the add button.

The screenshot shows a web browser window with the URL <http://localhost:8080/CustomerWebApp/AddCustomerServlet.do>. The title bar says "Add Customer Outcome Pa...". The main content area has a heading "Add customer outcome" and a message: "The customer has been successfully persisted into the database. Please click [here](#) to get back to the menu page or [here](#) to the main page."

Add four more customers. View all the customers.

- ✓ Click on the second link.

The screenshot shows a web browser window with the URL http://localhost:8080/CustomerWebApp/get_customers.jsp. The title bar says "Get Customers Page". The main content area has a heading "Get customers" and a message: "Please click on the button below to get the customers." Below the message is a button labeled "GET CUSTOMERS".

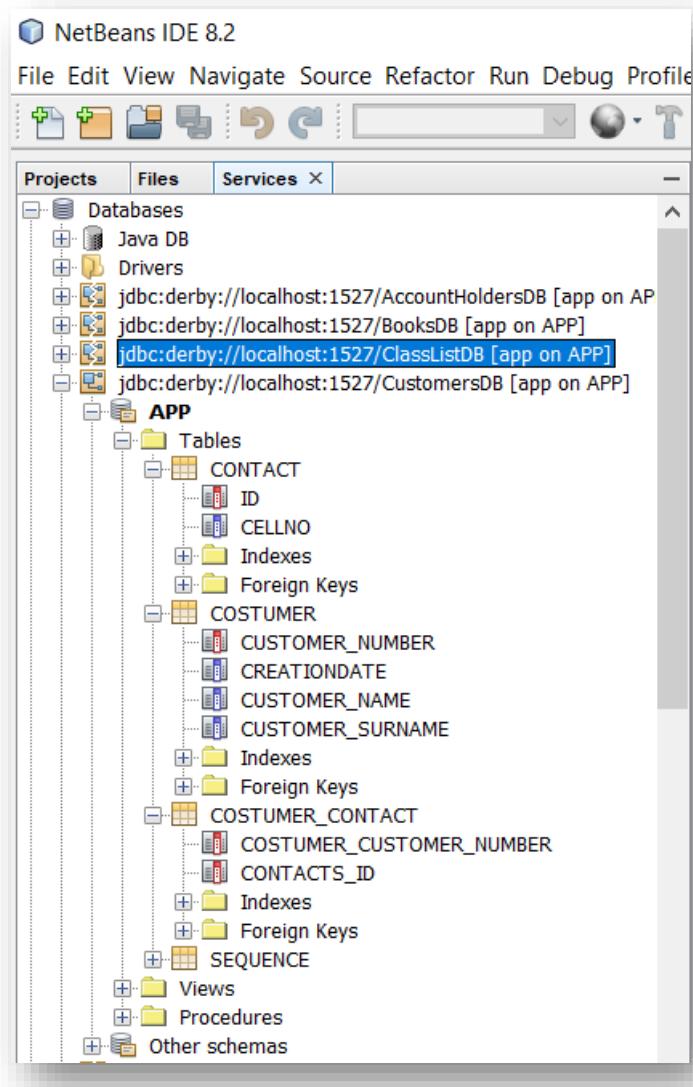
Click on the button.

The screenshot shows a web browser window with the following details:

- Address bar: http://localhost:8080/CustomerWebApp/GetCustomersServlet.do
- Title bar: Get Customers Outcome Pa... (partially visible)
- Content area:
 - Get books outcome**
 - Below the title, a message states: "Below are the customers retrieved from the database."
 - A list of customer records is displayed:
 - Customer number: 111
 - Name: Vuyisile
 - Surname: Memani
 - Cell number 1: 081
 - Cell number 2: 071
 - Customer number: 222
 - Name: Thato
 - Surname: Ranamane
 - Cell number 1: 061
 - Cell number 2: 078
 - Customer number: 333
 - Name: Patrice
 - Surname: Mulumba
 - Cell number 1: 068
 - Cell number 2: 081
 - Customer number: 444
 - Name: Sipho
 - Surname: Mbule
 - Cell number 1: 071
 - Cell number 2: 081
 - Customer number: 555
 - Name: Rendani
 - Surname: Baloyi
 - Cell number 1: 087
 - Cell number 2: 071
 - At the bottom of the content area, there is a message: "Please click [here](#) to get back to the menu page or [here](#) to the main page."

View the records in the database.

✓ View structure



Contents of the CONTACT table.

The screenshot shows the SQL3 tool within the NetBeans IDE. A query window displays the following SQL code:

```
1 | SELECT * FROM APP.CONTACT FETCH FIRST 100 ROWS ONLY;
2 | 
```

Below the query window, a result table shows the contents of the CONTACT table, with columns '#', 'ID', and 'CELLNO'. The data is as follows:

#	ID	CELLNO
1		1 081
2		2 071
3		4 078
4		3 061
5		5 068
6		6 081
7		8 081
8		7 071
9		10 071
10		9 087

Contents of the COSTUMER

The screenshot shows the SQL Developer interface with two tabs: SQL 1 and SQL 2. SQL 1 contains the query: `SELECT * FROM APP.COSTUMER FETCH FIRST 100 ROWS ONLY;`. SQL 2 displays the results of this query in a grid format:

#	CUSTOMER_NUMBER	CREATIONDATE	CUSTOMER_NAME	CUSTOMER_SURNAME
1		111 2023-03-31 18:09:10.736	Vuyisile	Memani
2		222 2023-03-31 18:10:15.378	Thato	Ranamane
3		333 2023-03-31 18:10:37.118	Patrice	Mulumba
4		444 2023-03-31 18:11:04.141	Sipho	Mbule
5		555 2023-03-31 18:12:18.121	Rendani	Baloyi

Contents of the COSTUMER_CONTACT

The screenshot shows the SQL Developer interface with two tabs: SQL 1 and SQL 2. SQL 1 contains the query: `SELECT * FROM APP.COSTUMER_CONTACT FETCH FIRST 100 ROWS ONLY;`. SQL 2 displays the results of this query in a grid format:

#	COSTUMER_CUSTOMER_NUMBER	CONTACTS_ID	Matching Rows:
1		111	1
2		111	2
3		222	3
4		222	4
5		333	5
6		333	6
7		444	7
8		444	8
9		555	9
10		555	10

8.5 Inheritance and JPA

Inheritance is a form of software reuse that allows a new class to be created from an existing one. The new class is called a subclass and old one a superclass. Consequently, the subclass possesses both the **form** and **functionality** of the superclass. By **form** we mean type, the subclass becomes the type of the superclass. By functionality we mean capability, the things that the subclass is capable of doing. So the subclass assumes or inherits all the methods possessed by the superclass. The subclass doesn't have to code the inherited methods from scratch, but can override them, that is change the inherited implementation.

So inheritance describes the “**is-a**” relationship existing between two classes. We say a new class “**is-a**” type of the old class. An example could be a **Student** who “is-a” a **Person**. Another example is an **Orange** which “**is-a**” **Fruit**.

JPA has three strategies to represent inheritance. These are:

- A single table per class hierarchy strategy.
- A joined-subclass strategy.
- A table per concrete class strategy.

Let us have a look at each of the strategies.

8.5.1 Single table per class hierarchy strategy

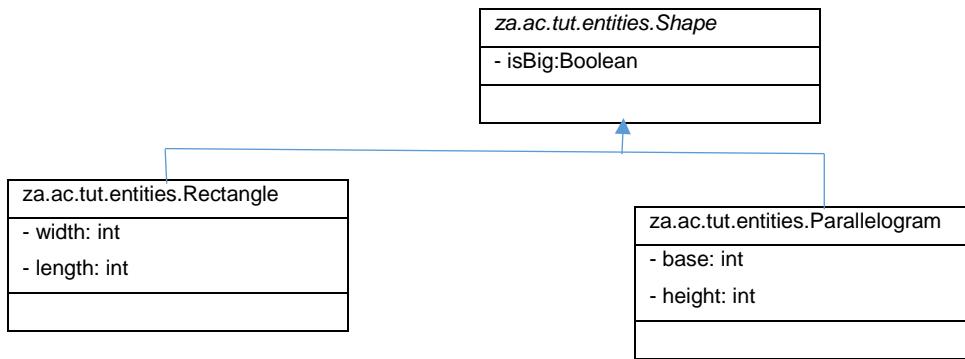
This is a default strategy. Using this strategy, a single table is created for the hierarchy. All the attributes of the superclass and subclasses are placed or mapped to a single table.

The root class can be annotated with the **@Inheritance** annotation. If the annotation is not explicitly written, the default inheritance strategy applied is the single table per class hierarchy. Let us take an example.

Example

Create a web application to represent a **Shape**. We have two shapes, a Parallelogram and a Rectangle. A Parallelogram has a **base** and a **height**, and a Rectangle has a **length** and a **width**. The area of Parallelogram can be calculated as **base * height** whilst that of a Rectangle can be calculated as **length * width**. Both shapes

have an extra attribute called `isBig`, which tells whether a shape is big or not. The inheritance class hierarchy can be represented as follows:



The equivalent table structure for the above hierarchy can be represented as follows:

SHAPE		
FIELD	DATA TYPE	CONSTRAINT
ID	INTEGER	PRIMARY KEY
ISBIG	BOOLEAN	NULLABLE=TRUE
WIDTH	INTEGER	NULLABLE=TRUE
LENGTH	INTEGER	NULLABLE = TRUE
BASE	INTEGER	NULLABLE = TRUE
HEIGHT	INTEGER	NULLABLE = TRUE

As can be seen, all the attributes of the classes are put in a single table. Let us do the actual coding.

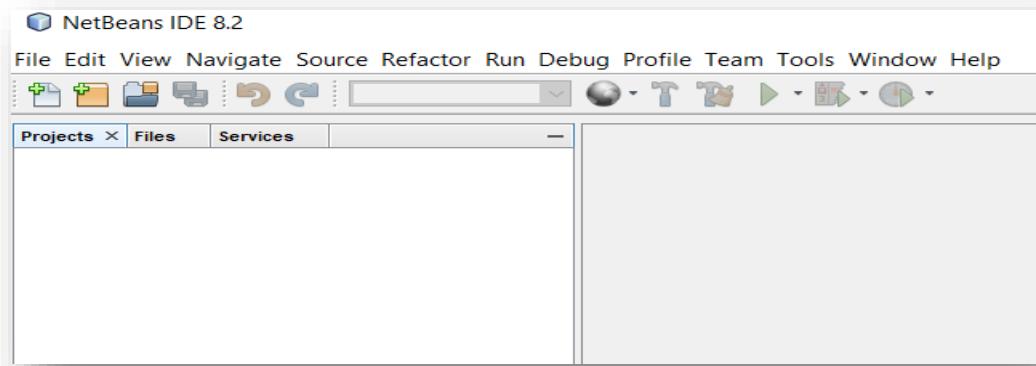
Solution approach

The solution to this problem is going to be done in five parts, namely:

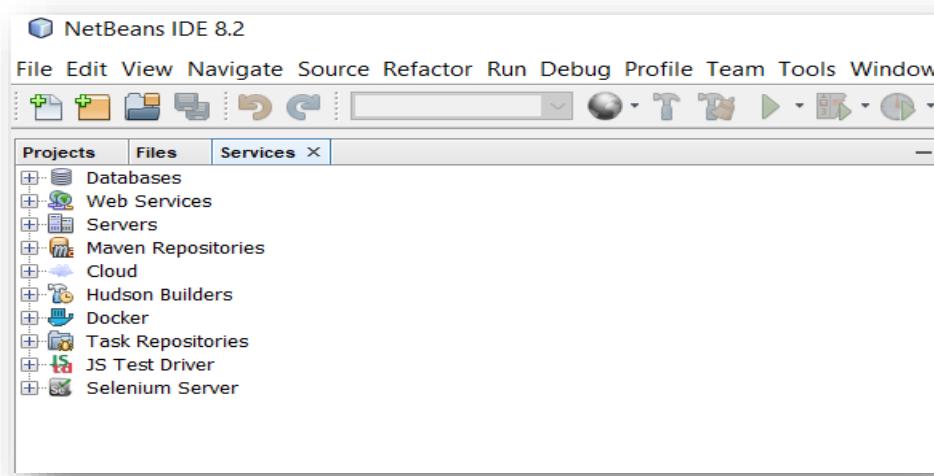
- **Part A:** we create a database.
- **Part B:** we establish connection between the created database and GlassFish.
- **Part C:** we create an EJB module. The module will have an entity with business logic.
- **Part D:** we create a client web application.
- **Part E:** we run the client web application.

Part A - Create a database

Launch NetBeans.



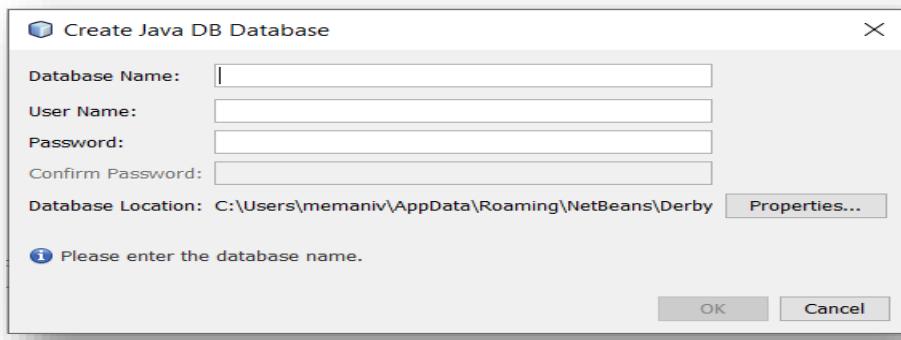
Click on the **Services** tab.



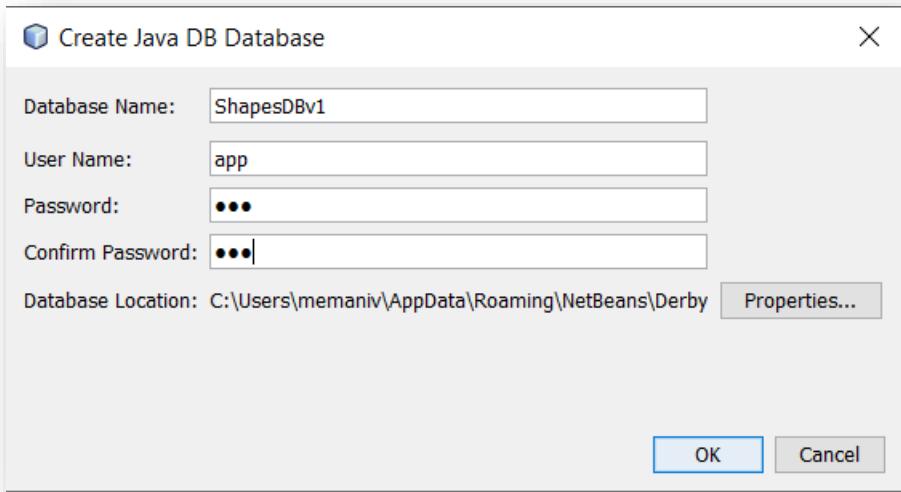
Expand the **Databases** option.



Right-click on **Java DB** and select **Create Database**.



Fill-in the form. I made the password to be **123**.

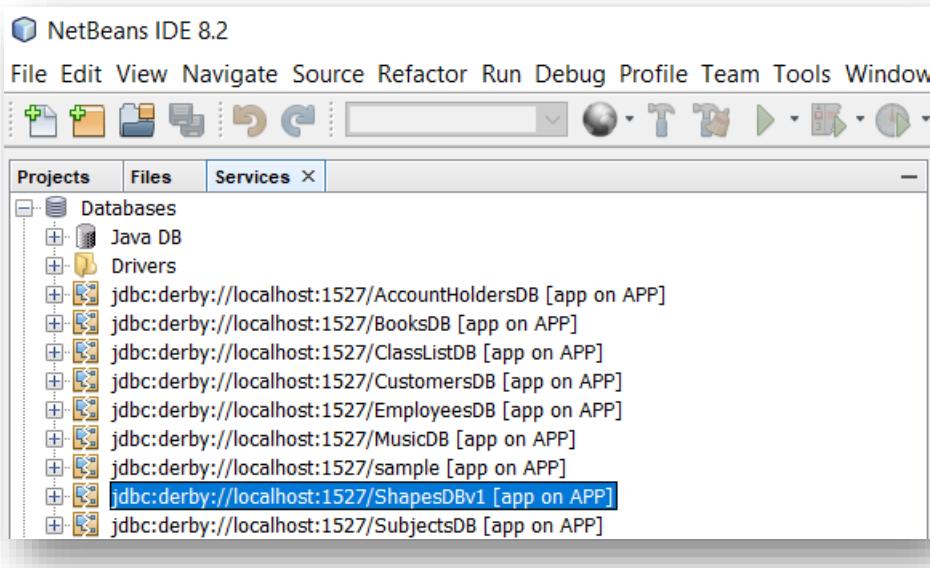


Click **OK**.

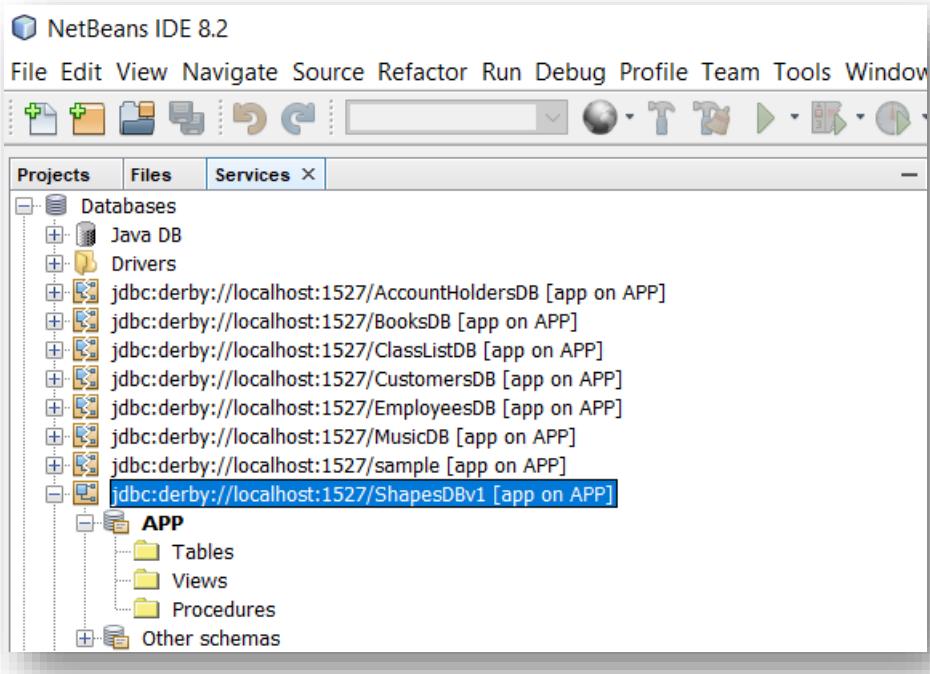
- ✓ An output message confirming that **Derby** has started and ready to accept connections on port 1527 is shown.

```
Output - Java DB Database Process X
Tue Apr 04 12:45:46 CAT 2023 : Security manager installed using the Basic server security policy.
Tue Apr 04 12:45:47 CAT 2023 : Apache Derby Network Server - 10.11.1.2 - (1629631) started and ready to accept connections on port 1527
```

- ✓ A database is created.

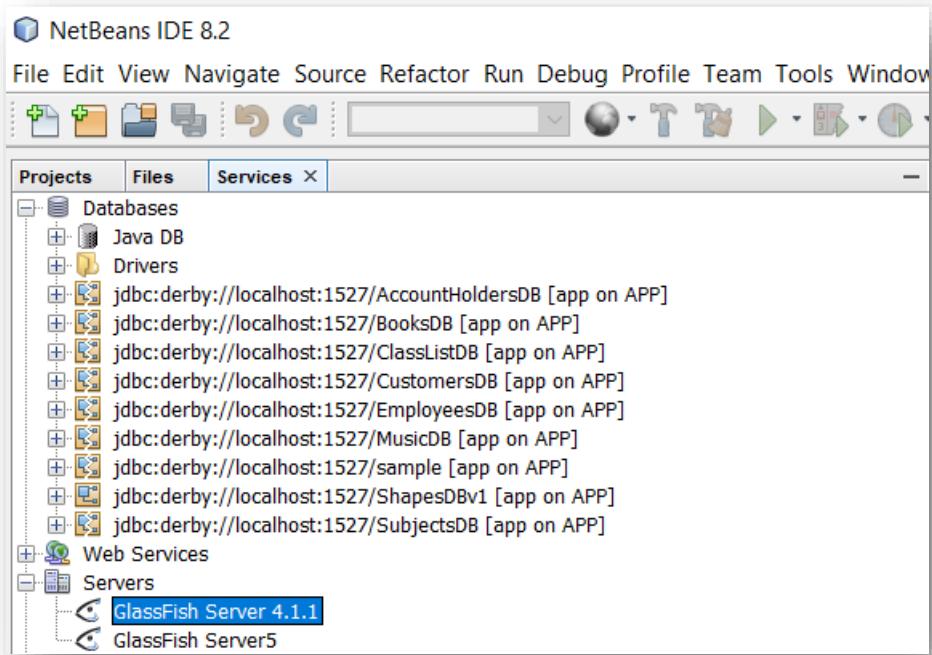


Right-click on the database and select **Connect**.

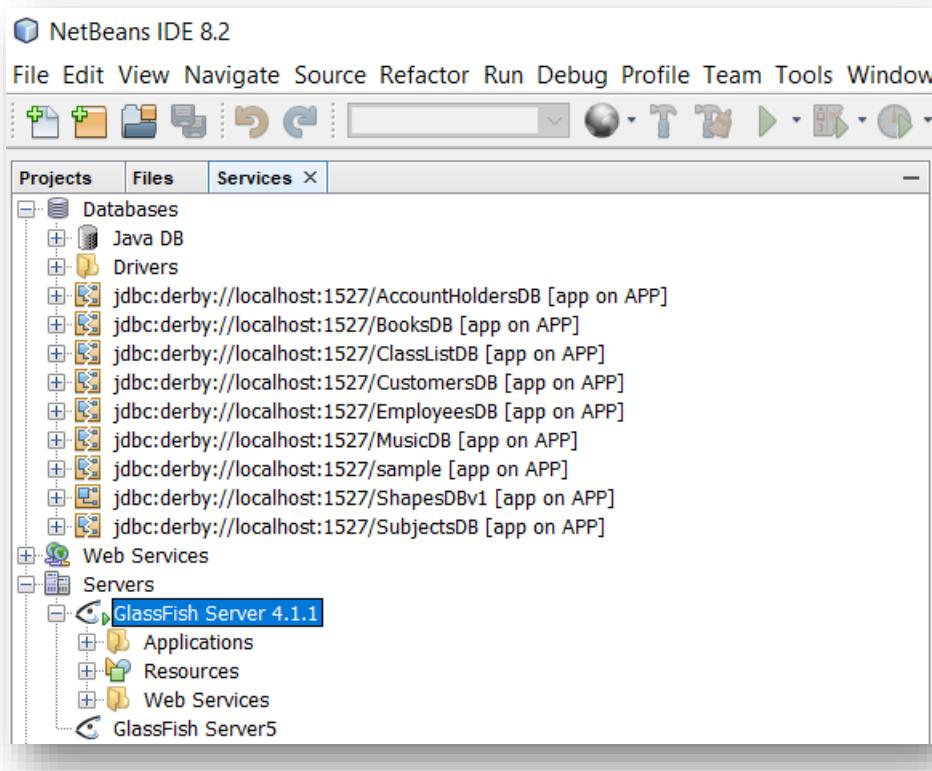


Part B – Connect the database to the application server.

Click on the **Services** tab.



Start the server.



Right-click on the server and select **View Domain Admin Console**.

The screenshot shows the GlassFish Admin Console interface. The URL in the address bar is `http://localhost:4848/common/index.jsf`. The title bar displays "GlassFish Console - Comm..." and "GlassFish™ Server Open Source Edition". The top menu bar includes "Home" and "About...". Below the menu, it shows "User: admin | Domain: domain1 | Server: localhost". The left sidebar is titled "Common Tasks" and contains the following categories: Domain, server (Admin Server), Clusters, Standalone Instances, Nodes, Applications, Lifecycle Modules, Monitoring Data, Resources (which is expanded to show Concurrent Resources, Connectors, JDBC, JMS Resources, JNDI, JavaMail Sessions, and Resource Adapter Configs), Configurations (which is expanded to show default-config and server-config), and Update Tool. The main content area is titled "GlassFish Console - Common Tasks" and includes sections for GlassFish News, Deployment (List Deployed Applications, Deploy an Application), Administration (Change Administrator Password, List Password Aliases), and Monitoring (Monitoring Data).

Under **Common Tasks** panel, expand **JDBC**.

This screenshot shows the same GlassFish Admin Console interface as above, but with the "Resources" category expanded in the "Common Tasks" sidebar. The "JDBC" item under "Resources" is specifically highlighted with a blue selection bar. The main content area is titled "JDBC" and lists "JDBC Resources" and "JDBC Connection Pools".

Modify the Connection pool to include the **ClassListDB**.

- ✓ Click on **JDBC Connection Pools**.

JDBC Connection Pools					
To store, organize, and retrieve data, most applications use relational databases. Java EE applications access relational databases through the JDBC API. Before an application can access a database, it must get a connection.					
Pools (3)					
Select	Pool Name	Resource Type	Classname	Description	
<input type="checkbox"/>	DerbyPool	javax.sql.DataSource	org.apache.derby.jdbc.ClientDataSource		
<input type="checkbox"/>	SamplePool	javax.sql.DataSource	org.apache.derby.jdbc.ClientDataSource		
<input type="checkbox"/>	TimerPool	javax.sql.XADatasource	org.apache.derby.jdbc.EmbeddedXADataSource		

- ✓ Click on the **DerbyPool** link.

General	Advanced	Additional Properties
Edit JDBC Connection Pool Modify an existing JDBC connection pool. A JDBC connection pool is a group of reusable connections for a particular database. <input type="button" value="Load Defaults"/> <input type="button" value="Flush"/> <input type="button" value="Ping"/>		
General Settings		
Pool Name:	DerbyPool	
Resource Type:	javax.sql.DataSource	
Must be specified if the datasource class implements more than 1 of the interface.		
Datasource Classname:	org.apache.derby.jdbc.ClientDataSource	
Vendor-specific classname that implements the DataSource and/or XADataSource APIs		
Driver Classname:		
Vendor-specific classname that implements the java.sql.Driver interface.		
Ping:	<input type="checkbox"/> Enabled	
When enabled, the pool is pinged during creation or reconfiguration to identify and warn of any erroneous values for its attributes		
Deployment Order:	100	
Specifies the loading order of the resource at server startup. Lower numbers are loaded first.		
Description:		
Pool Settings		
Initial and Minimum Pool Size:	8	Connections
Minimum and initial number of connections maintained in the pool		
Maximum Pool Size:	32	Connections
Maximum number of connections that can be created to satisfy client requests		
Pool Resize Quantity:	2	Connections

- ✓ Click on **Additional Properties**.

General	Advanced	Additional Properties
Edit JDBC Connection Pool Properties Modify properties of an existing JDBC connection pool.		
Pool Name: DerbyPool		
Additional Properties (6)		
Select	Name	Value
<input type="checkbox"/>	PortNumber	1527
<input type="checkbox"/>	Password	APP
<input type="checkbox"/>	User	APP
<input type="checkbox"/>	serverName	localhost
<input type="checkbox"/>	DatabaseName	sun-appserv-samples
<input type="checkbox"/>	connectionAttributes	;create=true

- ✓ Click on **Add Property** and add a **url** property with the following value:

`jdbc:derby://localhost:1527/ShapesDBv1`

- ✓ Delete the **connectionAttributes** property.

General	Advanced	Additional Properties
---------	----------	-----------------------

Edit JDBC Connection Pool Properties

Modify properties of an existing JDBC connection pool.

Pool Name: DerbyPool

Additional Properties (6)

Select	Name	Value
<input type="checkbox"/>	PortNumber	1527
<input type="checkbox"/>	Password	123
<input type="checkbox"/>	User	app
<input type="checkbox"/>	serverName	localhost
<input type="checkbox"/>	DatabaseName	ShapesDBv1
<input type="checkbox"/>	url	jdbc:derby://localhost:1527/ShapesDBv1

- ✓ Save the changes by clicking the **Save** button.

General	Advanced	Additional Properties
---------	----------	-----------------------

✓ New values successfully saved.

Edit JDBC Connection Pool Properties

Modify properties of an existing JDBC connection pool.

Pool Name: DerbyPool

Additional Properties (6)

Select	Name	Value
<input type="checkbox"/>	PortNumber	1527
<input type="checkbox"/>	Password	123
<input type="checkbox"/>	User	app
<input type="checkbox"/>	serverName	localhost
<input type="checkbox"/>	DatabaseName	ShapesDBv1
<input type="checkbox"/>	url	jdbc:derby://localhost:1527/ShapesDBv1

- ✓ Check if the created connection pool is working. Do this by clicking the **Ping** button.

General	Advanced	Additional Properties
----------------	-----------------	------------------------------

Ping Succeeded

Edit JDBC Connection Pool

Modify an existing JDBC connection pool. A JDBC connection pool is a group of reusable connections for a particular database.

[Load Defaults](#) [Flush](#) [Ping](#)

General Settings

Pool Name: DerbyPool

Resource Type: Must be specified if the datasource class implements more than 1 of the interface.

Datasource Classname: Vendor-specific classname that implements the DataSource and/or XDataSource APIs

Driver Classname: Vendor-specific classname that implements the java.sql.Driver interface.

Ping: Enabled When enabled, the pool is pinged during creation or reconfiguration to identify and warn of any erroneous values for its attributes

Deployment Order: Specifies the loading order of the resource at server startup. Lower numbers are loaded first.

Description:

Confirm that the resource points to the DerbyPool.

- ✓ Still under **JDBC**, click on **JDBC Resources**.

JDBC Resources						
JDBC resources provide applications with a means to connect to a database.						
Resources (3)						
Select	JNDI Name	Logical JNDI Name	Enabled	Connection Pool		
<input type="checkbox"/>	jdbc/_TimerPool		✓	TimerPool		
<input type="checkbox"/>	jdbc/_default	java.comp/DefaultDataSource	✓	DerbyPool		
<input type="checkbox"/>	jdbc/sample		✓	SamplePool		

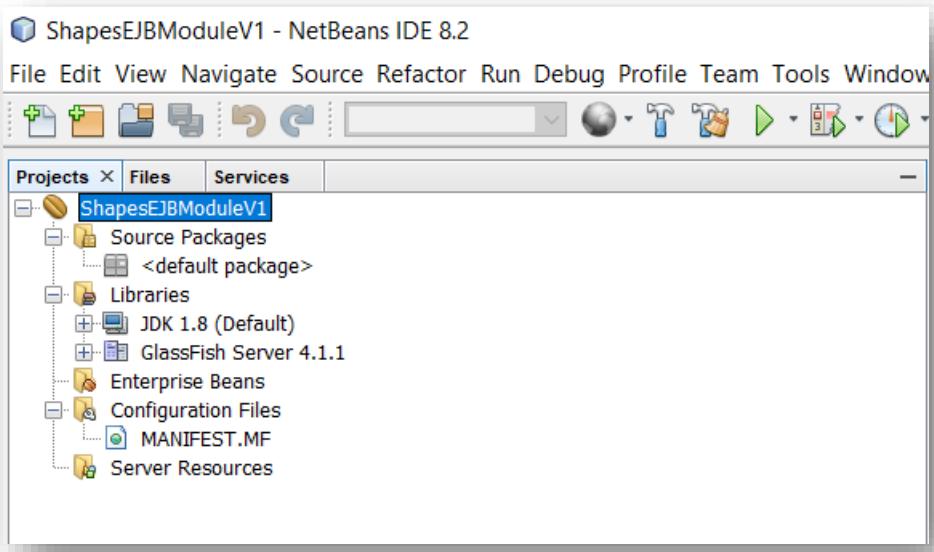
Take note of the purpose of JDBC Resources. It is said:

"JDBC resources provide applications with a means to connect to a database."

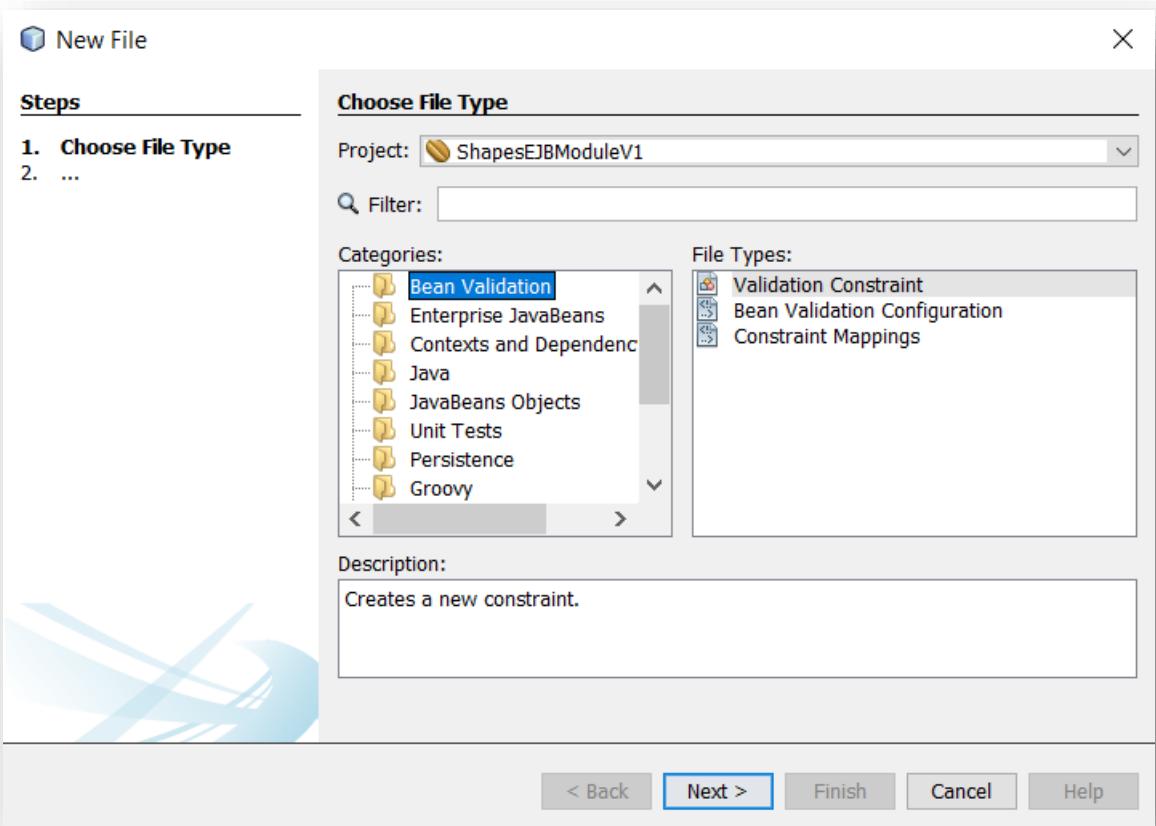
You can also see that the **jdbc/_default** resource is associated with the **DerbyPool** Connection Pool.

Part C – Create an entity.

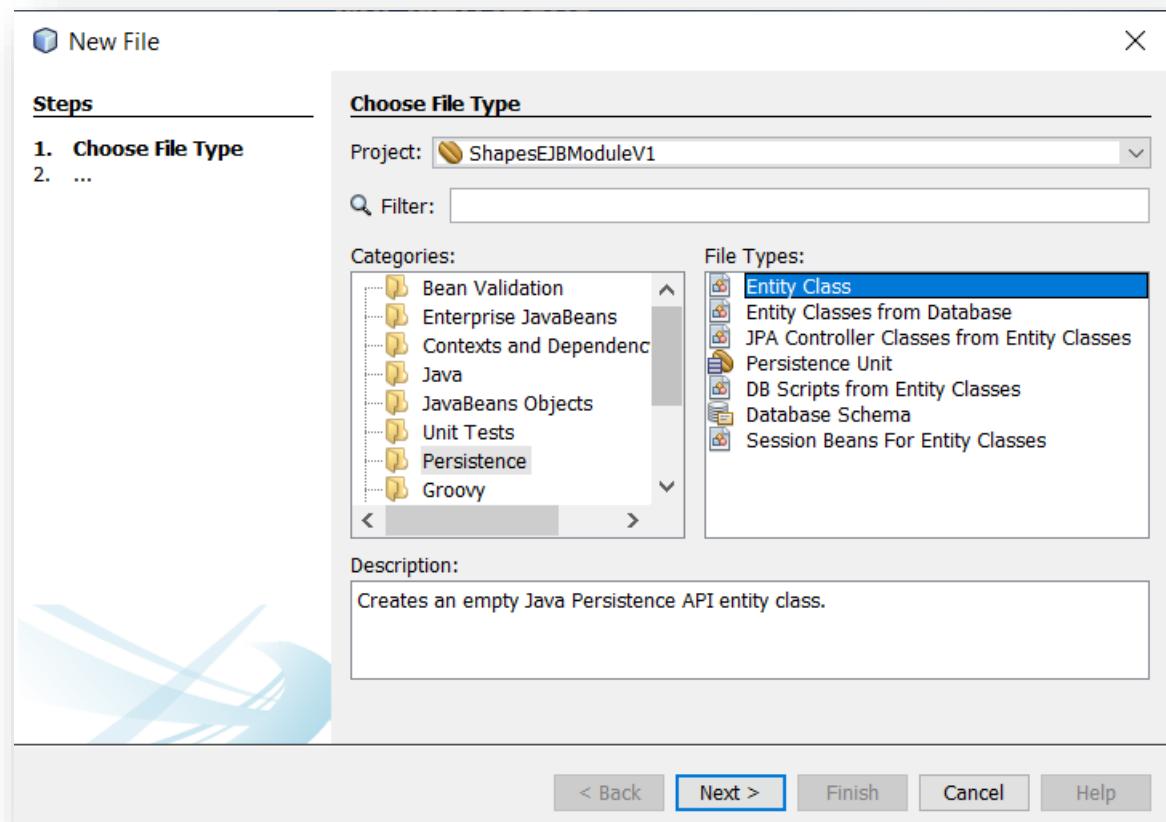
Create an EJB project called **ShapesEJBModuleV1**.



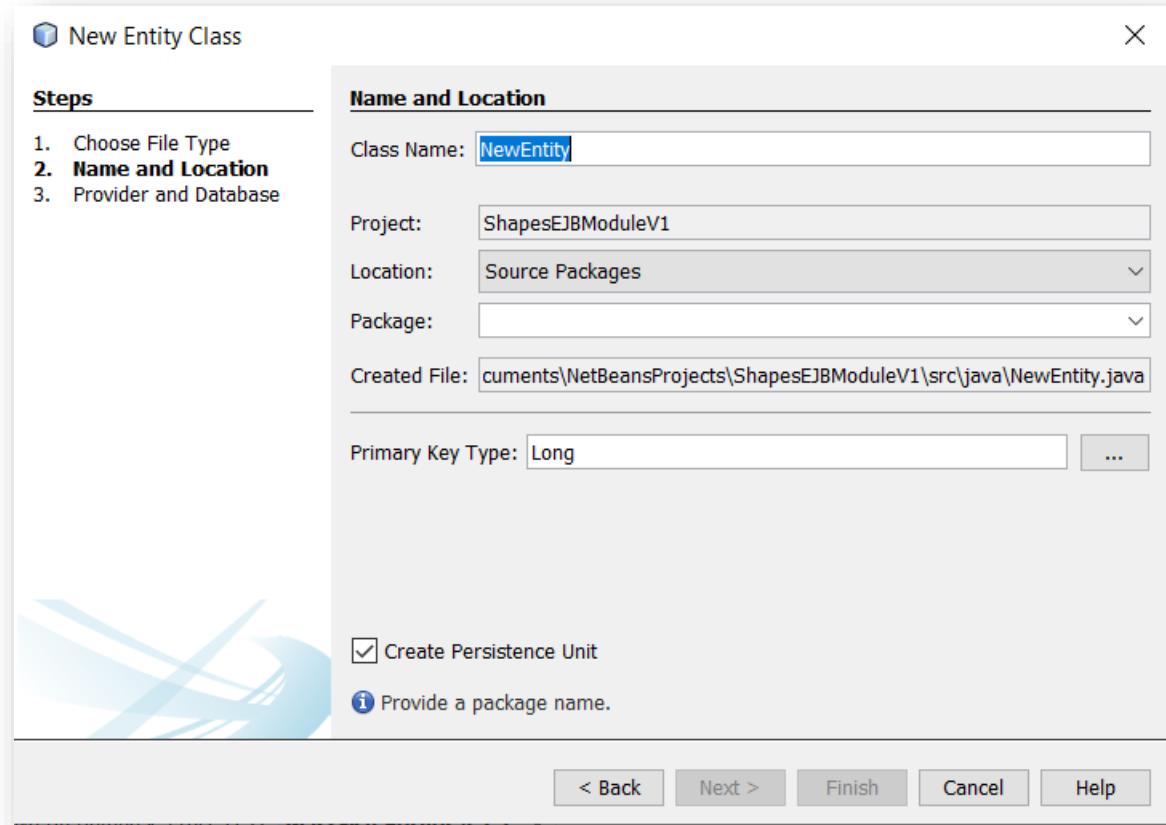
Right-click on the project and select **New | Other**.



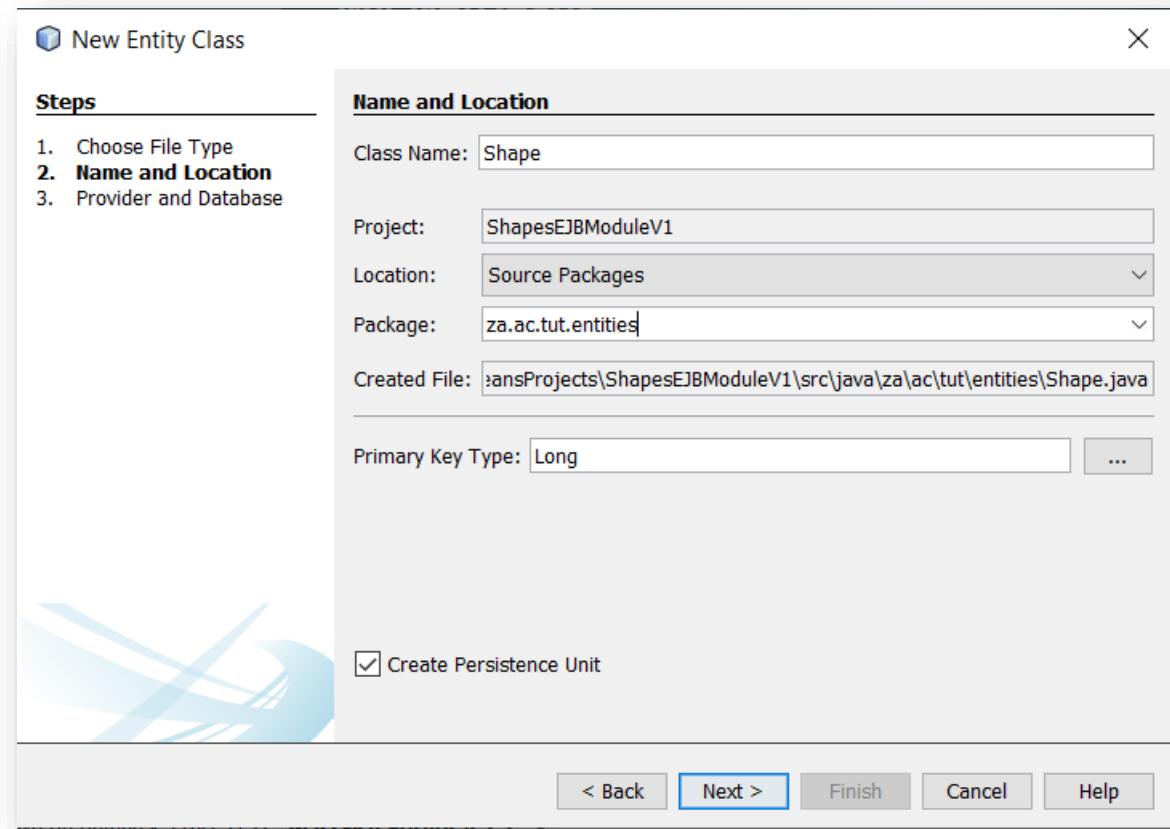
Under **Categories** select **Persistence**, and under **File Types** select **Entity Class**.



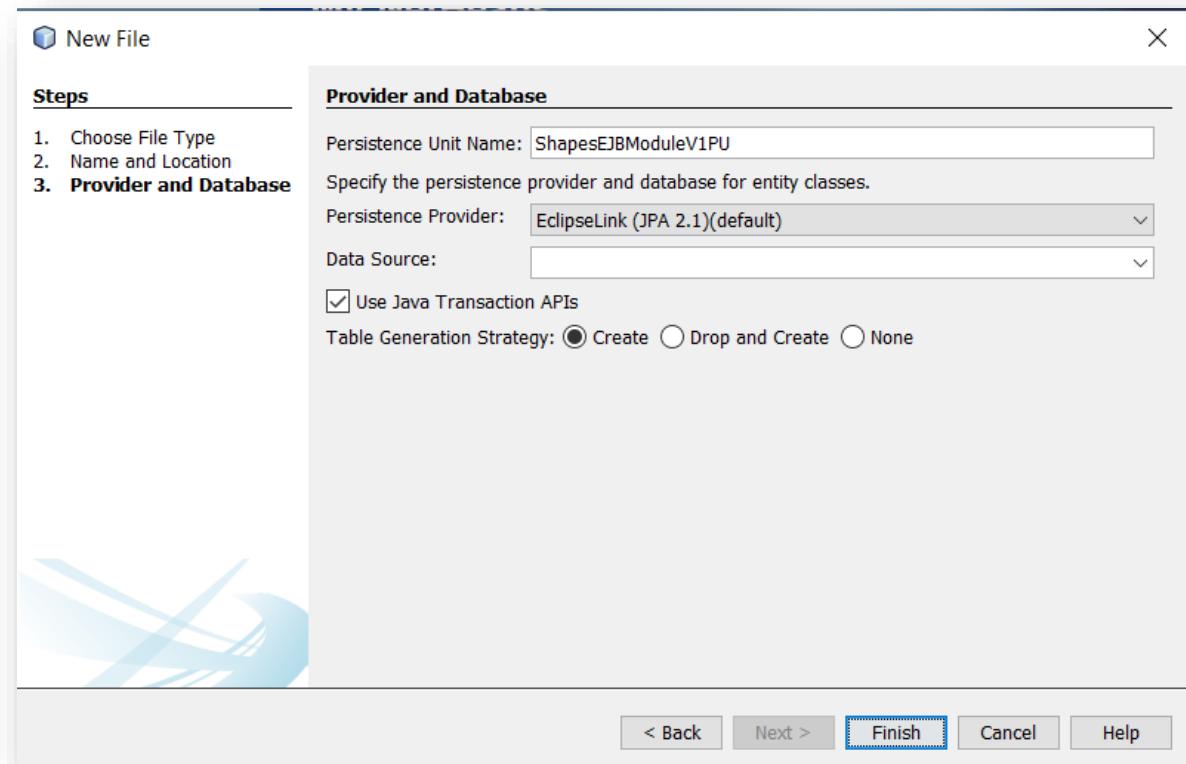
Click **Next**.



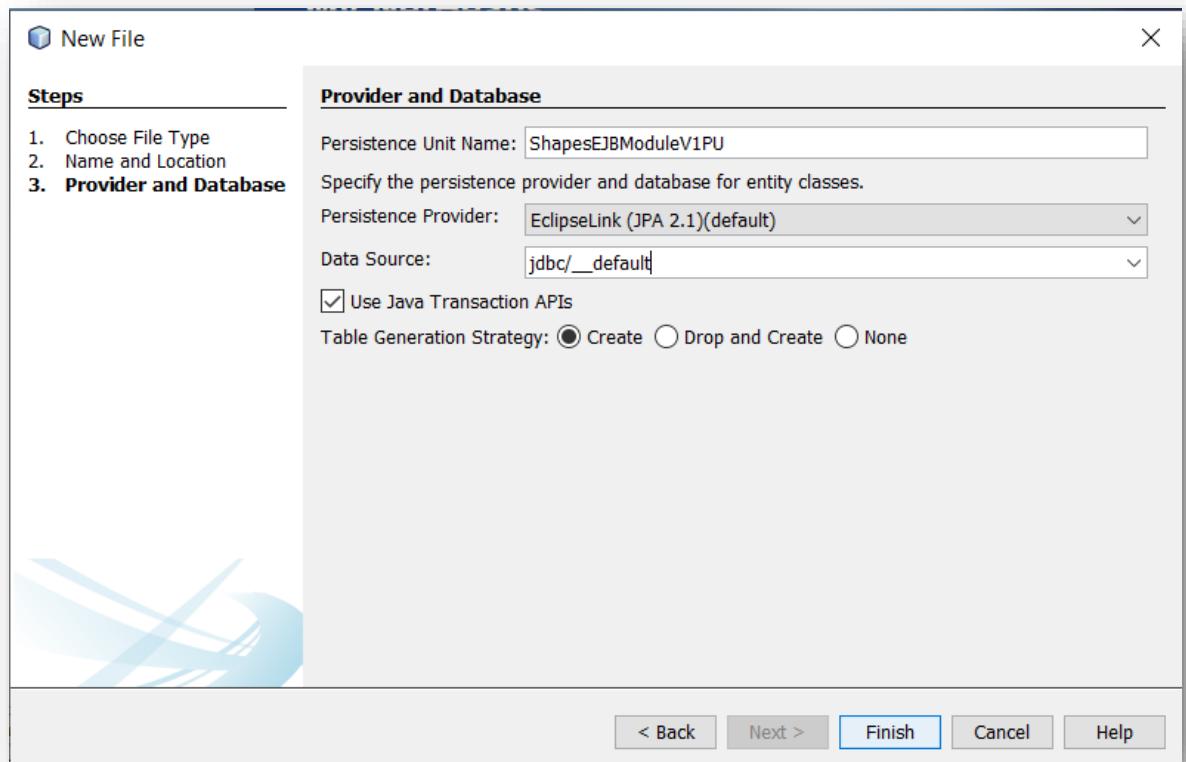
Fill-in the form.



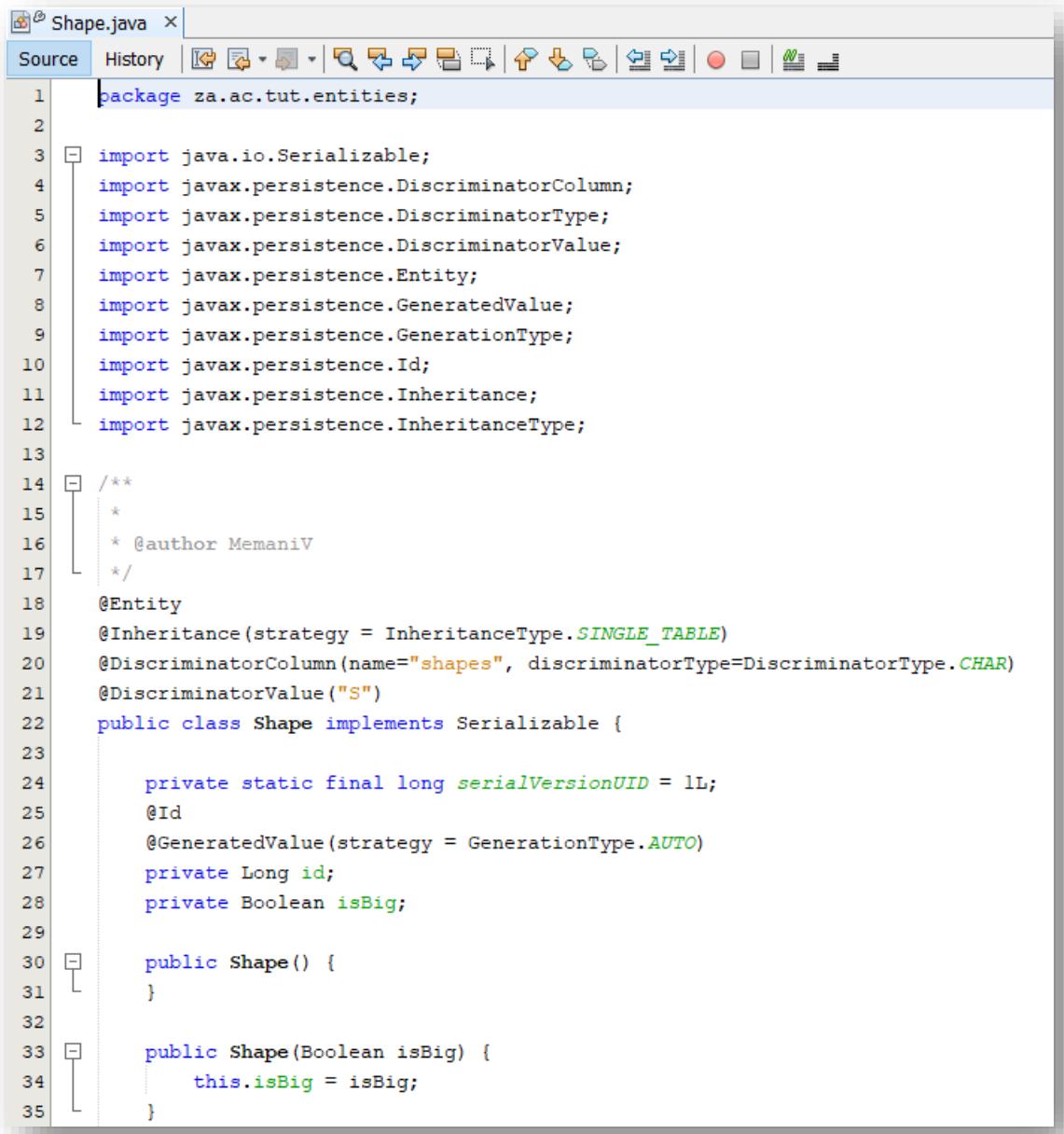
Click **Next**.



Under **Data Source**, select **jdbc/_default**.



Click **Finish**.



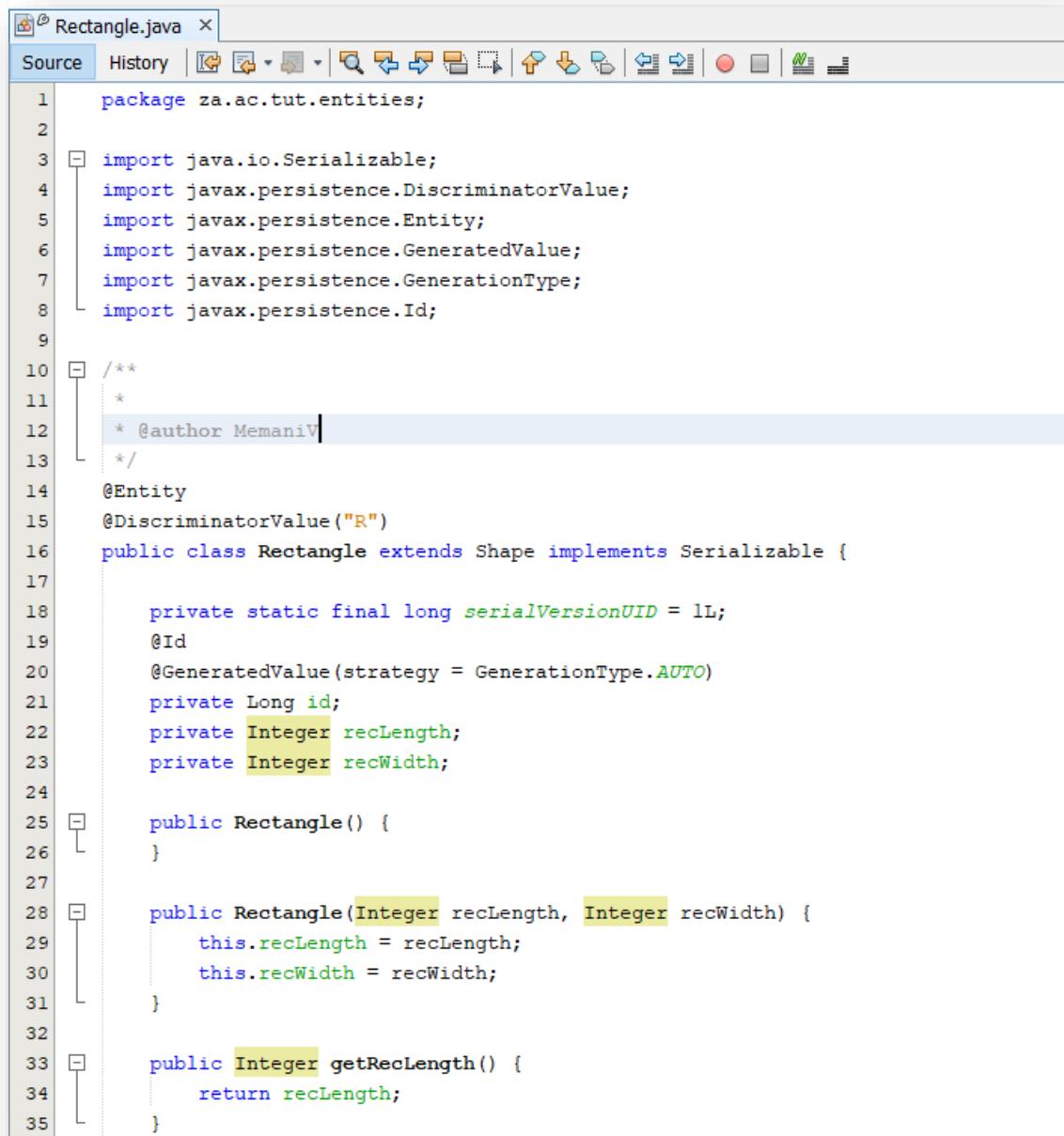
The screenshot shows a Java code editor window with the tab "Shape.java" selected. The code implements the Serializable interface and uses annotations from javax.persistence to define a single-table inheritance strategy for shapes. The class has a private constructor and a constructor taking a Boolean parameter.

```
1 package za.ac.tut.entities;
2
3 import java.io.Serializable;
4 import javax.persistence.DiscriminatorColumn;
5 import javax.persistence.DiscriminatorType;
6 import javax.persistence.DiscriminatorValue;
7 import javax.persistence.Entity;
8 import javax.persistence.GeneratedValue;
9 import javax.persistence.GenerationType;
10 import javax.persistence.Id;
11 import javax.persistence.Inheritance;
12 import javax.persistence.InheritanceType;
13
14 /**
15 *
16 * @author MemaniV
17 */
18 @Entity
19 @Inheritance(strategy = InheritanceType.SINGLE_TABLE)
20 @DiscriminatorColumn(name="shapes", discriminatorType=DiscriminatorType.CHAR)
21 @DiscriminatorValue("S")
22 public class Shape implements Serializable {
23
24     private static final long serialVersionUID = 1L;
25     @Id
26     @GeneratedValue(strategy = GenerationType.AUTO)
27     private Long id;
28     private Boolean isBig;
29
30     public Shape() {
31     }
32
33     public Shape(Boolean isBig) {
34         this.isBig = isBig;
35     }
}
```

```
37 |     public Boolean getIsBig() {
38 |         return isBig;
39 |     }
40 |
41 |     public void setIsBig(Boolean isBig) {
42 |         this.isBig = isBig;
43 |     }
44 |
45 |     public Long getId() {
46 |         return id;
47 |     }
48 |
49 |     public void setId(Long id) {
50 |         this.id = id;
51 |     }
52 |
53 |     @Override
54 |     public int hashCode() {
55 |         int hash = 0;
56 |         hash += (id != null ? id.hashCode() : 0);
57 |         return hash;
58 |     }
```

```
60 |     @Override
61 |     public boolean equals(Object object) {
62 |         if (!(object instanceof Shape)) {
63 |             return false;
64 |         }
65 |         Shape other = (Shape) object;
66 |         if ((this.id == null && other.id != null) ||
67 |             (this.id != null && !this.id.equals(other.id))) {
68 |             return false;
69 |         }
70 |         return true;
71 |     }
72 |
73 |     @Override
74 |     public String toString() {
75 |         return "za.ac.tut.entities.Shape[ id=" + id + " ]";
76 |     }
77 |
78 | }
```

Create a **Rectangle** sub entity.



The screenshot shows a Java code editor window with the file `Rectangle.java` open. The code defines a persistent entity named `Rectangle` that extends `Shape` and implements `Serializable`. It includes fields for `id`, `recLength`, and `recWidth`, and methods for their respective getters and setters. The code uses annotations from the `javax.persistence` package, such as `@Entity`, `@DiscriminatorValue`, `@Id`, and `@GeneratedValue`. The `recLength` and `recWidth` fields are highlighted in yellow, indicating they are selected or being edited.

```
1 package za.ac.tut.entities;
2
3 import java.io.Serializable;
4 import javax.persistence.DiscriminatorValue;
5 import javax.persistence.Entity;
6 import javax.persistence.GeneratedValue;
7 import javax.persistence.GenerationType;
8 import javax.persistence.Id;
9
10 /**
11 *
12 * @author MemaniV
13 */
14 @Entity
15 @DiscriminatorValue("R")
16 public class Rectangle extends Shape implements Serializable {
17
18     private static final long serialVersionUID = 1L;
19     @Id
20     @GeneratedValue(strategy = GenerationType.AUTO)
21     private Long id;
22     private Integer recLength;
23     private Integer recWidth;
24
25     public Rectangle() {
26     }
27
28     public Rectangle(Integer recLength, Integer recWidth) {
29         this.recLength = recLength;
30         this.recWidth = recWidth;
31     }
32
33     public Integer getRecLength() {
34         return recLength;
35     }
}
```

```
37  public void setRecLength(Integer recLength) {
38      this.recLength = recLength;
39  }
40
41  public Integer getRecWidth() {
42      return recWidth;
43  }
44
45  public void setRecWidth(Integer recWidth) {
46      this.recWidth = recWidth;
47  }
48
49  @Override
50  public Long getId() {
51      return id;
52  }
53
54  @Override
55  public void setId(Long id) {
56      this.id = id;
57  }
58
59  @Override
60  public int hashCode() {
61      int hash = 0;
62      hash += (id != null ? id.hashCode() : 0);
63      return hash;
64  }
```

```
66  @Override
67  public boolean equals(Object object) {
68      if (!(object instanceof Rectangle)) {
69          return false;
70      }
71      Rectangle other = (Rectangle) object;
72      if ((this.id == null && other.id != null) ||
73          (this.id != null && !this.id.equals(other.id))) {
74          return false;
75      }
76      return true;
77  }
78
79  @Override
80  public String toString() {
81      return "za.ac.tut.entities.Rectangle[ id=" + id + " ]";
82  }
83
84 }
```

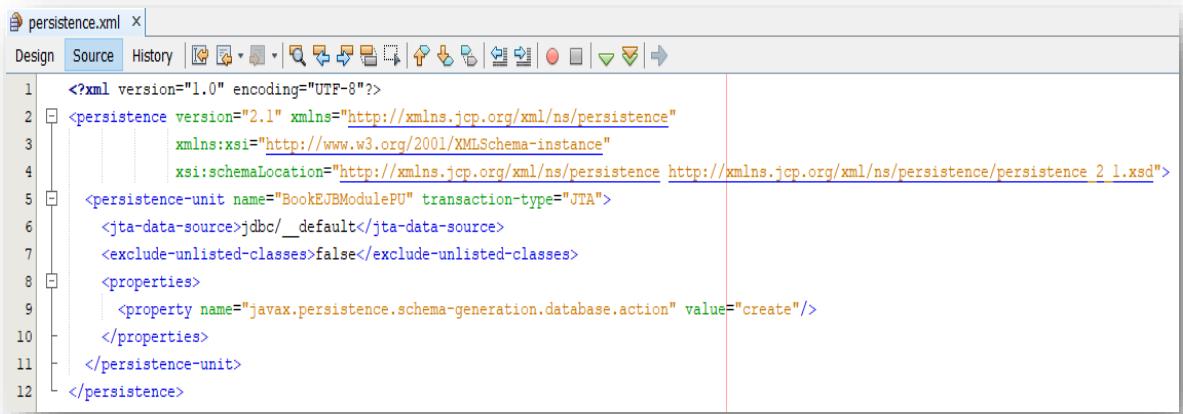
Create a **Parallelogram** sub entity.

```
1 package za.ac.tut.entities;
2
3 import java.io.Serializable;
4 import javax.persistence.DiscriminatorValue;
5 import javax.persistence.Entity;
6 import javax.persistence.GeneratedValue;
7 import javax.persistence.GenerationType;
8 import javax.persistence.Id;
9
10 /**
11 *
12 * @author MemaniV
13 */
14 @Entity
15 @DiscriminatorValue("P")
16 public class Parallelogram extends Shape implements Serializable {
17
18     private static final long serialVersionUID = 1L;
19     @Id
20     @GeneratedValue(strategy = GenerationType.AUTO)
21     private Long id;
22     private Integer base;
23     private Integer height;
24
25     public Parallelogram() {
26     }
27
28     public Parallelogram(Integer base, Integer height) {
29         this.base = base;
30         this.height = height;
31     }
32
33     public Integer getBase() {
34         return base;
35     }
```

```
37 |     public void setBase(Integer base) {
38 |         this.base = base;
39 |
40 |
41 |     public Integer getHeight() {
42 |         return height;
43 |     }
44 |
45 |     public void setHeight(Integer height) {
46 |         this.height = height;
47 |     }
48 |
49 |
50 |     public Long getId() {
51 |         return id;
52 |     }
53 |
54 |     @Override
55 |     public void setId(Long id) {
56 |         this.id = id;
57 |     }
58 |
59 |     @Override
60 |     public int hashCode() {
61 |         int hash = 0;
62 |         hash += (id != null ? id.hashCode() : 0);
63 |         return hash;
64 |     }
```

```
66 |     @Override
67 |     public boolean equals(Object object) {
68 |         if (!(object instanceof Parallelogram)) {
69 |             return false;
70 |         }
71 |         Parallelogram other = (Parallelogram) object;
72 |         if ((this.id == null &amp; other.id != null) ||
73 |             (this.id != null &amp; !this.id.equals(other.id))) {
74 |             return false;
75 |         }
76 |         return true;
77 |     }
78 |
79 |     @Override
80 |     public String toString() {
81 |         return "za.ac.tut.entities.Parallelogram[ id=" + id + " ]";
82 |     }
83 |
84 | }
```

View the **persistence.xml** file.

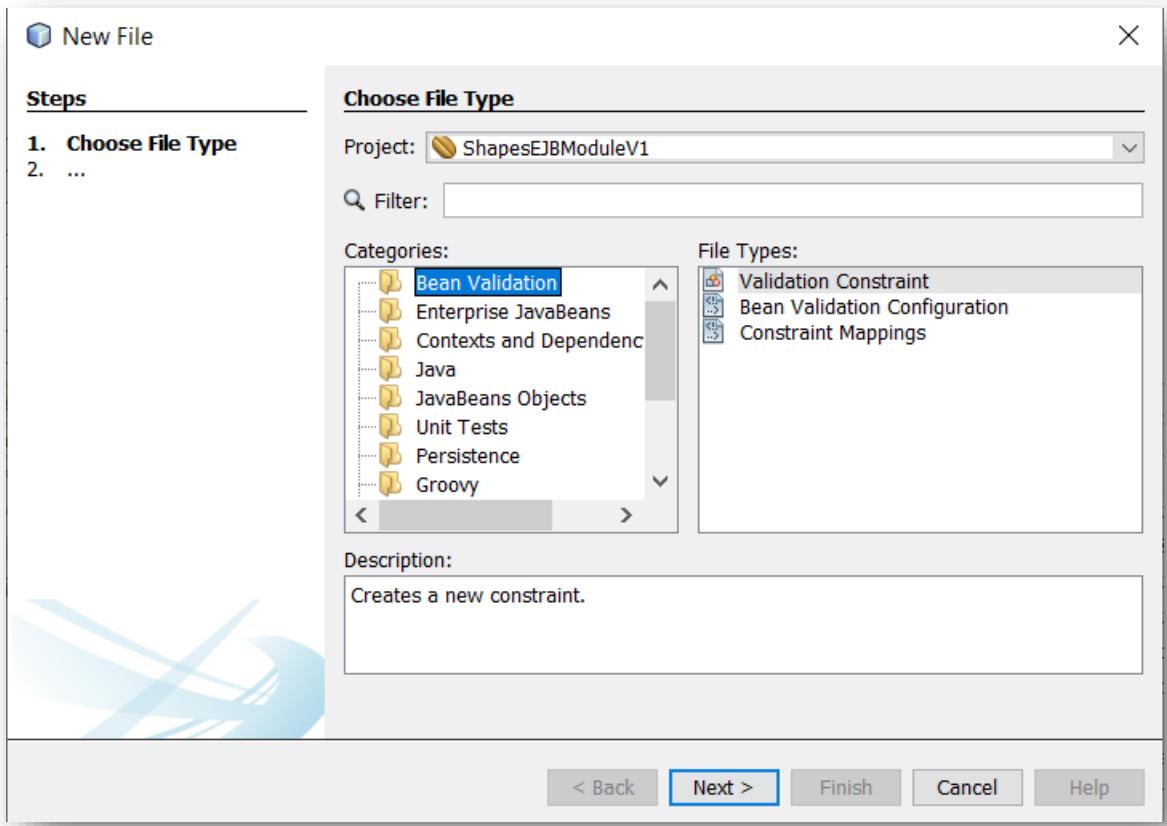


The screenshot shows the Eclipse IDE interface with the file "persistence.xml" open. The tab bar at the top has "Design", "Source" (which is selected), and "History". Below the tabs is a toolbar with various icons. The code editor displays the XML configuration for a persistence unit:

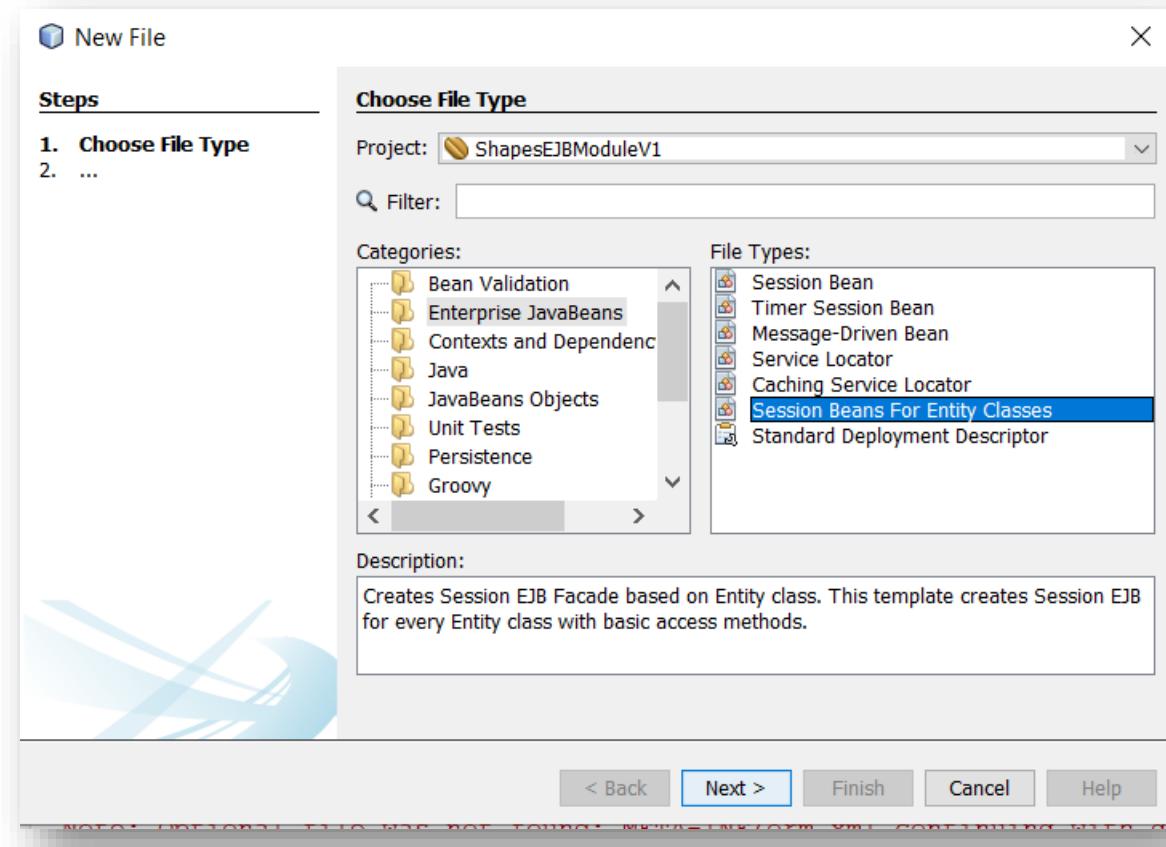
```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
    <persistence-unit name="BookEJBModulePU" transaction-type="JTA">
        <jta-data-source>jdbc/_default</jta-data-source>
        <exclude-unlisted-classes>false</exclude-unlisted-classes>
        <properties>
            <property name="javax.persistence.schema-generation.database.action" value="create"/>
        </properties>
    </persistence-unit>
</persistence>
```

Create business code (CRUD operations) for the entity. Perform the following steps:

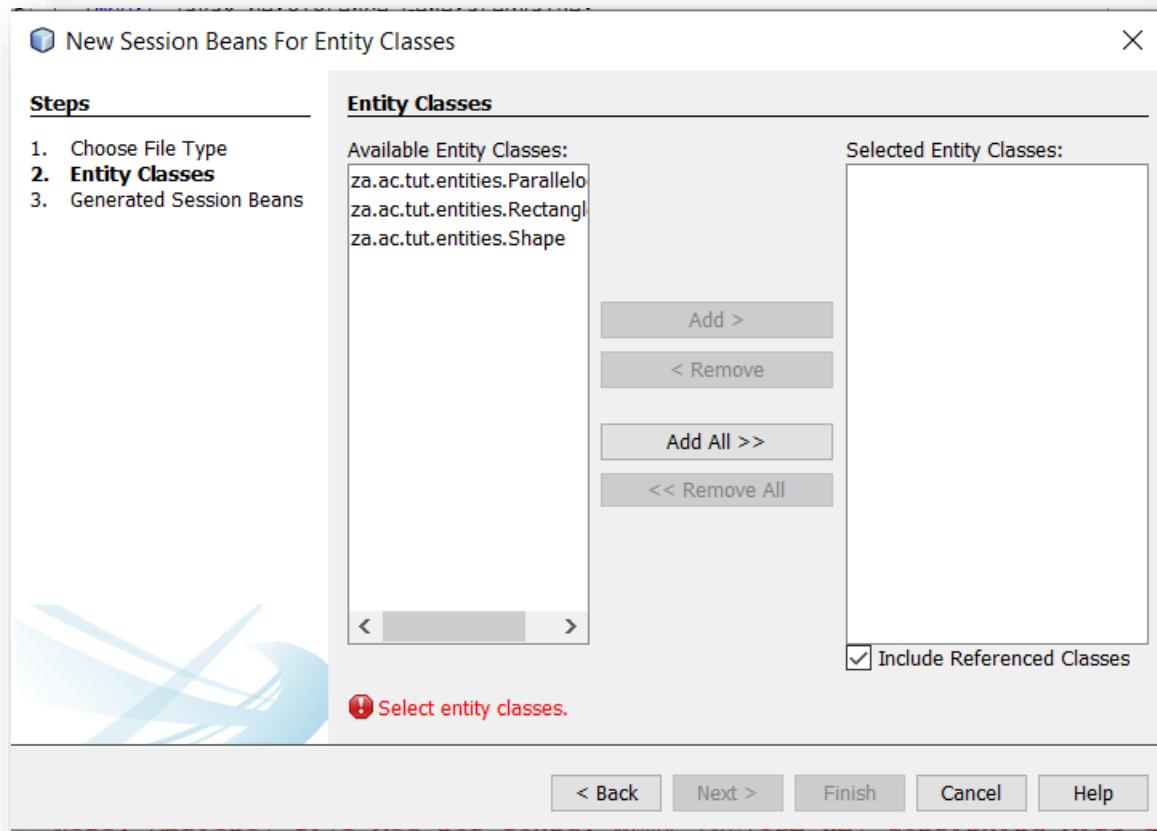
- ✓ Right-click on the project and select **New | Other**.



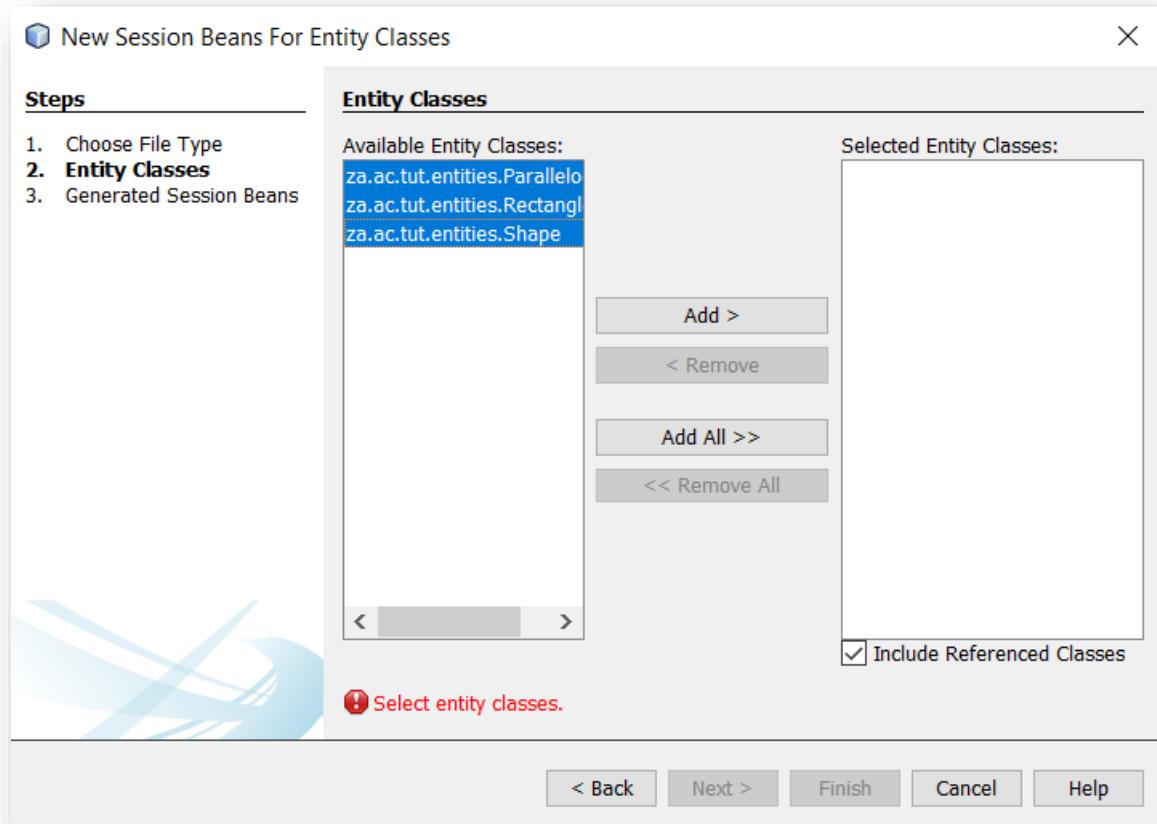
- ✓ Under **Categories**, select **Enterprise JavaBeans**. Under **File Types** select **Session Beans for Entity Classes**.



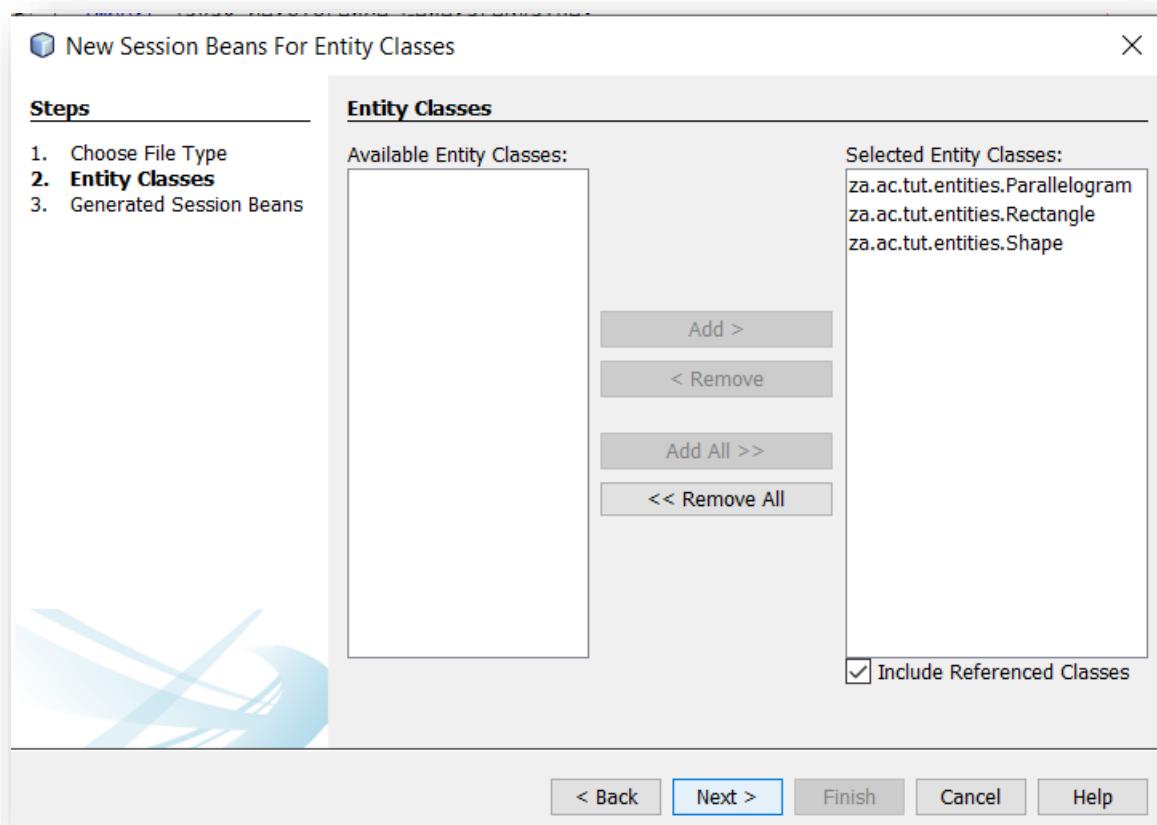
- ✓ Click **Next**.



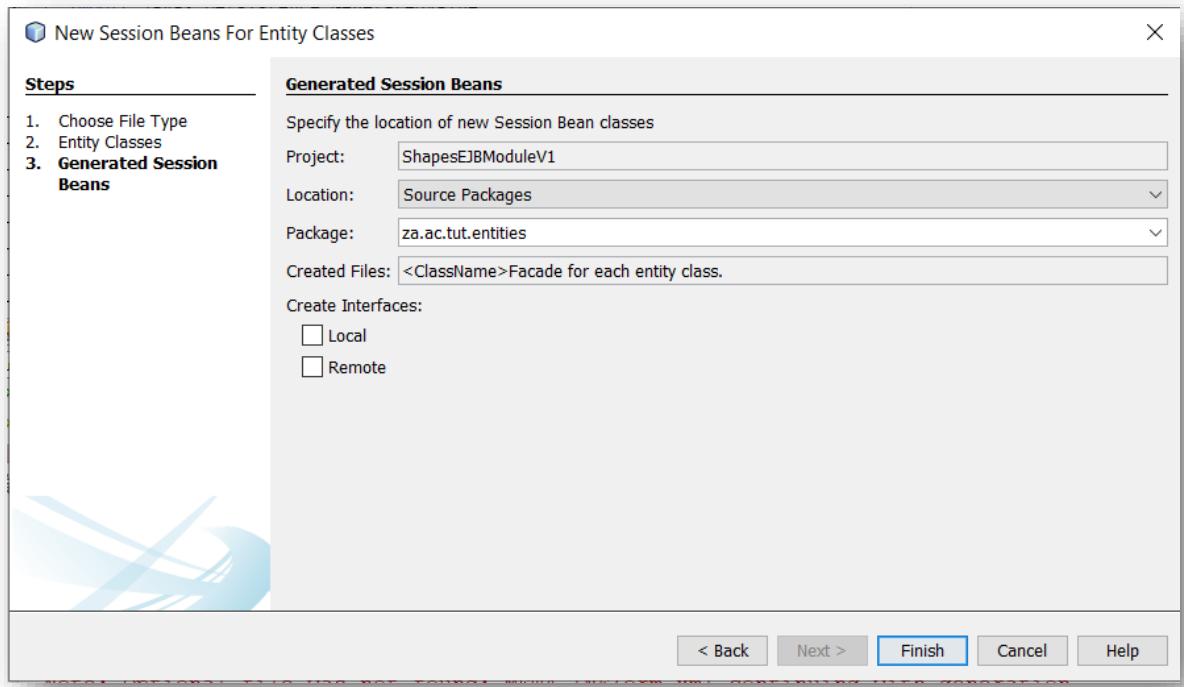
- ✓ Select **all** the entities under Available Entity Classes.



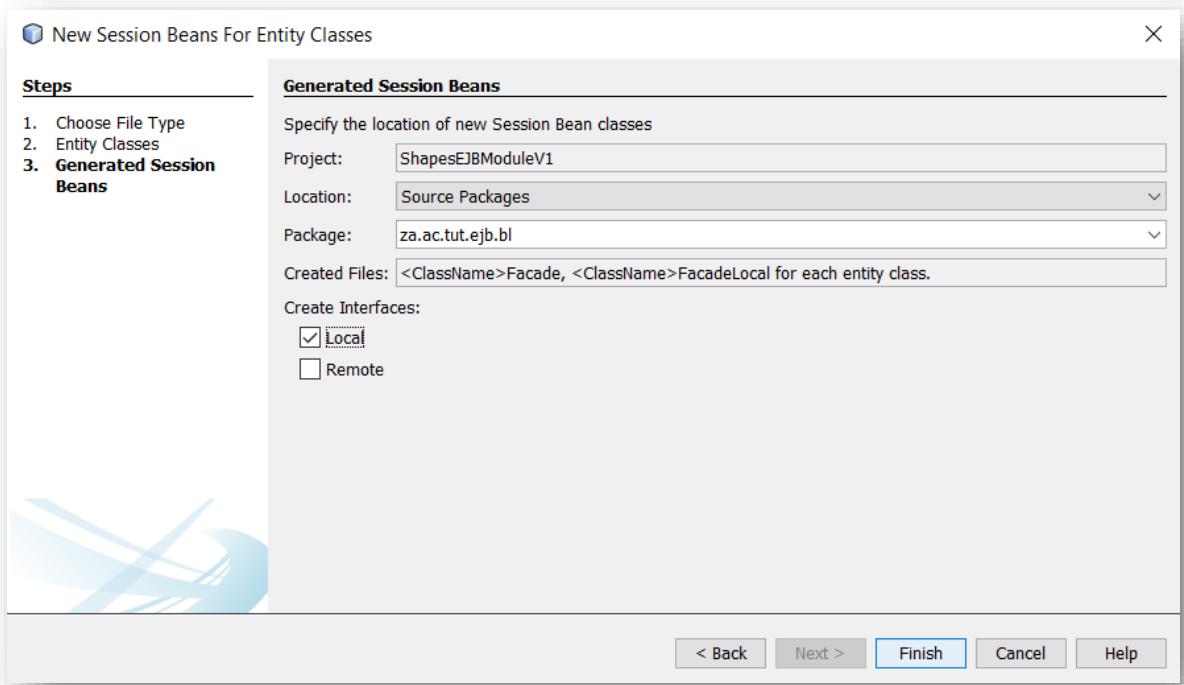
- ✓ Click Add.



✓ Click **Next**.



✓ Modify the package name to **za.ac.tut.ejb.bl**, where **bl** stands for **business logic**. Also select **Local** interface.



- ✓ Click **Finish**. View created files.
- **ParallelogramFacade**.

```

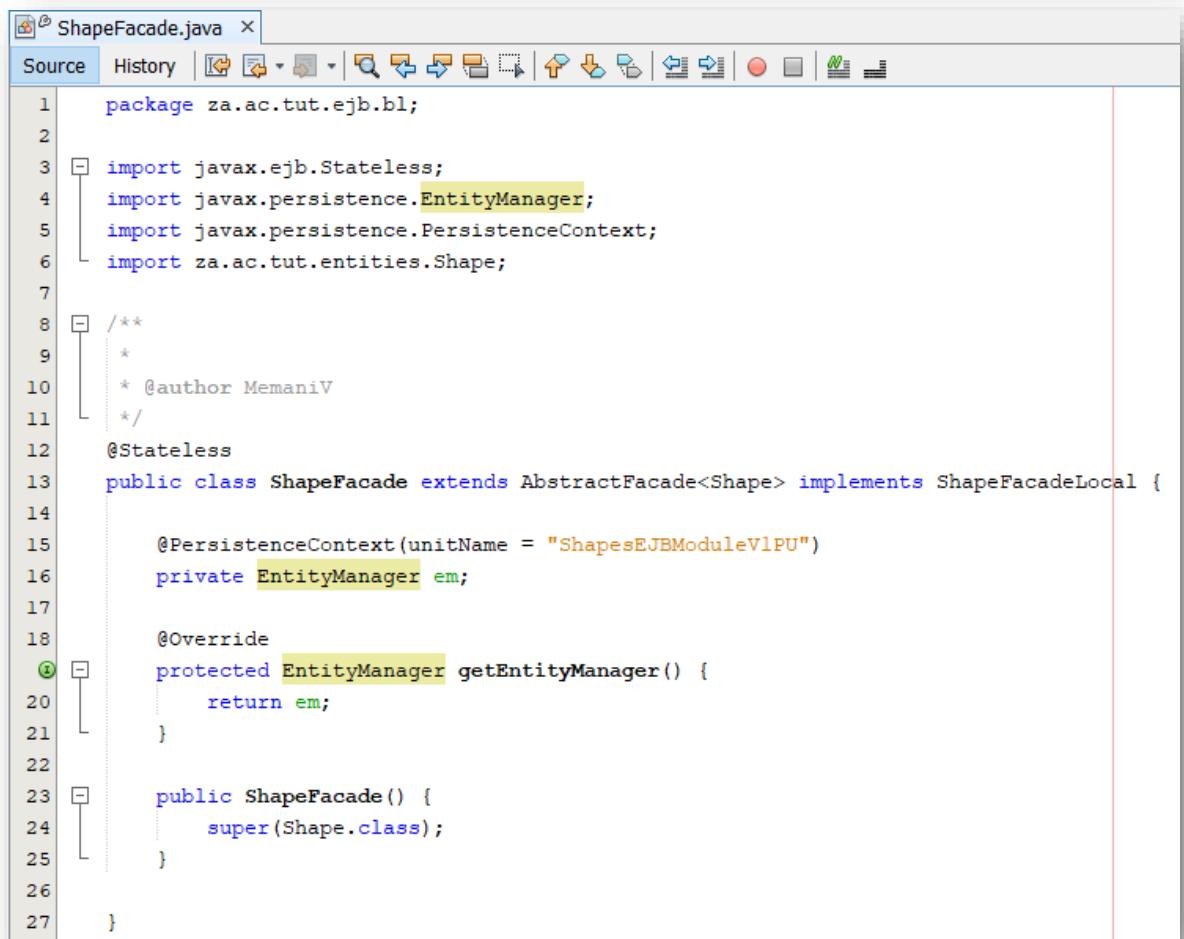
1 package za.ac.tut.ejb.bl;
2
3 import javax.ejb.Stateless;
4 import javax.persistence.EntityManager;
5 import javax.persistence.PersistenceContext;
6 import za.ac.tut.entities.Parallelogram;
7
8 /**
9 *
10 * @author MemaniV
11 */
12 @Stateless
13 public class ParallelogramFacade extends AbstractFacade<Parallelogram> implements ParallelogramFacadeLocal {
14
15     @PersistenceContext(unitName = "ShapesEJBModuleV1PU")
16     private EntityManager em;
17
18     @Override
19     protected EntityManager getEntityManager() {
20         return em;
21     }
22
23     public ParallelogramFacade() {
24         super(Parallelogram.class);
25     }
26 }
```

- **RectangleFacade**

```

1 package za.ac.tut.ejb.bl;
2
3 import javax.ejb.Stateless;
4 import javax.persistence.EntityManager;
5 import javax.persistence.PersistenceContext;
6 import za.ac.tut.entities.Rectangle;
7
8 /**
9 *
10 * @author MemaniV
11 */
12 @Stateless
13 public class RectangleFacade extends AbstractFacade<Rectangle> implements RectangleFacadeLocal {
14
15     @PersistenceContext(unitName = "ShapesEJBModuleV1PU")
16     private EntityManager em;
17
18     @Override
19     protected EntityManager getEntityManager() {
20         return em;
21     }
22
23     public RectangleFacade() {
24         super(Rectangle.class);
25     }
26 }
```

▪ ShapeFacade



The screenshot shows a Java code editor window titled "ShapeFacade.java". The code implements the ShapeFacade interface, which extends AbstractFacade<Shape> and implements ShapeFacadeLocal. It uses annotations like @Stateless, @PersistenceContext, and @Override. The EntityManager is injected via a private field and a protected getter method. The constructor calls super(Shape.class). The code is part of the za.ac.tut.ejb.bl package.

```
1 package za.ac.tut.ejb.bl;
2
3 import javax.ejb.Stateless;
4 import javax.persistence.EntityManager;
5 import javax.persistence.PersistenceContext;
6 import za.ac.tut.entities.Shape;
7
8 /**
9  * @author MemaniV
10 */
11 @Stateless
12 public class ShapeFacade extends AbstractFacade<Shape> implements ShapeFacadeLocal {
13
14     @PersistenceContext(unitName = "ShapesEJBModuleV1PU")
15     private EntityManager em;
16
17
18     @Override
19     protected EntityManager getEntityManager() {
20         return em;
21     }
22
23     public ShapeFacade() {
24         super(Shape.class);
25     }
26
27 }
```

▪ AbstractFacade.

AbstractFacade.java

```

1 package za.ac.tut.bl;
2
3 import java.util.List;
4 import javax.persistence.EntityManager;
5
6 /**
7  * @author MemaniV
8 */
9
10 public abstract class AbstractFacade<T> {
11
12     private Class<T> entityClass;
13
14     public AbstractFacade(Class<T> entityClass) {
15         this.entityClass = entityClass;
16     }
17
18     protected abstract EntityManager getEntityManager();
19
20     public void create(T entity) {
21         getEntityManager().persist(entity);
22     }
23
24     public void edit(T entity) {
25         getEntityManager().merge(entity);
26     }
27
28     public void remove(T entity) {
29         getEntityManager().remove(getEntityManager().merge(entity));
30     }
31

```

```

32     public T find(Object id) {
33         return getEntityManager().find(entityClass, id);
34     }
35
36     public List<T> findAll() {
37         javax.persistence.criteria.CriteriaQuery cq =
38             getEntityManager().getCriteriaBuilder().createQuery();
39         cq.select(cq.from(entityClass));
40         return getEntityManager().createQuery(cq).getResultList();
41     }
42
43     public List<T> findRange(int[] range) {
44         javax.persistence.criteria.CriteriaQuery cq =
45             getEntityManager().getCriteriaBuilder().createQuery();
46         cq.select(cq.from(entityClass));
47         javax.persistence.Query q = getEntityManager().createQuery(cq);
48         q.setMaxResults(range[1] - range[0] + 1);
49         q.setFirstResult(range[0]);
50         return q.getResultList();
51     }
52
53     public int count() {
54         javax.persistence.criteria.CriteriaQuery cq =
55             getEntityManager().getCriteriaBuilder().createQuery();
56         javax.persistence.criteria.Root<T> rt = cq.from(entityClass);
57         cq.select(getEntityManager().getCriteriaBuilder().count(rt));
58         javax.persistence.Query q = getEntityManager().createQuery(cq);
59         return ((Long) q.getSingleResult()).intValue();
60     }
61
62 }

```

▪ ParallelogramFacadeLocal

The screenshot shows a Java code editor window with the title "ParallelogramFacadeLocal.java". The code is as follows:

```
1 package za.ac.tut.ejb.bl;
2
3 import java.util.List;
4 import javax.ejb.Local;
5 import za.ac.tut.entities.Parallelogram;
6
7 /**
8 * @author MemaniV
9 */
10 @Local
11 public interface ParallelogramFacadeLocal {
12     void create(Parallelogram parallelogram);
13     void edit(Parallelogram parallelogram);
14     void remove(Parallelogram parallelogram);
15     Parallelogram find(Object id);
16     List<Parallelogram> findAll();
17     List<Parallelogram> findRange(int[] range);
18     int count();
19 }
20
21
22
23
24
25
26
27
28 }
```

▪ RectangleFacadeLocal.

The screenshot shows a Java code editor window with the title "RectangleFacadeLocal.java". The code is as follows:

```
1 package za.ac.tut.ejb.bl;
2
3 import java.util.List;
4 import javax.ejb.Local;
5 import za.ac.tut.entities.Rectangle;
6
7 /**
8 * @author MemaniV
9 */
10 @Local
11 public interface RectangleFacadeLocal {
12     void create(Rectangle rectangle);
13     void edit(Rectangle rectangle);
14     void remove(Rectangle rectangle);
15     Rectangle find(Object id);
16     List<Rectangle> findAll();
17     List<Rectangle> findRange(int[] range);
18     int count();
19 }
20
21
22
23
24
25
26
27
28 }
```

- **ShapeFacadeLocal.**

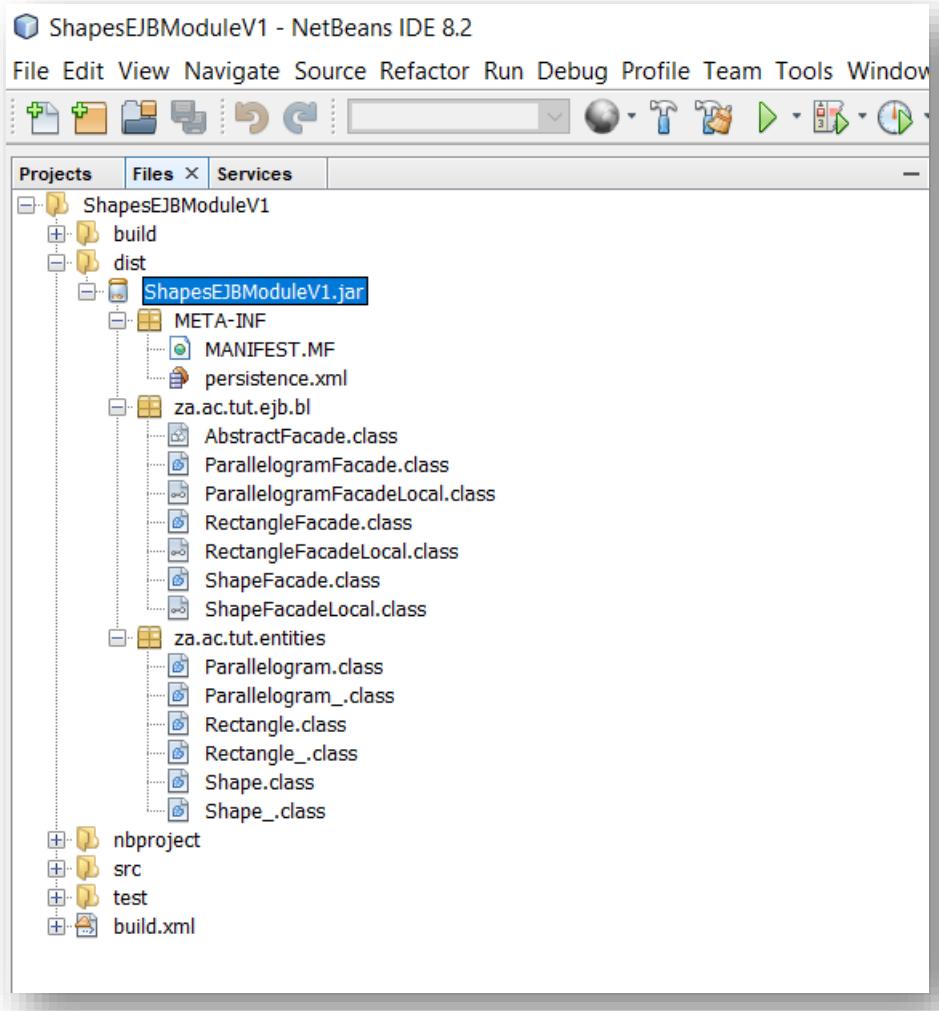
The screenshot shows the NetBeans IDE interface with the code editor open for the file `ShapeFacadeLocal.java`. The code defines a public interface `ShapeFacadeLocal` with methods for creating, editing, removing, finding, and finding all shapes. It includes imports for `java.util.List`, `javax.ejb.Local`, and `za.ac.tut.entities.Shape`. A Javadoc comment is present at the top, and the interface is annotated with `@Local`.

```
1 package za.ac.tut.ejb.bl;
2
3 import java.util.List;
4 import javax.ejb.Local;
5 import za.ac.tut.entities.Shape;
6
7 /**
8 *
9 * @author MemaniV
10 */
11 @Local
12 public interface ShapeFacadeLocal {
13
14     void create(Shape shape);
15
16     void edit(Shape shape);
17
18     void remove(Shape shape);
19
20     Shape find(Object id);
21
22     List<Shape> findAll();
23
24     List<Shape> findRange(int[] range);
25
26     int count();
27 }
28 }
```

- ✓ Clean and build the EJB project.

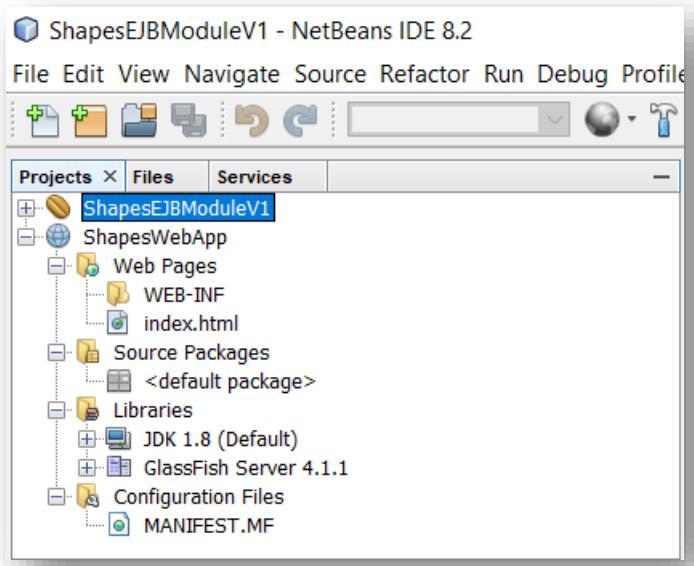
The screenshot shows the NetBeans IDE output window for the GlassFish Server 4.1.1 build process. The logs indicate the creation of static metadata factories, optional file generation, and the successful compilation and distribution of the EJB project. The final message shows a build time of 2 seconds.

```
Note: Creating static metadata factory ...
Note: Optional file was not found: META-INF/orm.xml continuing with generation.
Note: Optional file was not found: META-INF/eclipselink-orm.xml continuing with generation.
Note: Optional file was not found: META-INF/orm.xml continuing with generation.
Note: Optional file was not found: META-INF/eclipselink-orm.xml continuing with generation.
Note: C:\Users\memaniv\Documents\NetBeansProjects\ShapesEJBMo...
Note: Recompile with -Xlint:unchecked for details.
1 warning
compile:
library-inclusion-in-archive:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\ShapesEJBMo...
Building jar: C:\Users\memaniv\Documents\NetBeansProjects\ShapesEJBMo...
dist:
BUILD SUCCESSFUL (total time: 2 seconds)
```



Part D – Create a web client project.

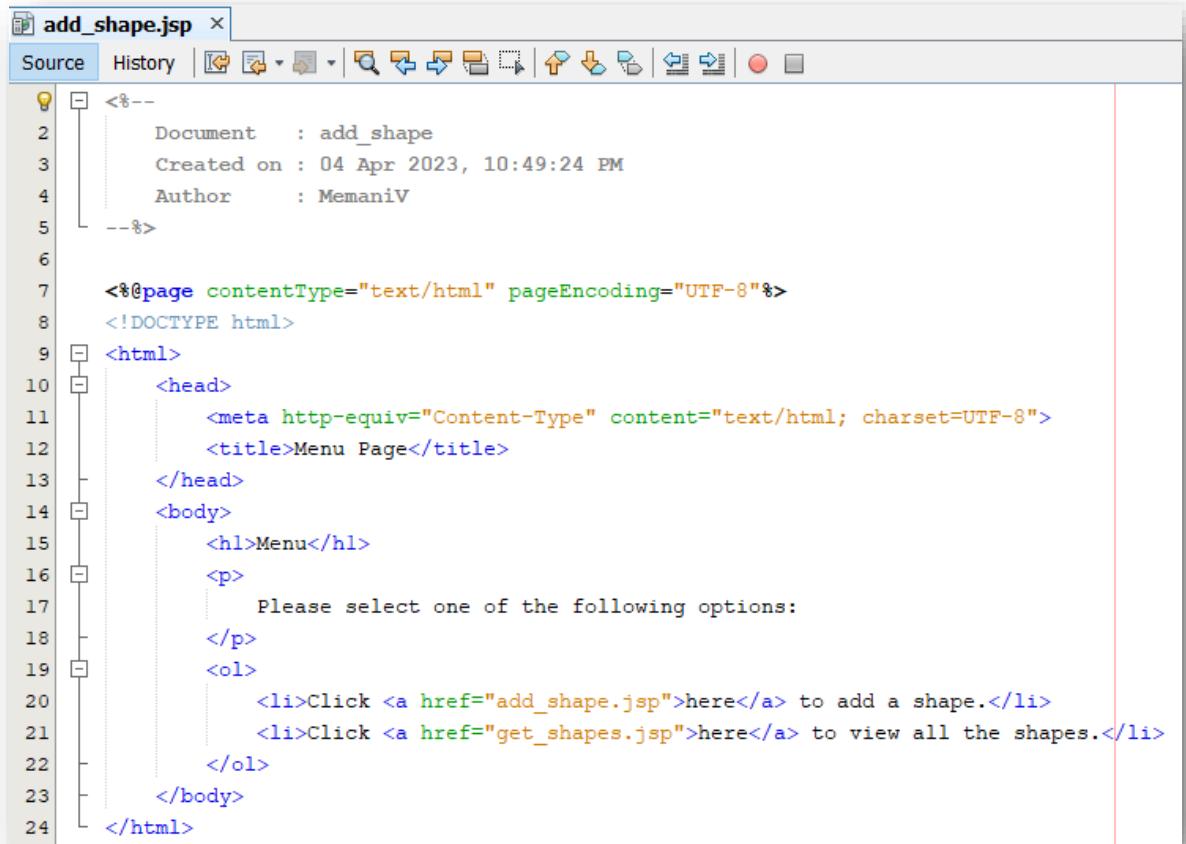
Create a web client project called **BookWebApp**.



Edit the **index.html** page.

```
<!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 page</h1>
        <p>
            Welcome to our web app. Please click <a href="menu.jsp">here</a> to start.
        </p>
    </body>
</html>
```

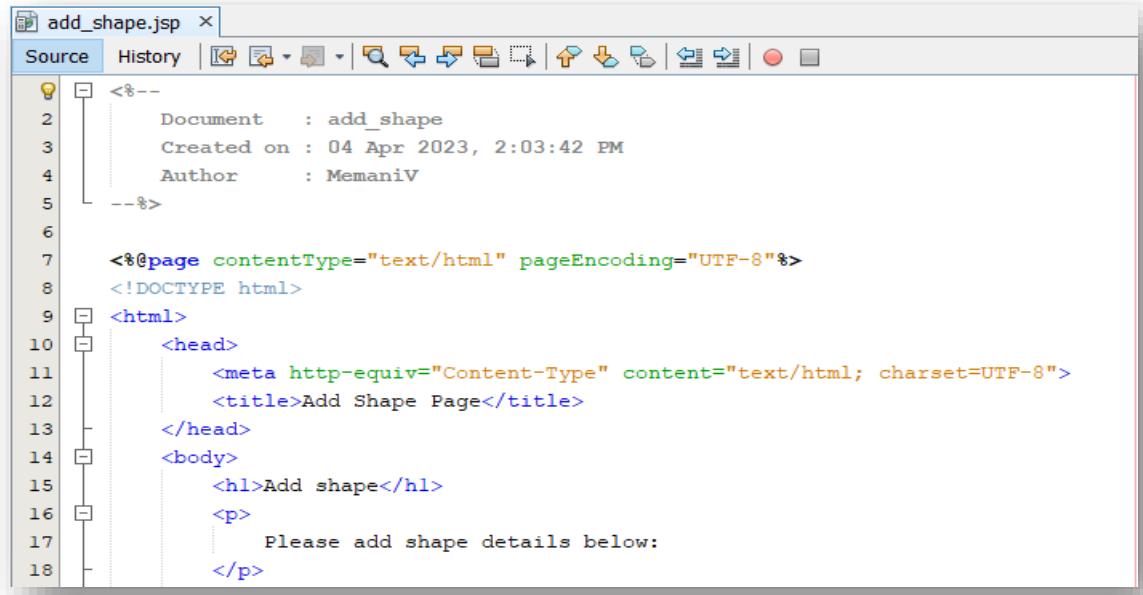
Create the menu.jsp file.



The screenshot shows a code editor window with the title bar "add_shape.jsp". The tab bar has "Source" selected. The code itself is a JSP page with the following content:

```
<%--  
1 Document      : add_shape  
2 Created on   : 04 Apr 2023, 10:49:24 PM  
3 Author        : MemaniV  
4 --%>  
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="add_shape.jsp">here</a> to add a shape.</li>  
21         <li>Click <a href="get_shapes.jsp">here</a> to view all the shapes.</li>  
22     </ol>  
23 </body>  
24 </html>
```

Create add_shape.jsp file.

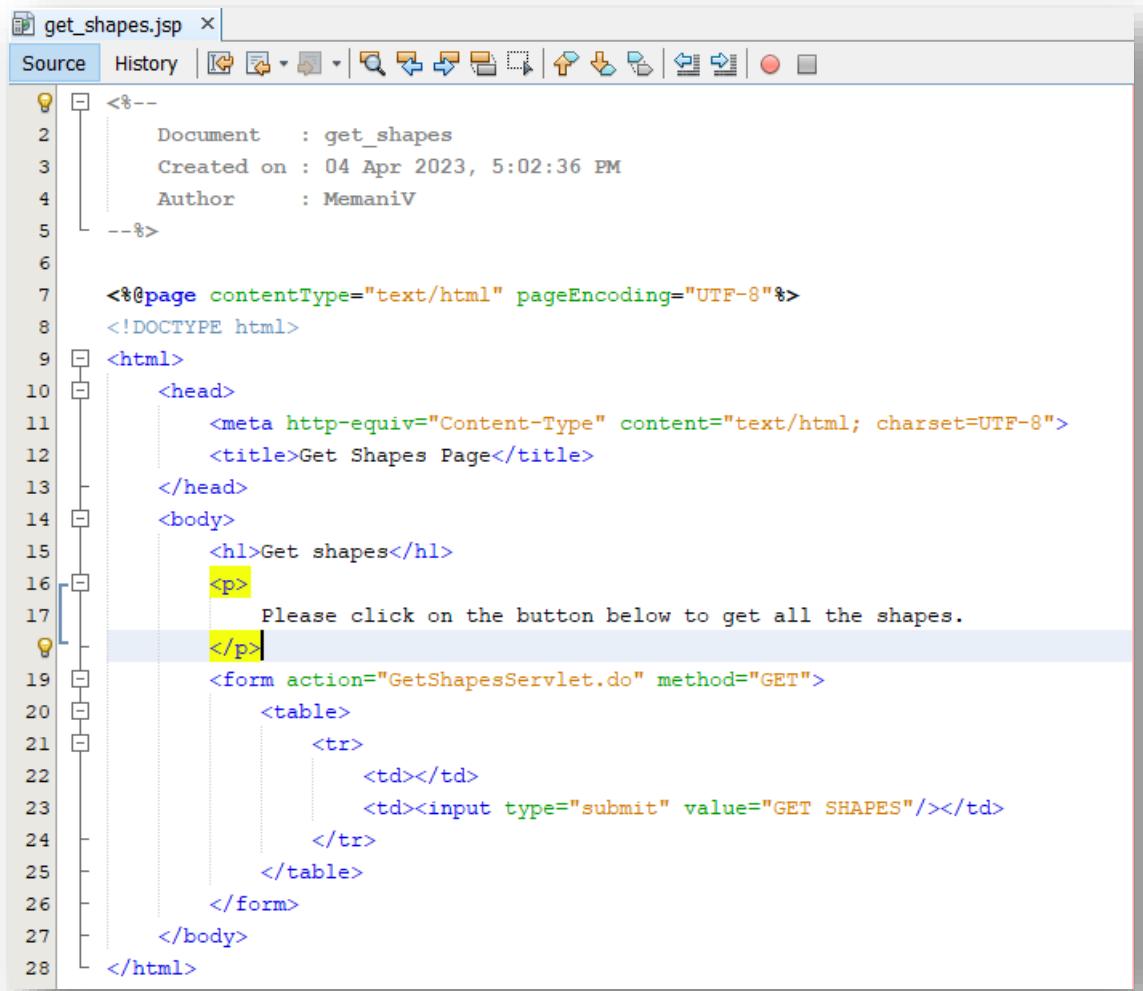


The screenshot shows a code editor window with the title bar "add_shape.jsp". The tab bar has "Source" selected. The code is a JSP page with the following content:

```
<%--  
1 Document      : add_shape  
2 Created on   : 04 Apr 2023, 2:03:42 PM  
3 Author        : MemaniV  
4 --%>  
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>Add Shape Page</title>  
13 </head>  
14 <body>  
15     <h1>Add shape</h1>  
16     <p>  
17         Please add shape details below:  
18     </p>
```

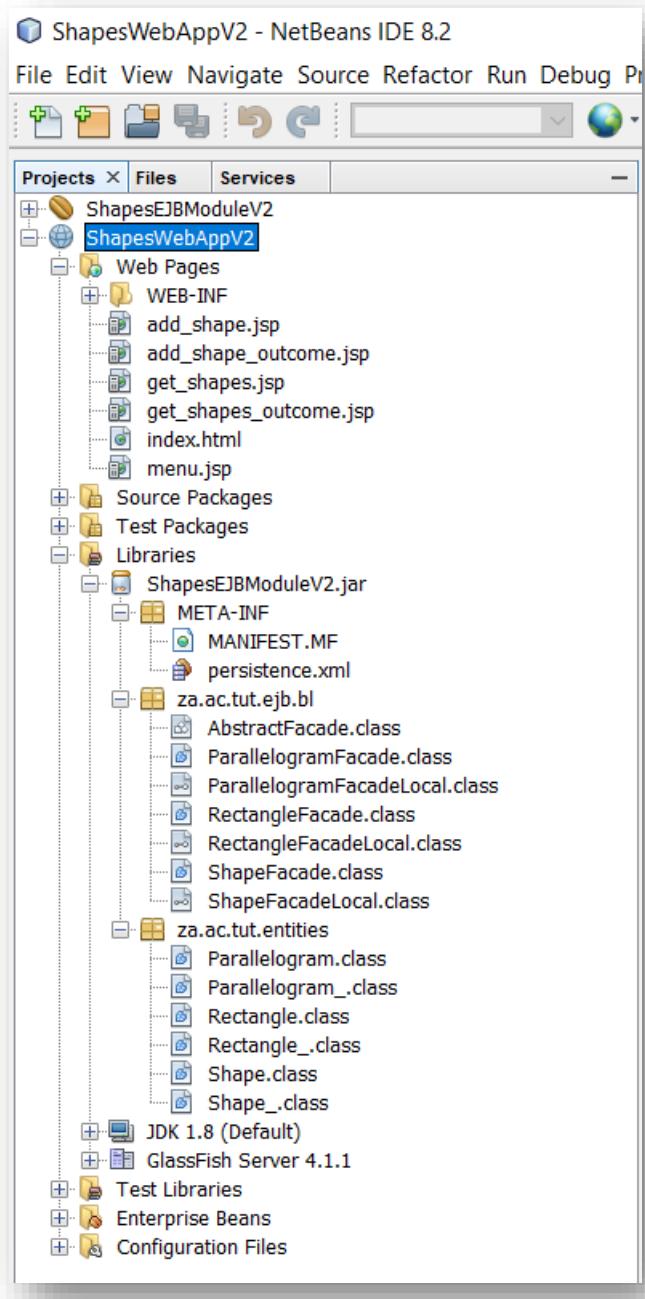
```
19 | 19 | <form action="AddShapeServlet.do" method="POST">
20 | 20 |   <table>
21 | 21 |     <tr>
22 | 22 |       <td>Type: </td>
23 | 23 |       <td>
24 | 24 |         <select name="type">
25 | 25 |           <option value="P">Parallelogram</option>
26 | 26 |           <option value="R">Rectangle</option>
27 | 27 |         </select>
28 | 28 |       </td>
29 | 29 |     </tr>
30 | 30 |     <tr>
31 | 31 |       <td>Side 1: </td>
32 | 32 |       <td><input type="text" name="side1"/></td>
33 | 33 |     </tr>
34 | 34 |     <tr>
35 | 35 |       <td>Side 2: </td>
36 | 36 |       <td><input type="text" name="side2"/></td>
37 | 37 |     </tr>
38 | 38 |     <tr>
39 | 39 |       <td>Is a big shape? </td>
40 | 40 |       <td>
41 | 41 |         <select name="isBig">
42 | 42 |           <option value="True">Yes</option>
43 | 43 |           <option value="False">No</option>
44 | 44 |         </select>
45 | 45 |       </td>
46 | 46 |     </tr>
47 | 47 |     <tr>
48 | 48 |       <td></td>
49 | 49 |       <td><input type="submit" value="ADD SHAPE"/></td>
50 | 50 |     </tr>
51 | 51 |   </table>
52 | 52 | </form>
53 | 53 | </body>
54 | 54 | </html>
```

Create the get_shapes.jsp file.

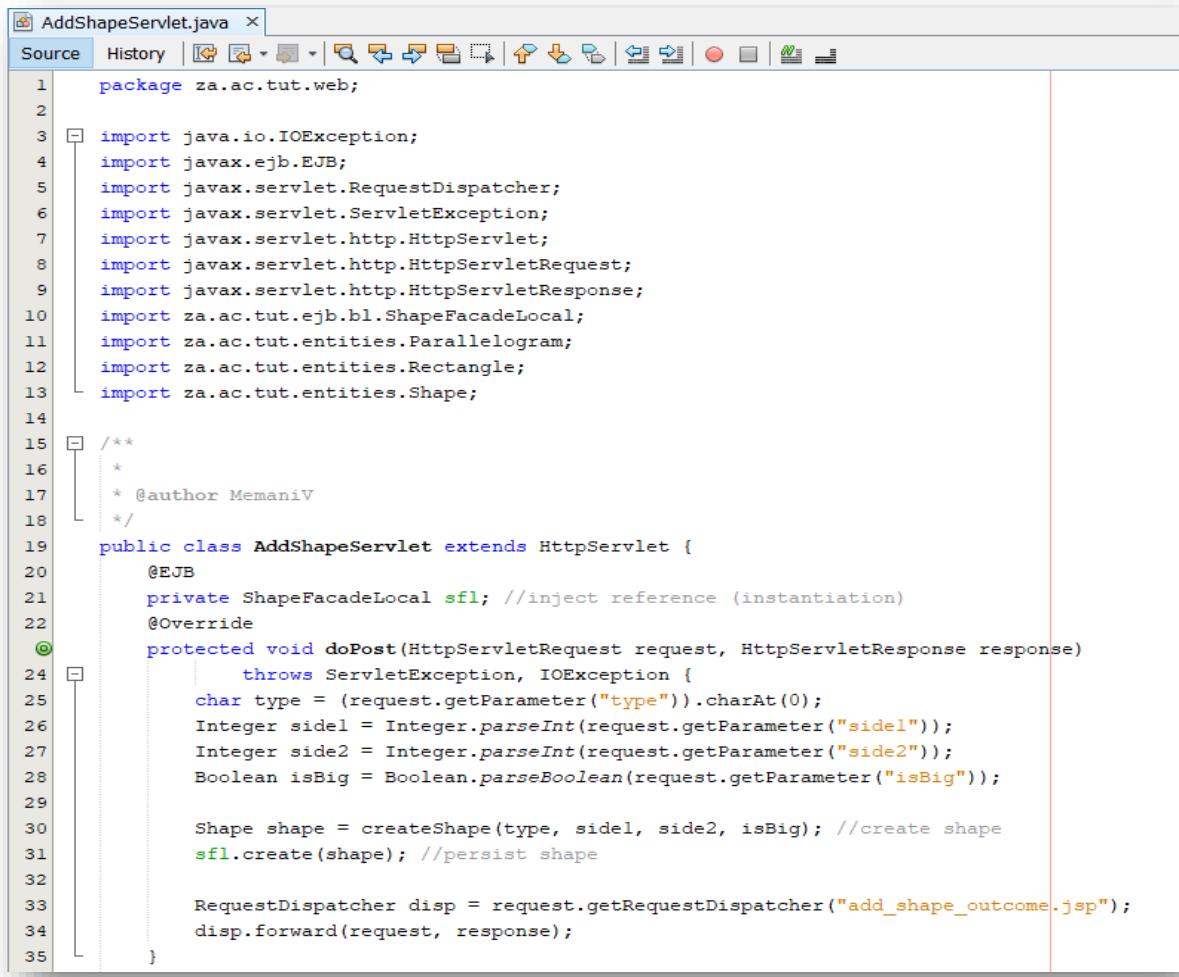


```
<%--  
1 Document      : get_shapes  
2 Created on   : 04 Apr 2023, 5:02:36 PM  
3 Author        : MemaniV  
4--%>  
  
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>  
8 <!DOCTYPE html>  
9 <html>  
10 <head>  
11     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
12     <title>Get Shapes Page</title>  
13 </head>  
14 <body>  
15     <h1>Get shapes</h1>  
16     <p>  
17         Please click on the button below to get all the shapes.  
18     </p>  
19     <form action="GetShapesServlet.do" method="GET">  
20         <table>  
21             <tr>  
22                 <td></td>  
23                 <td><input type="submit" value="GET SHAPES"/></td>  
24             </tr>  
25         </table>  
26     </form>  
27 </body>  
28 </html>
```

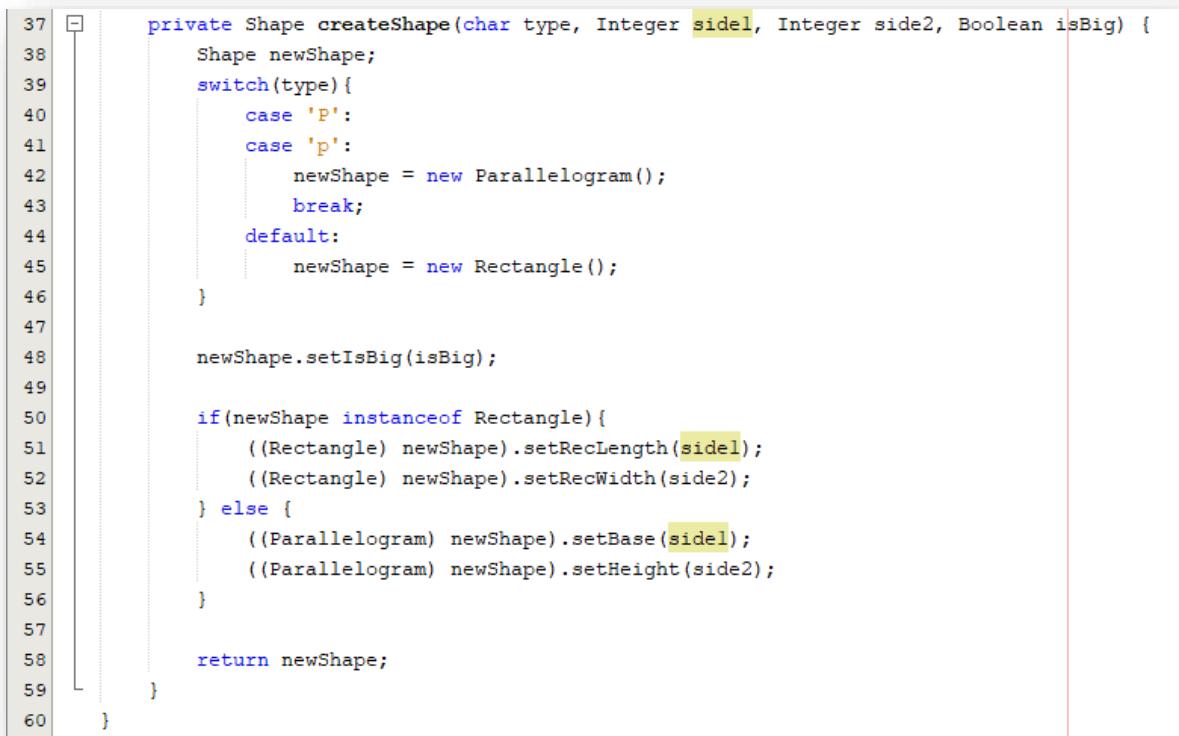
Add the **ShapesEJBModuleV2.jar** file to the library of **ShapesWebAppV2**.



Create the AddShapeServlet.java file.



```
1 package za.ac.tut.web;
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.ShapeFacadeLocal;
11 import za.ac.tut.entities.Parallelogram;
12 import za.ac.tut.entities.Rectangle;
13 import za.ac.tut.entities.Shape;
14
15 /**
16 *
17 * @author MemaniV
18 */
19 public class AddShapeServlet extends HttpServlet {
20     @EJB
21     private ShapeFacadeLocal sfl; //inject reference (instantiation)
22     @Override
23     protected void doPost(HttpServletRequest request, HttpServletResponse response)
24         throws ServletException, IOException {
25         char type = (request.getParameter("type")).charAt(0);
26         Integer sidel = Integer.parseInt(request.getParameter("sidel"));
27         Integer side2 = Integer.parseInt(request.getParameter("side2"));
28         Boolean isBig = Boolean.parseBoolean(request.getParameter("isBig"));
29
30         Shape shape = createShape(type, sidel, side2, isBig); //create shape
31         sfl.create(shape); //persist shape
32
33         RequestDispatcher disp = request.getRequestDispatcher("add_shape_outcome.jsp");
34         disp.forward(request, response);
35     }
36 }
```



```
37     private Shape createShape(char type, Integer sidel, Integer side2, Boolean isBig) {
38         Shape newShape;
39         switch(type){
40             case 'P':
41             case 'p':
42                 newShape = new Parallelogram();
43                 break;
44             default:
45                 newShape = new Rectangle();
46             }
47
48             newShape.setIsBig(isBig);
49
50             if(newShape instanceof Rectangle){
51                 ((Rectangle) newShape).setRecLength(sidel);
52                 ((Rectangle) newShape).setRecWidth(side2);
53             } else {
54                 ((Parallelogram) newShape).setBase(sidel);
55                 ((Parallelogram) newShape).setHeight(side2);
56             }
57
58             return newShape;
59         }
60     }
```

Create the **GetShapesServlet.java** file.

Create the **add_book_outcome.jsp** file.

Create the get_shapes_outcome.jsp file.

```
<%--  
1 Document      : get_shapes_outcome  
2 Created on   : 04 Apr 2023, 5:40:33 PM  
3 Author        : MemaniV  
4 --%>  
5  
6  
7 <%@page import="za.ac.tut.entities.Parallelogram"%>  
8 <%@page import="za.ac.tut.entities.Rectangle"%>  
9 <%@page import="java.util.List"%>  
10 <%@page import="za.ac.tut.entities.Shape"%>  
11 <%@page import="za.ac.tut.entities.Shape"%>  
12 <%@page contentType="text/html" pageEncoding="UTF-8"%>  
13 <!DOCTYPE html>  
14 <html>  
15     <head>  
16         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
17         <title>Get Shapes Outcome Page</title>  
18     </head>  
19     <body>  
20         <h1>Get shapes outcome</h1>  
21         <%  
22             List<Shape> shapes = (List<Shape>)request.getAttribute("shapes");  
23         %>  
24         <p>  
25             Below is the list of shapes retrieved from the database.  
26         </p>
```

```

27   <table>
28     <%
29       String shapeType = "Parallelogram";
30       Integer sidel, side2;
31       for(int i = 0; i < shapes.size(); i++) {
32         Shape p = shapes.get(i);
33         Long id = p.getId();
34         Boolean isBig = p.getIsBig();
35
36         if(p instanceof Rectangle) {
37             shapeType = "Rectangle";
38             Rectangle rec = (Rectangle)p;
39             sidel = rec.getRecLength();
40             side2 = rec.getRecWidth();
41         } else {
42             Parallelogram par = (Parallelogram)p;
43             sidel = par.getBase();
44             side2 = par.getHeight();
45         }
46     %>
47     <tr>
48       <td>ID: </td>
49       <td><%=id%></td>
50     </tr>
51     <tr>
52       <td>Shape type: </td>
53       <td><%=shapeType%></td>
54     </tr>
55     <tr>
56       <td>Side 1: </td>
57       <td><%=sidel%></td>
58     </tr>
59     <tr>
60       <td>Side2: </td>
61       <td><%=side2%></td>

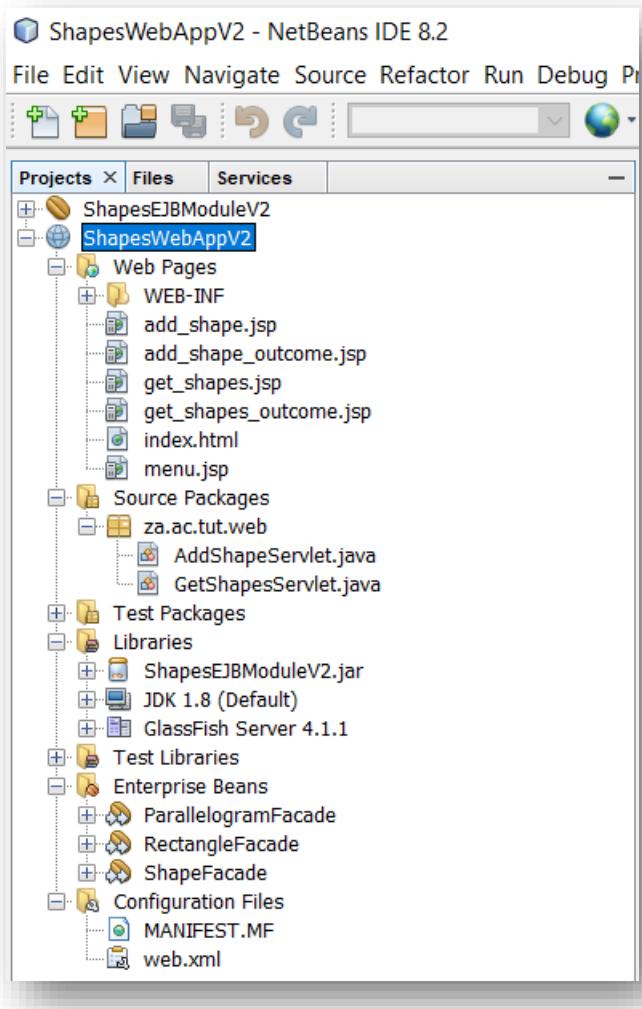
```

```

63   <tr>
64     <td>Is the shape big? </td>
65     <td><%=isBig%></td>
66   </tr>
67   <%
68   %>
69   </table>
70   <p>
71     Please click <a href="menu.jsp">here</a> to get back to the menu page or <a href="index.html">here</a>
72     to the main page.
73   </p>
74   </body>
75 </html>

```

View the complete project structure of **ShapesWebAppV2**.

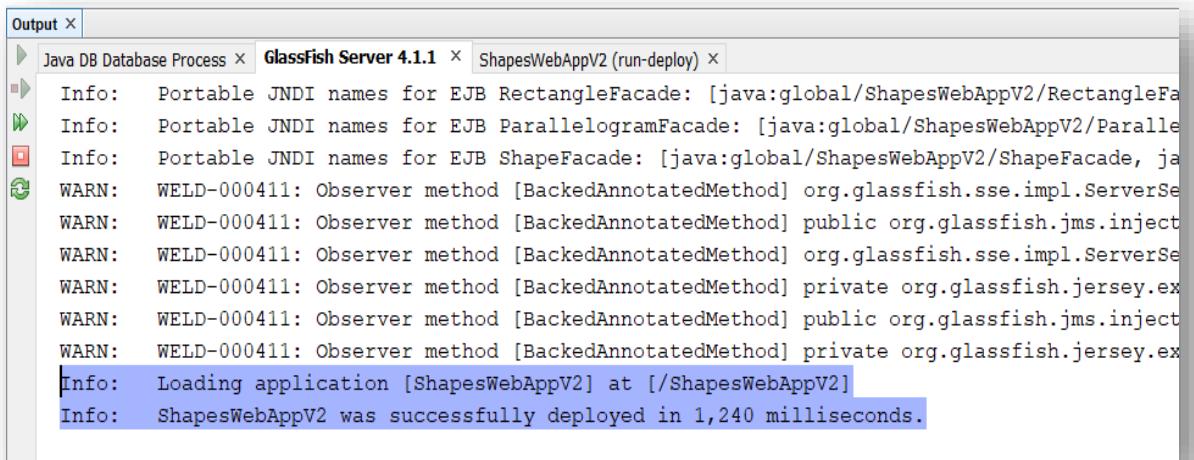


Compile the project.

The screenshot shows the NetBeans Output window with the following log output:

```
library-inclusion-in-manifest:  
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\ShapesWebAppV2\build\empty  
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\ShapesWebAppV2\build\generated-sources\ap-source-output  
Compiling 2 source files to C:\Users\memaniv\Documents\NetBeansProjects\ShapesWebAppV2\build\web\WEB-INF\classes  
compile:  
compile-jsp:  
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\ShapesWebAppV2\dist  
Building jar: C:\Users\memaniv\Documents\NetBeansProjects\ShapesWebAppV2\dist\ShapesWebAppV2.war  
do-dist:  
dist:  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Deploy the project.

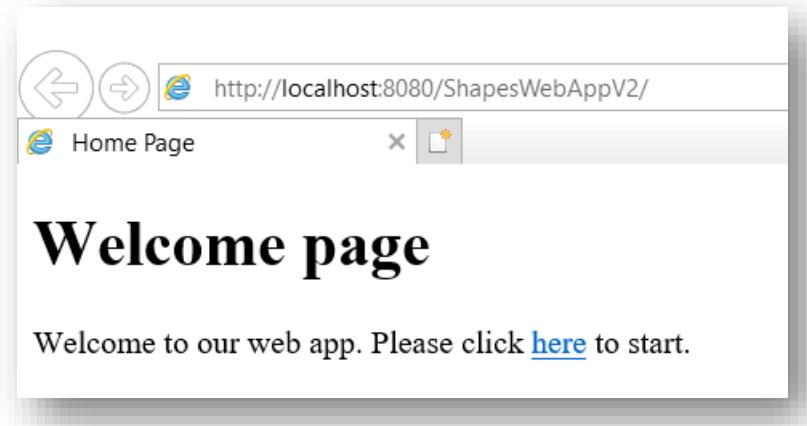


The screenshot shows the NetBeans IDE's Output window with the tab 'GlassFish Server 4.1.1' selected. The window displays deployment logs for the application 'ShapesWebAppV2'. The logs include various informational and warning messages related to JNDI names for EJBs and WELD observer methods, followed by a message indicating the successful deployment of the application in 1,240 milliseconds.

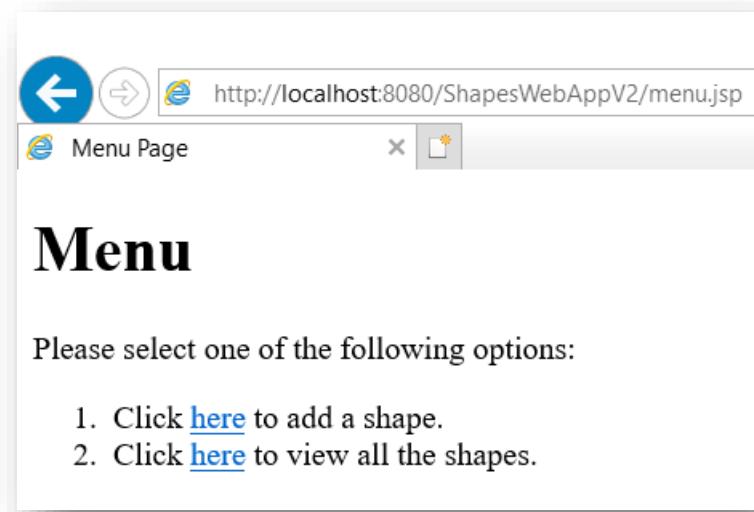
```
Output X
Java DB Database Process x GlassFish Server 4.1.1 x ShapesWebAppV2 (run-deploy) x
Info: Portable JNDI names for EJB RectangleFacade: [java:global/ShapesWebAppV2/RectangleFa
Info: Portable JNDI names for EJB ParallelogramFacade: [java:global/ShapesWebAppV2/Paralle
Info: Portable JNDI names for EJB ShapeFacade: [java:global/ShapesWebAppV2/ShapeFacade, ja
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] org.glassfish.sse.impl.ServerSe
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] public org.glassfish.jms.inject
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] org.glassfish.sse.impl.ServerSe
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] private org.glassfish.jersey.ex
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] public org.glassfish.jms.inject
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] private org.glassfish.jersey.ex
Info: Loading application [ShapesWebAppV2] at [/ShapesWebAppV2]
Info: ShapesWebAppV2 was successfully deployed in 1,240 milliseconds.
```

Part E – Run the web application.

Launch the application.



Click on the link



Add shape book.

- ✓ Click on the first link.

Please add shape details below:

Type:

Side 1:

Side 2:

Is a big shape?

ADD SHAPE

- ✓ Fill in the form.

Please add shape details below:

Type:

Side 1:

Side 2:

Is a big shape?

ADD SHAPE

Click on the add button.

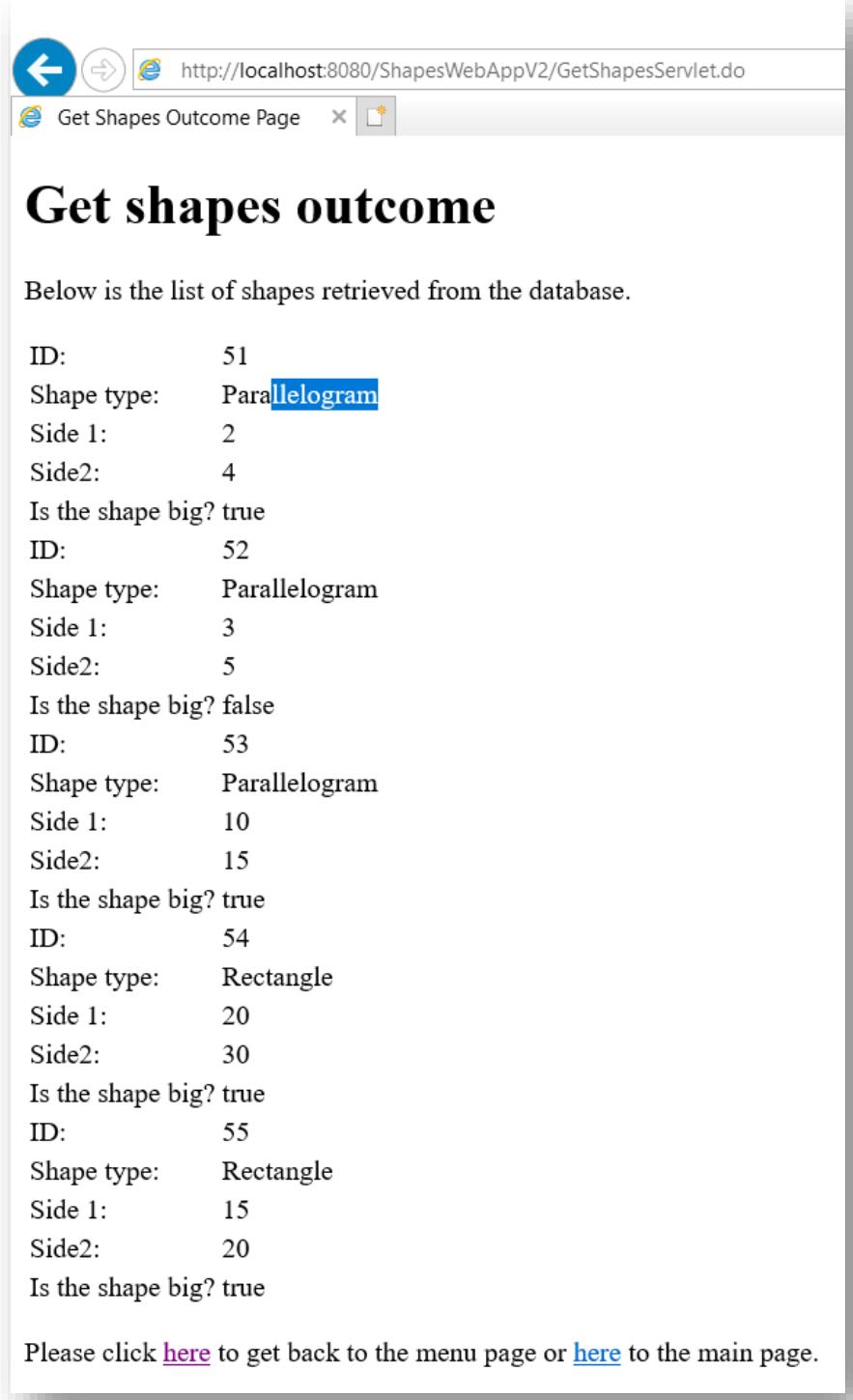
The screenshot shows a web browser window with the URL <http://localhost:8080/ShapesWebAppV2/AddShapeServlet.do>. The title bar says "Add Shape Outcome Page". The main content area has a heading "Add shape outcome" and a message: "The shape has been successfully persisted into the database. Please click [here](#) to get back to the menu page or [here](#) to the main page."

Add four more shapes. View all the shapes.

- ✓ Click on the second link.

The screenshot shows a web browser window with the URL http://localhost:8080/ShapesWebAppV2/get_shapes.jsp. The title bar says "Get Shapes Page". The main content area has a heading "Get shapes" and a message: "Please click on the button below to get all the shapes." Below the message is a button labeled "GET SHAPES".

- ✓ Click on the button.



The screenshot shows a web browser window with the URL <http://localhost:8080/ShapesWebAppV2/GetShapesServlet.do>. The title bar says "Get Shapes Outcome Page". The main content area displays the following text:

Get shapes outcome

Below is the list of shapes retrieved from the database.

ID: 51
Shape type: **Parallelogram**
Side 1: 2
Side2: 4
Is the shape big? true

ID: 52
Shape type: Parallelogram
Side 1: 3
Side2: 5
Is the shape big? false

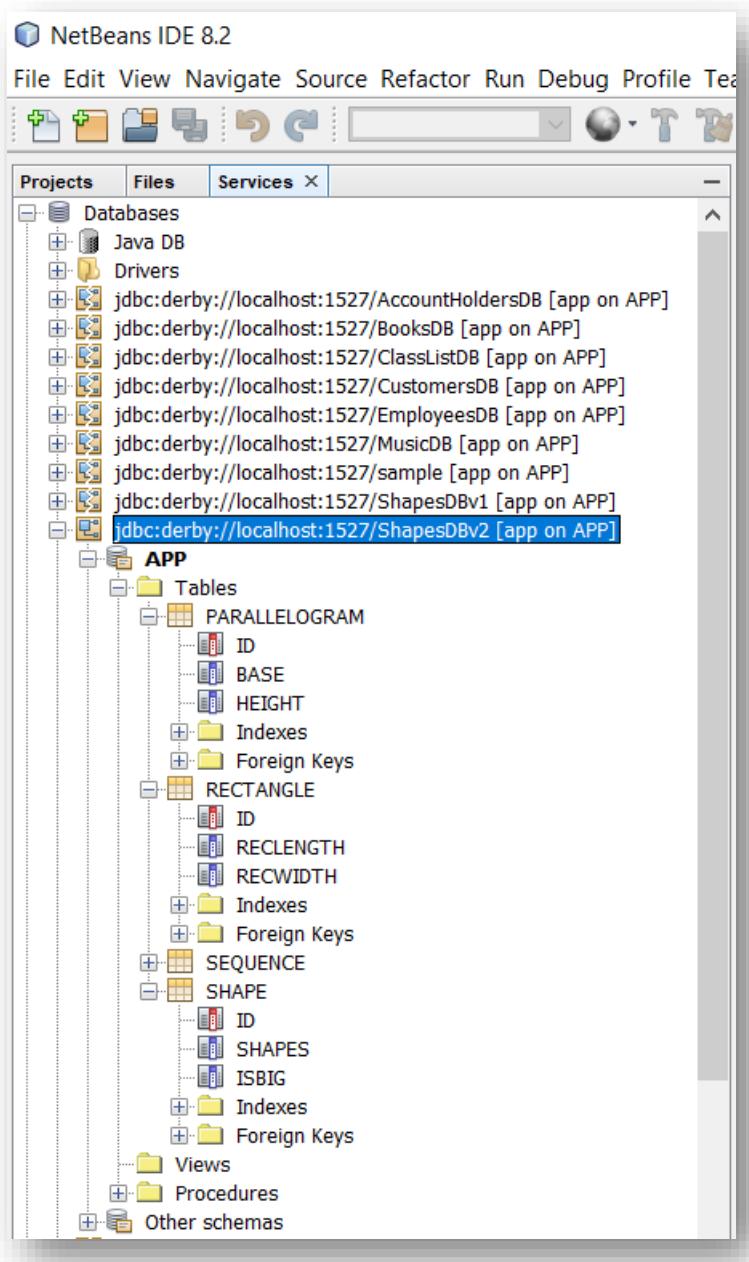
ID: 53
Shape type: Parallelogram
Side 1: 10
Side2: 15
Is the shape big? true

ID: 54
Shape type: Rectangle
Side 1: 20
Side2: 30
Is the shape big? true

ID: 55
Shape type: Rectangle
Side 1: 15
Side2: 20
Is the shape big? true

Please click [here](#) to get back to the menu page or [here](#) to the main page.

View the structure of the database.



Contents of the SHAPE table.

The screenshot shows the SQL window in NetBeans. The connection is set to 'jdbc:derby://localhost:1527/ShapesDBv2 [app on APP]'. The SQL query 'SELECT * FROM APP.SHAPE' is run, and the results are displayed in a table:

#	ID	SHAPES	ISBIG
1		S1 P	1
2		S2 P	0
3		S3 P	1
4		S4 R	1
5		S5 R	1

Contents of the PARALLELOGRAM table.

The screenshot shows the SQL Developer interface with two tabs: SQL 1 and SQL 2. SQL 1 contains the query: `SELECT * FROM APP.PARALLELOGRAM FETCH FIRST 100 ROWS ONLY;`. SQL 2 contains the query: `SELECT * FROM APP.PARALLELOGRAM`. The results of SQL 2 are displayed in a table:

#	ID	BASE	HEIGHT
1		51	2
2		52	3
3		53	10

Contents of the RECTANGLE table.

The screenshot shows the SQL Developer interface with two tabs: SQL 1 and SQL 2. SQL 1 contains the query: `SELECT * FROM APP.RECTANGLE FETCH FIRST 100 ROWS ONLY;`. SQL 2 contains the query: `SELECT * FROM APP.RECTANGLE`. The results of SQL 2 are displayed in a table:

#	ID	RECLENGTH	RECWIDTH
1		54	20
2		55	15

8.5.2 Joined-subclass strategy

When using this strategy, each entity in the hierarchy is mapped to its own table. The main table is mapped to the root entity. The main table will mainly contain the primary key, discriminator column and attributes defined in the root entity.

The sub-entities are mapped to their own tables. In those tables we find references to the primary key defined in the main table, the fields defined in the respective sub-entities, with no reference to inherited fields.

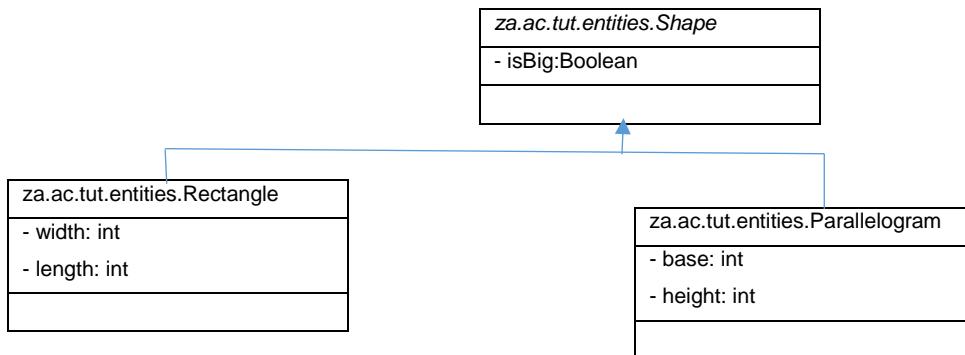
To define a Joined-subclass strategy, we use the `InheritanceType.JOINED` value inside the `@Inheritance` annotation. For example we have something as follows:

```
@Entity
@Inheritance(strategy = InheritanceType.JOINED)
public class Item { }
```

Example

Create a web application to represent a Shape. We have two shapes, a Parallelogram and a Rectangle. A Parallelogram has a **base** and a **height**, and a Rectangle has a **length** and a **width**. The area of Parallelogram can be calculated as **base * height** whilst that of a Rectangle can be calculated as **length * width**. Both shapes have an additional attribute called **isBig** which signals whether a shape is big or not.

The inheritance class hierarchy can be represented as follows:



The equivalent table structure for the above hierarchy will be as follows:

SHAPE		
FIELD	DATA TYPE	CONSTRAINT
ID	INTEGER	PRIMARY KEY
ISBIG	BOOLEAN	NULLABLE=TRUE
WIDTH	INTEGER	NULLABLE=TRUE
LENGTH	INTEGER	NULLABLE = TRUE
BASE	INTEGER	NULLABLE = TRUE
HEIGHT	INTEGER	NULLABLE = TRUE

As can be seen, all the attributes of the classes are put in a single table. Let us do the actual coding.

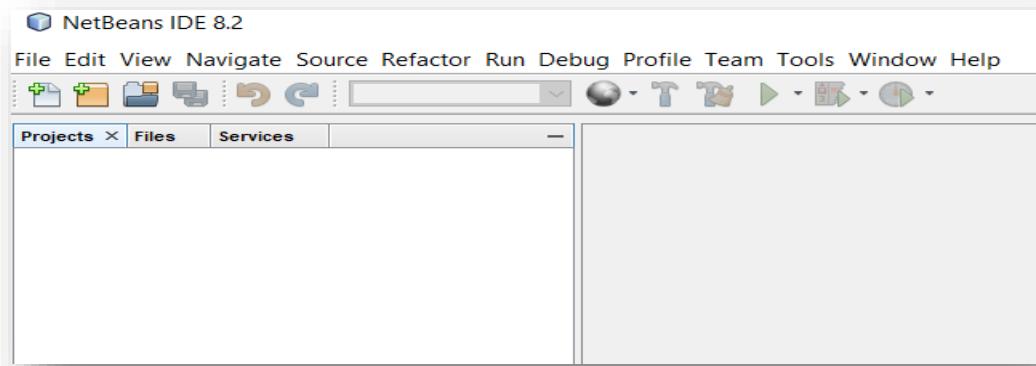
Solution approach

The solution to this problem is going to be done in five parts, namely:

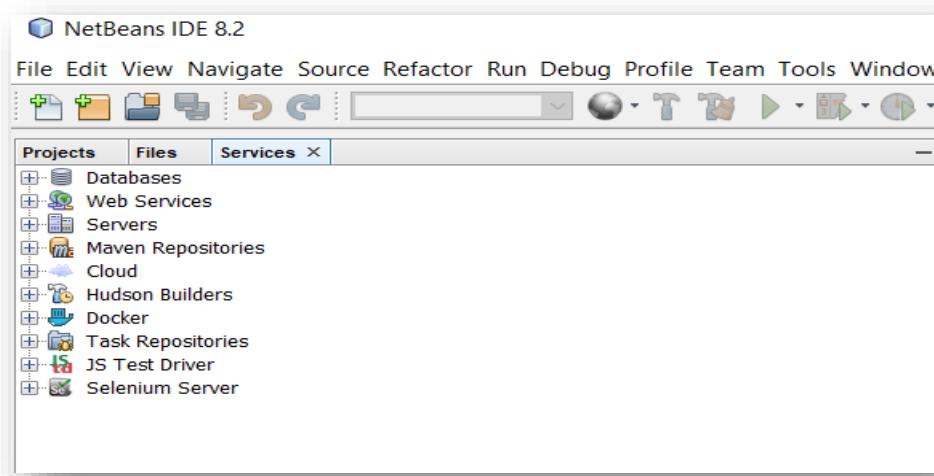
- **Part A:** we create a database.
- **Part B:** we establish connection between the created database and GlassFish.
- **Part C:** we create an EJB module. The module will have an entity with business logic.
- **Part D:** we create a client web application.
- **Part E:** we run the client web application.

Part A - Create a database

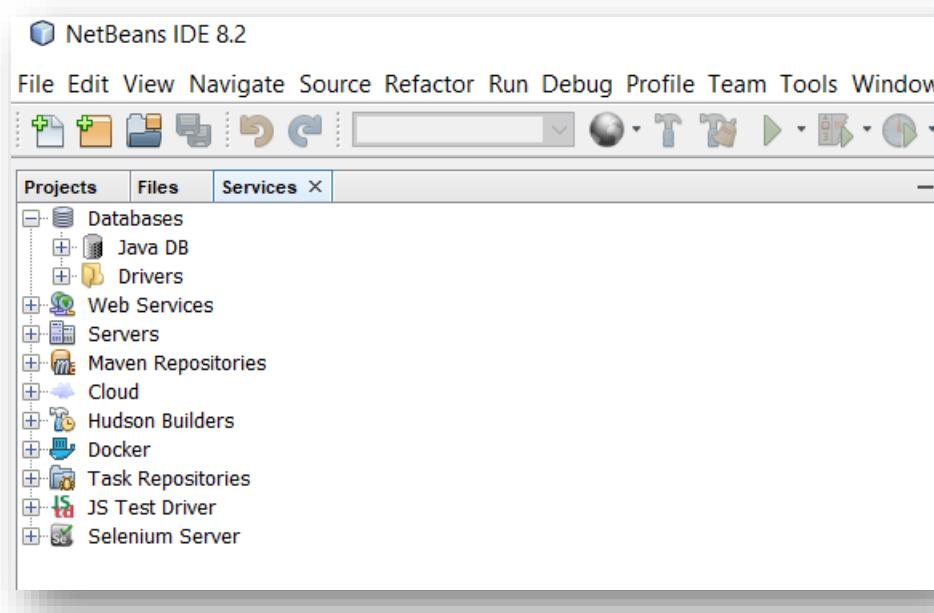
Launch NetBeans.



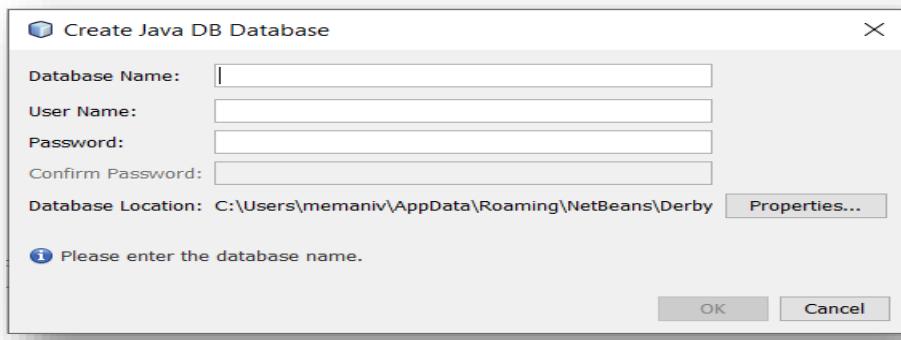
Click on the **Services** tab.



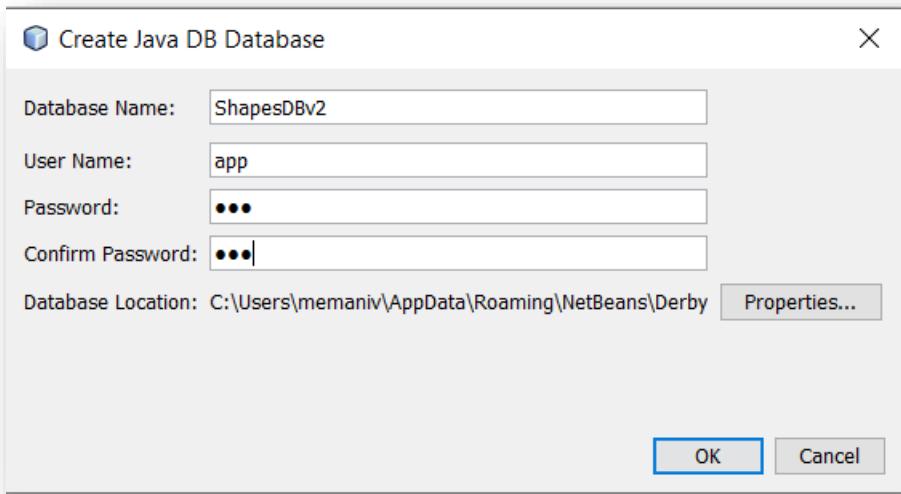
Expand the **Databases** option.



Right-click on **Java DB** and select **Create Database**.



Fill-in the form. I made the password to be **123**.

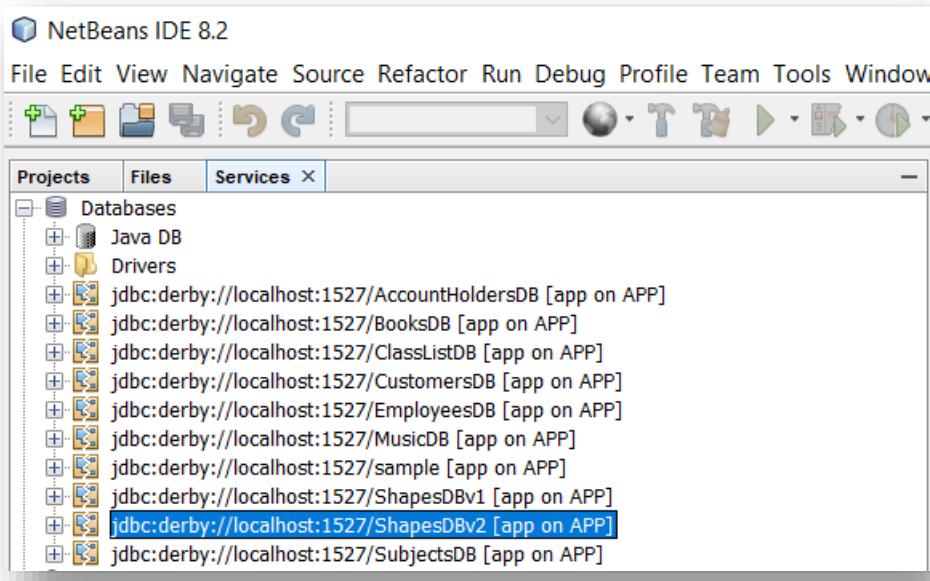


Click **OK**.

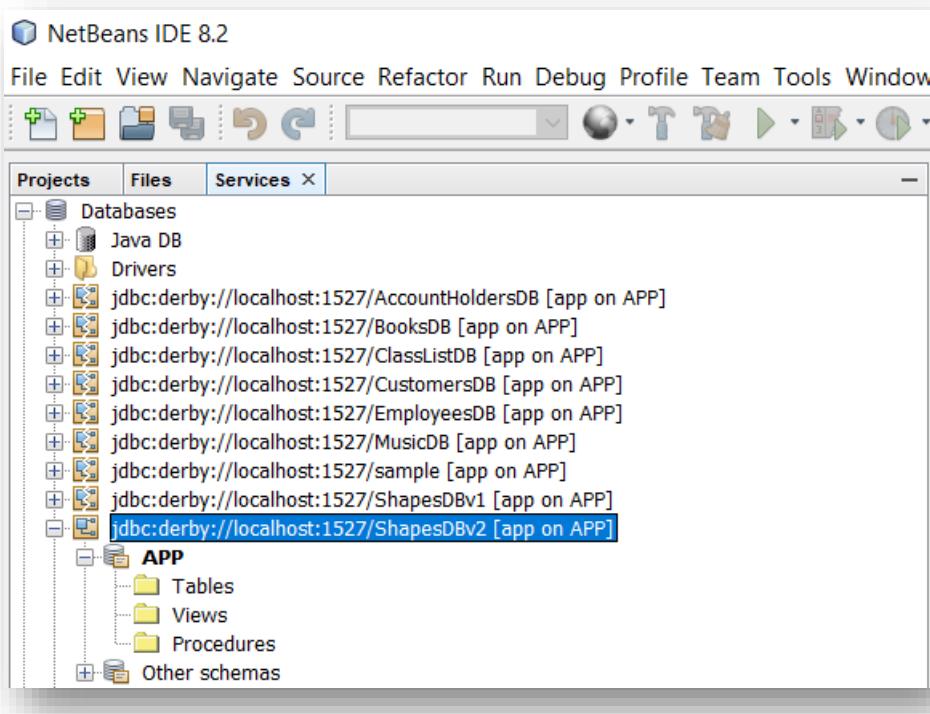
- ✓ An output message confirming that **Derby** has started and ready to accept connections on port 1527 is shown.

A screenshot of the 'Output - Java DB Database Process' window showing command-line output. It displays two lines of text: 'Tue Apr 04 12:45:46 CAT 2023 : Security manager installed using the Basic server security policy.' and 'Tue Apr 04 12:45:47 CAT 2023 : Apache Derby Network Server - 10.11.1.2 - (1629631) started and ready to accept connections on port 1527'.

- ✓ A database is created.

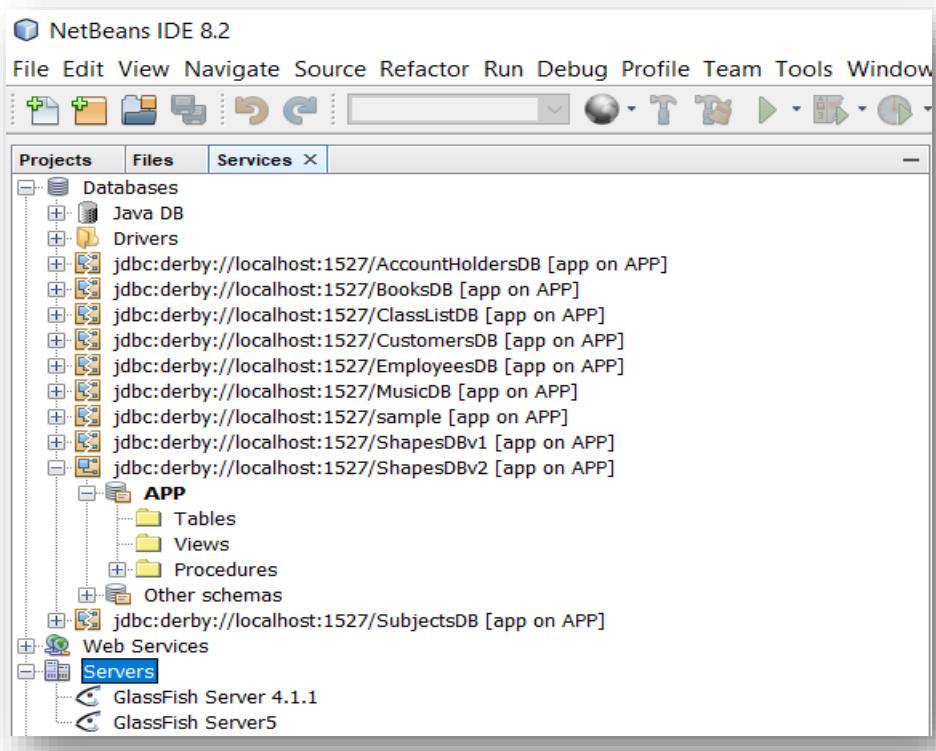


Right-click on the database and select **Connect**.

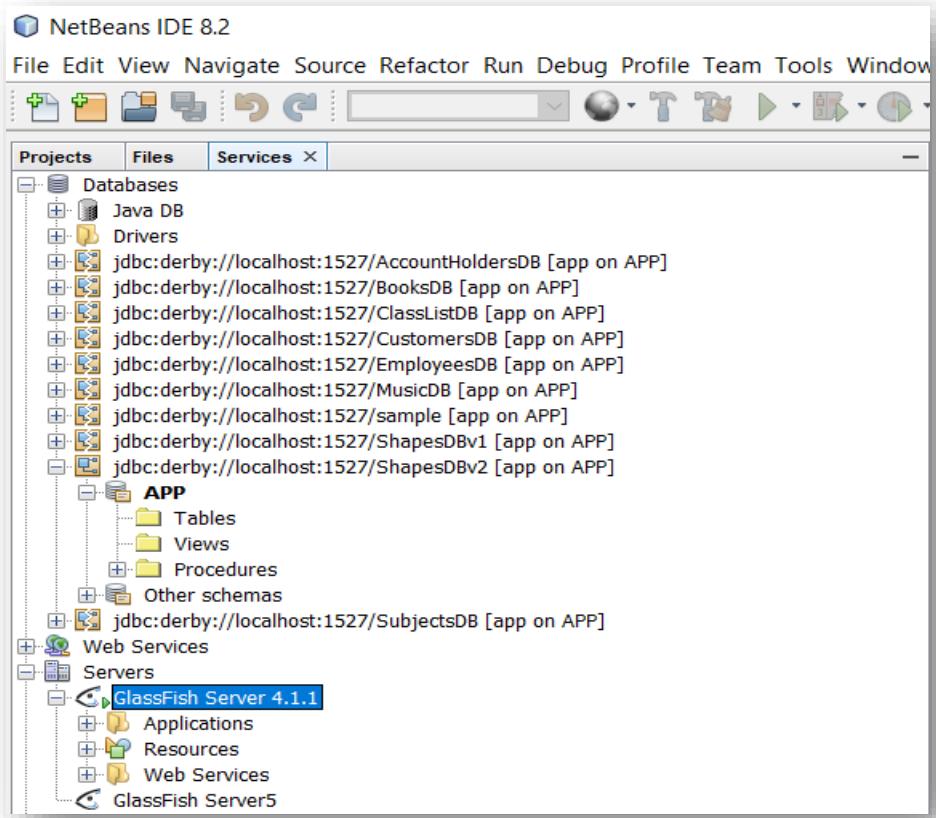


Part B – Connect the database to the application server.

Click on the **Services** tab.



Start the server.



Right-click on the server and select **View Domain Admin Console**.

The screenshot shows the GlassFish Server Open Source Edition Common Tasks console. The URL in the address bar is `http://localhost:4848/common/index.jsf`. The title bar says "GlassFish Console - Comm...". The top menu has "Home" and "About..." buttons. Below that, it shows "User: admin | Domain: domain1 | Server: localhost". The main title is "GlassFish™ Server Open Source Edition". On the left, there's a "Common Tasks" sidebar with the following categories and sub-items:

- Domain
- server (Admin Server)
- Clusters
- Standalone Instances
- Nodes
- Applications
- Lifecycle Modules
- Monitoring Data
- Resources
 - Concurrent Resources
 - Connectors
 - JDBC
 - JMS Resources
 - JNDI
 - JavaMail Sessions
 - Resource Adapter Configs
- Configurations
 - default-config
 - server-config
- Update Tool

The right panel is titled "GlassFish Console - Common Tasks" and contains sections for "GlassFish News", "Deployment" (List Deployed Applications, Deploy an Application), "Administration" (Change Administrator Password, List Password Aliases), and "Monitoring" (Monitoring Data).

Under **Common Tasks** panel, expand **JDBC**.

The screenshot shows the GlassFish Server Open Source Edition JDBC console. The URL in the address bar is `http://localhost:4848/common/index.jsf`. The title bar says "JDBC". The top menu has "Home" and "About..." buttons. Below that, it shows "User: admin | Domain: domain1 | Server: localhost". The main title is "GlassFish™ Server Open Source Edition". On the left, there's a "Common Tasks" sidebar with the following categories and sub-items, where "JDBC" is selected and highlighted in blue:

- Domain
- server (Admin Server)
- Clusters
- Standalone Instances
- Nodes
- Applications
- Lifecycle Modules
- Monitoring Data
- Resources
 - Concurrent Resources
 - Connectors
 - JDBC
 - JDBC Resources
 - JDBC Connection Pools
 - JMS Resources
 - JNDI
 - JavaMail Sessions
 - Resource Adapter Configs
- Configurations
 - default-config
 - server-config
- Update Tool

The right panel is titled "JDBC" and contains links for "JDBC Resources" and "JDBC Connection Pools".

Modify the Connection pool to include the **ShapesDBv2**.

- ✓ Click on **JDBC Connection Pools**.

JDBC Connection Pools					
To store, organize, and retrieve data, most applications use relational databases. Java EE applications access relational databases through the JDBC API. Before an application can access a database, it must get a connection.					
Pools (3)					
Select	Pool Name	Resource Type	Classname	Description	
<input type="checkbox"/>	DerbyPool	javax.sql.DataSource	org.apache.derby.jdbc.ClientDataSource		
<input type="checkbox"/>	SamplePool	javax.sql.DataSource	org.apache.derby.jdbc.ClientDataSource		
<input type="checkbox"/>	TimerPool	javax.sql.XADatasource	org.apache.derby.jdbc.EmbeddedXADataSource		

- ✓ Click on the **DerbyPool** link.

General	Advanced	Additional Properties																									
Edit JDBC Connection Pool Modify an existing JDBC connection pool. A JDBC connection pool is a group of reusable connections for a particular database. <input type="button" value="Load Defaults"/> <input type="button" value="Flush"/> <input type="button" value="Ping"/>																											
General Settings <table> <tr> <td>Pool Name:</td><td>DerbyPool</td></tr> <tr> <td>Resource Type:</td><td>javax.sql.DataSource</td></tr> <tr> <td colspan="2">Must be specified if the datasource class implements more than 1 of the interface.</td></tr> <tr> <td>Datasource Classname:</td><td>org.apache.derby.jdbc.ClientDataSource</td></tr> <tr> <td colspan="2">Vendor-specific classname that implements the DataSource and/or XADataSource APIs</td></tr> <tr> <td>Driver Classname:</td><td></td></tr> <tr> <td colspan="2">Vendor-specific classname that implements the java.sql.Driver interface.</td></tr> <tr> <td>Ping:</td><td><input type="checkbox"/> Enabled</td></tr> <tr> <td colspan="2">When enabled, the pool is pinged during creation or reconfiguration to identify and warn of any erroneous values for its attributes</td></tr> <tr> <td>Deployment Order:</td><td>100</td></tr> <tr> <td colspan="2">Specifies the loading order of the resource at server startup. Lower numbers are loaded first.</td></tr> <tr> <td>Description:</td><td colspan="2"></td></tr> </table>			Pool Name:	DerbyPool	Resource Type:	javax.sql.DataSource	Must be specified if the datasource class implements more than 1 of the interface.		Datasource Classname:	org.apache.derby.jdbc.ClientDataSource	Vendor-specific classname that implements the DataSource and/or XADataSource APIs		Driver Classname:		Vendor-specific classname that implements the java.sql.Driver interface.		Ping:	<input type="checkbox"/> Enabled	When enabled, the pool is pinged during creation or reconfiguration to identify and warn of any erroneous values for its attributes		Deployment Order:	100	Specifies the loading order of the resource at server startup. Lower numbers are loaded first.		Description:		
Pool Name:	DerbyPool																										
Resource Type:	javax.sql.DataSource																										
Must be specified if the datasource class implements more than 1 of the interface.																											
Datasource Classname:	org.apache.derby.jdbc.ClientDataSource																										
Vendor-specific classname that implements the DataSource and/or XADataSource APIs																											
Driver Classname:																											
Vendor-specific classname that implements the java.sql.Driver interface.																											
Ping:	<input type="checkbox"/> Enabled																										
When enabled, the pool is pinged during creation or reconfiguration to identify and warn of any erroneous values for its attributes																											
Deployment Order:	100																										
Specifies the loading order of the resource at server startup. Lower numbers are loaded first.																											
Description:																											
Pool Settings <table> <tr> <td>Initial and Minimum Pool Size:</td><td>8</td><td>Connections</td></tr> <tr> <td colspan="3">Minimum and initial number of connections maintained in the pool</td></tr> <tr> <td>Maximum Pool Size:</td><td>32</td><td>Connections</td></tr> <tr> <td colspan="3">Maximum number of connections that can be created to satisfy client requests</td></tr> <tr> <td>Pool Resize Quantity:</td><td>2</td><td>Connections</td></tr> </table>			Initial and Minimum Pool Size:	8	Connections	Minimum and initial number of connections maintained in the pool			Maximum Pool Size:	32	Connections	Maximum number of connections that can be created to satisfy client requests			Pool Resize Quantity:	2	Connections										
Initial and Minimum Pool Size:	8	Connections																									
Minimum and initial number of connections maintained in the pool																											
Maximum Pool Size:	32	Connections																									
Maximum number of connections that can be created to satisfy client requests																											
Pool Resize Quantity:	2	Connections																									

- ✓ Click on **Additional Properties**.

General	Advanced	Additional Properties																					
Edit JDBC Connection Pool Properties Modify properties of an existing JDBC connection pool.																							
Pool Name: DerbyPool																							
Additional Properties (6) <table border="1"> <thead> <tr> <th>Select</th><th>Name</th><th>Value</th></tr> </thead> <tbody> <tr> <td><input type="checkbox"/></td><td>PortNumber</td><td>1527</td></tr> <tr> <td><input type="checkbox"/></td><td>Password</td><td>APP</td></tr> <tr> <td><input type="checkbox"/></td><td>User</td><td>APP</td></tr> <tr> <td><input type="checkbox"/></td><td>serverName</td><td>localhost</td></tr> <tr> <td><input type="checkbox"/></td><td>DatabaseName</td><td>sun-appserv-samples</td></tr> <tr> <td><input type="checkbox"/></td><td>connectionAttributes</td><td>;create=true</td></tr> </tbody> </table>			Select	Name	Value	<input type="checkbox"/>	PortNumber	1527	<input type="checkbox"/>	Password	APP	<input type="checkbox"/>	User	APP	<input type="checkbox"/>	serverName	localhost	<input type="checkbox"/>	DatabaseName	sun-appserv-samples	<input type="checkbox"/>	connectionAttributes	;create=true
Select	Name	Value																					
<input type="checkbox"/>	PortNumber	1527																					
<input type="checkbox"/>	Password	APP																					
<input type="checkbox"/>	User	APP																					
<input type="checkbox"/>	serverName	localhost																					
<input type="checkbox"/>	DatabaseName	sun-appserv-samples																					
<input type="checkbox"/>	connectionAttributes	;create=true																					

- ✓ Click on **Add Property** and add a **url** property with the following value:

jdbc:derby://localhost:1527/ShapesDBv2

- ✓ Delete the **connectionAttributes** property.

General Advanced Additional Properties

Edit JDBC Connection Pool Properties

Modify properties of an existing JDBC connection pool.

Pool Name: DerbyPool

Additional Properties (6)		
Select	Name	Value
<input type="checkbox"/>	PortNumber	1527
<input type="checkbox"/>	Password	123
<input type="checkbox"/>	User	app
<input type="checkbox"/>	serverName	localhost
<input type="checkbox"/>	DatabaseName	ShapesDBv2
<input type="checkbox"/>	url	jdbc:derby://localhost:1527/ShapesDBv2

- ✓ Save the changes by clicking the **Save** button.

General Advanced Additional Properties

New values successfully saved.

Edit JDBC Connection Pool Properties

Modify properties of an existing JDBC connection pool.

Pool Name: DerbyPool

Additional Properties (6)		
Select	Name	Value
<input type="checkbox"/>	PortNumber	1527
<input type="checkbox"/>	Password	123
<input type="checkbox"/>	User	app
<input type="checkbox"/>	serverName	localhost
<input type="checkbox"/>	DatabaseName	ShapesDBv2
<input type="checkbox"/>	url	jdbc:derby://localhost:1527/ShapesDBv2

- ✓ Check if the created connection pool is working. Do this by clicking the **Ping** button.

Confirm that the resource points to the DerbyPool.

- ✓ Still under **JDBC**, click on **JDBC Resources**.

Select	JNDI Name	Logical JNDI Name	Enabled	Connection Pool
<input type="checkbox"/>	jdbc/_TimerPool		<input checked="" type="checkbox"/>	_TimerPool
<input type="checkbox"/>	jdbc/_default	java:comp/DefaultDataSource	<input checked="" type="checkbox"/>	DerbyPool
<input type="checkbox"/>	jdbc/sample		<input checked="" type="checkbox"/>	SamplePool

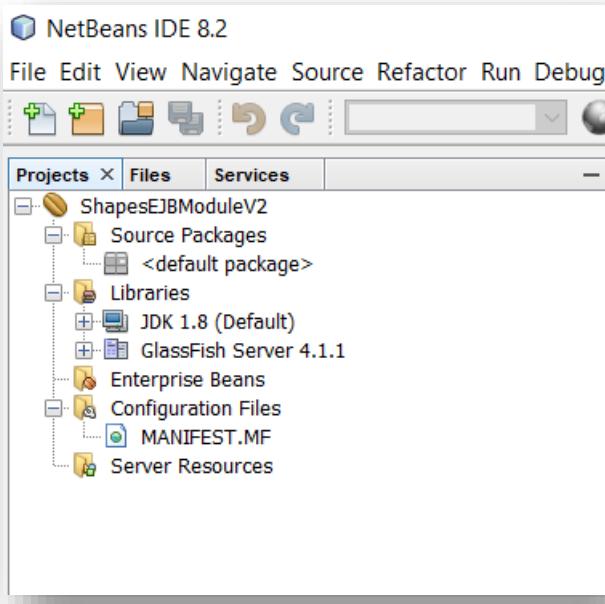
Take note of the purpose of JDBC Resources. It is said:

"JDBC resources provide applications with a means to connect to a database."

You can also see that the **jdbc/_default** resource is associated with the **DerbyPool** Connection Pool.

Part C – Create an entity.

Create an EJB project called **ShapesEJBModuleV2**.



Create the **Shape** entity. Let it use the **Joined-subclass strategy**.

A screenshot of the NetBeans IDE 8.2 code editor for "Shape.java". The code implements the **Joined-subclass strategy**. The annotations used are: `@Entity`, `@Inheritance(strategy = InheritanceType.JOINED)`, `@DiscriminatorColumn(name="shapes", discriminatorType=DiscriminatorType.CHAR)`, and `@DiscriminatorValue("S")`. The code also includes imports for Serializable, DiscriminatorColumn, DiscriminatorType, DiscriminatorValue, Entity, GeneratedValue, GenerationType, Id, Inheritance, and InheritanceType. The class definition starts with `public class Shape implements Serializable {`.

```
1 package za.ac.tut.entities;
2
3 import java.io.Serializable;
4 import javax.persistence.DiscriminatorColumn;
5 import javax.persistence.DiscriminatorType;
6 import javax.persistence.DiscriminatorValue;
7 import javax.persistence.Entity;
8 import javax.persistence.GeneratedValue;
9 import javax.persistence.GenerationType;
10 import javax.persistence.Id;
11 import javax.persistence.Inheritance;
12 import javax.persistence.InheritanceType;
13
14 /**
15 * @author MemaniV
16 */
17
18 @Entity
19 @Inheritance(strategy = InheritanceType.JOINED)
20 @DiscriminatorColumn(name="shapes", discriminatorType=DiscriminatorType.CHAR)
21 @DiscriminatorValue("S")
22 public class Shape implements Serializable {
23
24     private static final long serialVersionUID = 1L;
25     @Id
26     @GeneratedValue(strategy = GenerationType.AUTO)
27     private Long id;
28     private Boolean isBig;
```

```
30  ┌─┐    public Shape() {
31  ┌─┘    }
32
33  ┌─┐    public Shape(Boolean isBig) {
34  ┌─┘      this.isBig = isBig;
35  ┌─┘    }
36
37  ┌─┐    public Boolean getIsBig() {
38  ┌─┘      return isBig;
39  ┌─┘    }
40
41  ┌─┐    public void setIsBig(Boolean isBig) {
42  ┌─┘      this.isBig = isBig;
43  ┌─┘    }
44
45  ┌─┐    public Long getId() {
46  ┌─┘      return id;
47  ┌─┘    }
48
49  ┌─┐    public void setId(Long id) {
50  ┌─┘      this.id = id;
51  ┌─┘    }
52
53    @Override
54  ┌─┐    public int hashCode() {
55  ┌─┘      int hash = 0;
56  ┌─┘      hash += (id != null ? id.hashCode() : 0);
57  ┌─┘      return hash;
58  ┌─┘    }
```

```
60
61  ┌─┐    @Override
62  ┌─┐    public boolean equals(Object object) {
63  ┌─┘      if (!(object instanceof Shape)) {
64  ┌─┘          return false;
65  ┌─┘      }
66  ┌─┐      Shape other = (Shape) object;
67  ┌─┐      if ((this.id == null &amp; other.id != null) ||
68  ┌─┘          (this.id != null &amp; !this.id.equals(other.id))) {
69  ┌─┘          return false;
70  ┌─┘      }
71  ┌─┘      return true;
72  ┌─┘    }
73
74  ┌─┐    @Override
75  ┌─┐    public String toString() {
76  ┌─┘      return "za.ac.tut.entities.Shape[ id=" + id + " ]";
77  ┌─┘    }
78  ┌─┘ }
```

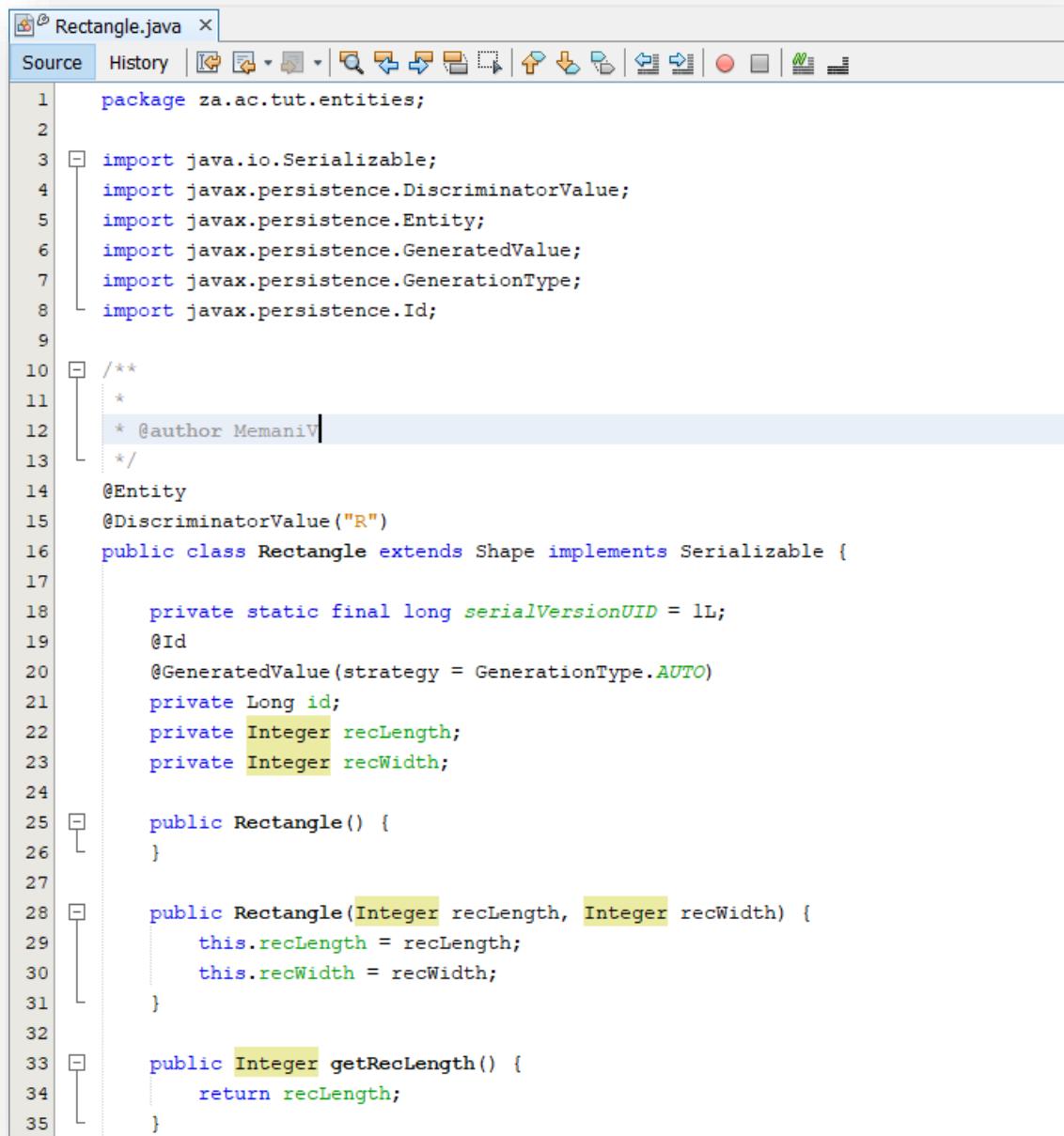
Create a sub entity called **Parallelogram**.

```
1 package za.ac.tut.entities;
2
3 import java.io.Serializable;
4 import javax.persistence.DiscriminatorValue;
5 import javax.persistence.Entity;
6 import javax.persistence.GeneratedValue;
7 import javax.persistence.GenerationType;
8 import javax.persistence.Id;
9
10 /**
11 *
12 * @author MemaniV
13 */
14 @Entity
15 @DiscriminatorValue("P")
16 public class Parallelogram extends Shape implements Serializable {
17
18     private static final long serialVersionUID = 1L;
19     @Id
20     @GeneratedValue(strategy = GenerationType.AUTO)
21     private Long id;
22     private Integer base;
23     private Integer height;
24
25     public Parallelogram() {
26     }
27
28     public Parallelogram(Integer base, Integer height) {
29         this.base = base;
30         this.height = height;
31     }
32
33     public Integer getBase() {
34         return base;
35     }
```

```
37 |     public void setBase(Integer base) {
38 |         this.base = base;
39 |
40 |
41 |     public Integer getHeight() {
42 |         return height;
43 |     }
44 |
45 |     public void setHeight(Integer height) {
46 |         this.height = height;
47 |     }
48 |
49 |
50 |     public Long getId() {
51 |         return id;
52 |     }
53 |
54 |     @Override
55 |     public void setId(Long id) {
56 |         this.id = id;
57 |     }
58 |
59 |     @Override
60 |     public int hashCode() {
61 |         int hash = 0;
62 |         hash += (id != null ? id.hashCode() : 0);
63 |         return hash;
64 |     }
```

```
66 |     @Override
67 |     public boolean equals(Object object) {
68 |         if (!(object instanceof Parallelogram)) {
69 |             return false;
70 |         }
71 |         Parallelogram other = (Parallelogram) object;
72 |         if ((this.id == null && other.id != null) ||
73 |             (this.id != null && !this.id.equals(other.id))) {
74 |             return false;
75 |         }
76 |         return true;
77 |     }
78 |
79 |     @Override
80 |     public String toString() {
81 |         return "za.ac.tut.entities.Parallelogram[ id=" + id + " ]";
82 |     }
83 |
84 | }
```

Create a **Rectangle** sub entity.



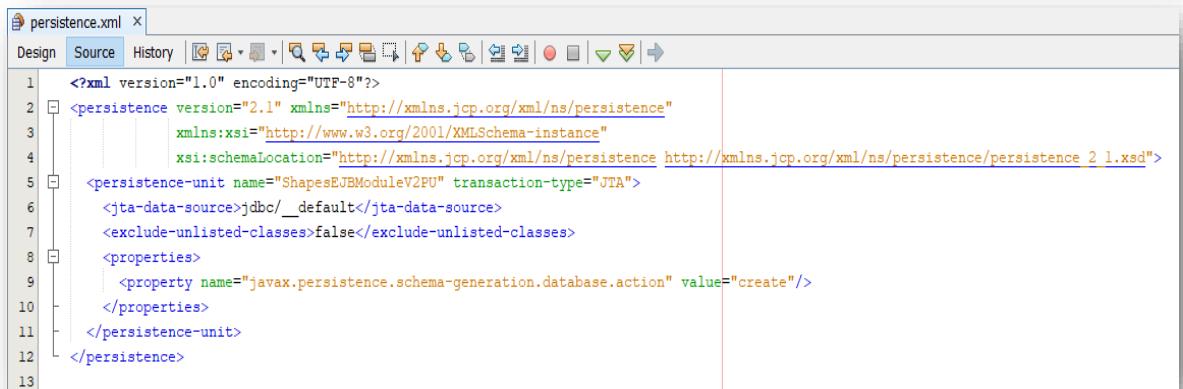
The screenshot shows a Java code editor window titled "Rectangle.java". The code defines a class named "Rectangle" that extends "Shape" and implements "Serializable". The class has two private fields: "recLength" and "recWidth", both of type Integer. It includes a constructor that takes two Integer parameters and initializes the fields. There are also getter and setter methods for "recLength". The code uses annotations from the javax.persistence package, such as @Entity, @DiscriminatorValue("R"), and @Id. The code editor interface includes tabs for "Source" and "History", and various toolbars with icons for file operations and code navigation.

```
1 package za.ac.tut.entities;
2
3 import java.io.Serializable;
4 import javax.persistence.DiscriminatorValue;
5 import javax.persistence.Entity;
6 import javax.persistence.GeneratedValue;
7 import javax.persistence.GenerationType;
8 import javax.persistence.Id;
9
10 /**
11 *
12 * @author MemaniV
13 */
14 @Entity
15 @DiscriminatorValue("R")
16 public class Rectangle extends Shape implements Serializable {
17
18     private static final long serialVersionUID = 1L;
19     @Id
20     @GeneratedValue(strategy = GenerationType.AUTO)
21     private Long id;
22     private Integer recLength;
23     private Integer recWidth;
24
25     public Rectangle() {
26     }
27
28     public Rectangle(Integer recLength, Integer recWidth) {
29         this.recLength = recLength;
30         this.recWidth = recWidth;
31     }
32
33     public Integer getRecLength() {
34         return recLength;
35     }
}
```

```
37  public void setRecLength(Integer recLength) {
38      this.recLength = recLength;
39  }
40
41  public Integer getRecWidth() {
42      return recWidth;
43  }
44
45  public void setRecWidth(Integer recWidth) {
46      this.recWidth = recWidth;
47  }
48
49  @Override
50  public Long getId() {
51      return id;
52  }
53
54  @Override
55  public void setId(Long id) {
56      this.id = id;
57  }
58
59  @Override
60  public int hashCode() {
61      int hash = 0;
62      hash += (id != null ? id.hashCode() : 0);
63      return hash;
64  }
```

```
66  @Override
67  public boolean equals(Object object) {
68      if (!(object instanceof Rectangle)) {
69          return false;
70      }
71      Rectangle other = (Rectangle) object;
72      if ((this.id == null && other.id != null) ||
73          (this.id != null && !this.id.equals(other.id))) {
74          return false;
75      }
76      return true;
77  }
78
79  @Override
80  public String toString() {
81      return "za.ac.tut.entities.Rectangle[ id=" + id + " ]";
82  }
83
84 }
```

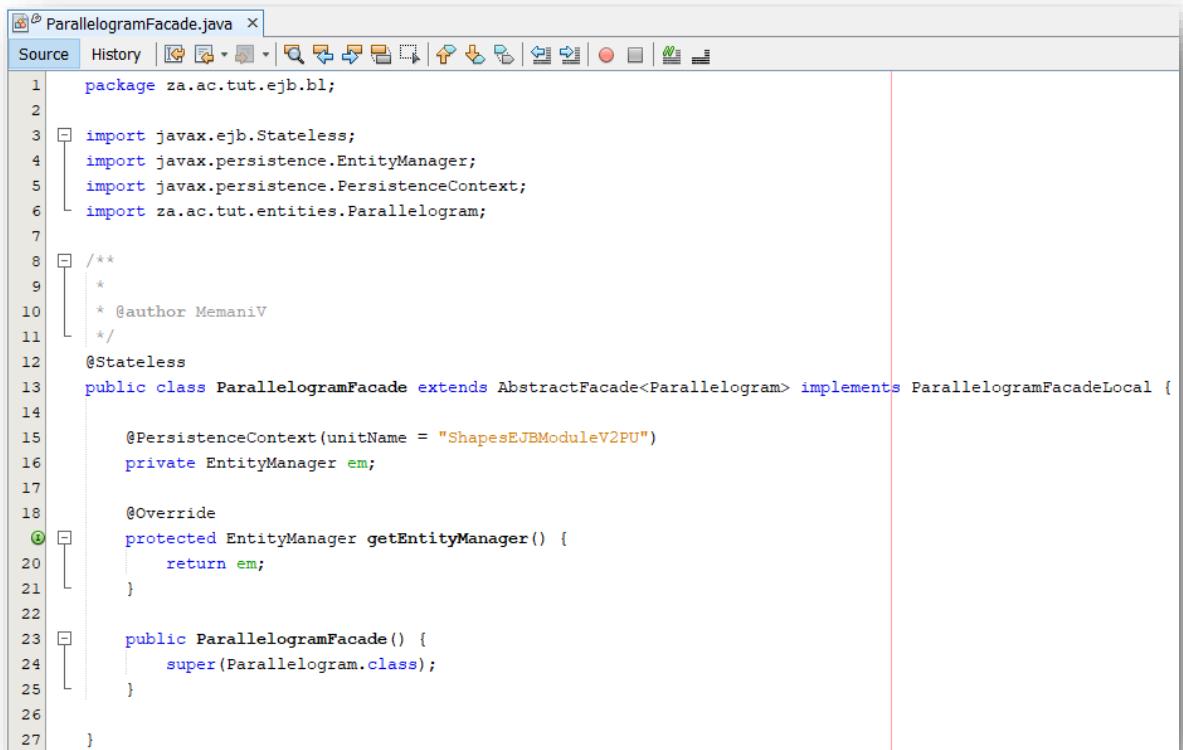
View the persistence.xml file.



```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence_2_1.xsd">
    <persistence-unit name="ShapesEJBModuleV2PU" transaction-type="JTA">
        <jta-data-source>jdbc/_default</jta-data-source>
        <exclude-unlisted-classes>false</exclude-unlisted-classes>
        <properties>
            <property name="javax.persistence.schema-generation.database.action" value="create"/>
        </properties>
    </persistence-unit>
</persistence>
```

Create business code (CRUD operations) for the entities

- **ParallelogramFacade.**



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

import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import za.ac.tut.entities.Parallelogram;

/**
 * 
 * @author MemaniV
 */
@Stateless
public class ParallelogramFacade extends AbstractFacade<Parallelogram> implements ParallelogramFacadeLocal {

    @PersistenceContext(unitName = "ShapesEJBModuleV2PU")
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    public ParallelogramFacade() {
        super(Parallelogram.class);
    }
}
```

▪ RectangleFacade

The screenshot shows a Java code editor window with the title "RectangleFacade.java". The code implements a Stateless EJB facade for a Rectangle entity. It imports javax.ejb.Stateless, javax.persistence.EntityManager, javax.persistence.PersistenceContext, and za.ac.tut.entities.Rectangle. The class is annotated with @Stateless and includes a constructor that calls super(Rectangle.class). It overrides the getEntityManager() method to return the private EntityManager em. A persistence context annotation (@PersistenceContext(unitName = "ShapesEJBModuleV2PU")) is present.

```
1 package za.ac.tut.ejb.bl;
2
3 import javax.ejb.Stateless;
4 import javax.persistence.EntityManager;
5 import javax.persistence.PersistenceContext;
6 import za.ac.tut.entities.Rectangle;
7
8 /**
9 *
10 * @author MemaniV
11 */
12 @Stateless
13 public class RectangleFacade extends AbstractFacade<Rectangle> implements RectangleFacadeLocal {
14
15     @PersistenceContext(unitName = "ShapesEJBModuleV2PU")
16     private EntityManager em;
17
18     @Override
19     protected EntityManager getEntityManager() {
20         return em;
21     }
22
23     public RectangleFacade() {
24         super(Rectangle.class);
25     }
26
27 }
```

▪ ShapeFacade

The screenshot shows a Java code editor window with the title "ShapeFacade.java". The code implements a Stateless EJB facade for a Shape entity. It imports javax.ejb.Stateless, javax.persistence.EntityManager, javax.persistence.PersistenceContext, and za.ac.tut.entities.Shape. The class is annotated with @Stateless and includes a constructor that calls super(Shape.class). It overrides the getEntityManager() method to return the private EntityManager em. A persistence context annotation (@PersistenceContext(unitName = "ShapesEJBModuleV2PU")) is present.

```
1 package za.ac.tut.ejb.bl;
2
3 import javax.ejb.Stateless;
4 import javax.persistence.EntityManager;
5 import javax.persistence.PersistenceContext;
6 import za.ac.tut.entities.Shape;
7
8 /**
9 *
10 * @author MemaniV
11 */
12 @Stateless
13 public class ShapeFacade extends AbstractFacade<Shape> implements ShapeFacadeLocal {
14
15     @PersistenceContext(unitName = "ShapesEJBModuleV2PU")
16     private EntityManager em;
17
18     @Override
19     protected EntityManager getEntityManager() {
20         return em;
21     }
22
23     public ShapeFacade() {
24         super(Shape.class);
25     }
26
27 }
```

▪ AbstractFacade.

AbstractFacade.java

```

1 package za.ac.tut.bl;
2
3 import java.util.List;
4 import javax.persistence.EntityManager;
5
6 /**
7 *
8 * @author MemaniV
9 */
10 public abstract class AbstractFacade<T> {
11
12     private Class<T> entityClass;
13
14     public AbstractFacade(Class<T> entityClass) {
15         this.entityClass = entityClass;
16     }
17
18     protected abstract EntityManager getEntityManager();
19
20     public void create(T entity) {
21         getEntityManager().persist(entity);
22     }
23
24     public void edit(T entity) {
25         getEntityManager().merge(entity);
26     }
27
28     public void remove(T entity) {
29         getEntityManager().remove(getEntityManager().merge(entity));
30     }
31

```

```

32     public T find(Object id) {
33         return getEntityManager().find(entityClass, id);
34     }
35
36     public List<T> findAll() {
37         javax.persistence.criteria.CriteriaQuery cq =
38             getEntityManager().getCriteriaBuilder().createQuery();
39         cq.select(cq.from(entityClass));
40         return getEntityManager().createQuery(cq).getResultList();
41     }
42
43     public List<T> findRange(int[] range) {
44         javax.persistence.criteria.CriteriaQuery cq =
45             getEntityManager().getCriteriaBuilder().createQuery();
46         cq.select(cq.from(entityClass));
47         javax.persistence.Query q = getEntityManager().createQuery(cq);
48         q.setMaxResults(range[1] - range[0] + 1);
49         q.setFirstResult(range[0]);
50         return q.getResultList();
51     }
52
53     public int count() {
54         javax.persistence.criteria.CriteriaQuery cq =
55             getEntityManager().getCriteriaBuilder().createQuery();
56         javax.persistence.criteria.Root<T> rt = cq.from(entityClass);
57         cq.select(getEntityManager().getCriteriaBuilder().count(rt));
58         javax.persistence.Query q = getEntityManager().createQuery(cq);
59         return ((Long) q.getSingleResult()).intValue();
60     }
61
62 }

```

- **ParallelogramFacadeLocal**

The screenshot shows a Java code editor window with the title "ParallelogramFacadeLocal.java". The code defines a local interface for managing parallelograms. It includes imports for java.util.List, javax.ejb.Local, and za.ac.tut.entities.Parallelogram. The interface is annotated with @Local and contains methods for creating, editing, removing, finding, and counting parallelograms.

```
1 package za.ac.tut.ejb.bl;
2
3 import java.util.List;
4 import javax.ejb.Local;
5 import za.ac.tut.entities.Parallelogram;
6
7 /**
8 * @author MemaniV
9 */
10
11 @Local
12 public interface ParallelogramFacadeLocal {
13
14     void create(Parallelogram parallelogram);
15
16     void edit(Parallelogram parallelogram);
17
18     void remove(Parallelogram parallelogram);
19
20     Parallelogram find(Object id);
21
22     List<Parallelogram> findAll();
23
24     List<Parallelogram> findRange(int[] range);
25
26     int count();
27
28 }
```

- **RectangleFacadeLocal.**

The screenshot shows a Java code editor window with the title "RectangleFacadeLocal.java". The code defines a local interface for managing rectangles. It includes imports for java.util.List, javax.ejb.Local, and za.ac.tut.entities.Rectangle. The interface is annotated with @Local and contains methods for creating, editing, removing, finding, and counting rectangles.

```
1 package za.ac.tut.ejb.bl;
2
3 import java.util.List;
4 import javax.ejb.Local;
5 import za.ac.tut.entities.Rectangle;
6
7 /**
8 * @author MemaniV
9 */
10
11 @Local
12 public interface RectangleFacadeLocal {
13
14     void create(Rectangle rectangle);
15
16     void edit(Rectangle rectangle);
17
18     void remove(Rectangle rectangle);
19
20     Rectangle find(Object id);
21
22     List<Rectangle> findAll();
23
24     List<Rectangle> findRange(int[] range);
25
26     int count();
27
28 }
```

- **ShapeFacadeLocal.**

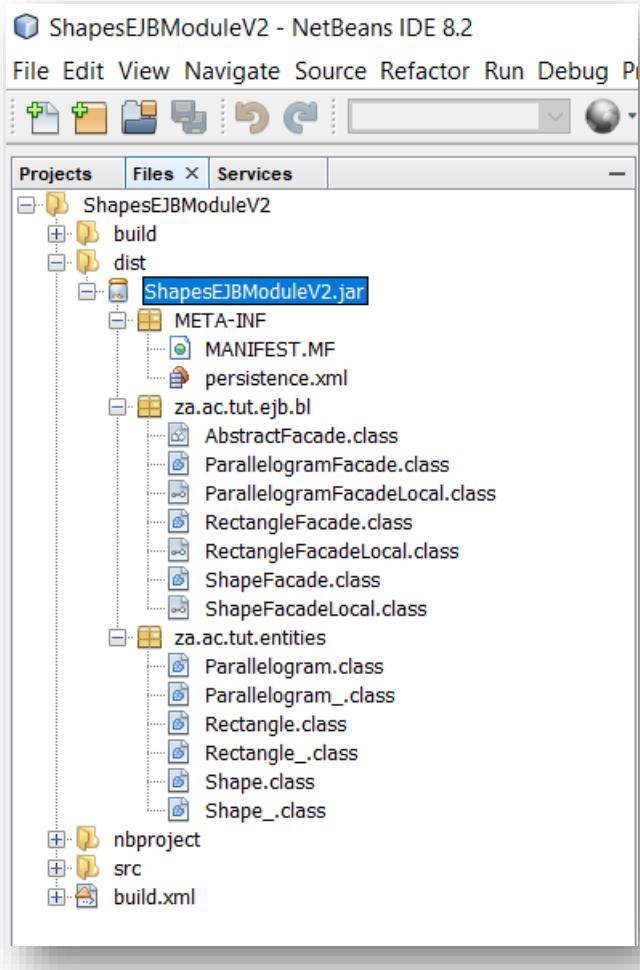
The screenshot shows the NetBeans IDE interface with the code editor open. The file is named ShapeFacadeLocal.java. The code defines a public interface for managing shapes. It includes imports for java.util.List, javax.ejb.Local, and za.ac.tut.entities.Shape. The interface is annotated with @Local. It contains methods for creating, editing, removing shapes, finding a single shape by ID, finding all shapes, finding shapes within a range, and counting the total number of shapes.

```
1 package za.ac.tut.ejb.bl;
2
3 import java.util.List;
4 import javax.ejb.Local;
5 import za.ac.tut.entities.Shape;
6
7 /**
8 * ...
9 * @author MemaniV
10 */
11 @Local
12 public interface ShapeFacadeLocal {
13
14     void create(Shape shape);
15
16     void edit(Shape shape);
17
18     void remove(Shape shape);
19
20     Shape find(Object id);
21
22     List<Shape> findAll();
23
24     List<Shape> findRange(int[] range);
25
26     int count();
27
28 }
```

- ✓ Clean and build the EJB project.

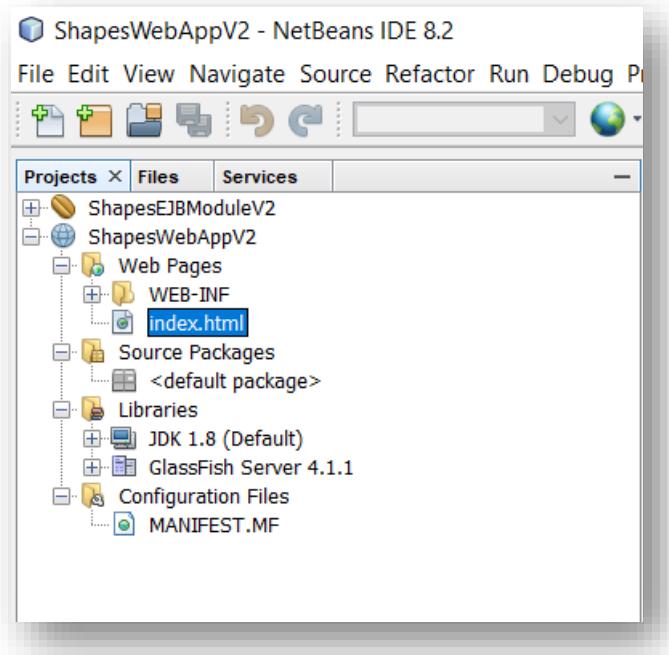
The screenshot shows the NetBeans IDE Output window. It displays the build logs for the ShapesEJBModuleV2 project. The logs show the compilation process, including the creation of a dist directory, the building of a jar file, and the final successful build message.

```
Java DB Database Process X GlassFish Server 4.1.1 X ShapesEJBModuleV2 (clean,dist) X
Note: C:\Users\memaniv\Documents\NetBeansProjects\ShapesEJBModuleV2\src\java\za\ac\tut\ejb\bl\AbstractFa
Note: Recompile with -Xlint:unchecked for details.
1 warning
compile:
library-inclusion-in-archive:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\ShapesEJBModuleV2\dist
Building jar: C:\Users\memaniv\Documents\NetBeansProjects\ShapesEJBModuleV2\dist\ShapesEJBModuleV2.jar
dist:
BUILD SUCCESSFUL (total time: 1 second)
```



Part D – Create a web client project.

Create a web client project called **ShapesWebAppV2**.

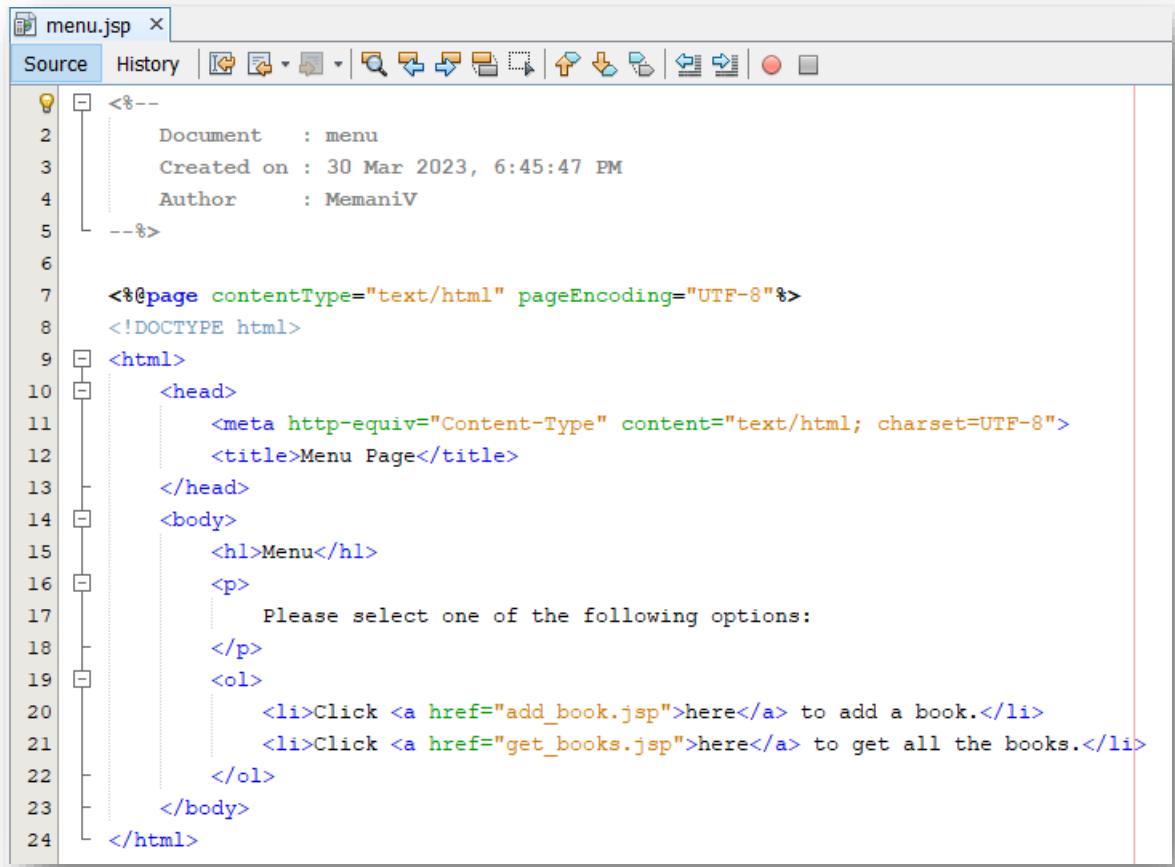


Edit the **index.html** page.

A screenshot of the NetBeans code editor showing the "index.html" file. The tab bar at the top says "index.html". The editor has a toolbar with various icons for file operations. The code is displayed in a source editor with line numbers from 1 to 19 on the left. The content of the file is:

```
<!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 page</h1>
        <p>
            Welcome to our web app. Please click <a href="menu.jsp">here</a> to start.
        </p>
    </body>
</html>
```

Create the **menu.jsp** file.



The screenshot shows a Java IDE interface with the menu.jsp file open in the Source view. The code is a JSP page that displays a menu. It includes a header with document information, an HTML structure with a title and body, and a list of options for adding a book or getting all books.

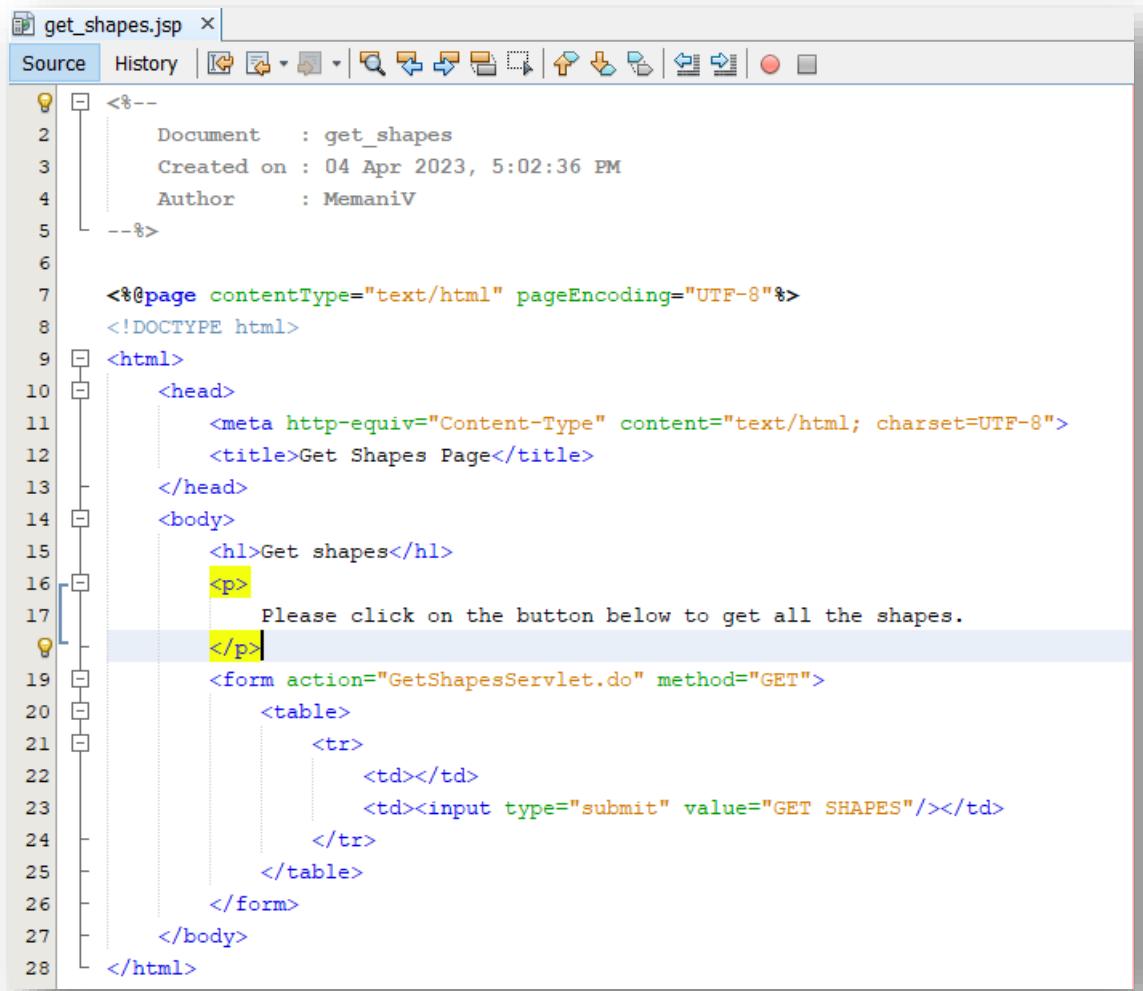
```
<%-->
Document : menu
Created on : 30 Mar 2023, 6:45:47 PM
Author : MemaniV
--%>

<%@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="add_book.jsp">here</a> to add a book.</li>
            <li>Click <a href="get_books.jsp">here</a> to get all the books.</li>
        </ol>
    </body>
</html>
```

Create **add_shape.jsp** file.

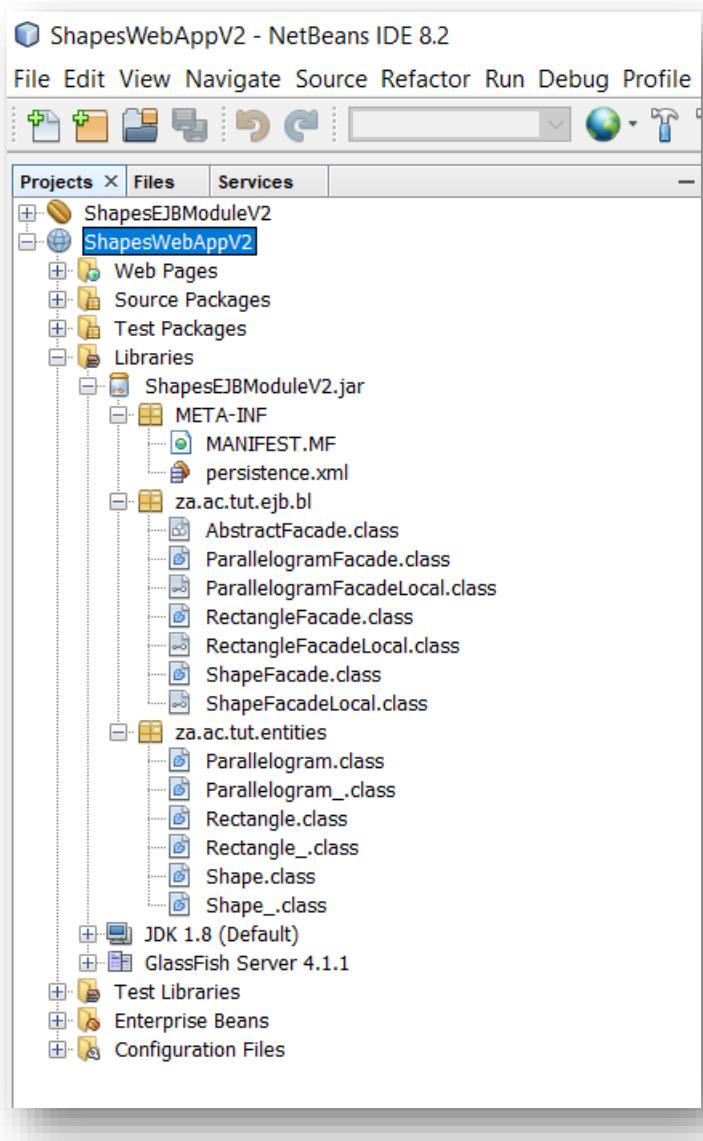
```
<%--  
    Document      : add_shape  
    Created on   : 04 Apr 2023, 2:03:42 PM  
    Author        : MemaniV  
--%>  
  
<%@page contentType="text/html" pageEncoding="UTF-8"%>  
<!DOCTYPE html>  
<html>  
    <head>  
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
        <title>Add Shape Page</title>  
    </head>  
    <body>  
        <h1>Add shape</h1>  
        <p>  
            Please add shape details below:  
        </p>  
  
<form action="AddShapeServlet.do" method="POST">  
    <table>  
        <tr>  
            <td>Type: </td>  
            <td>  
                <select name="type">  
                    <option value="P">Parallelogram</option>  
                    <option value="R">Rectangle</option>  
                </select>  
            </td>  
        </tr>  
        <tr>  
            <td>Side 1: </td>  
            <td><input type="text" name="side1"/></td>  
        </tr>  
        <tr>  
            <td>Side 2: </td>  
            <td><input type="text" name="side2"/></td>  
        </tr>  
        <tr>  
            <td>Is a big shape? </td>  
            <td>  
                <select name="isBig">  
                    <option value="True">Yes</option>  
                    <option value="False">No</option>  
                </select>  
            </td>  
        </tr>  
        <tr>  
            <td></td>  
            <td><input type="submit" value="ADD SHAPE"/></td>  
        </tr>  
    </table>  
    </form>  
</body>  
</html>
```

Create the get_shapes.jsp file.

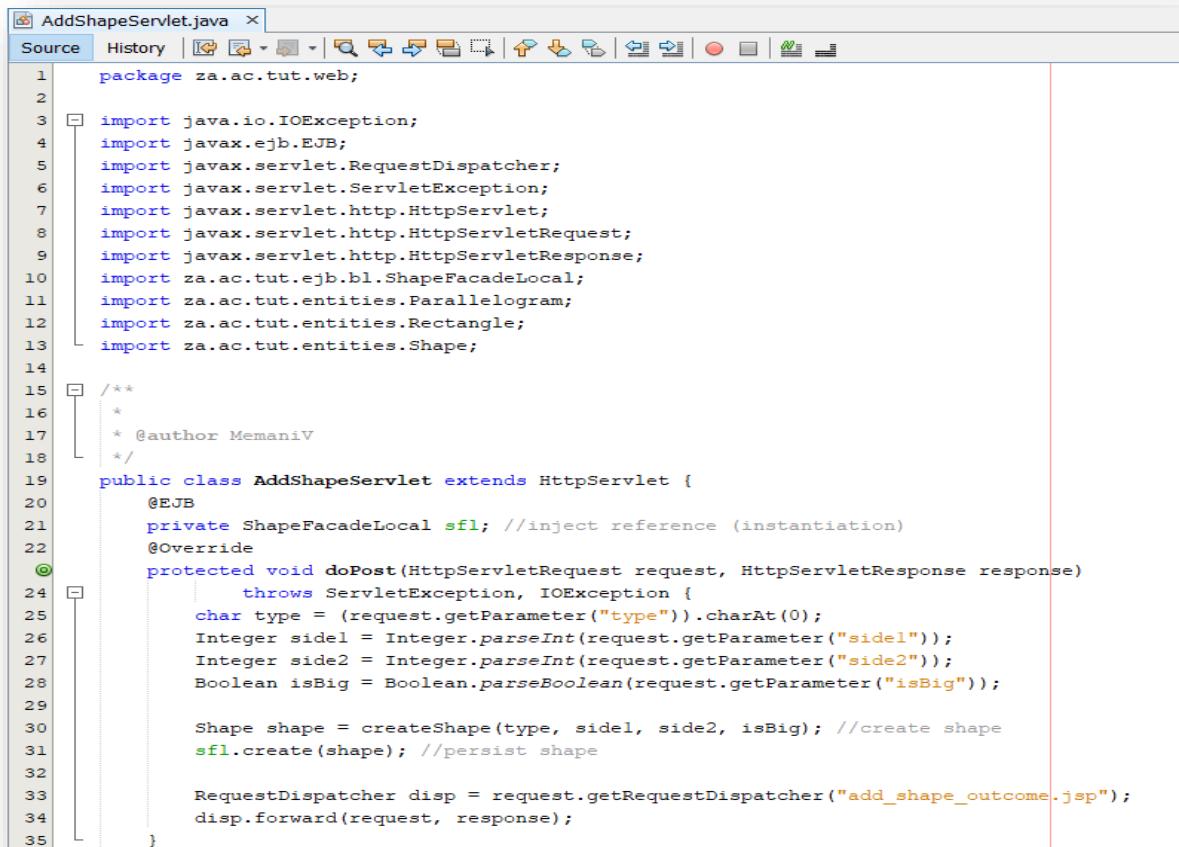


```
<%--  
1 Document      : get_shapes  
2 Created on   : 04 Apr 2023, 5:02:36 PM  
3 Author        : MemaniV  
4--%>  
  
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>  
8 <!DOCTYPE html>  
9 <html>  
10 <head>  
11     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
12     <title>Get Shapes Page</title>  
13 </head>  
14 <body>  
15     <h1>Get shapes</h1>  
16     <p>  
17         Please click on the button below to get all the shapes.  
18     </p>  
19     <form action="GetShapesServlet.do" method="GET">  
20         <table>  
21             <tr>  
22                 <td></td>  
23                 <td><input type="submit" value="GET SHAPES"/></td>  
24             </tr>  
25         </table>  
26     </form>  
27 </body>  
28 </html>
```

Add the **ShapesEJBModuleV2.jar** file to the library of **ShapesWebApp**.



Create the AddShapeServlet.java file.

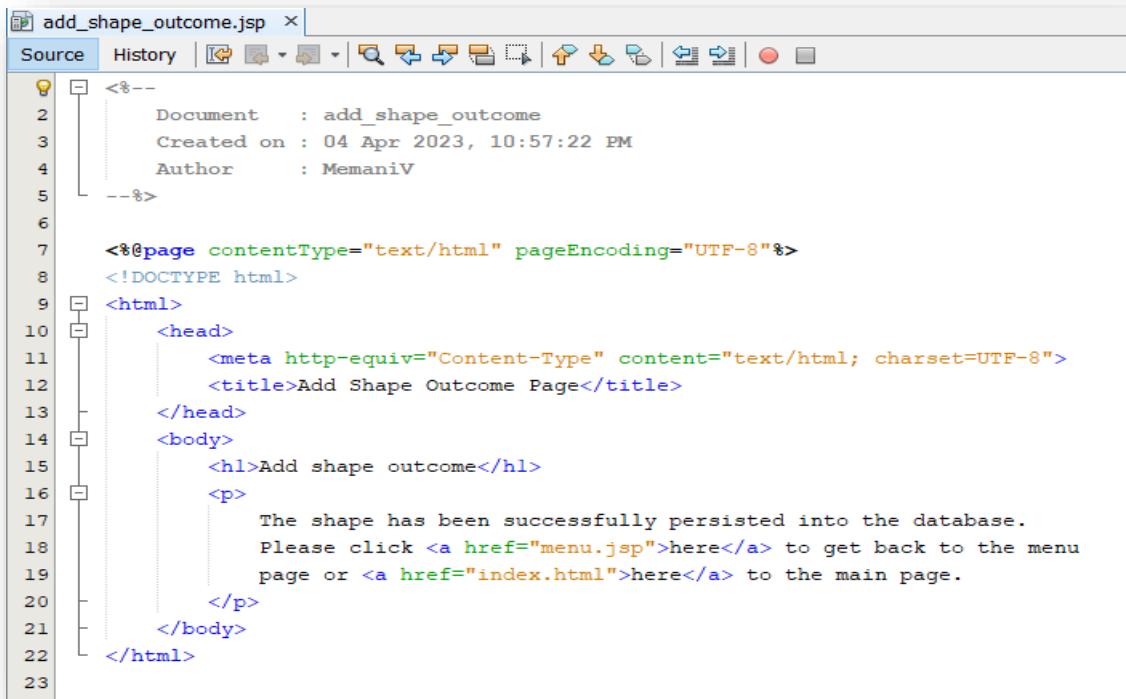


```
1 package za.ac.tut.web;
2
3 import java.io.IOException;
4 import javax.ejb.EJB;
5 import javax.servlet.RequestDispatcher;
6 import javax.servlet.ServletException;
7 import javax.servlet.http.HttpServlet;
8 import javax.servlet.http.HttpServletRequest;
9 import javax.servlet.http.HttpServletResponse;
10 import za.ac.tut.ejb.bl.ShapeFacadeLocal;
11 import za.ac.tut.entities.Parallelogram;
12 import za.ac.tut.entities.Rectangle;
13 import za.ac.tut.entities.Shape;
14
15 /**
16 *
17 * @author MemaniV
18 */
19 public class AddShapeServlet extends HttpServlet {
20     @EJB
21     private ShapeFacadeLocal sfl; //inject reference (instantiation)
22     @Override
23     protected void doPost(HttpServletRequest request, HttpServletResponse response)
24         throws ServletException, IOException {
25         char type = (request.getParameter("type")).charAt(0);
26         Integer sidel = Integer.parseInt(request.getParameter("sidel"));
27         Integer side2 = Integer.parseInt(request.getParameter("side2"));
28         Boolean isBig = Boolean.parseBoolean(request.getParameter("isBig"));
29
30         Shape shape = createShape(type, sidel, side2, isBig); //create shape
31         sfl.create(shape); //persist shape
32
33         RequestDispatcher disp = request.getRequestDispatcher("add_shape_outcome.jsp");
34         disp.forward(request, response);
35     }
}
```



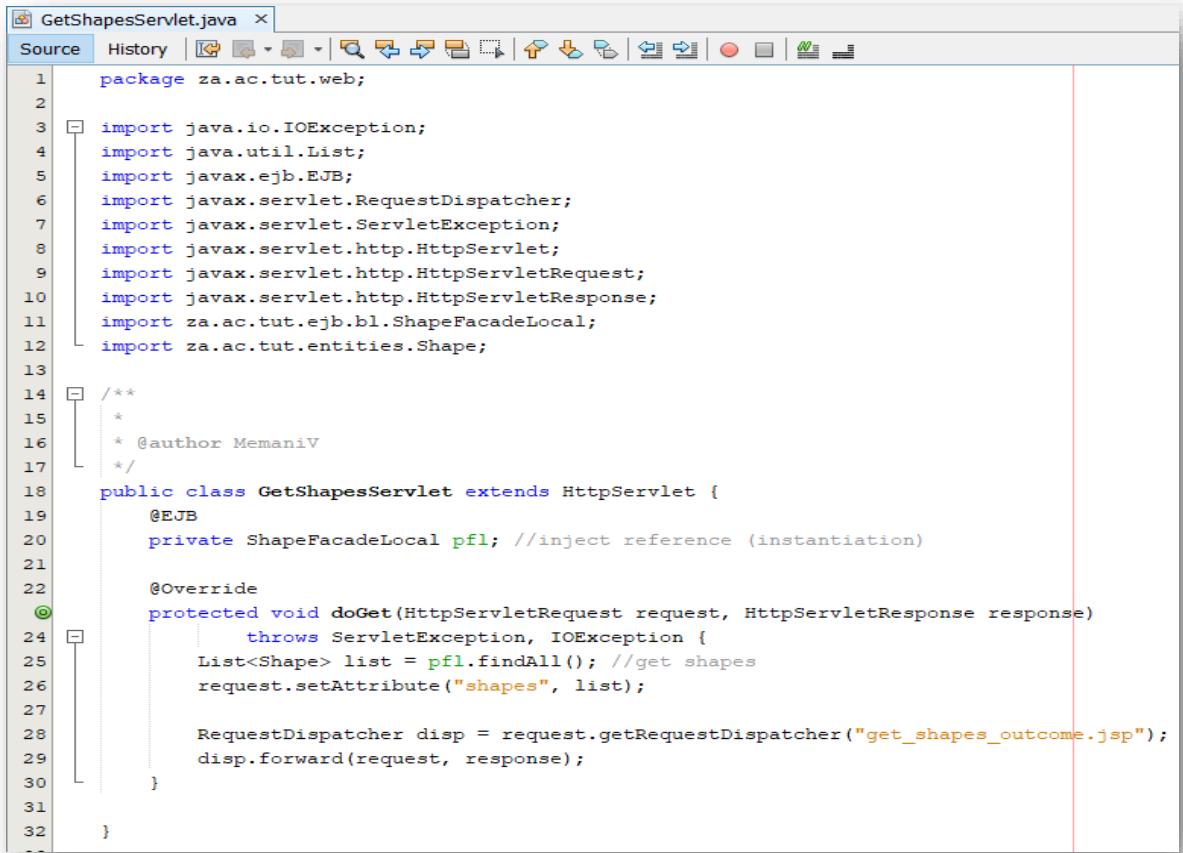
```
37     private Shape createShape(char type, Integer sidel, Integer side2, Boolean isBig) {
38         Shape newShape;
39         switch(type){
40             case 'P':
41             case 'p':
42                 newShape = new Parallelogram();
43                 break;
44             default:
45                 newShape = new Rectangle();
46             }
47
48             newShape.setIsBig(isBig);
49
50             if(newShape instanceof Rectangle){
51                 ((Rectangle) newShape).setRecLength(sidel);
52                 ((Rectangle) newShape).setRecWidth(side2);
53             } else {
54                 ((Parallelogram) newShape).setBase(sidel);
55                 ((Parallelogram) newShape).setHeight(side2);
56             }
57
58             return newShape;
59         }
60     }
61 }
```

Create **add_shape_outcome.jsp** file.



```
<%--  
    Document      : add_shape_outcome  
    Created on   : 04 Apr 2023, 10:57:22 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>Add Shape Outcome Page</title>  
    </head>  
    <body>  
        <h1>Add shape outcome</h1>  
        <p>  
            The shape has been successfully persisted into the database.  
            Please click <a href="menu.jsp">here</a> to get back to the menu  
            page or <a href="index.html">here</a> to the main page.  
        </p>  
    </body>  
</html>
```

Create **GetShapesServlet.java** file.



```
package za.ac.tut.web;  
  
import java.io.IOException;  
import java.util.List;  
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.ShapeFacadeLocal;  
import za.ac.tut.entities.Shape;  
  
/*  
 * @author MemaniV  
 */  
public class GetShapesServlet extends HttpServlet {  
    @EJB  
    private ShapeFacadeLocal pfl; //inject reference (instantiation)  
  
    @Override  
    protected void doGet(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
        List<Shape> list = pfl.findAll(); //get shapes  
        request.setAttribute("shapes", list);  
  
        RequestDispatcher disp = request.getRequestDispatcher("get_shapes_outcome.jsp");  
        disp.forward(request, response);  
    }  
}
```

Create get_shapes_outcome.jsp file.

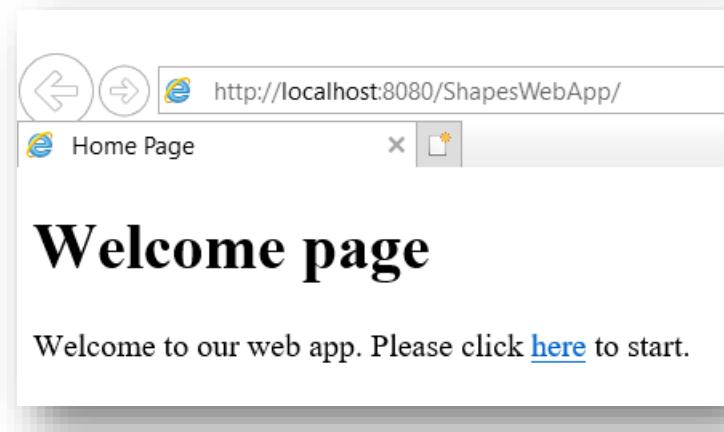
```
<%--  
1 Document : get_shapes_outcome  
2 Created on : 04 Apr 2023, 10:58:00 PM  
3 Author : MemaniV  
4 --%>  
5  
6  
7 <%@page import="za.ac.tut.entities.Parallelogram"%>  
8 <%@page import="za.ac.tut.entities.Rectangle"%>  
9 <%@page import="java.util.List"%>  
10 <%@page import="za.ac.tut.entities.Shape"%>  
11 <%@page contentType="text/html" pageEncoding="UTF-8"%>  
12 <!DOCTYPE html>  
13 <html>  
14 <head>  
15 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
16 <title>Get Shapes Outcome Page</title>  
17 </head>  
18 <body>  
19 <h1>Get shapes outcome</h1>  
20 <%  
21 List<Shape> shapes = (List<Shape>)request.getAttribute("shapes");  
22 %>  
23 <p>  
24 Below is the list of shapes retrieved from the database.  
25 </p>
```

```
26 <table>  
27 <%  
28 String shapeType = "Parallelogram";  
29 Integer sidel, side2;  
30 for(int i = 0; i < shapes.size(); i++){  
31 Shape p = shapes.get(i);  
32 Long id = p.getId();  
33 Boolean isBig = p.getIsBig();  
34  
35 if(p instanceof Rectangle){  
36 shapeType = "Rectangle";  
37 Rectangle rec = (Rectangle)p;  
38 sidel = rec.getRecLength();  
39 side2 = rec.getRecWidth();  
40 } else {  
41 Parallelogram par = (Parallelogram)p;  
42 sidel = par.getBase();  
43 side2 = par.getHeight();  
44 }  
45 %>
```

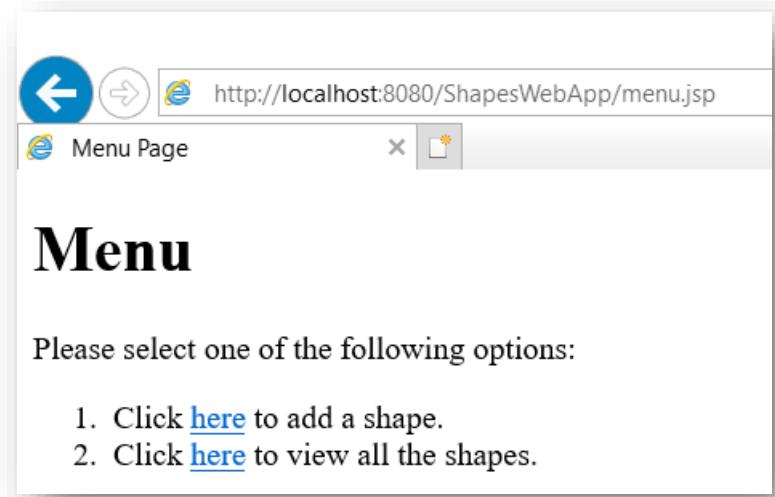
```
46 |     <tr>
47 |         <td>ID: </td>
48 |         <td><%=id%></td>
49 |     </tr>
50 |     <tr>
51 |         <td>Shape type: </td>
52 |         <td><%=shapeType%></td>
53 |     </tr>
54 |     <tr>
55 |         <td>Side 1: </td>
56 |         <td><%=side1%></td>
57 |     </tr>
58 |     <tr>
59 |         <td>Side2: </td>
60 |         <td><%=side2%></td>
61 |     </tr>
62 |     <tr>
63 |         <td>Is the shape big? </td>
64 |         <td><%=isBig%></td>
65 |     </tr>
66 |     <%
67 |         }
68 |     %>
69 |     </table>
70 |     <p>
71 |         Please click <a href="menu.jsp">here</a> to get back to the menu page or <a href="index.html">here</a>
72 |         to the main page.
73 |     </p>
74 |     </body>
75 | </html>
```

Part E – Run the web application.

Launch the application.



Click on the link



1. Click [here](#) to add a shape.
2. Click [here](#) to view all the shapes.

8.5.3 Table per concrete class strategy

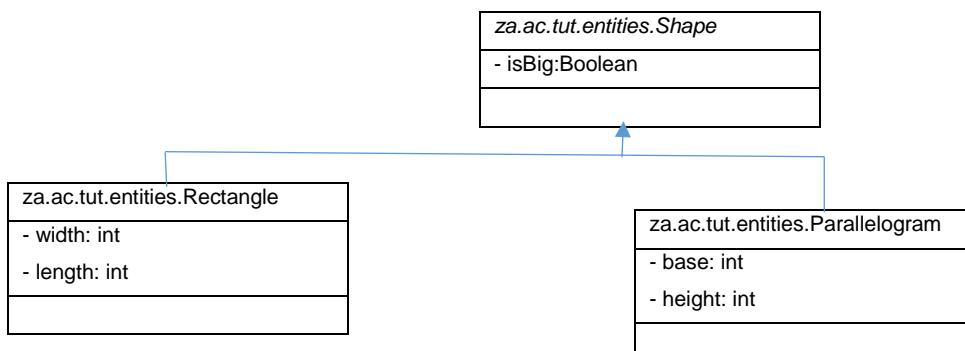
When using this strategy, each entity in the hierarchy is mapped to its own table. This is similar to the Joined-subclass strategy. The only difference is that the attributes of the superclass will be repeated in the tables of the subclasses.

To define a Joined-subclass strategy, we use the **InheritanceType.TABLE_PER_CLASS** value inside the **@Inheritance** annotation. For example we have something as follows:

```
@Entity  
 @Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)  
 public class Item {  
 }
```

Example

Create a web application to represent a Shape. We have two shapes, a Parallelogram and a Rectangle. A Parallelogram has a **base** and a **height**, and a Rectangle has a **length** and a **width**. The area of Parallelogram can be calculated as **base * height** whilst that of a Rectangle can be calculated as **length * width**. The inheritance class hierarchy can be represented as follows:



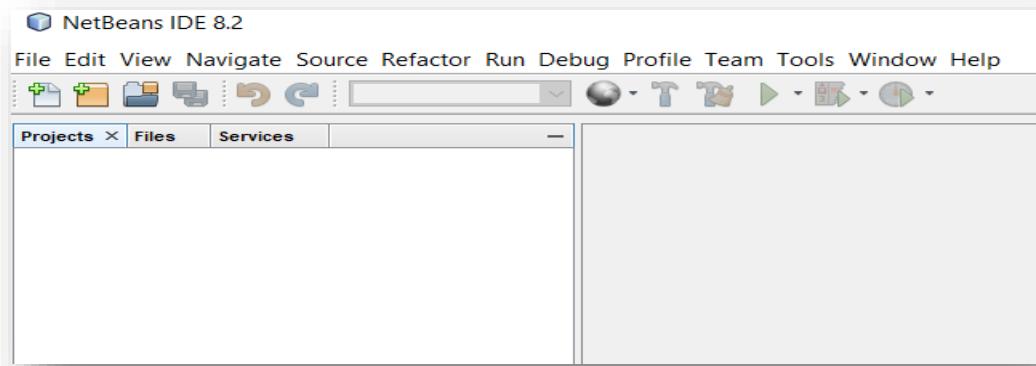
Solution approach

The solution to this problem is going to be done in five parts, namely:

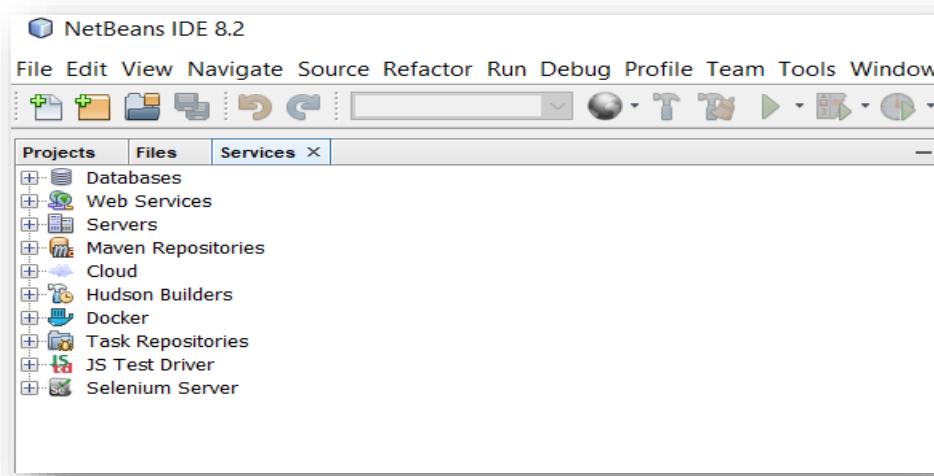
- **Part A:** we create a database.
- **Part B:** we establish connection between the created database and GlassFish.
- **Part C:** we create an EJB module. The module will have an entity with business logic.
- **Part D:** we create a client web application.
- **Part E:** we run the client web application.

Part A - Create a database

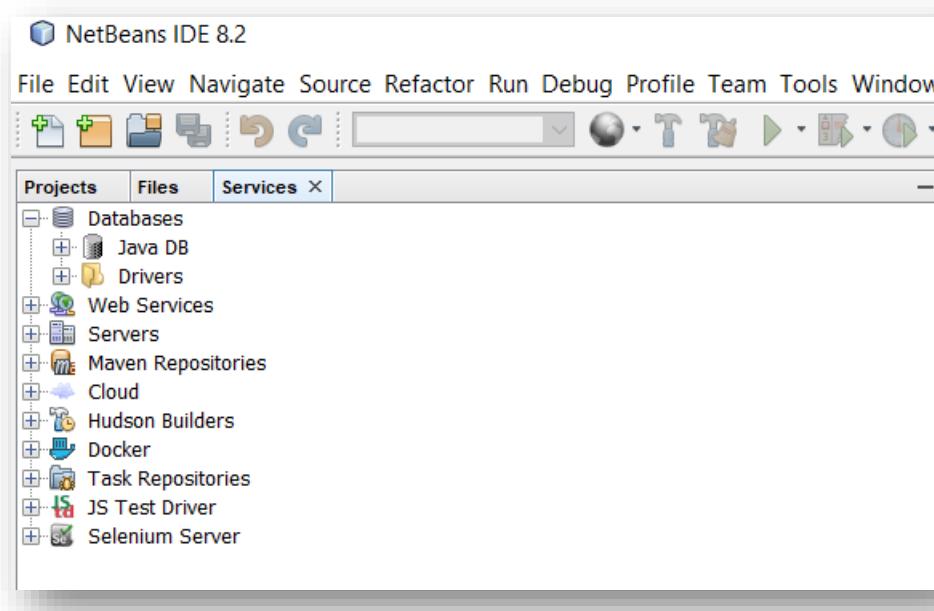
Launch NetBeans.



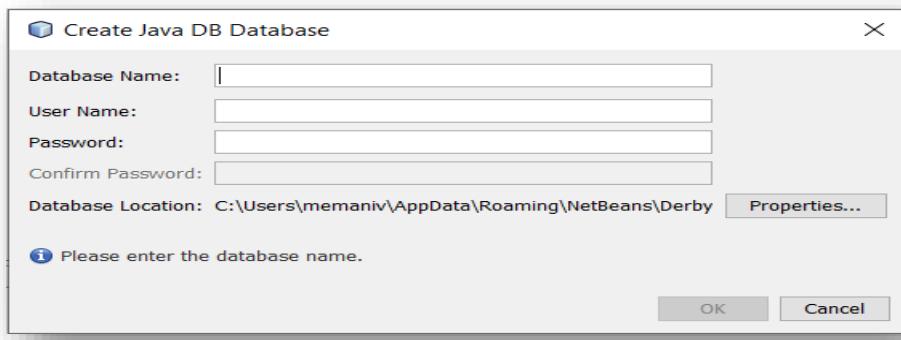
Click on the **Services** tab.



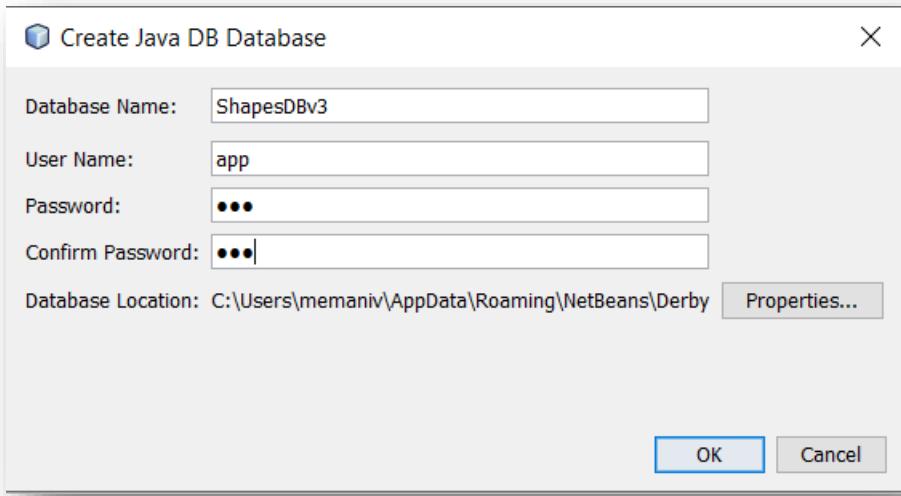
Expand the **Databases** option.



Right-click on **Java DB** and select **Create Database**.



Fill-in the form. I made the password to be **123**.

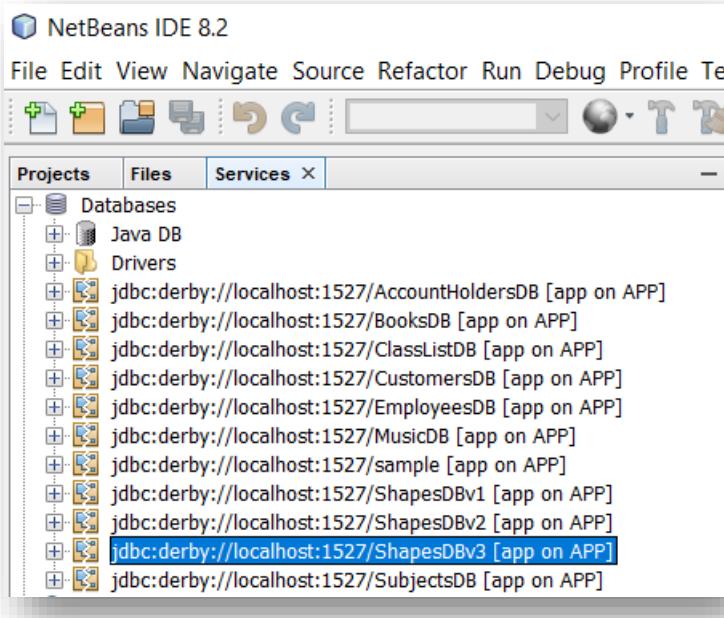


Click **OK**.

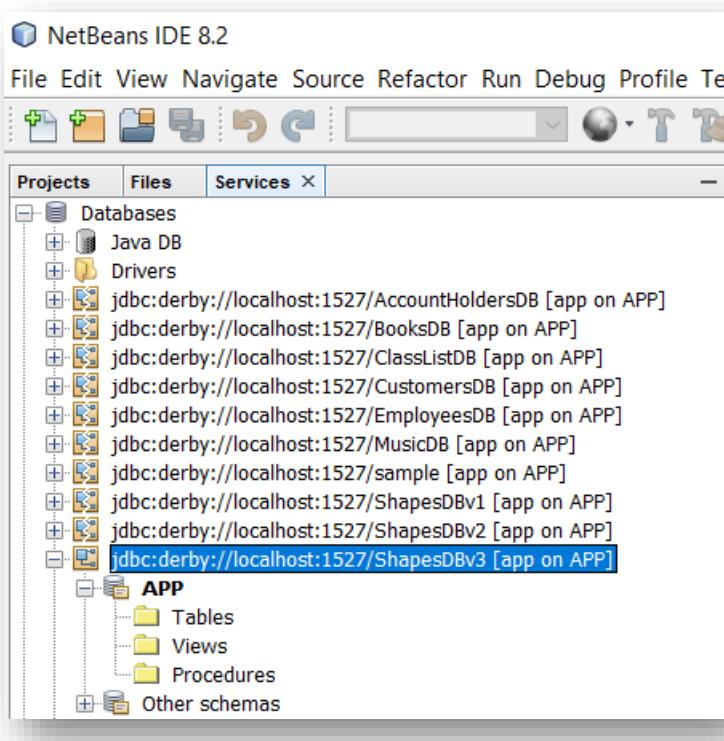
- ✓ An output message confirming that **Derby** has started and ready to accept connections on port 1527 is shown.

```
Output - Java DB Database Process X
Tue Apr 04 12:45:46 CAT 2023 : Security manager installed using the Basic server security policy.
Tue Apr 04 12:45:47 CAT 2023 : Apache Derby Network Server - 10.11.1.2 - (1629631) started and ready to accept connections on port 1527
```

- ✓ A database is created.

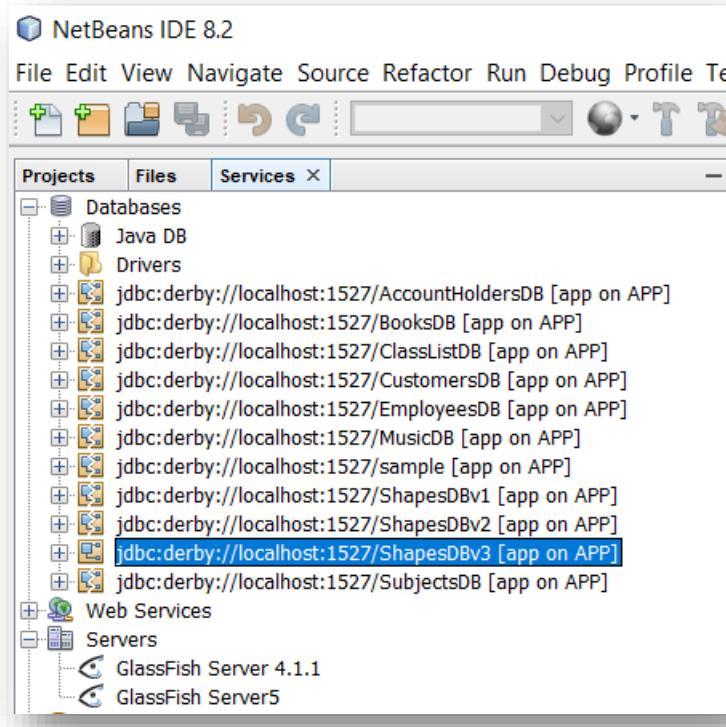


Right-click on the database and select **Connect**.

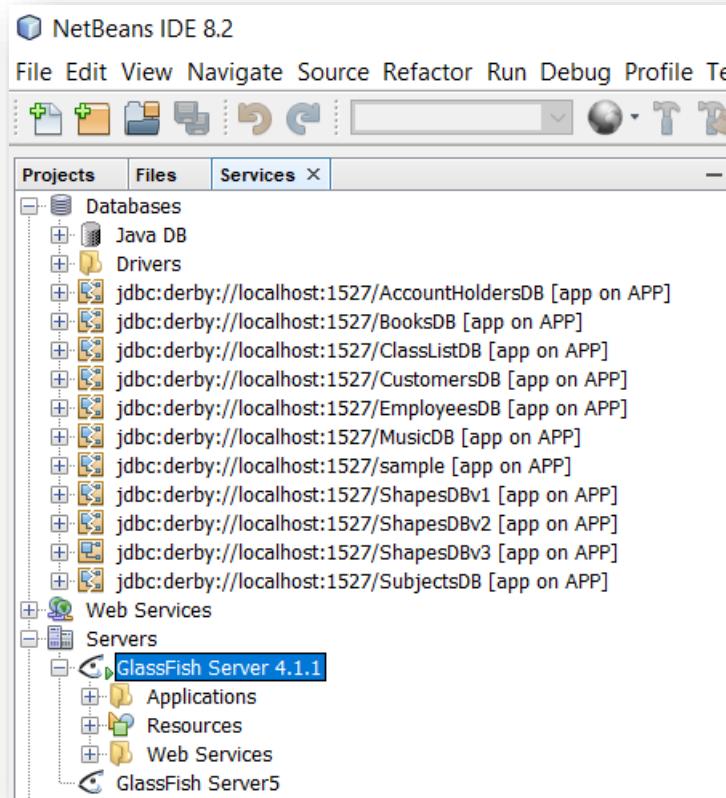


Part B – Connect the database to the application server.

Click on the **Services** tab.



Start the server.



Right-click on the server and select **View Domain Admin Console**.

The screenshot shows the GlassFish Server Open Source Edition Common Tasks console. The URL in the address bar is `http://localhost:4848/common/index.jsf`. The title bar says "GlassFish Console - Comm...". The top menu has "Home" and "About..." buttons. Below that, it shows "User: admin | Domain: domain1 | Server: localhost". The main title is "GlassFish™ Server Open Source Edition". The left sidebar is titled "Common Tasks" and contains the following items:

- Domain
- server (Admin Server)
- Clusters
- Standalone Instances
- Nodes
- Applications
- Lifecycle Modules
- Monitoring Data
- Resources
 - Concurrent Resources
 - Connectors
 - JDBC
 - JMS Resources
 - JNDI
 - JavaMail Sessions
 - Resource Adapter Configs
- Configurations
 - default-config
 - server-config
- Update Tool

The right panel is titled "GlassFish Console - Common Tasks" and contains sections for "GlassFish News" (with a "GlassFish News" button), "Deployment" (with "List Deployed Applications" and "Deploy an Application" buttons), "Administration" (with "Change Administrator Password" and "List Password Aliases" buttons), and "Monitoring" (with a "Monitoring Data" button).

Under **Common Tasks** panel, expand **JDBC**.

The screenshot shows the GlassFish Server Open Source Edition JDBC console. The URL in the address bar is `http://localhost:4848/common/index.jsf`. The title bar says "JDBC". The top menu has "Home" and "About..." buttons. Below that, it shows "User: admin | Domain: domain1 | Server: localhost". The main title is "GlassFish™ Server Open Source Edition". The left sidebar is titled "Common Tasks" and contains the same items as the previous screenshot, with the "Resources" item expanded. Under "Resources", the "JDBC" item is selected and highlighted with a blue bar. The right panel is titled "JDBC" and contains two buttons: "JDBC Resources" and "JDBC Connection Pools".

Modify the Connection pool to include the **ClassListDB**.

- ✓ Click on **JDBC Connection Pools**.

JDBC Connection Pools					
To store, organize, and retrieve data, most applications use relational databases. Java EE applications access relational databases through the JDBC API. Before an application can access a database, it must get a connection.					
Pools (3)					
Select	Pool Name	Resource Type	Classname	Description	
<input type="checkbox"/>	DerbyPool	javax.sql.DataSource	org.apache.derby.jdbc.ClientDataSource		
<input type="checkbox"/>	SamplePool	javax.sql.DataSource	org.apache.derby.jdbc.ClientDataSource		
<input type="checkbox"/>	TimerPool	javax.sql.XADatasource	org.apache.derby.jdbc.EmbeddedXADataSource		

- ✓ Click on the **DerbyPool** link.

General	Advanced	Additional Properties
Edit JDBC Connection Pool Modify an existing JDBC connection pool. A JDBC connection pool is a group of reusable connections for a particular database. <input type="button" value="Load Defaults"/> <input type="button" value="Flush"/> <input type="button" value="Ping"/>		
General Settings		
Pool Name:	DerbyPool	
Resource Type:	javax.sql.DataSource	
Must be specified if the datasource class implements more than 1 of the interface.		
Datasource Classname:	org.apache.derby.jdbc.ClientDataSource	
Vendor-specific classname that implements the DataSource and/or XADataSource APIs		
Driver Classname:		
Vendor-specific classname that implements the java.sql.Driver interface.		
Ping:	<input type="checkbox"/> Enabled	
When enabled, the pool is pinged during creation or reconfiguration to identify and warn of any erroneous values for its attributes		
Deployment Order:	100	
Specifies the loading order of the resource at server startup. Lower numbers are loaded first.		
Description:		
Pool Settings		
Initial and Minimum Pool Size:	8	Connections
Minimum and initial number of connections maintained in the pool		
Maximum Pool Size:	32	Connections
Maximum number of connections that can be created to satisfy client requests		
Pool Resize Quantity:	2	Connections

- ✓ Click on **Additional Properties**.

General	Advanced	Additional Properties
Edit JDBC Connection Pool Properties Modify properties of an existing JDBC connection pool.		
Pool Name: DerbyPool		
Additional Properties (6)		
Select	Name	Value
<input type="checkbox"/>	PortNumber	1527
<input type="checkbox"/>	Password	APP
<input type="checkbox"/>	User	APP
<input type="checkbox"/>	serverName	localhost
<input type="checkbox"/>	DatabaseName	sun-appserv-samples
<input type="checkbox"/>	connectionAttributes	;create=true

- ✓ Click on **Add Property** and add a **url** property with the following value:

jdbc:derby://localhost:1527/ShapesDBv3

- ✓ Delete the **connectionAttributes** property.

Select	Name	Value
<input type="checkbox"/>	PortNumber	1527
<input type="checkbox"/>	Password	123
<input type="checkbox"/>	User	app
<input type="checkbox"/>	serverName	localhost
<input type="checkbox"/>	DatabaseName	ShapesDBv3
<input type="checkbox"/>	url	jdbc:derby://localhost:1527/ShapesDBv3

- ✓ Save the changes by clicking the **Save** button.

Select	Name	Value
<input type="checkbox"/>	PortNumber	1527
<input type="checkbox"/>	Password	123
<input type="checkbox"/>	User	app
<input type="checkbox"/>	serverName	localhost
<input type="checkbox"/>	DatabaseName	ShapesDBv3
<input type="checkbox"/>	url	jdbc:derby://localhost:1527/ShapesDBv3

- ✓ Check if the created connection pool is working. Do this by clicking the **Ping** button.

Confirm that the resource points to the DerbyPool.

- ✓ Still under **JDBC**, click on **JDBC Resources**.

Select	JNDI Name	Logical JNDI Name	Enabled	Connection Pool
<input type="checkbox"/>	jdbc/_TimerPool	__TimerPool	<input checked="" type="checkbox"/>	__TimerPool
<input type="checkbox"/>	jdbc/_default	java:comp/DefaultDataSource	<input checked="" type="checkbox"/>	DerbyPool
<input type="checkbox"/>	jdbc/sample	sample	<input checked="" type="checkbox"/>	SamplePool

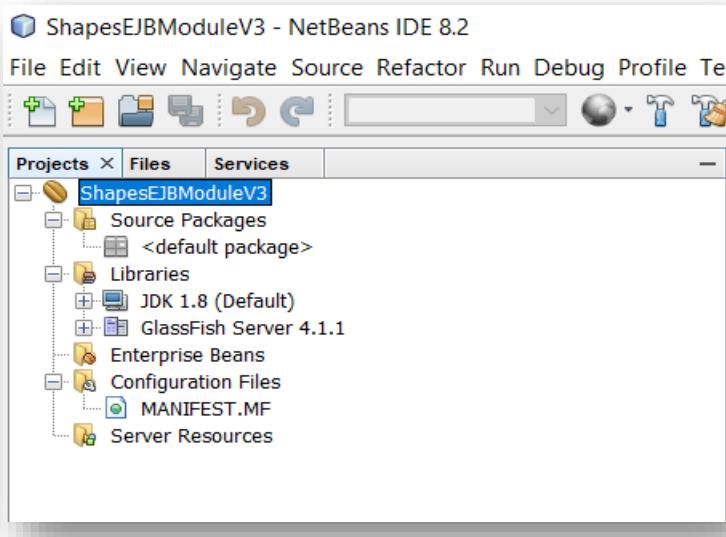
Take note of the purpose of JDBC Resources. It is said:

"JDBC resources provide applications with a means to connect to a database."

You can also see that the **jdbc/_default** resource is associated with the **DerbyPool** Connection Pool.

Part C – Create an entity.

Create an EJB project called **ShapesEJBModuleV3**.



Create the **Shape** entity. Use the **Table per concrete class strategy**.

A screenshot of the NetBeans code editor for the "Shape.java" file. The code implements the Serializable interface and uses annotations for persistence. It defines a class "Shape" with an @Entity annotation and specifies the inheritance strategy as TABLE_PER_CLASS. The class has an @Id annotation and a private attribute "id". It also has a private attribute "isBig" and two constructors: one empty and one taking a Boolean value. The code editor shows syntax highlighting for Java and annotations, with some annotations like TABLE_PER_CLASS and CHAR highlighted in green.

```
37  □    public Boolean getIsBig() {
38      return isBig;
39  }
40
41  □    public void setIsBig(Boolean isBig) {
42        this.isBig = isBig;
43    }
44
45  □    public Long getId() {
46        return id;
47    }
48
49  □    public void setId(Long id) {
50        this.id = id;
51    }
52
53    @Override
54    □    public int hashCode() {
55        int hash = 0;
56        hash += (id != null ? id.hashCode() : 0);
57        return hash;
58    }
```

```
60    @Override
61    □    public boolean equals(Object object) {
62        if (!(object instanceof Shape)) {
63            return false;
64        }
65        Shape other = (Shape) object;
66        if ((this.id == null && other.id != null) ||
67            (this.id != null && !this.id.equals(other.id))) {
68            return false;
69        }
70        return true;
71    }
72
73    @Override
74    □    public String toString() {
75        return "za.ac.tut.entities.Shape[ id=" + id + " ]";
76    }
77
78 }
```

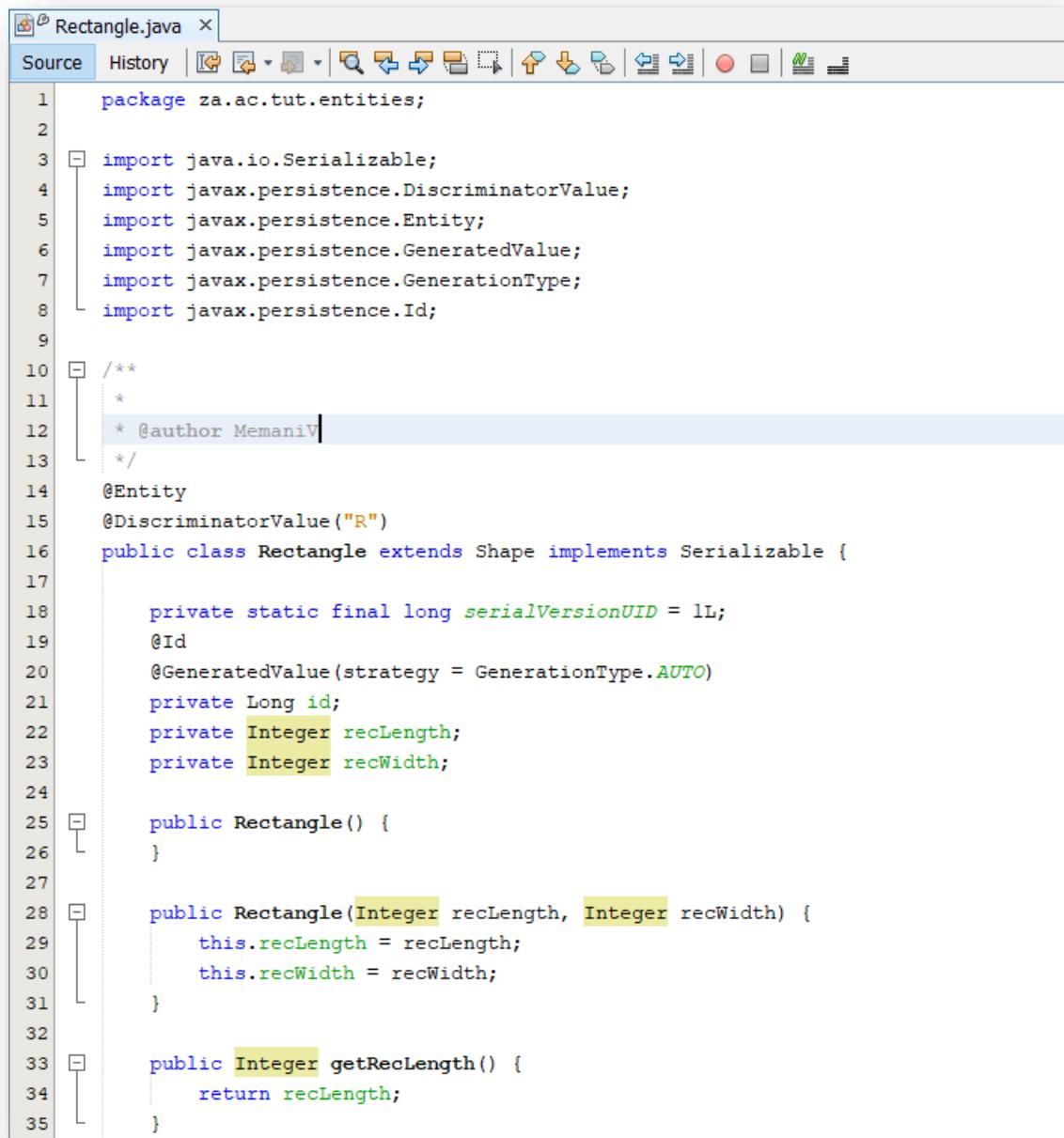
Create a sub entity called **Parallelogram**.

```
1 package za.ac.tut.entities;
2
3 import java.io.Serializable;
4 import javax.persistence.DiscriminatorValue;
5 import javax.persistence.Entity;
6 import javax.persistence.GeneratedValue;
7 import javax.persistence.GenerationType;
8 import javax.persistence.Id;
9
10 /**
11 *
12 * @author MemaniV
13 */
14 @Entity
15 @DiscriminatorValue("P")
16 public class Parallelogram extends Shape implements Serializable {
17
18     private static final long serialVersionUID = 1L;
19     @Id
20     @GeneratedValue(strategy = GenerationType.AUTO)
21     private Long id;
22     private Integer base;
23     private Integer height;
24
25     public Parallelogram() {
26     }
27
28     public Parallelogram(Integer base, Integer height) {
29         this.base = base;
30         this.height = height;
31     }
32
33     public Integer getBase() {
34         return base;
35     }
```

```
37 |     public void setBase(Integer base) {
38 |         this.base = base;
39 |
40 |
41 |     public Integer getHeight() {
42 |         return height;
43 |     }
44 |
45 |     public void setHeight(Integer height) {
46 |         this.height = height;
47 |     }
48 |
49 |
50 |     public Long getId() {
51 |         return id;
52 |     }
53 |
54 |     @Override
55 |     public void setId(Long id) {
56 |         this.id = id;
57 |     }
58 |
59 |     @Override
60 |     public int hashCode() {
61 |         int hash = 0;
62 |         hash += (id != null ? id.hashCode() : 0);
63 |         return hash;
64 |     }
```

```
66 |     @Override
67 |     public boolean equals(Object object) {
68 |         if (!(object instanceof Parallelogram)) {
69 |             return false;
70 |         }
71 |         Parallelogram other = (Parallelogram) object;
72 |         if ((this.id == null &amp; other.id != null) ||
73 |             (this.id != null &amp; !this.id.equals(other.id))) {
74 |             return false;
75 |         }
76 |         return true;
77 |     }
78 |
79 |     @Override
80 |     public String toString() {
81 |         return "za.ac.tut.entities.Parallelogram[ id=" + id + " ]";
82 |     }
83 |
84 | }
```

Create a **Rectangle** sub entity.



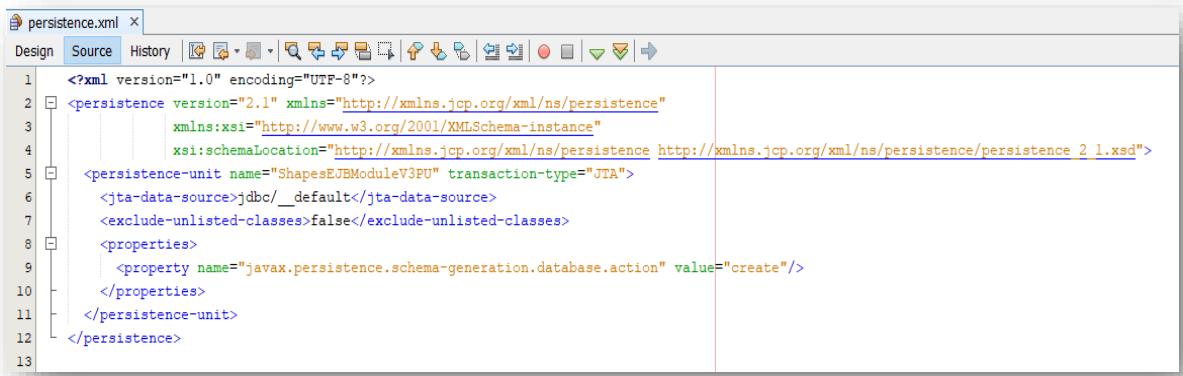
The screenshot shows a Java code editor window titled "Rectangle.java". The code defines a class named "Rectangle" that extends "Shape" and implements "Serializable". The class has private fields for id, recLength, and recWidth, and two constructors. It also has a getter method for recLength. The code uses annotations like @Entity, @DiscriminatorValue, and @Id. The file path is "za.ac.tut.entities.Rectangle.java".

```
1 package za.ac.tut.entities;
2
3 import java.io.Serializable;
4 import javax.persistence.DiscriminatorValue;
5 import javax.persistence.Entity;
6 import javax.persistence.GeneratedValue;
7 import javax.persistence.GenerationType;
8 import javax.persistence.Id;
9
10 /**
11 *
12 * @author MemaniV
13 */
14 @Entity
15 @DiscriminatorValue("R")
16 public class Rectangle extends Shape implements Serializable {
17
18     private static final long serialVersionUID = 1L;
19     @Id
20     @GeneratedValue(strategy = GenerationType.AUTO)
21     private Long id;
22     private Integer recLength;
23     private Integer recWidth;
24
25     public Rectangle() {
26     }
27
28     public Rectangle(Integer recLength, Integer recWidth) {
29         this.recLength = recLength;
30         this.recWidth = recWidth;
31     }
32
33     public Integer getRecLength() {
34         return recLength;
35     }
}
```

```
37  public void setRecLength(Integer recLength) {
38      this.recLength = recLength;
39  }
40
41  public Integer getRecWidth() {
42      return recWidth;
43  }
44
45  public void setRecWidth(Integer recWidth) {
46      this.recWidth = recWidth;
47  }
48
49  @Override
50  public Long getId() {
51      return id;
52  }
53
54  @Override
55  public void setId(Long id) {
56      this.id = id;
57  }
58
59  @Override
60  public int hashCode() {
61      int hash = 0;
62      hash += (id != null ? id.hashCode() : 0);
63      return hash;
64  }
```

```
66  @Override
67  public boolean equals(Object object) {
68      if (!(object instanceof Rectangle)) {
69          return false;
70      }
71      Rectangle other = (Rectangle) object;
72      if ((this.id == null && other.id != null) ||
73          (this.id != null && !this.id.equals(other.id))) {
74          return false;
75      }
76      return true;
77  }
78
79  @Override
80  public String toString() {
81      return "za.ac.tut.entities.Rectangle[ id=" + id + " ]";
82  }
83
84 }
```

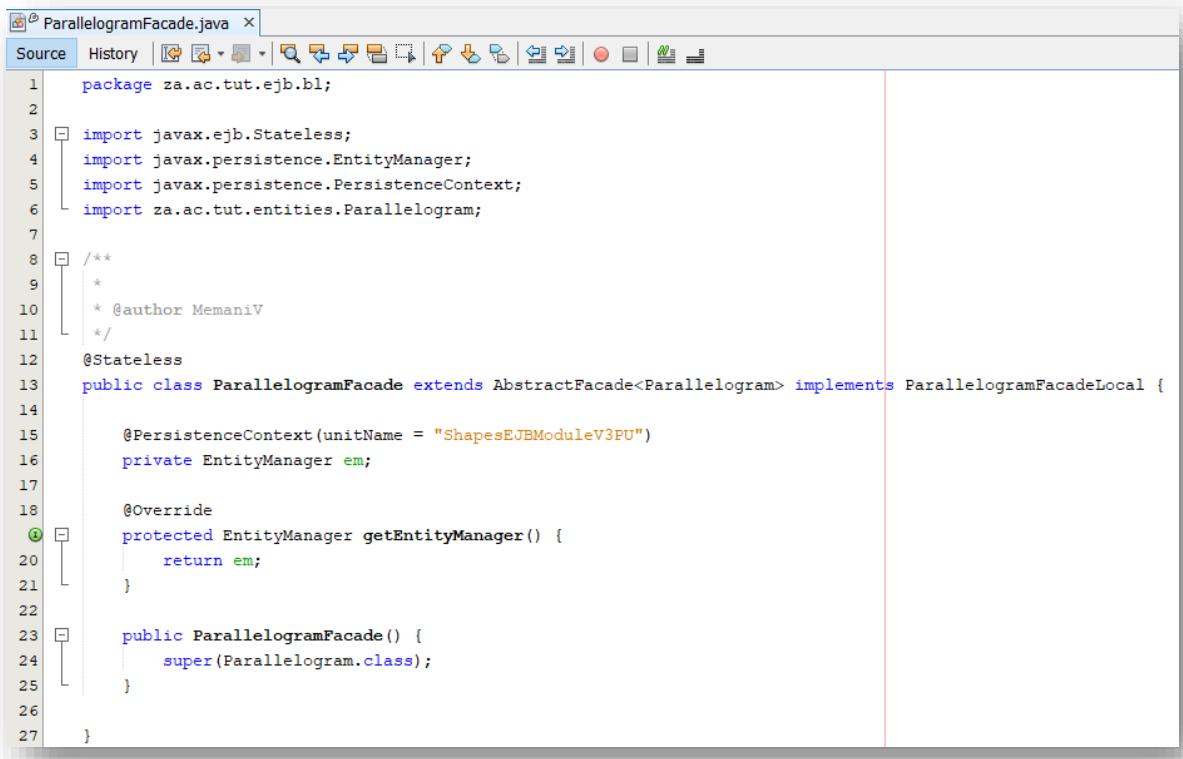
View the persistence.xml file.



```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence_2_1.xsd">
    <persistence-unit name="ShapesEJBModuleV3PU" transaction-type="JTA">
        <jta-data-source>jdbc/_default</jta-data-source>
        <exclude-unlisted-classes>false</exclude-unlisted-classes>
        <properties>
            <property name="javax.persistence.schema-generation.database.action" value="create"/>
        </properties>
    </persistence-unit>
</persistence>
```

Create business code (CRUD operations) for the entities

- **ParallelogramFacade.**



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

import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import za.ac.tut.entities.Parallelogram;

/**
 * 
 * @author MemaniV
 */
@Stateless
public class ParallelogramFacade extends AbstractFacade<Parallelogram> implements ParallelogramFacadeLocal {

    @PersistenceContext(unitName = "ShapesEJBModuleV3PU")
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    public ParallelogramFacade() {
        super(Parallelogram.class);
    }
}
```

- **RectangleFacade**

The screenshot shows a Java code editor with the following code:

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

import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import za.ac.tut.entities.Rectangle;

/*
 *
 * @author MemaniV
 */
@Stateless
public class RectangleFacade extends AbstractFacade<Rectangle> implements RectangleFacadeLocal {

    @PersistenceContext(unitName = "ShapesEJBModuleV3PU")
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    public RectangleFacade() {
        super(Rectangle.class);
    }
}
```

- **ShapeFacade**

The screenshot shows a Java code editor with the file `ShapeFacade.java` open. The code implements a facade pattern for managing shapes. It includes imports for `Stateless`, `EntityManager`, `PersistenceContext`, and the `Shape` entity. The class is annotated with `@Stateless` and `@PersistenceContext(unitName = "ShapesEJBModuleV3PU")`. It overrides the `getEntityManager` method and provides a constructor that calls the superclass constructor with `Shape.class`.

```
1 package za.ac.tut.ejb.bl;
2
3 import javax.ejb.Stateless;
4 import javax.persistence.EntityManager;
5 import javax.persistence.PersistenceContext;
6 import za.ac.tut.entities.Shape;
7
8 /**
9  * @author MemaniV
10 */
11
12 @Stateless
13 public class ShapeFacade extends AbstractFacade<Shape> implements ShapeFacadeLocal {
14
15     @PersistenceContext(unitName = "ShapesEJBModuleV3PU")
16     private EntityManager em;
17
18     @Override
19     protected EntityManager getEntityManager() {
20         return em;
21     }
22
23     public ShapeFacade() {
24         super(Shape.class);
25     }
26
27 }
```

▪ AbstractFacade.

AbstractFacade.java

```

1 package za.ac.tut.bl;
2
3 import java.util.List;
4 import javax.persistence.EntityManager;
5
6 /**
7  * @author MemaniV
8 */
9
10 public abstract class AbstractFacade<T> {
11
12     private Class<T> entityClass;
13
14     public AbstractFacade(Class<T> entityClass) {
15         this.entityClass = entityClass;
16     }
17
18     protected abstract EntityManager getEntityManager();
19
20     public void create(T entity) {
21         getEntityManager().persist(entity);
22     }
23
24     public void edit(T entity) {
25         getEntityManager().merge(entity);
26     }
27
28     public void remove(T entity) {
29         getEntityManager().remove(getEntityManager().merge(entity));
30     }
31

```

```

32     public T find(Object id) {
33         return getEntityManager().find(entityClass, id);
34     }
35
36     public List<T> findAll() {
37         javax.persistence.criteria.CriteriaQuery cq =
38             getEntityManager().getCriteriaBuilder().createQuery();
39         cq.select(cq.from(entityClass));
40         return getEntityManager().createQuery(cq).getResultList();
41     }
42
43     public List<T> findRange(int[] range) {
44         javax.persistence.criteria.CriteriaQuery cq =
45             getEntityManager().getCriteriaBuilder().createQuery();
46         cq.select(cq.from(entityClass));
47         javax.persistence.Query q = getEntityManager().createQuery(cq);
48         q.setMaxResults(range[1] - range[0] + 1);
49         q.setFirstResult(range[0]);
50         return q.getResultList();
51     }
52
53     public int count() {
54         javax.persistence.criteria.CriteriaQuery cq =
55             getEntityManager().getCriteriaBuilder().createQuery();
56         javax.persistence.criteria.Root<T> rt = cq.from(entityClass);
57         cq.select(getEntityManager().getCriteriaBuilder().count(rt));
58         javax.persistence.Query q = getEntityManager().createQuery(cq);
59         return ((Long) q.getSingleResult()).intValue();
60     }
61
62 }

```

- **ParallelogramFacadeLocal**

The screenshot shows a Java code editor window with the title "ParallelogramFacadeLocal.java". The code implements a local EJB interface for managing Parallelogram entities. It includes imports for java.util.List, javax.ejb.Local, and za.ac.tut.entities.Parallelogram. The interface is annotated with @Local and contains methods for create, edit, remove, find, findAll, findRange, and count.

```
1 package za.ac.tut.ejb.bl;
2
3 import java.util.List;
4 import javax.ejb.Local;
5 import za.ac.tut.entities.Parallelogram;
6
7 /**
8 * @author MemaniV
9 */
10
11 @Local
12 public interface ParallelogramFacadeLocal {
13
14     void create(Parallelogram parallelogram);
15
16     void edit(Parallelogram parallelogram);
17
18     void remove(Parallelogram parallelogram);
19
20     Parallelogram find(Object id);
21
22     List<Parallelogram> findAll();
23
24     List<Parallelogram> findRange(int[] range);
25
26     int count();
27
28 }
```

- **RectangleFacadeLocal.**

The screenshot shows a Java code editor window with the title "RectangleFacadeLocal.java". The code implements a local EJB interface for managing Rectangle entities. It includes imports for java.util.List, javax.ejb.Local, and za.ac.tut.entities.Rectangle. The interface is annotated with @Local and contains methods for create, edit, remove, find, findAll, findRange, and count.

```
1 package za.ac.tut.ejb.bl;
2
3 import java.util.List;
4 import javax.ejb.Local;
5 import za.ac.tut.entities.Rectangle;
6
7 /**
8 * @author MemaniV
9 */
10
11 @Local
12 public interface RectangleFacadeLocal {
13
14     void create(Rectangle rectangle);
15
16     void edit(Rectangle rectangle);
17
18     void remove(Rectangle rectangle);
19
20     Rectangle find(Object id);
21
22     List<Rectangle> findAll();
23
24     List<Rectangle> findRange(int[] range);
25
26     int count();
27
28 }
```

- **ShapeFacadeLocal.**

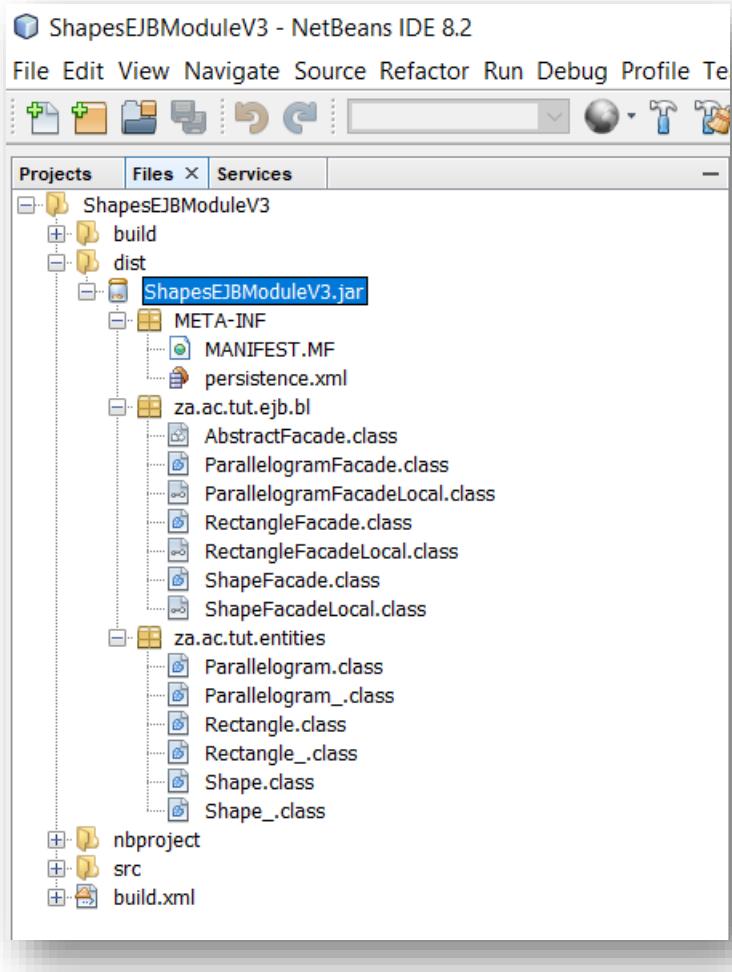
The screenshot shows the NetBeans IDE interface with the file `ShapeFacadeLocal.java` open. The code defines a local EJB interface for managing shapes. It includes imports for `java.util.List`, `javax.ejb.Local`, and `za.ac.tut.entities.Shape`. The interface is annotated with `@Local`. It contains methods for creating, editing, removing shapes, finding a single shape by ID, finding all shapes, finding shapes within a range, and counting the total number of shapes.

```
1 package za.ac.tut.ejb.bl;
2
3 import java.util.List;
4 import javax.ejb.Local;
5 import za.ac.tut.entities.Shape;
6
7 /**
8 * ...
9 * @author MemaniV
10 */
11 @Local
12 public interface ShapeFacadeLocal {
13
14     void create(Shape shape);
15
16     void edit(Shape shape);
17
18     void remove(Shape shape);
19
20     Shape find(Object id);
21
22     List<Shape> findAll();
23
24     List<Shape> findRange(int[] range);
25
26     int count();
27
28 }
```

- ✓ Clean and build the EJB project.

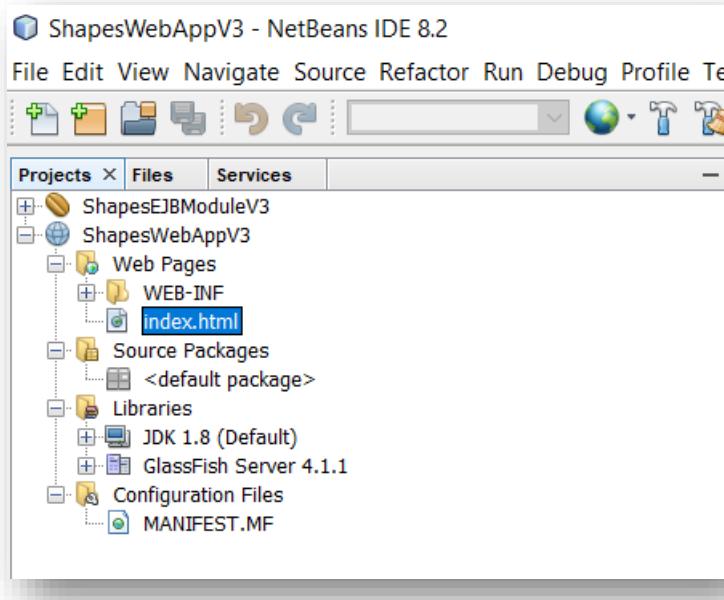
The screenshot shows the NetBeans IDE Output window with the tab `Java DB Database Process` selected. The window displays the build logs for the `GlassFish Server 4.1.1` and the `ShapesEJBModuleV3` project. The logs show the compilation process, including the creation of a directory and the generation of a JAR file. The message `BUILD SUCCESSFUL (total time: 1 second)` indicates a successful build.

```
Java DB Database Process x GlassFish Server 4.1.1 x ShapesEJBModuleV3 (clean,dist) x
Note: Optional file was not found. REINVOKE eclipselink_eclim.xml continuing with generation.
Note: C:\Users\memaniv\Documents\NetBeansProjects\ShapesEJBModuleV3\src\java\za\ac\tut\ejb\bl\AbstractFaca
Note: Recompile with -Xlint:unchecked for details.
1 warning
compile:
library-inclusion-in-archive:
created dir: C:\Users\memaniv\Documents\NetBeansProjects\ShapesEJBModuleV3\dist
Building jar: C:\Users\memaniv\Documents\NetBeansProjects\ShapesEJBModuleV3\dist\ShapesEJBModuleV3.jar
dist:
BUILD SUCCESSFUL (total time: 1 second)
```



Part D – Create a web client project.

Create a web client project called **ShapesWebAppV3**.

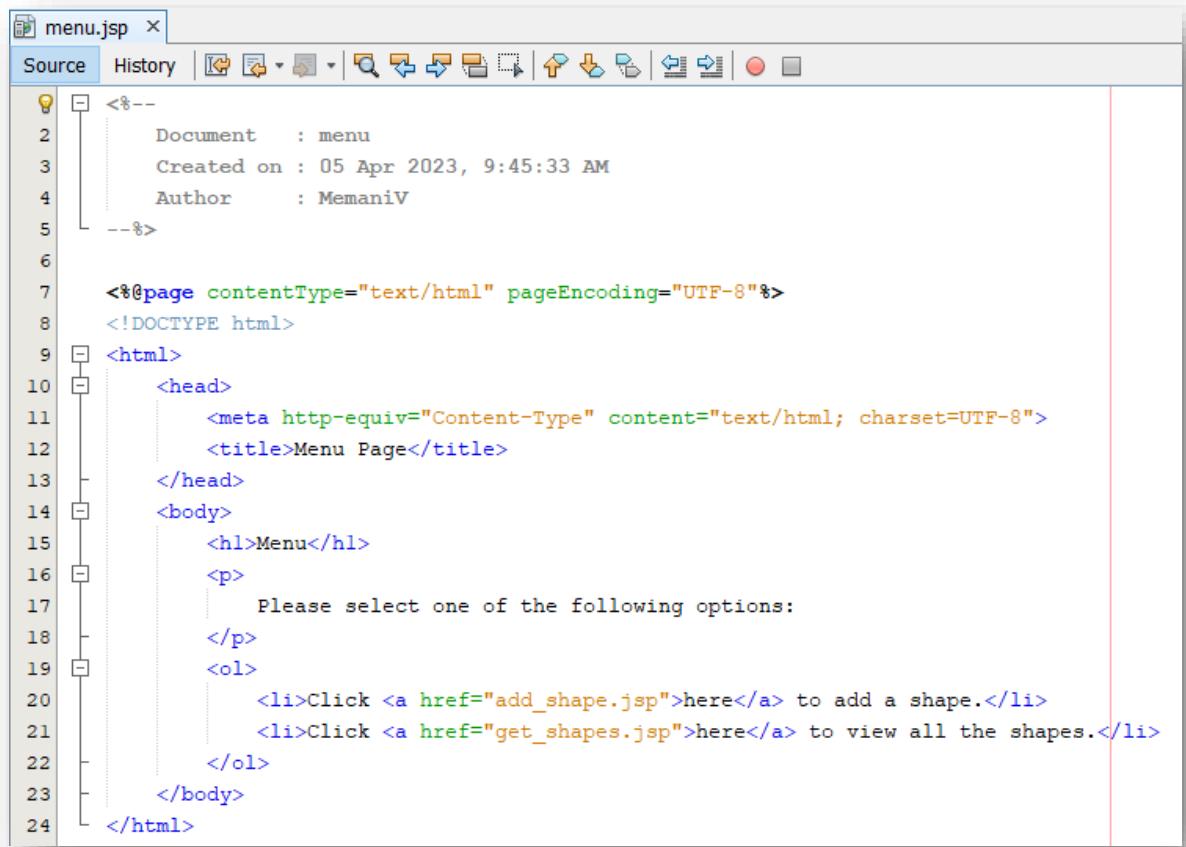


Edit the **index.html** page.

```
<!DOCTYPE html>

<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 page</h1>
        <p>
            Welcome to our web app. Please click <a href="menu.jsp">here</a> to start.
        </p>
    </body>
</html>
```

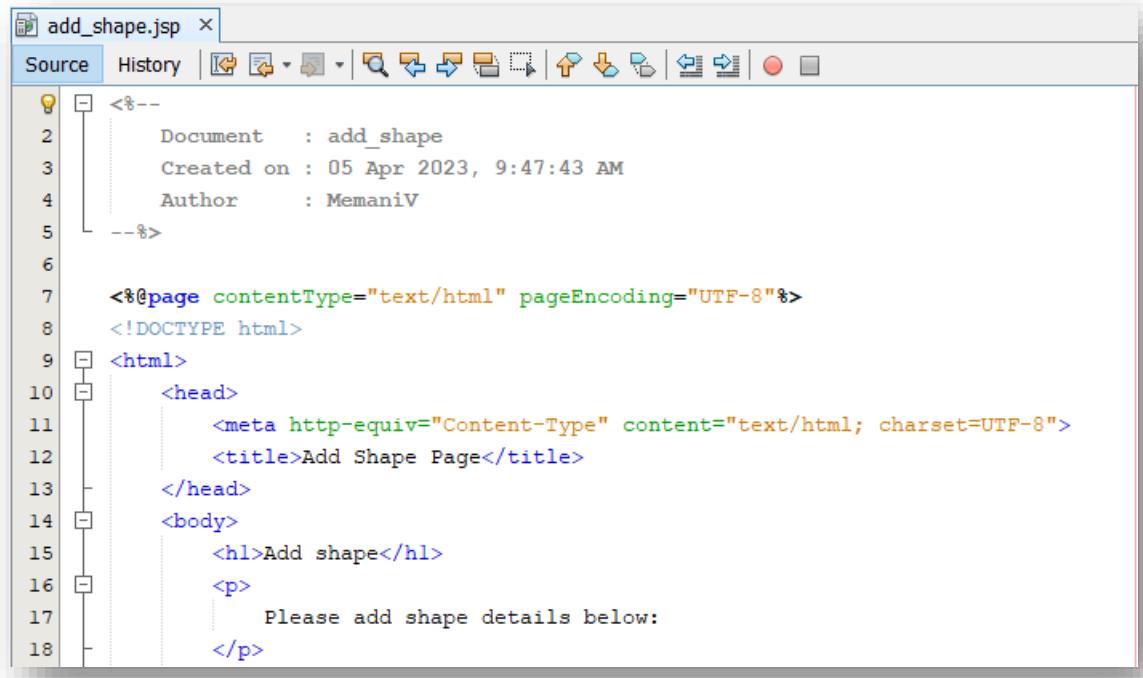
Create the menu.jsp file.



The screenshot shows a Java IDE interface with the menu.jsp file open. The code is a JSP page that generates an HTML menu. It includes a header with document information, an HTML structure with a head and body section, and a list of options in the body.

```
<%--  
1 Document : menu  
2 Created on : 05 Apr 2023, 9:45:33 AM  
3 Author : MemaniV  
4--%>  
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="add_shape.jsp">here</a> to add a shape.</li>  
21         <li>Click <a href="get_shapes.jsp">here</a> to view all the shapes.</li>  
22     </ol>  
23 </body>  
24 </html>
```

Create add_shape.jsp file.



The screenshot shows a Java IDE interface with the add_shape.jsp file open. The code is a JSP page that generates an HTML form for adding a shape. It includes a header with document information, an HTML structure with a head and body section, and a list of options in the body.

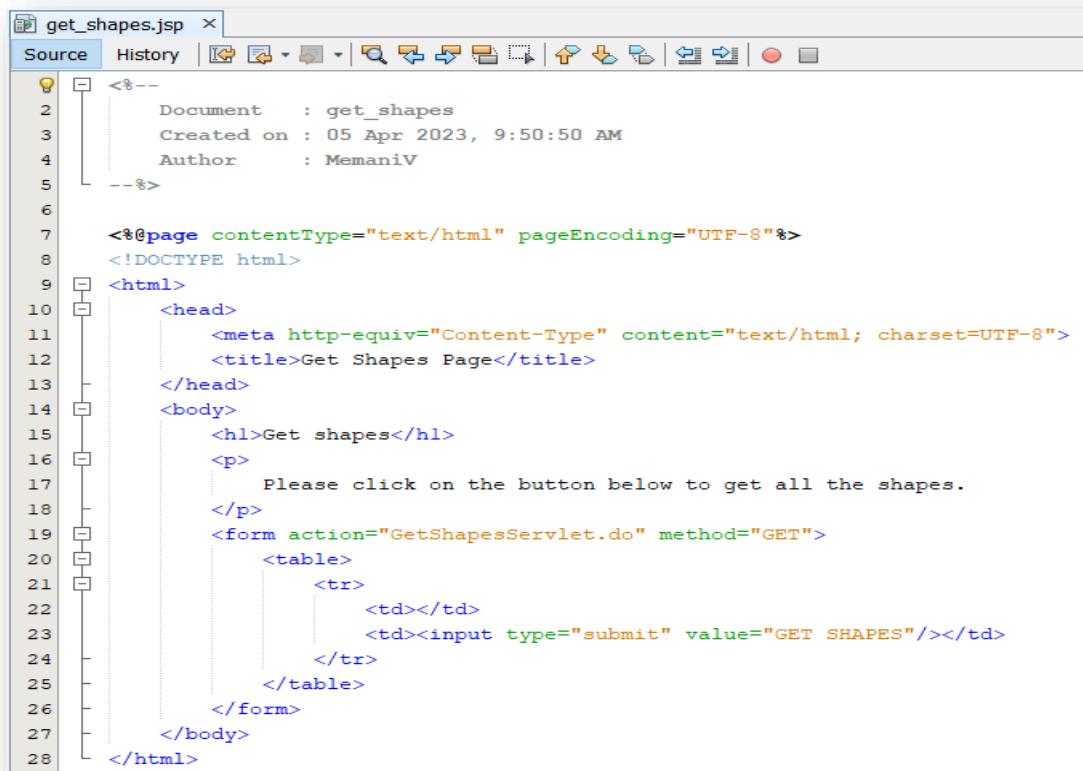
```
<%--  
1 Document : add_shape  
2 Created on : 05 Apr 2023, 9:47:43 AM  
3 Author : MemaniV  
4--%>  
5  
6  
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>  
8 <!DOCTYPE html>  
9 <html>  
10 <head>  
11     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
12     <title>Add Shape Page</title>  
13 </head>  
14 <body>  
15     <h1>Add shape</h1>  
16     <p>  
17         Please add shape details below:  
18     </p>
```

```

19 <form action="AddShapeServlet.do" method="POST">
20   <table>
21     <tr>
22       <td>Type: </td>
23       <td>
24         <select name="type">
25           <option value="P">Parallelogram</option>
26           <option value="R">Rectangle</option>
27         </select>
28       </td>
29     </tr>
30     <tr>
31       <td>Side 1: </td>
32       <td><input type="text" name="side1"/></td>
33     </tr>
34     <tr>
35       <td>Side 2: </td>
36       <td><input type="text" name="side2"/></td>
37     </tr>
38     <tr>
39       <td>Is a big shape? </td>
40       <td>
41         <select name="isBig">
42           <option value="True">Yes</option>
43           <option value="False">No</option>
44         </select>
45       </td>
46     </tr>
47     <tr>
48       <td></td>
49       <td><input type="submit" value="ADD SHAPE"/></td>
50     </tr>
51   </table>
52 </form>
53 </body>
54 </html>

```

Create the **get_shapes.jsp** file.

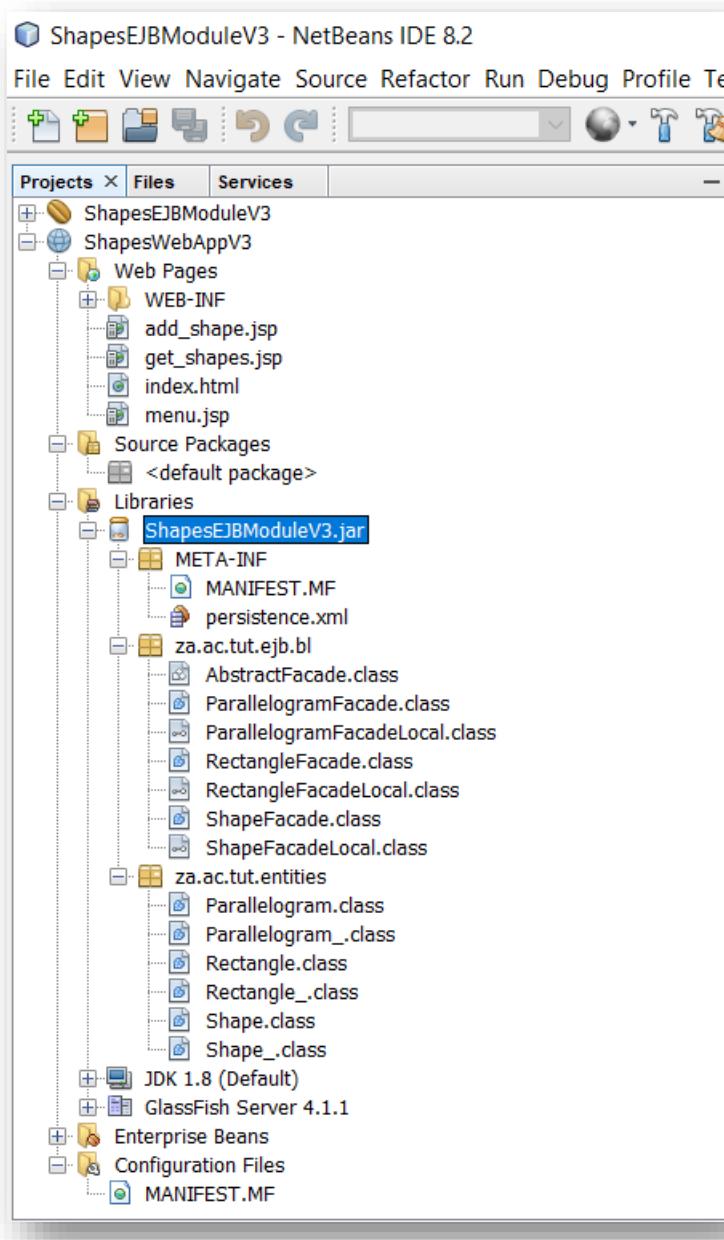


```

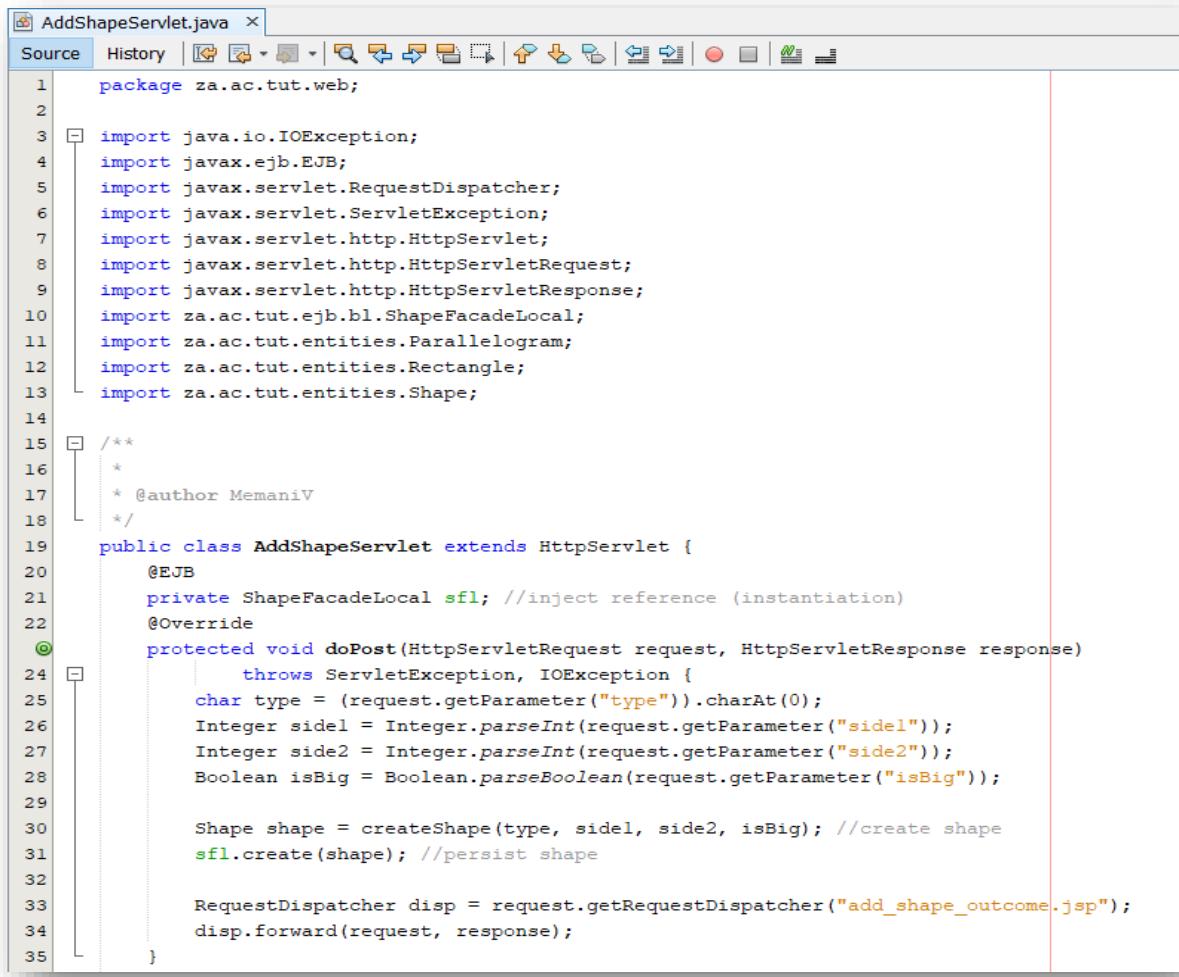
get_shapes.jsp
Source History | 
1 <%-- 
2   Document : get_shapes
3   Created on : 05 Apr 2023, 9:50:50 AM
4   Author : MemaniV
5 --%>
6
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>
8 <!DOCTYPE html>
9 <html>
10  <head>
11    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12    <title>Get Shapes Page</title>
13  </head>
14  <body>
15    <h1>Get shapes</h1>
16    <p>
17      Please click on the button below to get all the shapes.
18    </p>
19    <form action="GetShapesServlet.do" method="GET">
20      <table>
21        <tr>
22          <td></td>
23          <td><input type="submit" value="GET SHAPES"/></td>
24        </tr>
25      </table>
26    </form>
27  </body>
28 </html>

```

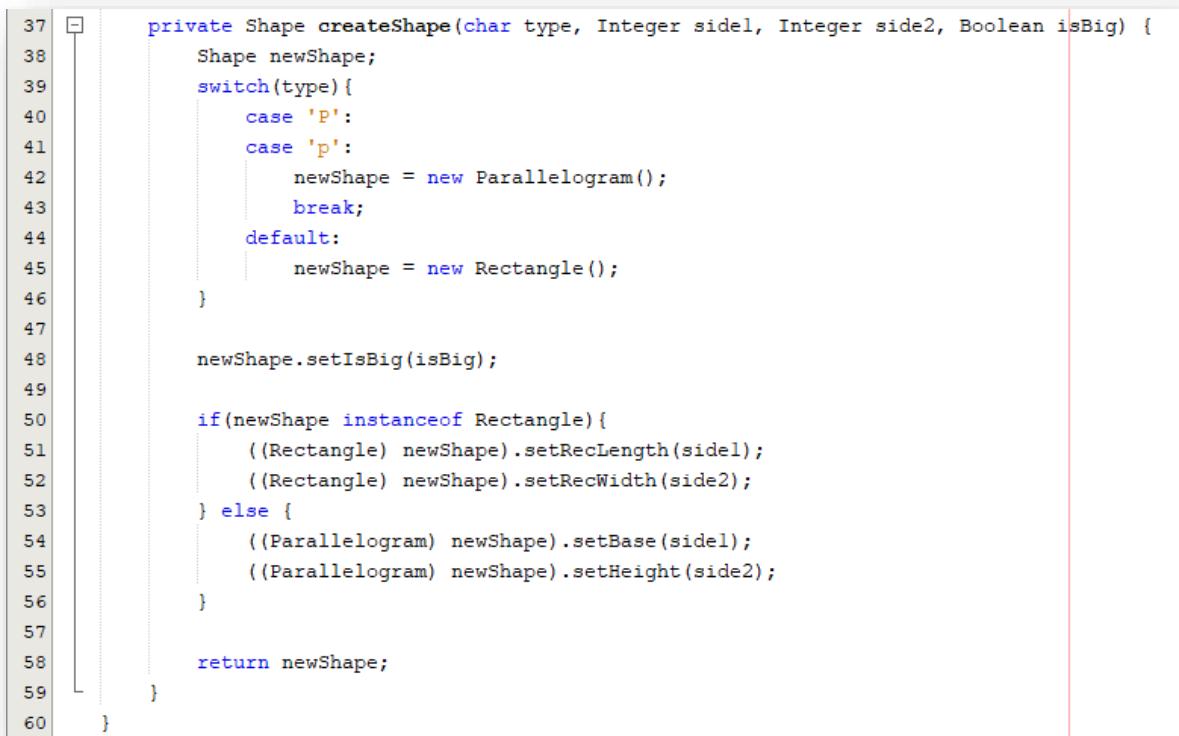
Add the **ShapesEJBModuleV3.jar** file to the library of **ShapesWebApp3**.



Create the AddShapeServlet.java file.



```
1 package za.ac.tut.web;
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.ShapeFacadeLocal;
11 import za.ac.tut.entities.Parallelogram;
12 import za.ac.tut.entities.Rectangle;
13 import za.ac.tut.entities.Shape;
14
15 /**
16 *
17 * @author MemaniV
18 */
19 public class AddShapeServlet extends HttpServlet {
20     @EJB
21     private ShapeFacadeLocal sfl; //inject reference (instantiation)
22     @Override
23     protected void doPost(HttpServletRequest request, HttpServletResponse response)
24         throws ServletException, IOException {
25         char type = (request.getParameter("type")).charAt(0);
26         Integer sidel = Integer.parseInt(request.getParameter("sidel"));
27         Integer side2 = Integer.parseInt(request.getParameter("side2"));
28         Boolean isBig = Boolean.parseBoolean(request.getParameter("isBig"));
29
30         Shape shape = createShape(type, sidel, side2, isBig); //create shape
31         sfl.create(shape); //persist shape
32
33         RequestDispatcher disp = request.getRequestDispatcher("add_shape_outcome.jsp");
34         disp.forward(request, response);
35     }
36 }
```



```
37     private Shape createShape(char type, Integer sidel, Integer side2, Boolean isBig) {
38         Shape newShape;
39         switch(type){
40             case 'P':
41             case 'p':
42                 newShape = new Parallelogram();
43                 break;
44             default:
45                 newShape = new Rectangle();
46             }
47
48             newShape.setIsBig(isBig);
49
50             if(newShape instanceof Rectangle){
51                 ((Rectangle) newShape).setRecLength(sidel);
52                 ((Rectangle) newShape).setRecWidth(side2);
53             } else {
54                 ((Parallelogram) newShape).setBase(sidel);
55                 ((Parallelogram) newShape).setHeight(side2);
56             }
57
58             return newShape;
59         }
60     }
```

Create the **GetShapesServlet.java** file.

The screenshot shows the Java code for `GetShapesServlet.java` in an IDE. The code implements a servlet that retrieves a list of shapes from an EJB and forwards the request to a JSP page.

```
1 package za.ac.tut.web;
2
3 import java.io.IOException;
4 import java.util.List;
5 import javax.ejb.EJB;
6 import javax.servlet.RequestDispatcher;
7 import javax.servlet.ServletException;
8 import javax.servlet.http.HttpServlet;
9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11 import za.ac.tut.ejb.bl.ShapeFacadeLocal;
12 import za.ac.tut.entities.Shape;
13
14 /**
15  * 
16  * @author MemaniV
17  */
18 public class GetShapesServlet extends HttpServlet {
19     @EJB
20     private ShapeFacadeLocal pfl; //inject reference (instantiation)
21
22     @Override
23     protected void doGet(HttpServletRequest request, HttpServletResponse response)
24         throws ServletException, IOException {
25         List<Shape> list = pfl.findAll(); //get shapes
26         request.setAttribute("shapes", list);
27
28         RequestDispatcher disp = request.getRequestDispatcher("get_shapes_outcome.jsp");
29         disp.forward(request, response);
30     }
31 }
32 }
```

Create the **add_book_outcome.jsp** file.

The screenshot shows a Java code editor with the file `add_shapes_outcome.jsp` open. The code is a JSP page that adds a shape outcome to a database. It includes standard HTML tags like `<html>`, `<head>`, and `<body>`, as well as JSP tags like `<%--`, `<%@page contentType="text/html" pageEncoding="UTF-8"%>`, and `<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">`. The code also contains several `<p>` tags with descriptive text for the user.

```
<%--  
1      Document      : add_shapes_outcome  
2      Created on   : 05 Apr 2023, 10:10:03 AM  
3      Author       : MemaniV  
4--%>  
  
<%@page contentType="text/html" pageEncoding="UTF-8"%>  
<!DOCTYPE html>  
<html>  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
<title>Add Shape Outcome Page</title>  
</head>  
<body>  
<h1>Add shape outcome</h1>  
<p>  
The shape has been successfully persisted into the database.  
Please click <a href="menu.jsp">here</a> to get back to the menu  
page or <a href="index.html">here</a> to the main page.  
</p>  
</body>  
</html>
```

Create the get_shapes_outcome.jsp file.

```
<%--  
2 Document : get_shapes_outcome  
3 Created on : 05 Apr 2023, 10:12:04 AM  
4 Author : MemaniV  
5 --%>  
6  
7 <%@page import="za.ac.tut.entities.Parallelogram"%>  
8 <%@page import="za.ac.tut.entities.Rectangle"%>  
9 <%@page import="java.util.List"%>  
10 <%@page import="za.ac.tut.entities.Shape"%>  
11 <%@page contentType="text/html" pageEncoding="UTF-8"%>  
12 <!DOCTYPE html>  
13 <html>  
14 <head>  
15 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
16 <title>Get Shapes Outcome Page</title>  
17 </head>  
18 <body>  
19 <h1>Get shapes outcome</h1>  
20 <%  
21 List<Shape> shapes = (List<Shape>)request.getAttribute("shapes");  
22 %>  
23 <p>  
24 Below is the list of shapes retrieved from the database.  
25 </p>
```

```

27   <table>
28     <%
29       String shapeType = "Parallelogram";
30       Integer sidel, side2;
31       for(int i = 0; i < shapes.size(); i++) {
32         Shape p = shapes.get(i);
33         Long id = p.getId();
34         Boolean isBig = p.getIsBig();
35
36         if(p instanceof Rectangle) {
37             shapeType = "Rectangle";
38             Rectangle rec = (Rectangle)p;
39             sidel = rec.getRecLength();
40             side2 = rec.getRecWidth();
41         } else {
42             Parallelogram par = (Parallelogram)p;
43             sidel = par.getBase();
44             side2 = par.getHeight();
45         }
46     %>
47     <tr>
48       <td>ID: </td>
49       <td><%=id%></td>
50     </tr>
51     <tr>
52       <td>Shape type: </td>
53       <td><%=shapeType%></td>
54     </tr>
55     <tr>
56       <td>Side 1: </td>
57       <td><%=sidel%></td>
58     </tr>
59     <tr>
60       <td>Side2: </td>
61       <td><%=side2%></td>

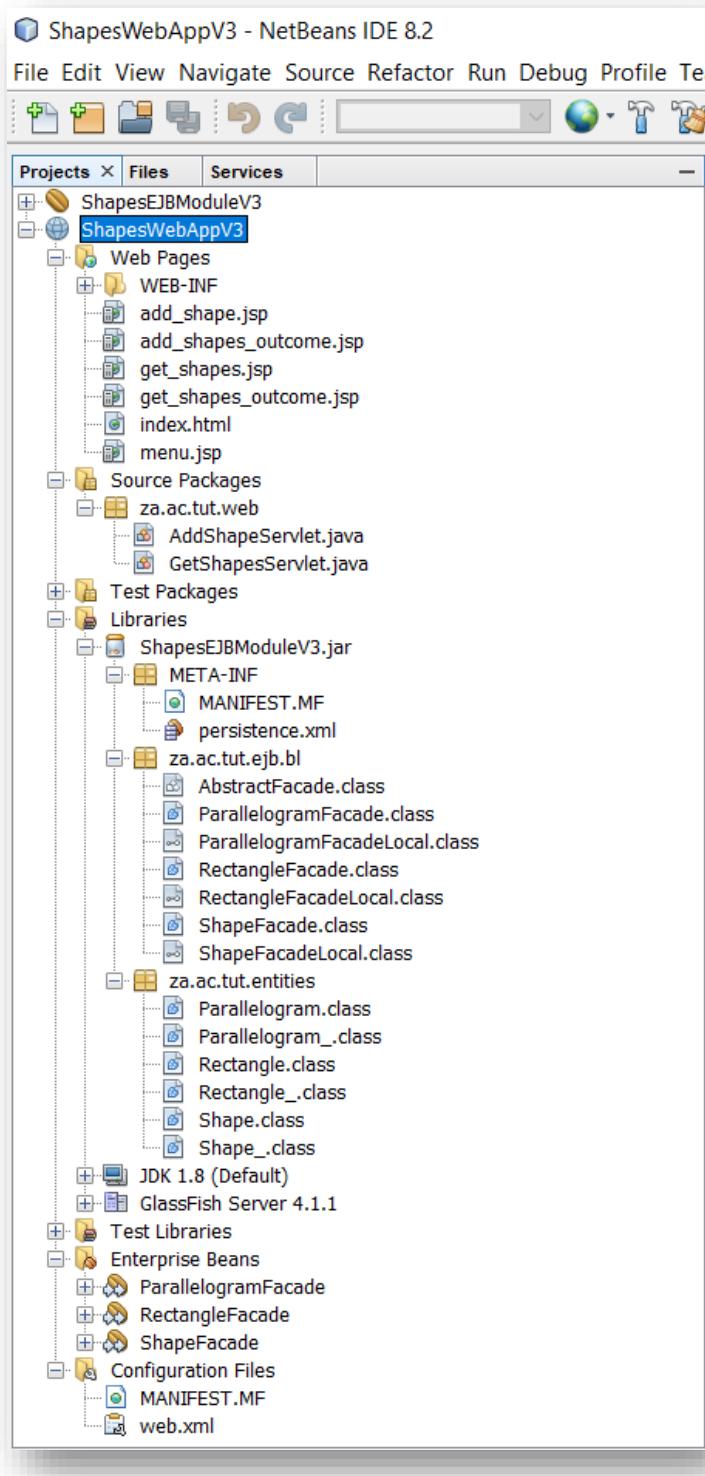
```

```

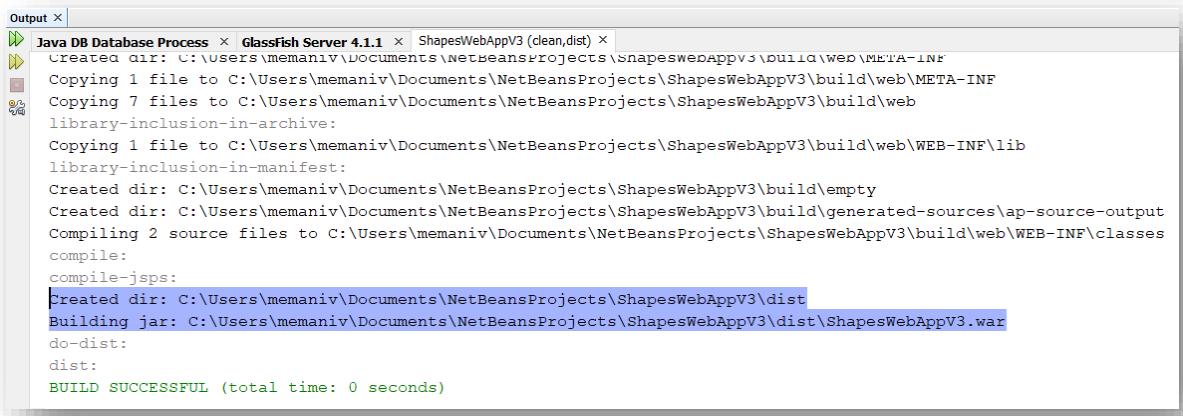
63   <tr>
64     <td>Is the shape big? </td>
65     <td><%=isBig%></td>
66   </tr>
67   <%
68   %>
69   </table>
70   <p>
71     Please click <a href="menu.jsp">here</a> to get back to the menu page or <a href="index.html">here</a>
72     to the main page.
73   </p>
74   </body>
75 </html>

```

View the complete project structure of **ShapesWebAppV3**.

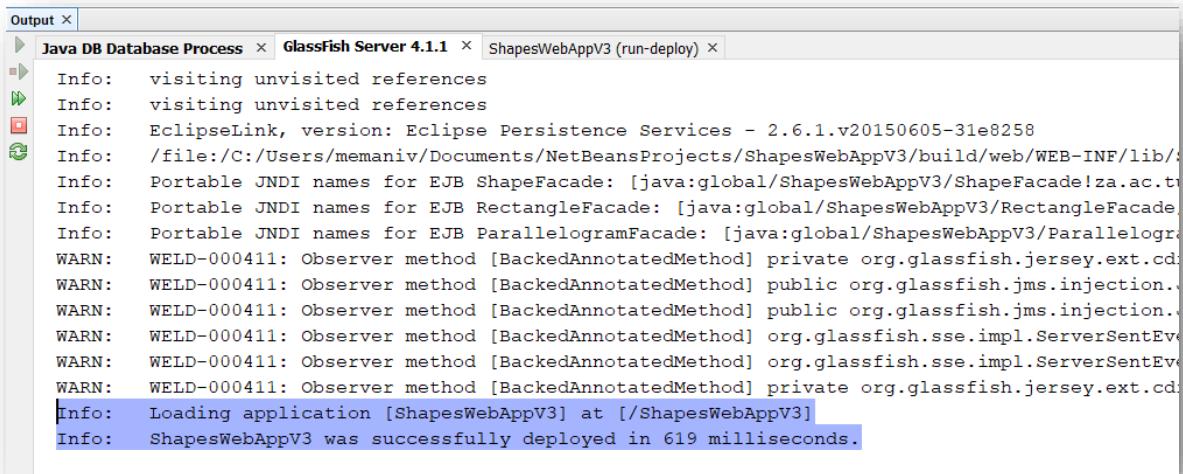


Compile the project.



```
Output ×
Java DB Database Process × GlassFish Server 4.1.1 × ShapesWebAppV3 (clean,dist) ×
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\ShapesWebAppV3\build\web\META-INF
Copying 1 file to C:\Users\memaniv\Documents\NetBeansProjects\ShapesWebAppV3\build\web\META-INF
Copying 7 files to C:\Users\memaniv\Documents\NetBeansProjects\ShapesWebAppV3\build\web
library-inclusion-in-archive:
Copying 1 file to C:\Users\memaniv\Documents\NetBeansProjects\ShapesWebAppV3\build\web\WEB-INF\lib
library-inclusion-in-manifest:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\ShapesWebAppV3\build\empty
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\ShapesWebAppV3\build\generated-sources\ap-source-output
Compiling 2 source files to C:\Users\memaniv\Documents\NetBeansProjects\ShapesWebAppV3\build\web\WEB-INF\classes
compile:
compile-jsp:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\ShapesWebAppV3\dist
Building jar: C:\Users\memaniv\Documents\NetBeansProjects\ShapesWebAppV3\dist\ShapesWebAppV3.war
do-dist:
dist:
BUILD SUCCESSFUL (total time: 0 seconds)
```

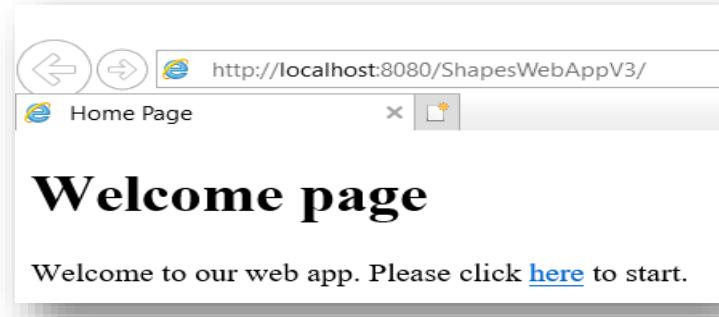
Deploy the project.



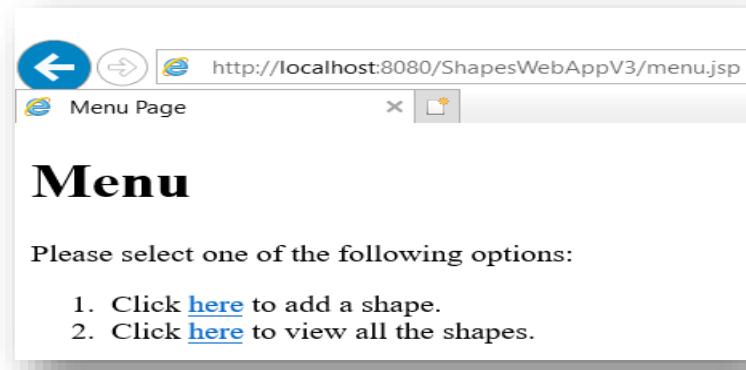
```
Output ×
Java DB Database Process × GlassFish Server 4.1.1 × ShapesWebAppV3 (run-deploy) ×
Info: visiting unvisited references
Info: visiting unvisited references
Info: EclipseLink, version: Eclipse Persistence Services - 2.6.1.v20150605-31e8258
Info: /file:/C:/Users/memaniv/Documents/NetBeansProjects/ShapesWebAppV3/build/web/WEB-INF/lib/
Info: Portable JNDI names for EJB ShapeFacade: [java:global/ShapesWebAppV3/ShapeFacade!za.ac.ti...
Info: Portable JNDI names for EJB RectangleFacade: [java:global/ShapesWebAppV3/RectangleFacade!...
Info: Portable JNDI names for EJB ParallelogramFacade: [java:global/ShapesWebAppV3/Parallelogra...
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] private org.glassfish.jersey.ext.cd...
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] public org.glassfish.jms.injection....
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] public org.glassfish.jms.injection....
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] org.glassfish.sse.impl.ServerSentEve...
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] org.glassfish.sse.impl.ServerSentEve...
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] private org.glassfish.jersey.ext.cd...
Info: Loading application [ShapesWebAppV3] at [/ShapesWebAppV3]
Info: ShapesWebAppV3 was successfully deployed in 619 milliseconds.
```

Part E – Run the web application.

Launch the application.



Click on the link



Add a shape.

- ✓ Click on the first link.

Type:

Side 1:

Side 2:

Is a big shape?

- ✓ Fill in the form.

Please add shape details below:

Type:

Side 1:

Side 2:

Is a big shape?

Click on the add button.

here to get back to the menu page or [here](#) to the main page.'"/>

Add shape outcome

The shape has been successfully persisted into the database. Please click [here](#) to get back to the menu page or [here](#) to the main page.

Add four more shapes. View all the shapes.

- ✓ Click on the second link.

Get shapes

Please click on the button below to get all the shapes.

- ✓ Click on the button.

The screenshot shows a web browser window with the URL <http://localhost:8080/ShapesWebAppV3/GetShapesServlet.do>. The title bar says "Get Shapes Outcome Page". The main content area displays the following text:

Get shapes outcome

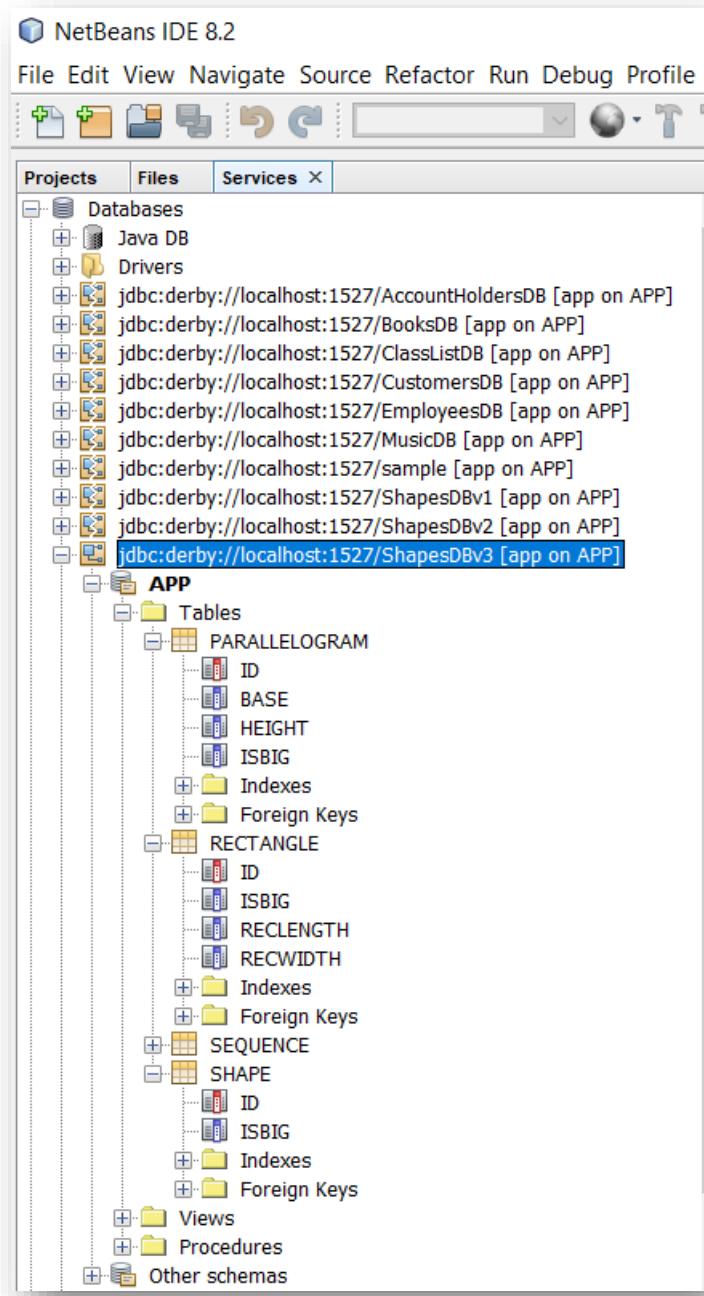
Below is the list of shapes retrieved from the database.

ID: 54
Shape type: Rectangle
Side 1: 20
Side2: 25
Is the shape big? true
ID: 55
Shape type: Rectangle
Side 1: 5
Side2: 8
Is the shape big? false
ID: 1
Shape type: Rectangle
Side 1: 40
Side2: 30
Is the shape big? true
ID: 51
Shape type: Rectangle
Side 1: 40
Side2: 30
Is the shape big? true
ID: 52
Shape type: Rectangle
Side 1: 60
Side2: 70
Is the shape big? true

ID: 53
Shape type: Rectangle
Side 1: 25
Side2: 30
Is the shape big? true

Please click [here](#) to get back to the menu page or [here](#) to the main page.

View the records in the database.



Contents of the **SHAPE** table.

The screenshot shows the SQL Developer interface with two tabs: 'SQL 2' and 'SELECT * FROM APP.SHAP...'. The SQL tab contains the query: 'SELECT * FROM APP.SHAP...'. The results tab shows the following data:

#	ID	ISBIG
1		
2		

Contents of the **RECTANGLE** table.

The screenshot shows the SQL Developer interface with two tabs: 'SQL 2' and 'SELECT * FROM APP.RECTANG...'. The SQL tab contains the query: 'SELECT * FROM APP.RECTANG...'. The results tab shows the following data:

#	ID	ISBIG	RECLENGTH	RECWIDTH
1		54	1	20
2		55	0	5

Contents of the **PARALLELOGRAM** table.

The screenshot shows the SQL Developer interface with two tabs: 'SQL 1' and 'SELECT * FROM APP.PARALLE...'. The SQL tab contains the query: 'SELECT * FROM APP.PARALLE...'. The results tab shows the following data:

#	ID	BASE	HEIGHT	ISBIG
1		1	40	30
2		51	40	30
3		52	60	70
4		53	25	30

9 JPQL

In this chapter we introduce the student to JPQL. We will define the concept, look at the syntax of JPQL, implement JPQL through applications that make queries and sub-queries.

9.1 What is JPQL?

The term JPQL stands for **J**ava **P**ersistence **Q**uery **L**anguage. Consequently JPQL is a language used to query databases. It is similar to SQL, the only difference is that JPQL is object-oriented.

JPQL works along side with JPA. As an API, JPA helps with the mapping of objects to tables of relational databases. Through the usage of the EntityManager class, JPA allows the programmer to perform CRUD operations on a database. So through JPA we are able to create records, read them, update, and delete. But all these functionalities require the invocation of the primary key. Without the primary key we can't find a record, nor update it.

The beauty about JPQL we can create customised query statements without involving the primary key. There are four ways of creating queries in JPQL, namely:

Dynamic queries – the simplest form of queries. It uses dynamic query strings that are handled at runtime.

Named queries – uses static and unchangeable query strings.

Native queries – uses basic SQL statements.

Criteria API – a new API that is used to formulate queries that are more object-oriented.

For now in this module we will confine ourselves to the first two strategies, Dynamic and Named queries. Let us now look at the syntax of JPQL.

9.2 JPQL syntax

JPQL also works with your normal SQL clauses such as SELECT, FROM, WHERE, ORDER BY, GROUP BY, HAVING, LIKE, UPDATE, DELETE and so forth. It also defines some predefined functions such as COUNT, MIN, MAX, AVG, SUM etc. Let us quickly have a look at the syntax of some of the clauses.

9.2.1 SELECT , FROM and WHERE

The SELECT statement is used to return a result or a list of results from the database. The generic syntax is as follows:

```
SELECT <select expression>
FROM <from clause>
WHERE <conditional expression>
```

Let us take an example.

Example

Say you have an entity called **Customer** mapped to a corresponding table in the database. Say that the Customer has two attributes, **name** and **age**. You can then use the SELECT statement in conjunction with the FROM and WHERE to create query statements that can perform a number of tasks, namely:

Get all the customers from the database:

```
SELECT c FROM Customer c
```

The letter c serves as an alias for Customer. This means that it serves as an instance/object of type Customer.

Get only the names:

```
SELECT c.name FROM Customer c
```

Get only the ages:

```
SELECT c.age FROM Customer c
```

Get all the customers with age greater than 20

```
SELECT c FROM Customer c WHERE c.age > 20
```

Get all the customers whose age is between **20** and **30**, both values inclusive

```
SELECT c FROM Customer c WHERE c.age >= 20 AND c.age <=30
```

Get the customer whose name is “Vuyisile”:

```
SELECT c FROM Customer c WHERE c.name = 'Vuyisile'
```

Get the number of customers whose age is above 50

```
SELECT COUNT(c) FROM Customer c WHERE c.age > 50
```

9.2.2 DELETE

The DELETE statement is used to delete an entity or group of entities from the database. The generic syntax is as follows:

```
DELETE FROM <entity name>
```

```
WHERE <conditional expression>
```

Let us work with the same Customer example.

We can delete all the records whose age value is greater than 50 as follows:

```
DELETE FROM Customer c
```

```
WHERE c.age > 50
```

We can delete a specific record.

```
DELETE FROM Customer c
```

```
WHERE c.name = 'Thato'
```

9.2.3 UPDATE

The UPDATE statement is used to update the details of an entity. The generic syntax is as follows:

```
UPDATE <entity name>
```

```
SET <update statement>
```

```
WHERE <conditional expression>
```

Let us continue with the same Customer example.

We can update the name of details of a specific person:

```
UPDATE Customer c
```

```
SET c.age = 10
```

```
WHERE c.name = 'Jabu'
```

So in this example we are changing the age of Jabu to 10.

We can update the details of many entities. In the example below we are changing the names of all older people to "Oldie":

```
UPDATE Customer c
```

```
SET c.name = 'Oldie'
```

```
WHERE c.age > 50
```

9.3 Binding parameters

Binding parameters is a mechanism for binding user given values to query statements. This mechanism is applied when a method needs dynamic values from the user to create a query statement. For example, if you need to create a query statement that needs to return a list of customers within a certain age range, since you do not know in advance the range, parameter binding will be most appropriate to use.

There two ways in which parameter binding is applied, namely:

- Positional parameters; and
- Named parameters.

Positional parameters work with the position of a parameter in a query. The syntax used to represent positional parameters is ? followed by a number. For example you can have ?1. This means the position of the parameter in the query statement is 1. For example if we have the age range as follows:

int minAge, and **int maxAge**.

We can have the following query:

```
query =
```

```
SELECT c FROM Customer c WHERE c.age >= ?1 AND c.age <= ?2
```

Subsequently we will have:

```
query.setParameter(1, minAge);  
query.setParameter(2, maxAge);
```

Named parameters use names to position a parameter in the query statement. The syntax used to represent named parameters is the colon symbol, :, followed by the name given to the parameter. For example if we have the age range as follows:

int minAge, and int maxAge.

We can have the following query:

Query =

```
SELECT c FROM Customer c WHERE c.age >= :targetMinAge AND c.age <= :targetMaxAge
```

Subsequently we will have:

```
query.setParameter("targetMinAge", minAge);  
query.setParameter("targetMaxAge", maxAge);
```

9.4 JPQL examples

9.4.1 Dynamic queries

Class Test:

No.	Student number	First name	Last name	Gender	Age	Percentage mark
1	100	Vuyi	Memza	M	21	90%
2	101	Tiro	Nthane	M	18	30%
3	103	Patrice	Mulumba	M	17	70%
4	104	Beauty	Zwane	F	22	40%
5	105	Zuki	Mahlangu	F	23	90%
6	106	Lira	Ranamane	F	25	100%
7	107	Loyi	Sibya	M	20	20%
8	108	Rato	Nthane	F	23	60%
9	109	Tumi	Musenga	M	19	90%
10	200	Tshilo	Baloyi	F	16	50%

Operations:

1. Add records.
2. Display all the records.
3. Display all the records of female students.
4. Display all the records of male students.
5. Display the records of all the students who have passed the test.
6. Display the records of all the students who have failed the test.
7. Display the records (student number, first name and last name) of all female students below 20.
8. Search for a student record using the student number.

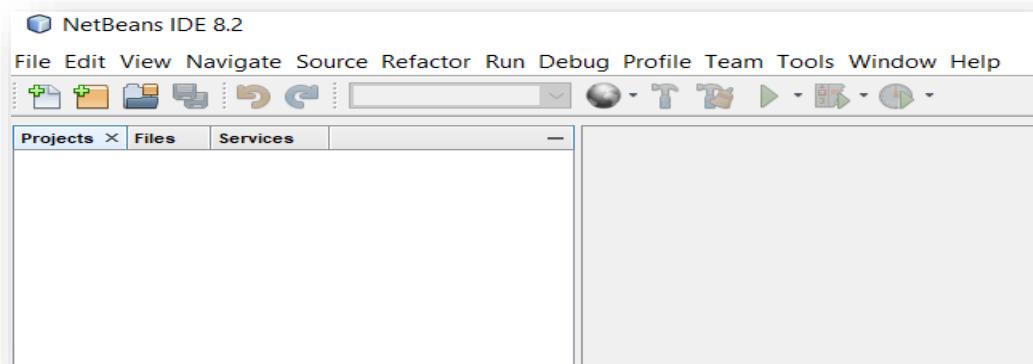
Solution approach

The solution to this problem is going to be done in five parts, namely:

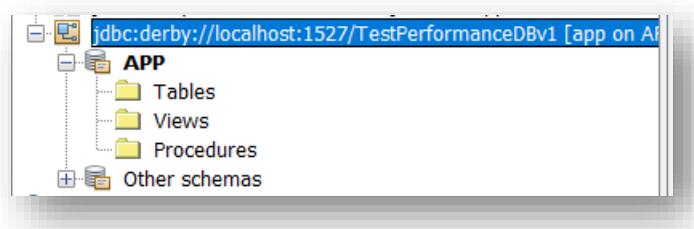
- **Part A:** we create a database.
- **Part B:** we establish connection between the created database and GlassFish.
- **Part C:** we create an EJB module. The module will have an entity with business logic.
- **Part D:** we create a client web application.
- **Part E:** we run the client web application.

Part A - Create a database

Launch NetBeans.



Create a database called **TestPerformanceDBv1**.



Connect the database to the server.

- ✓ Create a connection pool.

A screenshot of the "Edit JDBC Connection Pool Properties" dialog. The tab "General" is selected. A yellow message box at the top right says "New values successfully saved." The pool name is set to "DerbyPool".

Additional Properties (6)		
Select	Name	Value
<input type="checkbox"/>	PortNumber	1527
<input type="checkbox"/>	Password	123
<input type="checkbox"/>	User	app
<input type="checkbox"/>	serverName	localhost
<input type="checkbox"/>	DatabaseName	TestPerformanceDBv1
<input type="checkbox"/>	url	jdbc:derby://localhost:1527/TestPerformance

- ✓ Test if the pool is working.

General	Advanced	Additional Properties
✓ Ping Succeeded		
Edit JDBC Connection Pool Modify an existing JDBC connection pool. A JDBC connection pool is a group of reusable connections for a particular database. <input type="button" value="Load Defaults"/> <input type="button" value="Flush"/> <input type="button" value="Ping"/>		
General Settings <p> Pool Name: DerbyPool Resource Type: javax.sql.DataSource <small>Must be specified if the datasource class implements more than 1 of the interface.</small> Datasource Classname: org.apache.derby.jdbc.ClientDataSource <small>Vendor-specific classname that implements the DataSource and/or XADatasource APIs</small> Driver Classname: <input type="text"/> <small>Vendor-specific classname that implements the java.sql.Driver interface.</small> Ping: <input checked="" type="checkbox"/> Enabled <small>When enabled, the pool is pinged during creation or reconfiguration to identify and warn of any erroneous values for its attributes</small> Deployment Order: 100 <small>Specifies the loading order of the resource at server startup. Lower numbers are loaded first.</small> Description: <input type="text"/> </p>		

- ✓ Check if data source is pointing to the connection pool.

Edit JDBC Resource Edit an existing JDBC data source. <input type="button" value="Load Defaults"/>											
<p> JNDI Name: jdbc/__default Logical JNDI Name: java:comp/DefaultDataSource Pool Name: DerbyPool <input type="button" value=""/></p> <p>Use the JDBC Connection Pools page to create new pools</p> <p> Deployment Order: 100 <small>Specifies the loading order of the resource at server startup. Lower numbers are loaded first.</small> Description: <input type="text"/> Status: <input checked="" type="checkbox"/> Enabled </p>											
Additional Properties (0) <input type="button" value="Add Property"/> <input type="button" value="Delete Properties"/> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Select</th> <th>Name</th> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td colspan="4">No items found.</td> </tr> </tbody> </table>				Select	Name	Value	Description	No items found.			
Select	Name	Value	Description								
No items found.											

Part B – Connect the database to the application server.

Connect the database to the server.

- ✓ Create a connection pool.

The screenshot shows the 'Edit JDBC Connection Pool Properties' page. At the top, there are tabs for General, Advanced, and Additional Properties, with 'Additional Properties' selected. A yellow status bar at the top right says 'New values successfully saved.' Below the tabs, it says 'Edit JDBC Connection Pool Properties' and 'Modify properties of an existing JDBC connection pool.' Under 'Pool Name:', 'DerbyPool' is listed. The 'Additional Properties (6)' section contains a table with columns 'Select', 'Name', and 'Value'. The properties listed are PortNumber (1527), Password (123), User (app), serverName (localhost), DatabaseName (TestPerformanceDBv1), and url (jdbc:derby://localhost:1527/TestPerformance).

- ✓ Test if the pool is working.

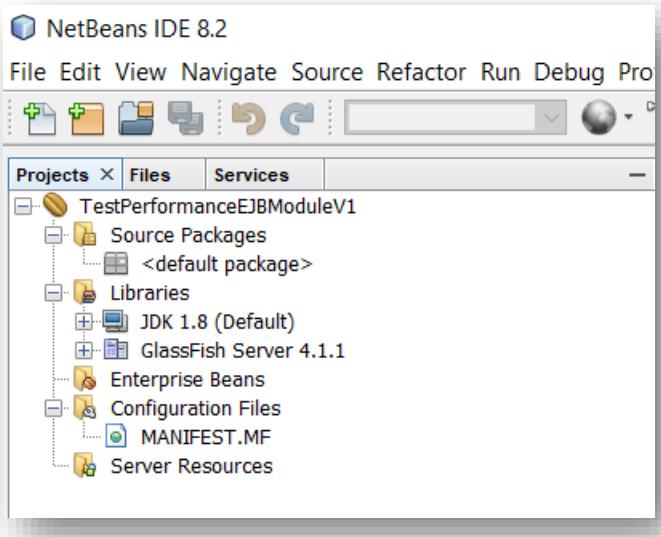
The screenshot shows the 'Edit JDBC Connection Pool' page. At the top, there are tabs for General, Advanced, and Additional Properties, with 'General' selected. A yellow status bar at the top right says 'Ping Succeeded.' Below the tabs, it says 'Edit JDBC Connection Pool' and 'Modify an existing JDBC connection pool. A JDBC connection pool is a group of reusable connections for a particular database.' Under 'General Settings', there are fields for Pool Name (DerbyPool), Resource Type (javax.sql.DataSource), Datasource Classname (org.apache.derby.jdbc.ClientDataSource), Driver Classname (empty), Ping (Enabled checked), Deployment Order (100), and Description (empty). The 'Resource Type' dropdown has a note: 'Must be specified if the datasource class implements more than 1 of the interface.'

- ✓ Check if data source is pointing to the connection pool.

The screenshot shows the 'Edit JDBC Resource' page. At the top, there are tabs for General, Advanced, and Additional Properties, with 'General' selected. A yellow status bar at the top right says 'Load Defaults.' Below the tabs, it says 'Edit JDBC Resource' and 'Edit an existing JDBC data source.' Under 'General' settings, there are fields for JNDI Name (jdbc/__default), Logical JNDI Name (java:comp/DefaultDataSource), Pool Name (DerbyPool), Deployment Order (100), Description (empty), and Status (Enabled checked). The 'Pool Name' dropdown has a note: 'Use the JDBC Connection Pools page to create new pools.' The 'Deployment Order' field has a note: 'Specifies the loading order of the resource at server startup. Lower numbers are loaded first.' The 'Additional Properties (0)' section shows a table with columns 'Select', 'Name', 'Value', and 'Description'. It says 'No items found.'

Part C – Create an entity.

Create an EJB project called **TestPerformanceEJBModuleV1**.



Create an entity called **Student**.

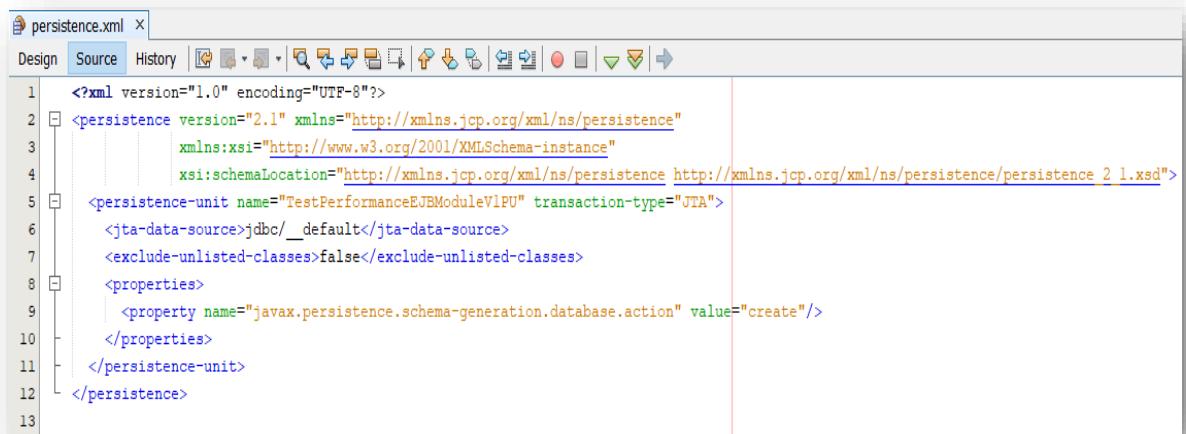
```
1 package za.ac.tut.entities;
2
3 import java.io.Serializable;
4 import javax.persistence.Entity;
5 import javax.persistence.Id;
6
7 /**
8 * @author MemaniV
9 */
10 @Entity
11 public class Student implements Serializable {
12
13     private static final long serialVersionUID = 1L;
14     @Id
15     private Long id;
16     private String firstName;
17     private String lastName;
18     private String gender;
19     private Integer age;
20     private Double percMarkObtained;
21
22     public Student() {
23     }
24
25
26     public Student(Long id, String firstName, String lastName, String gender,
27                     Integer age, Double percMarkObtained) {
28         this.id = id;
29         this.firstName = firstName;
30         this.lastName = lastName;
31         this.gender = gender;
32         this.age = age;
33         this.percMarkObtained = percMarkObtained;
34     }
35 }
```

```
36  □    public String getFirstName() {
37      return firstName;
38  }
39
40  □    public void setFirstName(String firstName) {
41        this.firstName = firstName;
42    }
43
44  □    public String getLastName() {
45        return lastName;
46    }
47
48  □    public void setLastName(String lastName) {
49        this.lastName = lastName;
50    }
51
52  □    public String getGender() {
53        return gender;
54    }
55
56  □    public void setGender(String gender) {
57        this.gender = gender;
58    }
59
60  □    public Integer getAge() {
61        return age;
62    }
63
64  □    public void setAge(Integer age) {
65        this.age = age;
66    }
```

```
68  □    public Double getPercMarkObtained() {
69      return percMarkObtained;
70  }
71
72  □    public void setPercMarkObtained(Double percMarkObtained) {
73        this.percMarkObtained = percMarkObtained;
74    }
75
76  □    public Long getId() {
77        return id;
78    }
79
80  □    public void setId(Long id) {
81        this.id = id;
82    }
83
84    @Override
85  ◎  □    public int hashCode() {
86        int hash = 0;
87        hash += (id != null ? id.hashCode() : 0);
88        return hash;
89    }
```

```
91     @Override
92     public boolean equals(Object object) {
93         if (!(object instanceof Student)) {
94             return false;
95         }
96         Student other = (Student) object;
97         if ((this.id == null && other.id != null) ||
98             (this.id != null && !this.id.equals(other.id))) {
99             return false;
100        }
101        return true;
102    }
103
104    @Override
105    public String toString() {
106        return "za.ac.tut.entities.Student[ id=" + id + " ]";
107    }
108
109 }
```

View the **persistence.xml** file.

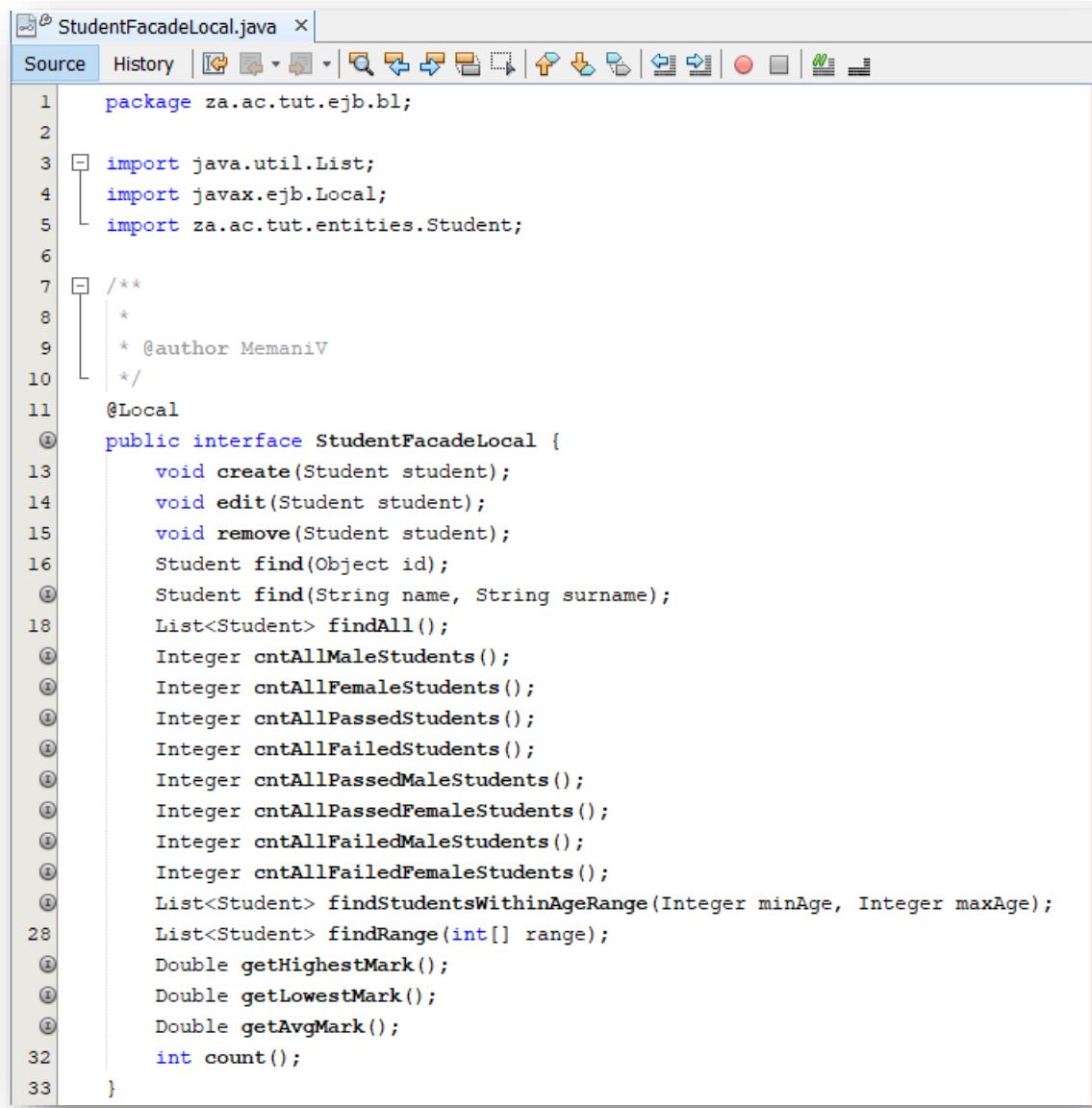


The screenshot shows the Eclipse IDE interface with the 'persistence.xml' file open. The tab bar at the top has 'Design' selected, followed by 'Source'. Below the tabs is a toolbar with various icons for file operations. The main area displays the XML code for the persistence unit:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
    <persistence-unit name="TestPerformanceEJBModuleV1PU" transaction-type="JTA">
        <jta-data-source>jdbc/_default</jta-data-source>
        <exclude-unlisted-classes>false</exclude-unlisted-classes>
        <properties>
            <property name="javax.persistence.schema-generation.database.action" value="create"/>
        </properties>
    </persistence-unit>
</persistence>
```

Create business code (CRUD operations) for the entities

- **StudentFacadeLocal**



The screenshot shows a Java code editor window with the title bar "StudentFacadeLocal.java". The tab bar has "Source" selected. Below the tabs is a toolbar with various icons for file operations like save, open, and search. The code itself is a Java interface named "StudentFacadeLocal" located in the package "za.ac.tut.ejb.bl". It includes imports for List, Local, and Student. The interface contains a Javadoc comment and several methods: create, edit, remove, find, findAll, and various counting methods (cntAllMaleStudents, cntAllFemaleStudents, etc.). There are also methods for finding students within a range and calculating marks.

```
1 package za.ac.tut.ejb.bl;
2
3 import java.util.List;
4 import javax.ejb.Local;
5 import za.ac.tut.entities.Student;
6
7 /**
8 * 
9 * @author MemaniV
10 */
11 @Local
12 public interface StudentFacadeLocal {
13     void create(Student student);
14     void edit(Student student);
15     void remove(Student student);
16     Student find(Object id);
17     Student find(String name, String surname);
18     List<Student> findAll();
19     Integer cntAllMaleStudents();
20     Integer cntAllFemaleStudents();
21     Integer cntAllPassedStudents();
22     Integer cntAllFailedStudents();
23     Integer cntAllPassedMaleStudents();
24     Integer cntAllPassedFemaleStudents();
25     Integer cntAllFailedMaleStudents();
26     Integer cntAllFailedFemaleStudents();
27     List<Student> findStudentsWithinAgeRange(Integer minAge, Integer maxAge);
28     List<Student> findRange(int[] range);
29     Double getHighestMark();
30     Double getLowestMark();
31     Double getAvgMark();
32     int count();
33 }
```

▪ StudentFacade.

```

1 package za.ac.tut.ejb.bl;
2
3 import java.util.List;
4 import javax.ejb.Stateless;
5 import javax.persistence.EntityManager;
6 import javax.persistence.PersistenceContext;
7 import javax.persistence.Query;
8 import za.ac.tut.entities.Student;
9
10 /**
11 * 
12 * @author MemaniV
13 */
14 @Stateless
15 public class StudentFacade extends AbstractFacade<Student> implements StudentFacadeLocal {
16
17     @PersistenceContext(unitName = "TestPerformanceEJBModuleV1PU")
18     private EntityManager em;
19
20     @Override
21     protected EntityManager getEntityManager() {
22         return em;
23     }
24
25     public StudentFacade() {
26         super(Student.class);
27     }

```

```

29     @Override
30     public Integer cntAllFemaleStudents() {
31         Query query = em.createQuery("SELECT count(s) FROM Student s WHERE s.gender = 'F' ORDER BY s.id DESC");
32         Integer cnt = (Integer)query.getSingleResult();
33         return cnt;
34     }
35
36     @Override
37     public Integer cntAllMaleStudents() {
38         Query query = em.createQuery("SELECT count(s) FROM Student s WHERE s.gender = 'M' ORDER BY s.id ASC");
39         Integer cnt = (Integer)query.getSingleResult();
40         return cnt;
41     }
42
43     @Override
44     public Integer cntAllPassedStudents() {
45         Query query = em.createQuery("SELECT count(s) FROM Student s WHERE s.percMarkObtained > 50.0");
46         Integer cnt = (Integer)query.getSingleResult();
47         return cnt;
48     }
49
50     @Override
51     public Integer cntAllPassedMaleStudents() {
52         Query query = em.createQuery("SELECT count(s) FROM Student s WHERE s.percMarkObtained >= 50.0 AND s.gender='M'");
53         Integer cnt = (Integer)query.getSingleResult();
54         return cnt;
55     }
56
57     @Override
58     public Integer cntAllPassedFemaleStudents() {
59         Query query = em.createQuery("SELECT count(s) FROM Student s WHERE s.percMarkObtained >= 50.0 AND s.gender='F'");
60         Integer cnt = (Integer)query.getSingleResult();
61         return cnt;
62     }

```

```

64     @Override
65     public Student find(String firstName, String surname) {
66         Query query = em.createQuery("SELECT s FROM Student s " + " WHERE s.firstName = :fname AND s.lastName = :surname");
67         query.setParameter("fname", firstName);
68         query.setParameter("surname", surname);
69         Student student = (Student)query.getSingleResult();
70         return student;
71     }
72
73     @Override
74     public Integer cntAllFailedStudents() {
75         Query query = em.createQuery("SELECT count(s) FROM Student s WHERE s.percMarkObtained < 50.0");
76         Integer cnt = (Integer)query.getSingleResult();
77         return cnt;
78     }
79
80     @Override
81     public Integer cntAllFailedMaleStudents() {
82         Query query = em.createQuery("SELECT count(s) FROM Student s WHERE s.percMarkObtained < 50.0 AND s.gender='M'");
83         Integer cnt = (Integer)query.getSingleResult();
84         return cnt;
85     }
86
87     @Override
88     public Integer cntAllFailedFemaleStudents() {
89         Query query = em.createQuery("SELECT count(s) FROM Student s WHERE s.percMarkObtained < 50.0 AND s.gender='F'");
90         Integer cnt = (Integer)query.getSingleResult();
91         return cnt;
92     }

```

```

94     @Override
95     public List<Student> findStudentsWithinAgeRange(Integer minAge, Integer maxAge) {
96         Query query = em.createQuery("SELECT s FROM Student s " + " WHERE s.age >= :minTargetAge AND s.age <= :maxTargetAge");
97         query.setParameter("minTargetAge", minAge);
98         query.setParameter("maxTargetAge", maxAge);
99         List<Student> students = query.getResultList();
100        return students;
101    }
102
103    @Override
104    public Double getHighestMark() {
105        Query query = em.createQuery("SELECT MAX(s.percMarkObtained) FROM Student s");
106        Double maxMark = (Double)query.getSingleResult();
107        return maxMark;
108    }
109
110    @Override
111    public Double getLowestMark() {
112        Query query = em.createQuery("SELECT MIN(s.percMarkObtained) FROM Student s");
113        Double minMark = (Double)query.getSingleResult();
114        return minMark;
115    }
116
117    @Override
118    public Double getAvgMark() {
119        Query query = em.createQuery("SELECT AVG(s.percMarkObtained) FROM Student s");
120        Double minMark = (Double)query.getSingleResult();
121        return minMark;
122    }
123
124 }

```

▪ AbstractFacade.

AbstractFacade.java

```

1 package za.ac.tut.bl;
2
3 import java.util.List;
4 import javax.persistence.EntityManager;
5
6 /**
7  * @author MemaniV
8 */
9
10 public abstract class AbstractFacade<T> {
11
12     private Class<T> entityClass;
13
14     public AbstractFacade(Class<T> entityClass) {
15         this.entityClass = entityClass;
16     }
17
18     protected abstract EntityManager getEntityManager();
19
20     public void create(T entity) {
21         getEntityManager().persist(entity);
22     }
23
24     public void edit(T entity) {
25         getEntityManager().merge(entity);
26     }
27
28     public void remove(T entity) {
29         getEntityManager().remove(getEntityManager().merge(entity));
30     }
31

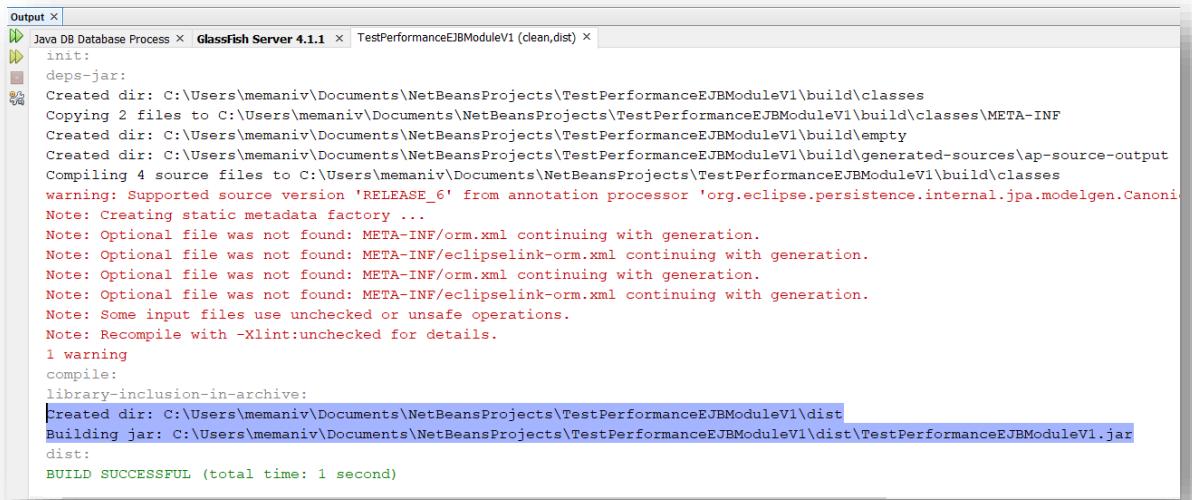
```

```

32     public T find(Object id) {
33         return getEntityManager().find(entityClass, id);
34     }
35
36     public List<T> findAll() {
37         javax.persistence.criteria.CriteriaQuery cq =
38             getEntityManager().getCriteriaBuilder().createQuery();
39         cq.select(cq.from(entityClass));
40         return getEntityManager().createQuery(cq).getResultList();
41     }
42
43     public List<T> findRange(int[] range) {
44         javax.persistence.criteria.CriteriaQuery cq =
45             getEntityManager().getCriteriaBuilder().createQuery();
46         cq.select(cq.from(entityClass));
47         javax.persistence.Query q = getEntityManager().createQuery(cq);
48         q.setMaxResults(range[1] - range[0] + 1);
49         q.setFirstResult(range[0]);
50         return q.getResultList();
51     }
52
53     public int count() {
54         javax.persistence.criteria.CriteriaQuery cq =
55             getEntityManager().getCriteriaBuilder().createQuery();
56         javax.persistence.criteria.Root<T> rt = cq.from(entityClass);
57         cq.select(getEntityManager().getCriteriaBuilder().count(rt));
58         javax.persistence.Query q = getEntityManager().createQuery(cq);
59         return ((Long) q.getSingleResult()).intValue();
60     }
61
62 }

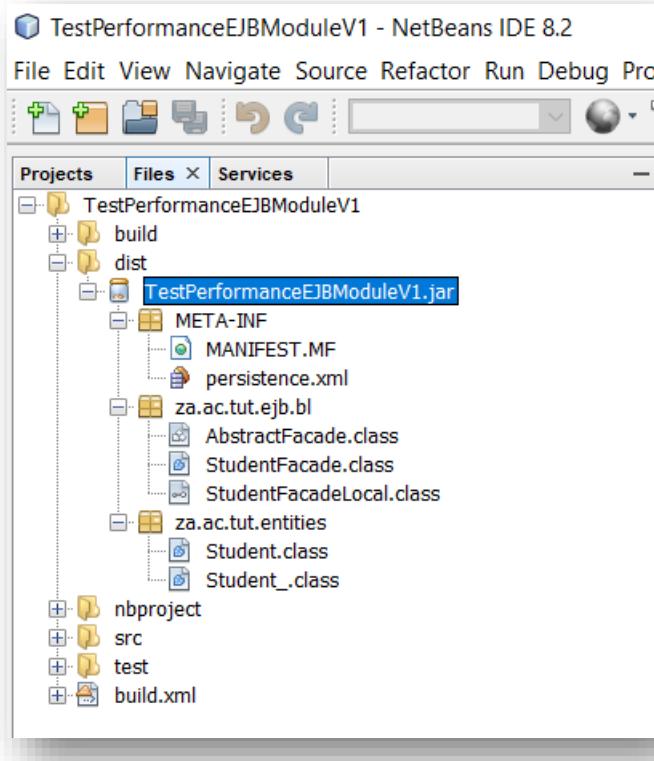
```

✓ Clean and build the EJB project.



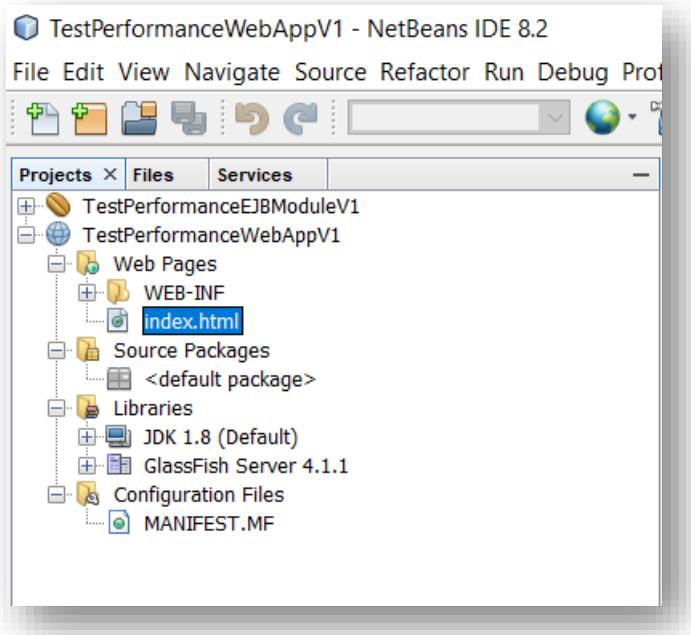
The screenshot shows the NetBeans IDE interface. The top part is the Output window, which displays the build logs for the 'TestPerformanceEJBModuleV1' project. The logs show the process of cleaning, building, and generating sources, resulting in a successful build of the 'TestPerformanceEJBModuleV1.jar' file. The bottom part is the Project Explorer, showing the structure of the project with folders like 'build', 'dist', 'nbproject', 'src', 'test', and 'build.xml', and files like 'MANIFEST.MF' and various class files under 'META-INF' and 'za.ac.tut.ejb.bl'.

```
Output X
Java DB Database Process × GlassFish Server 4.1.1 × TestPerformanceEJBModuleV1 (clean,dist) ×
init:
deps-jar:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\TestPerformanceEJBModuleV1\build\classes
Copying 2 files to C:\Users\memaniv\Documents\NetBeansProjects\TestPerformanceEJBModuleV1\build\classes\META-INF
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\TestPerformanceEJBModuleV1\build\empty
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\TestPerformanceEJBModuleV1\build\generated-sources\ap-source-output
Compiling 4 source files to C:\Users\memaniv\Documents\NetBeansProjects\TestPerformanceEJBModuleV1\build\classes
warning: Supported source version 'RELEASE_6' from annotation processor 'org.eclipse.persistence.internal.jpa.modelgen.Canonicalizer'.
Note: Creating static metadata factory ...
Note: Optional file was not found: META-INF/orm.xml continuing with generation.
Note: Optional file was not found: META-INF/eclipselink-orm.xml continuing with generation.
Note: Optional file was not found: META-INF/orm.xml continuing with generation.
Note: Optional file was not found: META-INF/eclipselink-orm.xml continuing with generation.
Note: Some input files use unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
1 warning
compile:
library-inclusion-in-archive:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\TestPerformanceEJBModuleV1\dist
Building jar: C:\Users\memaniv\Documents\NetBeansProjects\TestPerformanceEJBModuleV1\dist\TestPerformanceEJBModuleV1.jar
dist:
BUILD SUCCESSFUL (total time: 1 second)
```



Part D – Create a web client project.

Create a web client project called **TestPerformanceWebAppV1**.

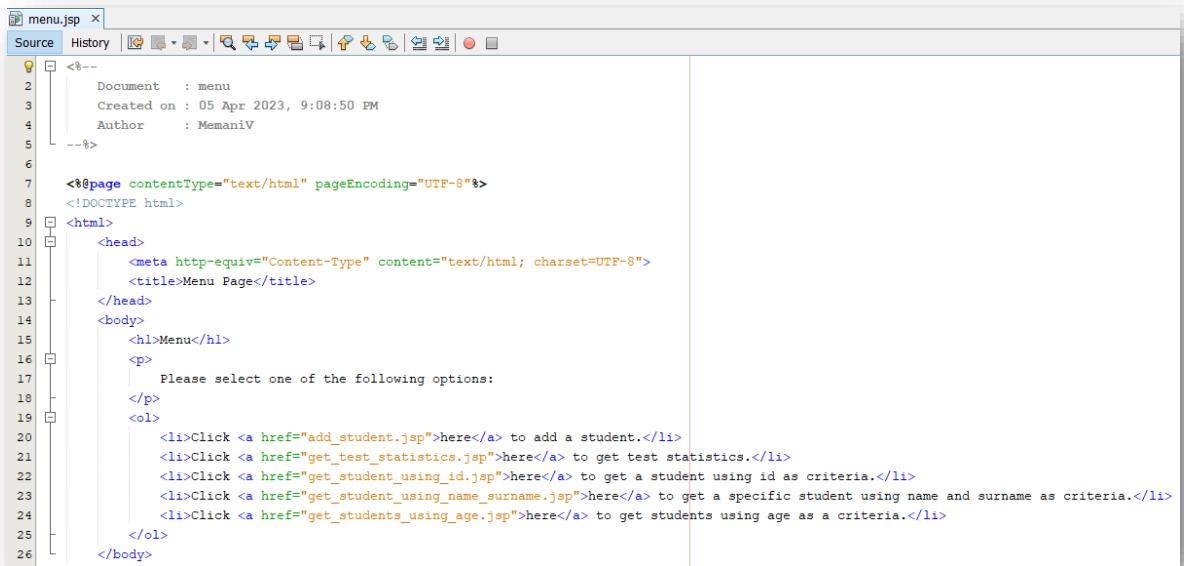


Edit the **index.html** page.

The screenshot shows the NetBeans code editor with the tab "index.html" selected. The window has tabs for Source and History, along with various toolbars. The code editor displays the following HTML content:

```
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 page</h1>
15    <p>
16      Welcome to our web app. Please click <a href="menu.jsp">here</a> to start.
17    </p>
18  </body>
19 </html>
```

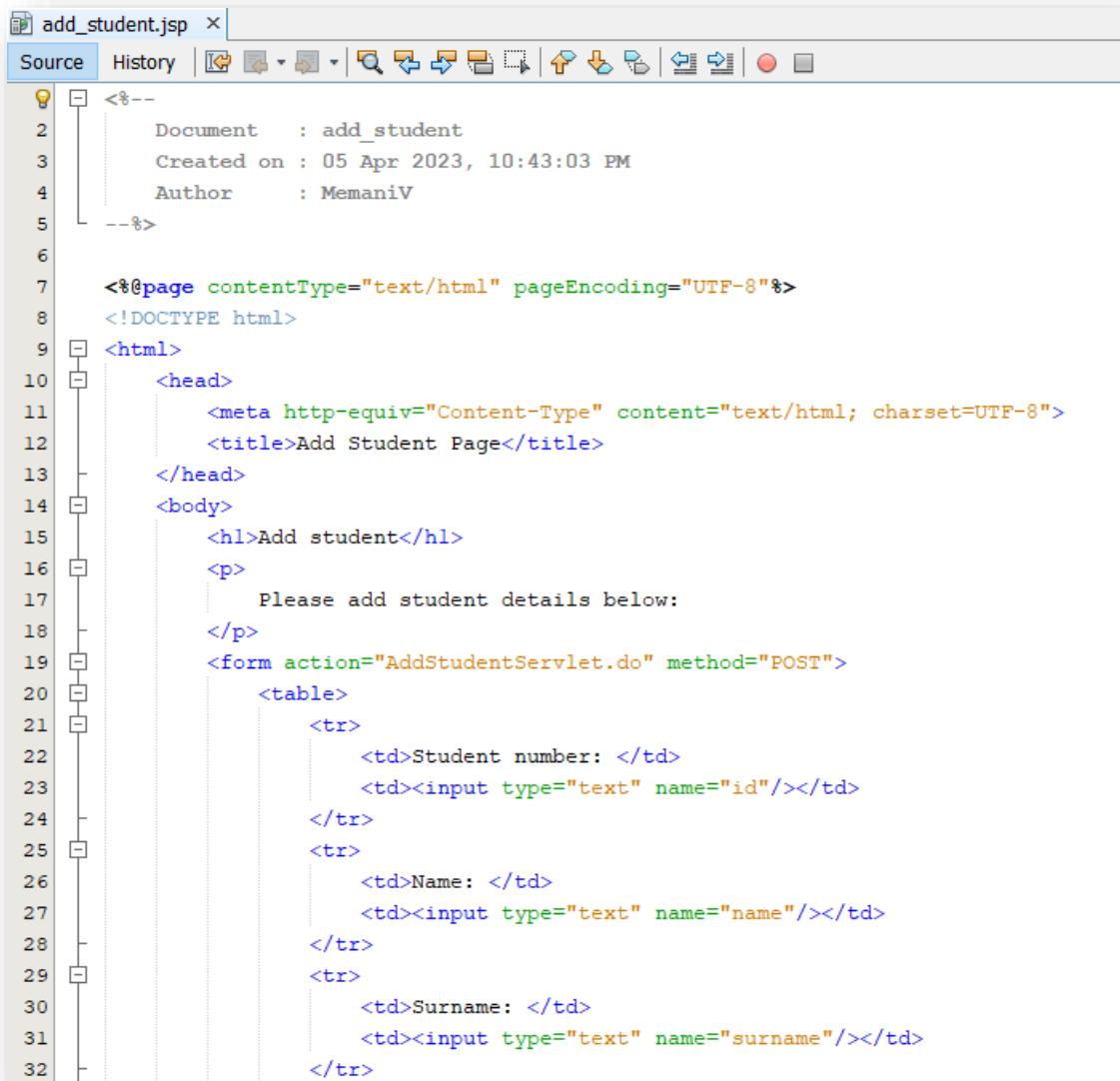
Create the menu.jsp file.



The screenshot shows the code editor of a Java IDE with the file 'menu.jsp' open. The code defines a JSP page named 'menu' with the following content:

```
<%--  
    Document : menu  
    Created on : 05 Apr 2023, 9:08:50 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>Menu Page</title>  
    </head>  
    <body>  
        <h1>Menu</h1>  
        <p>  
            Please select one of the following options:  
        </p>  
        <ol>  
            <li>Click <a href="add_student.jsp">here</a> to add a student.</li>  
            <li>Click <a href="get_test_statistics.jsp">here</a> to get test statistics.</li>  
            <li>Click <a href="get_student_using_id.jsp">here</a> to get a student using id as criteria.</li>  
            <li>Click <a href="get_student_using_name_surname.jsp">here</a> to get a specific student using name and surname as criteria.</li>  
            <li>Click <a href="get_students_using_age.jsp">here</a> to get students using age as a criteria.</li>  
        </ol>  
    </body>
```

Create add_student.jsp file.



The screenshot shows the code editor of a Java IDE with the file 'add_student.jsp' open. The code defines a JSP page named 'add_student' with the following content:

```
<%--  
    Document : add_student  
    Created on : 05 Apr 2023, 10:43:03 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>Add Student Page</title>  
    </head>  
    <body>  
        <h1>Add student</h1>  
        <p>  
            Please add student details below:  
        </p>  
        <form action="AddStudentServlet.do" method="POST">  
            <table>  
                <tr>  
                    <td>Student number:</td>  
                    <td><input type="text" name="id"/></td>  
                </tr>  
                <tr>  
                    <td>Name:</td>  
                    <td><input type="text" name="name"/></td>  
                </tr>  
                <tr>  
                    <td>Surname:</td>  
                    <td><input type="text" name="surname"/></td>  
                </tr>  
            </table>  
        </form>  
    </body>
```

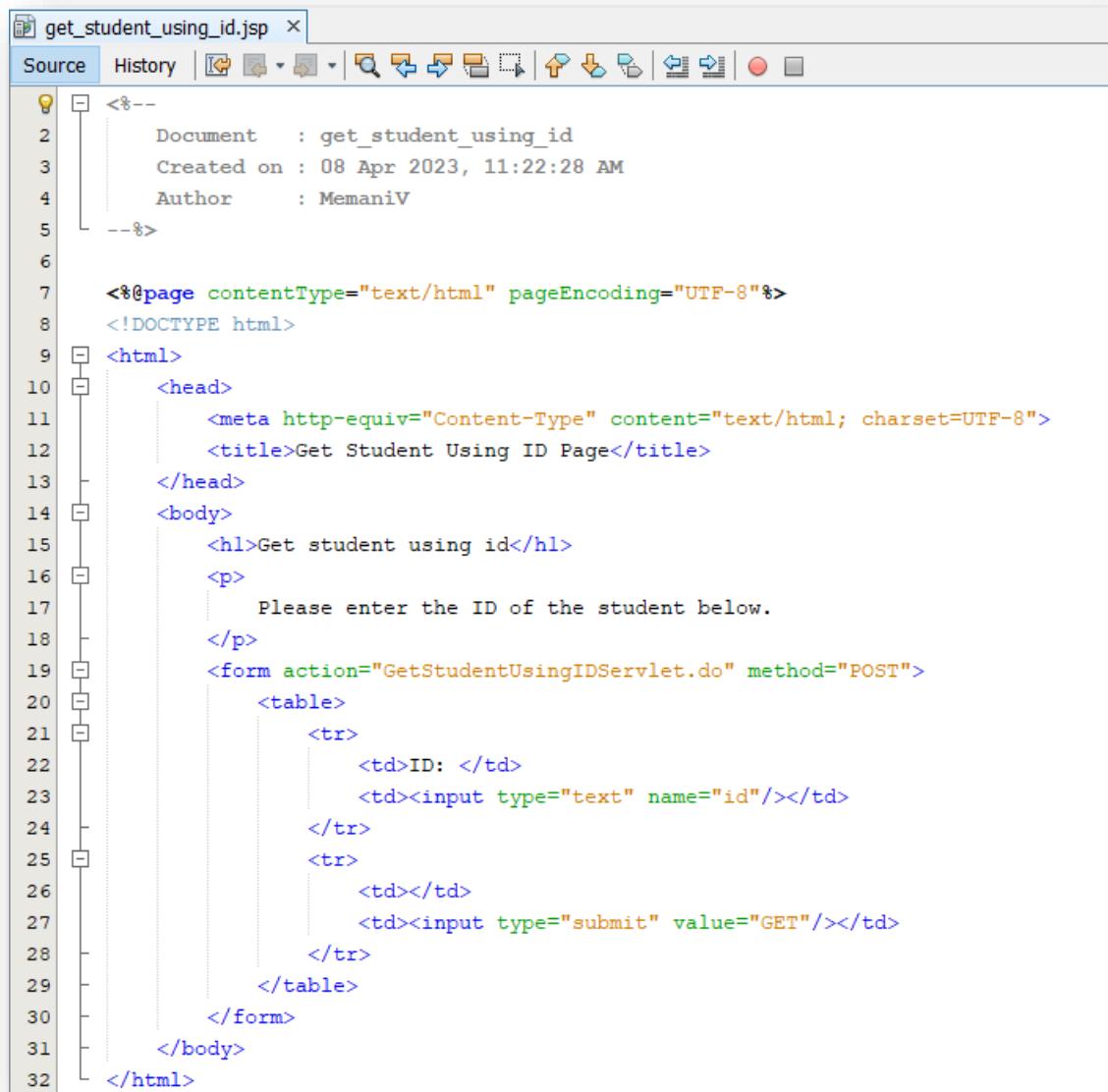
```
33 <tr>
34     <td>Age: </td>
35     <td><input type="text" name="age"/></td>
36 </tr>
37 <tr>
38     <td>Gender: </td>
39     <td>
40         <select name="gender">
41             <option value="F">F</option>
42             <option value="M">M</option>
43         </select>
44     </td>
45 </tr>
46 <tr>
47     <td>Mark obtained: </td>
48     <td><input type="text" name="percMarkObtained"/></td>
49 </tr>
50 <tr>
51     <td></td>
52     <td><input type="submit" value="ADD STUDENT"/></td>
53 </tr>
54 </table>
55 </form>
56 </body>
57 </html>
```

Create the `get_test_statistics.jsp` file.

The screenshot shows a Java code editor with the file 'get_test_statistics.jsp' open. The code is a JSP page that displays test statistics. It includes a header with document information, a meta tag for content type, a title, and a form with a submit button.

```
<%--  
    Document      : get_test_statistics  
    Created on   : 06 Apr 2023, 7:29:45 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>Get Test Statistics Page</title>  
    </head>  
    <body>  
        <h1>Get test statistics</h1>  
        <p>  
            Please click on the button below to get test statistics.  
        </p>  
        <form action="GetTestStatisticsServlet.do" method="GET">  
            <table>  
                <tr>  
                    <td></td>  
                    <td><input type="submit" value="GET TEST STATISTICS"/></td>  
                </tr>  
            </table>  
        </form>  
    </body>  
</html>
```

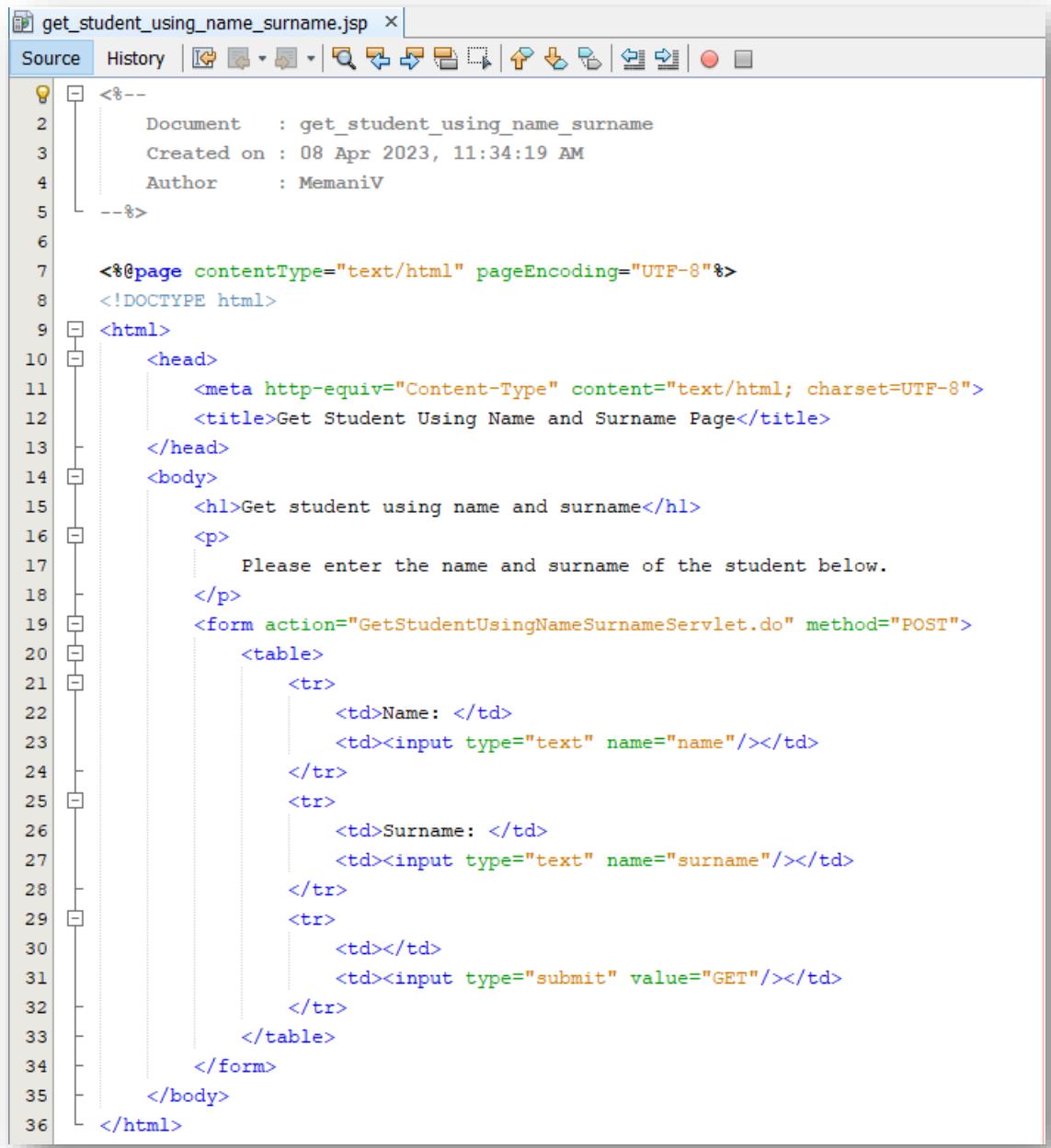
Create the get_student_using_id.jsp file.



The screenshot shows a Java IDE interface with a code editor window titled "get_student_using_id.jsp". The code is a JSP page that displays an HTML form for getting a student by ID. The code includes JSP comments, HTML tags, and a form with two input fields: one for the student ID and one for submission.

```
<%--  
1 Document      : get_student_using_id  
2 Created on   : 08 Apr 2023, 11:22:28 AM  
3 Author        : MemaniV  
4--%>  
  
<%@page contentType="text/html" pageEncoding="UTF-8"%>  
<!DOCTYPE html>  
<html>  
    <head>  
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
        <title>Get Student Using ID Page</title>  
    </head>  
    <body>  
        <h1>Get student using id</h1>  
        <p>  
            Please enter the ID of the student below.  
        </p>  
        <form action="GetStudentUsingIDServlet.do" method="POST">  
            <table>  
                <tr>  
                    <td>ID: </td>  
                    <td><input type="text" name="id"/></td>  
                </tr>  
                <tr>  
                    <td></td>  
                    <td><input type="submit" value="GET"/></td>  
                </tr>  
            </table>  
        </form>  
    </body>  
</html>
```

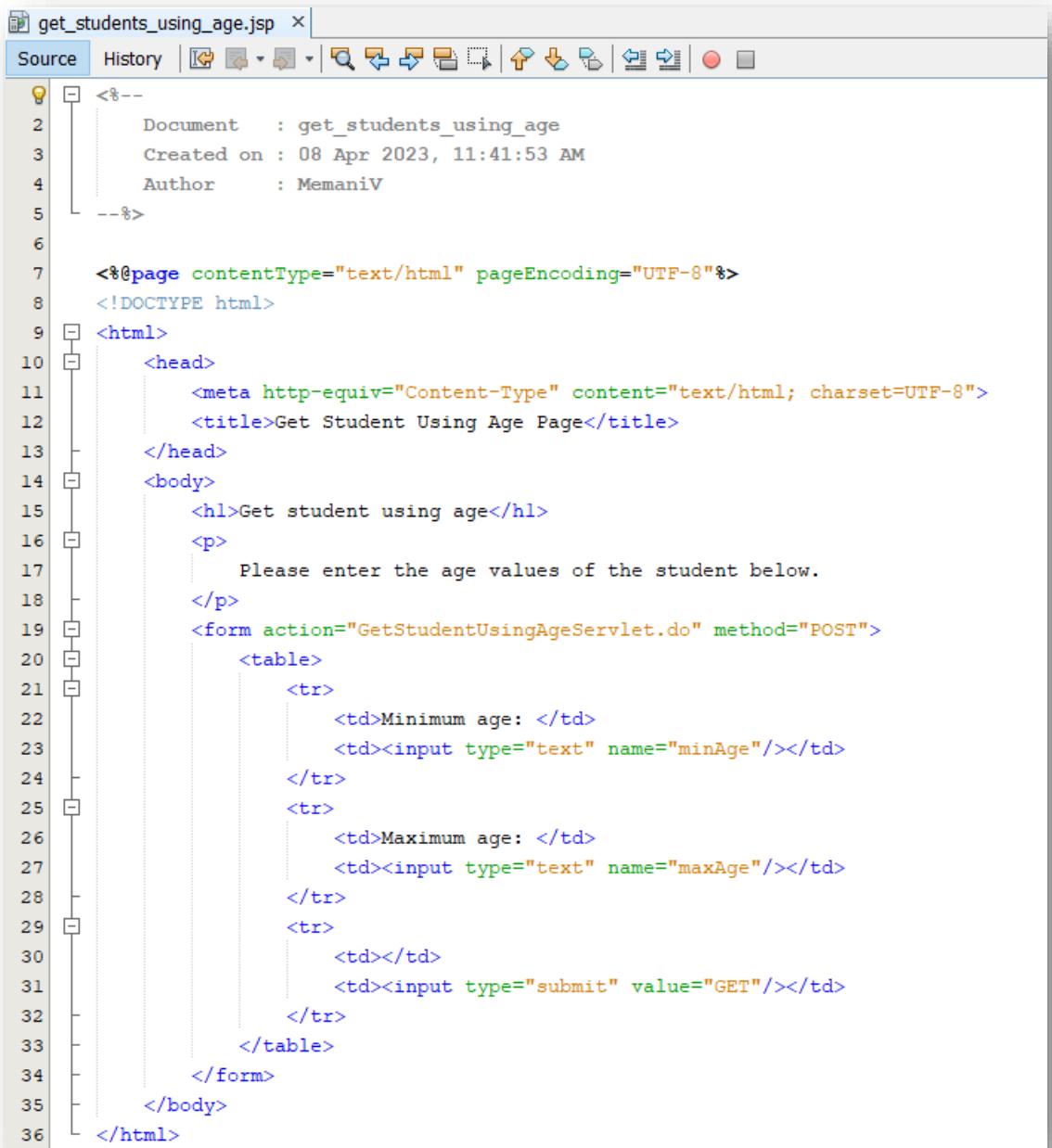
Create the get_student_using_name_surname.jsp file.



The screenshot shows a Java IDE interface with the file "get_student_using_name_surname.jsp" open. The code is a JSP page that displays a form for entering a student's name and surname. The code includes JSP comments, HTML tags, and a form with two input fields and a submit button.

```
<%--  
1 Document      : get_student_using_name_surname  
2 Created on   : 08 Apr 2023, 11:34:19 AM  
3 Author        : MemaniV  
4--%>  
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>Get Student Using Name and Surname Page</title>  
13 </head>  
14 <body>  
15     <h1>Get student using name and surname</h1>  
16     <p>  
17         Please enter the name and surname of the student below.  
18     </p>  
19     <form action="GetStudentUsingNameSurnameServlet.do" method="POST">  
20         <table>  
21             <tr>  
22                 <td>Name: </td>  
23                 <td><input type="text" name="name"/></td>  
24             </tr>  
25             <tr>  
26                 <td>Surname: </td>  
27                 <td><input type="text" name="surname"/></td>  
28             </tr>  
29             <tr>  
30                 <td></td>  
31                 <td><input type="submit" value="GET"/></td>  
32             </tr>  
33         </table>  
34     </form>  
35 </body>  
36 </html>
```

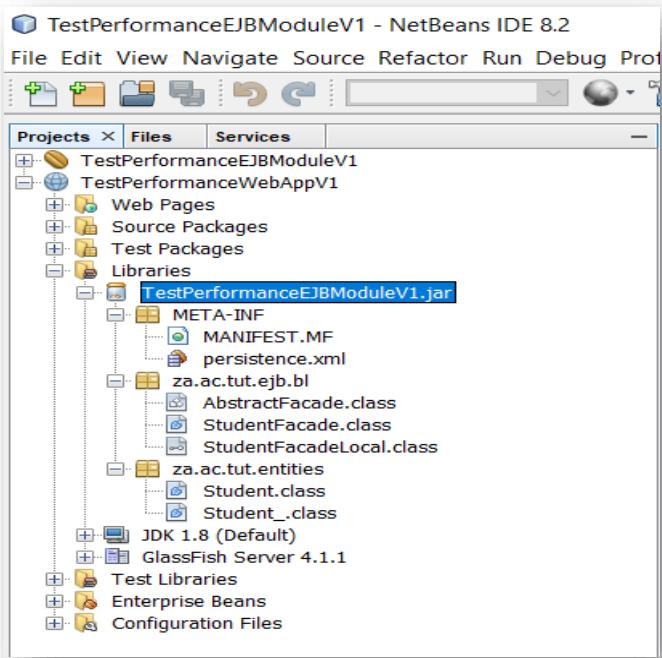
Create the get_student_using_age.jsp file.



The screenshot shows a Java IDE interface with the file "get_students_using_age.jsp" open. The code is a JSP page that displays a form for entering student age values. The code includes JSP comments, HTML tags, and a form action pointing to "GetStudentUsingAgeServlet.do".

```
<%--  
1 Document      : get_students_using_age  
2 Created on   : 08 Apr 2023, 11:41:53 AM  
3 Author        : MemaniV  
4--%>  
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>Get Student Using Age Page</title>  
13 </head>  
14 <body>  
15     <h1>Get student using age</h1>  
16     <p>  
17         Please enter the age values of the student below.  
18     </p>  
19     <form action="GetStudentUsingAgeServlet.do" method="POST">  
20         <table>  
21             <tr>  
22                 <td>Minimum age: </td>  
23                 <td><input type="text" name="minAge"/></td>  
24             </tr>  
25             <tr>  
26                 <td>Maximum age: </td>  
27                 <td><input type="text" name="maxAge"/></td>  
28             </tr>  
29             <tr>  
30                 <td></td>  
31                 <td><input type="submit" value="GET"/></td>  
32             </tr>  
33         </table>  
34     </form>  
35 </body>  
36 </html>
```

Add the **TestPerformanceEJBModuleV1.jar** file to the library of **TestPerformanceWebAppV1**.



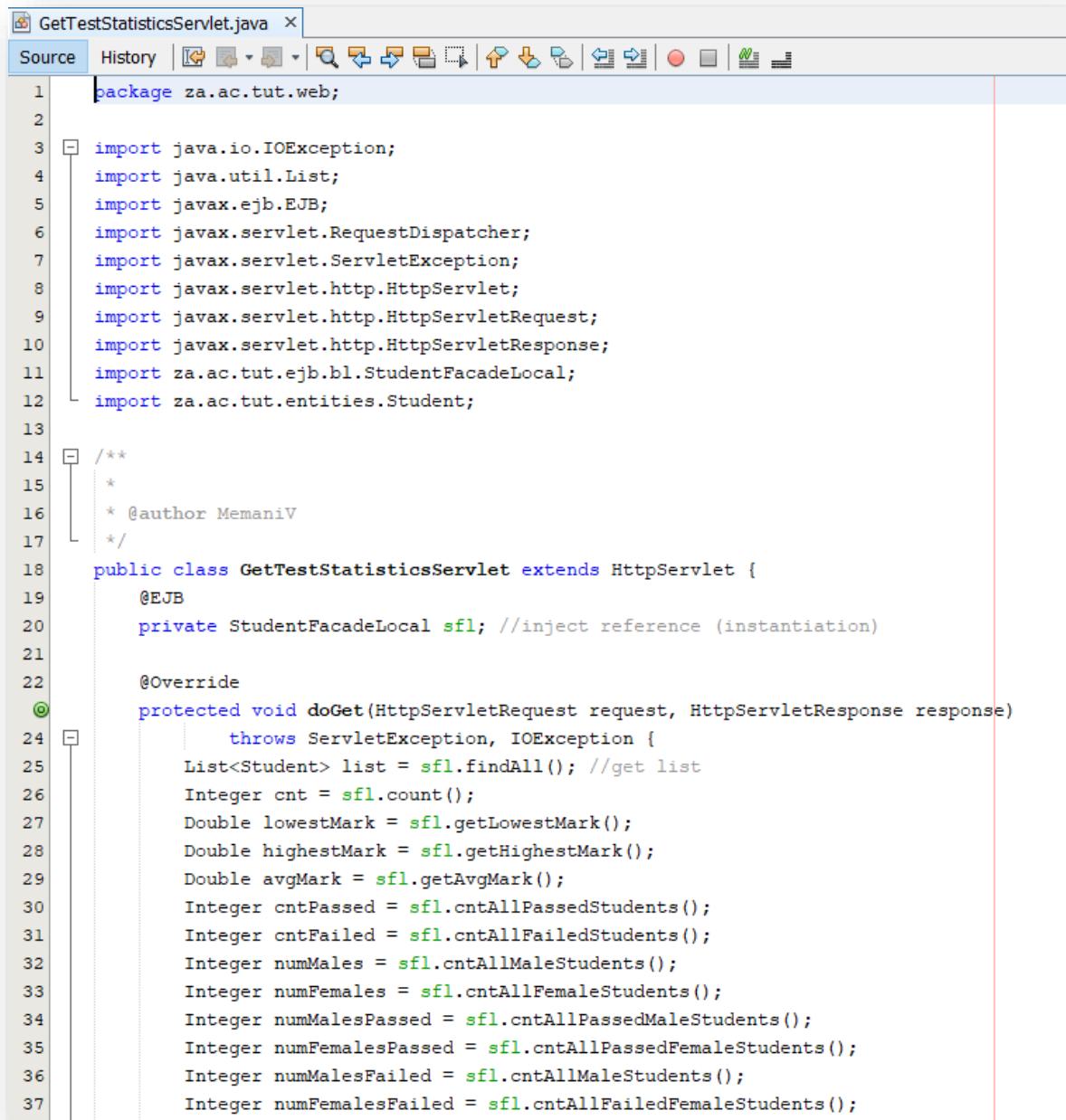
Create the **AddStudentServlet.java** file.

```
1 package za.ac.tut.web;
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.StudentFacadeLocal;
11 import za.ac.tut.entities.Student;
12
13 /**
14 * @author MemaniV
15 */
16
17 public class AddStudentServlet extends HttpServlet {
18     @EJB
19     private StudentFacadeLocal pfl; //inject reference (instantiation)
20     @Override
21     protected void doPost(HttpServletRequest request, HttpServletResponse response)
22             throws ServletException, IOException {
23         Long id = Long.parseLong(request.getParameter("id"));
24         String name = request.getParameter("name");
25         String surname = request.getParameter("surname");
26         String gender = request.getParameter("gender");
27         Integer age = Integer.parseInt(request.getParameter("age"));
28         Double percMarkObtained = Double.parseDouble(request.getParameter("percMarkObtained"));
29
30         Student student = createStudent(id, name, surname, gender, age, percMarkObtained); //create student
31         pfl.create(student); //persist student
32
33         RequestDispatcher disp = request.getRequestDispatcher("add_student_outcome.jsp");
34         disp.forward(request, response);
35     }
}
```

The screenshot shows the NetBeans code editor for the file "AddStudentServlet.java". The code is written in Java and defines a servlet that extends "HttpServlet". It uses CDI annotations like "@EJB" to inject a "StudentFacadeLocal" bean named "pfl". The "doPost" method retrieves parameters from the request (id, name, surname, gender, age, percMarkObtained), creates a "Student" object using these parameters, and then calls the "create" method of the injected bean to persist the student. Finally, it forwards the request to a JSP page named "add_student_outcome.jsp". The code editor has syntax highlighting and various toolbars at the top.

```
37     }
38     private Student createStudent(Long id, String name, String surname, String gender, Integer age, Double percMarkObtained) {
39         Student student = new Student();
40         student.setId(id);
41         student.setFirstName(name);
42         student.setLastName(surname);
43         student.setGender(gender);
44         student.setAge(age);
45         student.setPercMarkObtained(percMarkObtained);
46         return student;
47     }
```

Create the **GetTestStatisticsServlet.java** file.



The screenshot shows an IDE interface with the tab 'Source' selected. The code editor displays the `GetTestStatisticsServlet.java` file. The code implements a servlet that interacts with a StudentFacadeLocal EJB to retrieve statistics about students.

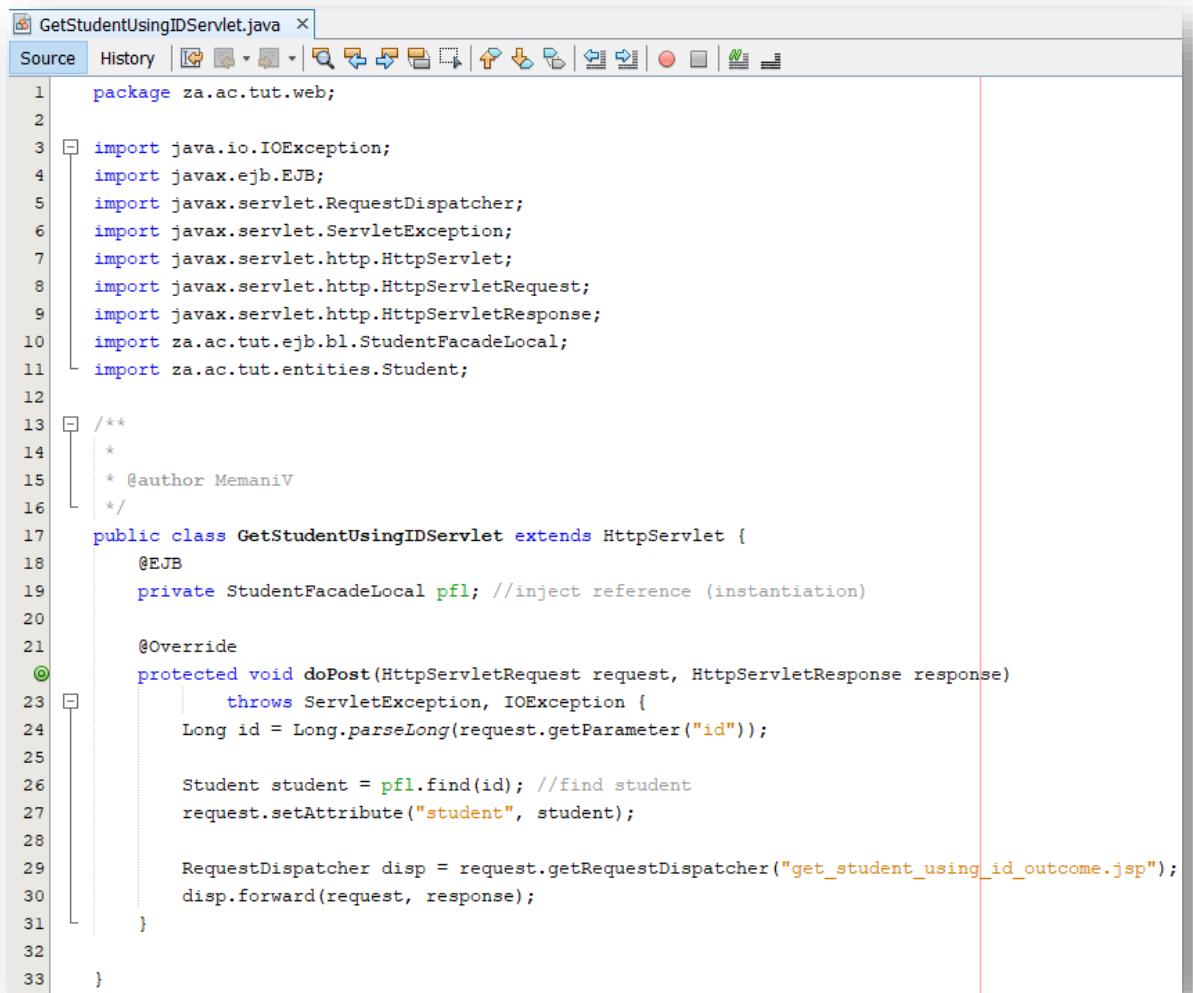
```
1 package za.ac.tut.web;
2
3 import java.io.IOException;
4 import java.util.List;
5 import javax.ejb.EJB;
6 import javax.servlet.DispatcherType;
7 import javax.servlet.ServletException;
8 import javax.servlet.http.HttpServlet;
9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11 import za.ac.tut.ejb.bl.StudentFacadeLocal;
12 import za.ac.tut.entities.Student;
13
14 /**
15 * @author MemaniV
16 */
17 public class GetTestStatisticsServlet extends HttpServlet {
18     @EJB
19     private StudentFacadeLocal sfl; //inject reference (instantiation)
20
21     @Override
22     protected void doGet(HttpServletRequest request, HttpServletResponse response)
23             throws ServletException, IOException {
24         List<Student> list = sfl.findAll(); //get list
25         Integer cnt = sfl.count();
26         Double lowestMark = sfl.getLowestMark();
27         Double highestMark = sfl.getHighestMark();
28         Double avgMark = sfl.getAvgMark();
29         Integer cntPassed = sfl.cntAllPassedStudents();
30         Integer cntFailed = sfl.cntAllFailedStudents();
31         Integer numMales = sfl.cntAllMaleStudents();
32         Integer numFemales = sfl.cntAllFemaleStudents();
33         Integer numMalesPassed = sfl.cntAllPassedMaleStudents();
34         Integer numFemalesPassed = sfl.cntAllPassedFemaleStudents();
35         Integer numMalesFailed = sfl.cntAllMaleStudents();
36         Integer numFemalesFailed = sfl.cntAllFailedFemaleStudents();
37     }
}
```

```

39     request.setAttribute("list", list);
40     request.setAttribute("cnt", cnt);
41     request.setAttribute("lowestMark", lowestMark);
42     request.setAttribute("highestMark", highestMark);
43     request.setAttribute("avgMark", avgMark);
44     request.setAttribute("cntPassed", cntPassed);
45     request.setAttribute("cntFailed", cntFailed);
46     request.setAttribute("numMales", numMales);
47     request.setAttribute("numFemales", numFemales);
48     request.setAttribute("numMalesPassed", numMalesPassed);
49     request.setAttribute("numFemalesPassed", numFemalesPassed);
50     request.setAttribute("numMalesFailed", numMalesFailed);
51     request.setAttribute("numFemalesFailed", numFemalesFailed);
52
53     RequestDispatcher disp = request.getRequestDispatcher("get_test_statistics_outcome.jsp");
54     disp.forward(request, response);
55 }
56
57 }

```

Create the **GetStudentUsingIDServlet.java** file.



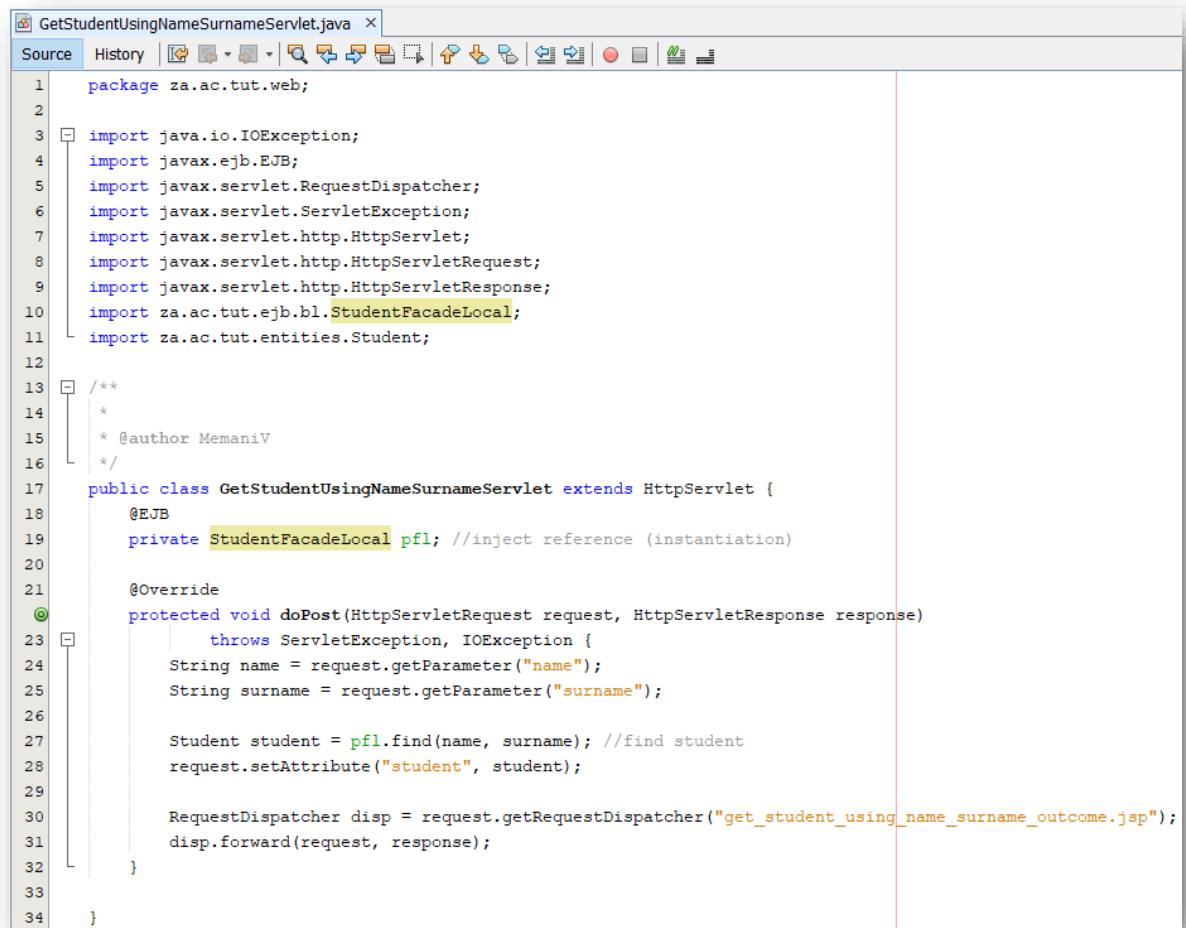
The screenshot shows an IDE interface with the tab "GetStudentUsingIDServlet.java" selected. The code editor displays Java code for a servlet. The code includes imports for various Java packages, annotations like @EJB and @Override, and logic for retrieving a student by ID from a database and forwarding the request to a JSP page. The code is well-formatted with line numbers and syntax highlighting.

```

1 package za.ac.tut.web;
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.StudentFacadeLocal;
11 import za.ac.tut.entities.Student;
12
13 /**
14 * 
15 * @author MemaniV
16 */
17 public class GetStudentUsingIDServlet extends HttpServlet {
18     @EJB
19     private StudentFacadeLocal pfl; //inject reference (instantiation)
20
21     @Override
22     protected void doPost(HttpServletRequest request, HttpServletResponse response)
23             throws ServletException, IOException {
24         Long id = Long.parseLong(request.getParameter("id"));
25
26         Student student = pfl.find(id); //find student
27         request.setAttribute("student", student);
28
29         RequestDispatcher disp = request.getRequestDispatcher("get_student_using_id_outcome.jsp");
30         disp.forward(request, response);
31     }
32
33 }

```

Create the **GetStudentUsingNameSurnameServlet.java** file.



The screenshot shows a Java code editor window with the title "GetStudentUsingNameSurnameServlet.java". The code is a Java servlet named "GetStudentUsingNameSurnameServlet" that extends "HttpServlet". It imports various Java packages and annotations, including "java.io.IOException", "javax.ejb.EJB", "javax.servlet.RequestDispatcher", "javax.servlet.ServletException", "javax.servlet.http.HttpServlet", "javax.servlet.http.HttpServletRequest", "javax.servlet.http.HttpServletResponse", "za.ac.tut.ejb.bl.StudentFacadeLocal", and "za.ac.tut.entities.Student". The class contains a constructor with a private field "StudentFacadeLocal pfl" annotated with "@EJB". The "doPost" method overrides the "HttpServlet" method. It retrieves parameters "name" and "surname" from the request, finds a student using the "pfl.find" method, sets the student as an attribute in the request, and then forwards the request to a JSP page named "get_student_using_name_surname_outcome.jsp". The code is annotated with a copyright notice and author information: "/* * @author MemaniV */".

```
1 package za.ac.tut.web;
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.StudentFacadeLocal;
11 import za.ac.tut.entities.Student;
12
13 /**
14  * @author MemaniV
15 */
16
17 public class GetStudentUsingNameSurnameServlet extends HttpServlet {
18     @EJB
19     private StudentFacadeLocal pfl; //inject reference (instantiation)
20
21     @Override
22     protected void doPost(HttpServletRequest request, HttpServletResponse response)
23             throws ServletException, IOException {
24         String name = request.getParameter("name");
25         String surname = request.getParameter("surname");
26
27         Student student = pfl.find(name, surname); //find student
28         request.setAttribute("student", student);
29
30         RequestDispatcher disp = request.getRequestDispatcher("get_student_using_name_surname_outcome.jsp");
31         disp.forward(request, response);
32     }
33 }
34 }
```

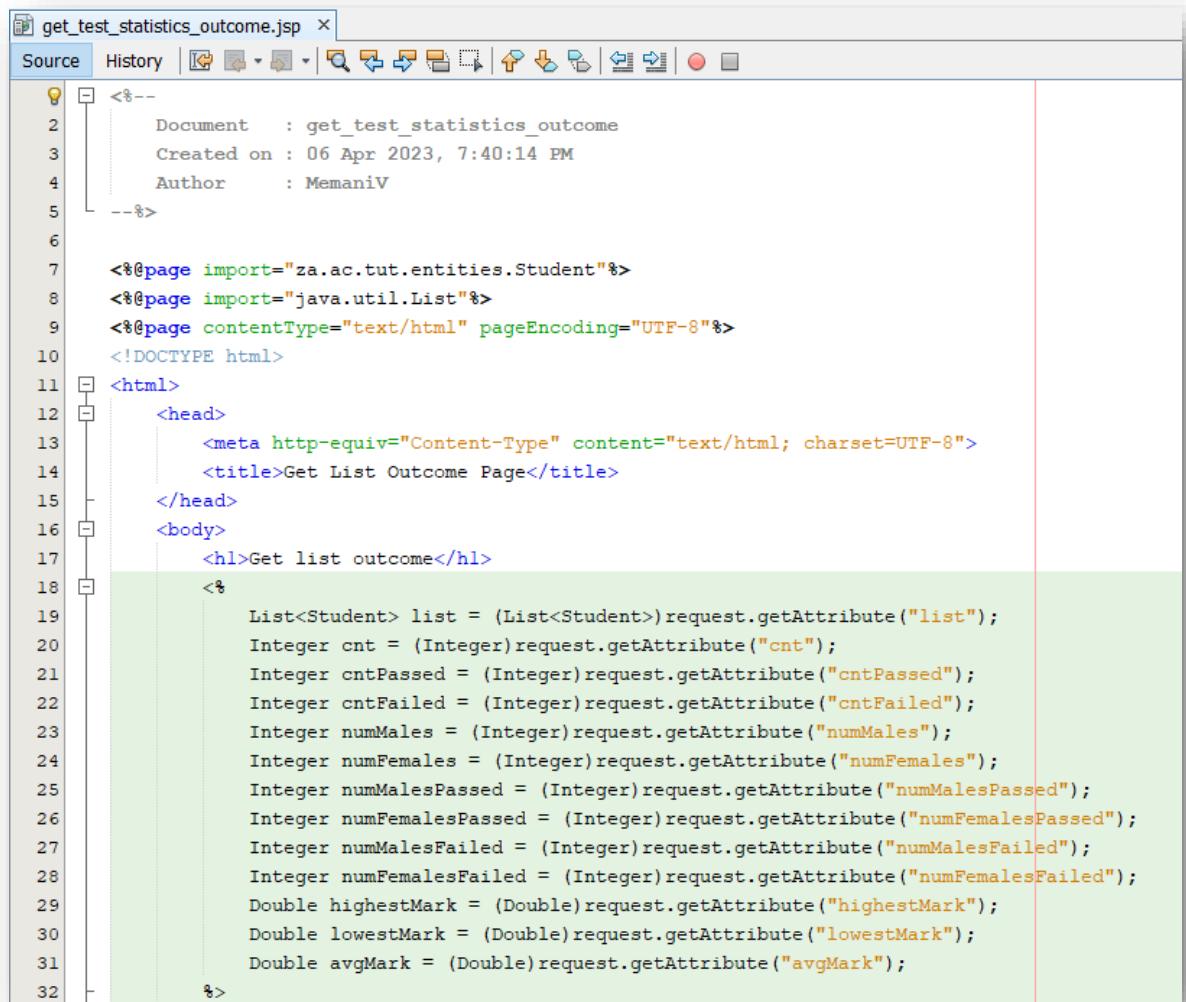
Create the **GetStudentUsingAgeServlet.java** file.

Create the **add_student_outcome.jsp** file.

The screenshot shows a Java IDE interface with the title bar "add_student_outcome.jsp x". The menu bar includes "Source", "History", and various file operations. The code editor displays the JSP file content:

```
<%--  
1 Document : add_student_outcome  
2 Created on : 08 Apr 2023, 11:20:29 AM  
3 Author : MemaniIV  
4  
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>Add Student Outcome Page</title>  
13 </head>  
14 <body>  
15     <h1>Add student outcome</h1>  
16     <p>  
17         The student has been successfully persisted into the database.  
18         Please click <a href="menu.jsp">here</a> to get back to the menu page or <a href="index.html">here</a>  
19         to the main page.  
20     </p>  
21 </body>  
22 </html>
```

Create the `get_test_statistics_outcome.jsp` file.



The screenshot shows a Java IDE interface with the file `get_test_statistics_outcome.jsp` open. The code is a JSP page that imports Student and List classes, sets the content type to text/html, and displays a list of student statistics. The statistics include counts of students, passed students, failed students, males, females, and their respective counts. It also calculates the highest, lowest, and average marks.

```
<%--  
1 Document : get_test_statistics_outcome  
2 Created on : 06 Apr 2023, 7:40:14 PM  
3 Author : MemaniV  
4--%>  
  
5  
6  
7 <%@page import="za.ac.tut.entities.Student"%>  
8 <%@page import="java.util.List"%>  
9 <%@page contentType="text/html" pageEncoding="UTF-8"%>  
10 <!DOCTYPE html>  
11 <html>  
12 <head>  
13 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
14 <title>Get List Outcome Page</title>  
15 </head>  
16 <body>  
17 <h1>Get list outcome</h1>  
18 <%  
19 List<Student> list = (List<Student>) request.getAttribute("list");  
20 Integer cnt = (Integer) request.getAttribute("cnt");  
21 Integer cntPassed = (Integer) request.getAttribute("cntPassed");  
22 Integer cntFailed = (Integer) request.getAttribute("cntFailed");  
23 Integer numMales = (Integer) request.getAttribute("numMales");  
24 Integer numFemales = (Integer) request.getAttribute("numFemales");  
25 Integer numMalesPassed = (Integer) request.getAttribute("numMalesPassed");  
26 Integer numFemalesPassed = (Integer) request.getAttribute("numFemalesPassed");  
27 Integer numMalesFailed = (Integer) request.getAttribute("numMalesFailed");  
28 Integer numFemalesFailed = (Integer) request.getAttribute("numFemalesFailed");  
29 Double highestMark = (Double) request.getAttribute("highestMark");  
30 Double lowestMark = (Double) request.getAttribute("lowestMark");  
31 Double avgMark = (Double) request.getAttribute("avgMark");  
32 %>
```

```

33 |   <table>
34 |     <tr>
35 |       <td>Number of students that wrote the test:</td>
36 |       <td><%=cnt%></td>
37 |     </tr> |
38 |     <tr>
39 |       <td>Number of students that passed:</td>
40 |       <td><%=cntPassed%></td>
41 |     </tr>
42 |     <tr>
43 |       <td>Number of students that failed:</td>
44 |       <td><%=cntFailed%></td>
45 |     </tr>
46 |     <tr>
47 |       <td>Number of male students:</td>
48 |       <td><%=numMales%></td>
49 |     </tr>
50 |     <tr>
51 |       <td>Number of female students:</td>
52 |       <td><%=numFemales%></td>
53 |     </tr>
54 |     <tr>
55 |       <td>Number of male students that passed:</td>
56 |       <td><%=numMalesPassed%></td>
57 |     </tr>
58 |     <tr>
59 |       <td>Number of female students that passed:</td>
60 |       <td><%=numFemalesPassed%></td>
61 |     </tr>
62 |     <tr>
63 |       <td>Number of male students that failed:</td>
64 |       <td><%=numMalesFailed%></td>
65 |     </tr>
66 |     <tr>
67 |       <td>Number of female students that failed:</td>
68 |       <td><%=numFemalesFailed%></td>
69 |     </tr>

```

```

70 |   <tr>
71 |     <td>Highest mark:</td>
72 |     <td><%=highestMark%></td>
73 |   </tr>
74 |   <tr>
75 |     <td>Lowest mark:</td>
76 |     <td><%=lowestMark%></td>
77 |   </tr>
78 |   <tr>
79 |     <td>Average mark:</td>
80 |     <td><%=avgMark%></td>
81 |   </tr>
82 | </table>
83 | <p>
84 |   Below is the class list:
85 | </p>

```

```
86 |     <table>
87 |         <%
88 |             for(int i = 0; i < list.size(); i++){
89 |                 Student p = list.get(i);
90 |                 Long id = p.getId();
91 |                 String name = p.getFirstName();
92 |                 String surname = p.getLastName();
93 |                 String gender = p.getGender();
94 |                 Integer age = p.getAge();
95 |                 Double percMarkObtained = p.getPercMarkObtained();
96 |
97 |             </tr>
98 |             <td>ID:</td>
99 |             <td><%=id%></td>
100 |
101 |             <tr>
102 |                 <td>Name:</td>
103 |                 <td><%=name%></td>
104 |
105 |             <tr>
106 |                 <td>Surname:</td>
107 |                 <td><%=surname%></td>
108 |
109 |             <tr>
110 |                 <td>Age:</td>
111 |                 <td><%=age%></td>
112 |
113 |             <tr>
114 |                 <td>Gender:</td>
115 |                 <td><%=gender%></td>
116 |
117 |             <tr>
118 |                 <td>Mark obtained:</td>
119 |                 <td><%=percMarkObtained%></td>
120 |             </tr>
```

```
122 |         <%
123 |             }
124 |         </table>
125 |         <p>
126 |             Please click <a href="menu.jsp">here</a> to get back to the menu page or <a href="index.html">here</a>
127 |             to the main page.
128 |         </p>
129 |     </body>
130 | </html>
```

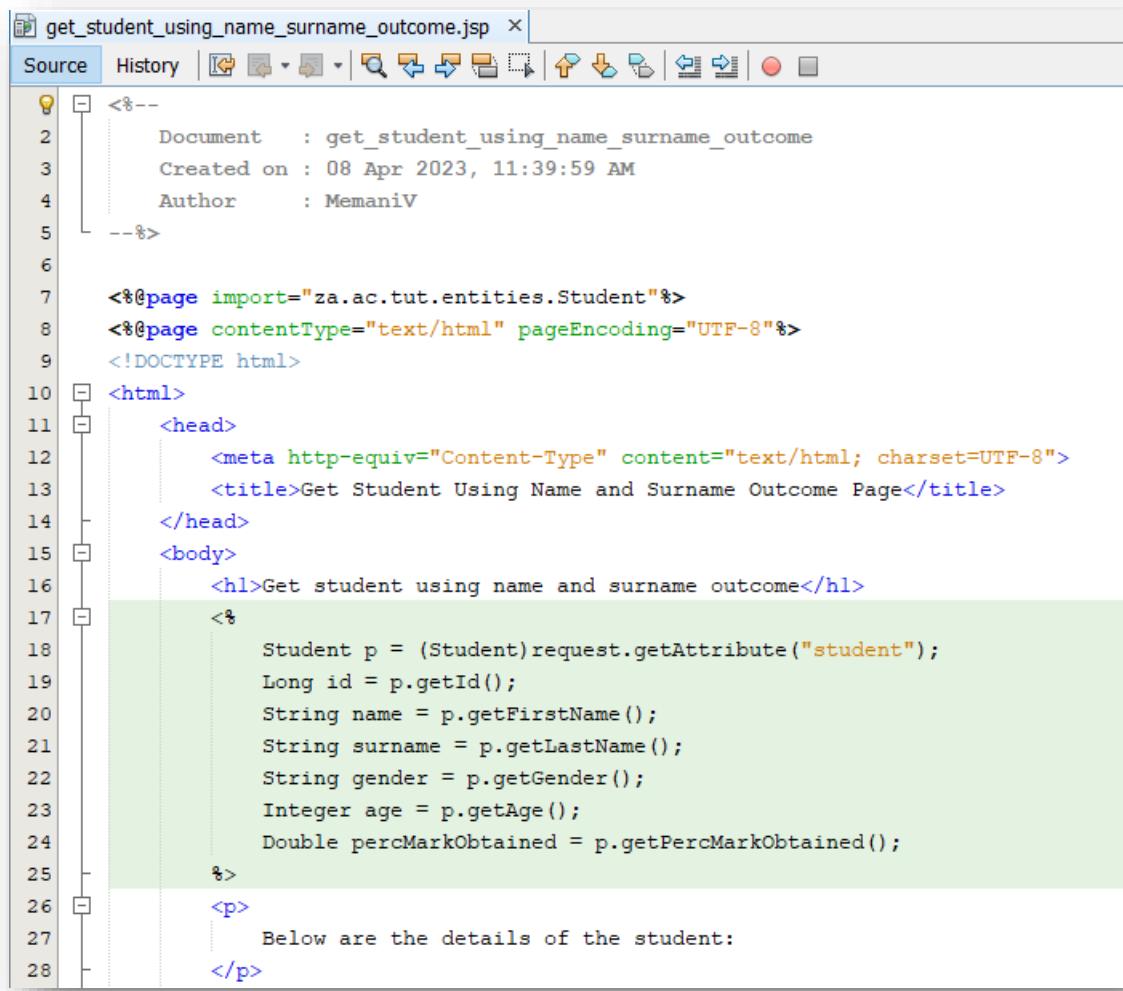
Create the get_student_using_id_outcome.jsp file.

The screenshot shows a Java IDE interface with the file 'get_student_using_id_outcome.jsp' open. The code is a JSP page that retrieves student details from a request attribute named 'student'. It then displays these details in an HTML table. The code includes imports for the Student entity, page directives for content type and encoding, and an HTML structure with a heading and a paragraph describing the student's details.

```
<%--  
    Document      : get_student_using_id_outcome  
    Created on   : 08 Apr 2023, 11:26:46 AM  
    Author       : MemaniV  
--%>  
  
<%@page import="za.ac.tut.entities.Student"%>  
<%@page contentType="text/html" pageEncoding="UTF-8"%>  
<!DOCTYPE html>  
<html>  
    <head>  
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
        <title>Get Student Using ID Outcome Page</title>  
    </head>  
    <body>  
        <h1>Get student using id outcome</h1>  
        <%  
            Student p = (Student)request.getAttribute("student");  
            Long id = p.getId();  
            String name = p.getFirstName();  
            String surname = p.getLastName();  
            String gender = p.getGender();  
            Integer age = p.getAge();  
            Double percMarkObtained = p.getPercMarkObtained();  
        %>  
        <p>  
            Below are the details of the student:  
        </p>  
  
<table>  
    <tr>  
        <td>ID:</td>  
        <td><%=id%></td>  
    </tr>  
    <tr>  
        <td>Name:</td>  
        <td><%=name%></td>  
    </tr>  
    <tr>  
        <td>Surname:</td>  
        <td><%=surname%></td>  
    </tr>  
    <tr>  
        <td>Age:</td>  
        <td><%=age%></td>  
    </tr>  
    <tr>  
        <td>Gender:</td>  
        <td><%=gender%></td>  
    </tr>  
    <tr>  
        <td>Mark obtained:</td>  
        <td><%=percMarkObtained%></td>  
    </tr>  
    <%  
    }  
    %>  
</table>  
<p>  
    Please click <a href="menu.jsp">here</a> to get back to the menu page or <a href="index.html">here</a>  
    to the main page.  
</p>  
</body>  
</html>
```

This screenshot shows the continuation of the JSP code. It starts with a closing brace for a scriptlet, followed by another scriptlet block that ends with a closing brace. The code then contains a closing tag for a table, a closing tag for a paragraph, and finally a closing tag for the entire HTML document.

Create the `get_student_using_name_surname_outcome.jsp` file.



The screenshot shows a Java IDE interface with the following details:

- Title Bar:** The title bar displays the file name: `get_student_using_name_surname_outcome.jsp`.
- Toolbar:** A standard toolbar with icons for file operations like New, Open, Save, Print, and others.
- Source Tab:** The tab is selected, showing the Java Server Page (JSP) code.
- Code Editor:** The code is displayed in a syntax-highlighted editor. The code is as follows:

```
<%--  
1 Document      : get_student_using_name_surname_outcome  
2 Created on   : 08 Apr 2023, 11:39:59 AM  
3 Author        : MemaniV  
4--%>  
  
<%@page import="za.ac.tut.entities.Student"%>  
<%@page contentType="text/html" pageEncoding="UTF-8"%>  
<!DOCTYPE html>  
<html>  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
<title>Get Student Using Name and Surname Outcome Page</title>  
</head>  
<body>  
<h1>Get student using name and surname outcome</h1>  
<%  
18     Student p = (Student)request.getAttribute("student");  
19     Long id = p.getId();  
20     String name = p.getFirstName();  
21     String surname = p.getLastName();  
22     String gender = p.getGender();  
23     Integer age = p.getAge();  
24     Double percMarkObtained = p.getPercMarkObtained();  
25 %>  
26 <p>  
27     Below are the details of the student:  
28 </p>
```

```
29 <table>
30     <tr>
31         <td>ID:</td>
32         <td><%=id%></td>
33     </tr>
34     <tr>
35         <td>Name:</td>
36         <td><%=name%></td>
37     </tr>
38     <tr>
39         <td>Surname:</td>
40         <td><%=surname%></td>
41     </tr>
42     <tr>
43         <td>Age:</td>
44         <td><%=age%></td>
45     </tr>
46     <tr>
47         <td>Gender:</td>
48         <td><%=gender%></td>
49     </tr>
50     <tr>
51         <td>Mark obtained:</td>
52         <td><%=percMarkObtained%></td>
53     </tr>
54     <%
55         }
56     %>
57 </table>
58 <p>
59     Please click <a href="menu.jsp">here</a> to get back to the menu page or <a href="index.html">here</a>
60     to the main page.
61 </p>
62 </body>
63 </html>
```

Create the `get_student_using_age_outcome.jsp` file.

The screenshot shows a Java JSP code editor with the following details:

- Title Bar:** The title bar displays "get_students_using_age_outcome.jsp".
- Toolbar:** The toolbar includes icons for Source, History, and various file operations like Open, Save, Print, and Copy.
- Code Structure:** The code is organized into sections:
 - Document Information: "Document : get_students_using_age_outcome", "Created on : 08 Apr 2023, 11:48:34 AM", "Author : MemaniV".
 - Page Directives:

```
<%@page import="java.util.List"%>
<%@page import="za.ac.tut.entities.Student"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
```
 - Doctype:

```
<!DOCTYPE html>
```
 - HTML Structure:

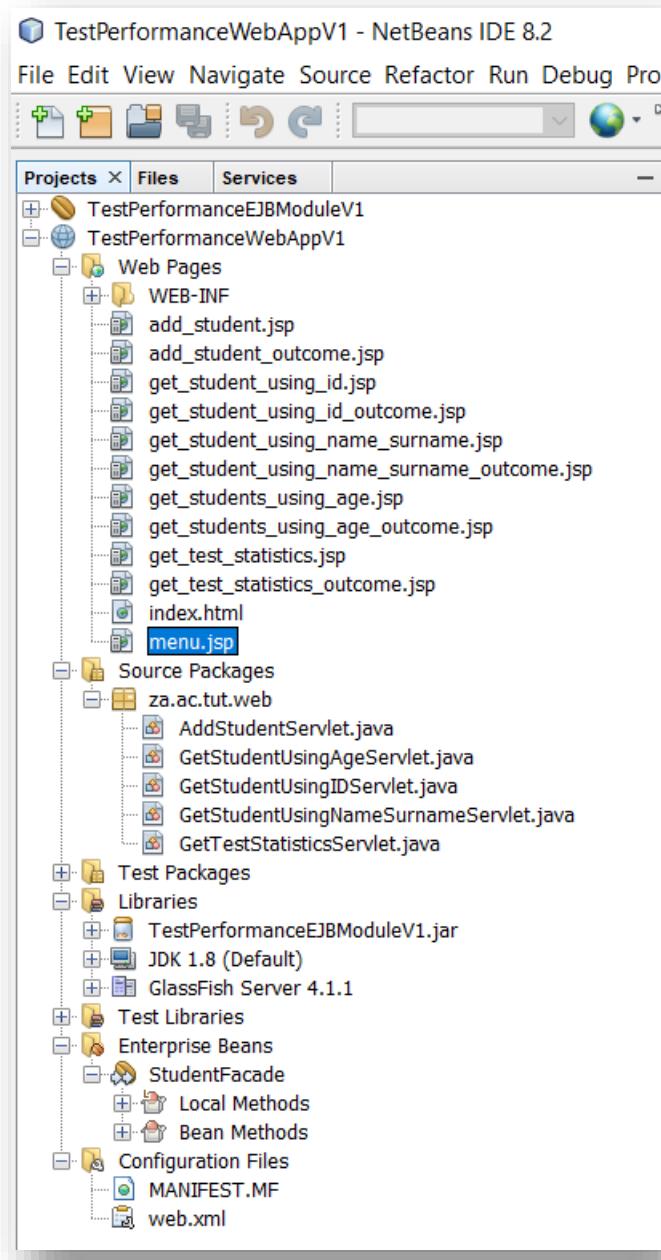
```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Get Students Using Age Outcome Page</title>
    </head>
    <body>
```
 - Main Content:

```
        <h1>Get students using age outcome</h1>
        <%
            List<Student> list = (List<Student>)request.getAttribute("list");
            Integer minAge = (Integer)request.getAttribute("minAge");
            Integer maxAge = (Integer)request.getAttribute("maxAge");
        %>
        <p>
            Below are the students within the age group <b><%=minAge%></b> and <b><%=maxAge%></b>.
        </p>
```

```
26 | 27 | <table>
28 | 29 |     <%
30 | 31 |         for(int i = 0; i < list.size(); i++){
32 | 33 |             Student p = list.get(i);
34 | 35 |             Long id = p.getId();
36 | 37 |             String name = p.getFirstName();
37 | 38 |             String surname = p.getLastName();
38 | 39 |             String gender = p.getGender();
39 | 40 |             Integer age = p.getAge();
40 | 41 |             Double percMarkObtained = p.getpercMarkObtained();
41 | 42 |         %>
42 | 43 |         <tr>
43 | 44 |             <td>ID:</td>
44 | 45 |             <td><%=id%></td>
45 | 46 |         </tr>
46 | 47 |         <tr>
47 | 48 |             <td>Name:</td>
48 | 49 |             <td><%=name%></td>
49 | 50 |         </tr>
50 | 51 |         <tr>
51 | 52 |             <td>Surname:</td>
52 | 53 |             <td><%=surname%></td>
53 | 54 |         </tr>
54 | 55 |         <tr>
55 | 56 |             <td>Age:</td>
56 | 57 |             <td><%=age%></td>
57 | 58 |         </tr>
58 | 59 |         <tr>
59 | 60 |             <td>Mark obtained:</td>
60 |         <td><%=percMarkObtained%></td>
60 |     </tr>
```

```
61 | 62 |     <%
63 | 64 |     %>
64 | 65 |     </table>
65 | 66 |     <p>
66 | 67 |         Please click <a href="menu.jsp">here</a> to get back to the menu page or <a href="index.html">here</a>
67 | 68 |         to the main page.
68 | 69 |     </p>
69 | 70 |     </body>
70 | 71 | </html>
```

View the complete project structure of **TestPerformanceWebAppV1**.



Compile the project.

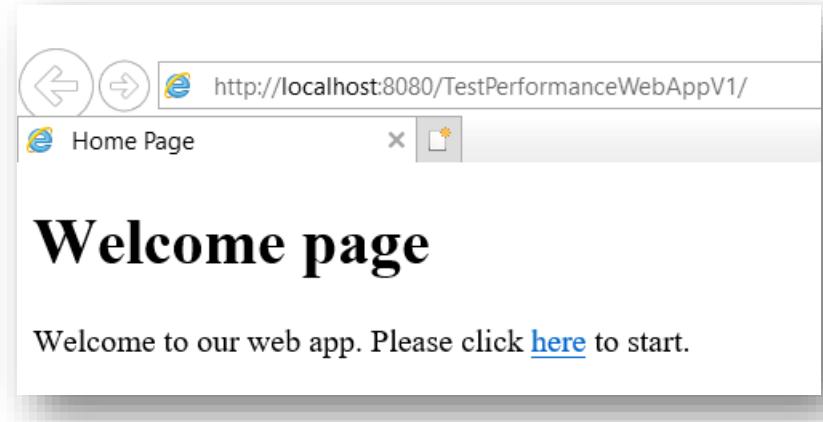
```
Output - TestPerformanceWebAppV1 (clean,dist) x
Copying 1 file to C:\Users\memaniv\Documents\NetBeansProjects\TestPerformanceWebAppV1\build\web\META-INF
Copying 13 files to C:\Users\memaniv\Documents\NetBeansProjects\TestPerformanceWebAppV1\build\web
library-inclusion-in-archive:
Copying 1 file to C:\Users\memaniv\Documents\NetBeansProjects\TestPerformanceWebAppV1\build\web\WEB-INF\lib
library-inclusion-in-manifest:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\TestPerformanceWebAppV1\build\empty
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\TestPerformanceWebAppV1\build\generated-sources\ap-source-output
Compiling 5 source files to C:\Users\memaniv\Documents\NetBeansProjects\TestPerformanceWebAppV1\build\web\WEB-INF\classes
compile:
compile-jsp:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\TestPerformanceWebAppV1\dist
Building jar: C:\Users\memaniv\Documents\NetBeansProjects\TestPerformanceWebAppV1\dist\TestPerformanceWebAppV1.war
do-dist:
dist:
BUILD SUCCESSFUL (total time: 0 seconds)
```

Deploy the project.

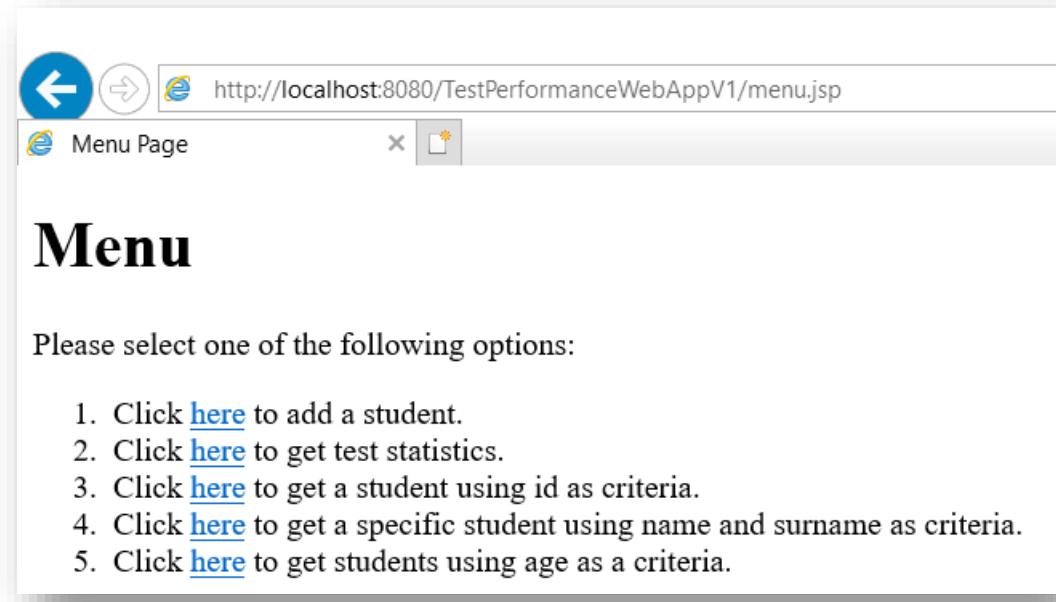
```
Output x
Java DB Database Process x GlassFish Server 4.1.1 x TestPerformanceWebAppV1 (run-deploy) x
Info: visiting unvisited references
Info: visiting unvisited references
Info: EclipseLink, version: Eclipse Persistence Services - 2.6.1.v20150605-31e8258
Info: /file:/C:/Users/memaniv/Documents/NetBeansProjects/TestPerformanceWebAppV1/build/web/WEB-INF
Info: Portable JNDI names for EJB StudentFacade: [java:global/TestPerformanceWebAppV1/StudentFacade]
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] private org.glassfish.jersey.ext.cdi1x.WeldObserverMethod@53333333
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] public org.glassfish.jms.injection.JMSObserverMethod@53333333
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] public org.glassfish.jms.injection.JMSObserverMethod@53333333
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] private org.glassfish.jersey.ext.cdi1x.WeldObserverMethod@53333333
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] org.glassfish.sse.impl.ServerSentEventObserverMethod@53333333
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] org.glassfish.sse.impl.ServerSentEventObserverMethod@53333333
Info: Loading application [TestPerformanceWebAppV1] at [/TestPerformanceWebAppV1]
Info: TestPerformanceWebAppV1 was successfully deployed in 718 milliseconds.
```

Part E – Run the web application.

Launch the application.



Click on the link



Add a shape.

- ✓ Click on the first link.

The screenshot shows a web browser window with the URL http://localhost:8080/TestPerformanceWebAppV1/add_student.jsp. The title bar says "Add Student Page". The main content area has a large heading "Add student". Below it, a message says "Please add student details below:". There are six input fields: "Student number" (empty), "Name" (empty), "Surname" (empty), "Age" (empty), "Gender" (a dropdown menu showing "F"), and "Mark obtained" (empty). A "ADD STUDENT" button is at the bottom.

- ✓ Fill in the form.

The screenshot shows the same web browser window after filling out the form. The "Student number" field contains "100", the "Name" field contains "Vuyi", the "Surname" field contains "Memza", the "Age" field contains "21", the "Gender" dropdown menu shows "M", and the "Mark obtained" field contains "90". The "ADD STUDENT" button is still at the bottom.

Click on the add button.

here to get back to the menu page or [here](#) to the main page.'"/>

Add student outcome

The student has been successfully persisted into the database. Please click [here](#) to get back to the menu page or [here](#) to the main page.

Add nine more students.

- ✓ View database structure.

jdbc:derby://localhost:1527/TestPerformanceDBv1 [app on APP]

APP

Tables

STUDENT

- ID
- AGE
- FIRSTNAME
- GENDER
- LASTNAME
- PERCMARKOBTAINED

Indexes

Foreign Keys

Views

Procedures

Other schemas

- ✓ View records.

SQL 1 [jdbc:derby://localhost:15...]

Connection: jdbc:derby://localhost:1527/TestPerformanceDBv1 [app on APP]

```
1 | SELECT * FROM APP.STUDENT FETCH FIRST 100 ROWS ONLY;
2 | 
```

SELECT * FROM APP.STUDENT...

#	ID	AGE	FIRSTNAME	GENDER	LASTNAME	PERCMARKOBTAINED
1	100	21	Vuyi	M	Memza	90.0
2	101	18	Tiro	M	Nthane	30.0
3	103	17	Patrice	M	Mulumba	70.0
4	104	22	Beauty	F	Zwane	40.0
5	105	23	Zuki	F	Mahangu	90.0
6	106	25	Lira	F	Ranamane	100.0
7	107	20	Loyi	M	Sibya	20.0
8	108	23	Rato	F	Nthene	60.0
9	109	19	Tumi	M	Musenga	90.0
10	200	16	Tshilo	F	Baloyi	50.0

View all the students.

- ✓ Click on the second link.

The screenshot shows a web browser window with the URL http://localhost:8080/TestPerformanceWebAppV1/get_test_statistics.jsp. The title bar says "Get Test Statistics Page". The main content area has a large heading "Get test statistics" and a sub-instruction "Please click on the button below to get test statistics.". A single button labeled "GET TEST STATISTICS" is centered below the instruction.

- ✓ Click on the button.

The screenshot shows a web browser window with the URL <http://localhost:8080/TestPerformanceWebAppV1/GetTestStatisticsServlet.do>. The title bar says "Get List Outcome Page". The main content area has a large heading "Get test statistics outcome". Below it, a section titled "Below are the test statistics:" lists various student statistics:

Number of students that wrote the test:	10
Number of students that passed:	7
Number of students that failed:	3
Number of male students:	5
Number of female students:	5
Number of male students that passed:	3
Number of female students that passed:	4
Number of male students that failed:	2
Number of female students that failed:	1
Highest mark:	100.0%
Lowest mark:	20.0%
Average mark:	64.0%

Below is the class list:

ID:	100
Name:	Vuyi
Surname:	Memza
Age:	21
Gender:	M
Mark obtained:	90.0%
ID:	101
Name:	Tiro
Surname:	Nthane
Age:	18
Gender:	M
Mark obtained:	30.0%

ID: 103
Name: Patrice
Surname: Mulumba
Age: 17
Gender: M
Mark obtained: 70.0%

ID: 104
Name: Beauty
Surname: Zwane
Age: 22
Gender: F
Mark obtained: 40.0%

ID: 105
Name: Zuki
Surname: Mahlangu
Age: 23
Gender: F
Mark obtained: 90.0%

ID: 106
Name: Lira
Surname: Ranamane
Age: 25
Gender: F
Mark obtained: 100.0%

ID: 107
Name: Loyi
Surname: Sibiya
Age: 20
Gender: M
Mark obtained: 20.0%

ID: 108

Name: Rato

Surname: Nthane

Age: 23

Gender: F

Mark obtained: 60.0%

ID: 109

Name: Tumi

Surname: Musenga

Age: 19

Gender: M

Mark obtained: 90.0%

ID: 200

Name: Tshilo

Surname: Baloyi

Age: 16

Gender: F

Mark obtained: 50.0%

Please click [here](#) to get back to the menu page or [here](#) to the main page.

ID: 53

Shape type: Rectangle

Side 1: 25

Side2: 30

Is the shape big? true

Please click [here](#) to get back to the menu page or [here](#) to the main page.

Get student using id.

- ✓ Click on the third link.

The screenshot shows a web browser window with the URL http://localhost:8080/TestPerformanceWebAppV1/get_student_using_id.jsp. The title bar says "Get Student Using ID Page". The main content area has a large heading "Get student using id" and a sub-instruction "Please enter the ID of the student below.". Below this is a form with an "ID:" label and a text input field containing "105", followed by a "GET" button.

- ✓ Fill-in the form.

The screenshot shows the same web browser window as before, but now the text input field contains the value "105". The "GET" button is still present below the input field.

- ✓ Click on the button.

The screenshot shows a web browser window with the URL <http://localhost:8080/TestPerformanceWebAppV1/GetStudentUsingIDServlet.do>. The title bar says "Get Student Using ID Outco...". The main content area has a large heading "Get student using id outcome" and a sub-instruction "Below are the details of the student:". Below this is a table of student details:

ID:	105
Name:	Zuki
Surname:	Mahlangu
Age:	23
Gender:	F
Mark obtained:	90.0%

At the bottom, there is a message: "Please click [here](#) to get back to the menu page or [here](#) to the main page."

Get student using name and surname.

- ✓ Click on the fourth link.

The screenshot shows a web browser window with the URL http://localhost:8080/TestPerformanceWebAppV1/get_student_using_name_surname.jsp. The title bar says "Get Student Using Name an...". The page content is titled "Get student using name and surname" in bold. It instructs the user to "Please enter the name and surname of the student below." There are two input fields: "Name:" and "Surname:", both currently empty. Below them is a "GET" button.

- ✓ Fill-in the form.

The screenshot shows the same web browser window after the user has filled in the form. The "Name:" field contains "Vuyi" and the "Surname:" field contains "Memza". The "GET" button is still present below the fields.

- ✓ Click on the button.

The screenshot shows the web browser window after the "GET" button was clicked. The title bar now says "Get Student Using Name an...". The page content is titled "Get student using name and surname outcome" in bold. It states "Below are the details of the student:". A table of student details is shown:

ID:	100
Name:	Vuyi
Surname:	Memza
Age:	21
Gender:	M
Mark obtained:	90.0%

Please click [here](#) to get back to the menu page or [here](#) to the main page.

Get student using age.

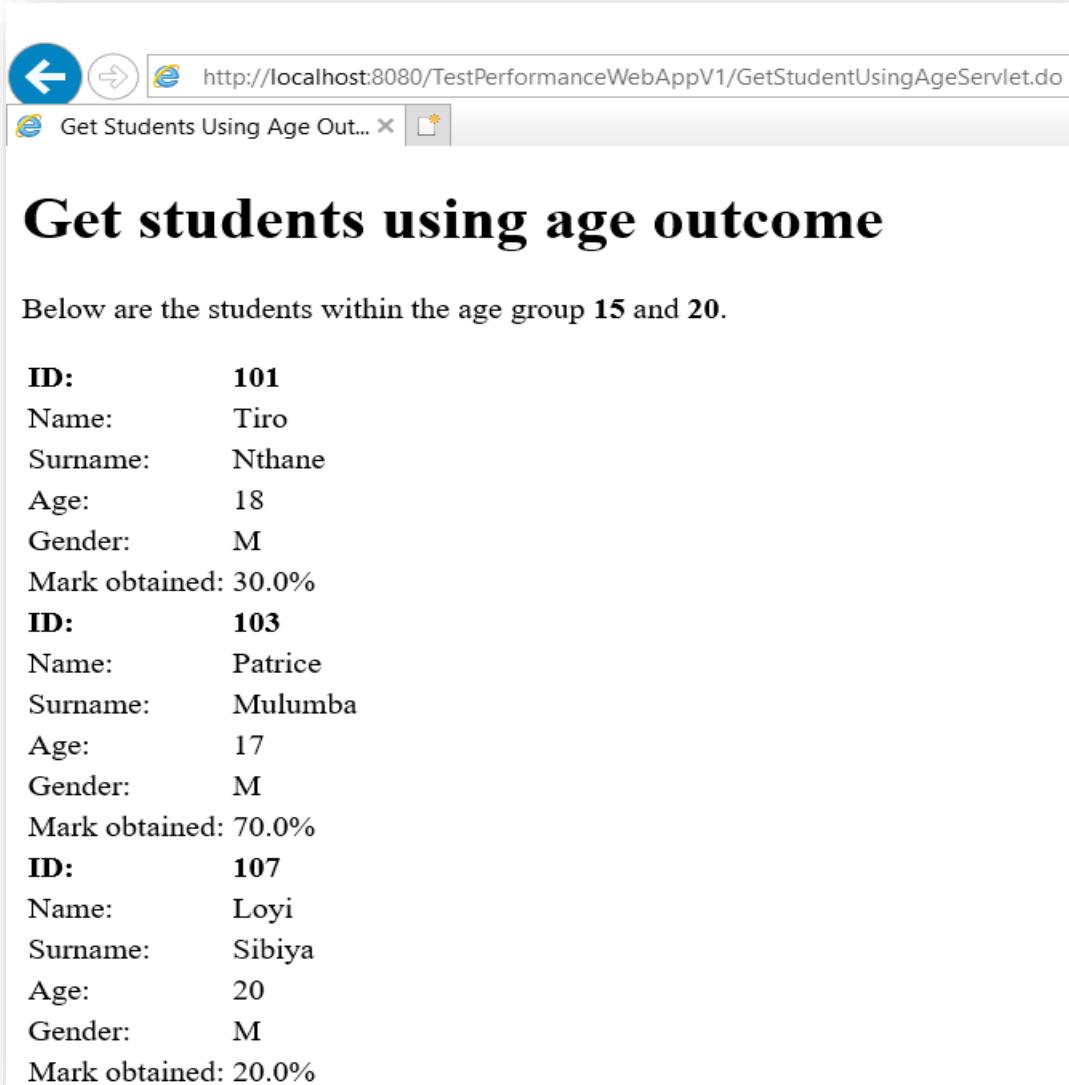
- ✓ Click on the fifth link.

The screenshot shows a web browser window with the URL http://localhost:8080/TestPerformanceWebAppV1/get_students_using_age.jsp. The title bar says "Get Student Using Age Page". The main content area has a large heading "Get student using age". Below it is a instruction: "Please enter the age values of the student below.". There are two input fields: "Minimum age:" and "Maximum age:", both currently empty. A "GET" button is positioned below the input fields.

- ✓ Fill-in the form.

The screenshot shows the same web browser window as before, but now with the form filled in. The "Minimum age:" field contains "15" and the "Maximum age:" field contains "20". The "GET" button is still present below the input fields.

- ✓ Click on the button.



The screenshot shows a web browser window with the URL <http://localhost:8080/TestPerformanceWebAppV1/GetStudentUsingAgeServlet.do>. The page title is "Get Students Using Age Out...". The main content is titled "Get students using age outcome". It lists three students matching the age range 15 to 20:

ID:	101
Name:	Tiro
Surname:	Nthane
Age:	18
Gender:	M
Mark obtained: 30.0%	
ID:	103
Name:	Patrice
Surname:	Mulumba
Age:	17
Gender:	M
Mark obtained: 70.0%	
ID:	107
Name:	Loyi
Surname:	Sibiya
Age:	20
Gender:	M
Mark obtained: 20.0%	

ID: **109**
Name: Tumi
Surname: Musenga
Age: 19
Gender: M
Mark obtained: 90.0%
ID: **200**
Name: Tshilo
Surname: Baloyi
Age: 16
Gender: F
Mark obtained: 50.0%

Please click [here](#) to get back to the menu page or [here](#) to the main page.

9.4.2 Named queries

Class Test:

No.	Student number	First name	Last name	Gender	Age	Percentage mark
1	100	Vuyi	Memza	M	21	90%
2	101	Tiro	Nthane	M	18	30%
3	103	Patrice	Mulumba	M	17	70%
4	104	Beauty	Zwane	F	22	40%
5	105	Zuki	Mahlangu	F	23	90%
6	106	Lira	Ranamane	F	25	100%
7	107	Loyi	Sibya	M	20	20%
8	108	Rato	Nthane	F	23	60%
9	109	Tumi	Musenga	M	19	90%
10	200	Tshilo	Baloyi	F	16	50%

Operations:

9. Add records.
10. Display all the records.
11. Display all the records of female students.
12. Display all the records of male students.
13. Display the records of all the students who have passed the test.
14. Display the records of all the students who have failed the test.
15. Display the records (student number, first name and last name) of all female students below 20.
16. Search for a student record using the student number.

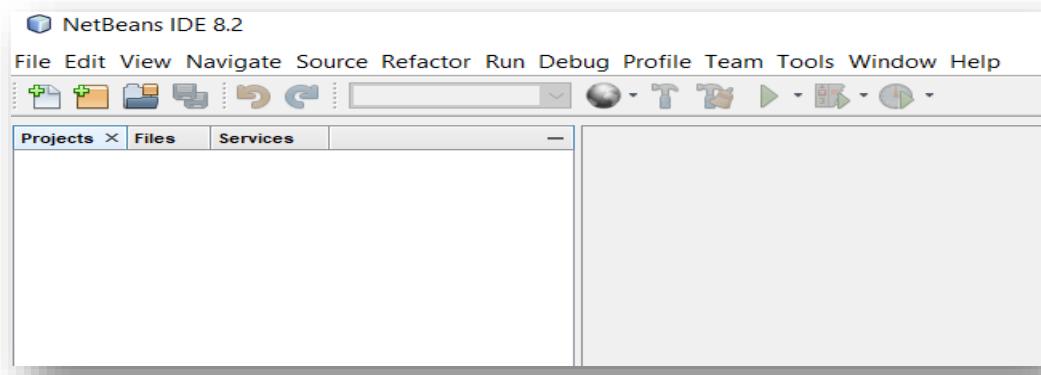
Solution approach

The solution to this problem is going to be done in five parts, namely:

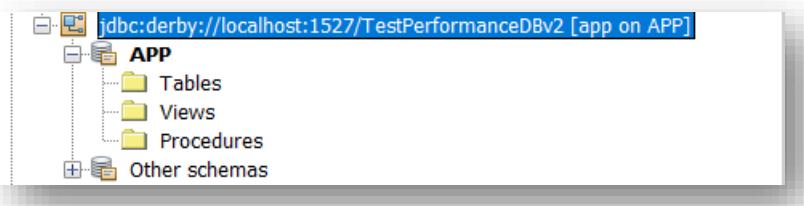
- **Part A:** we create a database.
- **Part B:** we establish connection between the created database and GlassFish.
- **Part C:** we create an EJB module. The module will have an entity with business logic.
- **Part D:** we create a client web application.
- **Part E:** we run the client web application.

Part A - Create a database

Launch NetBeans.



Create a database called **TestPerformanceDBv2**. Have the username as **app** and the password as **123**.



Part B – Connect the database to the application server.

Connect the database to the server.

- ✓ Create a connection pool.

A screenshot of the "Edit JDBC Connection Pool Properties" dialog in NetBeans. At the top, there are tabs for "General", "Advanced", and "Additional Properties". The "General" tab is selected. A yellow message box in the top right corner says "New values successfully saved." Below the tabs, it says "Edit JDBC Connection Pool Properties" and "Modify properties of an existing JDBC connection pool." The "Pool Name" is set to "DerbyPool". In the "Additional Properties (6)" section, there is a table with columns "Select", "Name", and "Value".

Select	Name	Value
<input type="checkbox"/>	PortNumber	1527
<input type="checkbox"/>	Password	123
<input type="checkbox"/>	User	app
<input type="checkbox"/>	serverName	localhost
<input type="checkbox"/>	DatabaseName	TestPerformanceDBv2
<input type="checkbox"/>	url	jdbc:derby://localhost:1527/TestPerformance

- ✓ Test if the pool is working.

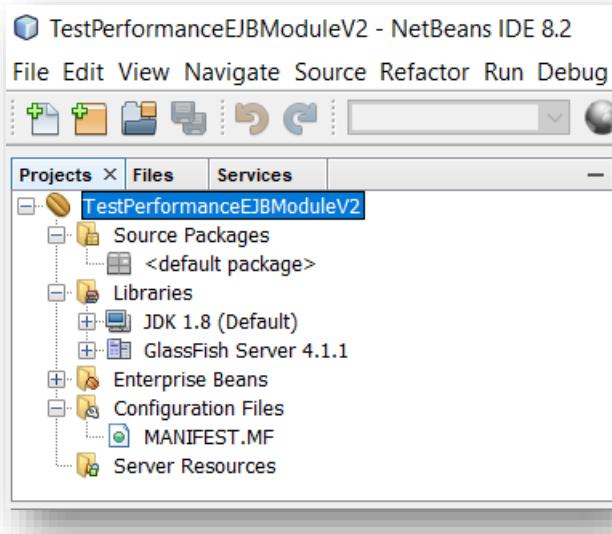
General	Advanced	Additional Properties																									
Ping Succeeded																											
Edit JDBC Connection Pool Modify an existing JDBC connection pool. A JDBC connection pool is a group of reusable connections for a particular database. <input type="button" value="Load Defaults"/> <input type="button" value="Flush"/> <input type="button" value="Ping"/>																											
General Settings <table border="0"> <tr> <td>Pool Name:</td> <td>DerbyPool</td> </tr> <tr> <td>Resource Type:</td> <td><input type="text" value="javax.sql.DataSource"/></td> </tr> <tr> <td colspan="2">Must be specified if the datasource class implements more than 1 of the interface.</td> </tr> <tr> <td>Datasource Classname:</td> <td><input type="text" value="org.apache.derby.jdbc.ClientDataSource"/></td> </tr> <tr> <td colspan="2">Vendor-specific classname that implements the DataSource and/or XADatasource APIs</td> </tr> <tr> <td>Driver Classname:</td> <td><input type="text"/></td> </tr> <tr> <td colspan="2">Vendor-specific classname that implements the java.sql.Driver interface.</td> </tr> <tr> <td>Ping:</td> <td><input type="checkbox"/> Enabled</td> </tr> <tr> <td colspan="2">When enabled, the pool is pinged during creation or reconfiguration to identify and warn of any erroneous values for its attributes</td> </tr> <tr> <td>Deployment Order:</td> <td><input type="text" value="100"/></td> </tr> <tr> <td colspan="2">Specifies the loading order of the resource at server startup. Lower numbers are loaded first.</td> </tr> <tr> <td>Description:</td> <td colspan="2"><input type="text"/></td> </tr> </table>			Pool Name:	DerbyPool	Resource Type:	<input type="text" value="javax.sql.DataSource"/>	Must be specified if the datasource class implements more than 1 of the interface.		Datasource Classname:	<input type="text" value="org.apache.derby.jdbc.ClientDataSource"/>	Vendor-specific classname that implements the DataSource and/or XADatasource APIs		Driver Classname:	<input type="text"/>	Vendor-specific classname that implements the java.sql.Driver interface.		Ping:	<input type="checkbox"/> Enabled	When enabled, the pool is pinged during creation or reconfiguration to identify and warn of any erroneous values for its attributes		Deployment Order:	<input type="text" value="100"/>	Specifies the loading order of the resource at server startup. Lower numbers are loaded first.		Description:	<input type="text"/>	
Pool Name:	DerbyPool																										
Resource Type:	<input type="text" value="javax.sql.DataSource"/>																										
Must be specified if the datasource class implements more than 1 of the interface.																											
Datasource Classname:	<input type="text" value="org.apache.derby.jdbc.ClientDataSource"/>																										
Vendor-specific classname that implements the DataSource and/or XADatasource APIs																											
Driver Classname:	<input type="text"/>																										
Vendor-specific classname that implements the java.sql.Driver interface.																											
Ping:	<input type="checkbox"/> Enabled																										
When enabled, the pool is pinged during creation or reconfiguration to identify and warn of any erroneous values for its attributes																											
Deployment Order:	<input type="text" value="100"/>																										
Specifies the loading order of the resource at server startup. Lower numbers are loaded first.																											
Description:	<input type="text"/>																										

- ✓ Check if data source is pointing to the connection pool.

Edit JDBC Resource Edit an existing JDBC data source. <input type="button" value="Load Defaults"/>	
JNDI Name:	jdbc/__default
Logical JNDI Name:	java:comp/DefaultDataSource
Pool Name:	<input type="text" value="DerbyPool"/> <input type="button" value="▼"/>
Use the JDBC Connection Pools page to create new pools	
Deployment Order:	<input type="text" value="100"/>
Specifies the loading order of the resource at server startup. Lower numbers are loaded first.	
Description:	<input type="text"/>
Status:	<input checked="" type="checkbox"/> Enabled

Part C – Create an entity.

Create an EJB project called **TestPerformanceEJBModuleV2**.



Create an entity called **Student**.

The screenshot shows the NetBeans IDE code editor for the `Student.java` file. The code defines an entity with attributes and constructors. The code is as follows:

```
1 package za.ac.tut.entities;
2
3 import java.io.Serializable;
4 import javax.persistence.Entity;
5 import javax.persistence.Id;
6
7 /**
8 * @author MemaniV
9 */
10 @Entity
11 public class Student implements Serializable {
12
13     private static final long serialVersionUID = 1L;
14     @Id
15     private Long id;
16     private String firstName;
17     private String lastName;
18     private String gender;
19     private Integer age;
20     private Double percMarkObtained;
21
22     public Student() {
23     }
24
25     public Student(Long id, String firstName, String lastName, String gender,
26                     Integer age, Double percMarkObtained) {
27         this.id = id;
28         this.firstName = firstName;
29         this.lastName = lastName;
30         this.gender = gender;
31         this.age = age;
32         this.percMarkObtained = percMarkObtained;
33     }
34 }
```

```
36  □    public String getFirstName() {
37      return firstName;
38  }
39
40  □    public void setFirstName(String firstName) {
41        this.firstName = firstName;
42    }
43
44  □    public String getLastName() {
45        return lastName;
46    }
47
48  □    public void setLastName(String lastName) {
49        this.lastName = lastName;
50    }
51
52  □    public String getGender() {
53        return gender;
54    }
55
56  □    public void setGender(String gender) {
57        this.gender = gender;
58    }
59
60  □    public Integer getAge() {
61        return age;
62    }
63
64  □    public void setAge(Integer age) {
65        this.age = age;
66    }
```

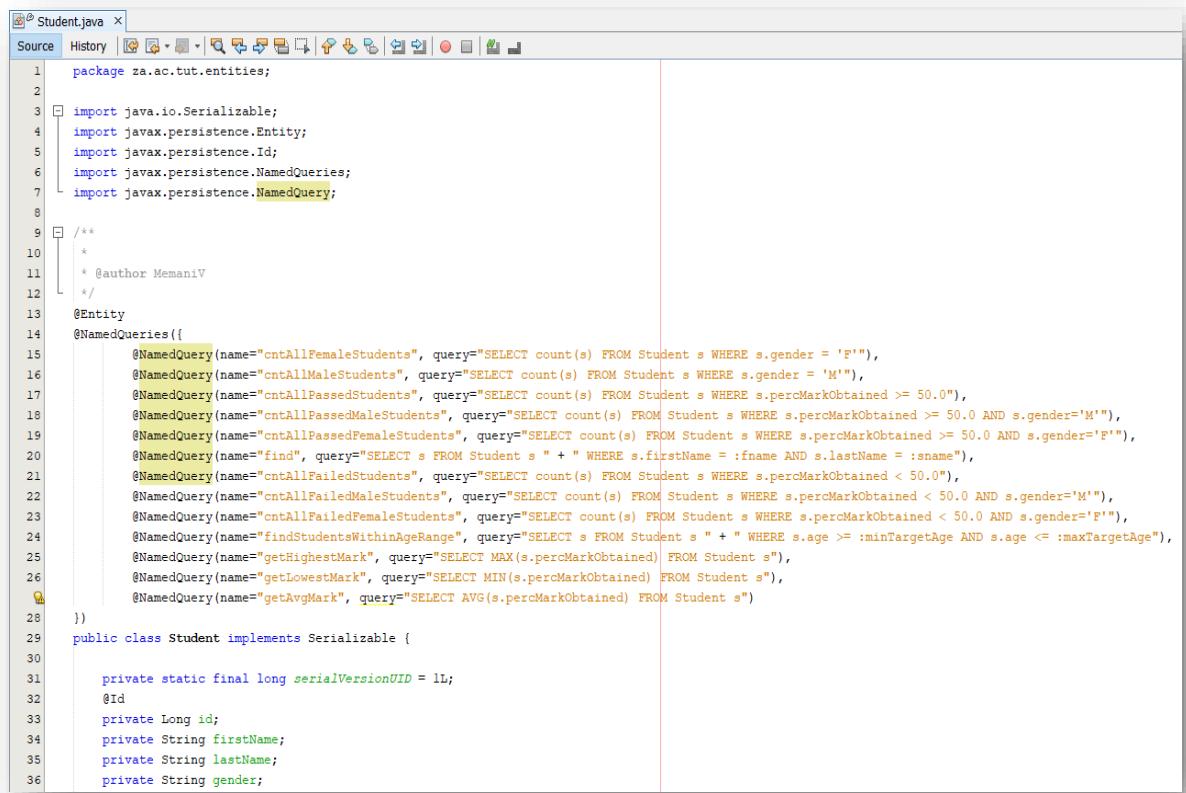
```
68  □    public Double getPercMarkObtained() {
69      return percMarkObtained;
70  }
71
72  □    public void setPercMarkObtained(Double percMarkObtained) {
73        this_percMarkObtained = percMarkObtained;
74    }
75
76  □    public Long getId() {
77        return id;
78    }
79
80  □    public void setId(Long id) {
81        this.id = id;
82    }
83
84    @Override
85  ◎  □    public int hashCode() {
86        int hash = 0;
87        hash += (id != null ? id.hashCode() : 0);
88        return hash;
89    }
```

```

91     @Override
92     public boolean equals(Object object) {
93         if (!(object instanceof Student)) {
94             return false;
95         }
96         Student other = (Student) object;
97         if ((this.id == null &amp; other.id != null) ||
98             (this.id != null &amp; !this.id.equals(other.id))) {
99             return false;
100        }
101        return true;
102    }
103
104    @Override
105    public String toString() {
106        return "za.ac.tut.entities.Student[ id=" + id + " ]";
107    }
108
109 }

```

Include named queries into the entity.

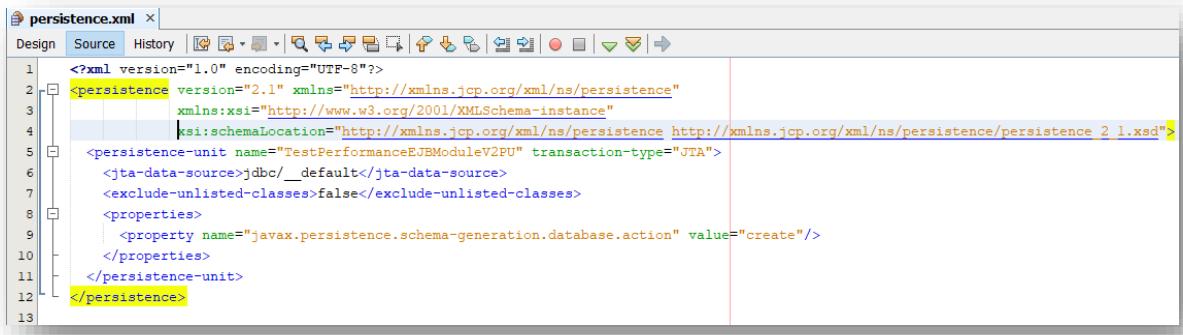


```

1 package za.ac.tut.entities;
2
3 import java.io.Serializable;
4 import javax.persistence.Entity;
5 import javax.persistence.Id;
6 import javax.persistence.NamedQueries;
7 import javax.persistenceNamedQuery;
8
9 /**
10  * @author MemaniV
11 */
12
13 @Entity
14 @NamedQueries({
15     @NamedQuery(name="cntAllFemaleStudents", query="SELECT count(s) FROM Student s WHERE s.gender = 'F'"),
16     @NamedQuery(name="cntAllMaleStudents", query="SELECT count(s) FROM Student s WHERE s.gender = 'M'"),
17     @NamedQuery(name="cntAllPassedStudents", query="SELECT count(s) FROM Student s WHERE s.percMarkObtained >= 50.0"),
18     @NamedQuery(name="cntAllPassedMaleStudents", query="SELECT count(s) FROM Student s WHERE s.percMarkObtained >= 50.0 AND s.gender='M'"),
19     @NamedQuery(name="cntAllPassedFemaleStudents", query="SELECT count(s) FROM Student s WHERE s.percMarkObtained >= 50.0 AND s.gender='F'"),
20     @NamedQuery(name="find", query="SELECT s FROM Student s " + " WHERE s.firstName = :fname AND s.lastName = :lname"),
21     @NamedQuery(name="cntAllFailedStudents", query="SELECT count(s) FROM Student s WHERE s.percMarkObtained < 50.0"),
22     @NamedQuery(name="cntAllFailedMaleStudents", query="SELECT count(s) FROM Student s WHERE s.percMarkObtained < 50.0 AND s.gender='M'"),
23     @NamedQuery(name="cntAllFailedFemaleStudents", query="SELECT count(s) FROM Student s WHERE s.percMarkObtained < 50.0 AND s.gender='F'"),
24     @NamedQuery(name="findStudentsWithinAgeRange", query="SELECT s FROM Student s " + " WHERE s.age >= :minTargetAge AND s.age <= :maxTargetAge"),
25     @NamedQuery(name="getHighestMark", query="SELECT MAX(s.percMarkObtained) FROM Student s"),
26     @NamedQuery(name="getLowestMark", query="SELECT MIN(s.percMarkObtained) FROM Student s"),
27     @NamedQuery(name="getAvgMark", query="SELECT AVG(s.percMarkObtained) FROM Student s")
28 })
29 public class Student implements Serializable {
30
31     private static final long serialVersionUID = 1L;
32     @Id
33     private Long id;
34     private String firstName;
35     private String lastName;
36     private String gender;

```

View the persistence.xml file.

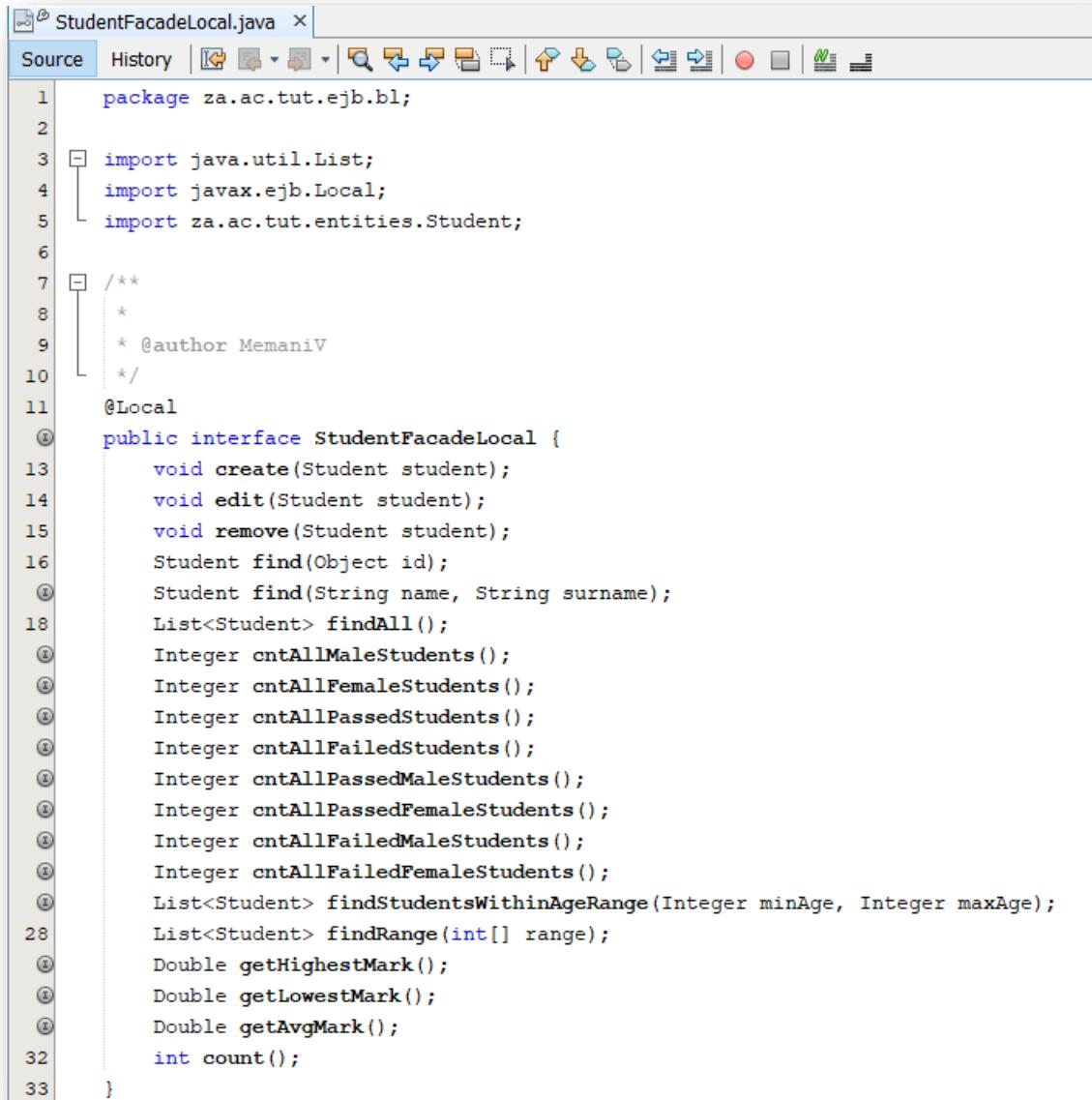


The screenshot shows the Eclipse IDE interface with the 'persistence.xml' file open in the source editor. The XML code defines a persistence unit named 'TestPerformanceEJBModuleV2PU' using the JTA transaction type. It includes a JDBC data source named 'default' and specifies that unlisted classes should not be excluded. A schema location is also defined.

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
    <persistence-unit name="TestPerformanceEJBModuleV2PU" transaction-type="JTA">
        <jta-data-source>jdbc/__default</jta-data-source>
        <exclude-unlisted-classes>false</exclude-unlisted-classes>
        <properties>
            <property name="javax.persistence.schema-generation.database.action" value="create"/>
        </properties>
    </persistence-unit>
</persistence>
```

Create business code (CRUD operations) for the entities

- **StudentFacadeLocal**



The screenshot shows the Eclipse IDE interface with the 'StudentFacadeLocal.java' file open in the source editor. The code defines a local interface for managing students, including methods for creating, editing, removing students, and performing various queries like finding students by name or age range, counting passed/failed students, and calculating average marks.

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

import java.util.List;
import javax.ejb.Local;
import za.ac.tut.entities.Student;

/**
 * 
 * @author MemaniV
 */
@Local
public interface StudentFacadeLocal {
    void create(Student student);
    void edit(Student student);
    void remove(Student student);
    Student find(Object id);
    Student find(String name, String surname);
    List<Student> findAll();
    Integer cntAllMaleStudents();
    Integer cntAllFemaleStudents();
    Integer cntAllPassedStudents();
    Integer cntAllFailedStudents();
    Integer cntAllPassedMaleStudents();
    Integer cntAllPassedFemaleStudents();
    Integer cntAllFailedMaleStudents();
    Integer cntAllFailedFemaleStudents();
    List<Student> findStudentsWithinAgeRange(Integer minAge, Integer maxAge);
    List<Student> findRange(int[] range);
    Double getHighestMark();
    Double getLowestMark();
    Double getAvgMark();
    int count();
}
```

▪ StudentFacade.

```

1 package za.ac.tut.ejb.bl;
2
3 import java.util.List;
4 import javax.ejb.Stateless;
5 import javax.persistence.EntityManager;
6 import javax.persistence.PersistenceContext;
7 import javax.persistence.Query;
8 import za.ac.tut.entities.Student;
9
10 /**
11  * 
12  * @author MemaniV
13  */
14 @Stateless
15 public class StudentFacade extends AbstractFacade<Student> implements StudentFacadeLocal {
16
17     @PersistenceContext(unitName = "TestPerformanceEJBMo

```

```

29     @Override
30     public Student find(String name, String surname) {
31         Query query = em.createNamedQuery("find");
32         query.setParameter("fname", name);
33         query.setParameter("sname", surname);
34         Student student = (Student)query.getSingleResult();
35         return student;
36     }
37
38     @Override
39     public Long cntAllMaleStudents() {
40         Query query = em.createNamedQuery("cntAllMaleStudents");
41         Long cnt = (Long)query.getSingleResult();
42         return cnt;
43     }
44
45     @Override
46     public Long cntAllFemaleStudents() {
47         Query query = em.createNamedQuery("cntAllFemaleStudents");
48         Long cnt = (Long)query.getSingleResult();
49         return cnt;
50     }
51
52     @Override
53     public Long cntAllPassedStudents() {
54         Query query = em.createNamedQuery("cntAllPassedStudents");
55         Long cnt = (Long)query.getSingleResult();
56         return cnt;
57     }
58
59     @Override
60     public Long cntAllFailedStudents() {
61         Query query = em.createNamedQuery("cntAllFailedStudents");
62         Long cnt = (Long)query.getSingleResult();
63         return cnt;
64     }

```

```
66     @Override
67     public Long cntAllPassedMaleStudents() {
68         Query query = em.createNamedQuery("cntAllPassedMaleStudents");
69         Long cnt = (Long)query.getSingleResult();
70         return cnt;
71     }
72
73     @Override
74     public Long cntAllPassedFemaleStudents() {
75         Query query = em.createNamedQuery("cntAllPassedFemaleStudents");
76         Long cnt = (Long)query.getSingleResult();
77         return cnt;
78     }
79
80     @Override
81     public Long cntAllFailedMaleStudents() {
82         Query query = em.createNamedQuery("cntAllFailedMaleStudents");
83         Long cnt = (Long)query.getSingleResult();
84         return cnt;
85     }
86
87     @Override
88     public Long cntAllFailedFemaleStudents() {
89         Query query = em.createNamedQuery("cntAllFailedFemaleStudents");
90         Long cnt = (Long)query.getSingleResult();
91         return cnt;
92     }
93
94     @Override
95     public List<Student> findStudentsWithinAgeRange(Integer minAge, Integer maxAge) {
96         Query query = em.createNamedQuery("findStudentsWithinAgeRange");
97         query.setParameter("minTargetAge", minAge);
98         query.setParameter("maxTargetAge", maxAge);
99         List<Student> students = query.getResultList();
100        return students;
101    }
```

```
103     @Override
104     public Double getHighestMark() {
105         Query query = em.createNamedQuery("getHighestMark");
106         Double highestMark = (Double)query.getSingleResult();
107         return highestMark;
108     }
109
110     @Override
111     public Double getLowestMark() {
112         Query query = em.createNamedQuery("getLowestMark");
113         Double lowestMark = (Double)query.getSingleResult();
114         return lowestMark;
115     }
116
117     @Override
118     public Double getAvgMark() {
119         Query query = em.createNamedQuery("getAvgMark");
120         Double avgMark = (Double)query.getSingleResult();
121         return avgMark;
122     }
123 }
124 }
```

▪ AbstractFacade.

AbstractFacade.java

```

1 package za.ac.tut.bl;
2
3 import java.util.List;
4 import javax.persistence.EntityManager;
5
6 /**
7  * @author MemaniV
8 */
9
10 public abstract class AbstractFacade<T> {
11
12     private Class<T> entityClass;
13
14     public AbstractFacade(Class<T> entityClass) {
15         this.entityClass = entityClass;
16     }
17
18     protected abstract EntityManager getEntityManager();
19
20     public void create(T entity) {
21         getEntityManager().persist(entity);
22     }
23
24     public void edit(T entity) {
25         getEntityManager().merge(entity);
26     }
27
28     public void remove(T entity) {
29         getEntityManager().remove(getEntityManager().merge(entity));
30     }
31

```

```

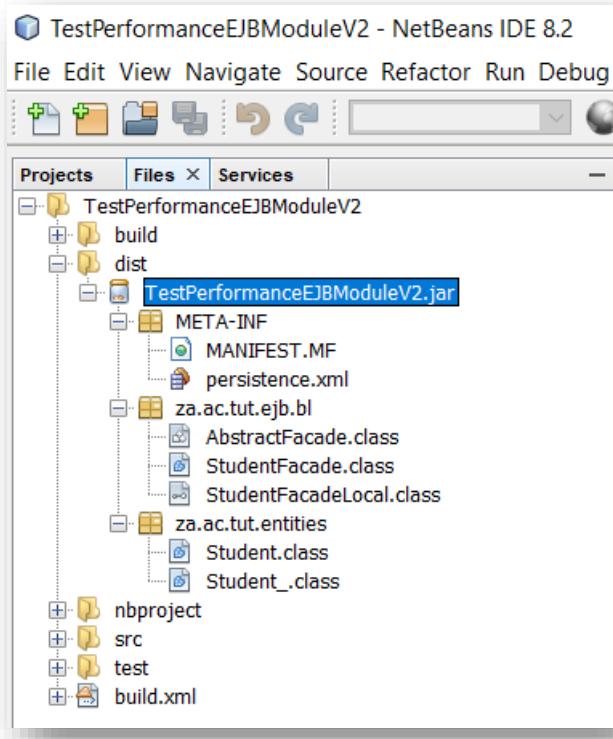
32     public T find(Object id) {
33         return getEntityManager().find(entityClass, id);
34     }
35
36     public List<T> findAll() {
37         javax.persistence.criteria.CriteriaQuery cq =
38             getEntityManager().getCriteriaBuilder().createQuery();
39         cq.select(cq.from(entityClass));
40         return getEntityManager().createQuery(cq).getResultList();
41     }
42
43     public List<T> findRange(int[] range) {
44         javax.persistence.criteria.CriteriaQuery cq =
45             getEntityManager().getCriteriaBuilder().createQuery();
46         cq.select(cq.from(entityClass));
47         javax.persistence.Query q = getEntityManager().createQuery(cq);
48         q.setMaxResults(range[1] - range[0] + 1);
49         q.setFirstResult(range[0]);
50         return q.getResultList();
51     }
52
53     public int count() {
54         javax.persistence.criteria.CriteriaQuery cq =
55             getEntityManager().getCriteriaBuilder().createQuery();
56         javax.persistence.criteria.Root<T> rt = cq.from(entityClass);
57         cq.select(getEntityManager().getCriteriaBuilder().count(rt));
58         javax.persistence.Query q = getEntityManager().createQuery(cq);
59         return ((Long) q.getSingleResult()).intValue();
60     }
61
62 }

```

✓ Clean and build the EJB project.

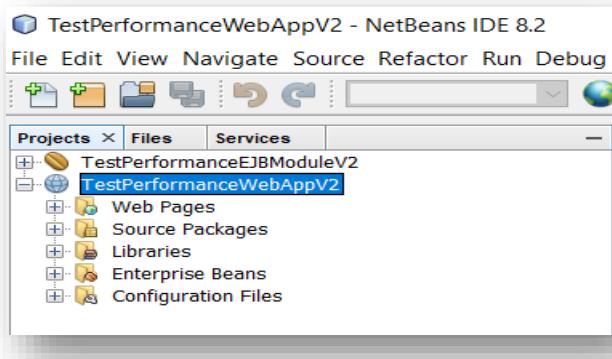
The screenshot shows the NetBeans IDE interface. At the top, there's a menu bar with File, Edit, View, Navigate, Source, Refactor, Run, and Debug. Below the menu is a toolbar with various icons. The main area has two panes: the left pane is the Project Explorer showing the structure of the 'TestPerformanceEJBModuleV2' project, and the right pane is the Output window displaying the build logs.

```
Output X
Java DB Database Process x GlassFish Server 4.1.1 x TestPerformanceEJBModuleV2 (clean,dist) x
ant -f C:\Users\memaniv\Documents\NetBeansProjects\TestPerformanceEJBModuleV2 -Dnb.internal.action.name=rebuild -Duser
init:
undeploy-clean:
deps-clean:
clean:
init:
deps-jar:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\TestPerformanceEJBModuleV2\build\classes
Copying 2 files to C:\Users\memaniv\Documents\NetBeansProjects\TestPerformanceEJBModuleV2\build\classes\META-INF
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\TestPerformanceEJBModuleV2\build\empty
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\TestPerformanceEJBModuleV2\build\generated-sources\ap-source-output
Compiling 4 source files to C:\Users\memaniv\Documents\NetBeansProjects\TestPerformanceEJBModuleV2\build\classes
warning: Supported source version 'RELEASE_6' from annotation processor 'org.eclipse.persistence.internal.jpa.modelgen.Can
Note: Creating static metadata factory ...
Note: Optional file was not found: META-INF/orm.xml continuing with generation.
Note: Optional file was not found: META-INF/eclipselink-orm.xml continuing with generation.
Note: Optional file was not found: META-INF/orm.xml continuing with generation.
Note: Optional file was not found: META-INF/eclipselink-orm.xml continuing with generation.
Note: Some input files use unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
1 warning
compile:
library-inclusion-in-archive:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\TestPerformanceEJBModuleV2\dist
Building jar: C:\Users\memaniv\Documents\NetBeansProjects\TestPerformanceEJBModuleV2\dist\TestPerformanceEJBModuleV2.jar
dist:
BUILD SUCCESSFUL (total time: 1 second)
```



Part D – Create a web client project.

Create a web client project called **TestPerformanceWebAppV2**.



Edit the **index.html** page.

```
<!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 page</h1>
        <p>
            Welcome to our web app. Please click <a href="menu.jsp">here</a> to start.
        </p>
    </body>
</html>
```

Create the **menu.jsp** file.

```
<%-->
Document : menu
Created on : 05 Apr 2023, 9:08:50 PM
Author : MemaniV
--%>

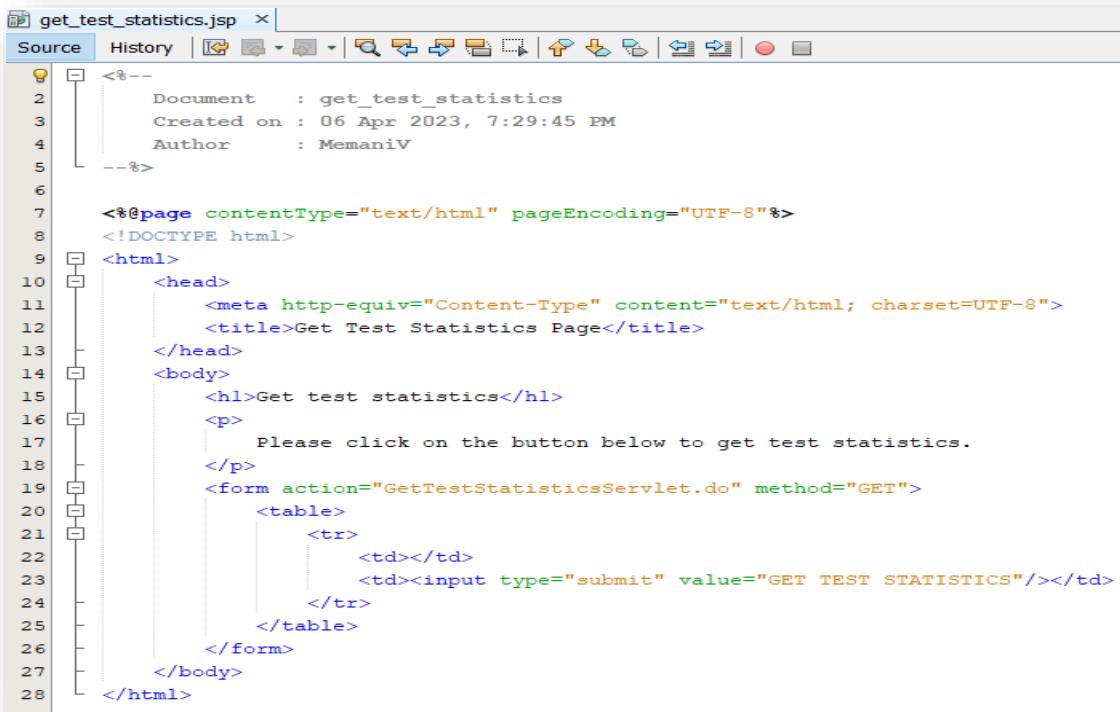
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Menu Page</title>
    </head>
    <body>
        <h1>Menu</h1>
        <p>
            Please select one of the following options:
        </p>
        <ol>
            <li>Click <a href="add_student.jsp">here</a> to add a student.</li>
            <li>Click <a href="get_test_statistics.jsp">here</a> to get test statistics.</li>
            <li>Click <a href="get_student_using_id.jsp">here</a> to get a student using id as criteria.</li>
            <li>Click <a href="get_student_using_name_surname.jsp">here</a> to get a specific student using name and surname as criteria.</li>
            <li>Click <a href="get_students_using_age.jsp">here</a> to get students using age as a criteria.</li>
        </ol>
    </body>
```

Create **add_student.jsp** file.

```
<%--  
    Document      : add_student  
    Created on   : 05 Apr 2023, 10:43:03 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>Add Student Page</title>  
    </head>  
    <body>  
        <h1>Add student</h1>  
        <p>  
            Please add student details below:  
        </p>  
        <form action="AddStudentServlet.do" method="POST">  
            <table>  
                <tr>  
                    <td>Student number:</td>  
                    <td><input type="text" name="id"/></td>  
                </tr>  
                <tr>  
                    <td>Name:</td>  
                    <td><input type="text" name="name"/></td>  
                </tr>  
                <tr>  
                    <td>Surname:</td>  
                    <td><input type="text" name="surname"/></td>  
                </tr>  
            </table>  
        </form>  
    </body>  
</html>
```

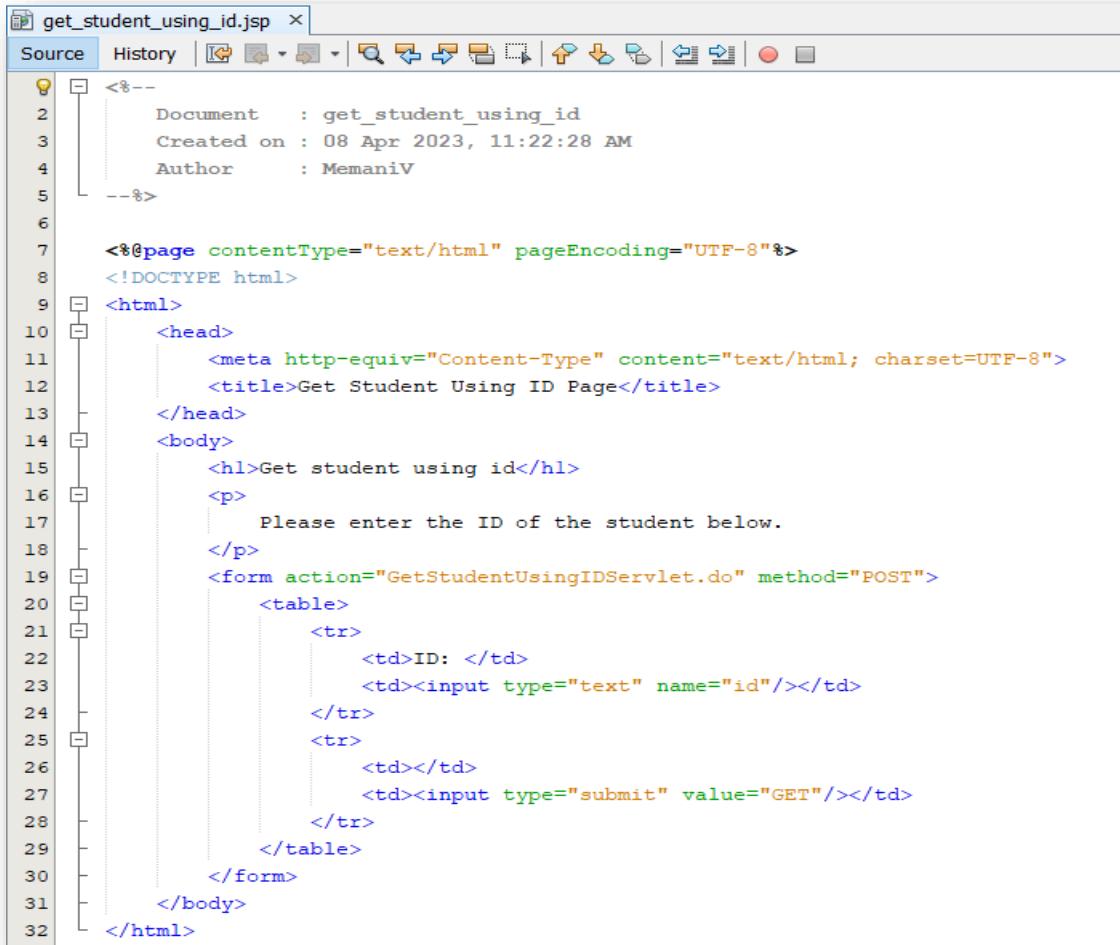
```
33 <tr>
34     <td>Age: </td>
35     <td><input type="text" name="age"/></td>
36 </tr>
37 <tr>
38     <td>Gender: </td>
39     <td>
40         <select name="gender">
41             <option value="F">F</option>
42             <option value="M">M</option>
43         </select>
44     </td>
45 </tr>
46 <tr>
47     <td>Mark obtained: </td>
48     <td><input type="text" name="percMarkObtained"/></td>
49 </tr>
50 <tr>
51     <td></td>
52     <td><input type="submit" value="ADD STUDENT"/></td>
53 </tr>
54 </table>
55 </form>
56 </body>
57 </html>
```

Create the `get_test_statistics.jsp` file.



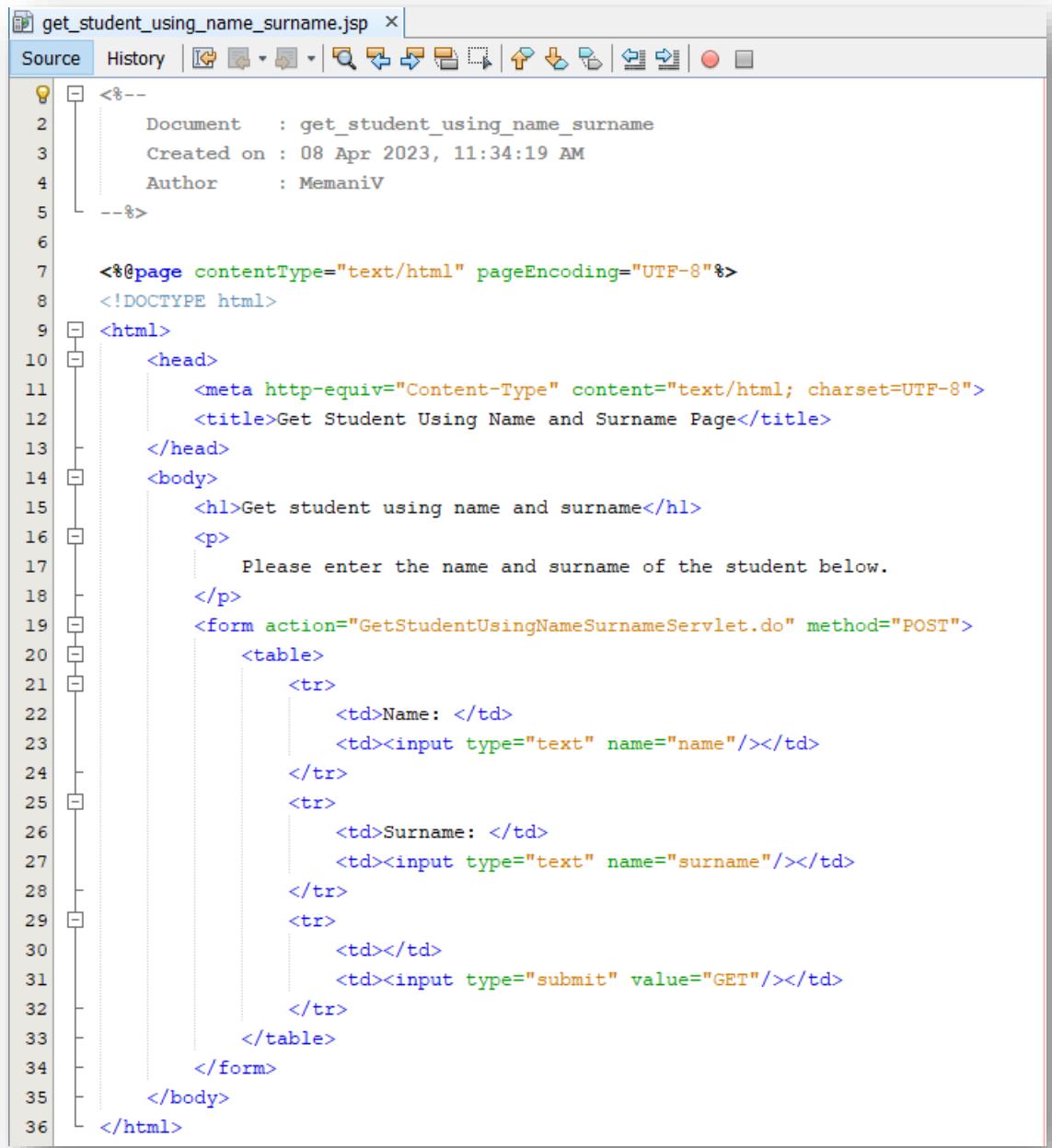
```
<%--  
    Document      : get_test_statistics  
    Created on   : 06 Apr 2023, 7:29:45 PM  
    Author        : MemaniV  
--%>  
  
<%@page contentType="text/html" pageEncoding="UTF-8"%>  
<!DOCTYPE html>  
<html>  
    <head>  
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
        <title>Get Test Statistics Page</title>  
    </head>  
    <body>  
        <h1>Get test statistics</h1>  
        <p>  
            Please click on the button below to get test statistics.  
        </p>  
        <form action="GetTestStatisticsServlet.do" method="GET">  
            <table>  
                <tr>  
                    <td></td>  
                    <td><input type="submit" value="GET TEST STATISTICS"/></td>  
                </tr>  
            </table>  
        </form>  
    </body>  
</html>
```

Create the `get_student_using_id.jsp` file.



```
<%--  
    Document      : get_student_using_id  
    Created on   : 08 Apr 2023, 11:22:28 AM  
    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>Get Student Using ID Page</title>  
    </head>  
    <body>  
        <h1>Get student using id</h1>  
        <p>  
            Please enter the ID of the student below.  
        </p>  
        <form action="GetStudentUsingIDServlet.do" method="POST">  
            <table>  
                <tr>  
                    <td>ID: </td>  
                    <td><input type="text" name="id"/></td>  
                </tr>  
                <tr>  
                    <td></td>  
                    <td><input type="submit" value="GET"/></td>  
                </tr>  
            </table>  
        </form>  
    </body>  
</html>
```

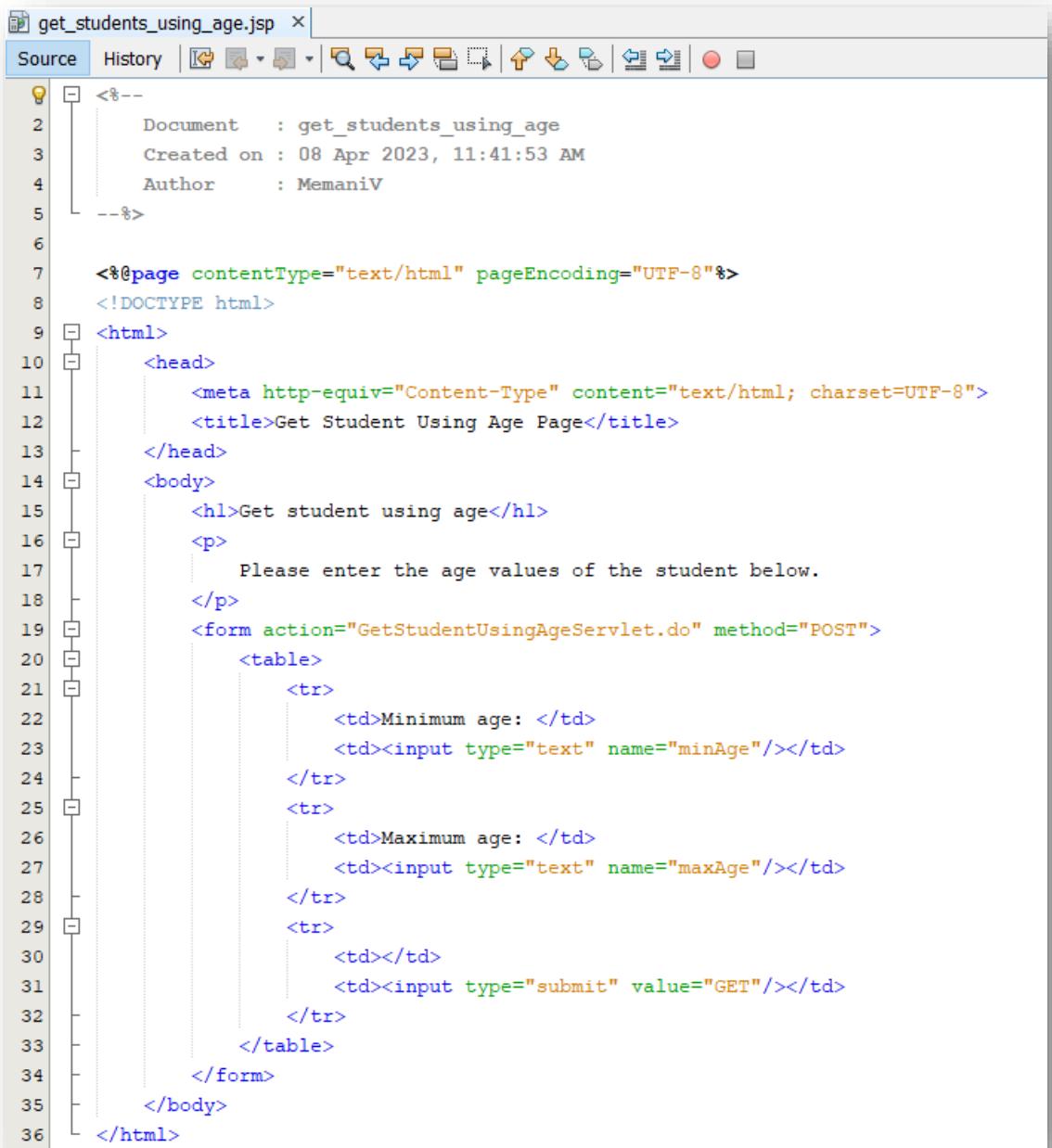
Create the get_student_using_name_surname.jsp file.



The screenshot shows a Java IDE interface with the file "get_student_using_name_surname.jsp" open. The code is a JSP page that displays a form for entering a student's name and surname. The code includes JSP comments, HTML tags, and a form with two text input fields and a submit button.

```
<%--  
1 Document      : get_student_using_name_surname  
2 Created on   : 08 Apr 2023, 11:34:19 AM  
3 Author        : MemaniV  
4--%>  
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>Get Student Using Name and Surname Page</title>  
13 </head>  
14 <body>  
15     <h1>Get student using name and surname</h1>  
16     <p>  
17         Please enter the name and surname of the student below.  
18     </p>  
19     <form action="GetStudentUsingNameSurnameServlet.do" method="POST">  
20         <table>  
21             <tr>  
22                 <td>Name: </td>  
23                 <td><input type="text" name="name"/></td>  
24             </tr>  
25             <tr>  
26                 <td>Surname: </td>  
27                 <td><input type="text" name="surname"/></td>  
28             </tr>  
29             <tr>  
30                 <td></td>  
31                 <td><input type="submit" value="GET"/></td>  
32             </tr>  
33         </table>  
34     </form>  
35 </body>  
36 </html>
```

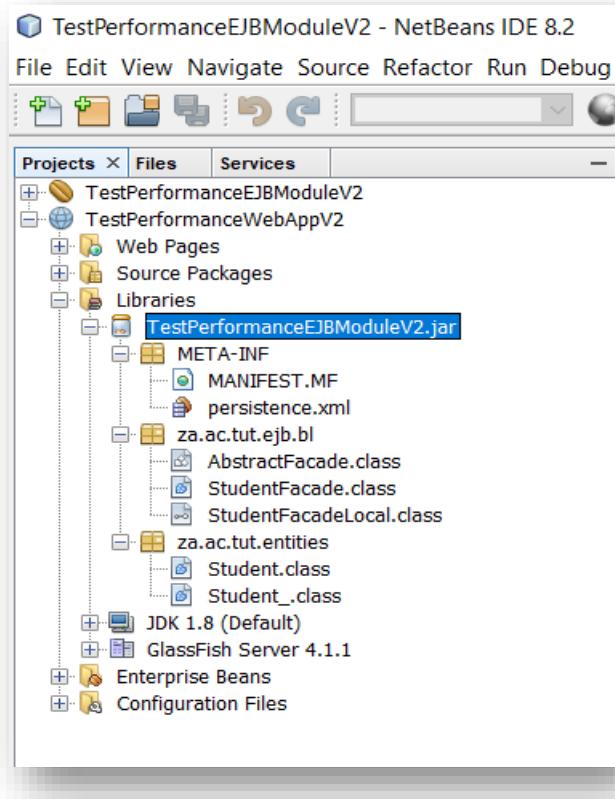
Create the get_student_using_age.jsp file.



The screenshot shows a Java IDE interface with the file "get_students_using_age.jsp" open. The code is a JSP page that displays a form for entering student age values. The code includes JSP comments, HTML tags, and a form action pointing to "GetStudentUsingAgeServlet.do".

```
<%--  
1 Document      : get_students_using_age  
2 Created on   : 08 Apr 2023, 11:41:53 AM  
3 Author        : MemaniV  
4--%>  
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>Get Student Using Age Page</title>  
13 </head>  
14 <body>  
15     <h1>Get student using age</h1>  
16     <p>  
17         Please enter the age values of the student below.  
18     </p>  
19     <form action="GetStudentUsingAgeServlet.do" method="POST">  
20         <table>  
21             <tr>  
22                 <td>Minimum age: </td>  
23                 <td><input type="text" name="minAge"/></td>  
24             </tr>  
25             <tr>  
26                 <td>Maximum age: </td>  
27                 <td><input type="text" name="maxAge"/></td>  
28             </tr>  
29             <tr>  
30                 <td></td>  
31                 <td><input type="submit" value="GET"/></td>  
32             </tr>  
33         </table>  
34     </form>  
35 </body>  
36 </html>
```

Add the **TestPerformanceEJBModuleV2.jar** file to the library of **TestPerformanceWebAppV2**.



Create the **AddStudentServlet.java** file.

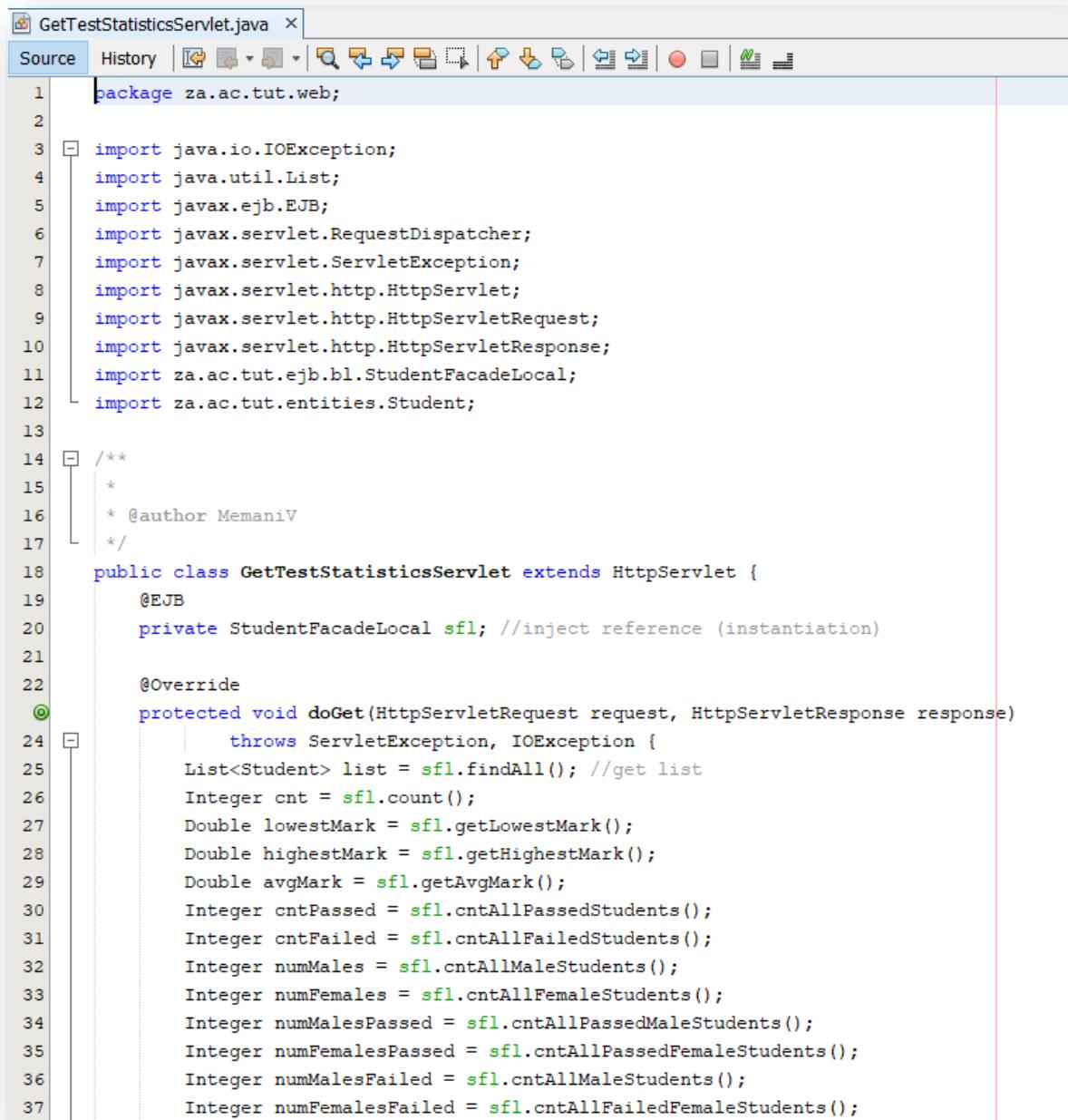
```
1 package za.ac.tut.web;
2
3 import java.io.IOException;
4 import javax.ejb.EJB;
5 import javax.servlet.RequestDispatcher;
6 import javax.servlet.ServletException;
7 import javax.servlet.http.HttpServlet;
8 import javax.servlet.http.HttpServletRequest;
9 import javax.servlet.http.HttpServletResponse;
10 import za.ac.tut.ejb.bl.StudentFacadeLocal;
11 import za.ac.tut.entities.Student;
12
13 /**
14 * 
15 * @author MemaniV
16 */
17 public class AddStudentServlet extends HttpServlet {
18     @EJB
19     private StudentFacadeLocal pfl; //inject reference (instantiation)
20     @Override
21     protected void doPost(HttpServletRequest request, HttpServletResponse response)
22             throws ServletException, IOException {
23         Long id = Long.parseLong(request.getParameter("id"));
24         String name = request.getParameter("name");
25         String surname = request.getParameter("surname");
26         String gender = request.getParameter("gender");
27         Integer age = Integer.parseInt(request.getParameter("age"));
28         Double percMarkObtained = Double.parseDouble(request.getParameter("percMarkObtained"));
29
30         Student student = createStudent(id, name, surname, gender, age, percMarkObtained); //create student
31         pfl.create(student); //persist student
32
33         RequestDispatcher disp = request.getRequestDispatcher("add_student_outcome.jsp");
34         disp.forward(request, response);
35     }
}
```

```

37 |     private Student createStudent(Long id, String name, String surname, String gender, Integer age, Double percMarkObtained) {
38 |         Student student = new Student();
39 |         student.setId(id);
40 |         student.setFirstName(name);
41 |         student.setLastName(surname);
42 |         student.setGender(gender);
43 |         student.setAge(age);
44 |         student.setPercMarkObtained(percMarkObtained);
45 |         return student;
46 |     }
47 |

```

Create the **GetTestStatisticsServlet.java** file.



The screenshot shows an IDE interface with the tab 'GetTestStatisticsServlet.java' selected. The code editor displays Java code for a servlet. The code includes imports for various Java packages and annotations like @EJB and @Override. It defines a class GetTestStatisticsServlet that extends HttpServlet. The doGet method uses a local reference to a StudentFacadeLocal bean to perform various calculations on a list of students, such as counting passed vs failed students and calculating average marks by gender.

```

1 package za.ac.tut.web;
2
3 import java.io.IOException;
4 import java.util.List;
5 import javax.ejb.EJB;
6 import javax.servlet.DispatcherType;
7 import javax.servlet.ServletException;
8 import javax.servlet.http.HttpServlet;
9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11 import za.ac.tut.ejb.bl.StudentFacadeLocal;
12 import za.ac.tut.entities.Student;
13
14 /**
15 * 
16 * @author MemaniV
17 */
18 public class GetTestStatisticsServlet extends HttpServlet {
19     @EJB
20     private StudentFacadeLocal sfl; //inject reference (instantiation)
21
22     @Override
23     protected void doGet(HttpServletRequest request, HttpServletResponse response)
24             throws ServletException, IOException {
25         List<Student> list = sfl.findAll(); //get list
26         Integer cnt = sfl.count();
27         Double lowestMark = sfl.getLowestMark();
28         Double highestMark = sfl.getHighestMark();
29         Double avgMark = sfl.getAvgMark();
30         Integer cntPassed = sfl.cntAllPassedStudents();
31         Integer cntFailed = sfl.cntAllFailedStudents();
32         Integer numMales = sfl.cntAllMaleStudents();
33         Integer numFemales = sfl.cntAllFemaleStudents();
34         Integer numMalesPassed = sfl.cntAllPassedMaleStudents();
35         Integer numFemalesPassed = sfl.cntAllPassedFemaleStudents();
36         Integer numMalesFailed = sfl.cntAllMaleStudents();
37         Integer numFemalesFailed = sfl.cntAllFailedFemaleStudents();

```

```

39     request.setAttribute("list", list);
40     request.setAttribute("cnt", cnt);
41     request.setAttribute("lowestMark", lowestMark);
42     request.setAttribute("highestMark", highestMark);
43     request.setAttribute("avgMark", avgMark);
44     request.setAttribute("cntPassed", cntPassed);
45     request.setAttribute("cntFailed", cntFailed);
46     request.setAttribute("numMales", numMales);
47     request.setAttribute("numFemales", numFemales);
48     request.setAttribute("numMalesPassed", numMalesPassed);
49     request.setAttribute("numFemalesPassed", numFemalesPassed);
50     request.setAttribute("numMalesFailed", numMalesFailed);
51     request.setAttribute("numFemalesFailed", numFemalesFailed);
52
53     RequestDispatcher disp = request.getRequestDispatcher("get_test_statistics_outcome.jsp");
54     disp.forward(request, response);
55 }
56
57 }

```

Create the GetStudentUsingIDServlet.java file.

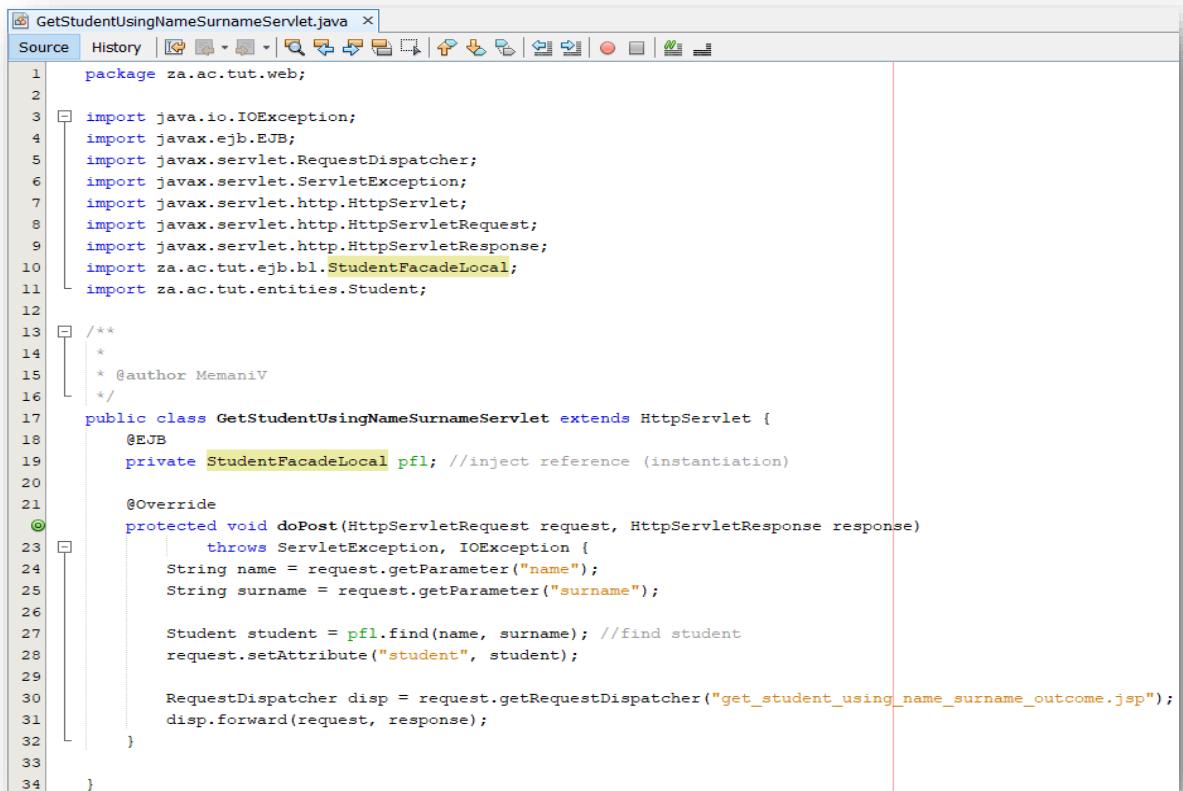
The screenshot shows an IDE interface with the tab 'GetStudentUsingIDServlet.java' selected. The code editor displays Java code for a servlet. The code includes imports for various Java packages, annotations like @EJB and @Override, and logic to handle an incoming POST request, retrieve a student by ID, and forward the request to a JSP page. The code is well-structured with proper indentation and comments.

```

1 package za.ac.tut.web;
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.StudentFacadeLocal;
11 import za.ac.tut.entities.Student;
12
13 /**
14 * 
15 * @author MemaniV
16 */
17 public class GetStudentUsingIDServlet extends HttpServlet {
18     @EJB
19     private StudentFacadeLocal pfl; //inject reference (instantiation)
20
21     @Override
22     protected void doPost(HttpServletRequest request, HttpServletResponse response)
23             throws ServletException, IOException {
24         Long id = Long.parseLong(request.getParameter("id"));
25
26         Student student = pfl.find(id); //find student
27         request.setAttribute("student", student);
28
29         RequestDispatcher disp = request.getRequestDispatcher("get_student_using_id_outcome.jsp");
30         disp.forward(request, response);
31     }
32 }

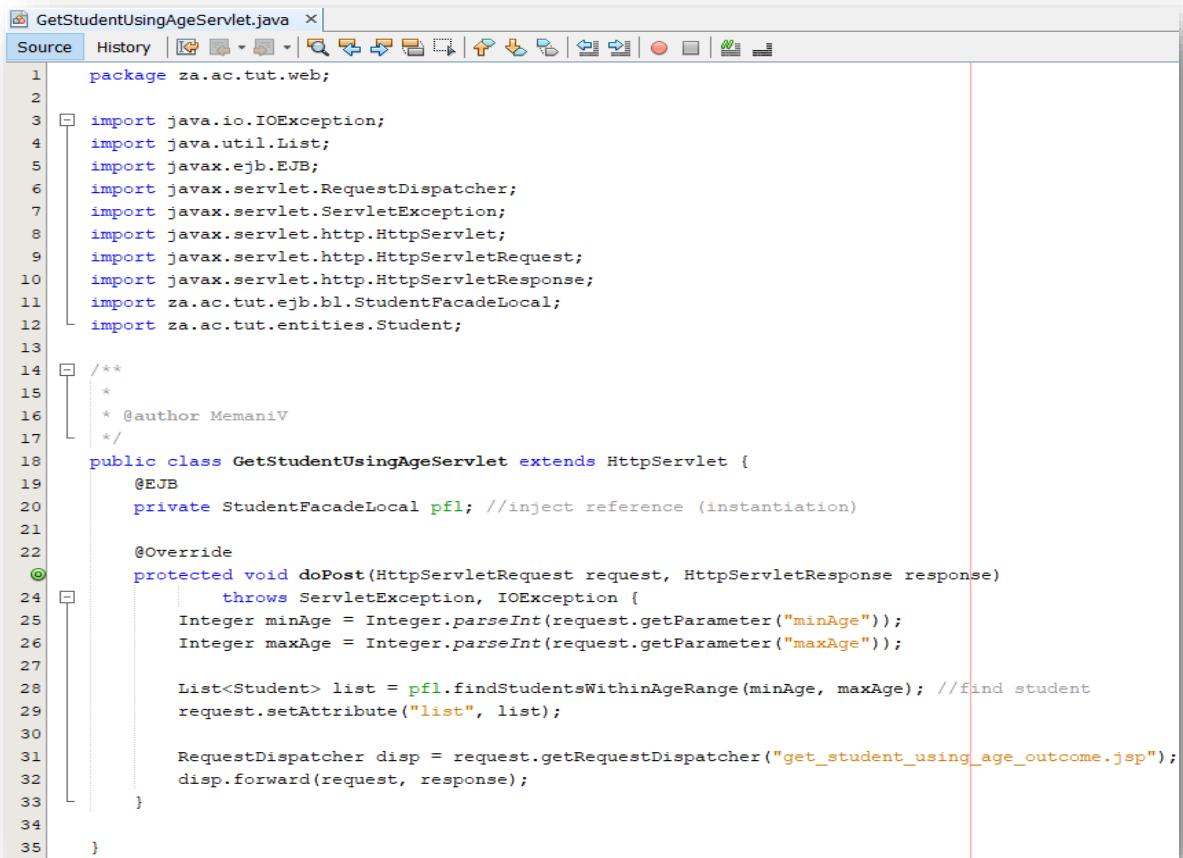
```

Create the **GetStudentUsingNameSurnameServlet.java** file.



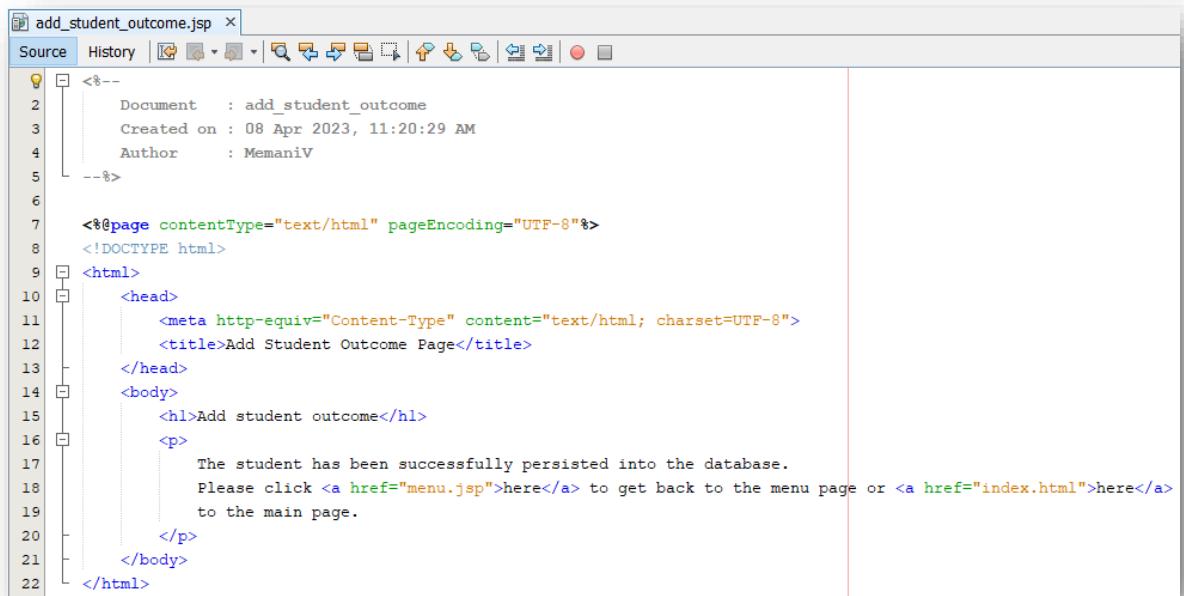
```
1 package za.ac.tut.web;
2
3 import java.io.IOException;
4 import javax.ejb.EJB;
5 import javax.servlet.RequestDispatcher;
6 import javax.servlet.ServletException;
7 import javax.servlet.http.HttpServlet;
8 import javax.servlet.http.HttpServletRequest;
9 import javax.servlet.http.HttpServletResponse;
10 import za.ac.tut.ejb.bl.StudentFacadeLocal;
11 import za.ac.tut.entities.Student;
12
13 /**
14 * @author MemaniV
15 */
16
17 public class GetStudentUsingNameSurnameServlet extends HttpServlet {
18     @EJB
19     private StudentFacadeLocal pfl; //inject reference (instantiation)
20
21     @Override
22     protected void doPost(HttpServletRequest request, HttpServletResponse response)
23             throws ServletException, IOException {
24         String name = request.getParameter("name");
25         String surname = request.getParameter("surname");
26
27         Student student = pfl.find(name, surname); //find student
28         request.setAttribute("student", student);
29
30         RequestDispatcher disp = request.getRequestDispatcher("get_student_using_name_surname_outcome.jsp");
31         disp.forward(request, response);
32     }
33 }
34 }
```

Create the **GetStudentUsingAgeServlet.java** file.



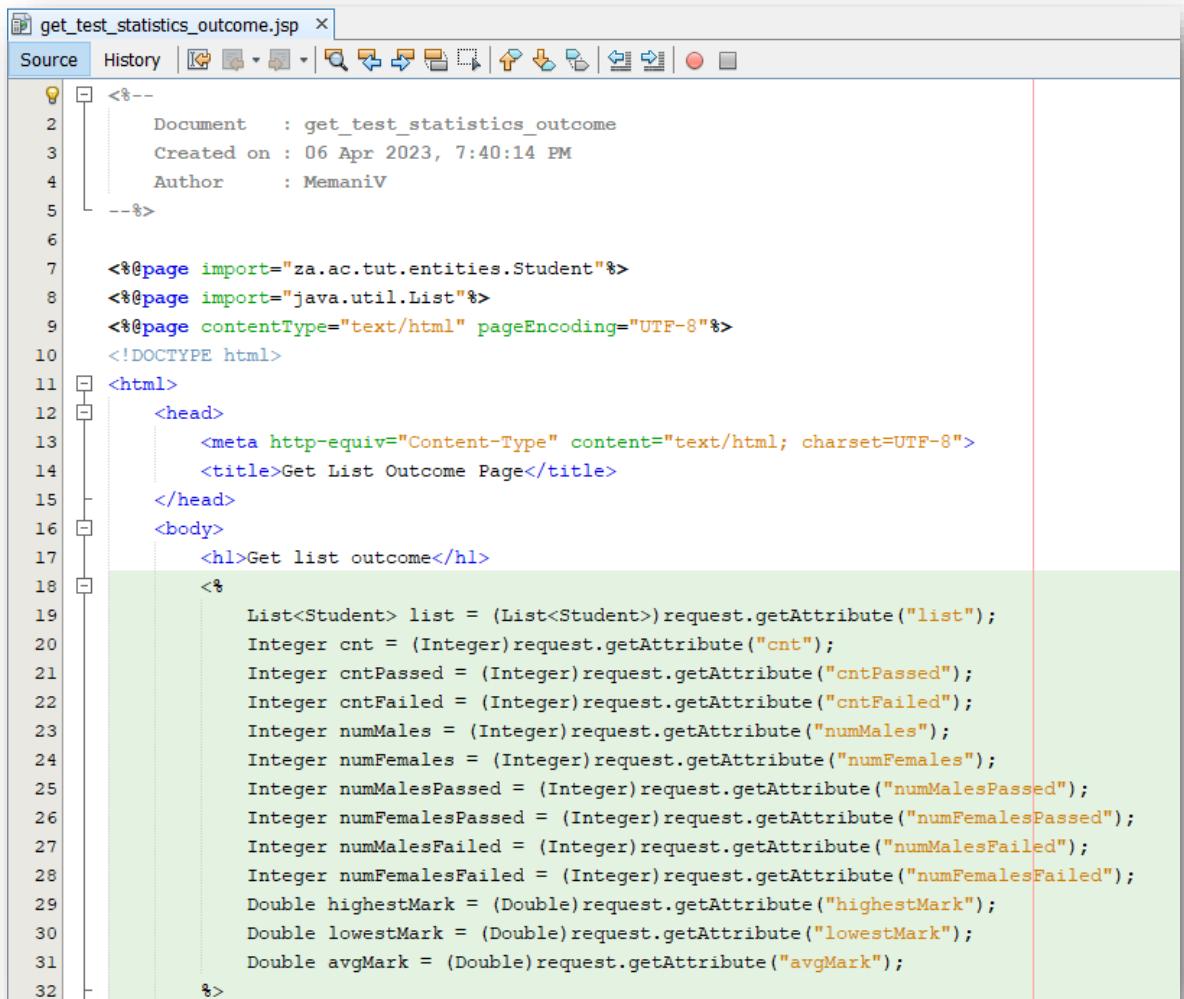
```
1 package za.ac.tut.web;
2
3 import java.io.IOException;
4 import java.util.List;
5 import javax.ejb.EJB;
6 import javax.servlet.RequestDispatcher;
7 import javax.servlet.ServletException;
8 import javax.servlet.http.HttpServlet;
9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11 import za.ac.tut.ejb.bl.StudentFacadeLocal;
12 import za.ac.tut.entities.Student;
13
14 /**
15 * @author MemaniV
16 */
17
18 public class GetStudentUsingAgeServlet extends HttpServlet {
19     @EJB
20     private StudentFacadeLocal pfl; //inject reference (instantiation)
21
22     @Override
23     protected void doPost(HttpServletRequest request, HttpServletResponse response)
24             throws ServletException, IOException {
25         Integer minAge = Integer.parseInt(request.getParameter("minAge"));
26         Integer maxAge = Integer.parseInt(request.getParameter("maxAge"));
27
28         List<Student> list = pfl.findStudentsWithinAgeRange(minAge, maxAge); //find student
29         request.setAttribute("list", list);
30
31         RequestDispatcher disp = request.getRequestDispatcher("get_student_using_age_outcome.jsp");
32         disp.forward(request, response);
33     }
34 }
```

Create the add_student_outcome.jsp file.



```
<%--  
1 Document      : add_student_outcome  
2 Created on   : 08 Apr 2023, 11:20:29 AM  
3 Author        : MemaniV  
4--%>  
5  
6 <%@page contentType="text/html" pageEncoding="UTF-8"%>  
7 <!DOCTYPE html>  
8 <html>  
9 <head>  
10 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
11 <title>Add Student Outcome Page</title>  
12 </head>  
13 <body>  
14 <h1>Add student outcome</h1>  
15 <p>  
16     The student has been successfully persisted into the database.  
17     Please click <a href="menu.jsp">here</a> to get back to the menu page or <a href="index.html">here</a>  
18     to the main page.  
19 </p>  
20 </body>  
21 </html>
```

Create the get_test_statistics_outcome.jsp file.



```
<%--  
1 Document      : get_test_statistics_outcome  
2 Created on   : 06 Apr 2023, 7:40:14 PM  
3 Author        : MemaniV  
4--%>  
5  
6 <%@page import="za.ac.tut.entities.Student"%>  
7 <%@page import="java.util.List"%>  
8 <%@page contentType="text/html" pageEncoding="UTF-8"%>  
9 <!DOCTYPE html>  
10 <html>  
11 <head>  
12 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
13 <title>Get List Outcome Page</title>  
14 </head>  
15 <body>  
16 <h1>Get list outcome</h1>  
17 <%  
18     List<Student> list = (List<Student>) request.getAttribute("list");  
19     Integer cnt = (Integer) request.getAttribute("cnt");  
20     Integer cntPassed = (Integer) request.getAttribute("cntPassed");  
21     Integer cntFailed = (Integer) request.getAttribute("cntFailed");  
22     Integer numMales = (Integer) request.getAttribute("numMales");  
23     Integer numFemales = (Integer) request.getAttribute("numFemales");  
24     Integer numMalesPassed = (Integer) request.getAttribute("numMalesPassed");  
25     Integer numFemalesPassed = (Integer) request.getAttribute("numFemalesPassed");  
26     Integer numMalesFailed = (Integer) request.getAttribute("numMalesFailed");  
27     Integer numFemalesFailed = (Integer) request.getAttribute("numFemalesFailed");  
28     Double highestMark = (Double) request.getAttribute("highestMark");  
29     Double lowestMark = (Double) request.getAttribute("lowestMark");  
30     Double avgMark = (Double) request.getAttribute("avgMark");  
31  
32 %>
```

```

33 |   <table>
34 |     <tr>
35 |       <td>Number of students that wrote the test:</td>
36 |       <td><%=cnt%></td>
37 |     </tr> |
38 |     <tr>
39 |       <td>Number of students that passed:</td>
40 |       <td><%=cntPassed%></td>
41 |     </tr>
42 |     <tr>
43 |       <td>Number of students that failed:</td>
44 |       <td><%=cntFailed%></td>
45 |     </tr>
46 |     <tr>
47 |       <td>Number of male students:</td>
48 |       <td><%=numMales%></td>
49 |     </tr>
50 |     <tr>
51 |       <td>Number of female students:</td>
52 |       <td><%=numFemales%></td>
53 |     </tr>
54 |     <tr>
55 |       <td>Number of male students that passed:</td>
56 |       <td><%=numMalesPassed%></td>
57 |     </tr>
58 |     <tr>
59 |       <td>Number of female students that passed:</td>
60 |       <td><%=numFemalesPassed%></td>
61 |     </tr>
62 |     <tr>
63 |       <td>Number of male students that failed:</td>
64 |       <td><%=numMalesFailed%></td>
65 |     </tr>
66 |     <tr>
67 |       <td>Number of female students that failed:</td>
68 |       <td><%=numFemalesFailed%></td>
69 |     </tr>

```

```

70 |   <tr>
71 |     <td>Highest mark:</td>
72 |     <td><%=highestMark%></td>
73 |   </tr>
74 |   <tr>
75 |     <td>Lowest mark:</td>
76 |     <td><%=lowestMark%></td>
77 |   </tr>
78 |   <tr>
79 |     <td>Average mark:</td>
80 |     <td><%=avgMark%></td>
81 |   </tr>
82 | </table>
83 | <p>
84 |   Below is the class list:
85 | </p>

```

```
86 |     <table>
87 |         <%
88 |             for(int i = 0; i < list.size(); i++){
89 |                 Student p = list.get(i);
90 |                 Long id = p.getId();
91 |                 String name = p.getFirstName();
92 |                 String surname = p.getLastName();
93 |                 String gender = p.getGender();
94 |                 Integer age = p.getAge();
95 |                 Double percMarkObtained = p.getPercMarkObtained();
96 |
97 |             </tr>
98 |             <td>ID:</td>
99 |             <td><%=id%></td>
100 |
101 |             <tr>
102 |                 <td>Name:</td>
103 |                 <td><%=name%></td>
104 |
105 |             <tr>
106 |                 <td>Surname:</td>
107 |                 <td><%=surname%></td>
108 |
109 |             <tr>
110 |                 <td>Age:</td>
111 |                 <td><%=age%></td>
112 |
113 |             <tr>
114 |                 <td>Gender:</td>
115 |                 <td><%=gender%></td>
116 |
117 |             <tr>
118 |                 <td>Mark obtained:</td>
119 |                 <td><%=percMarkObtained%></td>
120 |             </tr>
```

```
122 |         <%
123 |             }
124 |         </table>
125 |         <p>
126 |             Please click <a href="menu.jsp">here</a> to get back to the menu page or <a href="index.html">here</a>
127 |             to the main page.
128 |         </p>
129 |     </body>
130 | </html>
```

Create the get_student_using_id_outcome.jsp file.

The screenshot shows a Java IDE interface with the file 'get_student_using_id_outcome.jsp' open. The code is a JSP page that retrieves student details from a request attribute named 'student'. It uses JSTL tags to output the student's ID, name, surname, age, and gender. A message is displayed below the table indicating where to click to return to the menu or index page.

```
<%--  
    Document      : get_student_using_id_outcome  
    Created on   : 08 Apr 2023, 11:26:46 AM  
    Author       : MemaniV  
--%>  
  
<%@page import="za.ac.tut.entities.Student"%>  
<%@page contentType="text/html" pageEncoding="UTF-8"%>  
<!DOCTYPE html>  
<html>  
    <head>  
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
        <title>Get Student Using ID Outcome Page</title>  
    </head>  
    <body>  
        <h1>Get student using id outcome</h1>  
        <%  
            Student p = (Student)request.getAttribute("student");  
            Long id = p.getId();  
            String name = p.getFirstName();  
            String surname = p.getLastName();  
            String gender = p.getGender();  
            Integer age = p.getAge();  
            Double percMarkObtained = p.getPercMarkObtained();  
        %>  
        <p>  
            Below are the details of the student:  
        </p>  
  
<table>  
    <tr>  
        <td>ID:</td>  
        <td><%=id%></td>  
    </tr>  
    <tr>  
        <td>Name:</td>  
        <td><%=name%></td>  
    </tr>  
    <tr>  
        <td>Surname:</td>  
        <td><%=surname%></td>  
    </tr>  
    <tr>  
        <td>Age:</td>  
        <td><%=age%></td>  
    </tr>  
    <tr>  
        <td>Gender:</td>  
        <td><%=gender%></td>  
    </tr>  
    <tr>  
        <td>Mark obtained:</td>  
        <td><%=percMarkObtained%></td>  
    </tr>  
    <%  
    }  
    %>  
</table>  
<p>  
    Please click <a href="menu.jsp">here</a> to get back to the menu page or <a href="index.html">here</a>  
    to the main page.  
</p>  
</body>  
</html>
```

This screenshot shows the continuation of the JSP code. It includes a table displaying student details (ID, Name, Surname, Age, Gender, and Mark obtained) and a message at the bottom encouraging the user to click links for the menu or index pages.

```
<table>  
    <tr>  
        <td>ID:</td>  
        <td><%=id%></td>  
    </tr>  
    <tr>  
        <td>Name:</td>  
        <td><%=name%></td>  
    </tr>  
    <tr>  
        <td>Surname:</td>  
        <td><%=surname%></td>  
    </tr>  
    <tr>  
        <td>Age:</td>  
        <td><%=age%></td>  
    </tr>  
    <tr>  
        <td>Gender:</td>  
        <td><%=gender%></td>  
    </tr>  
    <tr>  
        <td>Mark obtained:</td>  
        <td><%=percMarkObtained%></td>  
    </tr>  
    <%  
    }  
    %>  
</table>  
<p>  
    Please click <a href="menu.jsp">here</a> to get back to the menu page or <a href="index.html">here</a>  
    to the main page.  
</p>  
</body>  
</html>
```

Create the get_student_using_name_surname_outcome.jsp file.

The screenshot shows the code editor interface for a JSP file named "get_student_using_name_surname_outcome.jsp". The code is as follows:

```
<%--  
    Document      : get_student_using_name_surname_outcome  
    Created on   : 08 Apr 2023, 11:39:59 AM  
    Author        : MemaniV  
--%>  
  
<%@page import="za.ac.tut.entities.Student"%>  
<%@page contentType="text/html" pageEncoding="UTF-8"%>  
<!DOCTYPE html>  
<html>  
    <head>  
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
        <title>Get Student Using Name and Surname Outcome Page</title>  
    </head>  
    <body>  
        <h1>Get student using name and surname outcome</h1>  
        <%  
            Student p = (Student) request.getAttribute("student");  
            Long id = p.getId();  
            String name = p.getFirstName();  
            String surname = p.getLastName();  
            String gender = p.getGender();  
            Integer age = p.getAge();  
            Double percMarkObtained = p.getPercMarkObtained();  
        %>  
        <p>  
            Below are the details of the student:  
        </p>  
    </body>  
</html>
```

The screenshot shows the continuation of the code editor for the same JSP file. The code is as follows:

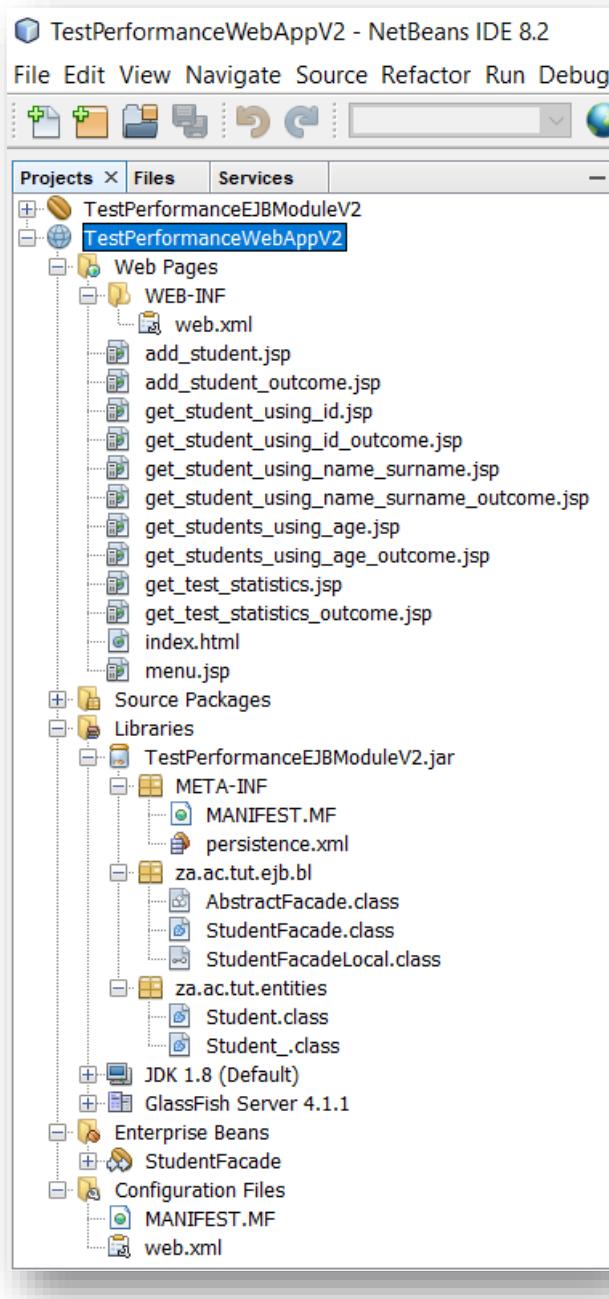
```
29 | <table>  
30 |     <tr>  
31 |         <td>ID:</td>  
32 |         <td><%=id%></td>  
33 |     </tr>  
34 |     <tr>  
35 |         <td>Name:</td>  
36 |         <td><%=name%></td>  
37 |     </tr>  
38 |     <tr>  
39 |         <td>Surname:</td>  
40 |         <td><%=surname%></td>  
41 |     </tr>  
42 |     <tr>  
43 |         <td>Age:</td>  
44 |         <td><%=age%></td>  
45 |     </tr>  
46 |     <tr>  
47 |         <td>Gender:</td>  
48 |         <td><%=gender%></td>  
49 |     </tr>  
50 |     <tr>  
51 |         <td>Mark obtained:</td>  
52 |         <td><%=percMarkObtained%></td>  
53 |     </tr>  
54 |     <%  
55 |         }  
56 |     %>  
57 | </table>  
58 | <p>  
59 |     Please click <a href="menu.jsp">here</a> to get back to the menu page or <a href="index.html">here</a>  
60 |     to the main page.  
61 | </p>  
62 | </body>  
63 | </html>
```

Create the **get_student_using_age_outcome.jsp** file.

```
26 | <table>
27 |     <%>
28 |         for(int i = 0; i < list.size(); i++){
29 |             Student p = list.get(i);
30 |             Long id = p.getId();
31 |             String name = p.getFirstName();
32 |             String surname = p.getLastName();
33 |             String gender = p.getGender();
34 |             Integer age = p.getAge();
35 |             Double percMarkObtained = p.getpercMarkObtained();
36 |
37 |             <tr>
38 |                 <td>ID:</td>
39 |                 <td><%=id%></td>
40 |             </tr>
41 |             <tr>
42 |                 <td>Name:</td>
43 |                 <td><%=name%></td>
44 |             </tr>
45 |             <tr>
46 |                 <td>Surname:</td>
47 |                 <td><%=surname%></td>
48 |             </tr>
49 |             <tr>
50 |                 <td>Age:</td>
51 |                 <td><%=age%></td>
52 |             </tr>
53 |             <tr>
54 |                 <td>Gender:</td>
55 |                 <td><%=gender%></td>
56 |             </tr>
57 |             <tr>
58 |                 <td>Mark obtained:</td>
59 |                 <td><%=percMarkObtained%></td>
60 |             </tr>
```

```
61 <%
62     }
63     %>
64 </table>
65 <p>
66     Please click <a href="menu.jsp">here</a> to get back to the menu page or <a href="index.html">here</a>
67     to the main page.
68 </p>
69 </body>
70 </html>
```

View the complete project structure of **TestPerformanceWebAppV2**.



Compile the project.

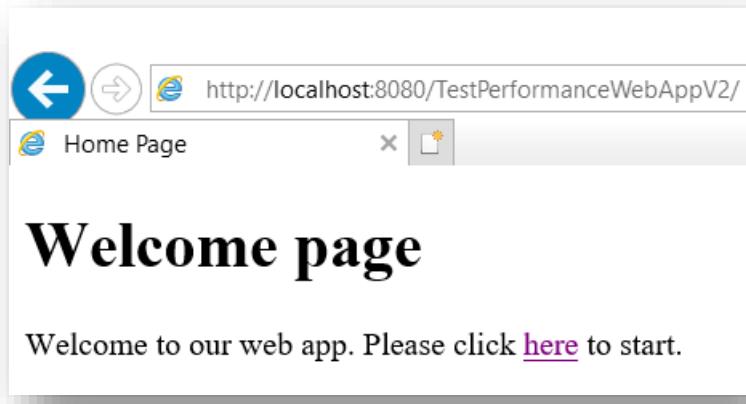
```
Output X
Java DB Database Process x GlassFish Server 4.1.1 x TestPerformanceWebAppV2 (clean,dist) x
library-inclusion-in-archive:
Copying 1 file to C:\Users\memaniv\Documents\NetBeansProjects\TestPerformanceWebAppV2\build\web\WEB-INF\lib
library-inclusion-in-manifest:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\TestPerformanceWebAppV2\build\empty
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\TestPerformanceWebAppV2\build\generated-sources\ap-source-output
Compiling 5 source files to C:\Users\memaniv\Documents\NetBeansProjects\TestPerformanceWebAppV2\build\web\WEB-INF\classes
compile:
compile-jsp:
Created dir: C:\Users\memaniv\Documents\NetBeansProjects\TestPerformanceWebAppV2\dist
Building jar: C:\Users\memaniv\Documents\NetBeansProjects\TestPerformanceWebAppV2\dist\TestPerformanceWebAppV2.war
do-dist:
dist:
BUILD SUCCESSFUL (total time: 0 seconds)
```

Deploy the project.

```
Output X
Java DB Database Process x GlassFish Server 4.1.1 x TestPerformanceWebAppV2 (run-deploy) x
Info: visiting unvisited references
Info: visiting unvisited references
Info: EclipseLink, version: Eclipse Persistence Services - 2.6.1.v20150605-31e8258
Info: /file:/C:/Users/memaniv/Documents/NetBeansProjects/TestPerformanceWebAppV2/build/web/WEB-INF/
Info: Portable JNDI names for EJB StudentFacade: [java:global/TestPerformanceWebAppV2/StudentFacade]
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] private org.glassfish.jersey.ext.cdi1x.i...
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] org.glassfish.sse.impl.ServerSentEventCo...
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] org.glassfish.sse.impl.ServerSentEventCo...
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] public org.glassfish.jms.injection.JMSCDI...
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] private org.glassfish.jersey.ext.cdi1x.i...
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] public org.glassfish.jms.injection.JMSCDI...
Info: Loading application [TestPerformanceWebAppV2] at [/TestPerformanceWebAppV2]
Info: TestPerformanceWebAppV2 was successfully deployed in 421 milliseconds.
```

Part E – Run the web application.

Launch the application.



Click on the link

The screenshot shows a web browser window with the URL <http://localhost:8080/TestPerformanceWebAppV2/menu.jsp>. The title bar says "Menu Page". The main content area has a large bold "Menu" heading. Below it, a message says "Please select one of the following options:" followed by a numbered list of five items, each with a link labeled "here".

1. Click [here](#) to add a student.
2. Click [here](#) to get test statistics.
3. Click [here](#) to get a student using id as criteria.
4. Click [here](#) to get a specific student using name and surname as criteria.
5. Click [here](#) to get students using age as a criteria.

Add a student.

- ✓ Click on the first link.

The screenshot shows a web browser window with the URL http://localhost:8080/TestPerformanceWebAppV2/add_student.jsp. The title bar says "Add Student Page". The main content area has a large bold "Add student" heading. Below it, a message says "Please add student details below:". There are six input fields: "Student number:", "Name:", "Surname:", "Age:", "Gender:" (with a dropdown menu showing "F"), and "Mark obtained:". At the bottom is a "ADD STUDENT" button.

- ✓ Fill in the form.

Please add student details below:

Student number:

Name:

Surname:

Age:

Gender:

Mark obtained:

ADD STUDENT

Click on the add button.

here to get back to the menu page or [here](#) to the main page.'"/>

Add student outcome

The student has been successfully persisted into the database. Please click [here](#) to get back to the menu page or [here](#) to the main page.

Add nine more students.

- ✓ View database structure.

The screenshot shows the database structure for the APP schema. The STUDENT table has the following columns:

ID	AGE	FIRSTNAME	GENDER	LASTNAME	PERCMARKOBTAINED
100	21	Vuyi	M	Memza	90.0
101	18	Tiro	M	Nthane	30.0
103	17	Patrice	M	Mulumba	70.0
104	22	Beauty	F	Zwane	40.0
105	23	Zuki	F	Mahlangu	90.0
106	25	Lira	F	Rananaime	100.0
107	20	Loyi	M	Sibya	20.0
108	23	Rato	F	Nthane	60.0
109	19	Tumi	M	Musenga	90.0
200	16	Tshilo	F	Baloyi	50.0

- ✓ View records.

#	AGE	FIRSTNAME	GENDER	LASTNAME	PERCMARKOBTAINED
1	100	Vuyi	M	Memza	90.0
2	101	Tiro	M	Nthane	30.0
3	103	Patrice	M	Mulumba	70.0
4	104	Beauty	F	Zwane	40.0
5	105	Zuki	F	Mahlangu	90.0
6	106	Lira	F	Rananaime	100.0
7	107	Loyi	M	Sibya	20.0
8	108	Rato	F	Nthane	60.0
9	109	Tumi	M	Musenga	90.0
10	200	Tshilo	F	Baloyi	50.0

View all the students.

- ✓ Click on the second link.

Get test statistics

Please click on the button below to get test statistics.

GET TEST STATISTICS

- ✓ Click on the button.

The screenshot shows a web browser window with the URL <http://localhost:8080/TestPerformanceWebAppV2/GetTestStatisticsServlet.do>. The title bar of the browser says "Get test statistics outcome". The page content displays the following text:

Below are the test statistics:

Number of students that wrote the test: 10
Number of students that passed: 7
Number of students that failed: 3
Number of male students: 5
Number of female students: 5
Number of male students that passed: 3
Number of female students that passed: 4
Number of male students that failed: 2
Number of female students that failed: 1
Highest mark: 100.0%
Lowest mark: 20.0%
Average mark: 64.0%

Below is the class list:

ID: 100
Name: Vuyi
Surname: Memza
Age: 21
Gender: M
Mark obtained: 90.0%

ID: 101
Name: Tiro
Surname: Nthane
Age: 18
Gender: M
Mark obtained: 30.0%

ID: 103
Name: Patrice
Surname: Mulumba
Age: 17
Gender: M
Mark obtained: 70.0%

ID: 104
Name: Beauty
Surname: Zwane
Age: 22
Gender: F
Mark obtained: 40.0%

ID: **105**
Name: Zuki
Surname: Mahlangu
Age: 23
Gender: F
Mark obtained: 90.0%

ID: **106**
Name: Lira
Surname: Rananame
Age: 25
Gender: F
Mark obtained: 100.0%

ID: **107**
Name: Loyi
Surname: Sibiya
Age: 20
Gender: M
Mark obtained: 20.0%

ID: **108**
Name: Rato
Surname: Nthane
Age: 23
Gender: F
Mark obtained: 60.0%

ID: **109**
Name: Tumi
Surname: Musenga
Age: 19
Gender: M
Mark obtained: 90.0%

ID: **200**
Name: Tshilo
Surname: Baloyi
Age: 16
Gender: F
Mark obtained: 50.0%

Please click [here](#) to get back to the menu page or [here](#) to the main page.

Get student using id.

- ✓ Click on the third link.

The screenshot shows a web browser window with the URL http://localhost:8080/TestPerformanceWebAppV2/get_student_using_id.jsp. The title bar says "Get Student Using ID Page". The main content area has a large heading "Get student using id" and a sub-instruction "Please enter the ID of the student below.". Below this is a form with an "ID:" label and a text input field containing "200", followed by a "GET" button.

- ✓ Fill-in the form.

The screenshot shows the same web browser window as before, but now the text input field contains "200". The "GET" button is still present below the input field.

- ✓ Click on the button.

The screenshot shows the web browser window after the button was clicked. The title bar now says "Get Student Using ID Outco...". The main content area has a large heading "Get student using id outcome" and a sub-instruction "Below are the details of the student:". Below this is a table of student details:

ID:	200
Name:	Tshilo
Surname:	Baloyi
Age:	16
Gender:	F
Mark obtained:	50.0%

At the bottom, there is a message: "Please click [here](#) to get back to the menu page or [here](#) to the main page."

Get student using name and surname.

- ✓ Click on the fourth link.

The screenshot shows a web browser window with the URL http://localhost:8080/TestPerformanceWebAppV2/get_student_using_name_surname.jsp. The title bar says "Get Student Using Name an...". The page content is titled "Get student using name and surname" and instructs the user to "Please enter the name and surname of the student below.". There are two input fields: "Name:" and "Surname:", both currently empty. Below them is a "GET" button.

- ✓ Fill-in the form.

The screenshot shows the same web browser window after the user has filled in the form. The "Name:" field contains "Rato" and the "Surname:" field contains "Nthane". The "GET" button is still present below the fields.

- ✓ Click on the button.

The screenshot shows the web browser after the "GET" button was clicked. The title bar now says "Get Student Using Name an...". The page content is titled "Get student using name and surname outcome" and states "Below are the details of the student:". A table displays the following information:

ID:	108
Name:	Rato
Surname:	Nthane
Age:	23
Gender:	F
Mark obtained:	60.0%

Please click [here](#) to get back to the menu page or [here](#) to the main page.

Get student using age.

- ✓ Click on the fifth link.

http://localhost:8080/TestPerformanceWebAppV2/get_students_using_age.jsp

Get Student Using Age Page

Get student using age

Please enter the age values of the student below.

Minimum age:

Maximum age:

GET

- ✓ Fill-in the form.

http://localhost:8080/TestPerformanceWebAppV2/get_students_using_age.jsp

Get Student Using Age Page

Get student using age

Please enter the age values of the student below.

Minimum age:

Maximum age:

GET

- ✓ Click on the button.

http://localhost:8080/TestPerformanceWebAppV2/GetStudentUsingAgeServlet.do

Get students using age outcome

Below are the students within the age group 22 and 23.

ID:	104
Name:	Beauty
Surname:	Zwane
Age:	22
Gender:	F
Mark obtained:	40.0%
ID:	105
Name:	Zuki
Surname:	Mahlangu
Age:	23
Gender:	F
Mark obtained:	90.0%
ID:	108
Name:	Rato
Surname:	Nthane
Age:	23
Gender:	F
Mark obtained:	60.0%

Please click [here](#) to get back to the menu page or [here](#) to the main page.

10 Appendix A: How to download and install JDK

The purpose of this document is to take you through the steps of downloading and installing JDK in your computer.

- Click on the link below which will take you to the Oracle website.

[Java Downloads | Oracle South Africa](https://www.oracle.com/za/java/technologies/downloads/#jdk19-windows)

The screenshot shows a web browser window with the URL <https://www.oracle.com/za/java/technologies/downloads/#jdk19-windows>. The page title is "Java Downloads | Oracle South Africa". Below the title, there are tabs for "Java downloads", "Tools and resources", and "Java archive". The main content area is titled "Java SE Development Kit 19.0.1 downloads". It says "Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications and components using the Java programming language." Below this, there are three tabs: "Linux", "macOS", and "Windows", with "Windows" being the active tab. A table lists three download options:

Product/file description	File size	Download
x64 Compressed Archive	179.13 MB	https://download.oracle.com/java/19/latest/jdk-19_windows-x64_bin.zip (sha256)
x64 Installer	158.91 MB	https://download.oracle.com/java/19/latest/jdk-19_windows-x64_bin.exe (sha256)
x64 MSI Installer	157.76 MB	https://download.oracle.com/java/19/latest/jdk-19_windows-x64_bin.msi (sha256)

- Click on the **x64 Compressed Archive** link and download to your local drive.

The screenshot shows a Windows File Explorer window with the path "This PC > Documents > software > jdk". The file "jdk-19_windows-x64_bin" is selected. The file details are shown in the table below:

Name	Date modified	Type	Size
jdk-19_windows-x64_bin	2023/01/10 12:54	ZipGenius Zip File	183 426 KB

- Unzip the file.

The screenshot shows a Windows File Explorer window with the path "This PC > Documents > software > jdk". The file "jdk-19_windows-x64_bin" is selected. The file details are shown in the table below:

Name	Date modified	Type	Size
jdk-19.0.1	2023/01/10 13:00	File folder	
jdk-19_windows-x64_bin	2023/01/10 12:54	ZipGenius Zip File	183 426 KB

- Expand the folder.

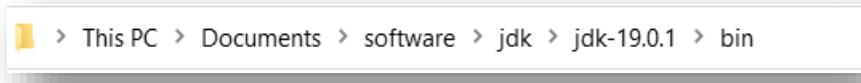
Name	Date modified	Type	Size
bin	2023/01/10 13:00	File folder	
conf	2023/01/10 13:00	File folder	
include	2023/01/10 13:00	File folder	
jmods	2023/01/10 13:00	File folder	
legal	2023/01/10 13:00	File folder	
lib	2023/01/10 13:00	File folder	
LICENSE	2022/09/14 13:06	File	7 KB
README	2022/09/14 13:06	File	1 KB
release	2022/09/14 13:06	File	2 KB

- Expand the **bin** folder.

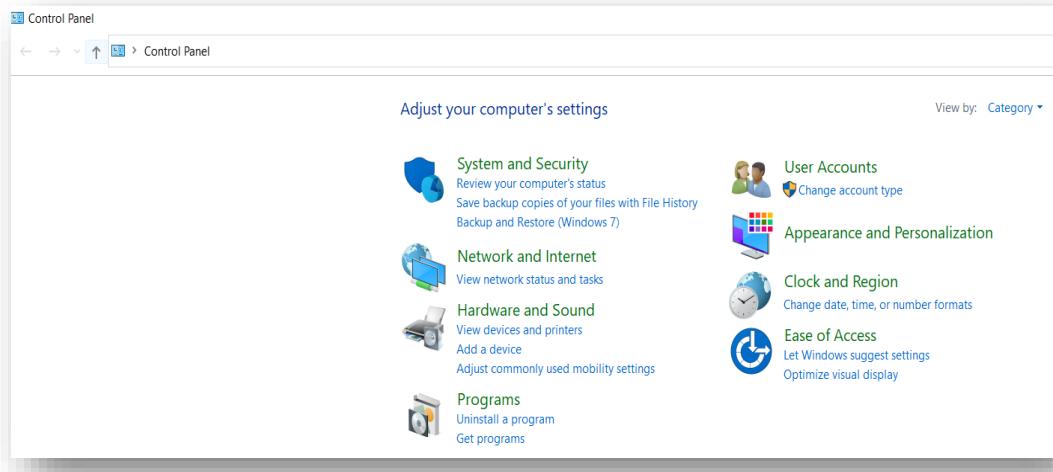
Name	Date modified	Type	Size
java	2022/09/14 13:06	Application	54 KB
javaaccessbridge.dll	2022/09/14 13:06	Application extens...	285 KB
javac	2022/09/14 13:06	Application	24 KB
javadoc	2022/09/14 13:06	Application	24 KB
javajpeg.dll	2022/09/14 13:06	Application extens...	178 KB
javap	2022/09/14 13:06	Application	24 KB
javaw	2022/09/14 13:06	Application	54 KB
jawt.dll	2022/09/14 13:06	Application extens...	21 KB
jcmd	2022/09/14 13:06	Application	24 KB
jconsole	2022/09/14 13:06	Application	24 KB

The **bin** directory contains all the applications available to a Java programmer. In our case we are going to use **javac (java compiler)** for compiling programs **and java** for running programs.

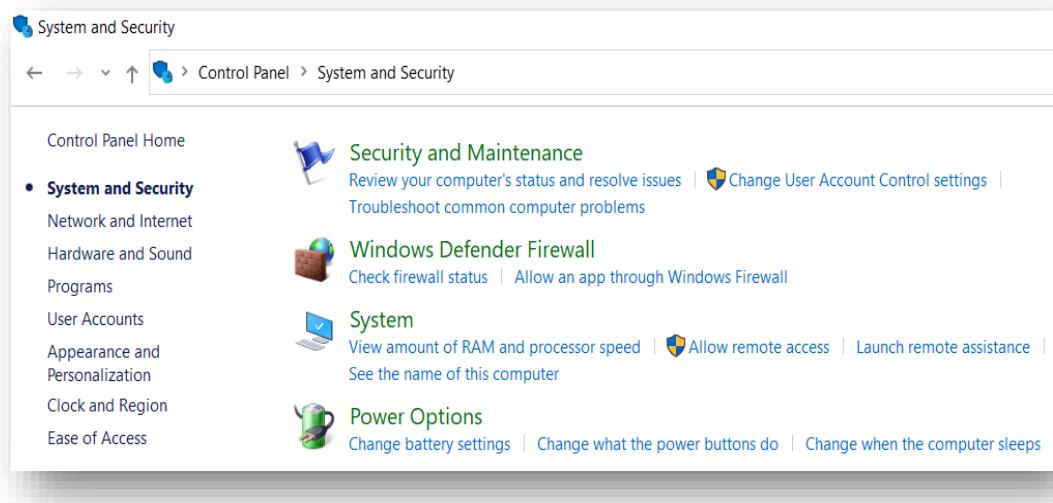
- Select and copy the **bin** location path.



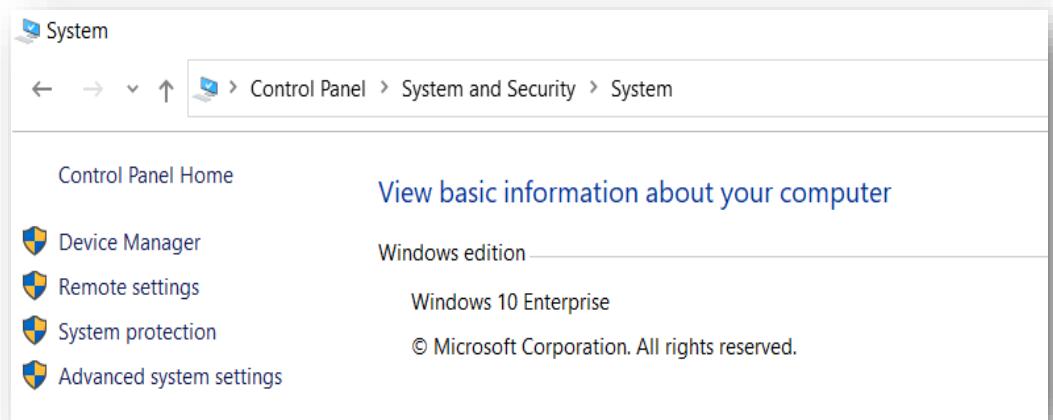
- Go to the control panel of your computer to set the PATH environment variable. On the search space of your computer (normally bottom left of your task bar), enter Control Panel.



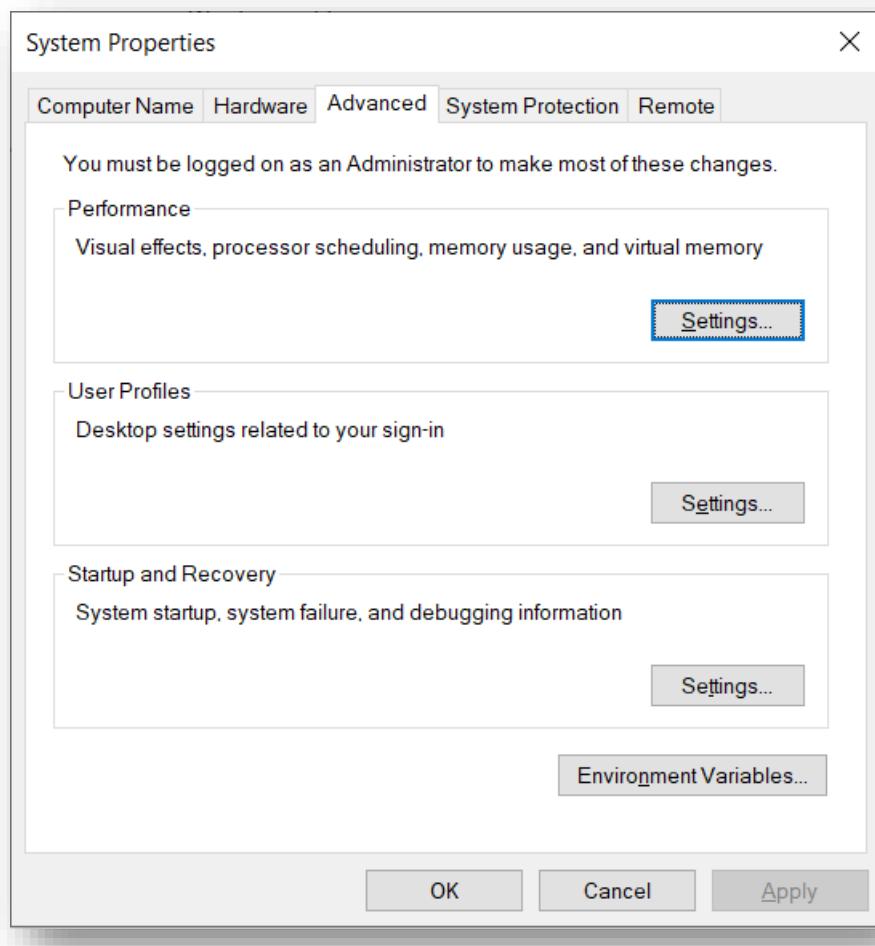
- Click on **System and Security**.



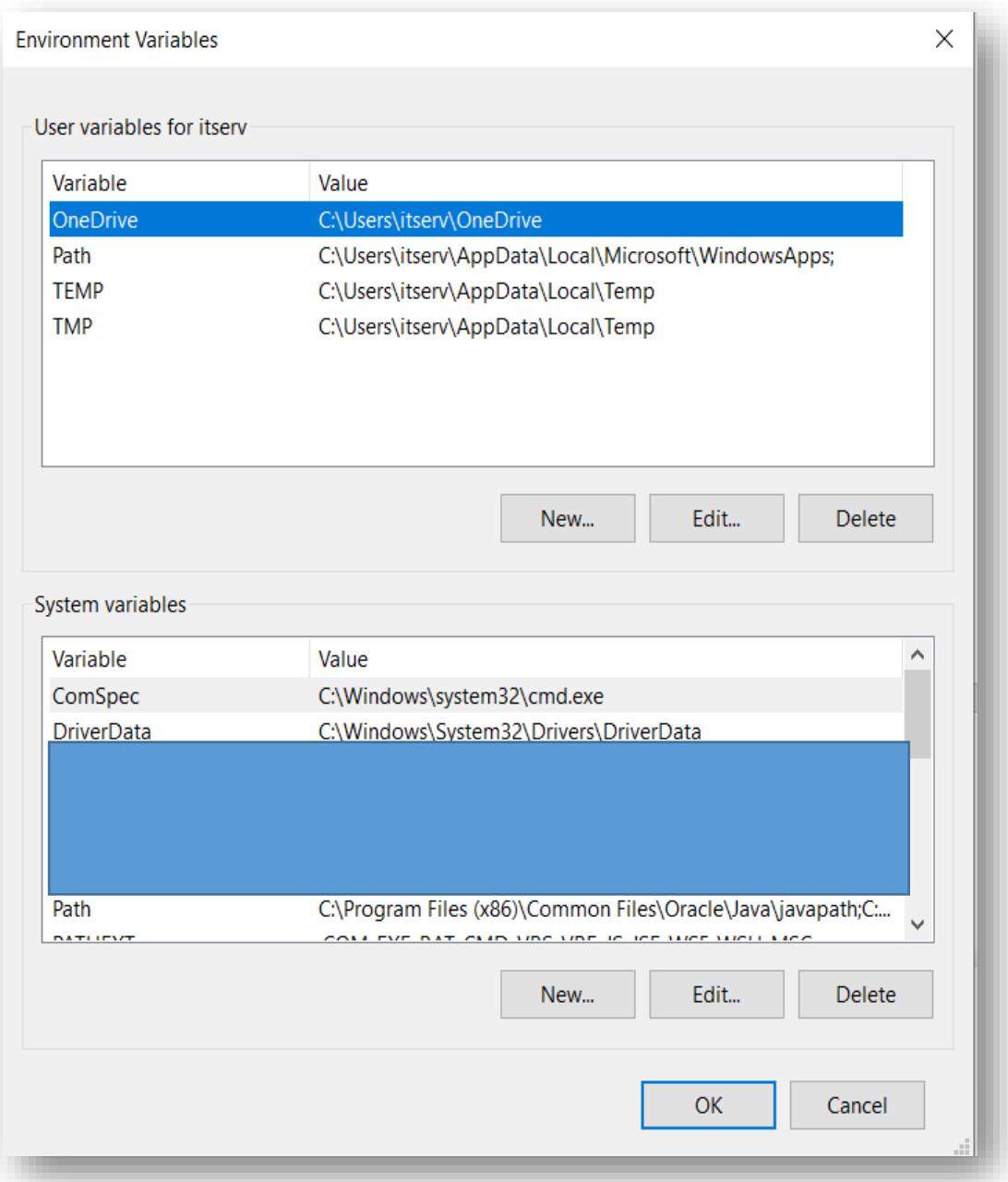
- Click on **System**.



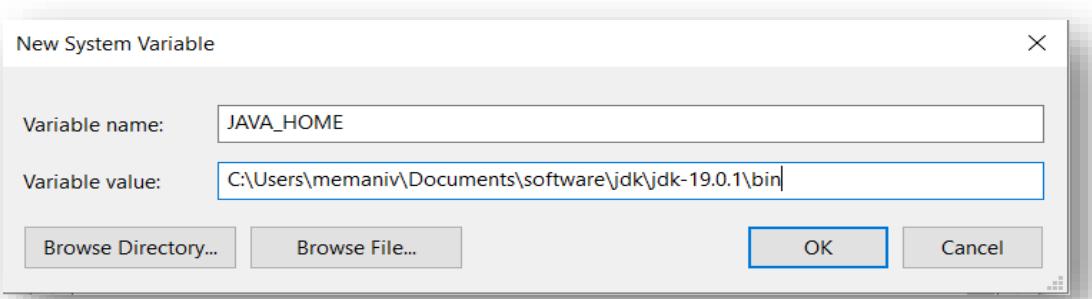
- Click on Advanced system settings.



- Click on **Environment Variables**.



- Under System variables click on **New**.

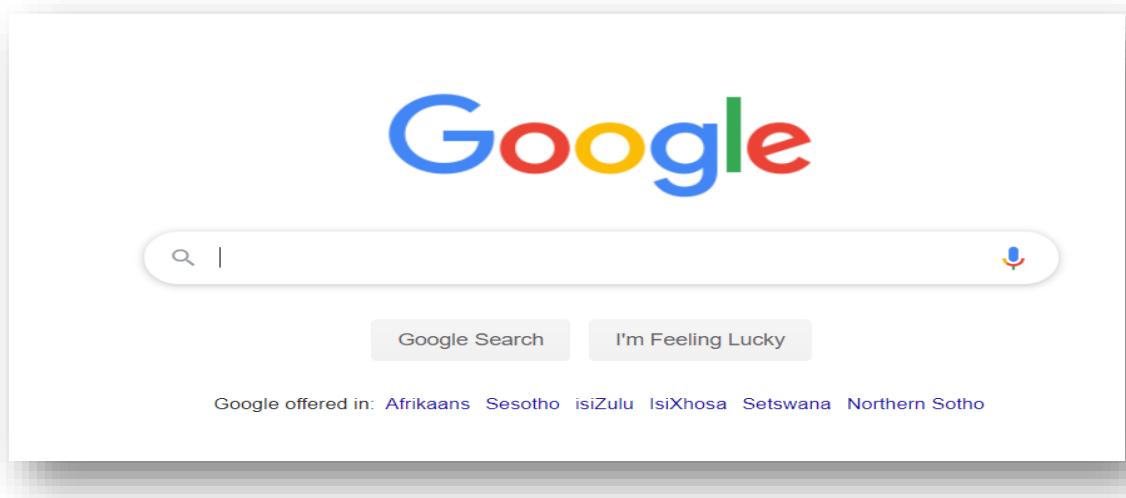


- Click OK, OK, OK until you are done.

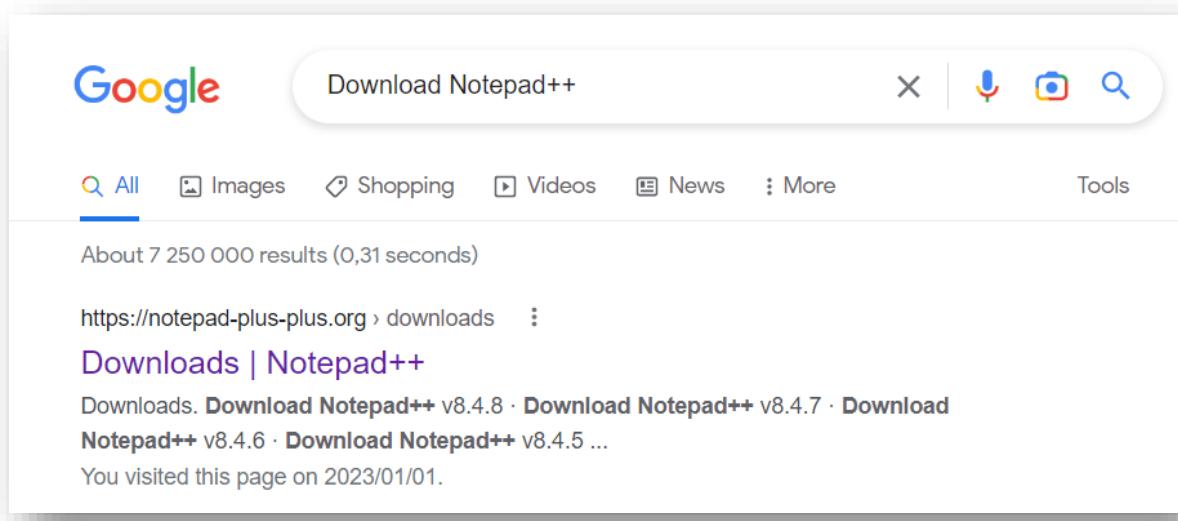
11 Appendix B: How to download and install Notepad++

The purpose of this document is to take you through the steps of downloading and installing Notepad++ in your computer.

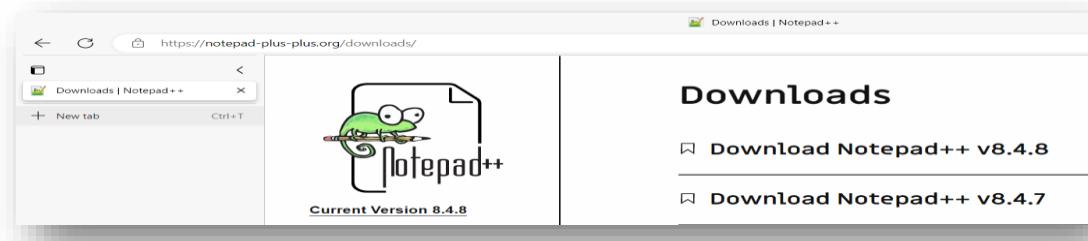
- Open Google (<http://www.google.com>)



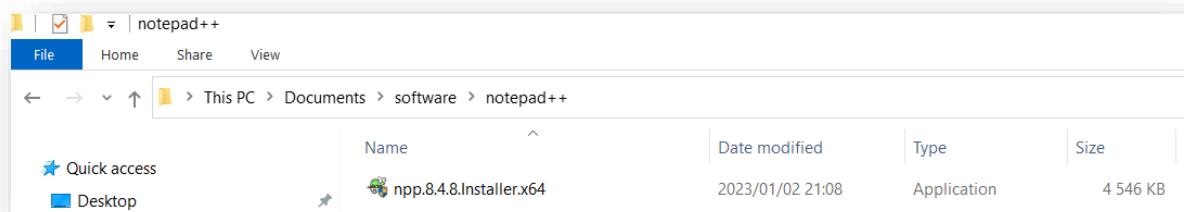
- Enter “Download Notepad++” on the search field.



- Click on the first link.



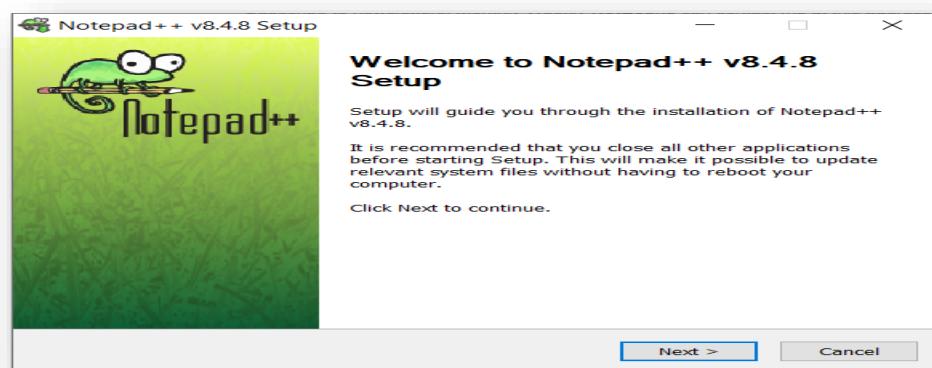
- Click on **Download Notepad++ v8.4.8** and store in your local drive.



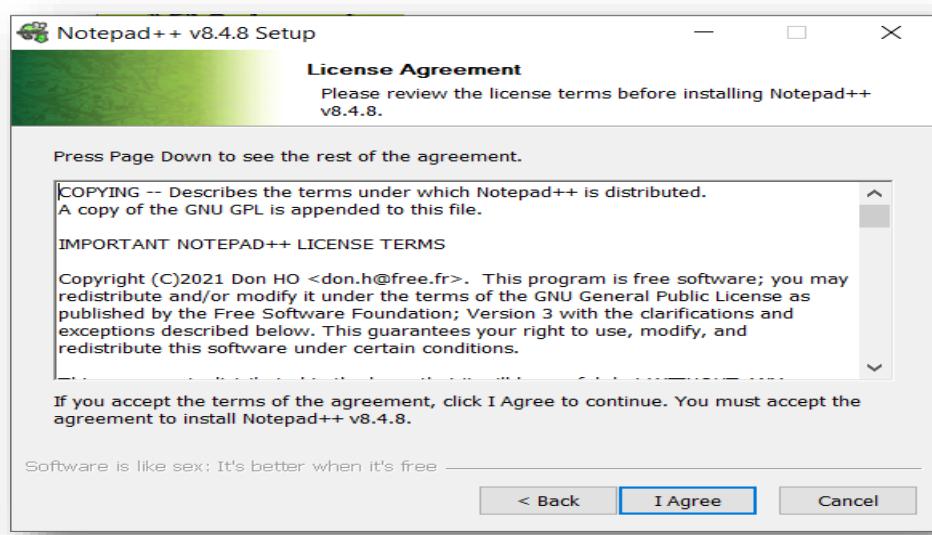
- Double click on the application.



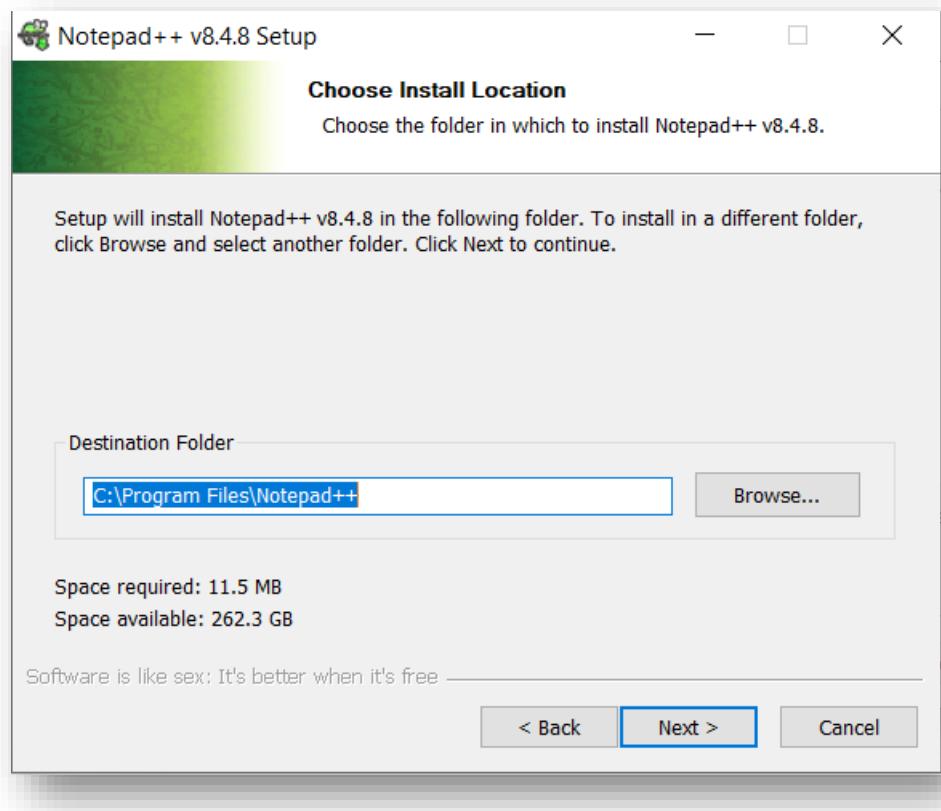
- Click on **OK**.



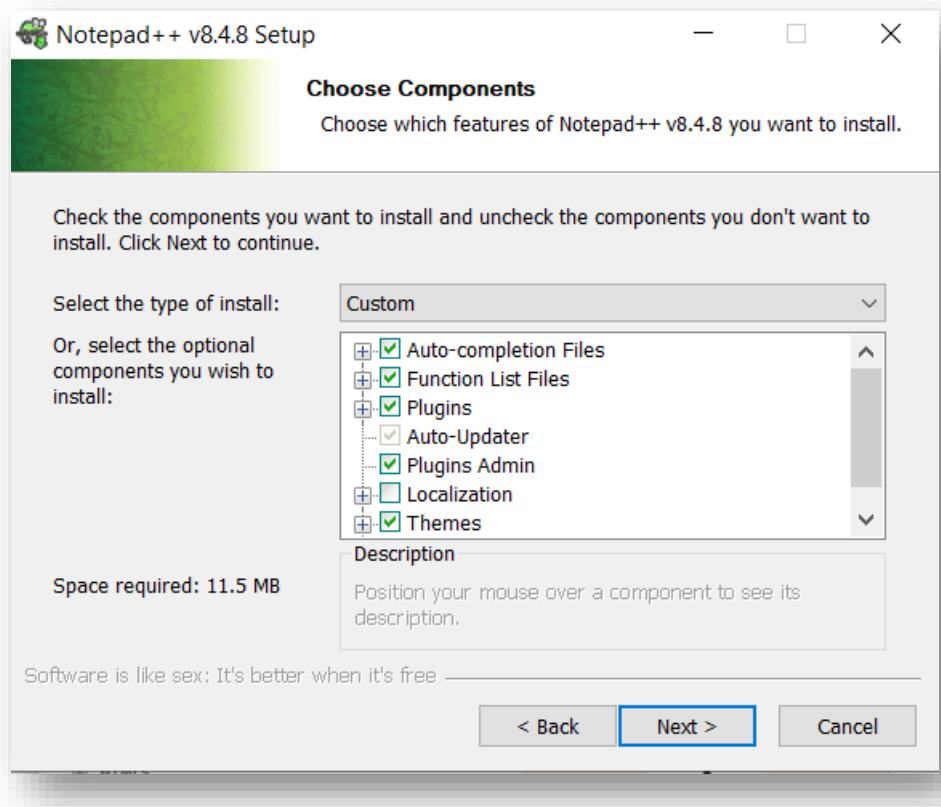
- Click on **Next**.



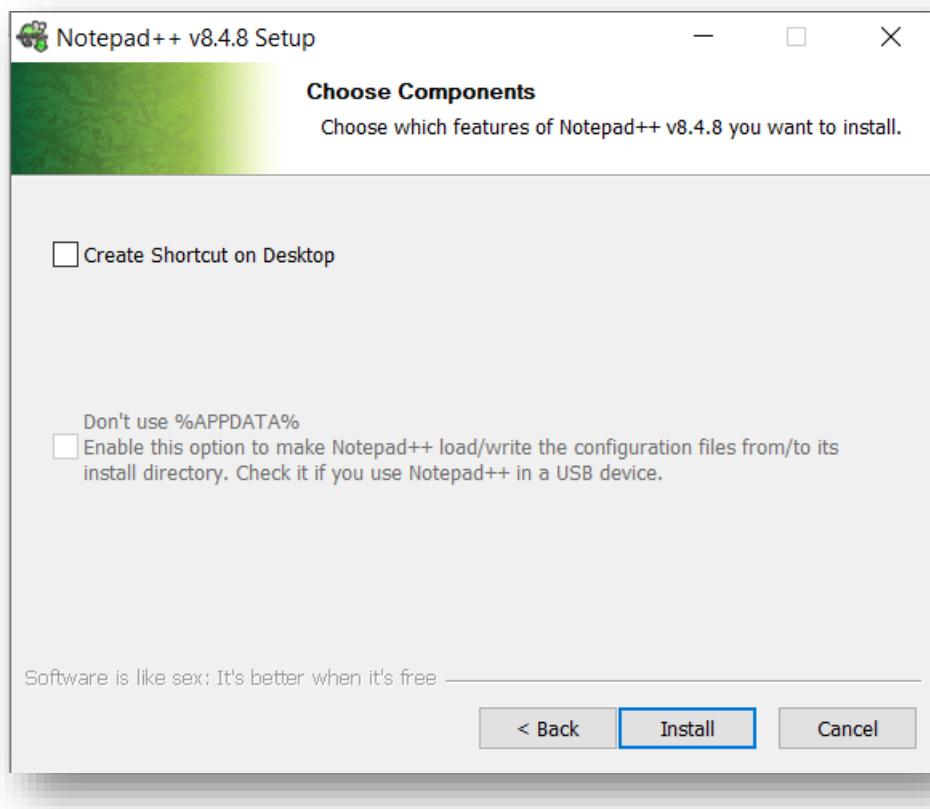
- Click on “I Agree”



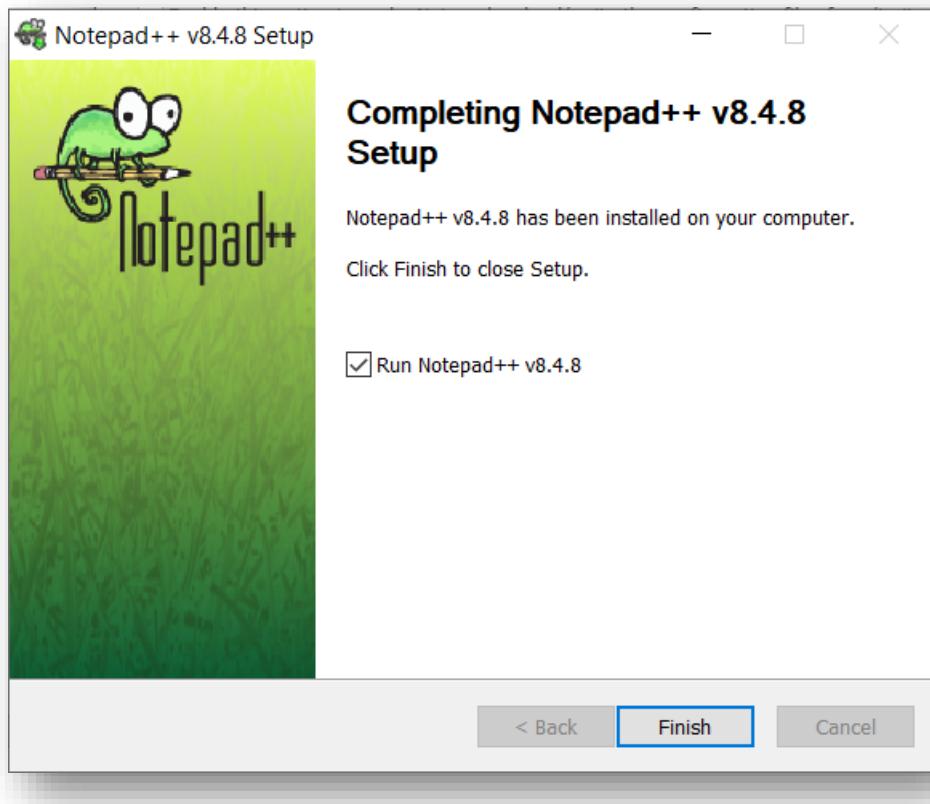
- Click on **Next**.



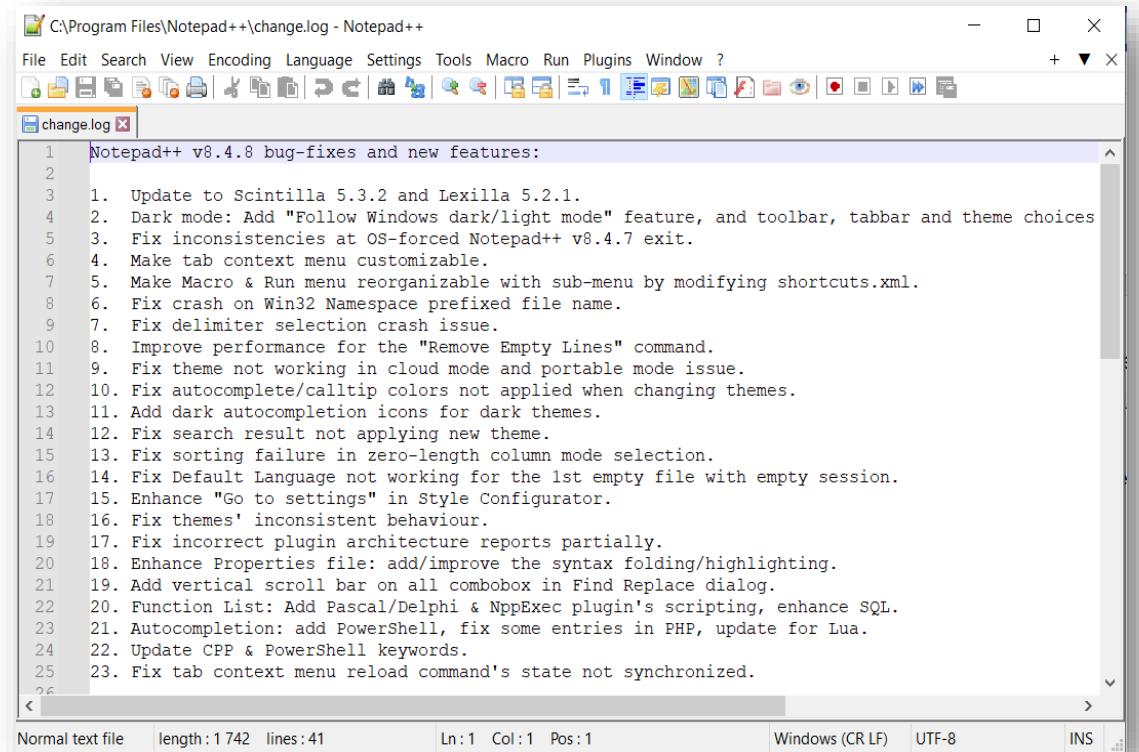
- Click on **Next**.



- Select the **Create Shortcut on Desktop** checkbox and click on **Install**.



- Click on **Finish**.



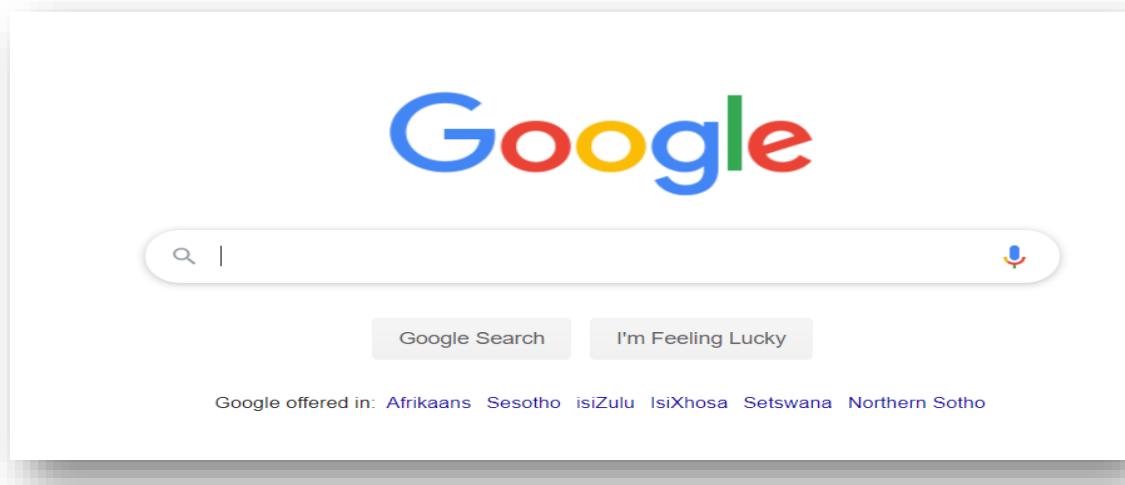
The screenshot shows the Notepad++ application window with the title bar "C:\Program Files\Notepad++\change.log - Notepad++". The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Tools, Macro, Run, Plugins, Window, and Help. The toolbar below the menu bar contains various icons for file operations like Open, Save, Print, Find, Replace, and Plugins. The main editor window displays the "change.log" file content, which is a list of 25 items detailing bug-fixes and new features for Notepad++ v8.4.8. The text is numbered from 1 to 25. The status bar at the bottom shows "Normal text file", "length: 1742 lines: 41", "Ln: 1 Col: 1 Pos: 1", "Windows (CR LF)", "UTF-8", and "INS".

```
1 Notepad++ v8.4.8 bug-fixes and new features:
2
3 1. Update to Scintilla 5.3.2 and Lexilla 5.2.1.
4 2. Dark mode: Add "Follow Windows dark/light mode" feature, and toolbar, tabbar and theme choices
5 3. Fix inconsistencies at OS-forced Notepad++ v8.4.7 exit.
6 4. Make tab context menu customizable.
7 5. Make Macro & Run menu reorganizable with sub-menu by modifying shortcuts.xml.
8 6. Fix crash on Win32 Namespace prefixed file name.
9 7. Fix delimiter selection crash issue.
10 8. Improve performance for the "Remove Empty Lines" command.
11 9. Fix theme not working in cloud mode and portable mode issue.
12 10. Fix autocomplete/calltip colors not applied when changing themes.
13 11. Add dark autocompletion icons for dark themes.
14 12. Fix search result not applying new theme.
15 13. Fix sorting failure in zero-length column mode selection.
16 14. Fix Default Language not working for the 1st empty file with empty session.
17 15. Enhance "Go to settings" in Style Configurator.
18 16. Fix themes' inconsistent behaviour.
19 17. Fix incorrect plugin architecture reports partially.
20 18. Enhance Properties file: add/improve the syntax folding/highlighting.
21 19. Add vertical scroll bar on all combobox in Find Replace dialog.
22 20. Function List: Add Pascal/Delphi & NppExec plugin's scripting, enhance SQL.
23 21. Autocompletion: add PowerShell, fix some entries in PHP, update for Lua.
24 22. Update CPP & PowerShell keywords.
25 23. Fix tab context menu reload command's state not synchronized.
26
```

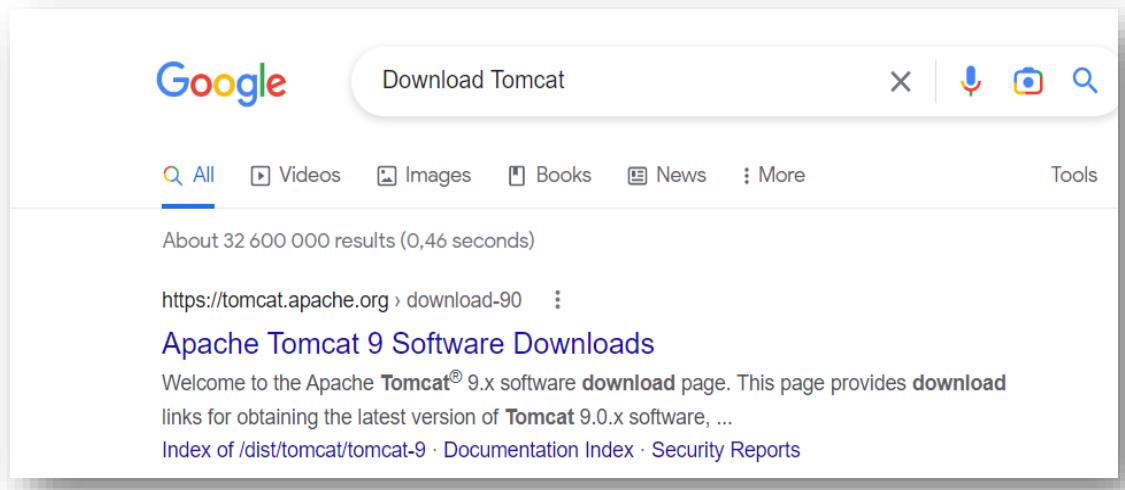
12 Appendix C: How to download and install Tomcat

In this section we take you through the steps of downloading and installing Tomcat.

- Open Google (<http://www.google.com>)



- Type “Download Tomcat”



- Click on the link.

Tomcat 8 Software Downloads

Welcome to the Apache Tomcat® 8.x software download page. This well as links to the archives of older releases.

Unsure which version you need? Specification versions implemented in the ['which version?' page](#).

Quick Navigation

[KEYS](#) | [8.5.84](#) | [Browse](#) | [Archives](#)

Release Integrity

- Scroll down to **Binaries**.

8.5.84

Please see the [README](#) file for packaging information. It explains what every distribution contains.

Binary Distributions

- Core:
 - [zip \(pgp, sha512\)](#)
 - [tar.gz \(pgp, sha512\)](#)
 - [32-bit Windows zip \(pgp, sha512\)](#)
 - [64-bit Windows zip \(pgp, sha512\)](#)
 - [32-bit/64-bit Windows Service Installer \(pgp, sha512\)](#)
- Full documentation:
 - [tar.gz \(pgp, sha512\)](#)
 - [tar.gz \(pgp, sha512\)](#)
- Deployer:
 - [zip \(pgp, sha512\)](#)
 - [tar.gz \(pgp, sha512\)](#)
- Extras:
 - [Web services.jar \(pgp, sha512\)](#)
- Embedded:
 - [tar.gz \(pgp, sha512\)](#)
 - [zip \(pgp, sha512\)](#)

Source Code Distributions

- [tar.gz \(pgp, sha512\)](#)
- [zip \(pgp, sha512\)](#)

- Under Core, click on the **64-bit Windows** zip link to download to your local drive, and unzip.
- Open the root folder of Tomcat.

Name	Date modified	Type	Size
bin	2022/12/12 10:11	File folder	
conf	2022/12/12 10:11	File folder	
lib	2022/12/12 10:11	File folder	
logs	2023/01/20 15:01	File folder	
temp	2022/12/12 10:11	File folder	
webapps	2023/01/20 13:19	File folder	
work	2022/12/12 10:11	File folder	
LICENSE	2022/11/16 15:34	File	57 KB
NOTICE	2022/11/16 15:34	File	2 KB
RELEASE-NOTES	2022/11/16 15:34	File	8 KB
tomcat	2022/11/16 15:34	Icon	22 KB
Uninstall	2022/11/16 15:34	Application	86 KB

The **bin** directory contains applications for working with Tomcat. Two of the commonly used applications is **startup** and **shutdown**. The startup application is used to start Tomcat, and shutdown is used to stop Tomcat.

Name	Date modified	Type	Size
bootstrap	2022/11/16 15:34	Executable Jar File	36 KB
catalina	2022/11/16 15:34	Windows Batch File	17 KB
ciphers	2022/11/16 15:34	Windows Batch File	3 KB
configtest	2022/11/16 15:34	Windows Batch File	2 KB
digest	2022/11/16 15:34	Windows Batch File	3 KB
service	2022/11/16 15:34	Windows Batch File	9 KB
setclasspath	2022/11/16 15:34	Windows Batch File	4 KB
shutdown	2022/11/16 15:34	Windows Batch File	2 KB
startup	2022/11/16 15:34	Windows Batch File	2 KB
Tomcat8	2022/11/16 15:34	Application	142 KB
Tomcat8w	2022/11/16 15:34	Application	126 KB
tomcat-juli	2022/11/16 15:34	Executable Jar File	52 KB
tool-wrapper	2022/11/16 15:34	Windows Batch File	5 KB
version	2022/11/16 15:34	Windows Batch File	2 KB

The **config** directory consists of files to configure Tomcat. One such file is `server.xml`

Name	Date modified	Type	Size
Catalina	2022/12/12 10:11	File folder	
catalina.policy	2022/11/16 15:34	POLICY File	13 KB
catalina.properties	2022/11/16 15:34	PROPERTIES File	8 KB
context	2022/11/16 15:34	XML Document	2 KB
jaspic-providers	2022/11/16 15:34	XML Document	2 KB
jaspic-providers.xsd	2022/11/16 15:34	XSD File	3 KB
logging.properties	2022/11/16 15:34	PROPERTIES File	4 KB
server	2023/01/13 11:57	XML Document	8 KB
tomcat-users	2022/12/12 10:11	XML Document	3 KB
tomcat-users.xsd	2022/11/16 15:34	XSD File	3 KB
web	2022/11/16 15:34	XML Document	173 KB

The **lib** directory contains APIs (libraries) that are supported by Tomcat. You can also put your user-defined APIs in this folder.

Name	Date modified	Type	Size
annotations-api	2022/11/16 15:34	Executable Jar File	13 KB
catalina	2022/11/16 15:34	Executable Jar File	1 686 KB
catalina-ant	2022/11/16 15:34	Executable Jar File	53 KB
catalina-ha	2022/11/16 15:34	Executable Jar File	119 KB
catalina-storeconfig	2022/11/16 15:34	Executable Jar File	76 KB
catalina-tribes	2022/11/16 15:34	Executable Jar File	288 KB
ejc-4.6.3	2022/11/16 15:34	Executable Jar File	2 393 KB
el-api	2022/11/16 15:34	Executable Jar File	87 KB
jasper	2022/11/16 15:34	Executable Jar File	589 KB
jasper-el	2022/11/16 15:34	Executable Jar File	167 KB
jaspic-api	2022/11/16 15:34	Executable Jar File	27 KB
jsp-api	2022/11/16 15:34	Executable Jar File	61 KB
servlet-api	2022/11/16 15:34	Executable Jar File	244 KB
tomcat-api	2022/11/16 15:34	Executable Jar File	11 KB

The **logs** folder contains the files that logs the activities of the server. This is one folder we are going to use a lot. When exceptions take place in our server, we will check these files for detailed explanations as to say what happened.

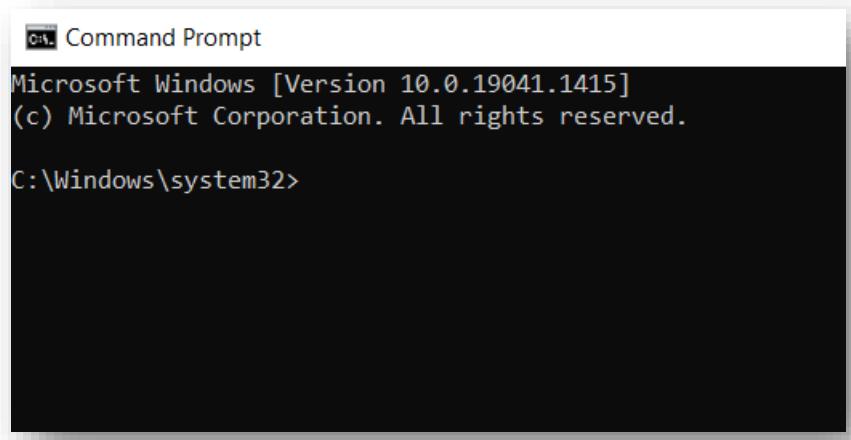
Name	Date modified	Type	Size
catalina.2022-12-12	2022/12/12 10:11	Text Document	7 KB
catalina.2022-12-20	2022/12/20 20:12	Text Document	1 KB
commons-daemon.2022-12-12	2022/12/20 20:12	Text Document	1 KB
host-manager.2022-12-12	2022/12/12 10:11	Text Document	0 KB
localhost.2022-12-12	2022/12/12 10:11	Text Document	1 KB
localhost.2022-12-20	2022/12/20 20:12	Text Document	1 KB
localhost_access_log.2022-12-12	2022/12/12 10:11	Text Document	0 KB
manager.2022-12-12	2022/12/12 10:11	Text Document	0 KB
service-install	2022/12/12 10:11	Text Document	2 KB
tomcat8-stderr.2022-12-12	2022/12/20 20:12	Text Document	8 KB
tomcat8-stdout.2022-12-12	2022/12/12 10:11	Text Document	1 KB

The **webapps** folder stores all the web applications deployed in the server. So if you want your application to be deployed in Tomcat, you must place it in this folder.

Name	Date modified	Type
docs	2022/12/12 10:11	File folder
examples	2022/12/12 10:11	File folder
host-manager	2022/12/12 10:11	File folder
manager	2022/12/12 10:11	File folder
ROOT	2022/12/12 10:11	File folder

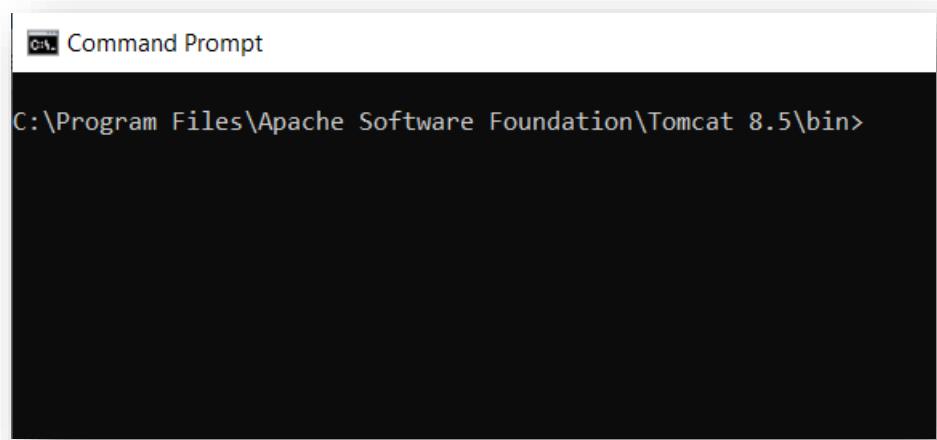
- Start Tomcat.

Open the command line.



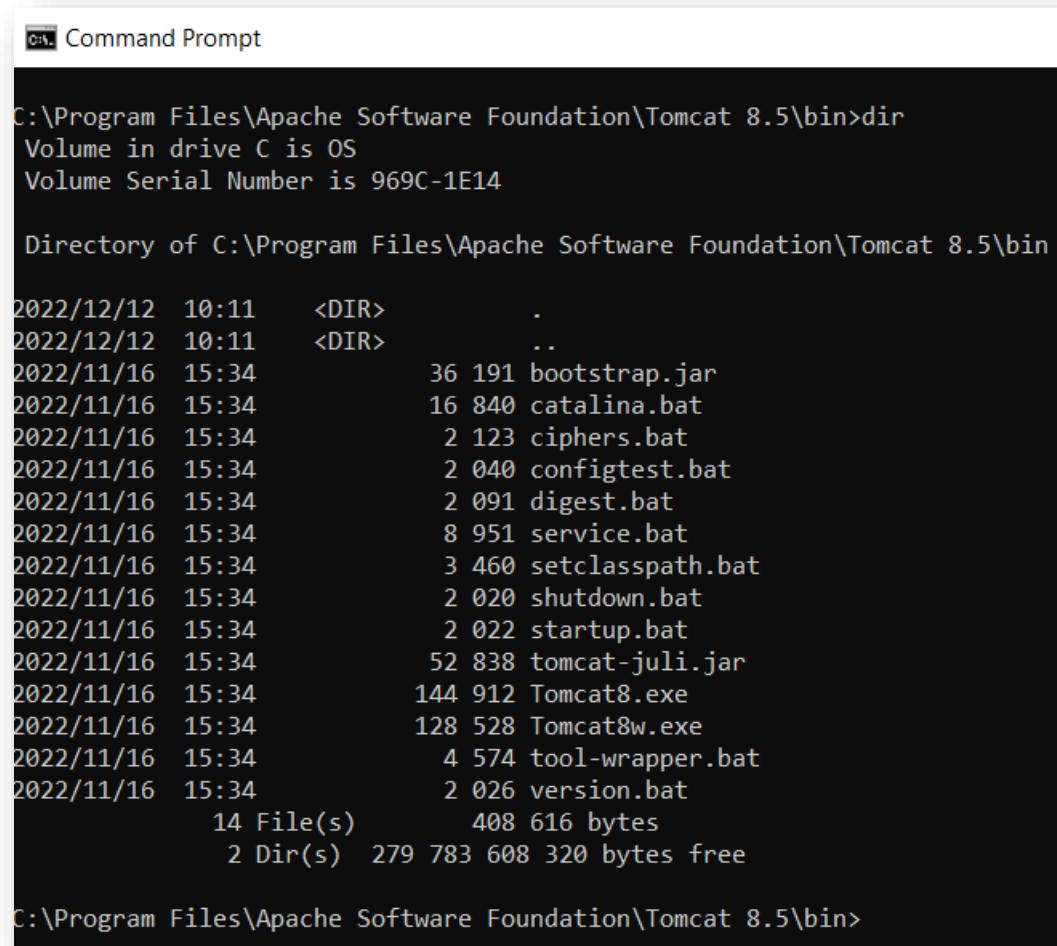
A screenshot of a Windows Command Prompt window titled "Command Prompt". The window shows the following text:
Microsoft Windows [Version 10.0.19041.1415]
(c) Microsoft Corporation. All rights reserved.
C:\Windows\system32>

Navigate to the **bin** directory of Tomcat.



A screenshot of a Windows Command Prompt window titled "Command Prompt". The window shows the following text:
C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>

Check the applications available to you.



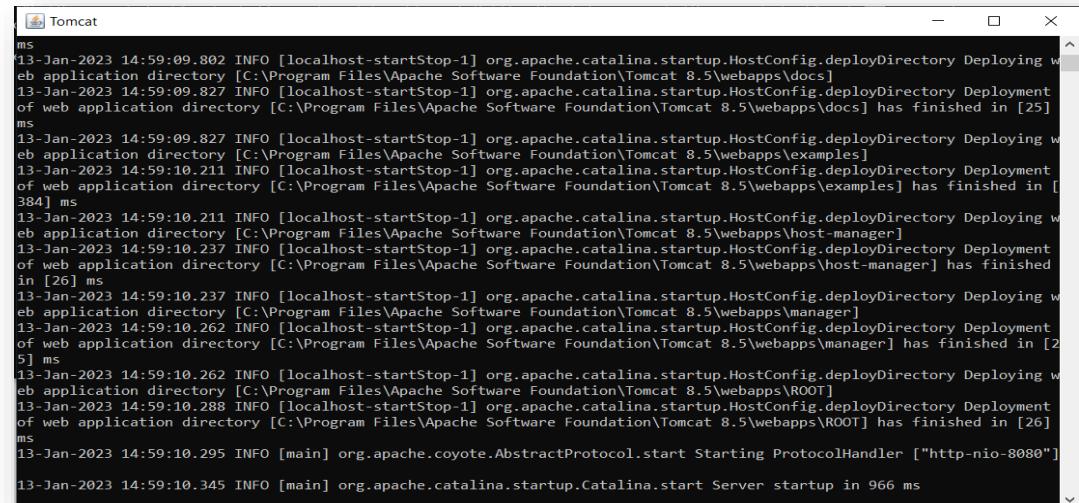
```
C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>dir
 Volume in drive C is OS
 Volume Serial Number is 969C-1E14

 Directory of C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin

2022/12/12  10:11    <DIR>      .
2022/12/12  10:11    <DIR>      ..
2022/11/16  15:34           36 191 bootstrap.jar
2022/11/16  15:34           16 840 catalina.bat
2022/11/16  15:34           2 123 ciphers.bat
2022/11/16  15:34           2 040 configtest.bat
2022/11/16  15:34           2 091 digest.bat
2022/11/16  15:34           8 951 service.bat
2022/11/16  15:34           3 460 setclasspath.bat
2022/11/16  15:34           2 020 shutdown.bat
2022/11/16  15:34           2 022 startup.bat
2022/11/16  15:34           52 838 tomcat-juli.jar
2022/11/16  15:34          144 912 Tomcat8.exe
2022/11/16  15:34          128 528 Tomcat8w.exe
2022/11/16  15:34           4 574 tool-wrapper.bat
2022/11/16  15:34           2 026 version.bat
                           14 File(s)     408 616 bytes
                           2 Dir(s)   279 783 608 320 bytes free

C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin>
```

To run Tomcat, type “**startup**” and press the Enter key.



```
Tomcat
ms
13-Jan-2023 14:59:09.802 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory [C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\docs]
13-Jan-2023 14:59:09.827 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application directory [C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\docs] has finished in [25] ms
13-Jan-2023 14:59:09.827 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory [C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\examples]
13-Jan-2023 14:59:10.211 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application directory [C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\examples] has finished in [384] ms
13-Jan-2023 14:59:10.211 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory [C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\host-manager]
13-Jan-2023 14:59:10.237 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application directory [C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\host-manager] has finished in [26] ms
13-Jan-2023 14:59:10.237 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory [C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\manager]
13-Jan-2023 14:59:10.262 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application directory [C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\manager] has finished in [25] ms
13-Jan-2023 14:59:10.262 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory [C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\ROOT]
13-Jan-2023 14:59:10.288 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application directory [C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\ROOT] has finished in [26] ms
13-Jan-2023 14:59:10.295 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["http-nio-8080"]
13-Jan-2023 14:59:10.345 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in 966 ms
```

- Stop Tomcat.

Type “**shutdown**” and press the Enter key to stop Tomcat.

13 Appendix D: How to download and install MySQL

14 References

14.1 Prescribed textbook



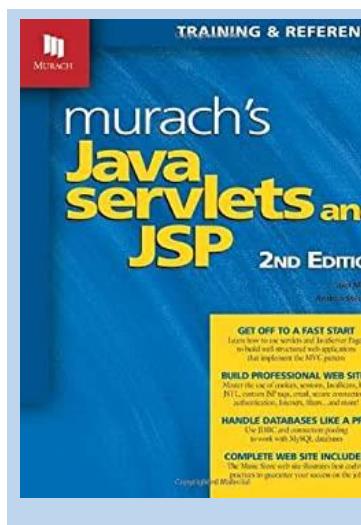
Head First Servlets & JSP – 2nd Edition

By Bryan Bashan, Kathy Sierra & Bert Bates.

Published by O'Reilly

ISBN-13: 978-0-596-51668-0

14.2 Recommended Additional Resources



Murach's Java Servlets and JSP

By Joey Murach

Published by Mike Murach & Associates, Inc.

ISBN: 1890774448