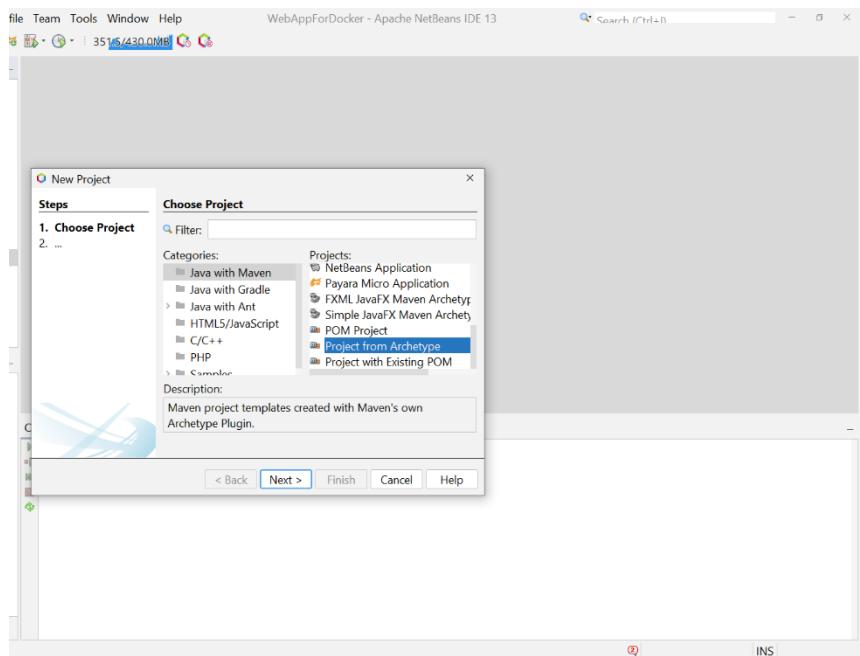
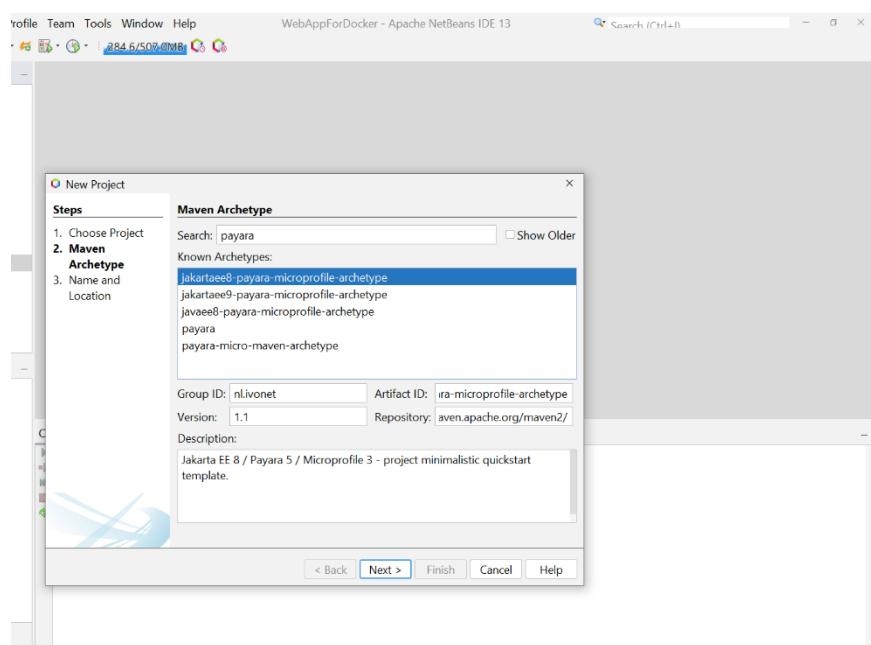


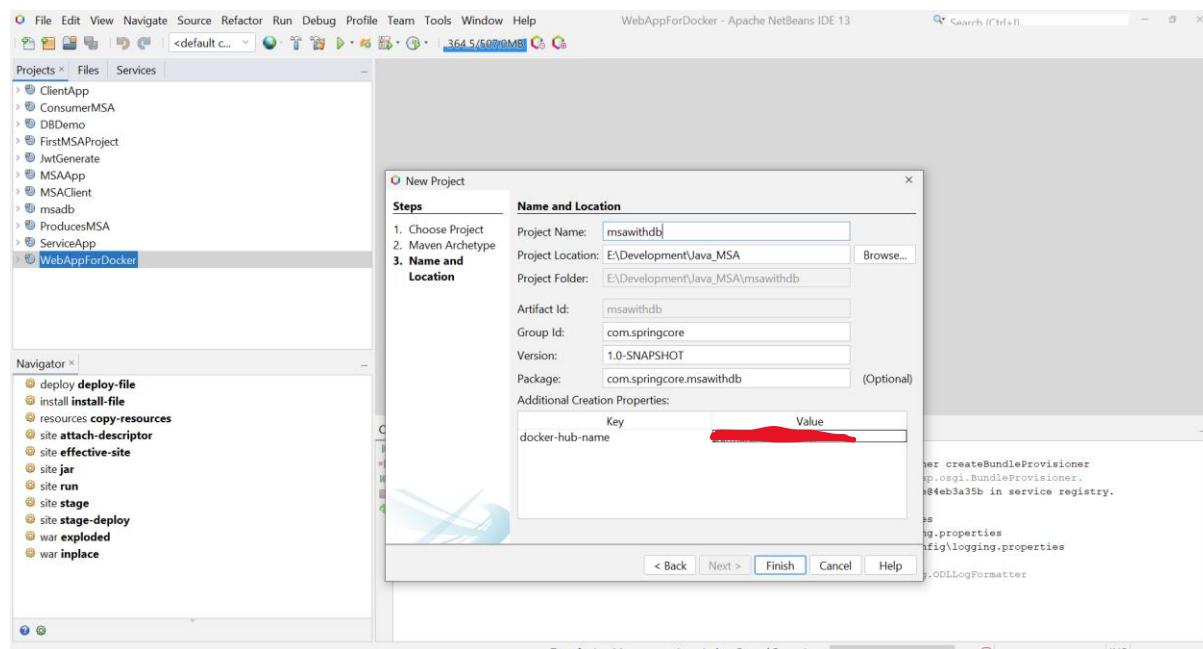
1. Create project



2. Search jakartaee8-payara-microprofile-archetype



3. Enter project name



4. I've created database named testdb and it contains table named students

The screenshot shows the MySQL Workbench interface. The main window displays the 'Structure' tab for a table named 'students' in the 'testdb' database. The table has four columns: 'id', 'name', 'email', and 'course'. The 'id' column is defined as INT(11) with a primary key constraint and AUTO_INCREMENT. The 'name', 'email', and 'course' columns are defined as VARCHAR(100) with the utf8mb4_general_ci collation. The 'Indexes' section shows a primary key index on the 'id' column. At the bottom, there are buttons for 'Print', 'Propose table structure', 'Track table', 'Move columns', 'Normalize', 'Add', and 'Indexes'.

5. Make some changes in pom.xml file

```

<project>
    <build>
        <plugins>
            <maven.compiler.plugin>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </maven.compiler.plugin>
        </plugins>
    </build>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
    </properties>
</project>

```

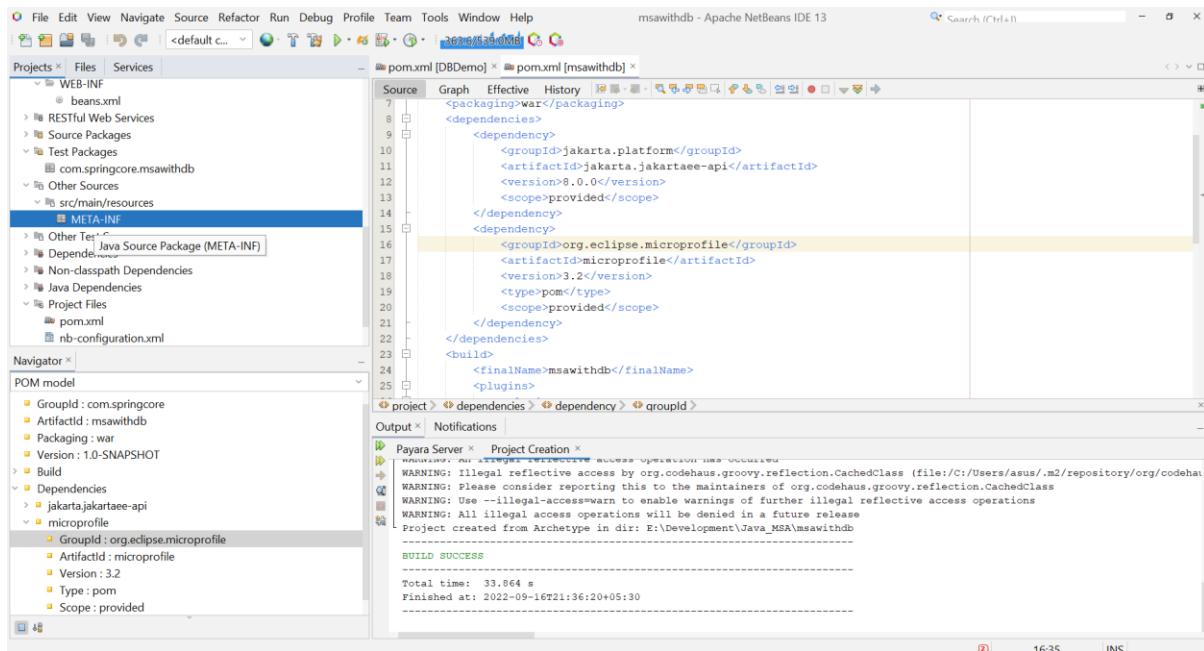
6. Change version from 3.2 to 3.1 (Note : If 3.1 gives an error then don't change it)

```

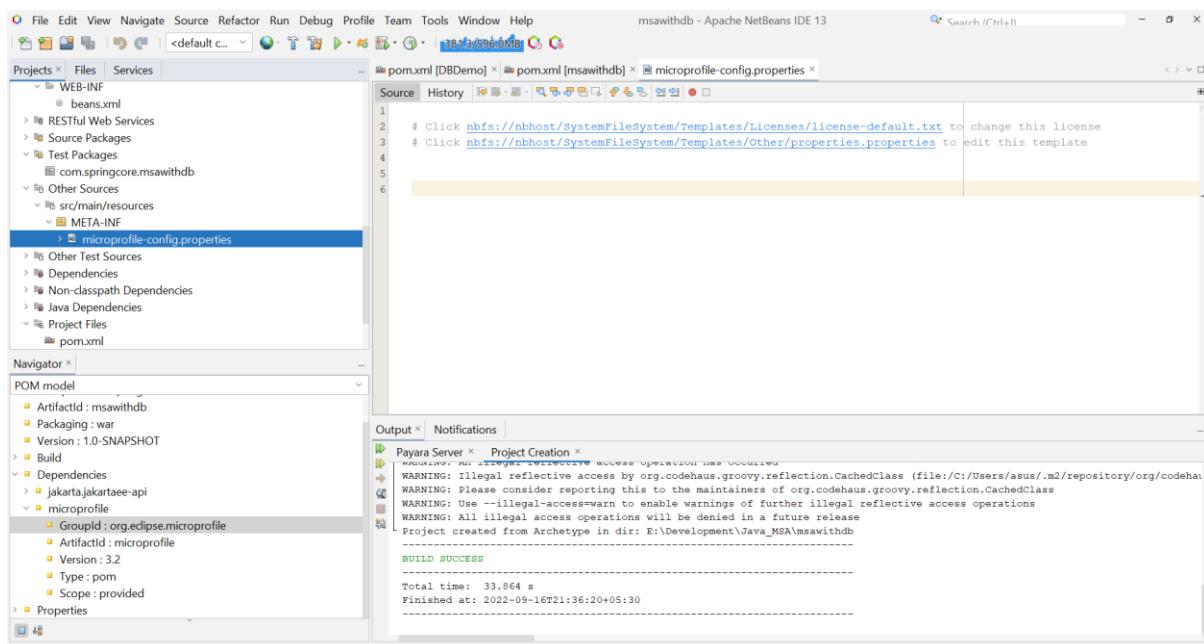
<dependencies>
    <dependency>
        <groupId>jakarta.platform</groupId>
        <artifactId>jakarta.jakartae-api</artifactId>
        <version>8.0.0</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>org.eclipse.microprofile</groupId>
        <artifactId>micropackaging</artifactId>
        <version>3.2</version>
        <type>pom</type>
        <scope>provided</scope>
    </dependency>
</dependencies>

```

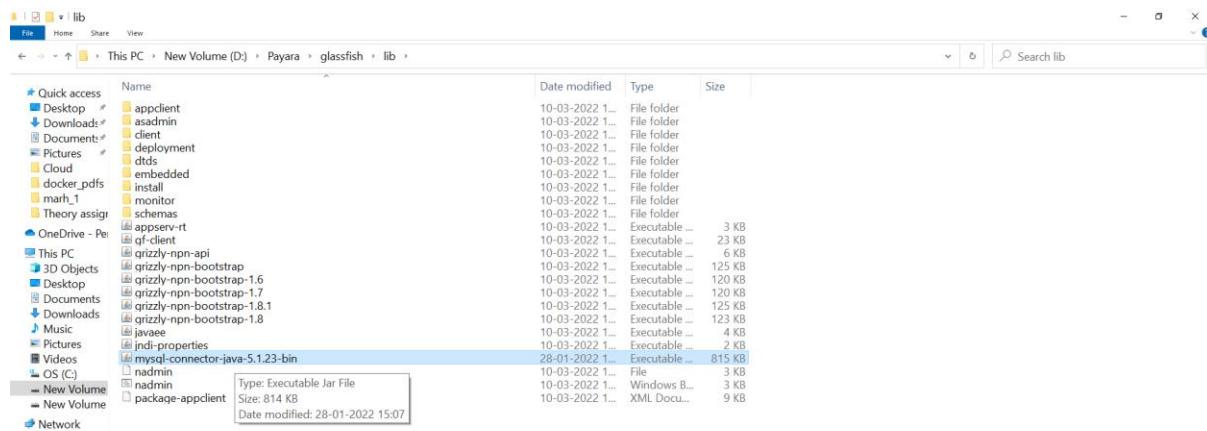
7. Create folder named META-INF in Other Sources -> src/main/resources



8. Add properties file named microprofile-config in that folder



9. Before Moving on I wanna to check you whether the mysql connector jar is existed in following location of your payara server.



10. Start the payara server and add URL localhost:4848 in your browser

11. Create JDBC connection pool

The screenshot shows the Payara Administration Console interface. The left sidebar navigation tree includes sections like jdbc/_TimerPool, jdbc/_default, jdbc/msadb, jdbc/mymssql, and jdbc/restdbpool under the JDBC Connection Pools category. The main content area is titled "JDBC Connection Pools" and contains a table titled "Pools (6)". The table columns are Select, Pool Name, Resource Type, Classname, and Description. The data rows are:

Select	Pool Name	Resource Type	Classname	Description
	ComputerProjectPool	javax.sql.DataSource	com.mysql.jdbc.jdbc2.optional.MysqlDataSource	
	H2Pool	javax.sql.DataSource	org.h2.jdbc.JdbcDataSource	
	_TimerPool	javax.sql.XADatasource	org.h2.jdbc.JdbcDataSource	
	msadbpool	javax.sql.DataSource	com.mysql.jdbc.jdbc2.optional.MysqlDataSource	
	mymssqlpool	javax.sql.DataSource	com.mysql.jdbc.jdbc2.optional.MysqlDataSource	
	restdbpool	javax.sql.DataSource	com.mysql.jdbc.jdbc2.optional.MysqlDataSource	

12. Add the details and click next

The screenshot shows the "New JDBC Connection Pool (Step 1 of 2)" configuration page. The left sidebar navigation tree is identical to the previous screenshot. The main form is titled "New JDBC Connection Pool (Step 1 of 2)" and contains the following fields:

- General Settings**
 - Pool Name:** * testdbpool
 - Resource Type:** javax.sql.DataSource
 - Database Driver Vendor:** MySQL
 - Introspect:** Enabled

Help text and validation messages are provided for each field, such as "Must be specified if the datasource class implements more than 1 of the interface." and "Select or enter a database driver vendor".

13. Add the following properties , there will be more properties in your pc, Leave rest of the properties as it is and click next

Property	Value	
URL	jdbc:mysql://localhost:3306/testdb?useSSL=false	
url	jdbc:mysql://localhost:3306/testdb?useSSL=false	
driverClass	com.mysql.jdbc.Driver	
driver	com.mysql.jdbc.Driver	
password	root	
databaseName		
roleName		
serverName	localhost	
datasourceName		
user	root	
networkProtocol		
portNumber	3306	

14. Add the driver in netbeans , If you already the add the driver file then skip this following step

New Connection Wizard

Locate Driver

Driver: MySQL (Connector/J driver)

Driver f MySQL (Connector/J driver)

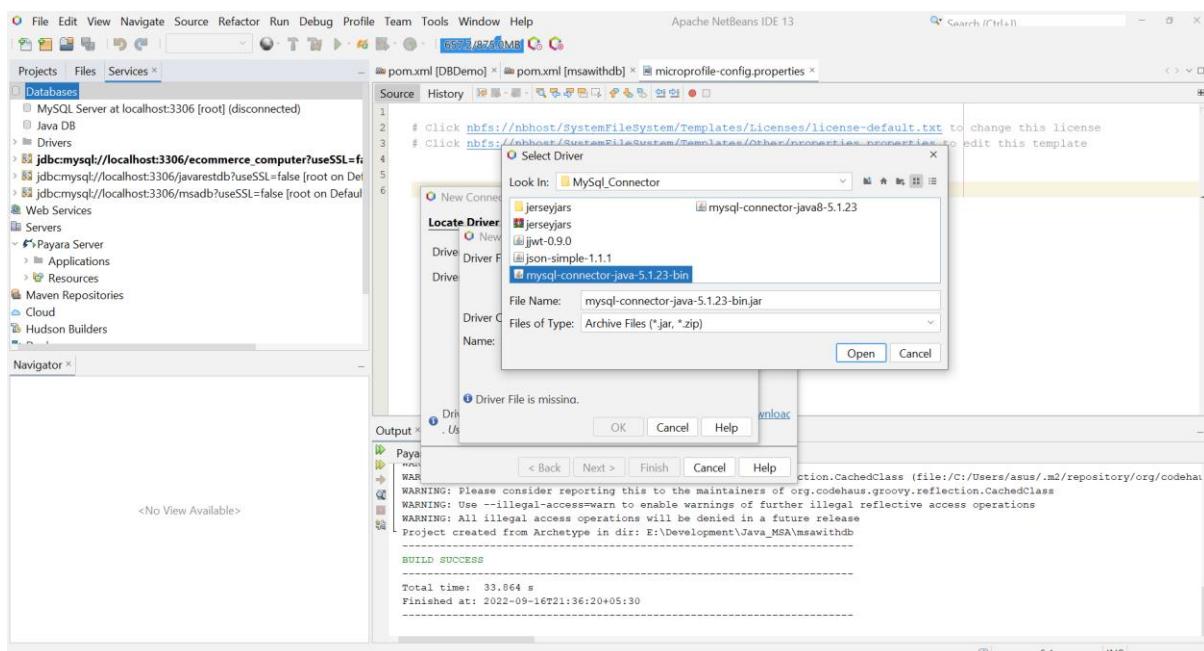
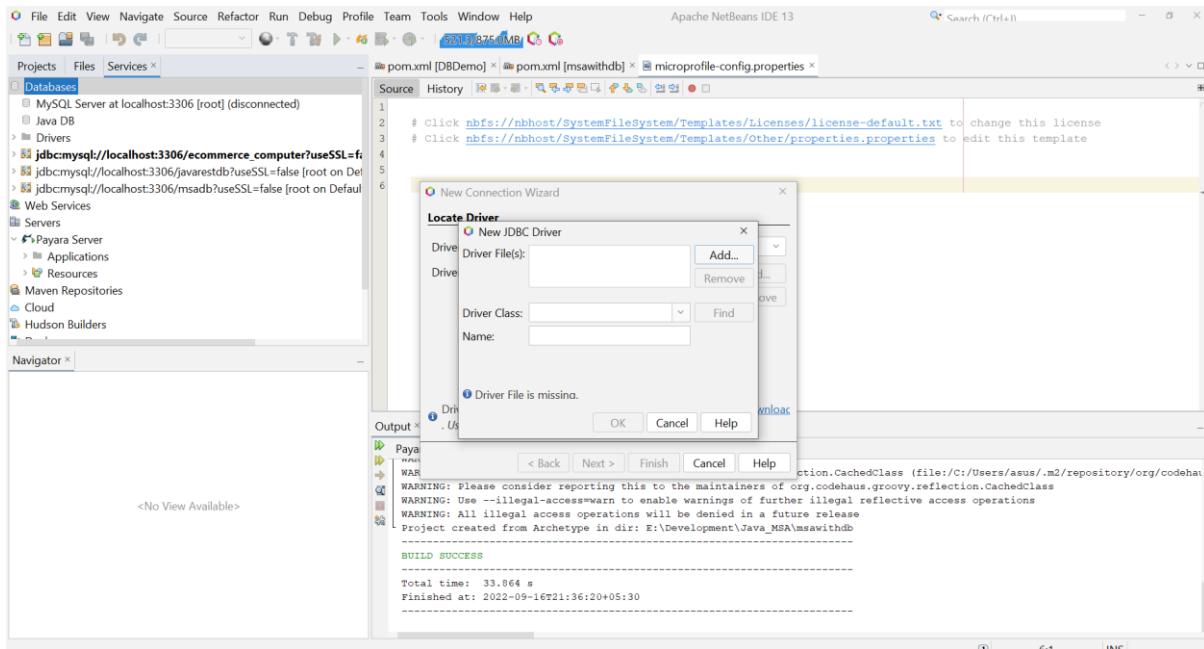
Oracle OCI

Oracle Thin

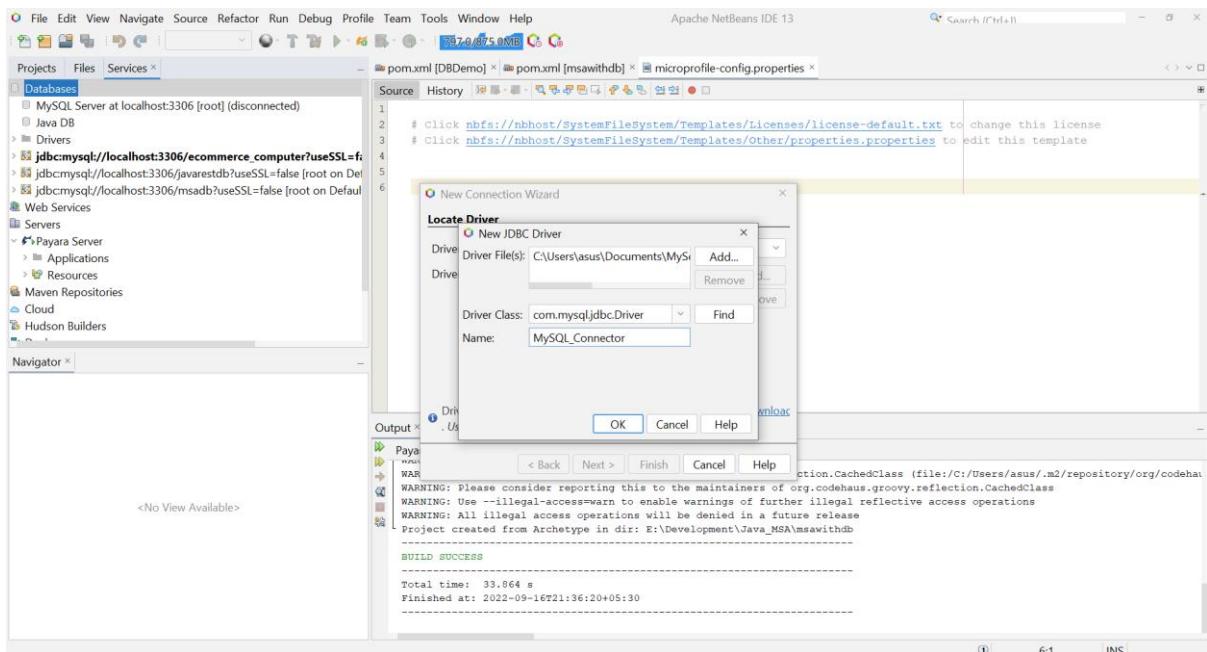
PostgreSQL

New Driver...

15. Click add Button and add the mysql-connector jar file.

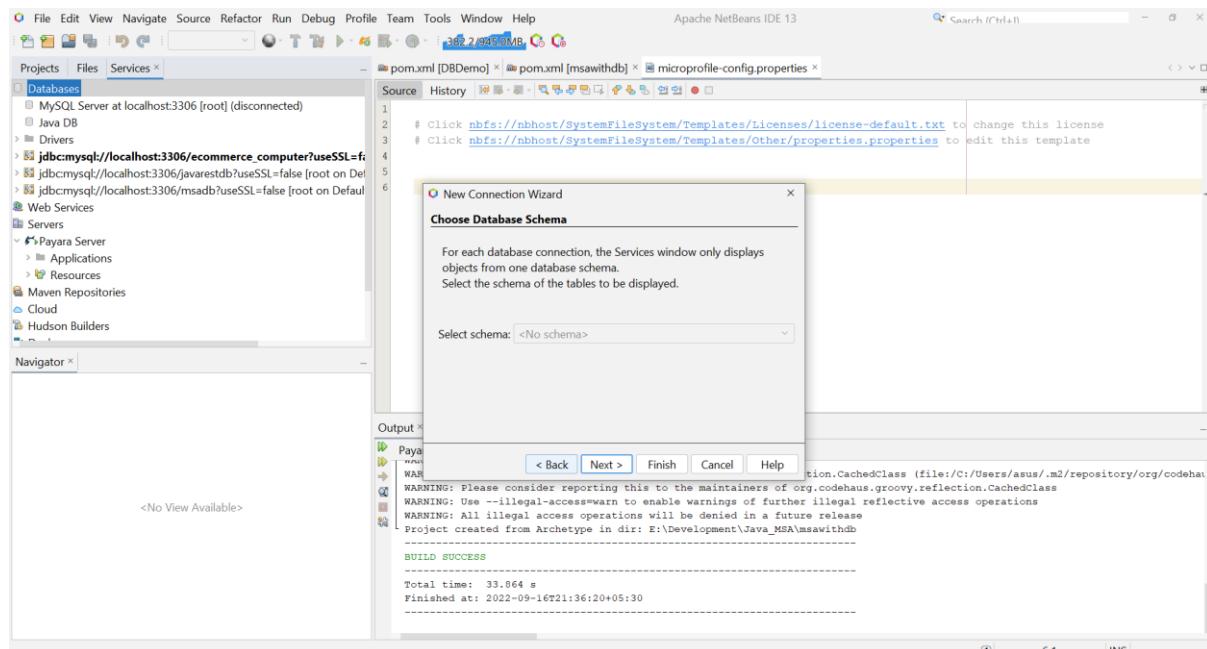


16. Add the driver class and name of the driver

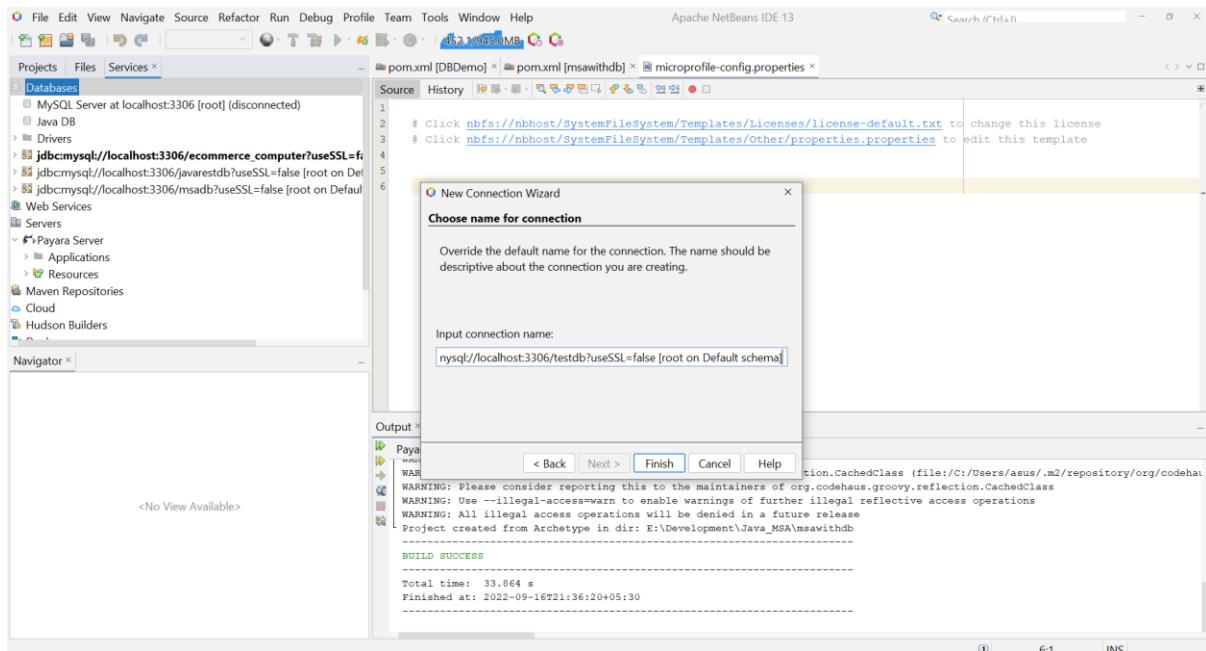


17. Enter use name and password of your mysql database and add JDBC URL related to your database name. In my case I am using testdb so my url would be jdbc:mysql://localhost:3306/testdb?useSSL=false , Click next

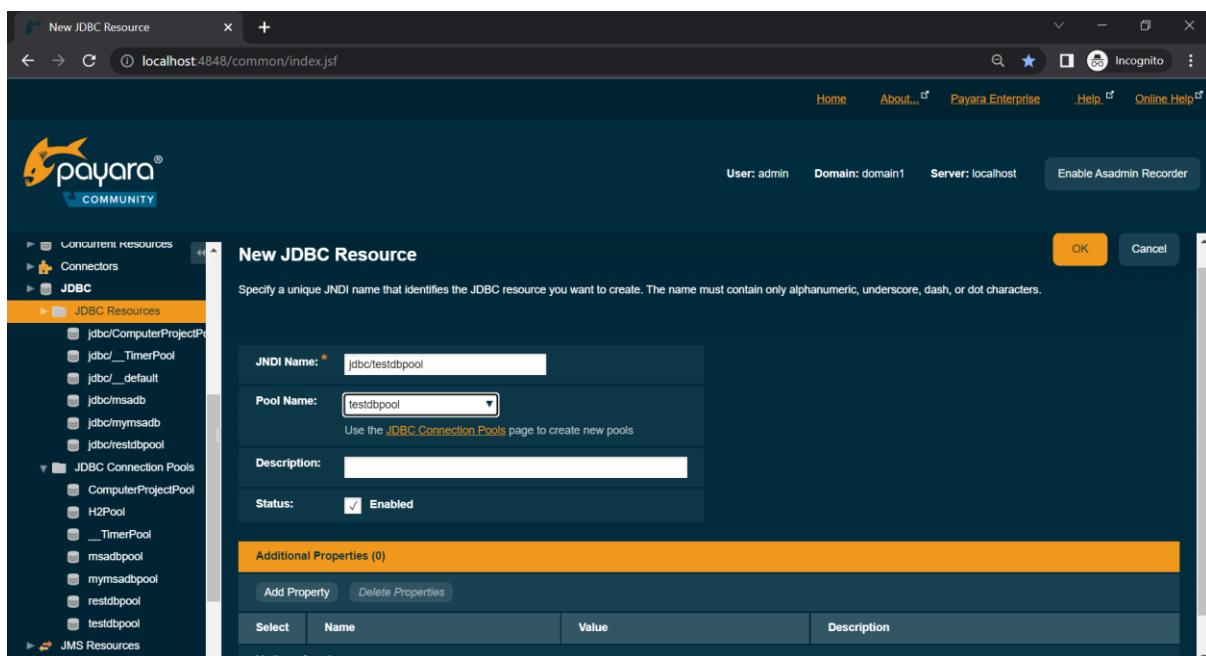
URL : jdbc:mysql://localhost:3306/**your_db_name**?useSSL=false



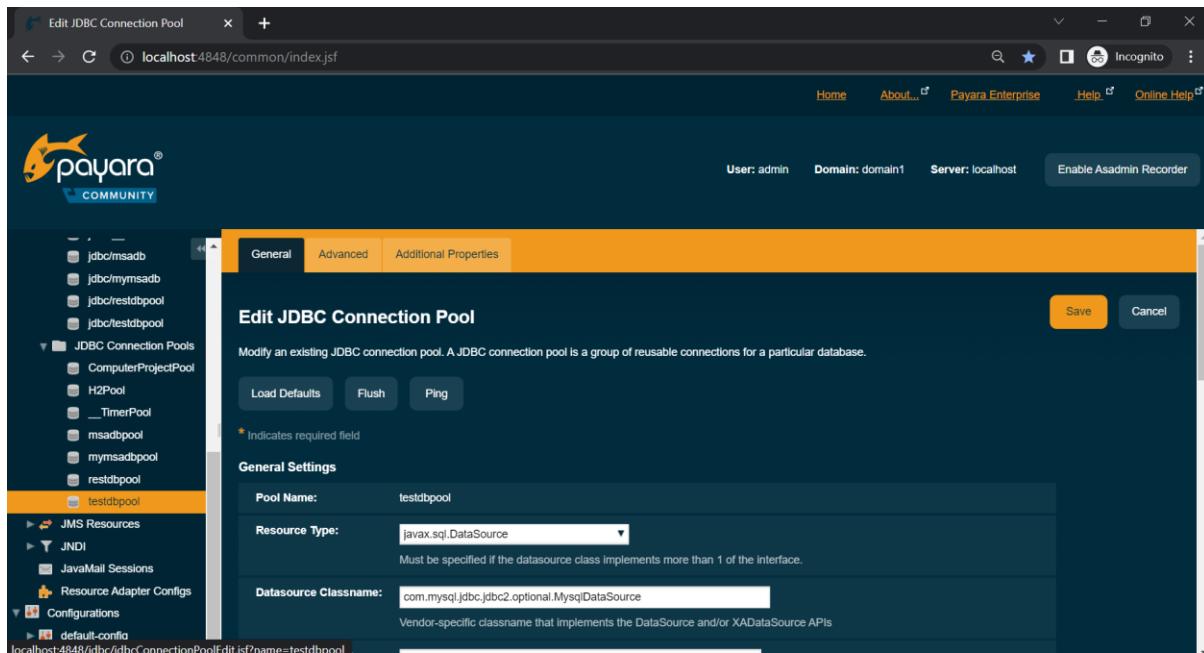
18. Leave as it is , click next.



19. Create JDBC Resources and link with your JDBC connection pool which you've created earlier one.
In my case, JNDI name I choose jdbc/testdbpool and select your jdbc connection pool.

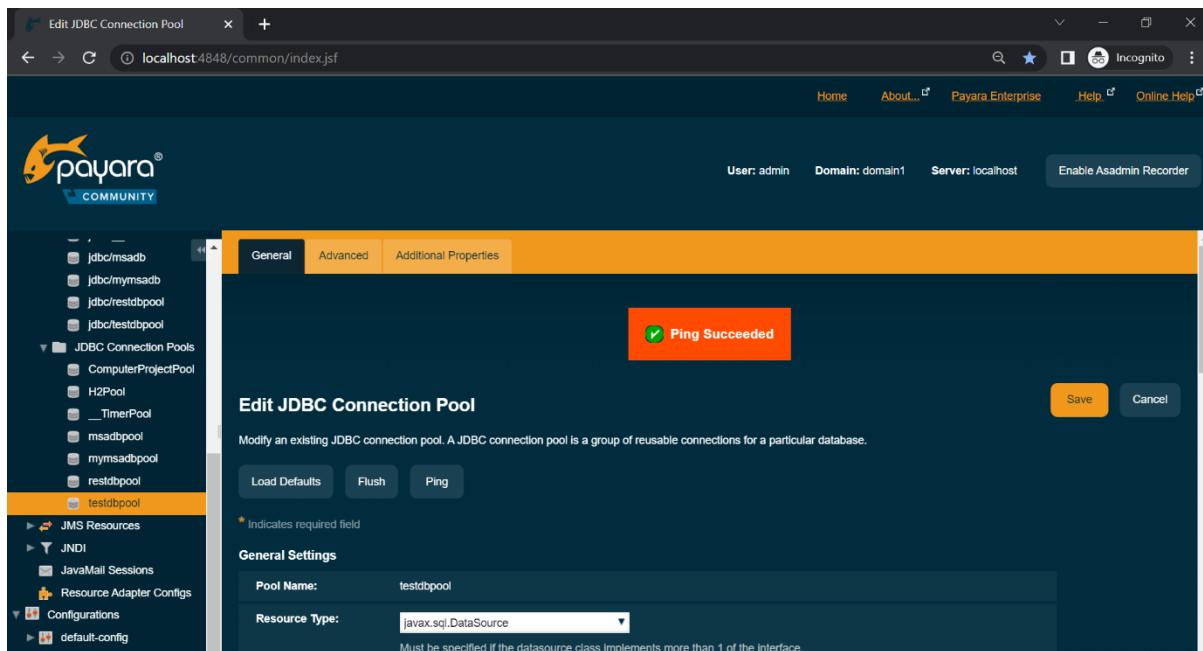


20. Go to your JDBC Connection pool that you've created earlier and click Ping



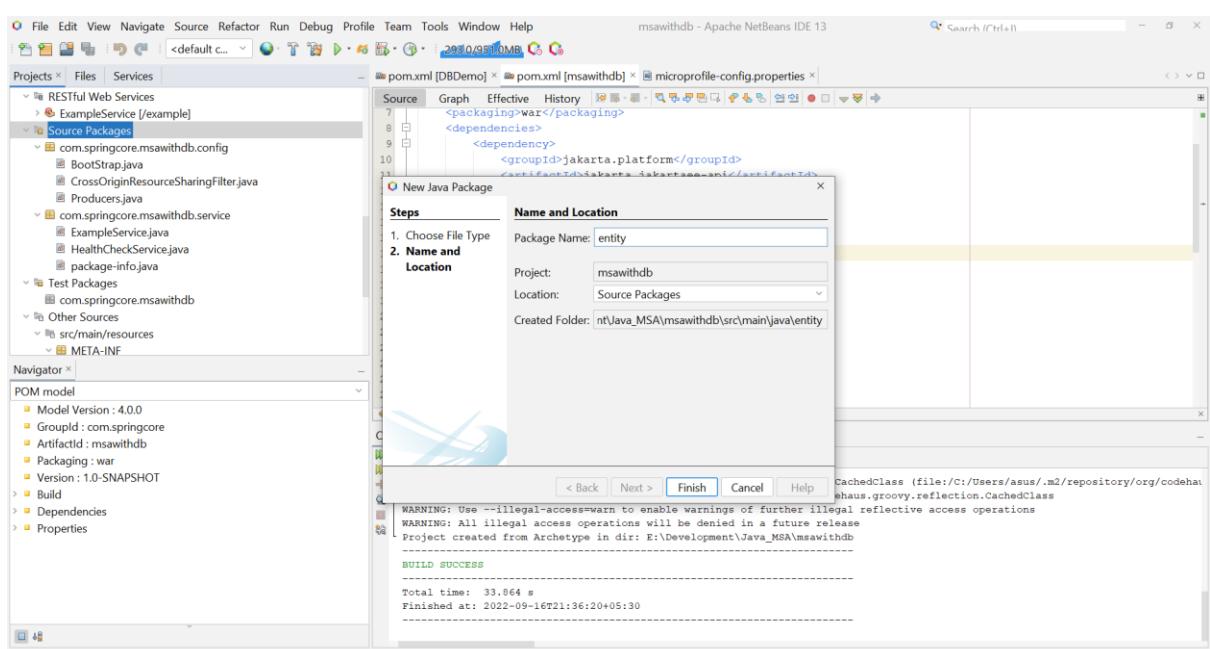
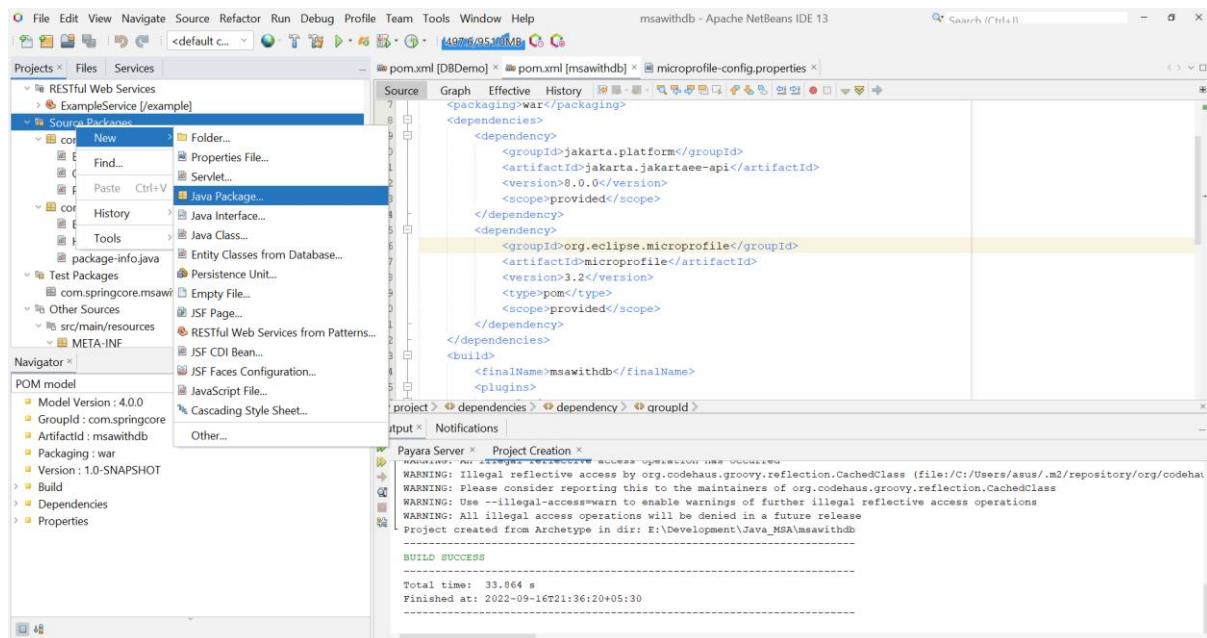
The screenshot shows the Payara Admin Console interface. On the left, there's a sidebar with various resource categories like JMS Resources, JNDI, JavaMail Sessions, and Resource Adapter Configs. The main panel is titled 'Edit JDBC Connection Pool' and shows settings for a pool named 'testdbpool'. It includes fields for 'Pool Name' (set to 'testdbpool'), 'Resource Type' (set to 'javax.sql.DataSource'), and 'Datasource Classname' (set to 'com.mysql.jdbc.optional.MysqlDataSource'). At the bottom of the panel, there are three buttons: 'Load Defaults', 'Flush', and 'Ping'. The 'Ping' button is highlighted with a yellow box. A note above the buttons says: 'Modify an existing JDBC connection pool. A JDBC connection pool is a group of reusable connections for a particular database.' Below the buttons, there's a note: 'Must be specified if the datasource class implements more than 1 of the interface.' and 'Vendor-specific classname that implements the DataSource and/or XADatasource APIs'.

21. Ping should be successful

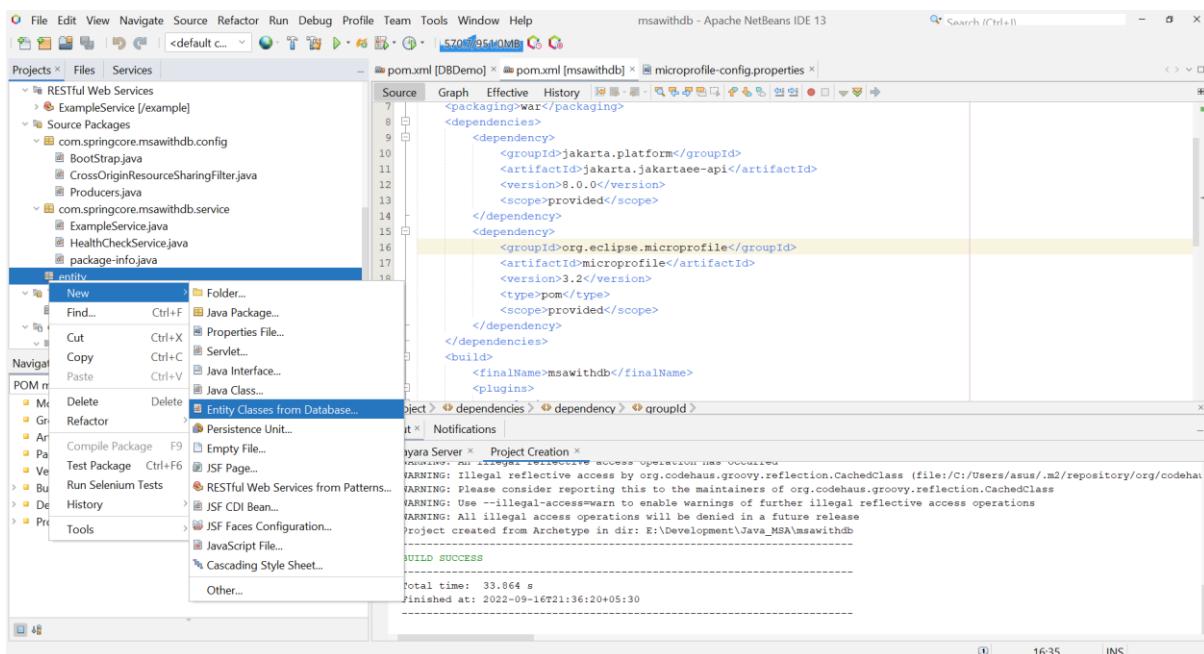


This screenshot is similar to the previous one but shows the result of the ping operation. The 'Ping' button has been clicked, and a large orange box at the top right of the dialog displays the message 'Ping Succeeded' with a green checkmark icon. The rest of the dialog and sidebar are identical to the previous screenshot.

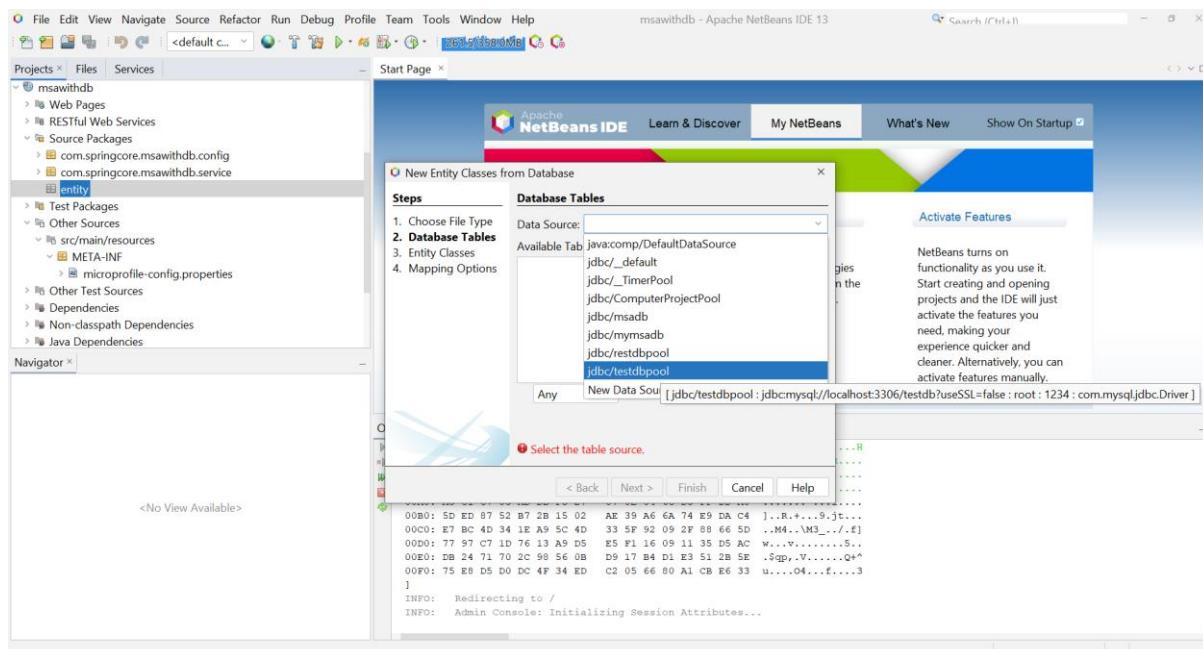
22. Create new package



23 . Right click on that package and Select entity classes from database



24. Select Data source based on Your jndi resource and select your table and click next till finish.



25. Create one java class file for data model. In my case, I've created StudentModel.java in model package and add the following code. For unitName of Persistence Context see the persistence.xml file and add the unitName

```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
msawithdb - Apache NetBeans IDE 13
Search (Ctrl+I)
Projects x Files Services
Source History <default c... persistence.xml
Source
import entity.Students;
import java.util.Collection;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

@Stateless
public class StudentModel {
    @PersistenceContext(unitName = "com.springcore_msawithdb_war_1.0-SNAPSHOT")
    EntityManager em;

    public Collection<Students> getStudents() {
        return em.createNamedQuery("Students.findAll").getResultList();
    }
}

model.StudentModel > em >
Output Notifications
Payara Server x Build (msawithdb) x
Building war: E:\Development\Java_MSA\msawithdb\artifact\msawithdb.war
--- maven-install-plugin:2.4:install (default-install) @ msawithdb ---
Installing artifact\msawithdb.war to C:\Users\asus\.m2\repository\com\springcore\msawithdb\1.0-SNAPSHOT\msawithdb-1.0-SNAPSHOT\msawithdb.war
Installing E:\Development\Java_MSA\msawithdb\pom.xml to C:\Users\asus\.m2\repository\com\springcore\msawithdb\1.0-SNAPSHOT\msawithdb\pom.xml
BUILD SUCCESS
Total time: 9.514 s
Finished at: 2022-09-17T08:59:02+05:30

```

26. Go to the ExampleService.java file and add the following code

```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
msawithdb - Apache NetBeans IDE 13
Search (Ctrl+I)
Projects x Files Services
Source History <default c... persistence.xml ExampleService.java
Source
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
import model.StudentModel;

@Path("/example")
public class ExampleService {

    @Inject StudentModel sm;

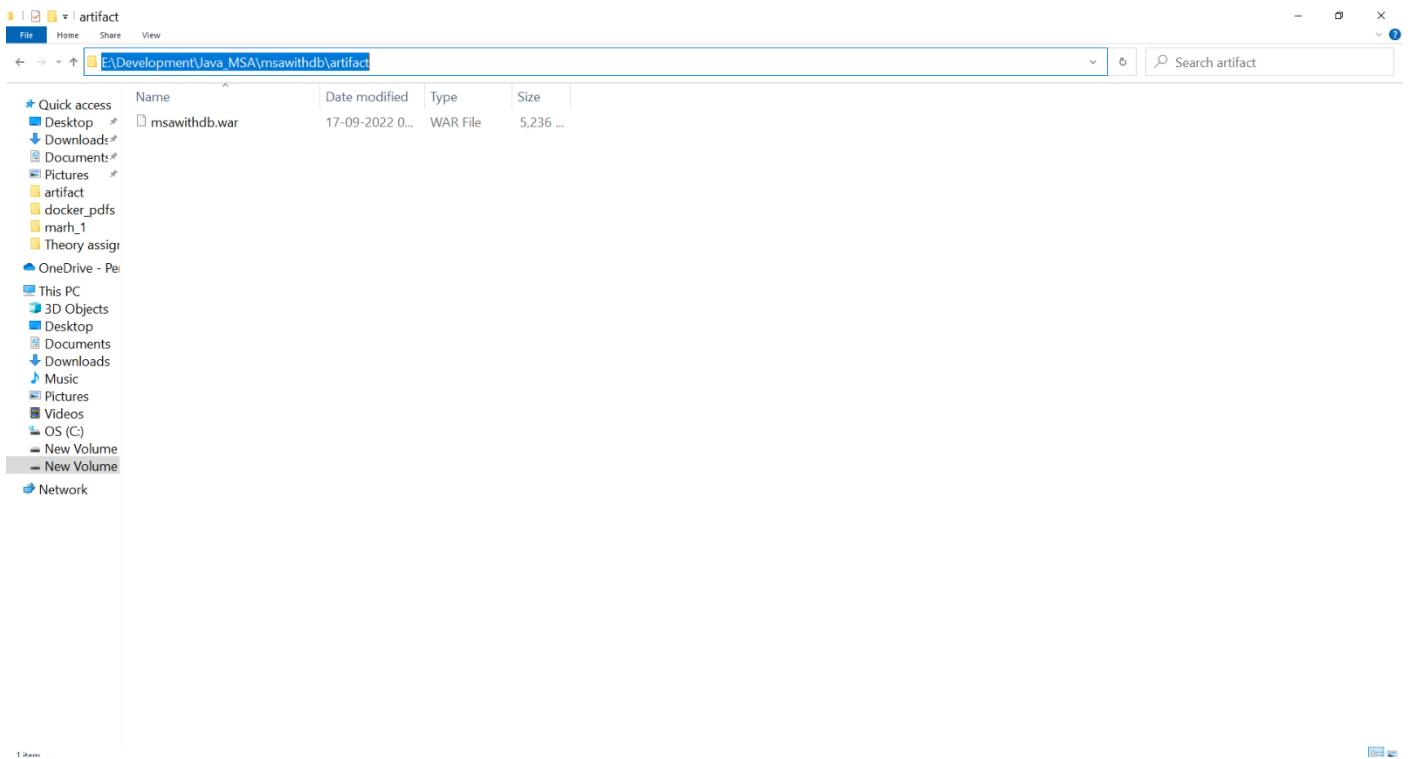
    @Path("students")
    @GET
    @Produces(MediaType.APPLICATION_JSON)
    public Collection<Students> getStudents() {
        return sm.getStudents();
    }
}

com.springcore.msawithdb.service.ExampleService > getStudents >
Output Notifications
Payara Server x Build (msawithdb) x
Building war: E:\Development\Java_MSA\msawithdb\artifact\msawithdb.war
--- maven-install-plugin:2.4:install (default-install) @ msawithdb ---
Installing artifact\msawithdb.war to C:\Users\asus\.m2\repository\com\springcore\msawithdb\1.0-SNAPSHOT\msawithdb-1.0-SNAPSHOT\msawithdb.war
Installing E:\Development\Java_MSA\msawithdb\pom.xml to C:\Users\asus\.m2\repository\com\springcore\msawithdb\1.0-SNAPSHOT\msawithdb\pom.xml
BUILD SUCCESS
Total time: 9.514 s
Finished at: 2022-09-17T08:59:02+05:30

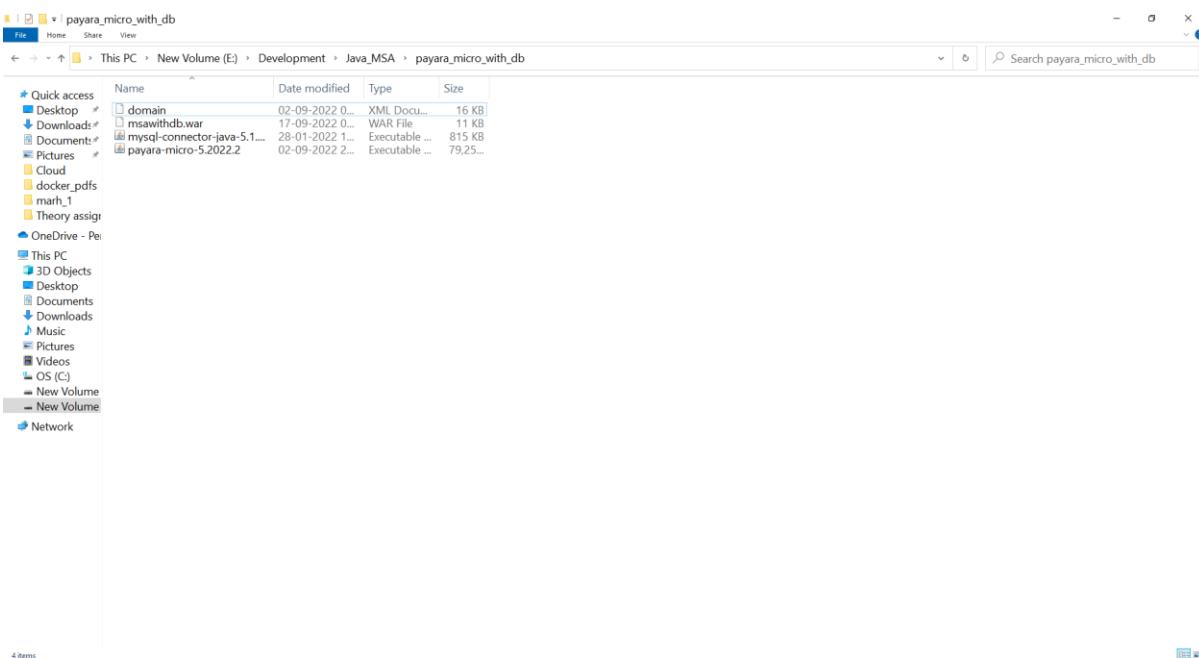
```

Okay till now, we've completed the and will run to project using api. Right click on your project click -> Clean and Build

27. Location your project folder and copy the war file and do paste on some blank folder



28. In that file Blank folder add domain.xml file, mysql-connector jar file and payara micro jar file



29. Open domain.xml file in your favourite text editor and make some changes following changes highlighted by red underline in domain.xml file

```

58         </jdbc-connection-pool>
59     <jdbc-connection-pool
60         datasource-classname="com.mysql.jdbc.optional.MysqlDataSource" name="testdbpool"
61         res-type="javax.sql.DataSource"
62         steady-pool-size="1"
63         is-connection-validation-required="true"
64         connection-validation-method="meta-data"
65         max-pool-size="10">
66             <property name="password" value="1234"/>
67             <property name="user" value="root"/>
68             <property name="databaseName" value="testdb"/>
69             <property name="serverName" value="localhost"/>
70             <property name="portNumber" value="3306"/>
71             <property name="useSSL" value="false"/>
72             <property name="zeroDateTimeBehavior" value="convertToNull"/>
73     </jdbc-connection-pool>
74     <jdbc-resource pool-name="testdbpool" jndi-name="jdbc/testdbpool">
75         <context-service description="context-service" jndi-name="concurrent/_defaultContextService" object-type="system-all"></context
76             <managed-executor-service max-threads="200" core-pool-size="0" long-running-tasks="true" keep-alive-seconds="300" hung-after-seconds="3
77                 <managed-scheduled-executor-service core-pool-size="0" long-running-tasks="true" keep-alive-seconds="300" hung-after-seconds="3
78                     <managed-thread-factory description="thread factory" jndi-name="concurrent/_defaultManagedThreadFactory" object-type="system-s
79             </resources>
80         <servers>
81             <server name="server" config-ref="server-config">
82                 <resource-ref ref="jdbc/_default" />
83                 <resource-ref ref="jdbc/testdbpool" />
84                 <resource-ref ref="concurrent/_defaultContextService" />
85                 <resource-ref ref="concurrent/_defaultManagedExecutorService" />
86                 <resource-ref ref="concurrent/_defaultManagedScheduledExecutorService" />
87                 <resource-ref ref="concurrent/_defaultManagedThreadFactory" />
88             </server>
89         </servers>
90     <configs>
91         <config name="server-config">
92             <payara-executor-service-configuration/>
93                 <cdi-service enable-concurrent-deployment="true" pre-loader-thread-pool-size="2" />
94                 <hazelcast-config-specific-configuration enabled="true" />
95                 <health-check-service-configuration enabled="false" />
96                     <log-notifier enabled="true" />

```

30. Close the text editor, Open the command prompt and enter the following command

```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.1889]
(c) Microsoft Corporation. All rights reserved.

E:\Development\Java_MSA\payara_micro_with_db>java -jar payara-micro-5.2022.2.jar --deploy msawithdb.war --port 8085 --addlibs mysql-connector-java-5.1.23-bin.jar --domainconfig domain.xml

```

31. Wait and watch you will get the following result.

```
C:\Windows\System32\cmd.exe - java -jar payara-micro-5.2022.2.jar --deploy msawithdb.war --port 8085 --addlibs mysql-connector-java-5.1.23-bin.jar --domainconfig domain.xml
[2022-09-17T01:09:41.204+0530] [] [+1;92mINFO*[0m] [AS-WEB-GLUE-00130] [-1;94mjavax.enterprise.web*[0m] [tid: _ThreadID=1 _ThreadName=main] [timeMillis: 1663357181204] [levelValue: 800] In
distributable app [msawithdb] - defaulting to memory: persistence-type = [hazelcast] / persistenceFrequency = [web-method] / persistenceScope = [modified-session]
[2022-09-17T01:09:41.310+0530] [] [+1;92mINFO*[0m] [AS-WEB-GLUE-00130] [-1;94mjavax.enterprise.web*[0m] [tid: _ThreadID=1 _ThreadName=main] [timeMillis: 1663357181310] [le
ara-p5 for context '/msawithdb'
[2022-09-17T01:09:41.852+0530] [] [+1;92mINFO*[0m] [AS-WEB-GLUE-00172] [-1;94mjavax.enterprise.web*[0m] [tid: _ThreadID=1 _ThreadName=main] [timeMillis: 1663357181852] [levelValue: 800] Lo
[2022-09-17T01:09:41.885+0530] [] [+1;92mINFO*[0m] [] [+1;94mPayaraMicro*[0m] [tid: _ThreadID=1 _ThreadName=main] [timeMillis: 1663357181885] [levelValue: 800]
{
  "Instance Configuration": [
    {
      "Host": "LAPTOP-UQ74F4H6",
      "Http Port(s)": "8085",
      "Https Port(s)": "",
      "Instance Name": "Eager-Daggetooth",
      "Instance Group": "Microshoal",
      "Hazelcast Member UUID": "23a84bfa-a989-4c28-bda9-03edfffc178d7",
      "Deployed": [
        {
          "Name": "msawithdb",
          "Type": "war",
          "Context Root": "/msawithdb"
        }
      ]
    }
  ]
}
[2022-09-17T01:09:41.896+0530] [] [+1;92mINFO*[0m] [] [+1;94mPayaraMicro*[0m] [tid: _ThreadID=1 _ThreadName=main] [timeMillis: 1663357181896] [levelValue: 800]
Payara Micro URLs:
http://LAPTOP-UQ74F4H6:8085/msawithdb

'msawithdb' REST Endpoints:
GET /msawithdb/rest/application.wadl
GET /msawithdb/rest/example
GET /msawithdb/rest/example/students

[2022-09-17T01:09:41.896+0530] [] [+1;92mINFO*[0m] [] [+1;94mPayaraMicro*[0m] [tid: _ThreadID=1 _ThreadName=main] [timeMillis: 1663357181896] [levelValue: 800] Payara Micro 5.2022.2 #bada
```

32. Enter the URL localhost:port_number/projectName/rest/example/students

In my case URL: localhost:8080/msawithdb/rest/example/students

