1 . check JDBC driver Feilds

password->

url ->jdbc:mysql://localhost:3306/ejdatabase?useSSL=false

URL ->jdbc:mysql://localhost:3306/ejdatabase?useSSL=false

serverName ->localhost

driver ->com.mysql.jdbc.Driver

user ->root

driverClass ->com.mysql.jdbc.Driver

port ->3306

2 . Start payara server , connect database by servers->drivers->rightclick->add driver->add->select connector file 5.0.8-bin->add->select database and giver url->test connection->ok

Payara micro with two application showing data and then add jwt

You Need 4 files Initially

* payara-micro-5.2022.2.jar
* mysql-connector-java-5.1.47-bin.jar
* jwtenizr.jar
* domain.xml

3 . do necessory changes in domain.xml(their will be 5 changes 2 of pool and 2 of jdbc/pool and one will be database name and also check password)

1>

Create New Project (MSAApp) -> Java with maven -> project from Archetype

Select : jakartaee8-payara-microprofile-archetype

Dockerhub:value ok

Do changes in pom.xml

<dependency>

<groupId>org.eclipse.microprofile</groupId>

<artifactId>microprofile</artifactId>

<version>3.2</version>

<type>pom</type>

<scope>provided</scope>

</dependency>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

2>

* Create persistence unit

(in tager error go to properties->run->select payara server)

* Create Entity Classes from database. (Model)
* Create JAVA class named DataModel for business logic with entityManager. (Beans)

public class DataModel {

@PersistenceContext(unitName = "punit")

EntityManager em;

Collection<Stocks> stock;

public Collection<Stocks> Stocks() {

return em.createNamedQuery("Stocks.findAll").getResultList();

}

public Collection<Stocks> StockByCat(String category) {

stock = (Collection<Stocks>) em.createNamedQuery("Stocks.findByCategory").setParameter("category", category).getResultList();

return stock;

}

}

3).

* Make Changes in Example Service means Add your rest resources in that file

import beans.DataModel;

import java.util.Collection;

import javax.inject.Inject;

import javax.ws.rs.GET;

import javax.ws.rs.Path;

import javax.ws.rs.PathParam;

import javax.ws.rs.Produces;

import javax.ws.rs.core.MediaType;

import model.Stocks;

@Path("/example")

public class ExampleService {

@Inject

DataModel dm;

@Path("/stocks")

// @RolesAllowed("admin")

// @RolesAllowed({"admin","user"})

@GET

// @Produces(MediaType.APPLICATION\_JSON)

public Collection<Stocks> Stocks() {

return dm.Stocks();

}

@Path("/StockByCat/{category}")

@GET

@Produces(MediaType.APPLICATION\_JSON)

public Collection<Stocks> StockByCat(@PathParam("category") String category) {

return dm.StockByCat(category);

}

}

4).

Create another app client like u create uper app

Create New Project (MSAClirnt) -> Java with maven -> project from Archetype

Select : jakartaee8-payara-microprofile-archetype

Dockerhub:value ok

Do changes in pom.xml

<dependency>

<groupId>org.eclipse.microprofile</groupId>

<artifactId>microprofile</artifactId>

<version>3.2</version>

<type>pom</type>

<scope>provided</scope>

</dependency>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

5) Paste whole folder of entity in this app to have all table entities

6).Create MSAClient INTERFACE to call the other app api s

import java.util.Collection;

import javax.enterprise.context.ApplicationScoped;

import javax.ws.rs.GET;

import javax.ws.rs.Path;

import javax.ws.rs.Produces;

import javax.ws.rs.core.MediaType;

import model.Stocks;

import org.eclipse.microprofile.rest.client.inject.RegisterRestClient;

/\*\*

\*

\* @author radhika

\*/

//@RegisterRestClient(configKey = "myclient")

@RegisterRestClient(baseUri = "<http://localhost:8085/stockApp/rest/example>")

@ApplicationScoped

//@Named

public interface myclient {

// @ClientHeaderParam(name="authorization",value="{generateJWTToken}")

@GET

@Path("/stocks")

// @RolesAllowed("admin")

@Produces(MediaType.APPLICATION\_JSON)

public Collection<Stocks> Stocks();

// default String generateJWTToken()

// {

// Config config = ConfigProvider.getConfig();

// String jwt = config.getValue("jwt-string", String.class);

// String authtoken = "Bearer "+jwt;

// return authtoken;

// }

}

7). Add Folder “META-INF” at location below:

Other Sources -> src/main/resources -> META-INF

Inside META-INF, create one properties file named “microprofile-config.properties”

.add MSAApp url into MSACLient App in microprofile-config.properties on top

myclient/mp-rest/url=http://localhost:8085/demo/rest

* Create servlet(do some changes like)

public class TestClientServlet extends HttpServlet {

@Inject @RestClient MSAClient msacl;

Collection<Students> docApps;

* Add this inside body tag

try{

docApps=msacl.students();

out.println("<h1>Available Doctor's</h1><br/><br/><br/>");

out.println("<table><tr><td>DocName</td><td>Specialist</td><td>Address</td><td>FromTime</td><td>ToTime</td></tr>");

for(Students dapp:docApps){

out.println("<tr><td>"+dapp.getName()+"</td><td>"+dapp.getEmail()+"</td><td>"+dapp.getCourse()+"</td></tr>");

}

out.println("<table>");

}catch(Exception e){

out.println("Error Error Error");

}

**========================For Jsf================================**

**create CDI Jsf bean in controller folder**

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/JSF/JSFManagedBean.java to edit this template

\*/

package Controller;

import Client.MSAClient;

import Entity.Stocks;

import java.util.ArrayList;

import java.util.Collection;

import java.util.Date;

import java.util.List;

import javax.annotation.PostConstruct;

import javax.inject.Named;

import javax.enterprise.context.ApplicationScoped;

import javax.inject.Inject;

/\*\*

\*

\* @author radhika

\*/

@Named(value = "stockManagedBean")

@ApplicationScoped

public class stockManagedBean {

@Inject

MSAClient msacl;

Collection<Stocks> stockApps;

private String searchVal;

List<String> specArr;

@PostConstruct

public void init() {

specArr = new ArrayList<>();

specArr.add("banking");

specArr.add("metal");

specArr.add("oil");

specArr.add("infra");

specArr.add("it");

searchVal = "banking";

}

public Collection<Stocks> getStockApps() {

return stockApps;

}

public void setStockApps(Collection<Stocks> stockApps) {

this.stockApps = stockApps;

}

public List<String> getSpecArr() {

return specArr;

}

public void setSpecArr(List<String> specArr) {

this.specArr = specArr;

}

public MSAClient getMsacl() {

return msacl;

}

/\*\*

\* Creates a new instance of stockManagedBean

\*/

public void setMsacl(MSAClient msacl) {

this.msacl = msacl;

}

public String getSearchVal() {

return searchVal;

}

public void setSearchVal(String searchVal) {

this.searchVal = searchVal;

}

public stockManagedBean() {

}

// public List<Stocks> findAll() {

// return (List<Stocks>) msacl.stocks();

//

// }

public List<Stocks> StockByCat() {

return (List<Stocks>) msacl.StockByCat(this.searchVal);

}

}

**=======================display.xhtml============================**

<?xml version='1.0' encoding='UTF-8' ?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "<http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd>">

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://xmlns.jcp.org/jsf/html"

xmlns:f="http://xmlns.jcp.org/jsf/core">

<h:head>

<title>Facelet Title</title>

</h:head>

<h:body>

Hello from Facelets<f:view>

<h:form>

<h:selectOneMenu id="searchVal" value="#{stockManagedBean.searchVal}">

<f:selectItems var="d" value="#{stockManagedBean.specArr}" itemValue="#{d}" itemLabel="#{d}">

</f:selectItems>

</h:selectOneMenu>

<br/><br/>

<h:commandButton id="b1" value="Search" action="#{stockManagedBean.StockByCat()}">

<!-- <f:ajax event="click" execute="deptName" render="tb1" />-->

</h:commandButton>

<h1><h:outputText value="List"/></h1>

<h:dataTable value="#{stockManagedBean.StockByCat()}" var="item">

<h:column>

<f:facet name="header">

<h:outputText value="SId"/>

</f:facet>

<h:outputText value="#{item.SId}"/>

</h:column>

<h:column>

<f:facet name="header">

<h:outputText value="Company"/>

</f:facet>

<h:outputText value="#{item.company}"/>

</h:column>

<h:column>

<f:facet name="header">

<h:outputText value="SDate"/>

</f:facet>

<h:outputText value="#{item.SDate}">

<f:convertDateTime pattern="MM/dd/yyyy" />

</h:outputText>

</h:column>

<h:column>

<f:facet name="header">

<h:outputText value="CPrice"/>

</f:facet>

<h:outputText value="#{item.CPrice}"/>

</h:column>

<h:column>

<f:facet name="header">

<h:outputText value="Category"/>

</f:facet>

<h:outputText value="#{item.category}"/>

</h:column>

<h:column>

<f:facet name="header">

<h:outputText value="SensexCvalue"/>

</f:facet>

<h:outputText value="#{item.sensexCvalue}"/>

</h:column>

</h:dataTable>

</h:form>

</f:view>

</h:body>

</html>

—-----------------------------------------------------------------------------------ADD JWT

Run jwtenizr.jar by typing command java -jar jwtenizr.jar in CMD.

Change jwt-token.json and update by running again

(this will create 4 different files)

* Add 2 lines in bootstrap file of MSAApp

@LoginConfig(authMethod="MP-JWT")

@DeclareRoles({"admin","user"})

* IN example service assign role like

@Inject DataModel dm;

@GET

@Path("student")

@RolesAllowed("admin")

public Collection<Students> getStudent()

{

return dm.getStudents();

}

* In other sources/src/META-INF/microprofile-config add all fields of newly generated microprofile-config file(mp.jwt.verify.issuar and mp.jwt.verify.publickey)

2)CLIENT APP

Create JAVA INTERFACE inside client app named MSAClient inside that assign role and generate token

@RegisterRestClient(baseUri = "<http://localhost:8085/ProducerDB/rest/example>")

public interface ApiInterface {

@ClientHeaderParam(name="authorization",value="{generateJWTToken}")

@GET

@Path("student")

@RolesAllowed("admin")

public Collection<Students> getStudent();

default String generateJWTToken()

{

Config config = ConfigProvider.getConfig();

String jwt = config.getValue("jwt-string", String.class);

String authtoken = "Bearer "+jwt;

return authtoken;

}

}

* In servlet

public class app extends HttpServlet {

@Inject @RestClient ApiInterface a;

* Inside body tag

try

{

Collection<Students> students = a.getStudent();

for(Students s:students)

{

out.println(s.getName());

}

}

catch(Exception e)

{

out.println("Something went wrong");

out.println("<br>");

out.println(e);

}

* Inside microprofile-config addd (token file token)

jwt-string=eyJraWQiOiJqd3Qua2V5IiwidHlwIjoiSldUIiwiYWxnIjoi..

Running both application in different terminal

Run both files using below 2 commands (Use 2 terminals)

java -jar payara-micro-5.2022.2.jar --deploy MSAApp/artifact/MSAApp.war --port 8085 --addlibs mysql-connector-java-5.0.8-bin.jar --domainconfig domain.xml

java -jar payara-micro-5.2022.2.jar --deploy MSAClient/artifact/MSAClient.war --port 8087 --addlibs mysql-connector-java-5.0.8-bin.jar --domainconfig domain.xml

api checking:=

<http://localhost:8085/MSAApp/rest(bootstrep)/example(exampleservice)/student(methoduperpath)>

Url for running DocFront App : <http://localhost:8087/DocFrontApp/TestClientServlet>

jsf page running url : <http://localhost:8087/tryprod/faces/display.xhtml>

—---------------------------------------------------add—--------------------------------------------

@Override

public void addCategory(String CategoryName, String Description,String Image) {

Category cat=new Category(CategoryName,Description,Image);

em.persist(cat);

}

@Path("addCity/{CityName}/{StateID}")

@POST

public void addCity(@PathParam("CityName") String CityName,@PathParam("StateID") Integer StateID) {

pbl.addCity(CityName, StateID);

}