Create a registration form in angular application on click on button the form data will be passed to a controller from controller to service and form angular service to spring rest controller through it using DAO must be stored in oracle database

BackEnd

-------------

Database configuration Oracle jar installation:

1. Download ojdbc6.jar from the link: “ <http://www.oracle.com/technetwork/apps-tech/jdbc-112010-090769.html> “
2. Copy the ojdbc6.jar inside the “E:\” drive. (don’t save this file in any folder named with space)
3. Set the path of MAVEN to run “mvn” command from command prompt to the system variable.

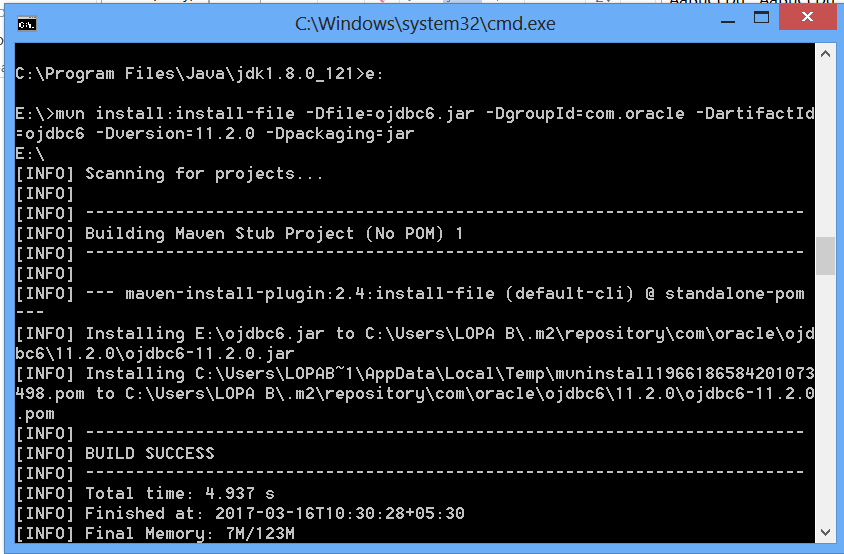
“E:\DevOps\Required Software\apache-maven-3.3.9-bin\apache-maven-3.3.9\bin\;”

“path\_to\_maven\apache-maven-X.X.X\bin”

1. Open command prompt and type the following command to install ojdbc6.jar in maven.

C:\> cd C:\Program Files\Java\jdk1.8.0\_121;

C:\Program Files\Java\jdk1.8.0\_121>mvn install:install-file -Dfile=ojdbc.jar -DgroupId=com.oracle -DartifactId=ojdbc6 -Dversion=11.2.0 -Dpackaging=jar



1. Open Backend project pom.xml file and add the following dependency code

<dependency>

<groupId>com.oracle</groupId>

<artifactId>ojdbc6</artifactId>

<version>11.2.0</version>

</dependency>

1. Create “User” model class inside model package and type the following code:

package com.coll.OnlineCollaborateBackEnd.model;

@Entity

public class User implements Serializable{

private static final long serialVersionUID = -2800377271876935833L;

@Id

@GeneratedValue(strategy=GenerationType.***SEQUENCE***)

int userId;

String fullname,username,password,confirm,email,role;

public int getUserId() {

return userId;

}

public void setUserId(int userId) {

this.userId = userId;

}

public User() {

super();

// TODO Auto-generated constructor stub

}

public User(int userId, String fullname, String username, String password, String confirm, String email,

String role) {

super();

this.userId = userId;

this.fullname = fullname;

this.username = username;

this.password = password;

this.confirm = confirm;

this.email = email;

this.role = role;

}

public String getFullname() {

return fullname;

}

public void setFullname(String fullname) {

this.fullname = fullname;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public String getConfirm() {

return confirm;

}

public void setConfirm(String confirm) {

this.confirm = confirm;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getRole() {

return role;

}

public void setRole(String role) {

this.role = role;

}

public static long getSerialversionuid() {

return serialVersionUID;

}

}

1. Open pom.xml file and add dependencies for hibernate and spring-orm as following:

<!-- HIBERNATE -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>${hibernate.version}</version>

</dependency>

<!-- SPRING ORM -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>${spring.version}</version>

</dependency>

1. Open Oracle sql\* plus and create an User with following commands:

Create user collaborate identified by collab;

Alter user collaborate identified by collab account unlock;

Grant connect to collaborate;

Grant resource, dba to collaborate;

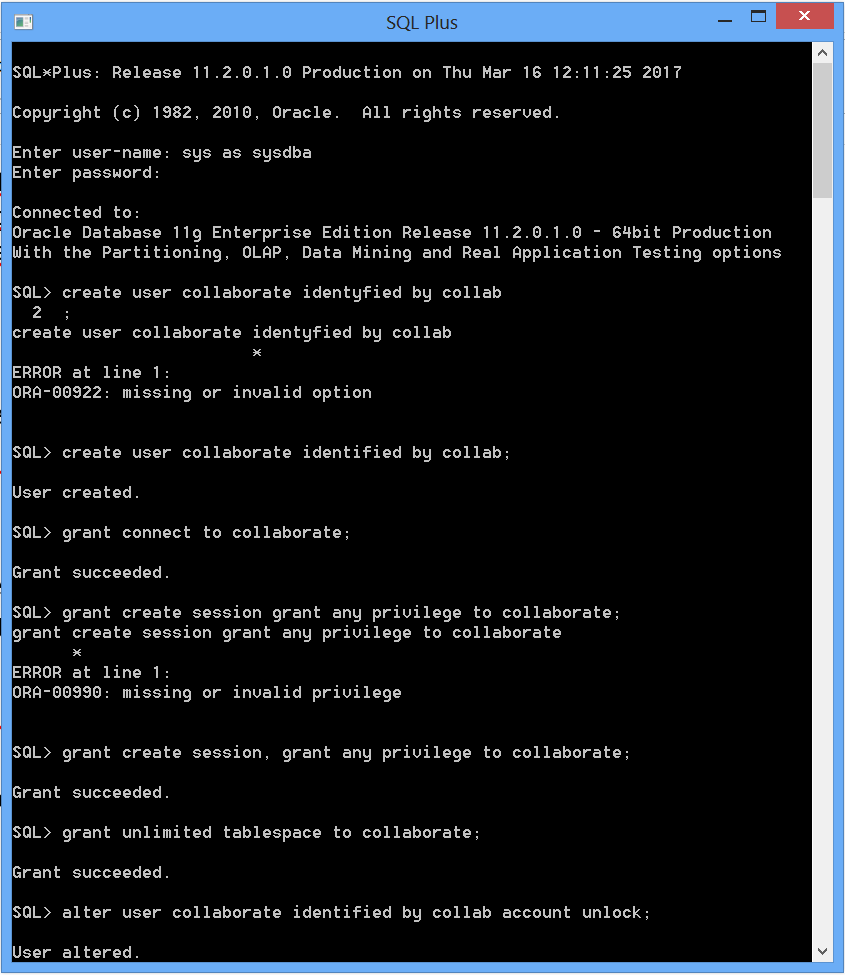
Grant create session, grant any privilege to collaborate;

Grant create sequence to collaborate;

Grant unlimited tablespace to collaborate;

Grant create table to collaborate;

Grant select, update, insert, delete on collaborate.UserDetail to collaborate;



1. Create a class HibernateConfig class inside the config package and type the following code:

package com.coll.OnlineCollaborateBackEnd.config;

import java.util.Properties;

import javax.sql.DataSource;

import org.hibernate.SessionFactory;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.context.annotation.Configuration;

import org.springframework.jdbc.datasource.DriverManagerDataSource;

import org.springframework.orm.hibernate5.HibernateTransactionManager;

import org.springframework.orm.hibernate5.LocalSessionFactoryBuilder;

import org.springframework.transaction.annotation.EnableTransactionManagement;

@Configuration

@ComponentScan(basePackages={"com.coll.OnlineCollaborateBackEnd"})

@EnableTransactionManagement

public class HibernateConfig {

//Database related information

public static final String DATABASE\_URL="jdbc:oracle:thin:@localhost:1521:orcl1";

public static final String DATABASE\_DRIVER="oracle.jdbc.driver.OracleDriver";

public static final String DATABASE\_DIALECT="org.hibernate.dialect.Oracle10gDialect";

public static final String DATABASE\_USERNAME="collaborate";

public static final String DATABASE\_PASSWORD="collab";

@Bean(name="dataSource")

public DataSource getDataSource(){

DriverManagerDataSource dataSource=new DriverManagerDataSource();

dataSource.setDriverClassName(DATABASE\_DRIVER);

dataSource.setUrl(DATABASE\_URL);

dataSource.setUsername(DATABASE\_USERNAME);

dataSource.setPassword(DATABASE\_PASSWORD);

//datasource bean will be available

return dataSource;

}

//sessionFactory will be available

@Bean

public SessionFactory getSessionFactory(DataSource dataSource){

LocalSessionFactoryBuilder builder= new LocalSessionFactoryBuilder(dataSource);

builder.addProperties(getHibernateProperties());

builder.scanPackages("com.health.HealthMedicineQuestBackEnd");

return builder.buildSessionFactory();

}

//All the hibernate properties will be returned by this method

public Properties getHibernateProperties(){

Properties prop=new Properties();

prop.put("hibernate.dialect", DATABASE\_DIALECT);

prop.put("hibernate.show\_sql", "true");

prop.put("hibernate.hbm2ddl.auto", "update");

prop.put("hibernate.format\_sql", "true");

return prop;

}

//Transaction Management

@Bean

public HibernateTransactionManager getTransactionManager(SessionFactory sessionFactory){

HibernateTransactionManager transactionManager=new HibernateTransactionManager(sessionFactory);

return transactionManager;

}

}

1. Create dao and implDao packages inside the context root path and create IUserDAO interface and UserDAOImpl class inside dao and implDao respectively.
2. Type the following code inside the IUserDAO interface:

package com.coll.OnlineCollaborateBackEnd.dao;

import java.util.List;

import com.coll.OnlineCollaborateBackEnd.model.User;

public interface IUserDAO {

public List<User> getAllUsers();

public User getUser(int userId);

public User getUserByUserName(String username);

public boolean addUser(User u);

public boolean updateUser(User u);

public boolean deleteUser(int userId);

}

1. Type the following code inside the UserDAOImpl class:

package com.coll.OnlineCollaborateBackEnd.implDao;

import java.util.List;

import javax.transaction.Transactional;

import org.hibernate.SessionFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Repository;

import com.coll.OnlineCollaborateBackEnd.dao.IUserDAO;

import com.coll.OnlineCollaborateBackEnd.model.User;;

@Repository("userDAO")

@Transactional

public class UserDAOImpl implements IUserDAO{

@Autowired

SessionFactory sessionFactory;

@Override

public List<User> getAllUsers() {

return sessionFactory.getCurrentSession().createQuery("from User", User.class).getResultList();

}

@Override

public User getUser(int userId) {

return sessionFactory.getCurrentSession().get(User.class,Integer.valueOf(userId));

}

@Override

public User getUserByUserName(String username) {

String selectUser="from User where username=:username";

return sessionFactory

.getCurrentSession()

.createQuery(selectUser, User.class)

.setParameter("username",username)

.getSingleResult();

}

@Override

public boolean addUser(User u) {

try {

sessionFactory.getCurrentSession().persist(u);

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

@Override

public boolean updateUser(User u) {

try {

sessionFactory.getCurrentSession().update(u);

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

@Override

public boolean deleteUser(int userId) {

try {

sessionFactory.getCurrentSession().delete(getUser(userId));

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

}

1. Create test package and create UserDAOTestCase and type the following code snippets to test User dao:

package com.coll.OnlineCollaborateBackEnd.test;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

import static org.junit.Assert.\*;

import org.junit.BeforeClass;

import org.junit.Test;

import com.coll.OnlineCollaborateBackEnd.dao.IUserDAO;

import com.coll.OnlineCollaborateBackEnd.model.User;

public class UserDAOTestCase {

private static AnnotationConfigApplicationContext context;

private static IUserDAO userDAO;

private User user;

@BeforeClass

public static void init(){

context=new AnnotationConfigApplicationContext();

context.scan("com.health.HealthMedicineQuestBackEnd");

context.refresh();

userDAO=(IUserDAO)context.getBean("userDAO");

}

@Test

public void testAddUser(){

user=new User();

user.setFullname("Lopamudra Bera");

user.setUsername("LopaB");

user.setPassword("lopab");

user.setConfirm("lopab");

user.setEmail("pinkfriend@gmail.com");

user.setRole("user");

assertEquals("Successfully added a user inside the table!", true, userDAO.addUser(user));

}

// @Test

// public void testUpdateUser(){

// user=userDAO.getUser(1);

// //assertEquals("Successfully added a user inside the table!", "LopaB", user.getUserName());

// //user=new User();

// user.setFullName("Lopamudra Bera");

// user.setUserName("LopaB");

// user.setPassword("lopabm");

// user.setUserConfirmPassword("lopabm");

// user.setUserEmail("pinkfriend@gmail.com");

// user.setRole("user");

//

// assertEquals("Successfully updated a user inside the table!", true, userDAO.updateUser(user));

//

// }

// @Test

// public void testDeleteUser(){

// assertEquals("Successfully deleted a user from the table!", true, userDAO.deleteUser(1));

//

// }

}

Creating sequence for UserDetail table

CREATE SEQUENCE user\_seq

START WITH 1

INCREMENT BY 1

NOCACHE

NOCYCLE;

------------------

FrontEnd

------------------

1. Create a registration folder inside the component and create 3 files named RegisterServices.js, RegisterController.js and register.html
2. Open register.html and design the form with the following code:

<div class="container">

<div class="row main">

<div class="form-container">

<h5>Sign up</h5>

<form class="" method="post" action="#">

<div class="alert alert-success" ng-if="regCtrl.message!=''">

{{regCtrl.message}}

</div>

<div class="form-group">

<label for="name" class="cols-sm-2 control-label">Full Name</label>

<div class="cols-sm-10">

<div class="input-group">

<span class="input-group-addon"><i class="fa fa-user fa" aria-hidden="true"></i></span>

<input type="text" ng-model="regCtrl.user.fullname" class="form-control" name="fullname" id="fullname" placeholder="Enter Full Name"/>

</div>

</div>

</div>

<div class="form-group">

<label for="email" class="cols-sm-2 control-label">Email</label>

<div class="cols-sm-10">

<div class="input-group">

<span class="input-group-addon"><i class="fa fa-envelope fa" aria-hidden="true"></i></span>

<input type="text" ng-model="regCtrl.user.email" class="form-control" name="email" id="email" placeholder="Enter Email Id"/>

</div>

</div>

</div>

<div class="form-group">

<label for="username" class="cols-sm-2 control-label">Username</label>

<div class="cols-sm-10">

<div class="input-group">

<span class="input-group-addon"><i class="fa fa-users fa" aria-hidden="true"></i></span>

<input type="text" ng-model="regCtrl.user.username" class="form-control" name="username" id="username" placeholder="Enter Username"/>

</div>

</div>

</div>

<div class="form-group">

<label for="username" class="cols-sm-2 control-label">Role</label>

<div class="cols-sm-10">

<div class="input-group">

<span class="input-group-addon"><i class="fa fa-users fa" aria-hidden="true"></i></span>

<input type="text" ng-model="regCtrl.user.role" class="form-control" name="role" id="role" placeholder="Enter Role"/>

</div>

</div>

</div>

<div class="form-group">

<label for="password" class="cols-sm-2 control-label">Password</label>

<div class="cols-sm-10">

<div class="input-group">

<span class="input-group-addon"><i class="fa fa-lock fa-lg" aria-hidden="true"></i></span>

<input type="password" ng-model="regCtrl.user.password" class="form-control" name="password" id="password" placeholder="Enter Password"/>

</div>

</div>

</div>

<div class="form-group">

<label for="confirm" class="cols-sm-2 control-label">Confirm Password</label>

<div class="cols-sm-10">

<div class="input-group">

<span class="input-group-addon"><i class="fa fa-lock fa-lg" aria-hidden="true"></i></span>

<input type="password" ng-model="regCtrl.user.confirm" class="form-control" name="confirm" id="confirm" placeholder="Confirm Password"/>

</div>

</div>

</div>

<div class="form-group ">

<input type="button" ng-click="regCtrl.register()" value="Register" class="btn btn-primary btn-lg btn-block login-button"/>

</div>

</form>

</div>

</div>

</div>

Open RegisterServices.js add type the following code:

var RegisterModule=angular.module('RegisterModule',[]);

RegisterModule.service('RegisterService',['$http','$q','REST\_URI',function($http,$q,REST\_URI){

this.register=function(user){

//console.log(user);

var deferred=$q.defer();

$http.post(REST\_URI+'/user/add',user).then(

function(response){

deferred.resolve(response.data.responseMessage);

},

function(error){

deferred.reject(error);

}

);

return deferred.promise;

}

}]);

Open RegisterController.js and add following code:

RegisterModule.controller('RegisterController', ['RegisterService', function (RegisterService) {

//declaring self object

var me = this;

me.user = {};

me.message = '';

me.register = function () {

RegisterService.register(me.user).then(

function (message) {

me.message = message;

console.log(message);

},

function (error) {

console.log(me.user);

}

);

}

}]);

Open app.module.js file add update as following:

var app=angular.module("CollaborationApp",

[

'ngRoute',

'BasicModule',

**'RegisterModule'**

]

);

Open app.route.js and update as following:

window.routes={

"/home":{

templateUrl:'app/components/basic/home.html',

controller: 'BasicController',

controllerAs: 'basicCtrl',

requiredLogin:'false',

roles:['USER']

},

"/about":{

templateUrl:'app/components/basic/about.html',

controller: 'BasicController',

controllerAs: 'basicCtrl',

requiredLogin:'false',

roles:['USER']

},

"/login":{

templateUrl:'app/components/authentication/login.html',

controller: 'AuthenticationController',

controllerAs: 'authCtrl',

requiredLogin:'false',

roles:['USER']

},

**"/register":{**

**templateUrl:'app/components/register/register.html',**

**controller: 'RegisterController',**

**controllerAs: 'regCtrl',**

**requiredLogin:'false',**

**roles:['USER']**

}

};

//specify the the backend url from where you are going to get the values

app.constant('REST\_URI','http://localhost:8080/Online-Collaborate');

app.config(['$routeProvider',function($routeProvider){

for (var path in window.routes){

$routeProvider.when(path,window.routes[path]);

}

$routeProvider.otherwise({redirectTo:'/login'});

}]);

Open index.html file and update as following:

<html lang="en" ng-app="CollaborationApp">

<head>

<title></title>

<!-- Loading the angular library -->

<script src="./bower\_components/angular/angular.js"></script>

<script src="./bower\_components/angular-route/angular-route.js"></script>

**<script src="./bower\_components/jquery/dist/jquery.js"></script>**

**<script src="./bower\_components/bootstrap/dist/js/bootstrap.js"></script>**

<!-- Loading the main module -->

<script src="./app/app.module.js"></script>

<script src="./app/app.route.js"></script>

<script src="./app/components/basic/BasicServices.js"></script>

<script src="./app/components/basic/BasicController.js"></script>

**<script src="./app/components/register/RegisterServices.js"></script>**

**<script src="./app/components/register/RegisterController.js"></script>**

<!-- Loading custom stylesheet -->

**<link href="./bower\_components/bootstrap/dist/css/bootstrap.css"/>**

**<link href="./assets/css/reg.css"/>**

</head>

<body>

<div ng-view></div>

</div>

</body>

</html>