**Task :** 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

a)Configuring oracle

1.Open the folder created for socialProject , in eclipse.Location is e:/project2/online-collaboration/socialProject , I’ve used e:/project2/online-collaboration as my workspace.

2.Download ojdbc6 jar for oracle from the following link :

<http://www.oracle.com/technetwork/apps-tech/jdbc-112010-090769.html>

3.Set the path of maven in the path environment variable :

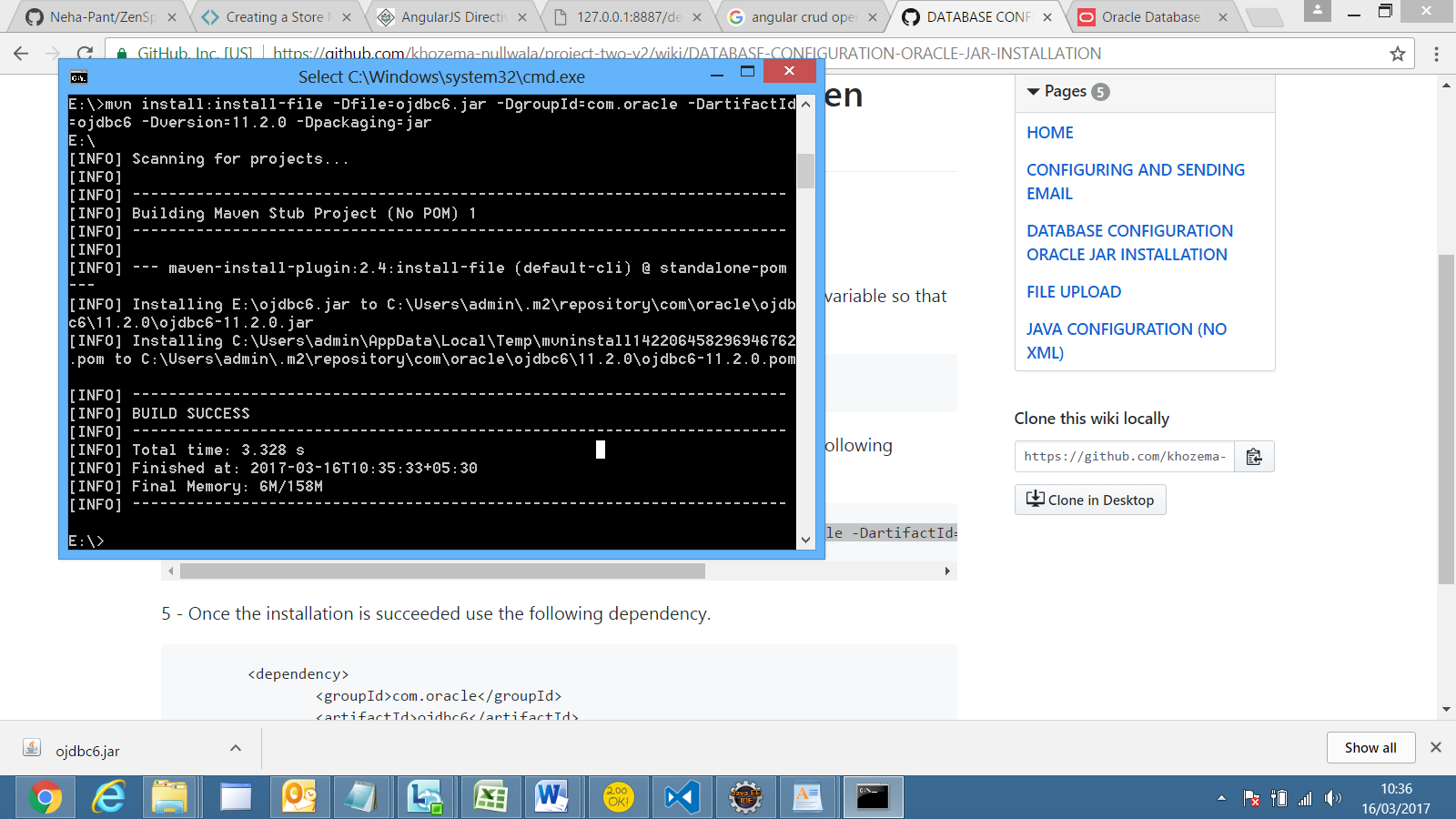
E:\Dev Oops TTT (27feb17\_17mar17)\x64\apache-maven-3.3.9\bin\

4.Check by typing “mvn” in cmd to check the path variable configuration.

5.In the environment variable JAVA\_HOME , put the root path of the jdk.

Eg: C:\Program Files\Java\jdk1.8.0\_11

6.Type : (copy the jar ojdbc6.jar to e: and goto that location and type)



7.In the backend pom , add the following dependency for ojdbc6:

<oracle.version>11.2.0</oracle.version>

<!-- oracle dependency ojdbc6 -->

<dependency>

<groupId>com.oracle</groupId>

<artifactId>ojdbc6</artifactId>

<version>11.2.0</version>

</dependency>

b)Create a new user

1.After the oracle is installed , create a new user , open cmd , and write following commands in cmd after sqlplus , username as / as sysdba password is admin :

create user social identified by social;

grant connect , resource , dba to social;

Grant create session , grant any privilege to social;

Grant unlimited tablespace to social;

Grant select,insert,update,delete on User to social;

Alter user social identified by social account unlock;

c)Starting Oracle

1.Open services.msc through cmd.

2.Start services :

XE , Agent,Listener

d)Adding dependencies in eclipse project pom.

1.Add the hibernate dependency:

<hibernatecore.version>5.2.5.Final</hibernatecore.version>

<!-- Hibernate-core -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

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

</dependency>

2.Add the spring orm dependency:

<spring.version>4.3.6.RELEASE</spring.version>

<!-- Spring orm dependency -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

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

</dependency>

3.Jackson dependency has to be added :

<jackson.version>2.8.7</jackson.version>

<!-- JACKSON -->

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-core</artifactId>

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

</dependency>

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

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

</dependency>

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-annotations</artifactId>

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

</dependency>

4.Properties:

<properties>

<spring.version>4.3.6.RELEASE</spring.version>

<jackson.version>2.8.7</jackson.version>

<oracle.version>11.2.0</oracle.version>

<hibernatecore.version>5.2.5.Final</hibernatecore.version>

<commons-fileupload.version>1.3.2</commons-fileupload.version>

<commons-io.version>2.5</commons-io.version>

<hibernate.validator.version>5.1.2.Final</hibernate.validator.version>

</properties>

5.Dependencies;  
<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.12</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>javax</groupId>

<artifactId>javaee-api</artifactId>

<version>7.0</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

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

</dependency>

<!-- JACKSON -->

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-core</artifactId>

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

</dependency>

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

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

</dependency>

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-annotations</artifactId>

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

</dependency>

<!-- oracle dependency ojdbc6 -->

<dependency>

<groupId>com.oracle</groupId>

<artifactId>ojdbc6</artifactId>

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

</dependency>

<!-- Hibernate-core -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

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

</dependency>

<!-- Spring orm dependency -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

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

</dependency>

<!-- Fileupload Dependency -->

<dependency>

<groupId>commons-fileupload</groupId>

<artifactId>commons-fileupload</artifactId>

<version>${commons-fileupload.version}</version>

</dependency>

<!-- Commons IO dependency -->

<dependency>

<groupId>commons-io</groupId>

<artifactId>commons-io</artifactId>

<version>${commons-io.version}</version>

</dependency>

<!-- Spring web dependency -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

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

</dependency>

<!-- Hibernate validator dependency -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-validator</artifactId>

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

</dependency>

e)Configuring hibernateconfig.java

1.Inside the config package of the backend project , add a class ,HibernateConfig.java.

2.Inside it , add:

@Configuration

@ComponentScan(basePackages = { "org.socialnetworking" })

@EnableTransactionManagement

**public** **class** HibernateConfig

{

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

**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*** = "social";

**public** **static** **final** String ***DATABASE\_PASSWORD*** = "social";

// dataSource bean

@Bean(name="dataSource")

**public** DataSource getDataSource() {

DriverManagerDataSource dataSource=**new** DriverManagerDataSource();

// Providing the database connection information

dataSource.setDriverClassName(***DATABASE\_DRIVER***);

dataSource.setUrl(***DATABASE\_URL***);

dataSource.setUsername(***DATABASE\_USERNAME***);

dataSource.setPassword(***DATABASE\_PASSWORD***);

**return** dataSource;

}

// SessionFactory bean

@Bean(name="sessionFactory")

**public** SessionFactory getSessionFactory(DataSource dataSource) {

LocalSessionFactoryBuilder builder=**new** LocalSessionFactoryBuilder(dataSource);

builder.addProperties(getHibernateProperties());

builder.scanPackages("org.socialnetworking");

**return** builder.buildSessionFactory();

}

**private** Properties getHibernateProperties() {

Properties properties=**new** Properties();

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

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

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

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

**return** properties;

}

//Transaction manager

@Bean

**public** HibernateTransactionManager getTransactionManager(SessionFactory sessionFactory)

{

HibernateTransactionManager transactionManager=**new** HibernateTransactionManager(sessionFactory);

**return** transactionManager;

}

}

f)User DTO,DAO and DAOImpl

1.In the backend project , create three packages , model , dao and daoImpl.

2.Inside the model,add a class, User.java.

**package** org.socialnetworking.model;

**import** java.io.Serializable;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.SequenceGenerator;

**import** javax.persistence.Table;

**import** javax.persistence.Transient;

**import** org.springframework.web.multipart.MultipartFile;

@Entity

@Table(name = "UserDetails")

**public** **class** User **extends** DomainResponse **implements** Serializable {

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = -3460250219598355519L;

@Id

@GeneratedValue(strategy = GenerationType.***SEQUENCE***, generator = "generator")

@SequenceGenerator(name = "generator", sequenceName = "users\_seq", allocationSize = 1)

**private** **int** userId;

**private** String userName;

**private** String fullName;

**private** String password;

**private** String conPassword;

**private** String email;

**private** String phone;

**private** String role;

**private** String isOnline="false";

**public** String userImage;

**public** String getUserImage() {

**return** userImage;

}

**public** **void** setUserImage(String userImage) {

**this**.userImage = userImage;

}

// @Transient

// private MultipartFile file;

// public MultipartFile getFile()

// {

// return file;

// }

// public void setFile(MultipartFile file)

// {

// this.file=file;

// }

**public** User(**int** userId, String userName, String fullName, String password, String conPassword, String email,

String phone, String role) {

**super**();

**this**.userId = userId;

**this**.userName = userName;

**this**.fullName = fullName;

**this**.password = password;

**this**.conPassword = conPassword;

**this**.email = email;

**this**.phone = phone;

**this**.role = role;

**this**.isOnline = "false";

}

**public** String getIsOnline() {

**return** isOnline;

}

**public** **void** setIsOnline(String isOnline) {

**this**.isOnline = isOnline;

}

**public** **static** **long** getSerialversionuid() {

**return** ***serialVersionUID***;

}

**public** User() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** **int** getUserId() {

**return** userId;

}

**public** **void** setUserId(**int** userId) {

**this**.userId = userId;

}

**public** String getUserName() {

**return** userName;

}

**public** **void** setUserName(String userName) {

**this**.userName = userName;

}

**public** String getFullName() {

**return** fullName;

}

**public** **void** setFullName(String fullName) {

**this**.fullName = fullName;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

**public** String getConPassword() {

**return** conPassword;

}

**public** **void** setConPassword(String conPassword) {

**this**.conPassword = conPassword;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** String getPhone() {

**return** phone;

}

**public** **void** setPhone(String phone) {

**this**.phone = phone;

}

**public** String getRole() {

**return** role;

}

**public** **void** setRole(String role) {

**this**.role = role;

}

@Override

**public** String toString() {

**return** "User [userId=" + userId + ", userName=" + userName + ", fullName=" + fullName + ", password=" + password

+ ", conPassword=" + conPassword + ", email=" + email + ", phone=" + phone + ", role=" + role

+ ", isOnline=" + isOnline + "]";

}

}

3.Inside the dao package , add an interface IUserDAO.java.

**package** dao;

**import** java.util.List;

**import** org.socialnetworking.model.User;

**public** **interface** IUserDAO{

**public** User getUser(**int** id);

**public** List<User> getAllUsers();

**public** List<User> getDUsers();

**public** List<User> getAUsers();

**public** User getUserByUserName(String userName);

**public** **boolean** activateUser(**int** id);

**public** **boolean** addUser(User u);

**public** **boolean** updateUser(User u);

**public** **boolean** deleteUser(**int** id);}

4.Inside the daoimpl package , add a class , UserDAOImpl.java.

**package** daoImpl;

**import** java.util.List;

**import** org.hibernate.SessionFactory;

**import** org.socialnetworking.model.User;

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

**import** org.springframework.stereotype.Repository;

**import** org.springframework.transaction.annotation.Transactional;

**import** dao.IUserDAO;

@Repository("userDAO")

@Transactional

**public** **class** UserDAOImpl **implements** IUserDAO

{

@Autowired

**private** SessionFactory sessionFactory;

**public** List<User> getAllUsers() {

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

}

**public** List<User> getDUsers() {

**return** sessionFactory.getCurrentSession().createQuery("FROM User u where u.isOnline='false'", User.**class**).getResultList();

}

**public** List<User> getAUsers() {

**return** sessionFactory.getCurrentSession().createQuery("FROM User u where u.isOnline='true'", User.**class**).getResultList();

}

**public** User getUser(**int** id) {

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

}

**public** **boolean** updateUser(User u) {

**try** {

sessionFactory.getCurrentSession().update(u);

**return** **true**;

} **catch** (Exception ex) {

ex.printStackTrace();

**return** **false**;

}

}

**public** **boolean** deleteUser(**int** id) {

**try** {

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

**return** **true**;

} **catch** (Exception ex) {

ex.printStackTrace();

**return** **false**;

}

}

**public** **boolean** addUser(User u) {

**try** {

sessionFactory.getCurrentSession().persist(u);

**return** **true**;

} **catch** (Exception ex) {

ex.printStackTrace();

**return** **false**;

}

}

@Override

**public** User getUserByUserName(String userName) {

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

**return** (User) sessionFactory.getCurrentSession().createQuery(selectUser, User.**class**)

.setParameter("userName", userName).getSingleResult();

}

@Override

**public** **boolean** activateUser(**int** id) {

**try** {

User user=getUser(id);

user.setIsOnline("true");

sessionFactory.getCurrentSession().update(user);

**return** **true**;

} **catch** (Exception ex) {

ex.printStackTrace();

**return** **false**;

}

}

}

g)UserTestCases

1.Add a java file inside test package:

package org.socialnetworking.test;

import org.junit.BeforeClass;

import org.junit.Test;

import org.socialnetworking.dao.IUserDAO;

import static org.junit.Assert.\*;

import org.junit.BeforeClass;

import org.junit.Test;

import org.socialnetworking.model.User;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class UserTestCase {

private static AnnotationConfigApplicationContext context;

private static IUserDAO userDAO;

private User user;

@BeforeClass

public static void init() {

context = new AnnotationConfigApplicationContext();

context.scan("org.socialnetworking");

context.refresh();

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

}

// @Test

// public void testAddUser()

// {

// user=new User();

// user.setUserId(1);

// user.setUserName("ed");

// user.setFullName("Edward");

// user.setEmail("edward@cullen.com");

// user.setPhone("8826488032");

// user.setRole("user");

// user.setPassword("ed");

// user.setConPassword("ed");

//

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

// }

//

// @Test

// public void testGetUser()

// {

// user=userDAO.getUser(1);

// assertEquals("Retrieving data based on id from table","Edward",user.getFullName());

// }

//

// @Test

// public void testUpdateUser()

// {

// user=userDAO.getUser(1);

// user.setUserId(1);

// user.setUserName("ed");

// user.setFullName("Edward");

// user.setEmail("edward@cullen.com");

// user.setPhone("8826488032");

// user.setRole("user");

// user.setPassword("pass");

// user.setConPassword("pass");

//

// assertEquals("Updating user based on id from table",true,userDAO.updateUser(user));

// }

//

// @Test

// public void testDeleteUser()

// {

// assertEquals("Deleting user based on id from table",true,userDAO.deleteUser(1));

// }

}

h)Adding controller

1.Inside the java folder , add a package for controller,add a java class UserCOntroller.java.

2.Write the following :

**package** org.socialnetworking.controller;

**import** java.io.File;

**import** javax.servlet.http.HttpServletRequest;

**import** org.socialnetworking.dao.IUserDAO;

**import** org.socialnetworking.model.User;

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

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.CrossOrigin;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.PostMapping;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

**import** org.springframework.web.multipart.MultipartFile;

@CrossOrigin(origins = "http://localhost:8887")

@RestController

@RequestMapping("/user")

**public** **class** UserController {

@Autowired

IUserDAO userDAO;

@Autowired

**private** HttpServletRequest request;

@PostMapping("/add")

**public** ResponseEntity<User> addUser(@RequestBody User user) {

// if (!(user.getFile().getOriginalFilename().equals(""))) {

// user.setUserImage(uploadImage(user.getFile()));

// }

**if** (userDAO.addUser(user)) {

System.***out***.println(user);

user.setResponseMessage("You have been successfully registered .");

**return** **new** ResponseEntity<User>(user, HttpStatus.***OK***);

} **else** {

**return** **new** ResponseEntity<User>(user, HttpStatus.***INTERNAL\_SERVER\_ERROR***);

}

}

@GetMapping("/view")

**public** ResponseEntity<User> getUsers() {

**return** **new** ResponseEntity(userDAO.getAllUsers(), HttpStatus.***OK***);

}

@GetMapping("/viewUser")

**public** ResponseEntity<String> getUser(@RequestBody **int** id) {

**return** **new** ResponseEntity(userDAO.getUser(id), HttpStatus.***OK***);

}

@GetMapping("/viewD")

**public** ResponseEntity<User> getDUser() {

**return** **new** ResponseEntity(userDAO.getDUsers(), HttpStatus.***OK***);

}

@GetMapping("/viewA")

**public** ResponseEntity<User> getAUser() {

**return** **new** ResponseEntity(userDAO.getAUsers(), HttpStatus.***OK***);

}

@PostMapping("/delete")

**public** ResponseEntity<String> deleteUser(@RequestBody **int** id) {

**if** (userDAO.deleteUser(id)) {

**return** **new** ResponseEntity<String>("User successfully deleted .", HttpStatus.***OK***);

} **else** {

**return** **new** ResponseEntity<String>("Deletion failure.", HttpStatus.***INTERNAL\_SERVER\_ERROR***);

}

}

@PostMapping("/activate")

**public** ResponseEntity<String> activateUser(@RequestBody **int** id) {

**if** (userDAO.activateUser(id)) {

**return** **new** ResponseEntity<String>("User successfully activated .", HttpStatus.***OK***);

} **else** {

**return** **new** ResponseEntity<String>("User activation failure.", HttpStatus.***INTERNAL\_SERVER\_ERROR***);

}

}}

i)FrontEnd Application

1.Index.html

<html ng-app="EAllianzApp">

<head>

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

<link rel="stylesheet" href="./bower\_components/bootstrap/dist/css/bootstrap-theme.css" />

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

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

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

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

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

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

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

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

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

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

</head>

<body>

<div ng-view></div>

</body>

</html>

2.RegisterService.js

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

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

this.register = function (user) {

var deffered = $q.defer();

console.log(user);

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

function (response) {

deffered.resolve(response.data.responseMessage);

},

function (error) {

deffered.reject(error);

}

);

return deffered.promise;

}

this.viewAll = function () {

var deffered = $q.defer();

$http.get(REST\_URI + '/user/view').then(

function (response) {

deffered.resolve(response.data);

},

function (error) {

deffered.reject(error);

}

);

return deffered.promise;

}

this.init = function () {

this.viewA();

this.viewD();

}

this.viewA = function () {

var deffered = $q.defer();

$http.get(REST\_URI + '/user/viewA').then(

function (response) {

deffered.resolve(response.data);

},

function (error) {

deffered.reject(error);

}

);

return deffered.promise;

}

this.viewD = function () {

var deffered = $q.defer();

$http.get(REST\_URI + '/user/viewD').then(

function (response) {

deffered.resolve(response.data);

},

function (error) {

deffered.reject(error);

}

);

return deffered.promise;

}

this.deleteUser = function (id) {

var deffered = $q.defer();

console.log(id);

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

function (response) {

deffered.resolve(response.data.text());

},

function (error) {

deffered.reject(error);

}

);

this.init();

return deffered.promise;

}

this.activateUser = function (id) {

var deffered = $q.defer();

console.log(id);

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

function (response) {

deffered.resolve(response.data.text());

},

function (error) {

deffered.reject(error);

}

);

this.init();

return deffered.promise;

}

}]);

3.RegisterController.js

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

var me=this;

me.user={};

me.message='';

me.users={};

me.usersA={};

me.usersD={};

me.init=function()

{

}

me.register=function(){

console.log(me.user);

RegisterService.register(me.user).then(

function(response){

me.message=response;

console.log(me.message);

},

function(error)

{

console.log(me.user);

}

);

}

RegisterService.viewAll().then(

function(response)

{

me.users=response;

console.log(response);

},

function(error)

{

console.log(error);

}

);

RegisterService.viewA().then(

function(response)

{

me.usersA=response;

console.log(response);

},

function(error)

{

console.log(error);

}

);

RegisterService.viewD().then(

function(response)

{

me.usersD=response;

console.log(response);

},

function(error)

{

console.log(error);

}

);

me.deleteUser=function(id){

RegisterService.deleteUser(id).then(

function(response)

{

console.log(response);

window.location.reload();

},

function(error)

{

console.log(error);

}

);

}

me.activateUser=function(id){

RegisterService.activateUser(id).then(

function(response)

{

console.log(response);

window.location.reload();

},

function(error)

{

console.log(error);

}

);

}

me.userNameError = false;

me.CheckUserName = function()

{

var reg = /^[a-z A-Z]{4,12}$/;

me.UserNameError = !reg.test( $scope.cvv );

}

}]);

4.register.html

<div class="row">

<div class="col-md-2"></div>

<div class="col-md-8">

<h1>Register</h1>

<form name="userForm" ng-submit="submitForm(userForm.$valid)" novalidate>

<div ng-show="error" class="alert alert-danger">{{errorMessage}}</div>

<div class="form-group">

<label for="username">USER NAME</label>

<input type="text" name="username" ng-model="regCtrl.user.userName" ng-change="CheckUserName();" />

<label ng-if="userNameError" class="text text-danger">Invalid User Name</label>

</div>

<div class="form-group">

<label for="fullname">FULL NAME</label>

<input type="text" name="fullname" ng-model="regCtrl.user.fullName" />

</div>

<div class="form-group">

<label for="password">PASSWORD</label>

<input type="text" name="password" ng-model="regCtrl.user.password" />

</div>

<div class="form-group">

<label for="conpassword">CONFIRM PASSWORD</label>

<input type="text" name="conpassword" ng-model="regCtrl.user.conPassword" />

</div>

<div class="form-group">

<label for="email">EMAIL</label>

<input type="text" name="email" ng-model="regCtrl.user.email">

</div>

<div class="form-group">

<label for="phone">MOBILE</label>

<input type="text" name="phone" ng-model="regCtrl.user.phone">

</div>

<div class="form-group">

<label for="role">ROLE</label>

<input type="text" name="role" ng-model="regCtrl.user.role">

</div>

<div class="row">

<div class="form-group col-md-12">

<label for="file">User Image</label>

<input type="text" ng-model="regCtrl.user.userImage">

</div>

</div>

<div class="form-group">

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

</div>

</form>

</div>

<div class="col-md-2"></div>

</div>

</div>

5.users.html

<div class="row">

<div class="col-md-12">

<center>

<h1 style="text-decoration:underline">User's Details</h1>

</center>

<br/>

<center>

<h3>Non Approved Users</h3>

</center>

<table border="1" class="table table-bordered table-hover table-striped">

<tr>

<th>User Image</th>

<th>User ID</th>

<th>User Name</th>

<th>Full Name</th>

<th>Password</th>

<th>Email Id</th>

<th>Role</th>

<th>Phone</th>

<th>Approve</th>

<th>Disapprove</th>

</tr>

<tr ng-repeat="x in regCtrl.usersD">

<td><img ng-src="./assets/images/{{x.userImage}}" height="80" width="80"/></td>

<td>{{x.userId}}</td>

<td>{{x.userName}}</td>

<td>{{x.fullName}}</td>

<td>{{x.password}}</td>

<td>{{x.email}}</td>

<td>{{x.role}}</td>

<td>{{x.phone}}</td>

<td><button ng-click="regCtrl.activateUser(x.userId)" class="btn btn-default" onclick="this.parentElement.style.display='none';"><span class="glyphicon glyphicon-ok"></span></button></td>

<td><button ng-click="regCtrl.deleteUser(x.userId)" class="btn btn-default" onclick="this.parentElement.style.display='none';"><span class="glyphicon glyphicon-remove"></span></button></td>

</tr>

</table>

<center>

<h3>Approved Users</h3>

</center>

<table border="1" class="table table-bordered table-hover table-striped">

<tr>

<th>User Image</th>

<th>User ID</th>

<th>User Name</th>

<th>Full Name</th>

<th>Password</th>

<th>Email Id</th>

<th>Role</th>

<th>Phone</th>

</tr>

<tr ng-repeat="x in regCtrl.usersA">

<td><img ng-src="./assets/images/{{x.userImage}}" height="80" width="80"/></td>

<td>{{x.userId}}</td>

<td>{{x.userName}}</td>

<td>{{x.fullName}}</td>

<td>{{x.password}}</td>

<td>{{x.email}}</td>

<td>{{x.role}}</td>

<td>{{x.phone}}</td>

</tr>

</table>

</div>

</div>

6. app.module.js.

var app=angular.module("EAllianzApp",

[

'ngRoute',

'BasicModule',

'RegisterModule'

]

);

7.app.route.js

window.routes =

{

"/home":

{

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

controller: 'BasicController',

controllerAs: 'basicCtrl',

requireLogin: false,

roles: ['GUEST', 'ADMIN', 'STAFF', 'STUDENT']

},

"/about":

{

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

controller: 'BasicController',

controllerAs: 'basicCtrl',

requireLogin: false,

roles: ['GUEST', 'ADMIN', 'STAFF', 'STUDENT']

},

"/register":

{

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

controller: 'RegisterController',

controllerAs: 'regCtrl',

requireLogin: false,

roles: ['GUEST']

},

"/users":

{

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

controller: 'RegisterController',

controllerAs: 'regCtrl',

requireLogin: false,

roles: ['GUEST']

},

"/login":

{

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

controller: 'AuthenticationController',

controllerAs: 'authCtrl',

requireLogin: false,

roles: ['GUEST', 'ADMIN', 'STAFF', 'STUDENT']

}

};

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

app.constant('REST\_URI', 'http://localhost:9090/socialnetworking');

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

for (var path in window.routes) {

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

}

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

}]);