Register page and configuration and DAO and DTO and test case and insertion in db

1.Include the register page in views folder of the frontendproject.

a.Create a jsp page under the hierarchy and name it register.

b.Include the spring form for registration of the user in it.

<%@taglib prefix="spring" uri="http://www.springframework.org/tags"%>

<%@taglib prefix="form" uri="http://www.springframework.org/tags/form"%>

<form class="form-horizontal" action="userData.do" method="POST"

commandname="user" modelattribute="user">

<div class="container">

<div class="row">

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

<div class="form-container col-md-8 jumbotron">

<form:form class="form-horizontal" action="userData.do" method="POST"

commandName="user" modelAttribute="user">

<h3>Registration Form</h3>

<h1>Register</h1>

<div class="row">

<div class="form-group">

<label class="col-md-3 control-lable" for="userName">UserName</label>

<form:input path="userName" type="text" class="form-control"

placeholder="flint" />

<div class="has-error">

<form:errors style="color:red" path="userName"

class="help-inline" />

</div>

</div>

</div>

<div class="row">

<div class="form-group">

<label class="col-md-3 control-lable" for="userFullName">UserFullName</label>

<form:input path="userFullName" type="text" class="form-control"

placeholder="flint salvatore" />

<div class="has-error">

<form:errors style="color:red" path="userFullName"

class="help-inline" />

</div>

</div>

</div>

<div class="row">

<div class="form-group">

<label class="col-md-3 control-lable" for="userRole">Role</label>

<form:select path="userRole" type="text" class="form-control">

<form:option value="Customer" label="User" selected="true" />

<form:option value="Supplier" label="Supplier" />

</form:select>

<div class="has-error">

<form:errors style="color:red" path="userRole"

class="help-inline" />

</div>

</div>

</div>

<div class="row">

<div class="form-group">

<label class="col-md-3 control-lable" for="userPassword">Password</label>

<form:input path="userPassword" type="password"

class="form-control" placeholder="pass@123" />

<div class="has-error">

<form:errors style="color:red" path="userPassword"

class="help-inline" />

</div>

</div>

</div>

<div class="row">

<div class="form-group">

<label class="col-md-3 control-lable" for="userConPassword">Confirm

Password</label>

<form:input path="userConPassword" type="password"

class="form-control" placeholder="pass@123" />

<div class="has-error">

<form:errors style="color:red" path="userConPassword"

class="help-inline" />

</div>

</div>

</div>

<div class="row">

<div class="form-group">

<label class="col-md-3 control-lable" for="userAddress">Address</label>

<form:input path="userAddress" type="text" class="form-control"

placeholder="flat no,society name,city,country" />

<div class="has-error">

<form:errors style="color:red" path="userAddress"

class="help-inline" />

</div>

</div>

</div>

<div class="row">

<div class="form-group">

<label class="col-md-3 control-lable" for="userEmail">Email</label>

<form:input path="userEmail" type="email"

placeholder="abc@xyz.com" class="form-control" />

<div class="has-error">

<form:errors style="color:red" path="userEmail"

class="help-inline" />

</div>

</div>

</div>

<div class="row">

<div class="form-group">

<label class="col-md-3 control-lable" for="userMobile">Mobile

Number</label>

<form:input path="userMobile" type="text"

placeholder="+919945678902" class="form-control" maxlength="13"

minlength="13" />

<div class="has-error">

<form:errors style="color:red" path="userMobile"

class="help-inline" />

</div>

</div>

</div>

<div class="row">

<div class="form-group">

<label class="col-md-3 control-lable" for="userSecurityQ">Security

Question</label>

<form:select path="userSecurityQ" class="form-control">

<form:option value="Name of the first pet"

label="Name of the first pet" selected="true" />

<form:option value="Name of the favourite book"

label="Name of the favourite book" />

<form:option value="Name of the favourite sport"

label="Name of the favourite sport" />

</form:select>

<div class="has-error">

<form:errors style="color:red" path="userSecurityQ"

class="help-inline" />

</div>

</div>

</div>

<div class="row">

<div class="form-group">

<label class="col-md-3 control-lable" for="userSecurityA">Security

Answer</label>

<form:input path="userSecurityA" type="text" placeholder="demo"

class="form-control" />

<div class="has-error">

<form:errors style="color:red" path="userSecurityA"

class="help-inline" />

</div>

</div>

</div>

<div class="row">

<div>

<div class="form-actions floatRight">

<input type="submit" name="action" class="btn btn-primary btn-sm"

value="Register" />

</th>

</div>

</div>

</div>

</form:form>

</div>

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

</div>

</div>

c.Goto the navbar page in shared folder , and add admin menu and inside it relocate productcrud page link .

<li class="dropdown">ADMIN

<ul>

<li id="productcrud"><a href="${contextRoot}/productCRUD"><span

class="glyphicon glyphicon-list"></span> Product Management</a></li>

</ul>

</li>

d. Inside it give link to register link.

<li id="register"><a href="${contextRoot}/register"><span

class="glyphicon glyphicon-plus"></span> Register</a></li>

e.Goto myapp.js , and add a case.

case 'Register':

$('#register').addClass('active');

break;

f.Goto the page.jsp , and in the content of body , add the following :

<!-- Register Page -->

<c:if test="${userClickRegister == true }">

<%@include file="./register.jsp"%>

</c:if>

2.Add dto for user inside the model folder of the backendproject .

a.Code:

@Entity

public class User implements Serializable{

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int userId;

@Size(min=1, max=30,message="user name should be between 1 to 30 characters long")

private String userName;

@Size(min=1, max=30,message="user full name should be between 1 to 30 characters long")

private String userFullName;

@NotBlank

private String userRole;

@Size(min=1, max=30,message="user password should be between 1 to 30 characters long")

private String userPassword;

@Size(min=1, max=30,message="user confirm password should be between 1 to 30 characters long")

@Transient

private String userConPassword;

@Size(min=1, max=30,message="user address should be between 1 to 30 characters long")

private String userAddress;

@Size(min=1, max=50,message="user email should be between 1 to 50 characters long")

private String userEmail;

@Size(min=1, max=30,message="user name should be between 1 to 30 characters long")

private String userMobile;

@NotBlank

private String userSecurityQ;

@Size(min=1, max=30,message="user security answer should be between 1 to 30 characters long")

private String userSecurityA;

private boolean enabled;

public boolean isEnabled() {

return enabled;

}

public void setEnabled(boolean enabled) {

this.enabled = enabled;

}

b.Add constructors ,parameterised and non parameterised inside it.

c.Add getters and setters for all the fields.

3.Create DAO entity.

a.Add an interface , name it IUserDAO,inside the dao folder.

b.Add the following code :

public List<User> getAllUsers();

public User getUser(int id);

public boolean updateUser(User u);

public boolean deleteUser(int id);

public boolean addUser(User u);

4.Create DAOImpl.

a.Add a class , UserDAOimpl,inside the DAOImpl.

b.Inside it implement the IUserDAO interface.

c.Add the declarations with dummy body inside , for all the declarations done in IUserDAO.

d.Add the following code inside UserDAOImpl.

@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 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;

}

}

}

4.Creation of UsersTestCase.

a.Inside the src>test>java>net>sports>ZenSportsBackEnd>test , create a java class with the name UsersTestCase.

b.Inside the file , add the test cases for get and add method of Impl.

private static AnnotationConfigApplicationContext context;

private static IUserDAO userDAO;

private User user;

@BeforeClass

public static void init() {

context = new AnnotationConfigApplicationContext();

context.scan("net.sports.ZenSportsBackEnd");

context.refresh();

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

}

@Test

public void testAddUser()

{

user=new User();

user.setEnabled(false);

user.setUserName("Ed");

user.setUserFullName("Edward Cullen");

user.setUserAddress("Forks");

user.setUserMobile("8010017053");

user.setUserPassword("user@1");

user.setUserRole("User");

user.setUserSecurityQ("Name of the first pet");

user.setUserSecurityA("Milo");

user.setUserEmail("edward@cullen.com");

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

}

@Test

public void testGetProduct()

{

user=userDAO.getUser(1);

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

}

@Test

public void testUpdateUser()

{

user=userDAO.getUser(1);

user.setEnabled(false);

user.setUserName("Ed");

user.setUserFullName("Edward Cullen");

user.setUserAddress("Seattle");

user.setUserMobile("8826488032");

user.setUserPassword("bella");

user.setUserRole("User");

user.setUserSecurityQ("Name of the first pet");

user.setUserSecurityA("Milo");

user.setUserEmail("edward@cullen.com");

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));

}

5.Creation of Controller to insert the user details in db

a.Inside the frontendproject , inside the src>main>java>net>sports>ZenSportsFrontEnd>controller , add a class file ,FrontEndUserController.java.

b.Inside this file , add the following :

@Controller

public class FrontEndUserController {

@Autowired

private IUserDAO userDAO;

@RequestMapping(value = { "/register" })

public ModelAndView register() {

ModelAndView model = new ModelAndView("page");

model.addObject("title", "Register");

model.addObject("userClickRegister", true);

model.addObject("user",new User());

return model;

}

@RequestMapping(value = "/userData.do", method = RequestMethod.POST)

public ModelAndView doActions(@ModelAttribute("user") @Valid User user, BindingResult result) {

if (result.hasErrors()) {

ModelAndView model1 = new ModelAndView("page");

model1.addObject("title", "Register");

model1.addObject("userClickRegister", true);

return model1;

}

userDAO.addUser(user);

ModelAndView model1 = new ModelAndView("page");

model1.addObject("title", "Register");

model1.addObject("userClickHome", true);

return model1;

}

}