WebFlow

1. Add following dependency inside the pom.xml file of FrontEnd project

    <!-- Spring WebFlow -->

<dependency>

<groupId>org.springframework.webflow</groupId>

<artifactId>spring-webflow</artifactId>

<version>2.4.4.RELEASE</version>

</dependency>

1. Need to configure spring web flow in the dispatcher. For this we required the namespace

xmlns:webflow=<http://www.springframework.org/schema/webflow-config>

http://www.springframework.org/schema/webflow-config

http://www.springframework.org/schema/webflow-config/spring-webflow-config.xsd

<!-- WebFlow Configuration -->

<webflow:flow-executor id="flowExecutor" />

<webflow:flow-registry id="flowRegistry"

flow-builder-services="flowBuilderServices" base-path="/WEB-INF/views">

<!-- here the id is mapped to invoke this flow -->

<webflow:flow-location id="register" path="/memberShipFlow.xml" />

</webflow:flow-registry>

<webflow:flow-builder-services id="flowBuilderServices"

view-factory-creator="viewFactoryCreator" validator="validator"/>

<!-- Webflow Bean -->

<bean id="validator" class="org.springframework.validation.beanvalidation.LocalValidatorFactoryBean"/>

<bean id="viewFactoryCreator"

class="org.springframework.webflow.mvc.builder.MvcViewFactoryCreator">

<property name="viewResolvers">

<list>

<ref bean="viewResolver" />

</list>

</property>

</bean>

<bean class="org.springframework.webflow.mvc.servlet.FlowHandlerAdapter">

<property name="flowExecutor" ref="flowExecutor" />

</bean>

<bean class="org.springframework.webflow.mvc.servlet.FlowHandlerMapping">

<property name="flowRegistry" ref="flowRegistry" />

<property name="order" value="0" />

</bean>

1. Creating the membership-flow.xml file

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

<flow xmlns="<http://www.springframework.org/schema/webflow>"

xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>"

xsi:schemaLocation="http://www.springframework.org/schema/webflow

<http://www.springframework.org/schema/webflow/spring-webflow.xsd>">

<!-- It will create an instance and keep it inside flowScope -->

<var name="user" class="com.health.FrontEndHealthMedicineQuest.model.User"/>

<var name="address" class="com.health.FrontEndHealthMedicineQuest.model.Address"/>

<!-- Executed when webFlow is triggered -->

<on-start>

<!-- registerHandler is java class available through annotated RegisterHandler.java class. -->

<evaluate expression="registerHandler.initializeFlow()" result="flowScope.registerModel"/>

</on-start>

<view-state id="start" view="registration/register" model="user">

<transition on="submit" to="billingAddress" />

<on-exit>

<evaluate expression="registerHandler.saveUser(registerModel,user)"></evaluate>

</on-exit>

</view-state>

<!-- User enters Billing Address -->

<view-state id="billingAddress" view="registration/billing" model="flowScope.address">

<transition on="submit" to="confirm"/>

<on-exit>

<evaluate expression="registerHandler.saveBillingAddress(registerModel,address)"></evaluate>

</on-exit>

</view-state>

<view-state id="confirm" view="registration/preRegister" model="flowScope.registerModel">

<transition on="edit" to="start" />

<transition on="submit" to="storedetail" />

</view-state>

<action-state id="storedetail">

<evaluate expression="registerHandler.storeDetail(registerModel)" ></evaluate>

<transition on="success" to="welcome" />

<transition on="failure" to="start" />

</action-state>

<end-state id="welcome" view="registration/welcome" />

<end-state id="toHome" view="externalRedirect:contextRelative:/home" />

<end-state id="toException" view="externalRedirect:contextRelative:exception.jsp" />

<!-- these events are available in every state-->

<global-transitions>

<transition on="home" to="toHome" />

<transition on="error" to="toException" />

</global-transitions>

</flow>

1. Create a separate RegisterHandler class inside handler package of FrontEnd project to write methods which will be used inside the flow.

package com.health.HealthMedicineQuestFrontEnd.handler;

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

import org.springframework.stereotype.Component;

import com.health.HealthMedicineQuestBackEnd.dao.IUserDAO;

import com.health.HealthMedicineQuestBackEnd.dao.RegisterModel;

import com.health.HealthMedicineQuestBackEnd.model.Address;

import com.health.HealthMedicineQuestBackEnd.model.Cart;

import com.health.HealthMedicineQuestBackEnd.model.User;

@Component

public class RegisterHandler {

@Autowired

IUserDAO userDAO;

public RegisterModel initializeFlow() {

return new RegisterModel();

}

public void saveUser(RegisterModel registerModel,User user){

registerModel.setUser(user);

}

public void saveBillingAddress(RegisterModel registerModel,Address billing){

registerModel.setBillingAddress(billing);

}

public void storeDetail(RegisterModel registerModel){

User user = registerModel.getUser();

// save the user

userDAO.addUser(user);

Address billing = registerModel.getBillingAddress();

// set the user

billing.setUser(user);

// save the billing address

userDAO.addUserAddress(billing);

// if shipping is not same as billing

//if(!billing.isShipping()) {

//Address shipping = registerModel.getShipping();

// set the user for shipping

//shipping.setUser(user);

// save the shipping address

//userDAO.addUserAddress(shipping);

//}

// if user is not supplier

if(user.getUserRole().equals("CUSTOMER")) {

Cart cart = new Cart();

// set the user

cart.setUser(user);

// save the cart

userDAO.addUserCart(cart);

}

}

}

1. Open the register.jsp and replace the <form:form> tag with

<form:form modelAttribute="user">

and Register button tag with

<input type="submit" name="\_eventId\_submit" value="Register" class="btn btn-primary btn-md"/>

1. Open the billing.jsp and replace the <form:form> tag with

<form:form modelAttribute="address">

and Register button tag with

<input type="submit" name="\_eventId\_submit" value="Register" class="btn btn-primary btn-md"/>

1. Open the preRegister.jsp and replace the <form:form> tag with

<form:form modelAttribute="registerModel">

and Continue and Edit button tag with

<input type="submit" name="\_eventId\_submit" value="Continue" class="btn btn-primary btn-md">

<input type="submit" name="\_eventId\_edit" value="Edit" class="btn btn-primary btn-md">

1. Open IUserDAO interface in backend project and add following code:

public boolean addUserAddress(Address address);

public boolean addUserCart(Cart cart);

1. Open UserDAOImpl class add following code:

public boolean addUserAddress(Address address){

try {

sessionFactory.getCurrentSession().save(address);

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

public boolean addUserCart(Cart cart){

try {

sessionFactory.getCurrentSession().save(cart);

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

1. Open User model class in backend project and add following two attributes

@OneToMany(cascade=CascadeType.***ALL***)

List<Address> address=**new** ArrayList<>();

@OneToOne(cascade=CascadeType.***ALL***)

Cart cart;

1. Add getter and setter code of above said attributes.
2. Open Address model class and add following attribute and add respective getter and setter methods.

@ManyToOne(cascade=CascadeType.***ALL***)

User user;

1. Create Cart model class and add following code:

**package** com.health.HealthMedicineQuestBackEnd.model;

**import** java.io.Serializable;

**import** javax.persistence.CascadeType;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.OneToOne;

@Entity

**public** **class** Cart **implements** Serializable{

**private** **static** **final** **long** ***serialVersionUID*** = 6711662177389182695L;

@Id

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

**int** cartId;

**int** itemCounts;

**int** grandTotal;

@OneToOne(cascade=CascadeType.***ALL***, mappedBy="user")

User user;

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

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

}

**public** Cart() {

**super**();

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

}

**public** Cart(**int** cartId, **int** itemCounts, **int** grandTotal) {

**super**();

**this**.cartId = cartId;

**this**.itemCounts = itemCounts;

**this**.grandTotal = grandTotal;

}

**public** **int** getCartId() {

**return** cartId;

}

**public** **void** setCartId(**int** cartId) {

**this**.cartId = cartId;

}

**public** **int** getItemCounts() {

**return** itemCounts;

}

**public** **void** setItemCounts(**int** itemCounts) {

**this**.itemCounts = itemCounts;

}

**public** **int** getGrandTotal() {

**return** grandTotal;

}

**public** **void** setGrandTotal(**int** grandTotal) {

**this**.grandTotal = grandTotal;

}

**public** User getUser() {

**return** user;

}

**public** **void** setUser(User user) {

**this**.user = user;

}

}