This project uses Spring Boot for Model and Controller Implementation Availaible apis are -

* /shoe (CRUD)
* /purchaseReport (CRUD)
* shoe/all
* /purchaseReport/(category|all|dop)

Current Implementation relies simply on String for storing order list.

It can be extended to utilize many-to-many relationship b/w Shoe and PurchaseReport Entities.

Go Import -> Existing Maven Project.

Select the springtarter subfolder.

Edit application.properties.

Go to src/main/resources/application.properties Add the following:

spring.jpa.hibernate.ddl-auto=update

spring.datasource.url=jdbc:mysql://${MYSQL\_HOST:localhost}:3306/sporty\_shoes

spring.datasource.username=root

spring.datasource.password=Vipul@4567

logging.level.org.springframework.web: DEBUG

spring.mvc.view.prefix=/WEB-INF/jsp/

spring.mvc.view.suffix=.jsp

server.port=8090

Add some extra dependencies to pom.xml file

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>org.teksystems.sportyshoes</groupId>

<artifactId>sporty\_shoes</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>war</packaging>

<name>sporty\_shoes</name>

<description>Sporty Shoes is a company that manufactures and sells sports shoes.</description>

<properties>

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

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

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.16</version>

</dependency>

<dependency>

<groupId>commons-logging</groupId>

<artifactId>commons-logging</artifactId>

<version>1.2</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>4.0.1</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>5.4.27.Final</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>5.3.16</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.28</version>

</dependency>

<dependency>

<groupId>org.glassfish.web</groupId>

<artifactId>javax.servlet.jsp.jstl</artifactId>

<version>1.2.1</version>

</dependency>

<dependency>

<groupId>javax.servlet.jsp.jstl</groupId>

<artifactId>javax.servlet.jsp.jstl-api</artifactId>

<version>1.2.1</version>

</dependency>

</dependencies>

</project>

You may need to use eclipse to "Reload Maven" after changing the pom.xml

Create Entity Class: User

Create a new class User. It can be under the com.example.UserManager package.

It can look like this:

package com.example.UserManager.entities;

import javax.persistence.Entity;

import javax.persistence.Id;

@Entity

public class User {

@Id

private int id;

private String email;

private String name;

private String password;

}

Use your IDE of choice to generate setters and getters (Eclipse: Source -> Generate Setters and Getters). The result should look like this:

|  |
| --- |
|  |
| import java.io.Serializable; |
|  |
| import javax.persistence.Entity; |
| import javax.persistence.GeneratedValue; |
| import javax.persistence.GenerationType; |
| import javax.persistence.Id; |
| import javax.persistence.Table; |
|  |
| @Entity |
|  |
| @Table(name="user") |
| public class User implements Serializable { |
|  |
| private static final long serialVersionUID = 1L; |
|  |
| @Id |
| @GeneratedValue(strategy=GenerationType.AUTO) |
| private int userId; |
| private String userName; |
| private String userEmail; |
| private int userAge; |
| private String userGender; |
| private String userAddress; |
|  |
| public User() { |
| super(); |
| } |
|  |
| public User(String userName, String userEmail, int userAge, String userGender, String userAddress) { |
| super(); |
| this.userName = userName; |
| this.userEmail = userEmail; |
| this.userAge = userAge; |
| this.userGender = userGender; |
| this.userAddress = userAddress; |
| } |
|  |
| 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 getUserEmail() { |
| return userEmail; |
| } |
|  |
| public void setUserEmail(String userEmail) { |
| this.userEmail = userEmail; |
| } |
|  |
| public int getUserAge() { |
| return userAge; |
| } |
|  |
| public void setUserAge(int userAge) { |
| this.userAge = userAge; |
| } |
|  |
| public String getUserGender() { |
| return userGender; |
| } |
|  |
| public void setUserGender(String userGender) { |
| this.userGender = userGender; |
| } |
|  |
| public String getUserAddress() { |
| return userAddress; |
| } |
|  |
| public void setUserAddress(String userAddress) { |
| this.userAddress = userAddress; |
| } |
|  |
| public static long getSerialversionuid() { |
| return serialVersionUID; |
| } |
|  |
| @Override |
| public String toString() { |
| return String.format("User [userId=%s, userName=%s, userEmail=%s, userAge=%s, userGender=%s, userAddress=%s]", |
| userId, userName, userEmail, userAge, userGender, userAddress); |
| } |
| } |

Create A Repository Interface.

Create a new Java Interface called UserRepository.

Paste the following code:

package com.example.UserManager.repositories;

import org.springframework.data.repository.CrudRepository;

import com.example.UserManager.entities.User;

public interface UserRepository extends CrudRepository<User, Integer> {

}

Create a User Service

Create a new class called UserService

Paste the following Code:

package com.example.UserManager.services;

import java.util.Optional;

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

import org.springframework.stereotype.Service;

import com.example.UserManager.entities.User;

import com.example.UserManager.repositories.UserRepository;

@Service

public class UserService {

@Autowired

private UserRepository userRepository;

public Iterable<User> GetAllUsers()

{

return userRepository.findAll();

}

}

Create UserController

Create a UserController.java under the com.example.UserManager.controllers package.

package com.example.UserManager.controllers;

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

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

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

import org.springframework.web.bind.annotation.RequestMethod;

import com.example.UserManager.entities.User;

import com.example.UserManager.services.UserService;

@Controller

public class UserController {

@Autowired

private UserService userService;

@RequestMapping(value="/userquery", method = RequestMethod.GET)

public String showUsers()

{

return("userquery");

}

}

Create UserQuery View

Create a new JSP file in the folder main/webapp/WEB-INF/jsp/userquery.jsp. You will have to create all those folders yourself in order place the new view in that folder.

Here are the contents of userquery.jsp:

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix ="c" %>

<html>

<head>

<style>

table, th, td {

border: 1px solid black;

}

</style>

</head>

<body>

<h2>User Query</h2>

<form action="userquery" method="post">

<label for="uid">User Id:</label><br>

<input type="text" id="uid" name="uid"><br><br>

<input type="submit" value="Submit">

</form>

</body>

</html>

Create New Methods in UserService:

Add the import as folows:

import java.util.Optional;

Add new methods as follows:

public User GetUserById(int id)

{

//TODO: Handle non-existent user. Currently throws an exception

Optional<User> user = userRepository.findById(id);

return user.get();

}

public User EditUser(User user) {

//TODO: Not doing any error checking

return userRepository.save(user);

}

Handle post event in UserController:

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

public String handleQuery(

@RequestParam(value = "uid", required = true) Integer uid,

ModelMap model) {

//TODO: Handle empty or invalid UserId

User user = userService.GetUserById(uid);

model.addAttribute("user", user);

return("useredit");

}

Create UserEdit form

Create a useredit.jsp page as follows:

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix ="c" %>

<html>

<head>

<style>

table, th, td {

border: 1px solid black;

}

</style>

</head>

<body>

<h2>User Edit</h2>

<p><a href="userquery">Return to Query Page</a></p>

<form action="useredit" method="post">

<label for="uid">User Id:</label><br>

<input type="text" id="uid" name="uid" value="${user.id}"><br>

<label for="email">Email:</label><br>

<input type="text" id="email" name="email" value="${user.email}"><br>

<label for="username">Name:</label><br>

<input type="text" id="username" name="username" value="${user.name}"><br>

<label for="password">Password:</label><br>

<input type="text" id="password" name="password" value="${user.password}"><br>

<br>

<input type="submit" value="Submit">

</form>

</body>

</html>