package com.cg.ProductSpringBoot;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication

@ComponentScan("com.cg.ProductSpringBoot")

public class ProductSpringBootApplication {

public static void main(String[] args) {

SpringApplication.run(ProductSpringBootApplication.class, args);

System.out.println("Welcome to spring Boot");

}

}

Product controller

package com.cg.ProductSpringBoot.controller;

import java.util.List;

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

import org.springframework.data.jpa.repository.Query;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

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

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

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

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

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

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

import com.cg.ProductSpringBoot.dto.Inventory;

import com.cg.ProductSpringBoot.dto.Product;

import com.cg.ProductSpringBoot.services.ProductService;

@RestController

@RequestMapping("/product")

public class ProductController {

@Autowired

ProductService productservice;

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

public ResponseEntity<Product> addProduct(@RequestBody Product pro) { //for handle the exception we r using ResponseEntity

Product prod= productservice.addProduct(pro);

if(prod==null) {

return new ResponseEntity("Product not added",HttpStatus.NOT\_FOUND);

}

return new ResponseEntity<Product>(prod,HttpStatus.OK);

}

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

public ResponseEntity<List<Product>> showAllProduct(){

List<Product> myList = productservice.showAll();

if(myList.isEmpty()) {

return new ResponseEntity("No Product to show" ,HttpStatus.NOT\_FOUND);

}

return new ResponseEntity<List<Product>>(myList,HttpStatus.OK);

}

@RequestMapping(value="/search/{name}",method=RequestMethod.GET)

public ResponseEntity<List<Product>> searchByName(@PathVariable("name")String name){

List<Product> listofProduct = productservice.searchByName(name);

if(listofProduct.isEmpty()) {

return new ResponseEntity("No Product name found",HttpStatus.NOT\_FOUND);

}

return new ResponseEntity<List<Product>>(listofProduct,HttpStatus.OK);

}

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

public ResponseEntity<List<Product>> findPrice(@RequestParam("low")Double low,@RequestParam("high")Double high){

List<Product> listofProducts = productservice.findByPrice(low,high);

if(listofProducts.isEmpty()) {

return new ResponseEntity("No Product name found",HttpStatus.NOT\_FOUND);

}

return new ResponseEntity<List<Product>>(listofProducts,HttpStatus.OK);

}

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

public ResponseEntity<Product> addAll(@ModelAttribute Product pro){

Product prod =productservice.addProduct(pro);

if(pro==null) {

return new ResponseEntity("No Prodcut added",HttpStatus.NOT\_FOUND);

}

return new ResponseEntity<Product>(pro,HttpStatus.OK);

}

}

/\*

@RequestMapping(value="/delete/{id}", method =RequestMethod.DELETE)

public void deleteProduct(@PathVariable("id")int id) {

productservice.delete(id);

System.out.println("Deleted Successfully");

}

@RequestMapping(value= "/search/{pid}", method= RequestMethod.GET)

public Product getProductById(@PathVariable("pid") int id) {

return productservice.searchById(id);

}

@RequestMapping(value="/update/{pid}/{pprice}",method=RequestMethod.PUT)

public Product updateProduct(@PathVariable("pid")int id,@PathVariable("pprice")double price) {

Product pro1 = productservice.update(id,price);

System.out.println("Updated succesfully");

return pro1;

}

/\* @RequestParam("name")String name,

@RequestParam("price")double price,

@RequestParam("desc")String desc,

@RequestParam("inid")int inid,

@RequestParam("inname")String inname){

Inventory inventory = new Inventory();

inventory.setId(inid);

inventory.setName(inname);

Product pro = new Product();

pro.setId(pid);

pro.setName(name);

pro.setPrice(price);

pro.setDescription(desc);

pro.setInventory(inventory);\*/

/\*@RequestMapping(value="/checkname/{uname}",method=RequestMethod.GET) //in the postman we are giving in get

public String getName(@PathVariable("uname") String mname, //@pathvarible to print the string in the url

@RequestParam("prodid") String id) { //@RequestParam print the value ? in url

System.out.println("hiiii");

return id+"Capgemini" +mname;

}

@RequestMapping(method=RequestMethod.POST,value="/checkname") //in postman we are changing in post method

public String getData(@RequestParam("prodId") int pid, //these prodId ,prodname all in the postman body key

@RequestParam("prodname") String pname ,

@RequestParam("prodprice") double pprice) { // we are setting the value in the body like(key,value) in postman we are printing in the console

System.out.println(pid+" "+pname+" "+pprice); //printing in console

return"Welcome";

}\*/

Dao interface

package com.cg.ProductSpringBoot.dao;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

import com.cg.ProductSpringBoot.dto.Product;

public interface ProductDaoo extends JpaRepository<Product, Integer> {

public List<Product> findByName(String name);

public List<Product> findBypriceBetween(double low,double high);

}

Dto

package com.cg.ProductSpringBoot.dto;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

@Entity

public class Inventory {

@Id

@Column(name="inventory\_id")

private int id;

@Column(name="inventory\_name")

private String name;

public Inventory() {

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

@Override

public String toString() {

return "Inventory [id=" + id + ", name=" + name + "]";

}

public Inventory(int id, String name) {

super();

this.id = id;

this.name = name;

}

}

package com.cg.ProductSpringBoot.dto;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.OneToOne;

import javax.persistence.Table;

@Entity

@Table(name="data\_product")

public class Product {

@Id

@Column(name="prod\_id")

private int id;

@Column(name="prod\_name")

private String name;

@Column(name="prod\_price")

private double price;

@Column(name="prod\_desc")

private String description;

@OneToOne(cascade=CascadeType.ALL)

@JoinColumn(name="prod\_inventory")

private Inventory inventory;

public Product() {

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

public Inventory getInventory() {

return inventory;

}

public void setInventory(Inventory inventory) {

this.inventory = inventory;

}

@Override

public String toString() {

return "Product [id=" + id + ", name=" + name + ", price=" + price + ", description=" + description

+ ", inventory=" + inventory + "]";

}

public Product(int id, String name, double price, String description, Inventory inventory) {

super();

this.id = id;

this.name = name;

this.price = price;

this.description = description;

this.inventory = inventory;

}

}

**exception**

**package** com.cg.ProductSpringBoot.exception;

**public** **class** ProductException **extends** RuntimeException {

**public** ProductException() {

}

**public** ProductException(String msg) {

**super**(msg);

}

}

package com.cg.ProductSpringBoot.services;

import java.util.List;

import java.util.Optional;

import com.cg.ProductSpringBoot.dto.Product;

public interface ProductService {

public Product addProduct(Product pro);

public List<Product> showAll();

public List<Product> searchByName(String name);

public List<Product> findByPrice(double low,double high );

/\*public void delete(int id);

public Product update(int id, double price);\*/

}

package com.cg.ProductSpringBoot.services;

import java.util.List;

import java.util.Optional;

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

import org.springframework.stereotype.Service;

import com.cg.ProductSpringBoot.dao.ProductDaoo;

import com.cg.ProductSpringBoot.dto.Product;

@Service

public class ProductServiceImpl implements ProductService{

@Autowired

ProductDaoo productdao;

@Override

public Product addProduct(Product pro) {

return productdao.save(pro);

}

@Override

public List<Product> showAll() {

return productdao.findAll();

}

@Override

public List<Product> searchByName(String name) {

return productdao.findByName(name);

}

@Override

public List<Product> findByPrice( double low,double high) {

return productdao.findBypriceBetween(low,high);

}

}

/\*@Override

public List<Product> showAll() {

return productdao.showAll();

}

@Override

public Product searchById(int id) {

return productdao.findById(id);

}

@Override

public Product update(int id ,double price) {

return productdao.update(id,price);

}

@Override

public void delete(int id) {

productdao.delete(id);

}

}\*/

application.properties

server.port=9037

spring.datasource.driver-class-name=com.mysql.jdbc.Driver

spring.datasource.url =jdbc:mysql://localhost/springboot

spring.datasource.username =root

spring.datasource.password =Capgemini123

## Hibernate Properties

# The SQL dialect makes Hibernate generate better SQL for the chosen database

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect

# Hibernate ddl auto (create, create-drop, validate, update)

spring.jpa.hibernate.ddl-auto = update

pom.file

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

<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 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.1.5.RELEASE</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.cg</groupId>

<artifactId>ProductSpringBoot</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>ProductSpringBoot</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>mysql</groupId>

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

<scope>runtime</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>