package com.api.sportyShoes.service.impl;

import java.util.Date;

import java.util.List;

import java.util.NoSuchElementException;

import javax.annotation.PostConstruct;

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

import org.springframework.dao.EmptyResultDataAccessException;

import org.springframework.stereotype.Service;

import com.api.sportyShoes.exceptionHandler.BusinessException;

import com.api.sportyShoes.model.PurchaseReport;

import com.api.sportyShoes.model.Shoe;

import com.api.sportyShoes.repository.PurchaseReportRepository;

import com.api.sportyShoes.repository.ShoesRepository;

import com.api.sportyShoes.service.SportyShoesService;

import lombok.NoArgsConstructor;

@Service

@NoArgsConstructor

public class SportyShoesServiceImpl implements SportyShoesService{

@Autowired

private ShoesRepository shoesRepo;

@Autowired

private PurchaseReportRepository prRepo;

@PostConstruct

public void init() {

Shoe s1 = new Shoe(1,"Shoe Name 1","Basketball",1000.24);

Shoe s2 = new Shoe(2,"Shoe Name 2","Cricket",1100.24);

Shoe s3 = new Shoe(3,"Shoe Name 3","Running",900.24);

Shoe s4 = new Shoe(4,"Shoe Name 4","Football",1900.24);

shoesRepo.save(s1);

shoesRepo.save(s2);

shoesRepo.save(s3);

shoesRepo.save(s4);

Date d = new Date(0);

PurchaseReport pr1 = new PurchaseReport(5,"user\_1","Running",d,"adidas\_runner:5,nike\_airmax:10");

PurchaseReport pr2 = new PurchaseReport(6,"user\_2","Cricket",d,"adidas\_cricket:5,nike\_cricket:10");

PurchaseReport pr3 = new PurchaseReport(7,"user\_3","Basketball",d,"adidas\_basketball:5,nike\_basketball:10");

PurchaseReport pr4 = new PurchaseReport(8,"user\_4","Football",d,"adidas\_football:5,nike\_football:10");

prRepo.save(pr1);

prRepo.save(pr2);

prRepo.save(pr3);

prRepo.save(pr4);

}

public Shoe createShoe(Shoe shoe) throws BusinessException {

int id = shoe.getId();

Shoe oldShoe = null;

try {

oldShoe = shoesRepo.findById(id).get();

}catch(NoSuchElementException e) {

}

if(oldShoe!=null) throw new BusinessException("Shoe already exists with id: "+id);

return shoesRepo.save(shoe);

}

public Shoe getShoeById(int id) throws BusinessException {

Shoe shoe = null;

try {

if(id<=0) throw new BusinessException("Shoe Id can not be negative or zero");

shoe = shoesRepo.findById(id).get();

}catch(NoSuchElementException e) {

throw new BusinessException("Shoe not found with Id: "+id);

}

return shoe;

}

public Shoe updateShoe(Shoe shoe) {

return shoesRepo.save(shoe);

}

public void deleteShoeById(int id) throws BusinessException {

try {

shoesRepo.deleteById(id);

}catch(IllegalArgumentException e) {

throw new BusinessException("Invalid id: "+id);

}catch(EmptyResultDataAccessException e) {

throw new BusinessException("SHoe does not exist with id: "+id);

}

}

public List<Shoe> getAllShoes() {

return shoesRepo.findAll();

}

public PurchaseReport createPurchaseReport(PurchaseReport pr) throws BusinessException {

int id = pr.getId();

PurchaseReport oldPr = null;

try {

oldPr = prRepo.findById(id).get();

}catch(NoSuchElementException e) {

}

if(oldPr!=null) throw new BusinessException("Purchase report already exists with id: "+id);

return prRepo.save(pr);

}

public PurchaseReport getPurchaseReportById(int id) throws BusinessException {

PurchaseReport pr = null;

try {

if(id<=0) throw new BusinessException("Purchase Report Id can not be negative or zero");

pr = prRepo.findById(id).get();

}catch(NoSuchElementException e) {

throw new BusinessException("Purchase Report not found with Id: "+id);

}

return pr;

}

public PurchaseReport updatePurchaseReport(PurchaseReport pr) {

return prRepo.save(pr);

}

public void deletePurchaseReportById(int id) throws BusinessException {

try {

prRepo.deleteById(id);

}catch(IllegalArgumentException e) {

throw new BusinessException("Invalid id: "+id);

}catch(EmptyResultDataAccessException e) {

throw new BusinessException("Puchase Report does not exist with Id: "+id);

}

}

public List<PurchaseReport> getAllPurchaseReports() {

return prRepo.findAll();

}

public List<PurchaseReport> getAllPurchaseReportsByCategory(String category) {

return prRepo.findByCategory(category);

}

public List<PurchaseReport> getAllPurchaseReportsByDOP(Date dop) {

return prRepo.findByDop(dop);

}

}