**VERIFICA FINALE**

**UF: Programmazione – Java Enterprise Edition**

Docente: Alessandro Fiore

1) Creare un progetto spring boot

package it.its.configuration;

import org.springframework.context.MessageSource;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.validation.beanvalidation.LocalValidatorFactoryBean;

@Configuration

public class AppConfiguration {

@Bean

public LocalValidatorFactoryBean validator(MessageSource messageSource) {

LocalValidatorFactoryBean validatorFactoryBean = new LocalValidatorFactoryBean();

validatorFactoryBean.setValidationMessageSource(messageSource);

return validatorFactoryBean;

}

}

2) Importare le dipendenze necessarie per implementare un’applicazione di API REST e che utilizzi POSTGRES come database

dependencies {

implementation 'org.springframework.boot:spring-boot-starter-web'

implementation 'org.springframework.boot:spring-boot-starter-validation'

implementation 'org.springframework.boot:spring-boot-starter-data-jpa'

implementation group: 'org.postgresql', name: 'postgresql'

developmentOnly 'org.springframework.boot:spring-boot-devtools'

testImplementation 'org.springframework.boot:spring-boot-starter-test'

}

3) Modellare il seguente dominio di dati utilizzando le Entity Class: La compagnia aerea Farfalla spa possiede diversi Aerei (id,targa, modello, capienza, nome, anno di produzione) che vengono fabbricati da uno ed un solo Produttore (id, nome, nazione, sede legale)

package it.its.domain;

import java.util.List;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.ManyToOne;

import javax.persistence.OneToMany;

@Entity

public class Aereo {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

int id;

String targa;

String modello;

int capienza;

String nome;

int annoDiProduzione;

@ManyToOne

Produttore produttore;

@OneToMany(mappedBy = "aereo")

List<AereoTratta> aereoTratta;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getTarga() {

return targa;

}

public void setTarga(String targa) {

this.targa = targa;

}

public String getModello() {

return modello;

}

public void setModello(String modello) {

this.modello = modello;

}

public int getCapienza() {

return capienza;

}

public void setCapienza(int capienza) {

this.capienza = capienza;

}

public String getNome() {

return nome;

}

public void setNome(String nome) {

this.nome = nome;

}

public int getAnnoDiProduzione() {

return annoDiProduzione;

}

public void setAnnoDiProduzione(int annoDiProduzione) {

this.annoDiProduzione = annoDiProduzione;

}

public Produttore getProduttore() {

return produttore;

}

public void setProduttore(Produttore produttore) {

this.produttore = produttore;

}

}

4) Creare i repository per Aereo e Produttore

package it.its.repository;

import java.util.List;

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

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

import org.springframework.data.repository.query.Param;

import org.springframework.stereotype.Repository;

import it.its.domain.Aereo;

@Repository

public interface AereoRepository extends JpaRepository<Aereo, Integer>{

@Query("from Aereo as a where a.nome = :Nome")

public List<Aereo> getByName(@Param("Nome") String nome);

public List<Aereo> findByannoDiProduzione(int data);

@Query("from Aereo as a where a.produttore.id = :Id")

public List<Aereo> getAereoByIdProduttore(@Param("Id") int Id);

}

5) Estendere il repository Aereo per permettere la ricerca tramite nome

@Query("from Aereo as a where a.nome = :Nome")

public List<Aereo> getByName(@Param("Nome") String nome);

6) Estendere il repository Aereo per permettere la ricerca tramite anno di produzione

public List<Aereo> findByannoDiProduzione(int data);

7) Estendere il repository Aereo per recuperare la lista di tutti gli aerei di uno specifico produttore avendo l’id del produttore

@Query("from Aereo as a where a.produttore.id = :Id")

public List<Aereo> getAereoByIdProduttore(@Param("Id") int Id);

8) Implementare i seguenti DTO:

a) Produttore (id, nome, nazione, sede legale)

b) Aereo (id,targa, modello, capienza, nome, anno di produzione, id produttore)

package it.its.DTO;

import javax.validation.constraints.NotBlank;

import javax.validation.constraints.NotNull;

import org.hibernate.validator.constraints.Range;

import it.its.validators.AnnoCapienzaCustomConstraint;

@AnnoCapienzaCustomConstraint(annoDiProduzione = "annoDiProduzione", capienza = "capienza")

public class AereoDTO {

int id;

@NotBlank

String targa;

@NotBlank

String modello;

@NotNull

int capienza;

@NotBlank

String nome;

@Range(min = 1980, max = 2022, message="ERRORE")

int annoDiProduzione;

@NotNull

int idProduttore;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getTarga() {

return targa;

}

public void setTarga(String targa) {

this.targa = targa;

}

public String getModello() {

return modello;

}

public void setModello(String modello) {

this.modello = modello;

}

public int getCapienza() {

return capienza;

}

public void setCapienza(int capienza) {

this.capienza = capienza;

}

public String getNome() {

return nome;

}

public void setNome(String nome) {

this.nome = nome;

}

public int getAnnoDiProduzione() {

return annoDiProduzione;

}

public void setAnnoDiProduzione(int annoDiProduzione) {

this.annoDiProduzione = annoDiProduzione;

}

public int getIdProduttore() {

return idProduttore;

}

public void setIdProduttore(int idProduttore) {

this.idProduttore = idProduttore;

}

}

9) Creare interfacce e implementazioni dei Service per Aereo e Produttore affinchè siano disponibili le seguenti transazioni:

a) Produttore:

i) Salvataggio di un nuovo produttore

ii) Lista di tutti i produttori disponibili

iii) Recupero di uno specifico produttore dato un id altrimenti viene lanciata un’eccezione custom

b) Aereo

i) Salvataggio di un nuovo aereo

ii) Recupero di uno specifico aereo dato un id

iii) Recupero degli aerei in uno specifico anno di produzione

iv) Recupero di tutti gli aerei dato l’id di uno specifico produttore

v) Eliminazione di un aereo tramite id altrimenti viene lanciata un’eccezione custom

@Transactional(rollbackOn = EntityNotFoundException.class)

@Override

public AereoDTO save(AereoDTO aereoDTO) {

Aereo aereo = aereoRepository.save(Conversion.fromAereoToAereoDTO(aereoDTO));

Produttore produttore = produttoreRepository.findById(aereoDTO.getIdProduttore()).orElseThrow(()->new EntityNotFoundException());

aereo.setProduttore(produttore);

aereoDTO.setId(aereoRepository.save(aereo).getId());

return aereoDTO;

}

@Transactional

@Override

public List<AereoDTO> getByNome(String nome) {

// TODO Auto-generated method stub

return Conversion.aereoDTOListFromAereoList(aereoRepository.getByName(nome));

}

@Transactional

@Override

public List<AereoDTO> getByAnnoDiProduzione(int data) {

// TODO Auto-generated method stub

return Conversion.aereoDTOListFromAereoList(aereoRepository.findByannoDiProduzione(data));

}

@Transactional

@Override

public List<AereoDTO> getAereoByIdProduttore(int id) {

// TODO Auto-generated method stub

return Conversion.aereoDTOListFromAereoList(aereoRepository.getAereoByIdProduttore(id));

}

@Transactional(rollbackOn = AereoNotFoundException.class)

@Override

public void delete(int id) throws AereoNotFoundException {

Aereo aereo = aereoRepository.findById(id).orElseThrow(()-> new AereoNotFoundException());

aereoRepository.delete(aereo);

}

}

10) Implementare i RestController con i seguenti request handler method per ogni risorsa:

a) Produttore

i) Salvataggio di un nuovo produttore

ii) Lista di tutti i produttori disponibili

iii) Recupero di uno specifico produttore dato un id altrimenti viene lanciato un ResponseCode 404

b) Aereo

i) Salvataggio di un nuovo aereo

ii) Recupero di uno specifico aereo dato un id

iii) Recupero degli aerei in uno specifico anno di produzione

iv) Recupero di tutti gli aerei dato l’id di uno specifico produttore

v) Eliminazione di un aereo tramite id altrimenti viene lanciato un ResponseCode 404

package it.its.rest.controller;

import java.util.ArrayList;

import java.util.List;

import javax.validation.Valid;

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

import org.springframework.http.MediaType;

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.RestController;

import it.its.DTO.AereoDTO;

import it.its.Iservice.IServiceAereo;

import it.its.exception.AereoNotFoundException;

@RestController

@RequestMapping("/api/aereo")

public class AereoRestController {

@Autowired

IServiceAereo iserviceAereo;

@RequestMapping(value="/save", method=RequestMethod.POST, consumes = MediaType.APPLICATION\_JSON\_VALUE )

public AereoDTO save(@RequestBody @Valid AereoDTO aereoDTO) {

return iserviceAereo.save(aereoDTO);

}

@RequestMapping(value="/getByNome/{nome}", method = RequestMethod.GET)

public List<AereoDTO> getByName(@PathVariable("nome") String nome){

return iserviceAereo.getByNome(nome);

}

@RequestMapping(value="/getByIdProduttore/{id}", method = RequestMethod.GET)

public List<AereoDTO> getByName(@PathVariable("id") int id){

return iserviceAereo.getAereoByIdProduttore(id);

}

@RequestMapping(value="/getByAnnoDiProduzione/{data}", method = RequestMethod.GET)

public List<AereoDTO> getByAnnoDiProduzione(@PathVariable("data") int data){

return new ArrayList<AereoDTO>(iserviceAereo.getByAnnoDiProduzione(data));

}

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

public void deleta(@PathVariable("id") int id) throws AereoNotFoundException {

iserviceAereo.delete(id);

}

}

11) Implementare le seguenti validazioni per ogni DTO:

a) Produttore

i) Tutti i campi escluso ID non devono essere null

b) Aereo

i) Tutti i campi escluso ID non devono essere null

ii) L’anno di produzione deve essere compreso tra il 1980 e il 2022

iii) CrossField Validation: se l’anno è il 1985 e la capienza è inferiore a 69 la validazione fallisce.

package it.its.validators;

import javax.validation.ConstraintValidator;

import javax.validation.ConstraintValidatorContext;

import org.springframework.beans.BeanWrapper;

import org.springframework.beans.BeanWrapperImpl;

public class AnnoCapienzaValidator implements ConstraintValidator<AnnoCapienzaCustomConstraint, Object>{

String annoDiProduzione;

String capienza;

public void initialize(AnnoCapienzaCustomConstraint constraintAnnotation) {

// TODO Auto-generated method stub

this.annoDiProduzione = constraintAnnotation.annoDiProduzione();

this.capienza = constraintAnnotation.capienza();

}

@Override

public boolean isValid(Object value, ConstraintValidatorContext context) {

BeanWrapper beanWrapper = new BeanWrapperImpl(value);

int anno = (Integer)beanWrapper.getPropertyValue(annoDiProduzione);

int capienza2 = (Integer)beanWrapper.getPropertyValue(capienza);

if(anno==1985 && capienza2<69) {

return false;

}

return true;

}

}

12) Modellare il seguente dominio di dati utilizzando le Entity Class: una TRATTA (id, aeroporto partenza, aeroporto arrivo, km di percorrenza) è percorsa da più AEREI (giorno settimana) e lo stesso AEREO può percorrere più tratte.

@Entity

public class Tratta {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

int id;

String aereoportoPartenza;

String aereoportoArrivo;

int kmPercorsi;

@OneToMany(mappedBy = "tratta")

List<AereoTratta> aereoTratta;

13) Creare il repository per TRATTA e per la classe di supporto alla relazione M:N

@Entity

public class AereoTratta {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

int id;

@ManyToOne

Aereo aereo;

@ManyToOne

Tratta tratta;

String giornoDellaSettimana;

14) Definire un service che gestisca la transazione di inserimento. Ricevendo il seguente DTO( idtratta, aeroporto partenza, aeroporto arrivo, giorno della settimana, id aereo) deve implementare la seguente logica:

a) Se idTratta non è avvalorato e idAereo esiste allora crea una nuova tratta e provvede alla creazione della tabella

b) Se idTratta è avvalorato con un id esistente e idaereo contiene un id esistente allora non provvede alla creazione di una nuova tratta e si limita a creare il nuovo oggetto nella tabella di appoggio

c) Se idTratta non esiste o idAereo non esiste lancia un’eccezione EntityNotFoundException.

package it.its.Iservice.impl;

import javax.persistence.EntityNotFoundException;

import javax.transaction.Transactional;

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

import org.springframework.stereotype.Service;

import it.its.DTO.AereoTrattaDTO;

import it.its.Iservice.IServiceTrattaAereo;

import it.its.domain.Aereo;

import it.its.domain.AereoTratta;

import it.its.domain.Tratta;

import it.its.exception.AereoNotFoundException;

import it.its.repository.AereoRepository;

import it.its.repository.AereoTrattaRepository;

import it.its.repository.TrattaRepository;

@Service

public class AereoTrattaImpl implements IServiceTrattaAereo{

@Autowired

AereoRepository aereoRepository;

@Autowired

TrattaRepository trattaRepository;

@Autowired

AereoTrattaRepository aereoTrattaRepository;

@Transactional

@Override

public AereoTrattaDTO save(AereoTrattaDTO aereoTrattaDTO) throws AereoNotFoundException, EntityNotFoundException{

Tratta tratta = null;

if(aereoTrattaDTO.getIdTratta() != 0) {

tratta = trattaRepository.findById(aereoTrattaDTO.getIdTratta()).orElseThrow(() -> new EntityNotFoundException("tratta non presente"));

}

Aereo aereo = aereoRepository.findById(aereoTrattaDTO.getIdAereo()).orElseThrow(() -> new AereoNotFoundException());

if (tratta == null) {

tratta = new Tratta();

tratta.setAereoportoPartenza(aereoTrattaDTO.getAereoportoPartenza());

tratta.setAereoportoArrivo(aereoTrattaDTO.getAereoportoArrivo());

tratta.setKmPercorsi(aereoTrattaDTO.getKmPercorsi());

tratta.setId(trattaRepository.save(tratta).getId());

aereoTrattaDTO.setIdTratta(tratta.getId());

}

AereoTratta aereoTratta = new AereoTratta();

aereoTratta.setAereo(aereo);

aereoTratta.setTratta(tratta);

aereoTratta.setGiornoDellaSettimana(aereoTrattaDTO.getGiornoDellaSettimana());

aereoTrattaRepository.save(aereoTratta);

return aereoTrattaDTO;

}

}

15) Chiudi Eclipse

Studente/ssa (nome e firma)

BARI, 08/06/2022