Spring Boot with JPA

Create an order API with the following dependencies

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

Create a Model for Order

**package** com.ibm.model;

**import** java.time.LocalDateTime;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

**import** com.fasterxml.jackson.annotation.JsonFormat;

**import** com.fasterxml.jackson.annotation.JsonProperty;

**import** com.fasterxml.jackson.databind.annotation.JsonSerialize;

**import** com.fasterxml.jackson.datatype.jsr310.ser.LocalDateTimeSerializer;

**import** lombok.AllArgsConstructor;

**import** lombok.Data;

**import** lombok.EqualsAndHashCode;

**import** lombok.NoArgsConstructor;

**import** lombok.ToString;

@Data

@NoArgsConstructor

@AllArgsConstructor

@ToString

@EqualsAndHashCode

@Entity

@Table(name ="orders")

**public** **class** Order {

@Id

@GeneratedValue

**private** **long** orderId;

**private** String orderName;

**private** Float orderPrice;

@JsonSerialize(using = LocalDateTimeSerializer.**class**)

@JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss")

**private** LocalDateTime orderDateTime;

}

Create a Order repository

**package** com.ibm.repository;

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

**import** org.springframework.stereotype.Repository;

**import** com.ibm.model.Order;

@Repository

**public** **interface** OrdersRepository **extends** JpaRepository<Order, Long>{

}

Application class is updated

**package** com.ibm;

**import** java.time.LocalDateTime;

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

**import** org.springframework.boot.CommandLineRunner;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** com.ibm.model.Order;

**import** com.ibm.repository.OrdersRepository;

@SpringBootApplication

**public** **class** OrderapiApplication **implements** CommandLineRunner{

@Autowired

OrdersRepository orderRepo;

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(OrderapiApplication.**class**, args);

}

@Override

**public** **void** run(String... args) **throws** Exception {

Order o = **new** Order();

o.setOrderDateTime(LocalDateTime.*now*());

o.setOrderName("Order-0");

o.setOrderPrice(500.00f);

Order result = orderRepo.save(o);

System.***out***.println(result.toString());

}

}

application.properties

server.port=8080

spring.jpa.show-sql=true

spring.h2.console.enabled=true

spring.application.name=Order-API

logging.level.org.springframework.web=trace

spring.datasource.url=jdbc:h2:mem:ordersdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.mvc.format.date=yyyy-MM-dd

OrdersService.java

**package** com.ibm.service;

**import** java.util.List;

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

**import** org.springframework.stereotype.Service;

**import** com.ibm.model.Order;

**import** com.ibm.repository.OrdersRepository;

@Service

**public** **class** OrdersService {

@Autowired

**private** OrdersRepository ordersRepository;

**public** List<Order> getOrders()

{

**return** ordersRepository.findAll();

}

**public** Order getOrderById(**long** orderId)

{

**return** ordersRepository.findById(orderId).orElseThrow();

}

**public** Order createOrder(Order order)

{

**return** ordersRepository.saveAndFlush(order);

}

// getOrderById()

// getOrderByName()

// createOrder()

// updateOrder()

// deleteOrder()

}

OrdersController.java

**package** com.ibm.controller;

**import** java.util.List;

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

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.PostMapping;

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

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

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

**import** com.ibm.model.Order;

**import** com.ibm.service.OrdersService;

@RestController

@RequestMapping("/orders")

**public** **class** OrdersController {

@Autowired

**private** OrdersService ordersService;

@GetMapping

**public** ResponseEntity<List<Order>> getOrders()

{

**return** ResponseEntity.*ok*(ordersService.getOrders());

}

@GetMapping("/{id}")

**public** ResponseEntity<Order> getOrderById(@PathVariable **long** id)

{

**return** ResponseEntity.*ok*(ordersService.getOrderById(id));

}

@PostMapping

**public** ResponseEntity<Order> createOrder(@RequestBody Order order)

{

**return** ResponseEntity.*ok*(ordersService.createOrder(order));

}

}

Updated with additional methods

OrdersRepository.java

**package** com.ibm.repository;

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

**import** org.springframework.stereotype.Repository;

**import** com.ibm.model.Order;

**import** java.util.List;

@Repository

**public** **interface** OrdersRepository **extends** JpaRepository<Order, Long>{

**public** List<Order> findByOrderName(String orderName);

}

OrdersService.java

**package** com.ibm.service;

**import** java.util.List;

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

**import** org.springframework.stereotype.Service;

**import** com.ibm.model.Order;

**import** com.ibm.repository.OrdersRepository;

@Service

**public** **class** OrdersService {

@Autowired

**private** OrdersRepository ordersRepository;

**public** List<Order> getOrders() {

**return** ordersRepository.findAll();

}

**public** Order getOrderById(**long** orderId) {

**return** ordersRepository.findById(orderId).orElseThrow();

}

**public** List<Order> getOrdersByName(String orderName) {

**return** ordersRepository.findByOrderName(orderName);

}

**public** Order createOrder(Order order) {

**return** ordersRepository.saveAndFlush(order);

}

**public** Order updateOrder(Order updatedOrder) {

getOrderById(updatedOrder.getOrderId());

**return** ordersRepository.saveAndFlush(updatedOrder);

}

**public** **void** deleteOrder(**long** orderId) {

getOrderById(orderId);

ordersRepository.deleteById(orderId);

}

}

OrderController.java

**public** ResponseEntity<List<Order>> getOrders()

{

**return** ResponseEntity.*ok*(ordersService.getOrders());

}

@GetMapping("/{id}")

**public** ResponseEntity<Order> getOrderById(@PathVariable **long** id)

{

**return** ResponseEntity.*ok*(ordersService.getOrderById(id));

}

@GetMapping("name/{name}")

**public** ResponseEntity<List<Order>> getOrdersByName(@PathVariable String name)

{

**return** ResponseEntity.*ok*(ordersService.getOrdersByName(name));

}

@PostMapping

**public** ResponseEntity<Order> createOrder(@RequestBody Order order)

{

**return** ResponseEntity.*ok*(ordersService.createOrder(order));

}

}

Updated with REST standards

**package** com.ibm.controller;

**import** java.net.URI;

**import** java.util.List;

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

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.DeleteMapping;

**import** org.springframework.web.bind.annotation.GetMapping;

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

**import** org.springframework.web.bind.annotation.PostMapping;

**import** org.springframework.web.bind.annotation.PutMapping;

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

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

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

**import** org.springframework.web.servlet.support.ServletUriComponentsBuilder;

**import** com.ibm.model.Order;

**import** com.ibm.service.OrdersService;

@RestController

@RequestMapping("/orders")

**public** **class** OrdersController {

@Autowired

**private** OrdersService ordersService;

@GetMapping

**public** ResponseEntity<List<Order>> getOrders()

{

**return** ResponseEntity.*ok*(ordersService.getOrders());

}

@GetMapping("/{id}")

**public** ResponseEntity<Order> getOrderById(@PathVariable **long** id)

{

**return** ResponseEntity.*of*(ordersService.getOrderById(id));

}

@GetMapping("name/{name}")

**public** ResponseEntity<List<Order>> getOrdersByName(@PathVariable String name)

{

**return** ResponseEntity.*ok*(ordersService.getOrdersByName(name));

}

@PostMapping

**public** ResponseEntity<Order> createOrder(@RequestBody Order order)

{

Order createdOrder = ordersService.createOrder(order);

URI location = ServletUriComponentsBuilder.*fromCurrentRequest*().path("/{id}")

.buildAndExpand(order.getOrderId()).toUri();

**return** ResponseEntity.*created*(location).body(createdOrder);

}

@PutMapping("/{id}")

**public** ResponseEntity<Order> updateOrder(@PathVariable **long** id,@RequestBody Order order)

{

**if**(id!=order.getOrderId())

**throw** **new** RuntimeException();

**return** ResponseEntity.*ok*(ordersService.updateOrder(order));

}

@DeleteMapping("/{id}")

**public** ResponseEntity<Order> createOrder(@PathVariable **long** id)

{

ordersService.deleteOrder(id);

**return** ResponseEntity.*noContent*().build();

}

}

OrderService.java

**package** com.ibm.service;

**import** java.util.List;

**import** java.util.Optional;

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

**import** org.springframework.stereotype.Service;

**import** com.ibm.model.Order;

**import** com.ibm.repository.OrdersRepository;

@Service

**public** **class** OrdersService {

@Autowired

**private** OrdersRepository ordersRepository;

**public** List<Order> getOrders() {

**return** ordersRepository.findAll();

}

**public Optional<Order> getOrderById(long orderId) {**

**return ordersRepository.findById(orderId);**

**}**

**public** List<Order> getOrdersByName(String orderName) {

**return** ordersRepository.findByOrderName(orderName);

}

**public** Order createOrder(Order order) {

**return** ordersRepository.saveAndFlush(order);

}

**public** Order updateOrder(Order updatedOrder) {

getOrderById(updatedOrder.getOrderId());

**return** ordersRepository.saveAndFlush(updatedOrder);

}

**public** **void** deleteOrder(**long** orderId) {

getOrderById(orderId);

ordersRepository.deleteById(orderId);

}

}

Adding exception Handling Capabilities

**package** com.ibm.common.exceptions;

**public** **class** OrderIDMisMatchException **extends** RuntimeException {

}

**package com.ibm.common.exceptions;**

**public class OrderNotFoundException extends RuntimeException {**

**}**

OrderService.java

**package** com.ibm.service;

**import** java.util.List;

**import** java.util.Optional;

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

**import** org.springframework.stereotype.Service;

**import** com.ibm.common.exceptions.OrderNotFoundException;

**import** com.ibm.model.Order;

**import** com.ibm.repository.OrdersRepository;

@Service

**public** **class** OrdersService {

@Autowired

**private** OrdersRepository ordersRepository;

**public** List<Order> getOrders() {

**return** ordersRepository.findAll();

}

**public** Optional<Order> getOrderById(**long** orderId) {

**return** ordersRepository.findById(orderId);

}

**public** Order validateOrderById(**long** orderId) {

**return** ordersRepository.findById(orderId).orElseThrow(OrderNotFoundException::**new**);

}

**public** List<Order> getOrdersByName(String orderName) {

**return** ordersRepository.findByOrderName(orderName);

}

**public** Order createOrder(Order order) {

**return** ordersRepository.saveAndFlush(order);

}

**public** Order updateOrder(Order updatedOrder) {

validateOrderById(updatedOrder.getOrderId());

**return** ordersRepository.saveAndFlush(updatedOrder);

}

**public** **void** deleteOrder(**long** orderId) {

validateOrderById(orderId);

ordersRepository.deleteById(orderId);

}

}

ErrorDetails.java

**package** com.ibm.common.exceptions;

**import** java.time.LocalDateTime;

**import** java.util.List;

**import** org.springframework.validation.FieldError;

**import** com.fasterxml.jackson.annotation.JsonInclude;

**import** com.fasterxml.jackson.annotation.JsonInclude.Include;

**import** lombok.AllArgsConstructor;

**import** lombok.Data;

**import** lombok.NoArgsConstructor;

@NoArgsConstructor

@AllArgsConstructor

@Data

**public** **class** ErrorDetails {

**private** String errorCode;

**private** String errorDetails;

**private** LocalDateTime timestamp;

@JsonInclude(Include.***NON\_NULL***)

**private** List<FieldError> errorFields;

}

OrderControllerAdvice.java

@RestControllerAdvice

**public** **class** OrderControllerAdvice {

@ExceptionHandler(OrderNotFoundException.**class**)

**public** ResponseEntity<ErrorDetails> handleOrderNotFound(OrderNotFoundException e)

{

ErrorDetails ed = **new** ErrorDetails("Order-404", "Order Not Found!", LocalDateTime.*now*(),**null**);

**return** ResponseEntity.*status*(HttpStatus.***NOT\_FOUND***).body(ed);

}

@ExceptionHandler(OrderIDMisMatchException.**class**)

**public** ResponseEntity<ErrorDetails> handleOrderIDMisMatch(OrderIDMisMatchException e)

{

ErrorDetails ed = **new** ErrorDetails("Order-400", "Order ID Mismatch!", LocalDateTime.*now*(),**null**);

**return** ResponseEntity.*status*(HttpStatus.***NOT\_FOUND***).body(ed);

}

@ExceptionHandler(MethodArgumentNotValidException.**class**)

**public** ResponseEntity<ErrorDetails> handleInvalidTodo(MethodArgumentNotValidException e)

{

ErrorDetails ed = **new** ErrorDetails("Order-400", "Order Invalid!", LocalDateTime.*now*(), e.getFieldErrors());

**return** ResponseEntity.*status*(HttpStatus.***BAD\_REQUEST***).body(ed);

}

}

**Validations to be added to Order.java**

package com.ibm.model;

import java.time.LocalDateTime;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

import javax.persistence.Table;

import javax.validation.constraints.FutureOrPresent;

import javax.validation.constraints.NotBlank;

import javax.validation.constraints.Size;

import com.fasterxml.jackson.annotation.JsonFormat;

import com.fasterxml.jackson.databind.annotation.JsonSerialize;

import com.fasterxml.jackson.datatype.jsr310.ser.LocalDateTimeSerializer;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.EqualsAndHashCode;

import lombok.NoArgsConstructor;

import lombok.ToString;

@Data

@NoArgsConstructor

@AllArgsConstructor

@ToString

@EqualsAndHashCode

@Entity

@Table(name ="orders")

public class Order {

@Id

@GeneratedValue

private long orderId;

@NotBlank(message = "Order Name cannot be Empty!")

@Size(min = 3, max = 200, message = "Order Name must be b/w 3-200 Characters!")

private String orderName;

private Float orderPrice;

@JsonSerialize(using = LocalDateTimeSerializer.class)

@JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss")

@FutureOrPresent(message = "Order Date must be Present or Future only!")

private LocalDateTime orderDateTime;

}

OrderController.java

**package** com.ibm.controller;

**import** java.net.URI;

**import** java.util.List;

**import** javax.validation.Valid;

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

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.DeleteMapping;

**import** org.springframework.web.bind.annotation.GetMapping;

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

**import** org.springframework.web.bind.annotation.PostMapping;

**import** org.springframework.web.bind.annotation.PutMapping;

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

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

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

**import** org.springframework.web.servlet.support.ServletUriComponentsBuilder;

**import** com.ibm.common.exceptions.OrderIDMisMatchException;

**import** com.ibm.model.Order;

**import** com.ibm.service.OrdersService;

@RestController

@RequestMapping("/orders")

**public** **class** OrdersController {

@Autowired

**private** OrdersService ordersService;

@GetMapping

**public** ResponseEntity<List<Order>> getOrders()

{

**return** ResponseEntity.*ok*(ordersService.getOrders());

}

@GetMapping("/{id}")

**public** ResponseEntity<Order> getOrderById(@PathVariable **long** id)

{

**return** ResponseEntity.*of*(ordersService.getOrderById(id));

}

@GetMapping("name/{name}")

**public** ResponseEntity<List<Order>> getOrdersByName(@PathVariable String name)

{

**return** ResponseEntity.*ok*(ordersService.getOrdersByName(name));

}

@PostMapping

**public** ResponseEntity<Order> createOrder(@Valid @RequestBody Order order)

{

Order createdOrder = ordersService.createOrder(order);

URI location = ServletUriComponentsBuilder.*fromCurrentRequest*().path("/{id}")

.buildAndExpand(order.getOrderId()).toUri();

**return** ResponseEntity.*created*(location).body(createdOrder);

}

@PutMapping("/{id}")

**public** ResponseEntity<Order> updateOrder(@PathVariable **long** id,@Valid @RequestBody Order order)

{

**if**(id!=order.getOrderId())

**throw** **new** OrderIDMisMatchException();

**return** ResponseEntity.*ok*(ordersService.updateOrder(order));

}

@DeleteMapping("/{id}")

**public** ResponseEntity<Order> createOrder(@PathVariable **long** id)

{

ordersService.deleteOrder(id);

**return** ResponseEntity.*noContent*().build();

}

}

To handle future date

**package** com.ibm;

**import** java.time.LocalDateTime;

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

**import** org.springframework.boot.CommandLineRunner;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** com.ibm.model.Order;

**import** com.ibm.repository.OrdersRepository;

@SpringBootApplication

**public** **class** OrderapiApplication **implements** CommandLineRunner{

@Autowired

**private** OrdersRepository orderRepo;

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(OrderapiApplication.**class**, args);

}

@Override

**public** **void** run(String... args) **throws** Exception {

Order o = **new** Order();

**o.setOrderDateTime(LocalDateTime.*of*(2023, 12, 31, 23, 59));**

o.setOrderName("Order-0");

o.setOrderPrice(500.00f);

Order result = orderRepo.save(o);

System.***out***.println(result.toString());

}

}

Updated Code till now is available here: <https://drive.google.com/file/d/1bcqlslQ1LFgAVzkwY_b1wWxZNIPtxbI-/view?usp=drive_link>

Mongo DB Download: <https://fastdl.mongodb.org/windows/mongodb-windows-x86_64-7.0.2-signed.msi>

Adding Actuator

<dependency>

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

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

</dependency>

application.properties

management.endpoints.web.exposure.include=\*

Adding Spring Docs for OPEN API

<dependency>

<groupId>org.springdoc</groupId>

<artifactId>springdoc-openapi-ui</artifactId>

<version>1.7.0</version>

</dependency>

Go to this link :<http://localhost:8080/swagger-ui/index.html>

CORS: [Getting Started | Enabling Cross Origin Requests for a RESTful Web Service (spring.io)](https://spring.io/guides/gs/rest-service-cors/)

Index.html

<html>

<head>

<script type="text/javascript">

function getOrders(){

var requestOptions = {

method: 'GET',

redirect: 'follow'

};

fetch("http://localhost:8080/orders", requestOptions)

.then(response => response.text())

.then(result => console.log(result))

.catch(error => console.log('error', error));

}

</script>

</head>

<body>

<button onclick="getOrders()">Click</button>

</body>

</html>

On OrderController.java

@CrossOrigin annotation needs to be added

Spring Data REST

**package** com.ibm.repository;

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

**import** org.springframework.data.rest.core.annotation.RepositoryRestResource;

**import** org.springframework.stereotype.Repository;

**import** com.ibm.model.Order;

**import** java.util.List;

@RepositoryRestResource

@Repository

**public** **interface** OrdersRepository **extends** JpaRepository<Order, Long>{

**public** List<Order> findByOrderName(String orderName);

}

With MongoDB

Order.java

**package** com.ibm.model;

**import** java.time.LocalDateTime;

**import** javax.validation.constraints.FutureOrPresent;

**import** javax.validation.constraints.NotBlank;

**import** javax.validation.constraints.Size;

**import** org.springframework.data.annotation.Id;

**import** org.springframework.data.mongodb.core.mapping.Document;

**import** com.fasterxml.jackson.annotation.JsonFormat;

**import** com.fasterxml.jackson.databind.annotation.JsonSerialize;

**import** com.fasterxml.jackson.datatype.jsr310.ser.LocalDateTimeSerializer;

**import** lombok.AllArgsConstructor;

**import** lombok.Data;

**import** lombok.EqualsAndHashCode;

**import** lombok.NoArgsConstructor;

**import** lombok.ToString;

@Data

@NoArgsConstructor

@AllArgsConstructor

@ToString

@EqualsAndHashCode

@Document(collection = "orders")

**public** **class** Order {

**private** **long** orderId;

@NotBlank(message = "Order Name cannot be Empty!")

@Size(min = 3, max = 200, message = "Order Name must be b/w 3-200 Characters!")

**private** String orderName;

**private** Float orderPrice;

@JsonSerialize(using = LocalDateTimeSerializer.**class**)

@JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss")

@FutureOrPresent(message = "Order Date must be Present or Future only!")

**private** LocalDateTime orderDateTime;

}

OrderRepository.java

package com.ibm.repository;

import java.util.List;

import org.springframework.data.mongodb.repository.MongoRepository;

import org.springframework.stereotype.Repository;

import com.ibm.model.Order;

@Repository

public interface OrdersRepository extends MongoRepository<Order, Long>{

public List<Order> findByOrderName(String orderName);

}

OrderService.java

package com.ibm.service;

import java.util.List;

import java.util.Optional;

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

import org.springframework.stereotype.Service;

import com.ibm.common.exceptions.OrderNotFoundException;

import com.ibm.model.Order;

import com.ibm.repository.OrdersRepository;

@Service

public class OrdersService {

@Autowired

private OrdersRepository ordersRepository;

public List<Order> getOrders() {

return ordersRepository.findAll();

}

public Optional<Order> getOrderById(long orderId) {

return ordersRepository.findById(orderId);

}

public Order validateOrderById(long orderId) {

return ordersRepository.findById(orderId).orElseThrow(OrderNotFoundException::new);

}

public List<Order> getOrdersByName(String orderName) {

return ordersRepository.findByOrderName(orderName);

}

public Order createOrder(Order order) {

return ordersRepository.save(order);

}

public Order updateOrder(Order updatedOrder) {

validateOrderById(updatedOrder.getOrderId());

return ordersRepository.save(updatedOrder);

}

public void deleteOrder(long orderId) {

validateOrderById(orderId);

ordersRepository.deleteById(orderId);

}

}

application.properties

spring.data.mongodb.host=localhost

spring.data.mongodb.port=27017

spring.data.mongodb.database=OrderStore

For Auto generated IDs best to use Strings or ObjectID

**package** com.ibm.model;

**import** java.time.LocalDateTime;

**import** javax.validation.constraints.FutureOrPresent;

**import** javax.validation.constraints.NotBlank;

**import** javax.validation.constraints.Size;

**import** org.bson.types.ObjectId;

**import** org.springframework.data.annotation.Id;

**import** org.springframework.data.mongodb.core.mapping.Document;

**import** com.fasterxml.jackson.annotation.JsonFormat;

**import** com.fasterxml.jackson.databind.annotation.JsonSerialize;

**import** com.fasterxml.jackson.datatype.jsr310.ser.LocalDateTimeSerializer;

**import** lombok.AllArgsConstructor;

**import** lombok.Data;

**import** lombok.EqualsAndHashCode;

**import** lombok.NoArgsConstructor;

**import** lombok.ToString;

@Data

@NoArgsConstructor

@AllArgsConstructor

@ToString

@EqualsAndHashCode

@Document(collection = "orders")

**public** **class** Order {

@Id

**private** String orderId;

@NotBlank(message = "Order Name cannot be Empty!")

@Size(min = 3, max = 200, message = "Order Name must be b/w 3-200 Characters!")

**private** String orderName;

**private** Float orderPrice;

@JsonSerialize(using = LocalDateTimeSerializer.**class**)

@JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss")

@FutureOrPresent(message = "Order Date must be Present or Future only!")

**private** LocalDateTime orderDateTime;

}

**package** com.ibm.repository;

**import** java.util.List;

**import** org.springframework.data.mongodb.repository.MongoRepository;

**import** org.springframework.stereotype.Repository;

**import** com.ibm.model.Order;

@Repository

**public** **interface** OrdersRepository **extends** MongoRepository<Order, String>{

**public** List<Order> findByOrderName(String orderName);

}

**package** com.ibm.service;

**import** java.util.List;

**import** java.util.Optional;

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

**import** org.springframework.stereotype.Service;

**import** com.ibm.common.exceptions.OrderNotFoundException;

**import** com.ibm.model.Order;

**import** com.ibm.repository.OrdersRepository;

@Service

**public** **class** OrdersService {

@Autowired

**private** OrdersRepository ordersRepository;

**public** List<Order> getOrders() {

**return** ordersRepository.findAll();

}

**public** Optional<Order> getOrderById(String orderId) {

**return** ordersRepository.findById(orderId);

}

**public** Order validateOrderById(String orderId) {

**return** ordersRepository.findById(orderId).orElseThrow(OrderNotFoundException::**new**);

}

**public** List<Order> getOrdersByName(String orderName) {

**return** ordersRepository.findByOrderName(orderName);

}

**public** Order createOrder(Order order) {

**return** ordersRepository.save(order);

}

**public** Order updateOrder(Order updatedOrder) {

validateOrderById(updatedOrder.getOrderId());

**return** ordersRepository.save(updatedOrder);

}

**public** **void** deleteOrder(String orderId) {

validateOrderById(orderId);

ordersRepository.deleteById(orderId);

}

}

**package** com.ibm.controller;

**import** java.net.URI;

**import** java.util.List;

**import** javax.validation.Valid;

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

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.CrossOrigin;

**import** org.springframework.web.bind.annotation.DeleteMapping;

**import** org.springframework.web.bind.annotation.GetMapping;

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

**import** org.springframework.web.bind.annotation.PostMapping;

**import** org.springframework.web.bind.annotation.PutMapping;

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

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

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

**import** org.springframework.web.servlet.support.ServletUriComponentsBuilder;

**import** com.ibm.common.exceptions.OrderIDMisMatchException;

**import** com.ibm.model.Order;

**import** com.ibm.service.OrdersService;

@CrossOrigin

@RestController

@RequestMapping("/orders")

**public** **class** OrdersController {

@Autowired

**private** OrdersService ordersService;

@GetMapping

**public** ResponseEntity<List<Order>> getOrders()

{

**return** ResponseEntity.*ok*(ordersService.getOrders());

}

@GetMapping("/{id}")

**public** ResponseEntity<Order> getOrderById(@PathVariable String id)

{

**return** ResponseEntity.*of*(ordersService.getOrderById(id));

}

@GetMapping("name/{name}")

**public** ResponseEntity<List<Order>> getOrdersByName(@PathVariable String name)

{

**return** ResponseEntity.*ok*(ordersService.getOrdersByName(name));

}

@PostMapping

**public** ResponseEntity<Order> createOrder(@Valid @RequestBody Order order)

{

Order createdOrder = ordersService.createOrder(order);

URI location = ServletUriComponentsBuilder.*fromCurrentRequest*().path("/{id}")

.buildAndExpand(order.getOrderId()).toUri();

**return** ResponseEntity.*created*(location).body(createdOrder);

}

@PutMapping("/{id}")

**public** ResponseEntity<Order> updateOrder(@PathVariable String id,@Valid @RequestBody Order order)

{

**if**(id.equals(order.getOrderId()))

**throw** **new** OrderIDMisMatchException();

**return** ResponseEntity.*ok*(ordersService.updateOrder(order));

}

@DeleteMapping("/{id}")

**public** ResponseEntity<Order> createOrder(@PathVariable String id)

{

ordersService.deleteOrder(id);

**return** ResponseEntity.*noContent*().build();

}

}

Download ActiveMQ

<https://www.apache.org/dyn/closer.cgi?filename=/activemq/5.18.3/apache-activemq-5.18.3-bin.zip&action=download>

C:\Users\user191\Downloads>**cd apache-activemq-5.18.3\bin**

C:\Users\user191\Downloads\apache-activemq-5.18.3\bin>**activemq start**

Username: admin

Password: admin

Rest Template Client

Create a Spring starter project called Order-Client

**package** com.ibm;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.web.client.RestTemplate;

@SpringBootApplication

**public** **class** OrderClientApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(OrderClientApplication.**class**, args);

}

@Bean

RestTemplate loadRestTemplate() {

RestTemplate rt = **new** RestTemplate();

**return** rt;

}

}

**package** com.ibm.controller;

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

**import** org.springframework.web.bind.annotation.GetMapping;

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

**import** org.springframework.web.client.RestTemplate;

@RestController

**public** **class** ClientController {

@Autowired

RestTemplate rt;

@GetMapping

**public** String getOrders()

{

**return** rt.getForObject("http://localhost:8080/orders", String.**class**);

}

}

application.properties

server.port=9000

Go to <http://localhost:9000>

To handle the JSON

**package** com.ibm.controller;

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

**import** org.springframework.http.HttpHeaders;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.MediaType;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.GetMapping;

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

**import** org.springframework.web.client.RestTemplate;

@RestController

**public** **class** ClientController {

@Autowired

RestTemplate rt;

@GetMapping

**public** ResponseEntity<String> getOrders()

{

HttpHeaders myheaders = **new** HttpHeaders();

myheaders.setContentType(MediaType.***APPLICATION\_JSON***);

ResponseEntity<String> re = **new** ResponseEntity<>(rt.getForObject("http://localhost:8080/orders", String.**class**), myheaders, HttpStatus.***OK***);

**return** re;

}

}

Create a Async Publisher for Active MQ

**package** com.ibm.controller;

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

**import** org.springframework.jms.core.JmsTemplate;

**import** org.springframework.web.bind.annotation.GetMapping;

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

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

@RestController

**public** **class** PublishOrdersController {

@Autowired

JmsTemplate jt;

@GetMapping("/publish")

**public** **void** publish(@RequestParam String order)

{

jt.convertAndSend("orders-queue", order);

}

}

application.properties

spring.activemq.broker-url=tcp://localhost:61616

**Updating to POST instead of GET**

**package** com.ibm.controller;

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

**import** org.springframework.jms.core.JmsTemplate;

**import** org.springframework.web.bind.annotation.PostMapping;

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

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

@RestController

**public** **class** PublishOrdersController {

@Autowired

JmsTemplate jt;

@PostMapping("/publish")

**public** **void** publish(@RequestBody String order)

{

jt.convertAndSend("orders-queue", order);

}

}

Create a consumer with just Spring-ActiveMQ Starter

**package** com.ibm;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.jms.annotation.JmsListener;

@SpringBootApplication

**public** **class** ConsumeOrderApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(ConsumeOrderApplication.**class**, args);

}

**@JmsListener(destination = "orders-queue")**

**public void processOrder(String order) {**

**System.*out*.println("++++++++++++++++++++++++++++");**

**System.*out*.println(order);**

**System.*out*.println("++++++++++++++++++++++++++++");**

**}**

}

Update your Order-Mongo

package com.ibm.service;

import java.util.List;

import java.util.Optional;

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

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

import org.springframework.jms.JmsException;

import org.springframework.jms.core.JmsTemplate;

import org.springframework.stereotype.Service;

import com.fasterxml.jackson.core.JsonProcessingException;

import com.fasterxml.jackson.databind.ObjectMapper;

import com.ibm.common.exceptions.OrderNotFoundException;

import com.ibm.model.Order;

import com.ibm.repository.OrdersRepository;

@Service

public class OrdersService {

**@Autowired**

**JmsTemplate jt;**

**@Value("${my-orders-queue}")**

**private String queue;**

@Autowired

private OrdersRepository ordersRepository;

public List<Order> getOrders() {

return ordersRepository.findAll();

}

public Optional<Order> getOrderById(String orderId) {

return ordersRepository.findById(orderId);

}

public Order validateOrderById(String orderId) {

return ordersRepository.findById(orderId).orElseThrow(OrderNotFoundException::new);

}

public List<Order> getOrdersByName(String orderName) {

return ordersRepository.findByOrderName(orderName);

}

**public Order createOrder(Order order) {**

**Order createdOrder = ordersRepository.save(order);**

**ObjectMapper om = new ObjectMapper();**

**try {**

**jt.convertAndSend(queue, om.writeValueAsString(createdOrder));**

**} catch (JmsException e) {**

**e.printStackTrace();**

**} catch (JsonProcessingException e) {**

**e.printStackTrace();**

**}**

**return createdOrder;**

**}**

public Order updateOrder(Order updatedOrder) {

validateOrderById(updatedOrder.getOrderId());

return ordersRepository.save(updatedOrder);

}

public void deleteOrder(String orderId) {

validateOrderById(orderId);

ordersRepository.deleteById(orderId);

}

}

application.properties

spring.data.mongodb.host=localhost

spring.data.mongodb.port=27017

spring.data.mongodb.database=OrderStore

spring.activemq.broker-url=tcp://localhost:61616

my-orders-queue=orders-queue

<dependency>

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

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

</dependency>

Employee API with Document Database

**package** com.ibm;

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

**import** org.springframework.boot.CommandLineRunner;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** com.ibm.model.Address;

**import** com.ibm.model.Employee;

**import** com.ibm.repository.EmployeeRepository;

@SpringBootApplication

**public** **class** EmployeeMongoApplication **implements** CommandLineRunner{

@Autowired

EmployeeRepository employeeRepository;

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(EmployeeMongoApplication.**class**, args);

}

@Override

**public** **void** run(String... args) **throws** Exception {

Employee employee = **new** Employee();

employee.setEmpid("IBM-202");

employee.setAge(30);

employee.setDepartment("IT");

employee.setName("Raj");

employee.setAddress(**new** Address("MG ROAD", "Bangalore", 560017));

employeeRepository.save(employee);

}

}

**package** com.ibm.repository;

**import** org.springframework.data.mongodb.repository.MongoRepository;

**import** org.springframework.stereotype.Repository;

**import** com.ibm.model.Employee;

@Repository

**public** **interface** EmployeeRepository **extends** MongoRepository<Employee, String>{

}

**package** com.ibm.model;

**import** lombok.AllArgsConstructor;

**import** lombok.Data;

**import** lombok.NoArgsConstructor;

@NoArgsConstructor

@AllArgsConstructor

@Data

**public** **class** Address {

**private** String street;

**private** String city;

**private** **int** pincode;

}

**package** com.ibm.model;

**import** javax.validation.constraints.NotBlank;

**import** lombok.AllArgsConstructor;

**import** lombok.Data;

**import** lombok.NoArgsConstructor;

@NoArgsConstructor

@AllArgsConstructor

@Data

**public** **class** Person {

@NotBlank

**private** String name;

**private** **int** age;

**private** Address address;

}

package com.ibm.model;

import org.springframework.data.annotation.Id;

import org.springframework.data.mongodb.core.mapping.Document;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.EqualsAndHashCode;

import lombok.NoArgsConstructor;

@NoArgsConstructor

@AllArgsConstructor

@Data

@EqualsAndHashCode(callSuper = true)

@Document

public class Employee extends Person {

@Id

private String id;

private String empid;

private String department;

}

application.properties

spring.data.mongodb.host=localhost

spring.data.mongodb.port=27017

spring.data.mongodb.database=employees-store

Reactive Spring Boot

**package** com.ibm.controller;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.GetMapping;

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

**import** com.ibm.model.Greeting;

**import** reactor.core.publisher.Flux;

**import** reactor.core.publisher.Mono;

@RestController

**public** **class** GreetingController {

@GetMapping("/hello")

**public** ResponseEntity<Mono<Greeting>> hello() {

**return** ResponseEntity.*ok*(Mono.*just*(**new** Greeting("Hello Reactive!!!")));

}

@GetMapping("/hellos")

**public** ResponseEntity<Flux<Greeting>> hellos() {

**return** ResponseEntity.*ok*(Flux.*just*(**new** Greeting("Hello Reactive!!!"), **new** Greeting("Hello Reactive!!!"),

**new** Greeting("Hello Reactive!!!"), **new** Greeting("Hello Reactive!!!"), **new** Greeting("Hello Reactive!!!"),

**new** Greeting("Hello Reactive!!!"), **new** Greeting("Hello Reactive!!!")));

}

}

**package** com.ibm.model;

**import** lombok.AllArgsConstructor;

**import** lombok.Data;

**import** lombok.NoArgsConstructor;

@NoArgsConstructor

@AllArgsConstructor

@Data

**public** **class** Greeting {

**private** String message;

}