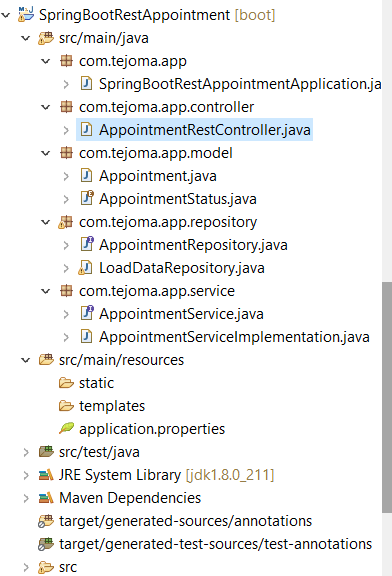
Project Architecture:



Pom.xml:

<?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 https://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>2.5.9</version>

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

</parent>

<groupId>com.tejoma</groupId>

<artifactId>SpringBootRestAppointment</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>SpringBootRestAppointment</name>

<description>Spring Boot App for user Registration</description>

<properties>

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

</properties>

<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-web</artifactId>

</dependency>

<dependency>

<groupId>org.apache.derby</groupId>

<artifactId>derby</artifactId>

<scope>runtime</scope>

</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>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-engine</artifactId>

<version>5.1.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>com.fasterxml.jackson.datatype</groupId>

<artifactId>jackson-datatype-jsr310</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

**Main class:**

package com.tejoma.app;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringBootRestAppointmentApplication {

public static void main(String[] args) {

SpringApplication.run(SpringBootRestAppointmentApplication.class, args);

}

}

**AppointmentRestController:**

**package** com.tejoma.app.controller;

**import** java.time.LocalDate;

**import** java.util.List;

**import** java.util.Optional;

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

**import** org.springframework.format.annotation.DateTimeFormat;

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

**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.ResponseStatus;

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

**import** com.tejoma.app.model.Appointment;

**import** com.tejoma.app.service.AppointmentService;

@RestController

@RequestMapping("/api/v1/appointments")

**public** **class** AppointmentRestController {

@Autowired

**private** AppointmentService appointmentService;

**public** AppointmentRestController() {

}

**public** AppointmentRestController(AppointmentService appointmentService) {

**this**.appointmentService = appointmentService;

}

/\*\* GET request to return specific appointments \*\*/

@RequestMapping(path = "/{appointmentId}", method = RequestMethod.***GET***)

**public** Optional<Appointment> findById(@PathVariable Long appointmentId) {

**return** appointmentService.findById(appointmentId);

}

/\*\* GET request to return all appointments \*\*/

@RequestMapping(path = "/", method = RequestMethod.***GET***)

List<Appointment> findAll() {

**return** appointmentService.findAll();

}

/\*\* GET request to return all appointments based on a date range and ordered by price \*\*/

@RequestMapping(method = RequestMethod.***GET***)

**public** List<Appointment> findByDateRangeSortedByPrice(@DateTimeFormat(iso = DateTimeFormat.ISO.***DATE***) @RequestParam("startDate") LocalDate startDate,

@DateTimeFormat(iso = DateTimeFormat.ISO.***DATE***) @RequestParam("endDate") LocalDate endDate) {

**return** appointmentService.findByDateRangeSortedByPrice(startDate, endDate);

}

/\*\* POST request to add new appointments \*\*/

@RequestMapping(method = RequestMethod.***POST***, produces = "application/json", consumes = "application/json")

@ResponseStatus(HttpStatus.***CREATED***)

**public** Appointment create(@RequestBody Appointment appointment) {

**return** appointmentService.create(appointment);

}

/\*\* PUT request to update appointments \*\*/

@RequestMapping(path = "/{appointmentId}", method = RequestMethod.***PUT***, produces = "application/json", consumes = "application/json")

@ResponseStatus(HttpStatus.***OK***)

**public** Appointment update(@PathVariable Long appointmentId, @RequestBody Appointment appointment) {

**return** appointmentService.update(appointmentId, appointment);

}

/\*\* PATCH request to update status of an appointment \*\*/

@RequestMapping(path = "/{appointmentId}", method = RequestMethod.***PATCH***, produces = "application/json", consumes = "application/json")

@ResponseStatus(HttpStatus.***OK***)

**public** Appointment updateStatus(@PathVariable Long appointmentId, @RequestBody Appointment appointment) {

**return** appointmentService.updateStatus(appointmentId, appointment);

}

/\*\* DELETE request to delete specific appointments \*\*/

@RequestMapping(path = "/{appointmentId}", method = RequestMethod.***DELETE***)

@ResponseStatus(HttpStatus.***OK***)

**void** deleteById(@PathVariable Long appointmentId) {

appointmentService.deleteById(appointmentId);

}

}

**Appointment**

**package** com.tejoma.app.model;

**import** java.math.BigDecimal;

**import** java.sql.Time;

**import** java.sql.Timestamp;

**import** java.time.LocalDate;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.Id;

**import** lombok.Data;

@Data

@Entity

**public** **class** Appointment {

**private** @Id @GeneratedValue Long id;

**private** Timestamp createdAt = **new** Timestamp(System.*currentTimeMillis*());

**private** LocalDate appointmentDate;

**private** Time appointmentStartTime;

**private** Time appointmentEndTime;

**private** String nameOfDoctor;

**private** AppointmentStatus status = AppointmentStatus.***Booked***;

**private** BigDecimal price;

**public** Appointment() {

}

**public** Appointment(Timestamp createdAt, LocalDate appointmentDate, Time appointmentStartTime, Time appointmentEndTime, String nameOfDoctor, AppointmentStatus status, BigDecimal price) {

**this**.createdAt = createdAt;

**this**.appointmentDate = appointmentDate;

**this**.appointmentStartTime = appointmentStartTime;

**this**.appointmentEndTime = appointmentEndTime;

**this**.nameOfDoctor = nameOfDoctor;

**this**.status = status;

**this**.price = price;

}

**public** Appointment(LocalDate appointmentDate, Time appointmentStartTime, Time appointmentEndTime, String nameOfDoctor, BigDecimal price) {

**this**.appointmentDate = appointmentDate;

**this**.appointmentStartTime = appointmentStartTime;

**this**.appointmentEndTime = appointmentEndTime;

**this**.nameOfDoctor = nameOfDoctor;

**this**.price = price;

}

**public** Appointment(LocalDate appointmentDate, String nameOfDoctor, BigDecimal price) {

**this**.appointmentDate = appointmentDate;

**this**.nameOfDoctor = nameOfDoctor;

**this**.price = price;

}

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** Timestamp getCreatedAt() {

**return** createdAt;

}

**public** **void** setCreatedAt(Timestamp createdAt) {

**this**.createdAt = createdAt;

}

**public** LocalDate getAppointmentDate() {

**return** appointmentDate;

}

**public** **void** setAppointmentDate(LocalDate appointmentDate) {

**this**.appointmentDate = appointmentDate;

}

**public** Time getAppointmentStartTime() {

**return** appointmentStartTime;

}

**public** **void** setAppointmentStartTime(Time appointmentStartTime) {

**this**.appointmentStartTime = appointmentStartTime;

}

**public** Time getAppointmentEndTime() {

**return** appointmentEndTime;

}

**public** **void** setAppointmentEndTime(Time appointmentEndTime) {

**this**.appointmentEndTime = appointmentEndTime;

}

**public** String getNameOfDoctor() {

**return** nameOfDoctor;

}

**public** **void** setNameOfDoctor(String nameOfDoctor) {

**this**.nameOfDoctor = nameOfDoctor;

}

**public** AppointmentStatus getStatus() {

**return** status;

}

**public** **void** setStatus(AppointmentStatus status) {

**this**.status = status;

}

**public** BigDecimal getPrice() {

**return** price;

}

**public** **void** setPrice(BigDecimal price) {

**this**.price = price;

}

}

**AppointmentStatus:**

**package** com.tejoma.app.model;

**public** **enum** AppointmentStatus{

***Available***, ***Booked***;

}

**AppointmentRepository:**

**package** com.tejoma.app.repository;

**import** java.time.LocalDate;

**import** java.util.List;

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

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

**import** com.tejoma.app.model.Appointment;

@Repository

**public** **interface** AppointmentRepository **extends** JpaRepository<Appointment, Long> {

List<Appointment> findAllByAppointmentDateBetweenOrderByPriceAsc(LocalDate startDate, LocalDate endDate);

}

**LoadDataRepository:**

**package** com.tejoma.app.repository;

**import** java.math.BigDecimal;

**import** java.sql.Time;

**import** java.time.LocalDate;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

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

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

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

**import** com.tejoma.app.model.Appointment;

**import** lombok.extern.slf4j.Slf4j;

@Configuration

@Slf4j

**public** **class** LoadDataRepository {

**static** **final** **private** Logger ***logger*** = LoggerFactory.*getLogger*(LoadDataRepository.**class**);

@Bean

CommandLineRunner initDatabase(AppointmentRepository appointmentRepository) {

**return** args -> {

***logger***.info("Preloading " + appointmentRepository.save(**new** Appointment(LocalDate.*of*(2019, 2, 3), Time.*valueOf*("11:00:00"), Time.*valueOf*("12:00:00"),"Markus", **new** BigDecimal(90))));

***logger***.info("Preloading " + appointmentRepository.save(**new** Appointment(LocalDate.*of*(2019, 2, 15), Time.*valueOf*("14:00:00"), Time.*valueOf*("15:00:00"),"Thomas", **new** BigDecimal(60))));

***logger***.info("Preloading " + appointmentRepository.save(**new** Appointment(LocalDate.*of*(2019, 1, 10), Time.*valueOf*("08:00:00"), Time.*valueOf*("09:00:00"),"Mikkel", **new** BigDecimal(30))));

***logger***.info("Preloading " + appointmentRepository.save(**new** Appointment(LocalDate.*of*(2019, 1, 23), Time.*valueOf*("18:00:00"), Time.*valueOf*("19:00:00"),"Chris", **new** BigDecimal(10))));

};

}

}

**AppointmentService:**

**package** com.tejoma.app.service;

**import** java.time.LocalDate;

**import** java.util.List;

**import** java.util.Optional;

**import** com.tejoma.app.model.Appointment;

**public** **interface** AppointmentService {

Optional<Appointment> findById(Long appointmentId);

List<Appointment> findAll();

List<Appointment> findByDateRangeSortedByPrice(LocalDate startDate, LocalDate endDate);

Appointment create(Appointment appointment);

Appointment update(Long appointmentId, Appointment appointment);

Appointment updateStatus(Long appointmentId, Appointment appointment);

**void** deleteById(Long appointmentId);

}

**AppointmentServiceImplementation:**

**package** com.tejoma.app.service;

**import** java.time.LocalDate;

**import** java.util.List;

**import** java.util.Optional;

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

**import** org.springframework.stereotype.Component;

**import** com.tejoma.app.model.Appointment;

**import** com.tejoma.app.repository.AppointmentRepository;

@Component("appointmentService")

**public** **class** AppointmentServiceImplementation **implements** AppointmentService {

@Autowired

AppointmentRepository appointmentRepository;

**public** AppointmentServiceImplementation() {

}

**public** AppointmentServiceImplementation(AppointmentRepository appointmentRepository) {

**this**.appointmentRepository = appointmentRepository;

}

@Override

**public** Optional<Appointment> findById(Long appointmentId) {

**return** appointmentRepository.findById(appointmentId);

}

@Override

**public** List<Appointment> findAll() {

**return** appointmentRepository.findAll();

}

@Override

**public** List<Appointment> findByDateRangeSortedByPrice(LocalDate startDate, LocalDate endDate) {

**return** appointmentRepository.findAllByAppointmentDateBetweenOrderByPriceAsc(startDate, endDate);

}

@Override

**public** Appointment create(Appointment appointment) {

**return** appointmentRepository.save(appointment);

}

@Override

**public** Appointment update(Long appointmentId, Appointment appointment) {

**return** appointmentRepository.save(appointment);

}

@Override

**public** Appointment updateStatus(Long appointmentId, Appointment appointment) {

Optional<Appointment> appointmentList = appointmentRepository.findById(appointmentId);

**if**(appointmentList.isPresent()){

**if**(appointment.getStatus() != **null**){

appointmentList.get().setStatus(appointment.getStatus());

}

**return** appointmentRepository.save(appointmentList.get());

}

**return** **null**;

}

@Override

**public** **void** deleteById(Long appointmentId) {

appointmentRepository.deleteById(appointmentId);

}

}

application.properties

server.port:9092

**REST Endpoints**

1.**GET** request to return specific appointments - /api/v1/appointments/{appointmentId}

<http://localhost:9092/api/v1/appointments/3>

2.**GET** request to return all appointments - /api/v1/appointments/

<http://localhost:9092/api/v1/appointments/>

3.**GET** request to return all appointments based on a date range and ordered by price - /api/v1/appointments?startDate={start\_date}&endDate={end\_date}

<http://localhost:9092/api/v1/appointments?startDate=2019-01-23&endDate=2019-02-15>

4.**POST** request to add new appointments - /api/v1/appointments

<http://localhost:9092/api/v1/appointments>

Pass Body as JSON:

{ "appointmentDate": "2019-02-14", "appointmentStartTime": "20:00:00", "appointmentEndTime": "21:00:00", "nameOfDoctor": "Venu", "status": "Booked", "price": 100 }

5.**PUT** request to update appointments - /api/v1/appointments/{appointmentId}

<http://localhost:9092/api/v1/appointments/5>

Pass Body as JSON:

{ "id": 5,  
"appointmentDate": "2019-02-14", "appointmentStartTime": "20:00:00", "appointmentEndTime": "21:00:00", "nameOfDoctor": "Venu", "status": "Available", "price": 100}

6.**PATCH** request to update status of an appointment - /api/v1/appointments/{appointmentId}

<http://localhost:9092/api/v1/appointments/4>

Pass Body as JSON:

{ "status": "Available" }

7.**DELETE** request to delete specific appointments - /api/v1/appointments/{appointmentId}

<http://localhost:9092/api/v1/appointments/3>