**Spring Boot**

Contents

[**1. Client Dependencies** 1](#_Toc75608897)

[**2. Server Dependencies** 1](#_Toc75608898)

[**3. Spring Boot With JavaServer Pages (JSP)** 1](#_Toc75608899)

[**4. Port – Client** 2](#_Toc75608900)

[**5. ClientApplication** 2](#_Toc75608901)

[**6. Client – SecurityConfig** 2](#_Toc75608902)

[**7. Client – Entity** 3](#_Toc75608903)

[**8. Client – Service** 4](#_Toc75608904)

[**9. Client – Controller** 5](#_Toc75608905)

[**10. Client – Application.properties** 6](#_Toc75608906)

[**11. Server – Application.properties** 6](#_Toc75608907)

[**12. Server – Entity** 6](#_Toc75608908)

[**13. Server – Service** 7](#_Toc75608909)

[**14. Server – Repository** 8](#_Toc75608910)

[**15. Server – Controller** 8](#_Toc75608911)

[**16. Note** 9](#_Toc75608912)

# **1. Client Dependencies**

- Spring Web

- Spring Security

- Spring Boot DevTools

# **2. Server Dependencies**

- Spring Web

- Spring Boot DevTools

- MS SQL Server Driver

- Spring Data JPA

# **3. Spring Boot With JavaServer Pages (JSP)**

3.1 Maven dependencies

<dependency>

<groupId>org.apache.tomcat.embed</groupId>

<artifactId>tomcat-embed-jasper</artifactId>

<version>9.0.44</version>

</dependency>

<!-- https://mvnrepository.com/artifact/javax.servlet/jstl -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

3.2 View Resolver Configuration

Trong file application.properties, gõ

spring.mvc.view.prefix: /WEB-INF/jsp/

spring.mvc.view.suffix: .jsp

Tạo thư mục src/main/webapp/WEB-INF/jsp/

# **4. Port – Client**

- Trong file application.properties, gõ: server.port=8081

# **5. ClientApplication**

@SpringBootApplication

**public** **class** Problem16ClientApplication {

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

SpringApplication.*run*(Problem16ClientApplication.**class**, args);

}

@Bean

**public** RestTemplate restTemplate() {

**return** **new** RestTemplate();

}

}

# **6. Client – SecurityConfig**

@Configuration

@EnableWebSecurity

**public** **class** SecurityConfig **extends** WebSecurityConfigurerAdapter{

@Override

**protected** **void** configure(AuthenticationManagerBuilder auth) **throws** Exception {

PasswordEncoder encoder = PasswordEncoderFactories.*createDelegatingPasswordEncoder*();

auth

.inMemoryAuthentication()

.withUser("user")

.password(encoder.encode("user"))

.roles("USER")

.and()

.withUser("admin")

.password(encoder.encode("admin"))

.roles("USER", "ADMIN");

}

@Override

**protected** **void** configure(HttpSecurity http) **throws** Exception {

http

.authorizeRequests()

.antMatchers("/lecturer/list").permitAll()

.antMatchers("/lecturer/create").hasRole("USER")

.antMatchers("/subject/create/\*\*").hasAnyRole("ADMIN")

.and()

.httpBasic();

}

}

# **7. Client – Entity**

@JsonIgnoreProperties(ignoreUnknown = **true**)

**public** **class** Lecturer {

**private** **int** id;

**private** String fullName;

**private** String researchArea;

**private** String phoneNumber;

**private** String email;

**private** List<Subject> subjects = **new** ArrayList<Subject>();

}

@JsonIgnoreProperties(ignoreUnknown = **true**)

**public** **class** Subject {

**private** **int** id;

**private** String name;

**private** **int** publishYear;

**private** String description;

**private** Lecturer lecturer;

}

# **8. Client – Service**

@Service

**public** **class** LecturerService {

**private** **static** **final** String ***URL*** = "http://localhost:8080/api/lecturers";

**private** **static** **final** String ***idURL*** = ***URL*** + "/{id}";

@Autowired

**private** RestTemplate restTemplate;

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

ResponseEntity<Lecturer[]> responseEntity = restTemplate.getForEntity(***URL***, Lecturer[].**class**);

Lecturer[] lecturerArray = responseEntity.getBody();

**return** Arrays.*asList*(lecturerArray);

}

**public** Lecturer getLecturer(**int** id) {

Map<String, Integer> params = **new** HashMap<String, Integer>();

params.put("id", id);

Lecturer lecturer = restTemplate.getForObject(***idURL***, Lecturer.**class**, params);

**return** lecturer;

}

**public** **void** deleteLecturer(**int** id) {

Map<String, Integer> params = **new** HashMap<String, Integer>();

params.put("id", id);

restTemplate.delete(***idURL***, params);

}

**public** **void** updateLecturer(Lecturer lecturer) {

Map<String, Integer> params = **new** HashMap<String, Integer>();

params.put("id", lecturer.getId());

restTemplate.put(***idURL***, lecturer, params);

}

**public** **void** saveLecturer(Lecturer lecturer) {

restTemplate.postForObject(***URL***, lecturer, Lecturer.**class**);

}

}

# **9. Client – Controller**

@Controller

@RequestMapping("/lecturer")

**public** **class** LecturerController {

@Autowired

**private** LecturerService lecturerService;

@GetMapping("/list")

**public** ModelAndView getListLecturer() {

List<Lecturer> lecturers = lecturerService.findAll();

**return** **new** ModelAndView("list-lecturer", "lecturers", lecturers);

}

@GetMapping("/detail/{id}")

**public** ModelAndView getLecturer(@PathVariable("id") **int** id) {

Lecturer lecturer = lecturerService.getLecturer(id);

**return** **new** ModelAndView("detail-lecturer", "lecturer", lecturer);

}

@GetMapping("/delete/{id}")

**public** ModelAndView deleteLecturer(@PathVariable("id") **int** id) {

lecturerService.deleteLecturer(id);

**return** **new** ModelAndView("redirect:/lecturer/list");

}

@GetMapping("/update/{id}")

**public** ModelAndView showUpdateLecturerForm(@PathVariable("id") **int** id) {

Lecturer lecturer = lecturerService.getLecturer(id);

**return** **new** ModelAndView("update-lecturer-form", "lecturer", lecturer);

}

@PostMapping("/update/{id}")

**public** ModelAndView processUpdateLecturer(@ModelAttribute("lecturer") Lecturer lecturer) {

lecturerService.updateLecturer(lecturer);

**return** **new** ModelAndView("redirect:/lecturer/list");

}

@GetMapping("/create")

**public** ModelAndView showAddLecturerForm() {

**return** **new** ModelAndView("add-lecturer-form", "lecturer", **new** Lecturer());

}

@PostMapping("/create")

**public** ModelAndView processAddLecturerForm(@ModelAttribute("lecturer") Lecturer lecturer) {

lecturerService.saveLecturer(lecturer);

**return** **new** ModelAndView("redirect:/lecturer/list");

}

}

# **10. Client – Application.properties**

spring.mvc.view.prefix: /WEB-INF/jsp/

spring.mvc.view.suffix: .jsp

server.port=8081

# **11. Server – Application.properties**

spring.datasource.url=jdbc:sqlserver://localhost:1433;databaseName=problem16

spring.datasource.username=sa

spring.datasource.password=sapassword

spring.jpa.hibernate.ddl-auto=create-drop

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.SQLServer2012Dialect

spring.jpa.properties.hibernate.format\_sql=true

spring.jpa.hibernate.naming.implicit-strategy=org.hibernate.boot.model.naming.ImplicitNamingStrategyLegacyJpaImpl

spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl

# **12. Server – Entity**

@Entity

**public** **class** Lecturer {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

**private** **int** id;

**private** String fullName;

**private** String researchArea;

**private** String phoneNumber;

**private** String email;

@JsonManagedReference

@OneToMany(cascade = CascadeType.***ALL***, mappedBy = "lecturer", fetch = FetchType.***EAGER***)

**private** List<Subject> subjects = **new** ArrayList<Subject>();

}

@Entity

**public** **class** Subject {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

**private** **int** id;

**private** String name;

**private** **int** publishYear;

**private** String description;

@JsonBackReference

@ManyToOne

**private** Lecturer lecturer;

}

# **13. Server – Service**

@Service

**public** **class** LecturerService {

@Autowired

**private** LecturerRepository lecturerRepository;

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

**return** lecturerRepository.findAll();

}

**public** **void** addLecturer(Lecturer lecturer) {

lecturerRepository.save(lecturer);

}

**public** Lecturer getLecturer(**int** id) {

**return** lecturerRepository.findById(id).get();

}

**public** **void** deleteLecturerById(**int** id) {

lecturerRepository.deleteById(id);

}

**public** **void** updateLecturer(Lecturer lecturer) {

lecturerRepository.save(lecturer);

}

**public** Lecturer getLecturerByPhoneNumberAndEmail(String phoneNumber, String email) {

**return** lecturerRepository.findByPhoneNumberAndEmail(phoneNumber, email);

}

}

# **14. Server – Repository**

@Repository

**public** **interface** LecturerRepository **extends** JpaRepository<Lecturer, Integer>{

// @Query("SELECT l FROM Lecturer l WHERE l.phoneNumber = ?1 AND l.email = ?2")

Lecturer findByPhoneNumberAndEmail(String phoneNumber, String email);

}

# **15. Server – Controller**

@RestController

@RequestMapping("/api/lecturers")

**public** **class** LecturerController {

@Autowired

**private** LecturerService lecturerService;

@GetMapping

**public** List<Lecturer> getListLecturer() {

**return** lecturerService.findAll();

}

@PostMapping

**public** **void** registerNewLecturer(@RequestBody Lecturer lecturer) {

lecturerService.addLecturer(lecturer);

}

@GetMapping("/{lecturerID}")

**public** Lecturer getLecturerById(@PathVariable("lecturerID") **int** id) {

**return** lecturerService.getLecturer(id);

}

@DeleteMapping("/{lecturerID}")

**public** **void** deleteLecturer(@PathVariable("lecturerID") **int** id) {

lecturerService.deleteLecturerById(id);

}

@PutMapping("/{lecturerID}")

**public** **void** updateLecturer(@RequestBody Lecturer lecturer) {

lecturerService.updateLecturer(lecturer);

}

@GetMapping("/l")

@ResponseBody

**public** Lecturer getLecturer(@RequestParam("phoneNumber") String phoneNumber, @RequestParam("email") String email) {

**return** lecturerService.getLecturerByPhoneNumberAndEmail(phoneNumber, email);

}

}

@RestController

@RequestMapping("api/subjects")

**public** **class** SubjectController {

@Autowired

**private** SubjectService subjectService;

@PostMapping

**public** **void** registerNewSubject(@RequestBody Subject subject) {

subjectService.saveSubject(subject);

}

@GetMapping

**public** List<Subject> getListSubject() {

**return** subjectService.findAll();

}

}

# **16. Note**

Fetct, by default:

OneToMany: LAZY

ManyToOne: EAGER

ManyToMany: LAZY

OneToOne: EAGER

Mối quan hệ 1 – 1 (One to One)

- Xác định được đối tượng chủ thể và đối tượng phụ thuộc, cả 2 bên đều khai báo @OneToOne

- Bên chủ thể thêm @PrimaryKeyJoinColumn

- Bên phụ thuộc thêm

+ @MapsId

+ @JoinColumn(name = “…”)

Thay đổi tên bảng -> @Table(name=”student”)

Thay đổi tên cột -> @Column(name=”firstname”)

Mối quan hệ 1 – n (One To Many)

- Phía One thêm @OneToMany với thuộc tính mappedBy

Ex:

public class Lecture {

@OneToMany(mappedBy = “lecturer”)

private List<Subject> subjects = new ArrayList<Subject>();

…

}

- Phía Many thêm:

+ @ManyToOne

+ @JoinColumn với thuộc tính name

Ex:

public class Subject {

@ManyToOne

@JoinColumn(name=”lecturerID”)

private Lecturer lecturer;

…

}