LAB7: **SCHEDULE TASKS & INTERCEPTOR – Java 5**

Tên sv: Nguyễn Hải Đăng PC03033

**PHẦNI**

**Bài 1 (2 điểm)** Hãy xây dựng **Spring Bean** class cho phép **gửi email** theo đặt tả của **interface** sau đây. MailerService Interface

//ảnh

Trong đó MailInfo là lớp mô tả thông tin đầy đủ của một email.

//ảnh

**Hướng dẫn:**

Để có thể thực hiện được yêu cầu trên, ta cần phải làm theo các hướng dẫn sau đây.

1. **Khai báo thư viện cần thiết (pom.xml)**

**<dependency>**

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

**<artifactId>spring-boot-starter-mail</artifactId>**

**</dependency>**

1. **Khai báo thông số kết nối Gmail Smtp Server**

spring.mail.host=smtp.gmail.com

spring.mail.port=587

spring.mail.username=dangnhpc03033@fpt.edu.vn

spring.mail.password=hwmynfbvmocrguqn

spring.mail.properties.mail.smtp.auth=true

spring.mail.properties.mail.smtp.starttls.enable=true

1. **Cài đặt mã nguồn cho MailerService qua lớp MailerServiceImpl**

**Lớp MailerService**

**public** **interface** MailerService {

**void** send(MailInfo mail) **throws** MessagingException;

**void** send(String to, String subject, String body) **throws** MessagingException;

**void** queue(MailInfo mail);

**void** queue(String to, String subject, String body);

}

Lớp **MailerServiceImpl**

@Service

**public** **class** MailerServiceImpl **implements** MailerService {

@Autowired

JavaMailSender sender;

@Override

**public** **void** send(MailInfo mail) **throws** MessagingException {

// **TODO** mã nguồn gửi email đặt ở đây

// Tạo message

MimeMessage message = sender.createMimeMessage();

// Sử dụng Helper để thiết lập các thông tin cần thiết cho message

MimeMessageHelper helper = **new** MimeMessageHelper(message, **true**, "utf-8");

helper.setFrom(mail.getFrom());

helper.setTo(mail.getTo());

helper.setSubject(mail.getSubject());

helper.setText(mail.getBody(), **true**);

helper.setReplyTo(mail.getFrom());

String[] cc = mail.getCc();

**if** (cc != **null** && cc.length > 0) {

helper.setCc(cc);

}

String[] bcc = mail.getBcc();

**if** (bcc != **null** && bcc.length > 0) {

helper.setBcc(bcc);

}

String[] attachments = mail.getAttachments();

**if** (attachments != **null** && attachments.length > 0) {

**for** (String path : attachments) {

File file = **new** File(path);

helper.addAttachment(file.getName(), file);

}

}

// Gửi message đến SMTP server

sender.send(message);

}

@Override

**public** **void** send(String to, String subject, String body) **throws** MessagingException {

**this**.send(**new** MailInfo(to, subject, body));

}

List<MailInfo> list = **new** ArrayList<>();

@Override

**public** **void** queue(MailInfo mail) {

list.add(mail);

}

@Override

**public** **void** queue(String to, String subject, String body) {

queue(**new** MailInfo(to, subject, body));

}

@Scheduled(fixedDelay = 5000

)

**public** **void** run() {

**while** (!list.isEmpty()) {

MailInfo mail = list.remove(0);

**try** {

**this**.send(mail);

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

}

1. **MailerController sử dụng MailerService để thực hiện gửi email đơn giản.**

@Controller

**public** **class** MailerController {

@Autowired

**private** MailerService mailerService;

@GetMapping("send-mail")

String sendMail(Model model) {

model.addAttribute("mailInfo", **new** MailInfo());

**return** "Mail/SendMail";

}

@PostMapping("send-mail")

**public** String sendEmail(@RequestParam("from") String from, @RequestParam("to") String to,

@RequestParam(value = "cc", required = **false**) String cc,

@RequestParam(value = "bcc", required = **false**) String bcc, @RequestParam("subject") String subject,

@RequestParam("body") String body,

@RequestParam(value = "attachments", required = **false**) MultipartFile[] attachments, Model model) {

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

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

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

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

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

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

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

**try** {

MailInfo mailInfo = **new** MailInfo();

mailInfo.setFrom(from);

mailInfo.setTo(to);

// Kiểm tra cc

**if** (cc != **null** && !cc.isEmpty()) {

**try** {

InternetAddress.*parse*(cc, **true**);

mailInfo.setCc(cc);

} **catch** (AddressException e) {

// Xử lý lỗi khi địa chỉ cc không hợp lệ

}

}

// Kiểm tra bcc

**if** (bcc != **null** && !bcc.isEmpty()) {

**try** {

InternetAddress.*parse*(bcc, **true**);

mailInfo.setBcc(bcc);

} **catch** (AddressException e) {

// Xử lý lỗi khi địa chỉ bcc không hợp lệ

}

}

mailInfo.setSubject(subject);

mailInfo.setBody(body);

System.***out***.println(attachments.length);

**if** (attachments != **null** && attachments.length > 0) {

String[] attachmentPaths = **new** String[attachments.length];

**for** (**int** i = 0; i < attachments.length; i++) {

MultipartFile attachment = attachments[i];

String filename = attachment.getOriginalFilename();

String filePath = "/path/send/attachments/" + filename;

File savedFile = ParamService.*save*(attachment, filePath); // Sử dụng ParamService để lưu file đính

// kèm

**if** (savedFile != **null**) {

attachmentPaths[i] = savedFile.getAbsolutePath();

} **else** {

// Xử lý lỗi không thể lưu file

}

}

mailInfo.setAttachments(attachmentPaths);

}

mailerService.send(mailInfo);

model.addAttribute("message", "Email sent successfully!");

System.***out***.println("Email sent successfully!");

} **catch** (Exception e) {

model.addAttribute("error", "An error occurred while sending the email: " + e.getMessage());

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

}

**return** "redirect:/send-mail";

}

}

1. **Cuối cùng là viết mã gửi email cho phương thức send(MailInfo) để hoàn thành yêu cầu đặt ra.**

mailerService.send(mailInfo);

**Bài 2 (2 điểm)**

**Toàn bộ code và thay đổi đã được cập nhật ở bài 1**

**Bài 3 (1 điểm)**

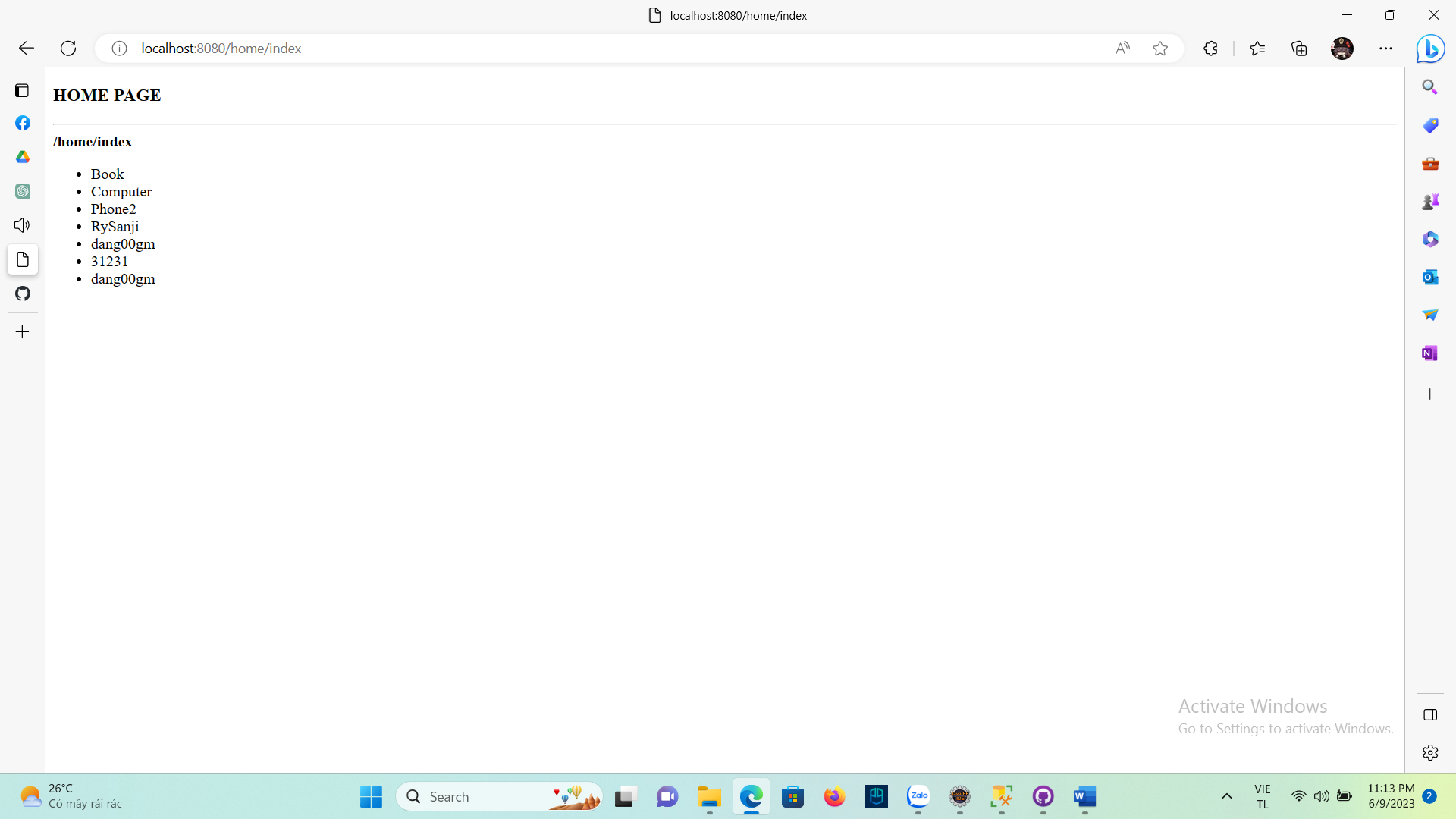
Xây dựng trang web cho phép người nhập đầy đủ thông tin **(from, to, cc, bcc, subject, body, attachments)** vào form gửi email, sử dụng MailerService để gửi email.

**Toàn bộ code và thay đổi đã được cập nhật ở bài 1**

**PHẦNII**

**Bài 4 (2 điểm)** Hiển thị tất cả loại hàng và địa chỉ truy cập trên tất cả mọi view của ứng dụng.

Sau đây là hiển thị trên trang chủ.



Dữ liệu được cung cấp cho tất cả các view, có nghĩa là không thể viết trong một controller cụ thể nào được mà phải viết trong một interceptor và cấu hình để chắn tất cả các url của các controller. Hãy thực hiện yêu cầu trên theo hướng dẫn sau đây:

1. **GlobalInterceptor chia sẻ url và loại hàng vào request scope**

• URL phải được lấy từ **request** của **preHandle**() vì nếu lấy ở **request** của **postHandle**() thì đó là URL của **view**

• Truy vấn và chia sẽ loại hàng phải viết ở **postHandle**() thì mới đảm bảo là loại hàng không bị lạc hậu vì có thể trong **controller** làm thay đổi dữ liệu này.

//ảnh

1. **InterceptorConfig Cấu hình cho phép GlobalInterceptor chạy khi request đến mọi URL ngoại trừ các URL đến tài nguyên tĩnh (js, css, images…) đặt trong thư mục assets**

@Configuration

**public** **class** InterceptorConfig **implements** WebMvcConfigurer {

@Autowired

**private** GlobalInterceptor global;

@Override

**public** **void** addInterceptors(InterceptorRegistry registry) {

registry.addInterceptor(global)

.addPathPatterns("/\*\*")

.excludePathPatterns("/assets/\*\*");

}

}

1. **HomeController: Chỉ đơn giản chuyển sang index.jsp để hiển thị**

@Controller

**public** **class** HomeController {

@RequestMapping("home/index")

**public** String index() {

**return** "home/index";

}

@RequestMapping("home/about")

**public** String about() {

**return** "home/about";

}

}

1. **View home/index.html**

Tách phần hiển thị thông tin yêu cầu thành một file riêng để tiện nhúng vào mọi view.

<!DOCTYPE html>

<html xmlns:th=*"http://www.thymeleaf.org"*>

<head>

<meta charset=*"UTF-8"*>

</head>

<body>

<h3>HOME PAGE</h3>

<hr>

<div th:include=*"home/\_global.html"*></div>

</body>

</html>

1. **Partial View \_global.html**

Hiển thị **url** và **categories** đã đặt vào trong **request**

<b th:text=*"${uri}"*></b>

<ul>

<li th:each=*"item : ${categories}"* th:text=*"${item.name}"*></li>

</ul>

**Bài 5 (2 điểm)**

Xây dựng **AuthInterceptor** và cấu hình để bảo mật ứng dụng theo yêu cầu sau:

• Cần phải đăng nhập mới được truy xuất các URL sau **o /account/edit o /account/chgpwd o /order/\*\* o Ngoại trừ /assets/\*\***

• Phải đăng nhập với vai trò admin thì mới truy xuất được các url **sau o /admin/\*\* o Ngoại trừ /admin/home/index**

• Nếu cố gắng truy xuất mà không thỏa mãn 2 điều kiện trên thì phải chuyển request về trang đăng nhập **/account/login** và đưa ra thông báo phù hợp. Sau khi đăng nhập thành thì quay trở lại url đã được bảo vệ trước đó (nếu có). Thực hiện theo hướng dẫn sau đây để giải quyết yêu cầu đặt ra **AuthInterceptor**

@Service

**public** **class** AuthInterceptor **implements** HandlerInterceptor {

@Autowired

SessionService session;

@Override

**public** **boolean** preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)

**throws** Exception {

Account user = session.get("userSession"); // lấy từ session

**if**(user == **null**) {

response.sendRedirect("/account/login");

**return** **false**;

}**else** **if** (user.isAdmin() == **false**) {

response.sendRedirect("/account/login");

**return** **false**;

}**else** **if** (user.isAdmin()) {

**return** **true**;

}

**return** **true**;

}

}

* **InterceptorConfig**

@Configuration

**public** **class** InterConfigLogin **implements** WebMvcConfigurer {

@Autowired

AuthInterceptor auth;

@Override

**public** **void** addInterceptors(InterceptorRegistry registry) {

registry.addInterceptor(auth)

.addPathPatterns("/account/edit", "/account/chgpwd", "/order/\*\*", "/admin/\*\*")

.excludePathPatterns("/assets/\*\*");

}

}

1. **AccountController**

@Controller

**public** **class** AccountController {

@Autowired

AccountDAO dao;

@Autowired

SessionService session;

@GetMapping("/account/login")

**public** String login() {

**return** "account/login";

}

@ResponseBody

@GetMapping("/account/edit")

**public** String edit() {

**return** "edit";

}

@ResponseBody

@GetMapping("/admin/index")

**public** String admin() {

**return** "admin";

}

@PostMapping("/account/login")

**public** String login(Model model, @RequestParam("username") String username,

@RequestParam("password") String password) {

**try** {

Account account = dao.findAccount(username);

**if** (!account.getPassword().equals(password)) {

model.addAttribute("message", "Mật khẩu không đúng");

} **else** {

session.set("userSession", account);

model.addAttribute("message", "Đăng nhập thành công");

}

} **catch** (Exception e) {

model.addAttribute("message", "Tên tài khoản không đúng");

}

**return** "account/login";

}

}

1. **Login form login.html**

<h3>LOGIN</h3>

<b th:text=*"${message}"*></b> <b th:text=*"${mes}"*></b>

<form action=*"/account/login"* method=*"post"*>

<input name=*"username"* placeholder=*"Username"*>

<input name=*"password"* placeholder=*"Password"*>

<button type=*"submit"*>Login</button>

</form>

**Bài 6 (1 điểm) Giảng viên cho thêm**