MỤC TIÊU:

Kết thúc bài thực hành này bạn có khả năng

- ✓ Gửi email với Spring Boot
- ✓ Sử dụng được dịch vụ nền Schedule Task để gửi email
- ✓ Sử dụng được Interceptor để chia sẻ dữ liệu và bảo vệ ứng dụng

PHẦN I

Bài 1 (2 điểm)

Hãy xây dựng Spring Bean class cho phép gửi email theo đặt tả của interface sau đây.

MailerService Interface

```
public interface MailerService {
        /**
         * Gửi email
         * @param mail thông tin email
         * @throws MessagingException lỗi gửi email
         */
        void send(MailInfo mail) throws MessagingException;
        /**
         * Gửi email đơn giản
         * @param to email người nhận
         * @param subject tiêu đề email
         * @param body nội dung email
         * @throws MessagingException lỗi gửi email
         */
        void send(String to, String subject, String body) throws MessagingException;
}
```

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

```
@Data
@NoArgsConstructor
@AllArgsConstructor
public class MailInfo {
```

```
        String from;
        String to;
        String[] cc;
        String[] bcc;
        String subject;
        String body;
        String[] attachments;
        public MailInfo(String to, String subject, String body) {
                this.from = "FPT Polytechnic <poly@fpt.edu.vn>";
                this.to = to;
                this.subject = subject;
                this.body = body;
        }
}
```

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>
```

2. Khai báo thông số kết nối GMail Smtp Server

```
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=???
spring.mail.password=***
spring.mail.properties.mail.smpt.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
```

3. Cài đặt mã nguồn cho MailerService qua 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
        }
        @Override
        public void send(String to, String subject, String body)
                                                        throws MessagingException {
                this.send(new MailInfo(to, subject, body));
        }
}
```

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

```
@ResponseBody
@RequestMapping("/mailer/demo")
public String demo(Model model) {
        try {
                mailer.send("receiver@gmail.com", "Subject", "Body");
                return "OK";
        } catch (MessagingException e) {
                return e.getMessage();
        }
}
```

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

```
// 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);
```

Bài 2 (2 điểm)

Nâng cấp MailerService để thay vì gửi mail trực tiếp thì xếp MailInfo vào hàng đợi và sử dụng @Schedule để tạo hoạt động gửi email từ phía hậu trường tránh trường hợp tắc nghẽn khi nhiều người gửi email đồng thời.

Hướng dẫn:

Hãy thực hiện nâng cấp MailerService theo hướng dẫn sau đây

1. Kích hoạt Schedule Task với @EnableScheduling

```
@SpringBootApplication
@EnableScheduling
public class J5DemoApplication {
        public static void main(String[] args) {
                SpringApplication.run(J5DemoApplication.class, args);
        }
}
```

2. Khai báo bổ sung 2 phương thức queue vào MailerService

```
/**
 * Xếp mail vào hàng đợi
```

```
 * @param mail thông tin email
 */
void queue(MailInfo mail);
/**
 * Tạo MailInfo và xếp vào hàng đợi
 * @param to email người nhận
 * @param subject tiêu đề email
 * @param body nội dung email
 */
void queue(String to, String subject, String body);
```

3. Cài đặt mã cho các phương thức trên trong MailerServiceImpl để xếp MailInfo vào List<MailInfo> (hàng đợi)

```
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));
}
```

4. Sử dụng @Scheduled để khai báo cho phương thức chạy nền thực hiện lấy MailInfo từ hàng đợi và gửi đi (5 giây sẽ kiểm tra và gửi một lần)

```
@Scheduled(fixedDelay = 5000)
public void run() {
        while(!list.isEmpty()) {
                MailInfo mail = list.remove(0);
                try {
                        this.send(mail);
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }
}
```

5. Hiệu chỉnh MailerController để xếp mail vào hàng đợi thay vì gửi trực tiếp.

```
@ResponseBody
@RequestMapping("/mailer/demo")
public String demo(Model model) {
        mailer.queue("receiver@gmail.com", "Subject", "Body");
        return "Mail của bạn sẽ được gửi đi trong giây lát";
}
```

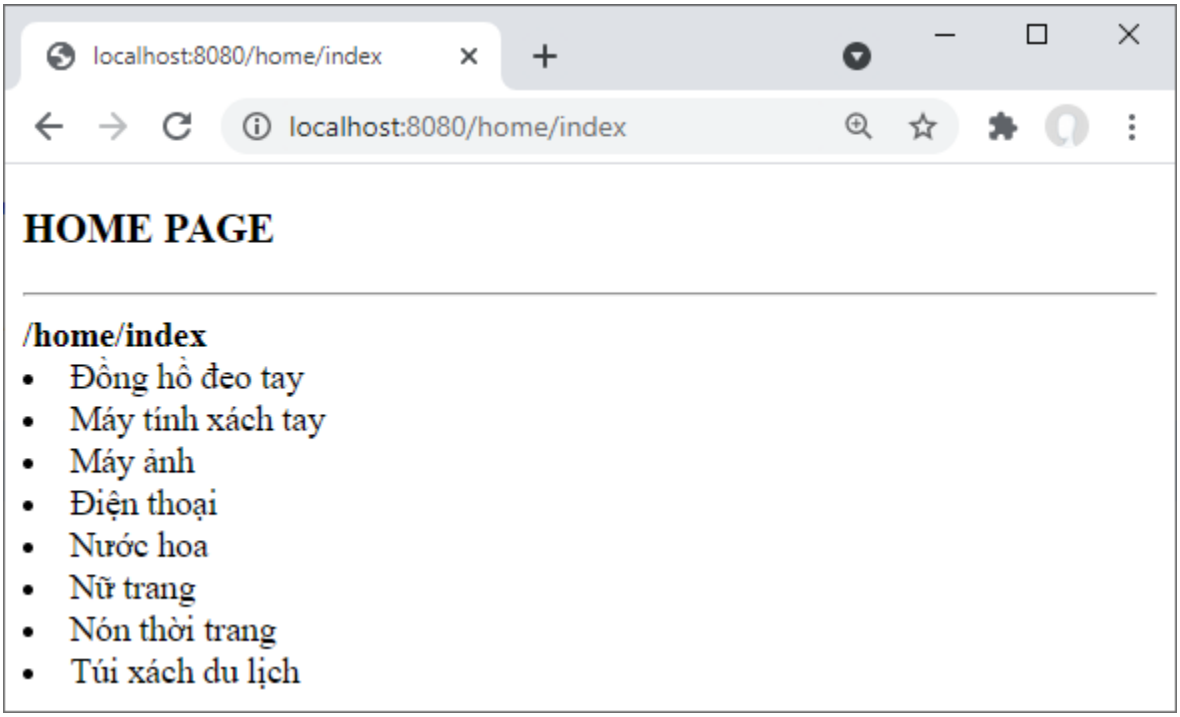
Bài 3 (1 điểm)

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.

PHẦN II

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.

```
@Service
public class GlobalInterceptor implements HandlerInterceptor{
        @Autowired
        CategoryDAO dao;

        @Override
        public boolean preHandle(HttpServletRequest request,
                HttpServletResponse response, Object handler) throws Exception {
                request.setAttribute("uri", request.getRequestURI());
                return true;
        }

        @Override
        public void postHandle(HttpServletRequest req, HttpServletResponse res,
                                Object handler, ModelAndView mv) throws Exception {
                req.setAttribute("categories", dao.findAll());
        }
}
```

2. **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 InterConfig implements WebMvcConfigurer{
```

```
        @Autowired
        GlobalInterceptor global;

        @Override
        public void addInterceptors(InterceptorRegistry registry) {
                registry.addInterceptor(global)
                        .addPathPatterns("/**")
                        .excludePathPatterns("/assets/**");
        }
}
```

3. 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";
        }
}
```

4. View home/index.jsp

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

```
<%@ page pageEncoding="utf-8"%>
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
</head>
<body>
        <h3>HOME PAGE</h3>
        <hr>
        <jsp:include page="_global.jsp"/>
```

```
</body>
</html>
```

5. Partial View _global.jsp: Hiển thị url và categories đã đặt vào trong request

```
<%@ page pageEncoding="utf-8"%>
<%@ taglib uri="http://java.sun.com/jstl/core_rt" prefix="c" %>

<b>${uri}</b>
<c:forEach var="item" items="${categories}">
        <li>${item.name}</li>
</c:forEach>
```

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
  - ***/account/edit***
  - ***/account/chgpwd***
  - ***/order/*****
  - ***Ngoại trừ /assets/*****
- Phải đăng nhập với **vai trò admin** thì mới truy xuất được các url sau
  - ***/admin/*****
  - ***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 {

            String uri = request.getRequestURI();
            Account user = session.get("user"); // lấy từ session

            String error = "";
            if(user == null) { // chưa đăng nhập
                error = "Please login!";
            }
            // không đúng vai trò
            else if(!user.isAdmin() && uri.startsWith("/admin/")) {
                error = "Access denied!";
            }
            if(error.length() > 0) { // có lỗi
                session.set("security-uri", uri);
                response.sendRedirect("/account/login?error=" + error;
                return false;
            }
            return true;
        }
}
```

InterceptorConfig

```
@Configuration
public class InterConfig implements WebMvcConfigurer{
        @Autowired
        AuthInterceptor auth;

        @Override
        public void addInterceptors(InterceptorRegistry registry) {
            registry.addInterceptor(auth)
                .addPathPatterns("/account/edit",
                                 "/account/chgpwd", "/order/**", "/admin/**")
                .excludePathPatterns("/assets/**", "/admin/home/index");
        }
}
```

AccountController

```
@Controller
public class AccountController {
	@Autowired
	AccountDAO dao;

	@Autowired
	SessionService session;

	@GetMapping("/account/login")
	public String login() {
		return "account/login";
	}
	@PostMapping("/account/login")
	public String login(Model model,
			@RequestParam("username") String username,
			@RequestParam("password") String password) {
		try {
			Account user = dao.getOne(username);
			if(!user.getPassword().equals(password)) {
				model.addAttribute("message", "Invalid password");
			} else {
				String uri = session.get("security-uri");
				if(uri != null) {
					return "redirect:" + uri;
				} else {
					model.addAttribute("message", "Login succeed");
				}
			}
		} catch (Exception e) {
			model.addAttribute("message", "Invalid username");
		}
		return "account/login";
	}
}
```

Login form login.jsp

```
<h3>LOGIN</h3>
<b><i>${message}${param.error}</i></b>
<form action="/account/login" method="post">
	<input name="username"  placeholder="Username?">
```

```
        <input name="password" placeholder="Password?">
        <button>Login</button>
</form>
```

Bài 6 (1 điểm)

Giảng viên cho thêm