

## MỤC TIÊU:

Kết thúc bài thực hành này bạn có khả năng

- ✓ Định nghĩa, khai báo và sử dụng bean
- ✓ Sử dụng bean CommonsMultipartResolver để upload file
- ✓ Sử dụng bean JavaMailSenderImpl để gửi email thông qua Gmail

## PHẦN I

### Bài 1 (2 điểm)

Hãy tạo bean User gồm 2 thuộc tính username và password. Khai báo bean vào file cấu hình spring-config-bean.xml sau đó tạo HomeController và sử dụng @Autowired tiêm bean vào để sử dụng.

Bước 1: Định nghĩa bean User

Bean User có 2 thuộc tính là username và password.

```
package poly.bean;

public class User {
    private String username;
    private String password;

    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
}
```

## Bước 2: Khai báo bean

Khai báo bean sau vào file spring-config-mvc.xml hoặc tạo một file spring-config-bean.xml cùng cấu trúc như spring-config-mvc.xml chứa khai báo sau.

```
<bean id="user" class="poly.bean.User">
    <property name="username" value="fpoly"/>
    <property name="password" value="thuchocthucnghiep"/>
</bean>
```

## Bước 3: Tiêm bean vào Controller

Sử dụng @Autowired để tiêm bean vào HomeController sau đó đặt nó vào ModelAttribute bằng cách sử dụng @ModelAttribute("user")

```
@Controller
@RequestMapping("/home/")
public class HomeController {
    @Autowired
    User user;
    @RequestMapping("index")
    public String index(ModelMap model) {
        return "home/index";
    }
    @ModelAttribute("user")
    public User getUser() {
        return user;
    }
}
```

## Bước 4: Hiển thị thông tin của bean lên view

Tạo view index.jsp và hiển thị thông tin của bean user như sau

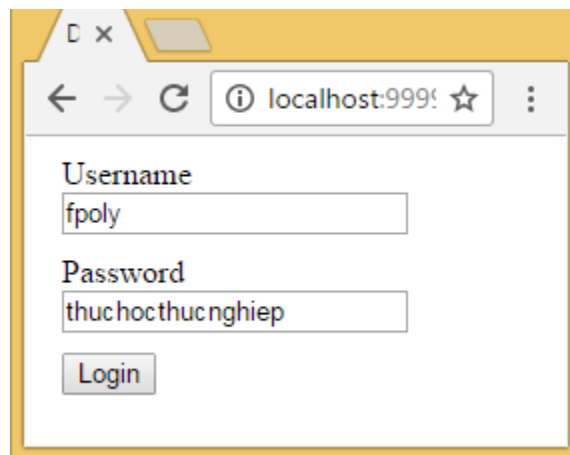
```
<%@ page pageEncoding="utf-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8"/>
    <title>Dependance Injection</title>
    <style>
        .form-group{
```

```

        margin: 10px;
    }
</style>
</head>
<body>
    <form>
        <div class="form-group">
            <div>Username</div>
            <input name="id" value="${user.username}">
        </div>
        <div class="form-group">
            <div>Password</div>
            <input name="id" value="${user.password}">
        </div>
        <div class="form-group">
            <button>Login</button>
        </div>
    </form>
</body>
</html>

```

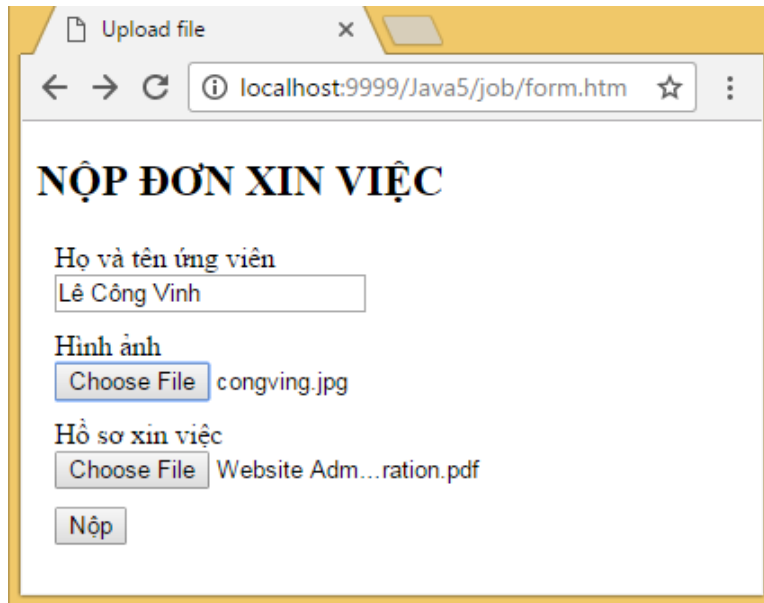
Bước 5: Chạy home/index.htm sẽ thấy thông tin của bean user hiển thị lên form



## Bài 2 (2 điểm)

Xây dựng trang web xin việc làm như mô tả của 2 hình sau

Khi chạy job/form.htm sẽ hiển thị form xin việc



Upload file

localhost:9999/Java5/job/form.htm

## NỘP ĐƠN XIN VIỆC

Họ và tên ứng viên  
Lê Công Vinh

Hình ảnh  
Choose File congving.jpg

Hồ sơ xin việc  
Choose File Website Adm...ration.pdf

Nộp

Sau khi điền đủ thông tin vào form xin việc và nhấn nút Nộp sẽ hiển thị thông tin như sau



Tiêu đề

localhost:9999/Java5/job/apply.htm

## Thông tin cá nhân



**Lê Công Vinh**

## Hồ sơ xin việc

- File Name: Website Administration.pdf
- File Type: application/pdf
- File Size: 91899

Bước 1: Khai báo bean CommonsMultipartResolver cho phép xử lý upload file. Chú ý: mặc định Spring chỉ cho tổng kích thước các file được upload là 2MB. Để thay đổi kích thước cho phép cần điều chỉnh giá trị thuộc tính maxUploadSize.

```
<bean id="multipartResolver"
      class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
    <!-- maxUploadSize=20MB -->
    <property name="maxUploadSize" value="20971520"/>
</bean>
```

Bước 2: Thư viện

Khi sử dụng bean CommonsMultipartResolver bạn cần phải cung cấp 2 thư viện sau vào thư mục lib.

- ✓ commons-fileupload-1.2.2.jar
- ✓ commons-io-1.3.2.jar

Bước 3: Xây dựng phương thức action hiển thị form xin việc

```
@Controller
@RequestMapping("/job/")
public class JobController {
    @RequestMapping("form")
    public String form() {
        return "job/form";
    }
}
```

Bước 4: Thiết kế form xin việc form.jsp

Form này chứa 3 control (2 file và 1 text) có tên là fullname(text), photo(file) và cv(file). Do có chứa file nên form cần có method là post và enctype phải là multipart/form-data

```
<%@ page pageEncoding="utf-8"%>
<!DOCTYPE html>
<html>
<head>
    <base href="${pageContext.servletContext.contextPath}/">
    <meta charset="utf-8"/>
    <title>Upload file</title>
```

```

<style>
    .form-group{
        margin: 10px;
    }
</style>
</head>
<body>
    <h2>NỘP ĐƠN XIN VIỆC</h2>
    ${message}
    <form action="job/apply.htm" method="post" enctype="multipart/form-data">
        <div class="form-group">
            <div>Họ và tên ứng viên</div>
            <input type="text" name="fullname">
        </div>
        <div class="form-group">
            <div>Hình ảnh</div>
            <input type="file" name="photo">
        </div>
        <div class="form-group">
            <div>Hồ sơ xin việc</div>
            <input type="file" name="cv">
        </div>
        <div class="form-group">
            <button>Nộp</button>
        </div>
    </form>
</body>
</html>

```

Bước 5: Xây dựng phương thức action xử lý file upload

Bổ sung mã sau vào JobController

```

@Autowired
ServletContext context;

@RequestMapping("apply")
public String apply(ModelMap model,
    @RequestParam("fullname") String fullname,
    @RequestParam("cv") MultipartFile cv,
    @RequestParam("photo") MultipartFile photo) {

```

```

    if(photo.isEmpty() || cv.isEmpty()){
        model.addAttribute("message", "Vui lòng chọn file !");
    }
    else{
        try {
            String photoPath = context.getRealPath("/files/"+photo.getOriginalFilename());
            photo.transferTo(new File(photoPath));

            String cvPath = context.getRealPath("/files/"+cv.getOriginalFilename());
            cv.transferTo(new File(cvPath));

            model.addAttribute("photo_name", photo.getOriginalFilename());
            model.addAttribute("cv_name", cv.getOriginalFilename());
            model.addAttribute("cv_type", cv.getContentType());
            model.addAttribute("cv_size", cv.getSize());

            return "job/apply";
        }
        catch (Exception e) {
            model.addAttribute("message", "Lỗi lưu file !");
        }
    }
    return "job/form";
}

```

- ✓ Phương thức action apply() sử dụng MultipartFile để nhận 2 file upload là photo và cv.
- ✓ Để làm việc với đường dẫn của website nên cần thêm ServletContext sau đó sử dụng phương thức getRealPath() để lấy đường dẫn thực từ đường dẫn ảo (tính từ website).
- ✓ Cả 2 file được lưu vào thư mục files. Để chạy được bài này, bạn cần tạo một thư mục có tên là files tại thư mục gốc của website (WebContent).
- ✓ Các thông tin của 2 file được đặt vào model để chia sẻ với view apply.jsp

Bước 6: Xây dựng view apply.jsp hiển thị thông tin file đã upload

```

<%@ page pageEncoding="utf-8"%>
<!DOCTYPE html>
<html>
<head>

```

```

<base href="${pageContext.servletContext.contextPath}/">
<meta charset="utf-8"/>
<title>Tiêu đề</title>
</head>
<body>
    <p>
        <h3>Thông tin cá nhân</h3>
        
        <br>
        <h3>${param.fullname}</h3>
    </p>
    <p>
        <h3>Hồ sơ xin việc</h3>
        <ul>
            <li>File Name: ${cv_name}</li>
            <li>File Type: ${cv_type}</li>
            <li>File Size: ${cv_size}</li>
        </ul>
    </p>
</body>
</html>
    
```

**Chú ý:**

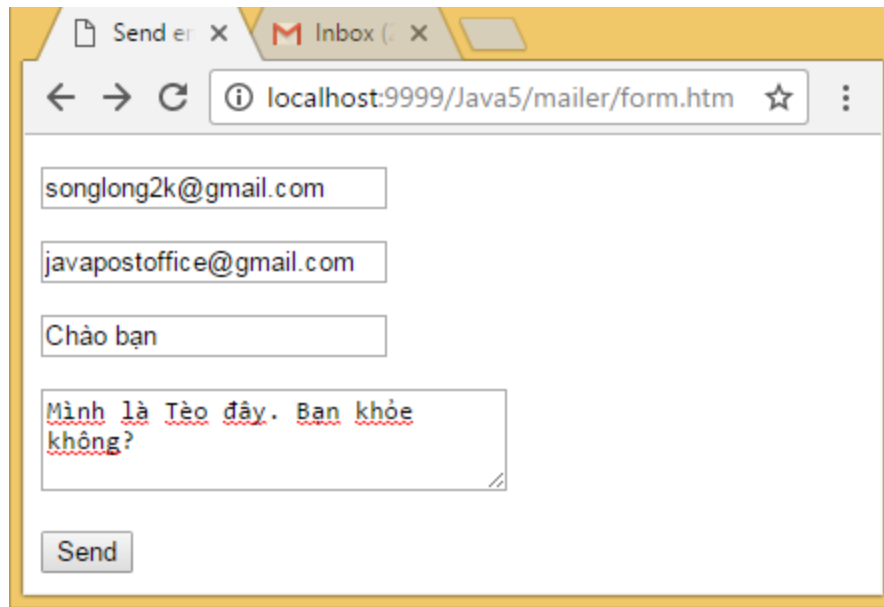
* chỉ đến file đặt trong thư mục files trong thư mục gốc nên dòng mã <base href="\${pageContext.servletContext.contextPath}/"> rất cần thiết, nếu không có sẽ sai đường dẫn hình.*

## PHẦN II

### Bài 3 (3 điểm)

Sử dụng bean JavaMailSender để xây dựng ứng dụng gửi email như hình mô tả sau đây.





Sau khi chạy mailer/form.htm và nhập thông tin đầy đủ sau đó nhấp nút send. Email sẽ được gửi đến người nhận, bạn mở hộp mail của mình sẽ nhận được email.

Bước 1: Khai báo bean JavaMailSender

```
<bean id="mailSender"
    class="org.springframework.mail.javamail.JavaMailSenderImpl">
    <property name="host" value="smtp.gmail.com" />
    <property name="port" value="587" />
    <property name="username" value="user@gmail.com" />
    <property name="password" value="*****" />
    <property name="defaultEncoding" value="UTF-8"/>
    <property name="javaMailProperties">
        <props>
            <prop key="mail.smtp.auth">true</prop>
            <prop key="mail.smtp.socketFactory.class">
                javax.net.ssl.SSLSocketFactory</prop>
            <prop key="mail.smtp.socketFactory.port">587</prop>
            <prop key="mail.smtp.starttls.enable">>false</prop>
            <prop key="mail.debug">true</prop>
        </props>
    </property>
</bean>
```

Trong bài này sử dụng gmail để gửi email nên bạn cần có tài khoản gmail và kích hoạt nó trước khi khai báo 2 thuộc tính username và password.

Bạn cũng cần phải bổ sung các thư viện cần thiết sau đây vào thư mục lib

- ✓ mail.jar
- ✓ activation.jar

Bước 2: Xây dựng phương thức action hiển thị form gửi email

```
@Controller
@RequestMapping("/mailer/")
public class MailerController {
    @RequestMapping("form")
    public String index() {
        return "mailer/form";
    }
}
```

Bước 4: Xây dựng form.jsp chứa form gửi email

```
<%@ page pageEncoding="utf-8"%>
<!DOCTYPE html>
<html>
<head>
    <base href="${pageContext.servletContext.contextPath}/">
    <meta charset="utf-8"/>
    <title>Send email</title>
</head>
<body>
    ${message}
    <form action="mailer/send.htm" method="post">
        <p><input name="from" placeholder="From"></p>
        <p><input name="to" placeholder="To"></p>
        <p><input name="subject" placeholder="Subject"></p>
        <p><textarea name="body" placeholder="Body" rows="3"
cols="30"></textarea></p>
        <button>Send</button>
    </form>
</body>
</html>
```

## Bước 5: Bổ sung phương thức action cho phép gửi email

```

@Autowired
JavaMailSender mailer;

@RequestMapping("send")
public String send(ModelMap model,
    @RequestParam("from") String from,
    @RequestParam("to") String to,
    @RequestParam("subject") String subject,
    @RequestParam("body") String body) {

    try{
        // Tạo mail
        MimeMessage mail = mailer.createMimeMessage();
        // Sử dụng lớp trợ giúp
        MimeMessageHelper helper = new MimeMessageHelper(mail);
        helper.setFrom(from, from);
        helper.setTo(to);
        helper.setReplyTo(from, from);
        helper.setSubject(subject);
        helper.setText(body, true);

        // Gửi mail
        mailer.send(mail);

        model.addAttribute("message", "Gửi email thành công !");
    }
    catch(Exception ex){
        model.addAttribute("message", "Gửi email thất bại !");
    }
    return "mailer/form";
}
    
```

Bước 6: Kiểm tra email của người nhận để xem kết quả

#### Bài 4 (3 điểm)

Xây dựng bean XMailer hỗ trợ cho việc gửi email

Trong bài 3 chúng ta thấy cứ mỗi lần gửi email thì phải thêm JavaMailSender và viết đoạn mã

```
try{
    // Tạo mail
    MimeMessage mail = mailer.createMimeMessage();
    // Sử dụng lớp trợ giúp
    MimeMessageHelper helper = new MimeMessageHelper(mail);
    helper.setFrom(from, from);
    helper.setTo(to);
    helper.setReplyTo(from, from);
    helper.setSubject(subject);
    helper.setText(body, true);

    // Gửi mail
    mailer.send(mail);

    model.addAttribute("message", "Gửi email thành công !");
}
catch(Exception ex){
    model.addAttribute("message", "Gửi email thất bại !");
}
```

Như vậy thì thực sự rất phức tạp. Để đơn giản hóa chúng ta xây dựng một bean chứa phương thức send và mỗi khi cần chỉ việc tiêm bean vào và gọi phương thức là xong.

Bước 1: Xây dựng bean XMailer

```
@Service("mailer")
public class Mailer {
    @Autowired
    JavaMailSender mailer;

    public void send(String from, String to, String subject, String body) {
        try{
            MimeMessage mail = mailer.createMimeMessage();
            MimeMessageHelper helper
                = new MimeMessageHelper(mail, true, "utf-8");
            helper.setFrom(from, from);
            helper.setTo(to);
            helper.setReplyTo(from, from);
            helper.setSubject(subject);
            helper.setText(body, true);
        }
```

```

        mailer.send(mail);
    }
    catch(Exception ex){
        throw new RuntimeException(ex);
    }
}
}

```

Lưu ý: Sử dụng **@Service** hay **@Component** đều được khi đó bean này sẽ tự khai báo mà không cần phải khai báo vào file cấu hình.

Bước 2: Khai báo bean

Mặc dù không phải khai báo vào file cấu hình nhưng phải khai báo package chứa bean Mailer. Mở file spring-config-mvc.xml và bổ sung package vào

```
<context:component-scan base-package="poly.controller,poly.bean"/>
```

Bước 3: Tiêm bean vào Controller để gửi email

```

@Controller
@RequestMapping("/mailer2/")
public class Mailer2Controller {
    @Autowired
    Mailer mailer;

    @RequestMapping("form")
    public String index() {
        return "mailer/form";
    }

    @RequestMapping("send")
    public String send(ModelMap model,
        @RequestParam("from") String from,
        @RequestParam("to") String to,
        @RequestParam("subject") String subject,
        @RequestParam("body") String body) {
        try{
            // Gửi mail
            mailer.send(from, to, subject, body);
        }
    }
}

```

```

        model.addAttribute("message", "Gửi email thành công !");
    }
    catch(Exception ex){
        model.addAttribute("message", "Gửi email thất bại !");
    }
    return "mailer/form";
}
}

```

Chạy mailer2/form.htm bạn sẽ có kết quả tương tự nhưng mã được viết đơn giản hơn rất nhiều.

### **Chú ý:**

- ✓ Phần I và Phần II chỉ áp dụng cho dạy tích hợp. Sinh viên làm phần 1 và phần 2 theo 2 bài khác nhau tương ứng với 2 phần lý thuyết đã dạy trong bài học.
- ✓ Nếu giảng dạy theo phương pháp truyền thống thì sinh viên phải thực hiện tất cả các bài trong một buổi.