

Lập trình web SPRING MVC



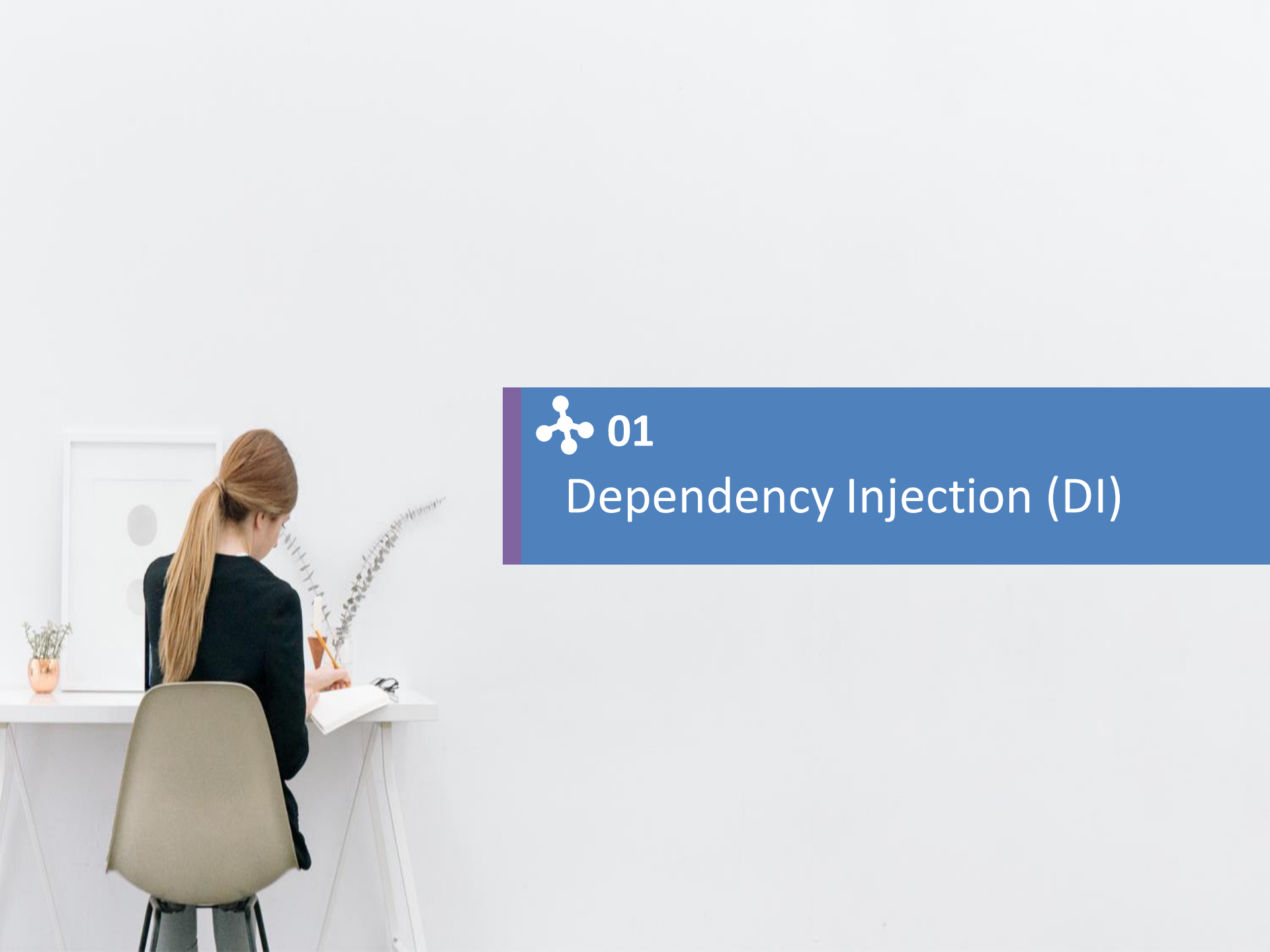
BÀI 5: Bean và Dependency Injection

Mục tiêu :

Nắm vững kỹ thuật lập trình giao diện trong JSP

1. Hiểu Dependency Injection (DI) là gì?
2. Xây dựng và sử dụng Bean
3. Sử dụng @Autowire và @Qualifier
4. Sử dụng bean CommonsMultipartResolver để upload file lên server
5. Sử dụng bean JavaMailSender để gửi email
6. Xây dựng bean gửi email





01

Dependency Injection (DI)

XÉT TÌNH HUỐNG VỀ DEPENDENCY

- ❑ Giả sử chúng ta có lớp Company nắm giữ thông về doanh nghiệp như tên công ty, khẩu hiệu và logo. Trong website chúng ta muốn sử dụng lớp này để làm việc về thông tin doanh nghiệp.
- ❑ Rõ ràng các lớp trong website phụ thuộc vào lớp Company. Vì vậy khi chúng ta muốn thay đổi thông tin của doanh nghiệp thì phải hiệu chỉnh lại mã các lớp trong website và dịch lại ứng dụng
- ❑ Vấn đề đặt ra là làm thế nào để thay đổi thông tin doanh nghiệp mà không phải hiệu chỉnh lại mã của website.

DEPENDENCE INJECTION

- ❑ DI là cách truyền một module vào một module khác thông qua cấu hình XML hay viết mã dưới sự hỗ trợ của DI container
- ❑ Spring framework có trang bị DI container nên có thể thực hiện DI một cách dễ dàng
- ❑ DI được dùng để làm giảm sự phụ thuộc giữa các module, dễ dàng hơn trong việc thay đổi module, bảo trì code và testing.

DEPENDENCY INJECTION

❑ Để cụ thể hóa DI chúng ta xét lớp bean Company gồm 3 thuộc tính

- ❖ Name: tên công ty
- ❖ Slogan: khẩu hiệu
- ❖ Logo: ảnh logo

```
package ptithcm.bean;  
  
public class Company {  
    private String name;  
    private String slogan;  
    private String logo;  
  
    getter/setters  
  
}
```


KHAI BÁO BEAN

- ❑ Mong muốn tạo một đối tượng từ Company chứa thông tin của một doanh nghiệp và được sử dụng trong website nhưng khi thay đổi thông tin sang doanh nghiệp khác thì không phải dịch lại website
- ❑ Để đạt được mong muốn trên bạn cần khai báo bean trong file cấu hình của Spring. DI container sẽ tạo đối tượng khi khởi khởi động.

```
<bean id="ptithcm" class="ptithcm.bean.Company">  
  <property name="name" value="PTITHCM Học Viện Công Nghệ Bưu Chính Viễn Thông"/>  
  <property name="slogan" value=""/>  
  <property name="logo" value="images/logos/ptithcm.png"/>  
</bean>
```

INJECTION (TIÊM)

- ❑ Sau khi bean được khai báo nó có thể được tiêm vào các thành phần khác để sử dụng bằng cách sử dụng **@Autowired** và **@Qualifier**

```
@Controller  
@RequestMapping("/home/")  
public class HomeController {  
    @Autowired  
    Company company;
```

Bean đã được tiêm vào và sẵn sàng phục vụ các action trong Controller

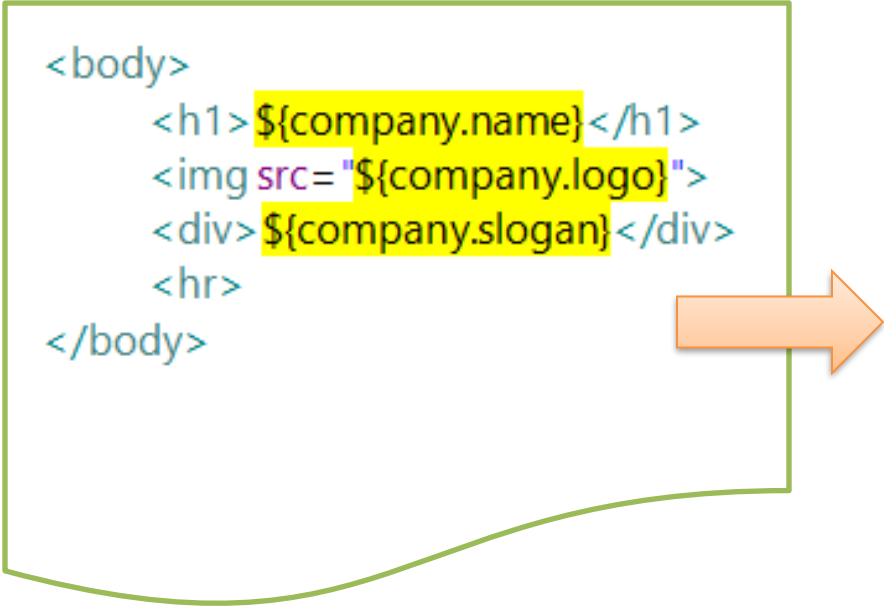
```
    @RequestMapping("index")  
    public String index(ModelMap model) {  
        model.addAttribute("company", company);  
        return "home/index";  
    }  
}
```

Sử dụng bean đã tiêm vào

HIỂN THỊ THÔNG TIN DOANH NGHIỆP

- ❑ View index.jsp được thiết kế để hiển thị thông tin doanh nghiệp.

```
<body>
  <h1> ${company.name} </h1>
  <img src= "${company.logo}">
  <div> ${company.slogan} </div>
  <hr>
</body>
```



❑ @Autowired được sử dụng để tiêm bean vào Controller dưới 3 hình thức sau

- ❖ Tiêm vào **field**
- ❖ Tiêm thông qua **constructor**
- ❖ Tiêm thông qua **setter**

```
@Controller
@RequestMapping("/home/")
public class HomeController {
    @Autowired
    Company company;
```

Tiêm vào field

```
@Controller
@RequestMapping("/home/")
public class HomeController {
    Company company;
    @Autowired
    public HomeController(Company company) {
        this.company = company;
    }
```

Tiêm thông qua constructor

```
@Controller
@RequestMapping("/home/")
public class HomeController {
    Company company;
    @Autowired
    public void setCompany(Company company) {
        this.company = company;
    }
```


Tiêm thông qua phương thức setter

DEPENDANCE INJECTION

- ❑ Bằng cách nào để DI container nhận biết được bean nào để truyền vào cho Controller khi sử dụng **@Autowired**?
- ❑ **@Autowired** sẽ nhận biết bean thông qua **kiểu dữ liệu**.

```
@Controller  
@RequestMapping("/home/")  
public class HomeController {  
    @Autowired  
    Company company;  
}
```

```
<bean id="ptithcm" class="ptithcm.bean.Company">  
    <property name="name" value="PTITHCM Học Viện Công Nghệ Bưu Chính Viễn Thông"/>  
    <property name="slogan" value=""/>  
    <property name="logo" value="images/logos/ptithcm.png"/>  
</bean>
```



DEPENDANCE INJECTION

- ❑ Khi có nhiều bean cùng kiểu dữ liệu thì **@Autowired** không là chưa đủ để xác định bean nào được truyền vào mà cần phải có thêm **@Qualifier** để nhận biết qua id

```
@Controller
@RequestMapping("/home/")
public class HomeController {
    @Autowired
    @Qualifier("ptithcm")
    Company company;

    <bean id="ptithcm" class="ptithcm.bean.Company">
        <property name="name" value="PTITHCM Học Viện Công Nghệ Bưu Chính Viễn Thông"/>
        <property name="slogan" value=""/>
        <property name="logo" value="images/logos/ptithcm.png"/>
    </bean>
    <bean id="ptit" class="ptithcm.bean.Company">
        <property name="name" value="PTIT Học Viện Công Nghệ Bưu Chính Viễn Thông"/>
        <property name="slogan" value=""/>
        <property name="logo" value="images/logos/ptit.png"/>
    </bean>
```

BEAN TỰ KHAI BÁO

- ❑ Lớp bean được chú thích bởi **@Component** hoặc **@Service**, **@Repository** sẽ tự khai báo mà bạn không cần phải khai báo bằng tay vào file cấu hình.
- ❑ Tuy nhiên bạn cần phải khai báo package chứa bean vào

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

*Sử dụng **dấu phẩy** để
phân cách các package.*

VÍ DỤ BEAN TỰ KHAI BÁO

```
package poly.components;
```

```
import org.springframework.stereotype.Component;
```

```
@Component("mailer")
```

```
public class Mailer {
```

```
    public void send(String from, String to, String subject, String body) {
```

```
        // Mã send email đặt ở đây
```

```
    }
```

```
}
```

Mã gửi email sẽ được hướng dẫn viết sau

Bean tự khai báo với id là mailer

```
@Controller
```

```
@RequestMapping("/mailer/")
```

```
public class MailerController {
```

```
    @Autowired
```

```
    JavaMailSender mailer;
```

```
    @RequestMapping("form")
```

```
    public String index() {
```

```
        return "mailer/form";
```

```
    }
```

```
    @RequestMapping("send")
```

```
    public String send(ModelMap model) {
```

```
        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
```

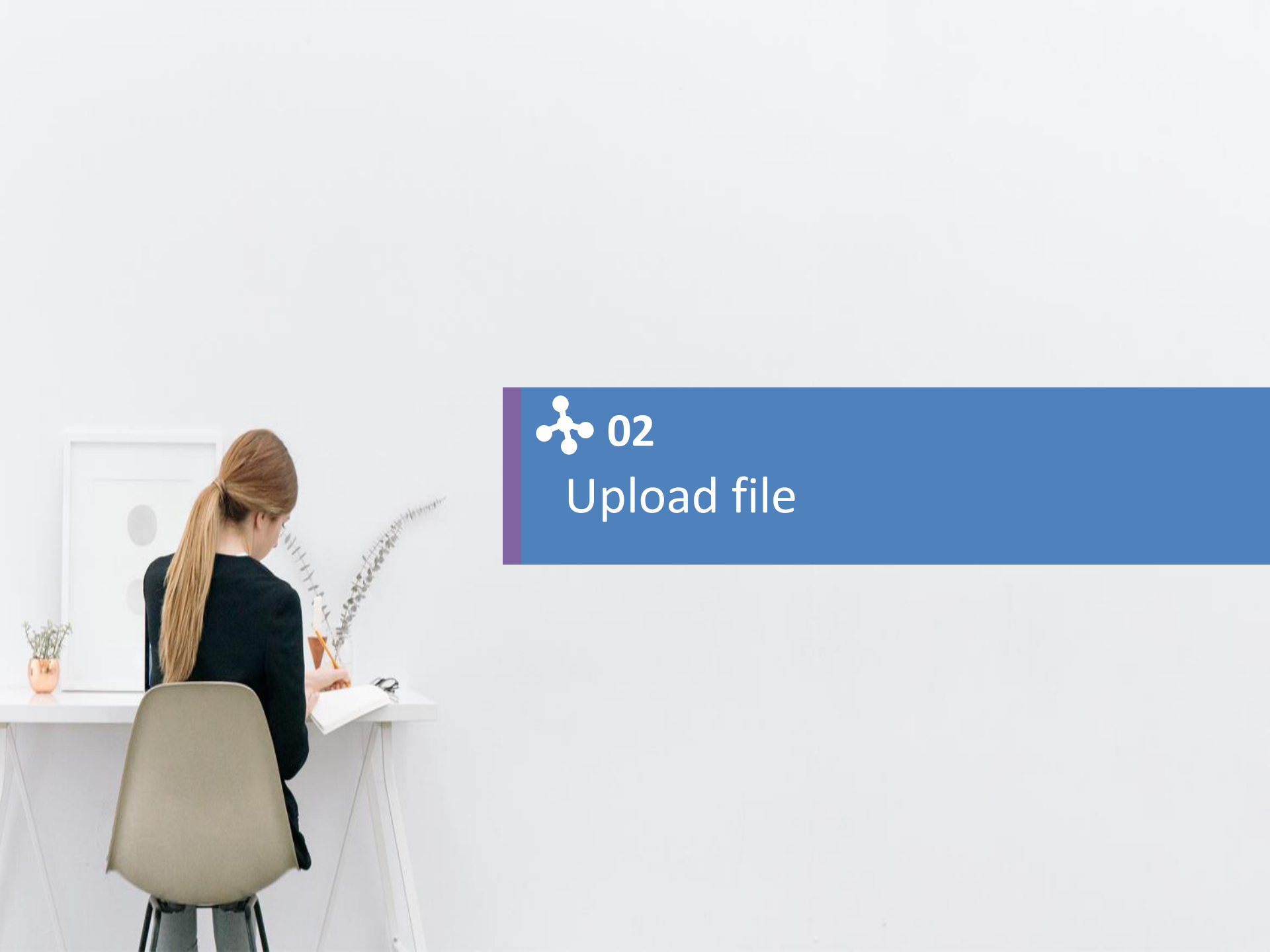
```
!");
```

```
        }
```

```
        catch(Exception ex){
```

```
            model.addAttribute("message", "Gửi email thất bại !");
```

```
        }
```



02

Upload file

- ❑ Upload file là một chức năng quan trọng trong ứng dụng web
- ❑ Các ứng dụng thường gặp
 - ❖ Gửi mail có kèm file
 - ❖ Upload hình đại diện trên facebook, gmail...
 - ❖ Upload video lên Youtube
 - ❖ Nộp hồ sơ xin việc
 - ❖ Nộp bài học lên LMS
 - ❖ ...

THƯ VIỆN VÀ CẤU HÌNH BEAN

- ❑ Để upload file, trước hết bạn cần khai báo bean **CommonsMultipartResolver** vào file cấu hình

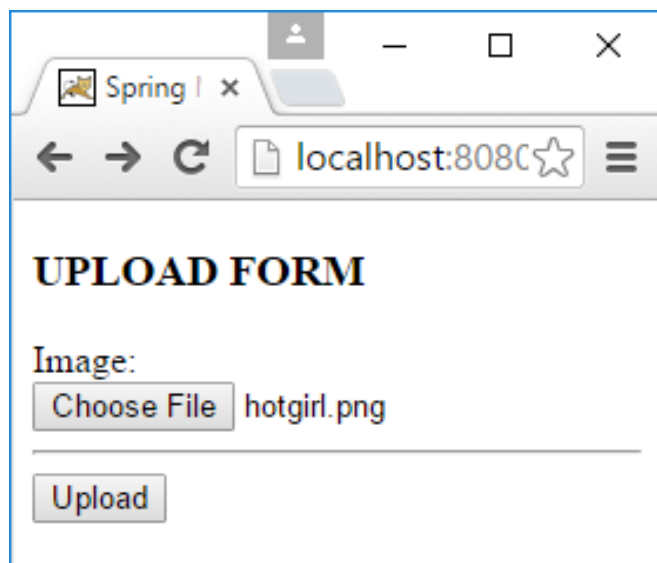
```
<bean id="multipartResolver"  
      class="org.springframework.web.multipart.commons.CommonsMultipartResolver">  
    <!-- maxUploadSize=20MB -->  
    <property name="maxUploadSize" value="20971520"/>  
</bean>
```

- ❖ Mặc định tổng kích thước file là 2MB. Bạn có thể cấu hình thuộc tính **maxUploadSize** để thay đổi thông số này

- ❑ Thư viện cần thiết

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

UPLOAD FILE CASE STUDY



A screenshot of a web browser window with the title 'Spring | x'. The address bar shows 'localhost:8080'. The page content includes the heading 'UPLOAD FORM', a label 'Image:', a 'Choose File' button, the filename 'hotgirl.png', and an 'Upload' button.

Form upload



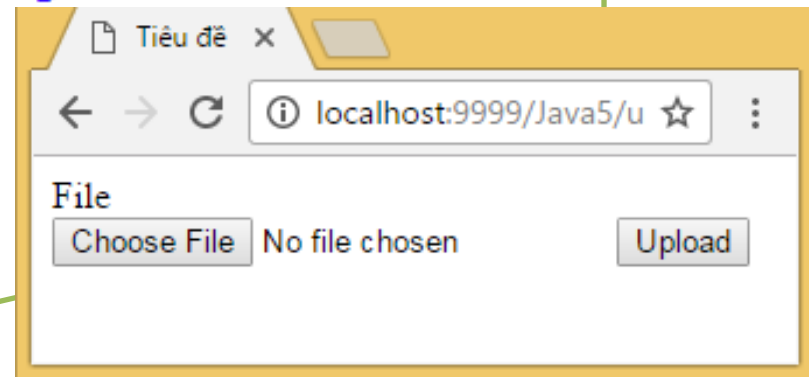
A screenshot of a web browser window with the title 'Spring | x'. The address bar shows 'localhost:8080'. The page content includes the heading 'UPLOAD SUCCESS', an illustration of a woman holding a red apple and a green book, and a list of file details:

- File Name: hotgirl.png
- File Size: 20776
- File Type: image/png

Kết quả upload

FORM UPLOAD FILE

```
${message}  
<form action="uploader/upload.htm"  
      method="post" enctype="multipart/form-data">  
  <div>File</div>  
  <input type="file" name="image">  
  <button>Upload</button>  
</form>
```



❑ Form upload file bắt buộc các thuộc tính

❖ **method**="POST"

❖ **enctype**="multipart/form-data"

XỬ LÝ FILE UPLOAD

```
@RequestMapping("upload")
public String upload(ModelMap model, @RequestParam("image") MultipartFile image) {
    if(image.isEmpty()){
        model.addAttribute("message", "Vui lòng chọn file !");
    }
    else{
        try {
            String path = context.getRealPath("/images/" + image.getOriginalFilename());
            image.transferTo(new File(path));

            model.addAttribute("name", image.getOriginalFilename());
            model.addAttribute("type", image.getContentType());
            model.addAttribute("size", image.getSize());
            return "uploader/success";
        }
        catch (Exception e) {
            model.addAttribute("message", "Lỗi lưu file !");
        }
    }
    return "uploader/form";
}
```

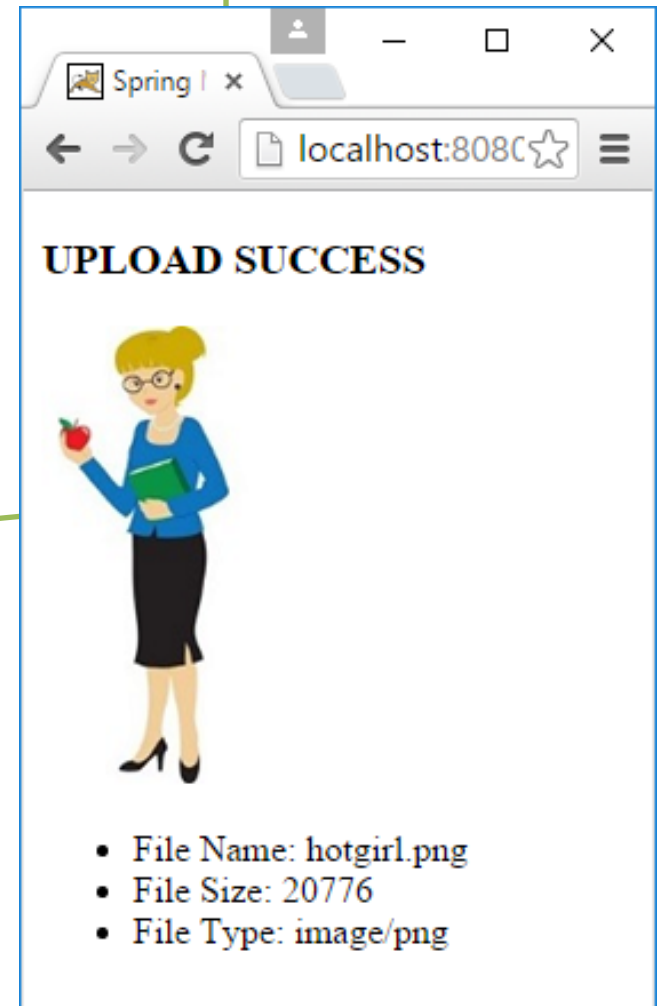
MULTIPARTFILE API

Phương thức	Công dụng
<code>isEmpty()</code>	Kiểm tra xem có file upload không
<code>getOriginalFilename()</code>	Lấy tên file gốc
<code>transferTo(File)</code>	Chuyển file đến đường dẫn mới
<code>getContentType()</code>	Lấy kiểu file
<code>getSize()</code>	Lấy kích thước file
<code>getBytes()</code>	Lấy nội dung file
<code>getInputStream()</code>	Lấy luồng dữ liệu để đọc file

XÂY DỰNG VIEW HIỂN THỊ FILE UPLOAD

```

<ul>
  <li>File Name: ${name}</li>
  <li>File Size: ${size}</li>
  <li>File Type: ${type}</li>
</ul>
```





03

Gửi email

❑ Chức năng gửi email đóng vai trò vô cùng quan trọng trong ứng dụng web

❖ Email kích hoạt tài khoản

Thông thường sau khi đăng ký thành viên thành công hệ thống sẽ gửi cho chúng ta một email chào và có liên kết để kích hoạt tài khoản.

❖ Đơn đặt hàng

Sau khi đặt hàng chúng ta cũng nhận được email báo đơn hàng

❖ Quên mật khẩu

Mật khẩu sẽ được gửi qua email nếu chúng ta cung cấp thông tin hợp lệ

❖ Gửi thông tin cho bạn bè

Khi xem hàng hóa trên internet nếu thấy hàng hóa đó phù hợp với bạn mình thì có thể gửi thông tin hàng hóa đó cho bạn của mình.

❖ ...

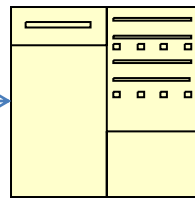
JAVAMAILSENDER

- ❑ Spring cung cấp bean JavaMailSender giúp thực hiện chức năng gửi email rất thuận tiện.
- ❑ Thư viện cần thiết cho bean này gồm
 - ❖ mail.jar
 - ❖ activation.jar
- ❑ Mô hình gửi nhận mail

Smtp server đóng vai trò như bưu điện thông thường. Trong môn học này chúng ta sử dụng gmail để phân phát email



Sender



Smtp Server



Receiver

❑ Khai báo bean JavaMailSender có cấu hình để gửi email thông qua Gmail như sau

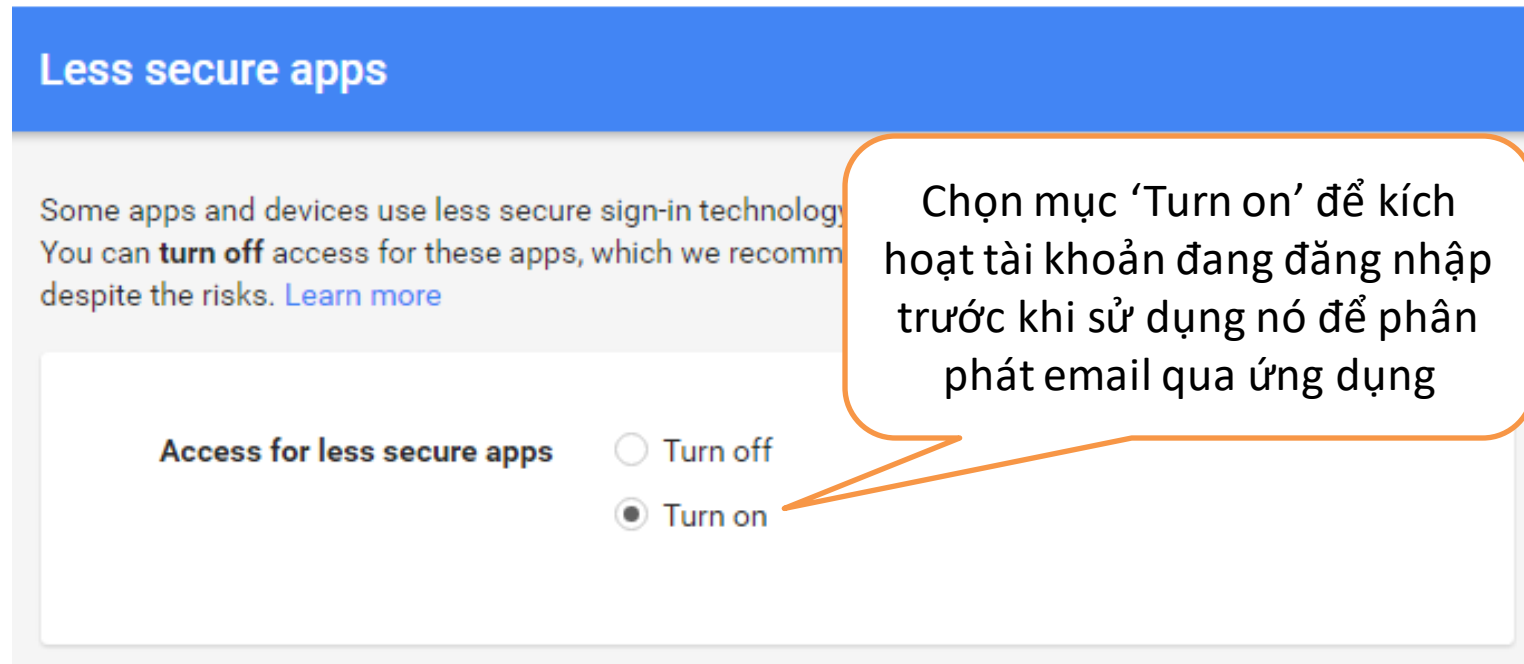
```
<bean id="mailSender"
  class="org.springframework.mail.javamail.JavaMailSenderImpl">
  <property name="host" value="smtp.gmail.com" />
  <property name="port" value="465" />
  <property name="username" value="user@gmail.com" />
  <property name="password" value="*****" />
  <property name="defaultEncoding" 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">465</prop>
      <prop key="mail.smtp.starttls.enable">>false</prop>
      <prop key="mail.debug">true</prop>
    </props>
  </property>
</bean>
```

Tài khoản Smtplib được sử dụng để phát mail đến người nhận

TÀI KHOẢN SMPT – BẬT CHỨC NĂNG GỬI MAIL CỦA GOOGLE

- ❑ Bạn phải đăng ký 1 tài khoản Gmail thông thường sau đó đăng nhập vào gmail và tiến hành kích hoạt thông qua liên kết sau

<https://www.google.com/settings/security/lesssecureapps>



TÀI KHOẢN SMPT – TẠO PASSWORD ỨNG DỤNG

Click vào biểu tượng Gmail

Click Quản lý Tài Khoản Google

The image shows a Google Account security settings page. On the left, a sidebar menu is visible with options: Trang chủ, Thông tin cá nhân, Dữ liệu và cá nhân hóa, **Bảo mật** (highlighted with a red box), Mọi người và chia sẻ, Thanh toán và đăng ký, and Giới thiệu. Below the sidebar, there's a section for 'Quản lý Tài khoản Google' with a button 'Quản lý Tài khoản Google' and a button 'Đăng xuất khỏi tất cả tài khoản'. The main content area is titled 'Đăng nhập vào Google' and contains a section for 'Mật khẩu' (Passwords) with a red box around it. This section includes 'Mật khẩu ứng dụng' (Application passwords) with a red box around it, showing '1 mật khẩu' (1 password). Below this, there's a section 'Các cách mà chúng tôi có thể xác minh bạn chính là chủ sở hữu tài khoản' (Ways we can verify you're the account owner) with options like 'Số điện thoại khôi phục' (Recovery phone number) and 'Email khôi phục' (Recovery email). The browser's address bar shows 'myaccount.google.com/security?gar=1'.

Google Tài khoản

Đăng nhập vào Google

Mật khẩu

Thay đổi lần gần đây nhất: 28 thg 2

Xác minh 2 bước ☒ Bật

Mật khẩu ứng dụng 1 mật khẩu

Các cách mà chúng tôi có thể xác minh bạn chính là chủ sở hữu tài khoản

Các tùy chọn này có thể dùng để đảm bảo rằng bạn chính là người đăng nhập hoặc để liên hệ với bạn nếu có hoạt động đáng ngờ trong tài khoản của bạn

Số điện thoại khôi phục 098 305 18 25

Email khôi phục Thêm địa chỉ email

Quyền riêng tư Điều khoản Trợ giúp

TÀI KHOẢN SMPT – TẠO PASSWORD ỨNG DỤNG

← Mật khẩu ứng dụng

Mật khẩu ứng dụng cho phép bạn đăng nhập vào Tài khoản Google của mình từ các ứng dụng trên các thiết bị không hỗ trợ Xác minh 2 bước. Bạn chỉ cần nhập mật khẩu một lần để bạn không cần phải nhớ nó. [Tìm hiểu thêm](#)

Mật khẩu ứng dụng của bạn

Tên	Đã tạo	Được sử dụng lần cuối
-----	--------	-----------------------

Click chọn ứng dụng

30 thg 3

Chọn ứng dụng và thiết bị bạn muốn tạo mật khẩu ứng dụng.

Chọn ứng dụng

Chọn thiết bị

Thư

Lịch

Danh bạ

YouTube

Khác (Tên tùy chỉnh)

TẠO

Click khác

← Mật khẩu ứng dụng

Mật khẩu ứng dụng cho phép bạn đăng nhập vào Tài khoản Google của mình từ các ứng dụng trên các thiết bị không hỗ trợ Xác minh 2 bước. Bạn chỉ cần nhập mật khẩu một lần để bạn không cần phải nhớ nó. [Tìm hiểu thêm](#)

Mật khẩu ứng dụng của bạn

Tên	Đã tạo	Được sử dụng lần cuối
-----	--------	-----------------------

Gửi mail SMTP	29 thg 3	30 thg 3
---------------	----------	----------

Chọn ứng dụng và thiết bị bạn muốn tạo mật khẩu ứng dụng.

Gửi mail SMTP

Đặt tên cho MK ứng dụng

TẠO

Mật khẩu ứng dụng đã tạo

Mật khẩu ứng dụng dành cho thiết bị của bạn

nqjm feoy jzbf qftf

Copy mật khẩu để lưu sử dụng cho SMTP

Email

securesally@gmail.com

Password

.....

khẩu của bạn bằng mật khẩu gồm 16 ký tự hiển thị bên trên.

Giống như mật khẩu thông thường của bạn, mật khẩu ứng dụng này cấp cho bạn toàn quyền truy cập vào tài khoản Google của bạn. Bạn sẽ không cần nhớ mật khẩu này, do đó đừng viết ra hay chia sẻ với bất kỳ ai.

XONG

SEND E-MAIL CASE STUDY

localhost:9999

Gửi email thành công !

KentPHP2@gmail.com

TamNT360@gmail.com

Chào bạn

Hân hạnh được làm quen với bạn

Send

1

Nhập thông tin hợp lệ vào form và nhấn nút send

2

Đăng nhập vào hộp mail của TamNT360 bạn sẽ thấy một email mới được gửi đến

COMPOSE

Inbox (1)
Starred
Sent Mail
Drafts

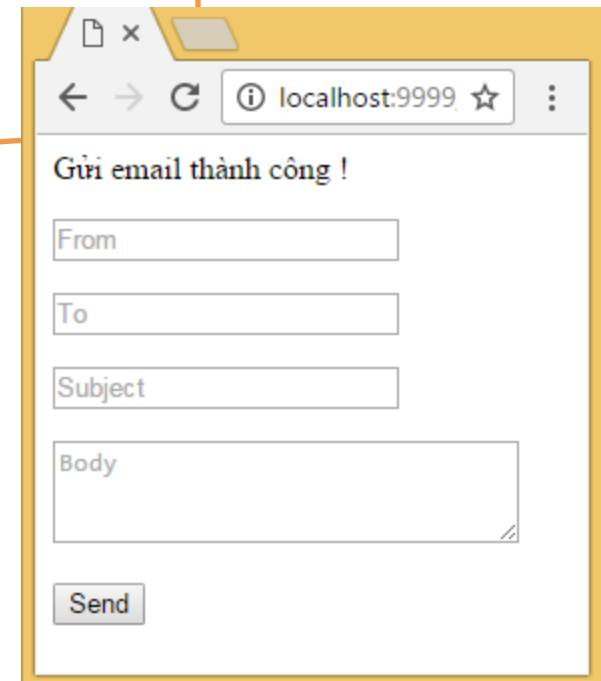
Thiện Tâm

Primary Social 11 new Promotions

<input type="checkbox"/>	☆	KentPHP2@gmail.com	Chào bạn - Hân hạnh được làm quen với bạn	10:28 am
<input type="checkbox"/>	☆	me, Nguyễn (3)	Bài giảng 7,8 - Cảm ơn Tâm !	Nov 9
<input type="checkbox"/>	☆	Facebook	Welcome to Facebook for Dev	Nov 3
<input type="checkbox"/>	☆	Facebook	091 374 57 89 added to your list	Nov 3
<input type="checkbox"/>	☆	Facebook	Just one more step to get started	Nov 3

FORM GỬI MAIL

```
${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>
```



A screenshot of a web browser window with a yellow border. The address bar shows 'localhost:9999'. The page content displays a confirmation message 'Gửi email thành công !' (Email sent successfully!). Below the message are four input fields: 'From', 'To', 'Subject', and 'Body'. The 'Body' field is a larger text area. At the bottom is a 'Send' button. An orange line connects the 'body' attribute in the code block to the 'Body' input field in the browser screenshot.

MAILERCONTROLLER

```
@Controller
@RequestMapping("/mailer/")
public class MailerController {
    @Autowired
    JavaMailSender mailer;
```

Tiêm bean vào để sử dụng

```
@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 !");
```

Tạo một email

Gửi email

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

```
}
```

❑ Trước hết phải upload file

- ❖ `<form action="mailer/send.htm"`
 `method="post" enctype="multipart/form-data">`
- ❖ `public String send(...`
 `@RequestParam("attach") MultipartFile attach)`

❑ Sau đó đính kèm file với phương thức `addAttachment(name, file)`

```
String fileName = attach.getOriginalFilename();  
String path = context.getRealPath("/images/" + fileName);  
helper.addAttachment(fileName, new File(path));
```

JAVAMAILSENDER API

JavaMailSender

Phương thức	Công dụng
createMimeMessage()	Tạo mail
Send(mail)	Gửi mail

MimeMessageHelper

Phương thức	Công dụng
setFrom(email, name)	Cấp thông tin người gửi
setTo(email)	Email người nhận
setCc(emails)	Danh sách email cùng nhận
setBcc(emails)	Danh sách email cùng nhận ẩn danh
setReplyTo(email, name)	Cấp thông tin người nhận phản hồi
setSubject(subject)	Tiêu đề email
setText(body, isHtml)	Nội dung email
addAttachment(name, file)	File đính kèm

XÂY DỰNG BEAN MAILER

```
@Service("mailer")
```

```
public class Mailer {
```

```
    @Autowired
```

```
    JavaMailSender mailer;
```

```
    public void send(String to, String subject, String body) {
```

```
        String from = "javapostoffice@gmail.com";
```

```
        this.send(from, to, subject, body);
```

```
    }
```

```
    public void send(String from, String to, String subject, String body) {
```

```
        this.send(from, to, "", "", subject, body, "");
```

```
    }
```

```
    public void send(String from, String to, String cc, String bcc,
```

```
        String subject, String body, String attachments) {
```

```
        ...
```

```
    }
```

```
}
```

SỬ DỤNG BEAN MAILER

```
@Controller
```

```
@RequestMapping("/mailer/")
```

```
public class MailerController {
```

```
    @Autowired
```

```
    Mailer mailer;
```

Tiêm bean vào

```
    @RequestMapping("send")
```

```
    public String send(ModelMap model,
```

```
        @RequestParam("from") String from,
```

```
        @RequestParam("to") String to,
```

```
        @RequestParam("subject") String subject,
```

```
        @RequestParam("body") String body) {
```

```
        try{
```

```
            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";
```

```
    }
```

```
}
```

Gọi phương thức phù hợp để gửi email

TỔNG KẾT NỘI DUNG BÀI HỌC

- ☑ Tìm hiểu DI
- ☑ Xây dựng, khai báo và sử dụng bean
- ☑ Upload file
- ☑ Gửi email
- ☑ Xây dựng bean Mailer