



LẬP TRÌNH JAVA 5

BÀI 5: BEAN & DI

- ⊙ Hiểu DI là gì?
- ⊙ Xây dựng và sử dụng Bean
- ⊙ Sử dụng @Autowire và @Qualifier
- ⊙ Sử dụng bean CommonsMultipartResolver để upload file lên server
- ⊙ Sử dụng bean JavaMailSender để gửi email
- ⊙ Xây dựng bean gửi email



- ❑ 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.

- ❑ 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.

❑ 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 poly.bean;

public class Company {
    private String name;
    private String slogan;
    private String logo;

    getters/setters
}
```

- ❑ 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="poly" class="poly.bean.Company">  
  <property name="name" value="FPT Polytechnic"/>  
  <property name="slogan" value="Thực Học - Thực Nghiệp"/>  
  <property name="logo" value="images/logos/poly.png"/>  
</bean>
```

- ❑ 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

- ❑ 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>
```





DEMO

Giải thích `home/index.htm`

+ Xây dựng bean

+ Cấu hình

+ Sử dụng `@Autowired`



❑ **@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

- ❑ 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="poly" class="poly.bean.Company">
    <property name="name" value="FPT Polytechnic"/>
    <property name="slogan" value="Thực Học - Thực Nghiệp"/>
    <property name="logo" value="images/logos/poly.png"/>
</bean>
```

- ❑ 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("poly")
    Company company;
```

```
<bean id="poly" class="poly.bean.Company">
    <property name="name" value="FPT Polytechnic"/>
    <property name="slogan" value="Thực Học - Thực Nghiệp"/>
    <property name="logo" value="images/logos/poly.png"/>
</bean>

<bean id="petro" class="poly.bean.Company">
    <property name="name" value="Petrolimex"/>
    <property name="slogan" value="Khởi nguồn mọi chuyển động"/>
    <property name="logo" value="images/logos/Petrolimex.jpg"/>
</bean>
```

- ❑ 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="poly.controller, poly.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
```

```
    }
```

```
}
```

Bean tự khai báo với id là mailer

Mã gửi email sẽ được hướng dẫn viết sau

```
@Controller
```

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

```
public class MailerController {
```

```
    @Autowired
```

```
    Mailer mailer;
```

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

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

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

```
    String to = "b@gmail.com";
```

```
    String subject = "Welcome mail";
```

```
    String body = "FPT Polytechnic";
```

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

```
    return "mail/form";
```

```
}
```

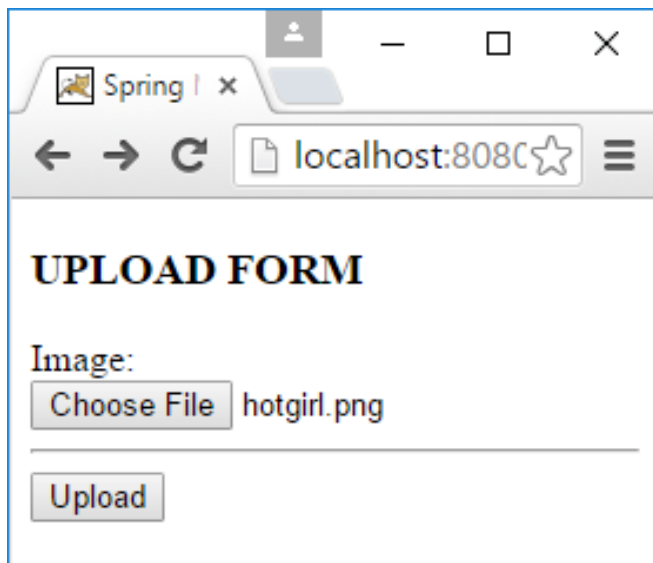
```
}
```

- ❑ 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
 - ❖ ...

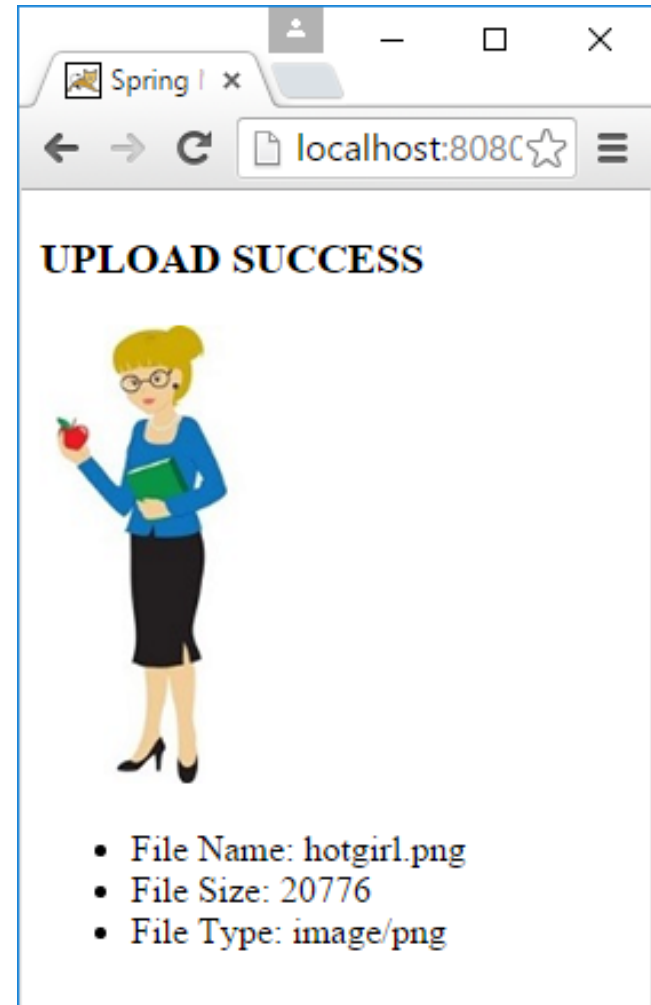
- ❑ Để upload file, trước hết bạn cần khai báo bean **CommonsMultipartResolver** vào file cấu hình

```
<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**

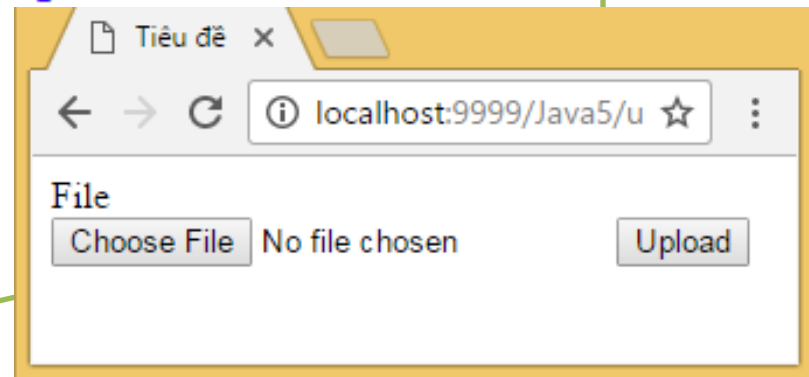


Form upload



Kết quả upload

```
${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"**

```

@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));

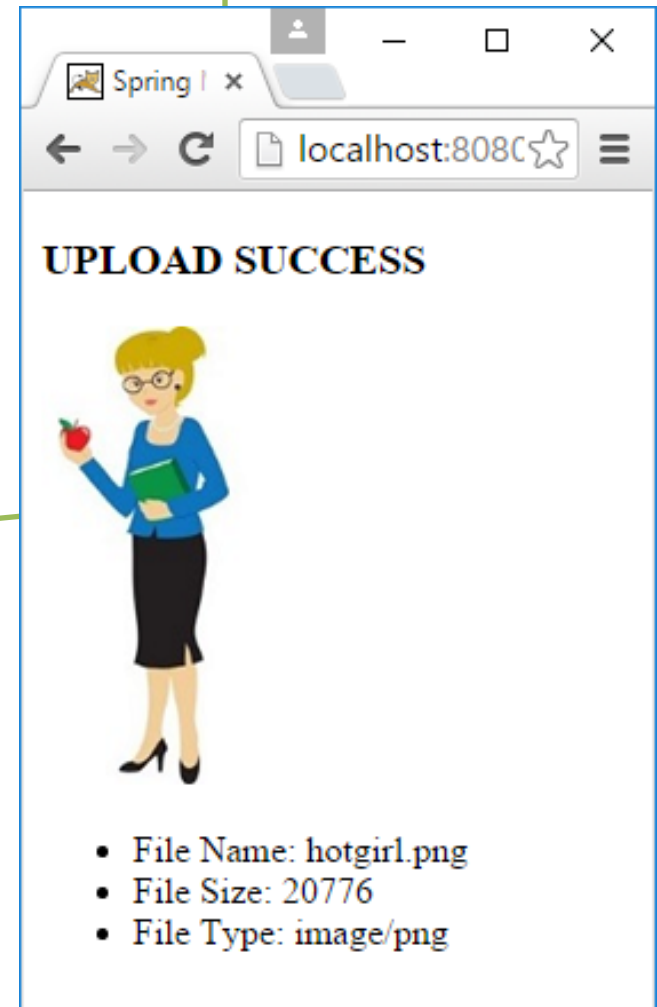
            model.addAttribute("name", image.getOriginalFilename());
            model.addAttribute("type", image.getContentType());
            model.addAttribute("size", image.getSize());
            return "uploader/success";
        }
        catch (Exception e) {
            model.addAttribute("message", "Lỗi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

```

| 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 |

```

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





DEMO

Giải thích upload/form.htm

- + Thư viện
- + Cấu hình
- + form và controller





LẬP TRÌNH JAVA 5

PHẦN 2

❑ 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.

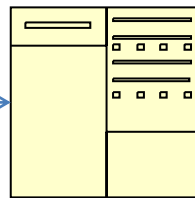
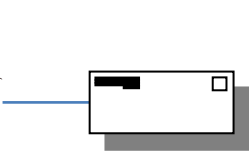
❖ ...

- ❑ 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



Smtip Server



Receiver

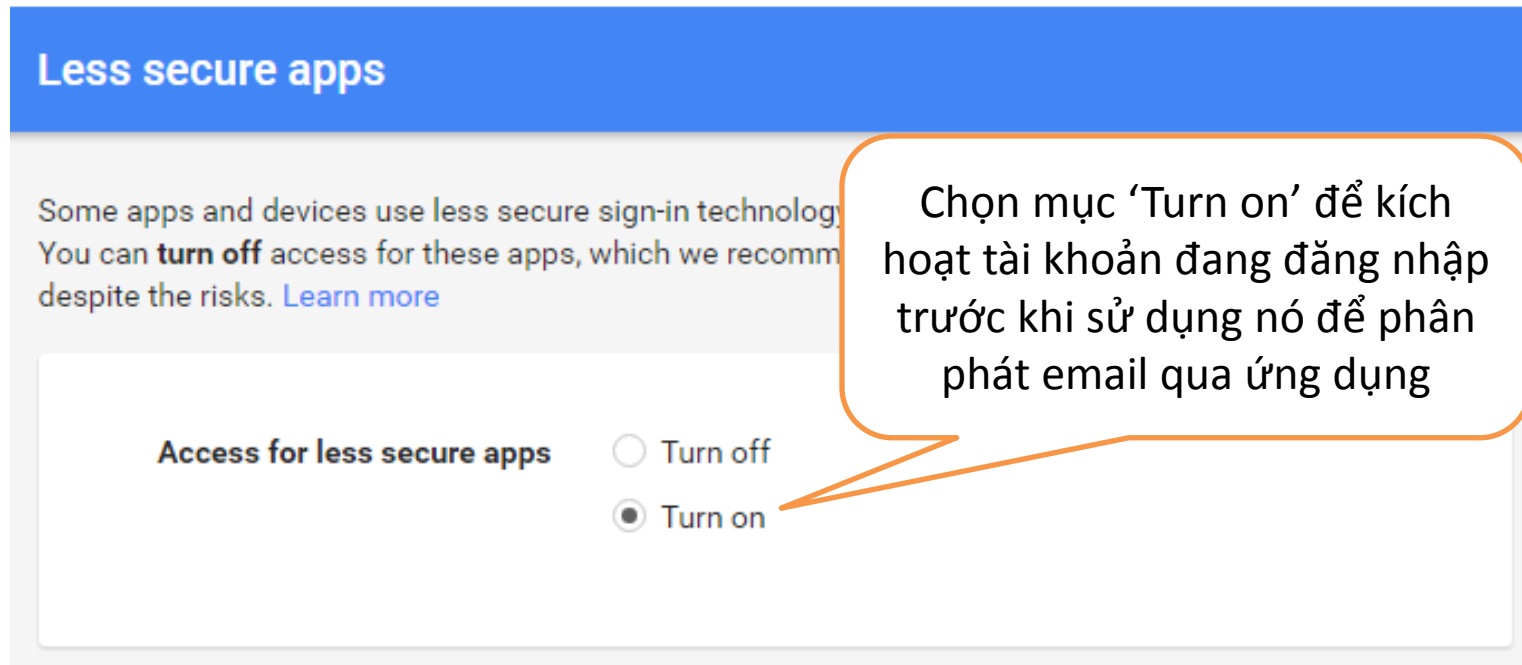
❑ Khai báo bean JavaMailSender có cấu hình để gửi email thông qua Gmail như sau

```
<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

- ❑ 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>



localhost:9999

Gửi email thành công !

KentPHP2@gmail.com

TamNT360@gmail.com

Chào bạn

Hân hạnh được làm quen với bạn

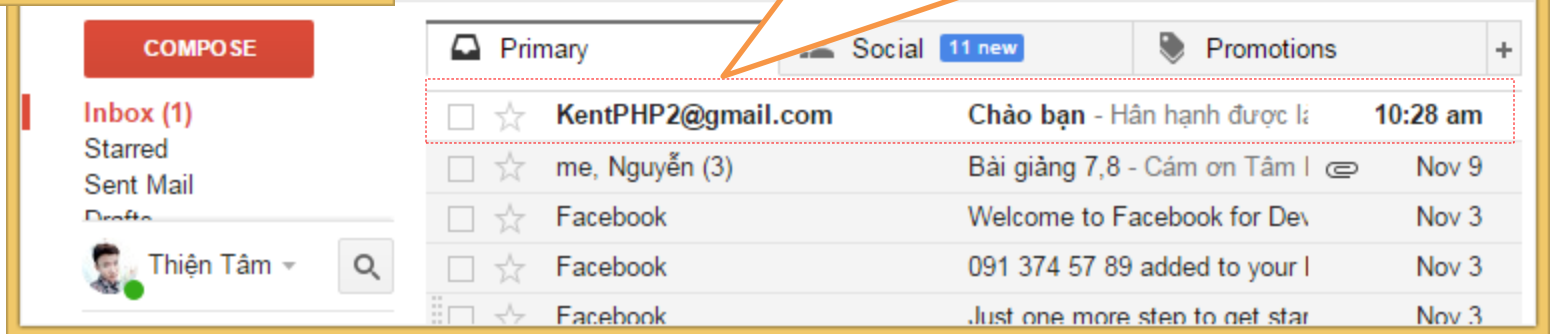
Send

1

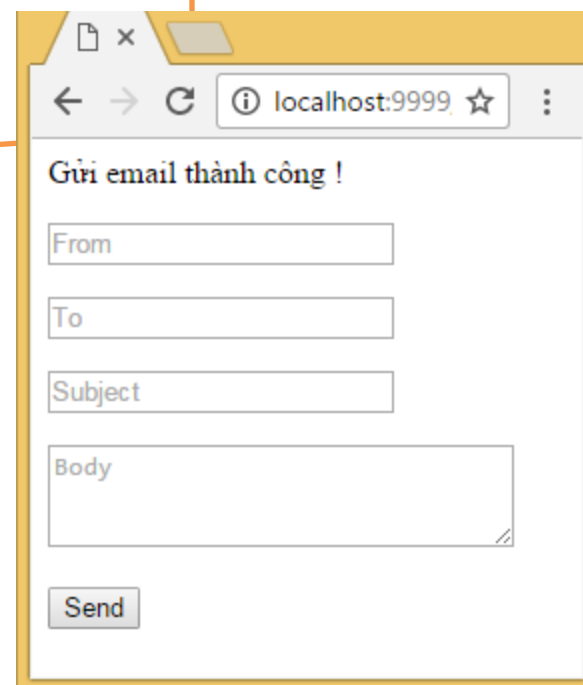
Nhập thông tin hợp lệ vào form và nhấn nút send

2

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



```
${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>
```



The screenshot shows a web browser window with the address bar displaying 'localhost:9999'. The page content includes a success message 'Gửi email thành công !' followed by four input fields labeled 'From', 'To', 'Subject', and 'Body'. Below these fields is a 'Send' button. An orange line connects the 'Send' button in the code snippet to the 'Send' button in the browser screenshot.

@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 !");

}

catch(Exception ex){

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

}

return "mailer/form";

}

}

Gửi email

Tạo một email



DEMO

Giải thích mailer/form.htm

- + Thư viện
- + Cấu hình
- + Form và controller



❑ 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

| 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 |

```
@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) {
```

```
        ...
```

```
    }
```

```
}
```

```
@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



DEMO

Giải thích mailer2/form.htm

+ Mailer

+ @Autowired Mailer



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





Cảm ơn