



SPRING MVC

❖ ThS Nguyễn Nghiệm
❖ 0913.745.789
❖ nghiemn@fpt.edu.vn

NỘI DUNG

- Spring MVC Framework
- Controller
- Dependency Injection
 - ☆ Send email & Upload file
- JdbcTemplate
 - ☆ CRUD
 - ☆ Truy vấn dữ liệu
 - ☆ Đăng ký thành viên
- Ajax & JSON
 - ☆ Tìm kiếm với ajax
- Thảo luận





GIỚI THIỆU

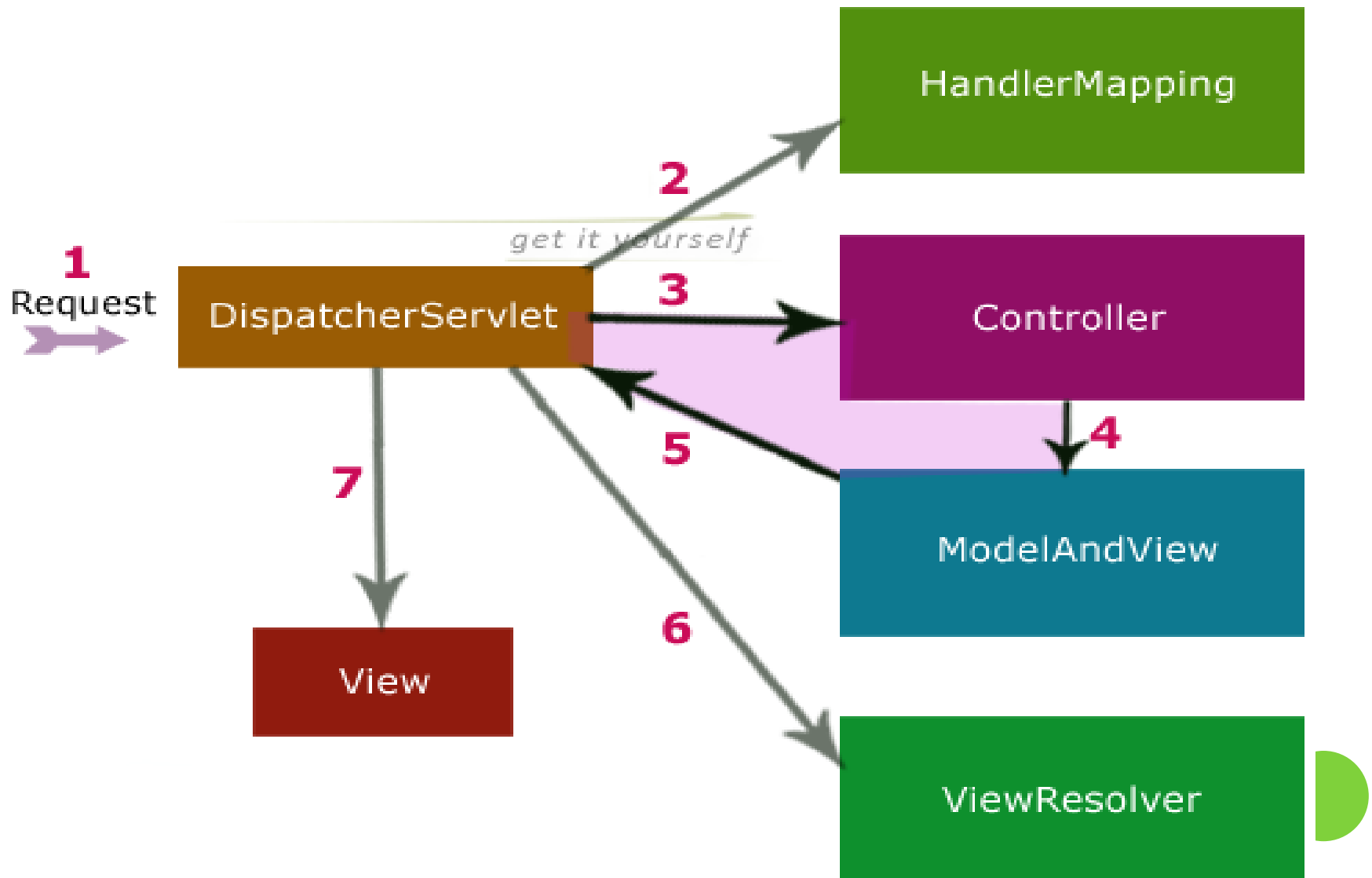
- Giới thiệu Spring MVC
- Giới thiệu quy trình xử lý một yêu cầu từ người dùng
- Tạo project Spring MVC
- Tạo các thành phần M, V, C thực hiện trang HelloWorld
- Đóng gói và triển khai

SPRING FRAMEWORK LÀ GÌ

- Framework mã nguồn mở (Rod Johnson năm 2003).
- Spring rất cần thiết với 3 lý do sau đây
 - ☆ Đơn giản:
 - Spring framework thực sự rất đơn giản vì nó áp dụng các mô hình **POJO** và **POJI**.
 - ☆ Dễ kiểm lỗi:
 - Môi trường giả lập đơn giản. Có thể sử dụng **Console** để kiểm thử các thành phần riêng lẻ.
 - ☆ Không phụ thuộc:
 - Trong Spring các đối tượng **ít phụ thuộc lẫn nhau**, đây là vấn đề cốt lõi của Spring framework -> dễ sửa đổi, nâng cấp bảo trì...

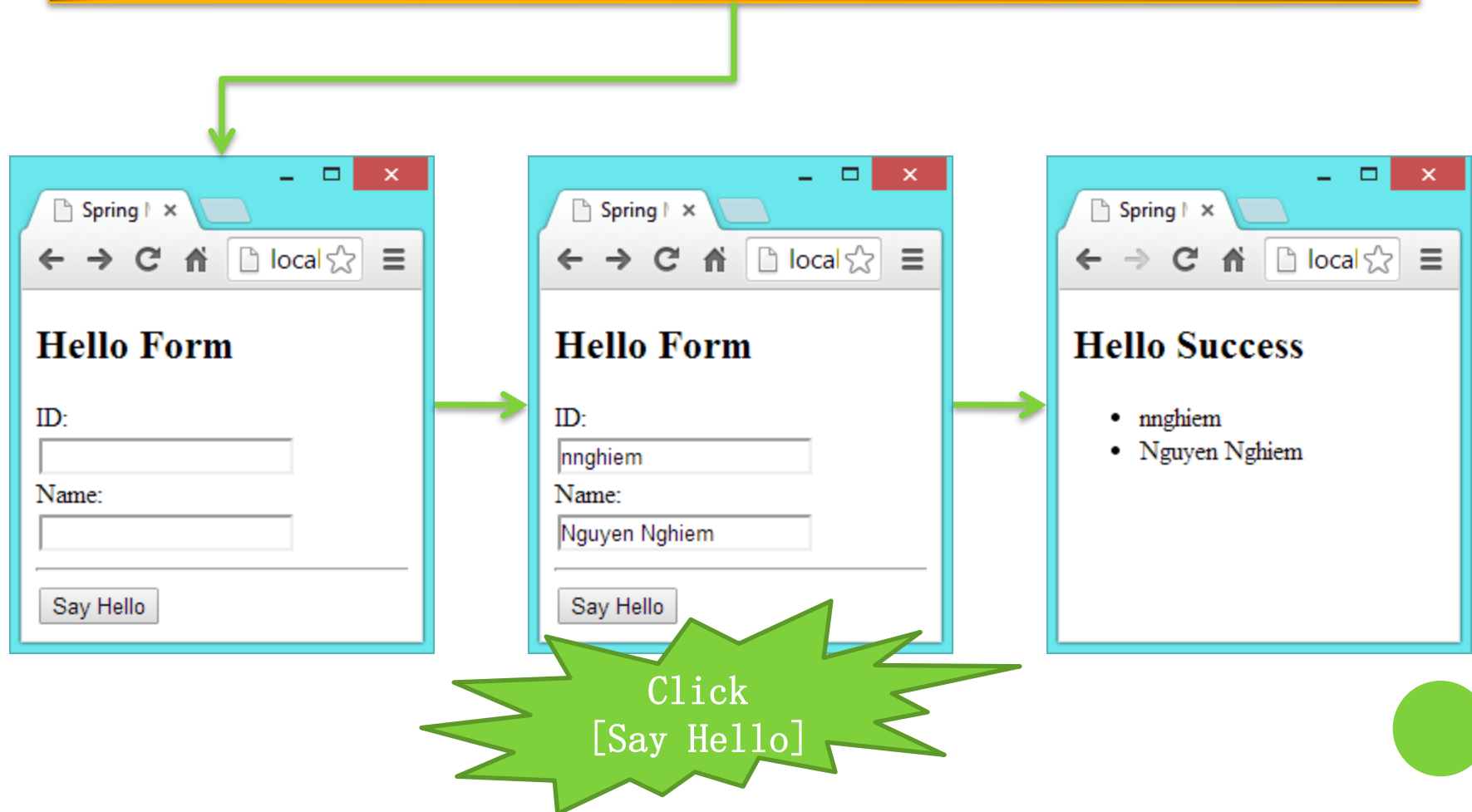


QUI TRÌNH XỬ LÝ YÊU CẦU



SPRING MVC HELLO WORLD

<http://localhost:8080/SpringMVCHello/input.html>



ĐỀ MÔ: SPRING MVC HELLO PROJECT

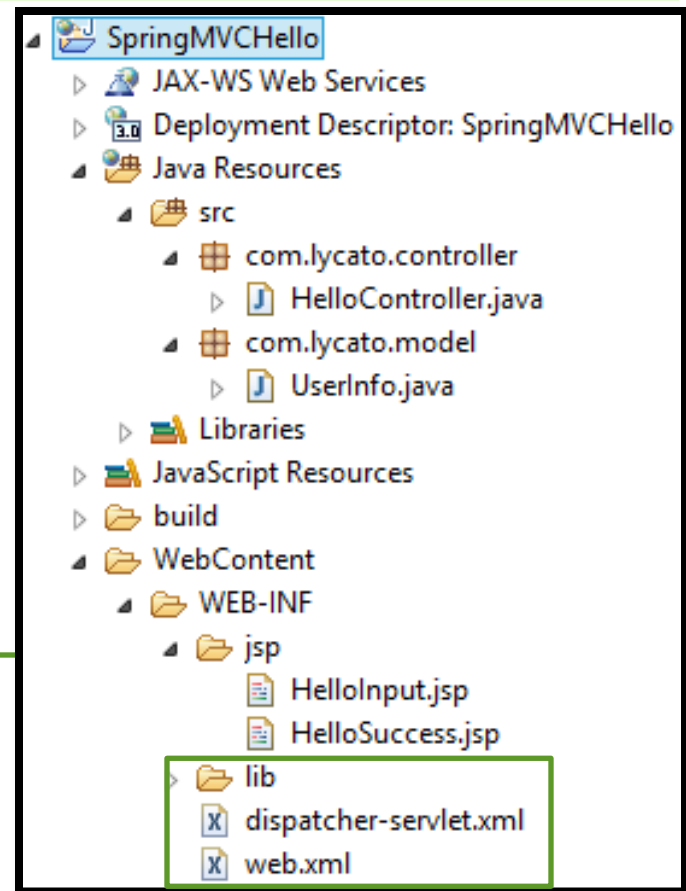
- B1: Tạo web project
- B2: Bổ sung thư viện và các file cấu hình
- B3: Tạo các thành phần M, V, C thực hiện trang HelloWorld
 - ☆ 3.1: Tạo form nhập
 - ☆ 3.2: Tạo Model chia sẻ dữ liệu giữa C và V
 - ☆ 3.3: Tạo View hiển thị kết quả
 - ☆ 3.4: Tạo Controller và Action
- B4: Đóng gói và triển khai
 - ☆ Export War
 - ☆ Deploy vào WebApp của Tomcat
- Chạy thử
 - ☆ <http://localhost:8080/SpringMVCHello/hello.htm>



CẤU TRÚC ỨNG DỤNG SPRINGMVCHello

- HelloController.java
 - ☆ Controller chứa các action
- UserInfo.java
 - ☆ Chứa thông tin user
- HelloInput.jsp
 - ☆ Form nhập thông tin user
- HelloSuccess.jsp
 - ☆ Hiển thị thông tin user đã nhập

- **Web.xml**
 - ☆ Khai báo DispatcherServlet
- **Dispatcher-servlet.xml**
 - ☆ Khai báo cấu hình ứng dụng Spring MVC
- **WEB-INF/lib/*.jar**
 - ☆ Thư viện cần thiết của ứng dụng Spring MVC



THÔNG TIN CẤU HÌNH – WEB.XML

- Khai báo DispatcherServlet tiếp nhận tất cả các yêu cầu có dạng url kết thúc bởi **htm**.

```
<servlet>
  <servlet-name>dispatcher</servlet-name>
  <servlet-class>
    org.springframework.web.servlet.DispatcherServlet
  </servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>dispatcher</servlet-name>
  <url-pattern>*.htm</url-pattern>
</servlet-mapping>
```

THÔNG TIN CẤU HÌNH — DISPATCHER-SERVLET.XML

- Khai báo ViewResolver xử lý view
- Khai báo cho phép ứng dụng sử dụng annotation
- Khai báo nơi tìm kiếm controller

```
<!-- Khai báo ViewResolver -->
<bean id="viewResolver"
      class="org.springframework.web.servlet.view.InternalResourceViewResolver"
      p:prefix="/WEB-INF/jsp/" p:suffix=".jsp" />

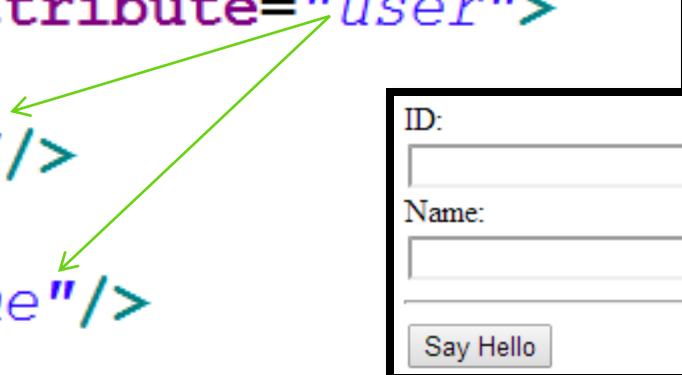
<!-- Khai báo cho phép ứng dụng sử dụng annotation -->
<mvc:annotation-driven />
<context:annotation-config />

<!-- Khai báo nơi tìm kiếm controller -->
<context:component-scan base-package="com.lycato.controller" />
```

HELLOINPUT.JSP

- Hiển thị thông tin user gồm id và name từ thuộc tính user đặt trong model.
- Gửi id và name nhập vào đến action **hello.htm** khi nhấn nút [Say Hello]

```
<form:form action="hello.htm"
  method="post" modelAttribute="user">
  <div>ID:</div>
  <form:input path="id"/>
  <div>Name:</div>
  <form:input path="name"/>
  <hr>
  <input type="submit" value="Say Hello">
</form:form>
```



ID:	<input type="text"/>
Name:	<input type="text"/>
<input type="button" value="Say Hello"/>	

USERINFO.JAVA

- Định nghĩa class mô tả thông tin user gồm 2 thuộc tính là id và name.

```
public class UserInfo {  
    String id;  
    String name;  
  
    public String getId() {  
        return id;  
    }  
    public void setId(String id) {  
        this.id = id;  
    }  
    public String getName() {..  
    public void setName(String name) {..  
}
```

HELLOCONTROLLER.JAVA

- Chứa các phương thức action (được ánh xạ với các action của web).
 - ☆ input.htm -> showForm()
 - ☆ hello.htm -> sayHello()

```
@Controller
public class HelloController {

    @RequestMapping(value="input.htm")
    public String showForm(ModelMap model) {
        model.addAttribute("user", new UserInfo());
        return "HelloInput";
    }

    @RequestMapping(value="hello.htm")
    public String sayHello(
        @ModelAttribute("user") UserInfo user) {
        return "HelloSuccess";
    }
}
```

Chọn view

modelAttribute

Tiếp nhận id, name từ form

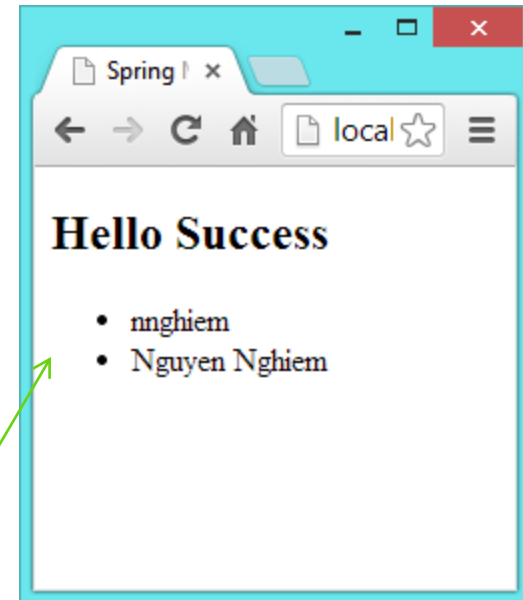
HELLOSUCCESS.JSP

@ModelAttribute("user")

\${user.id}

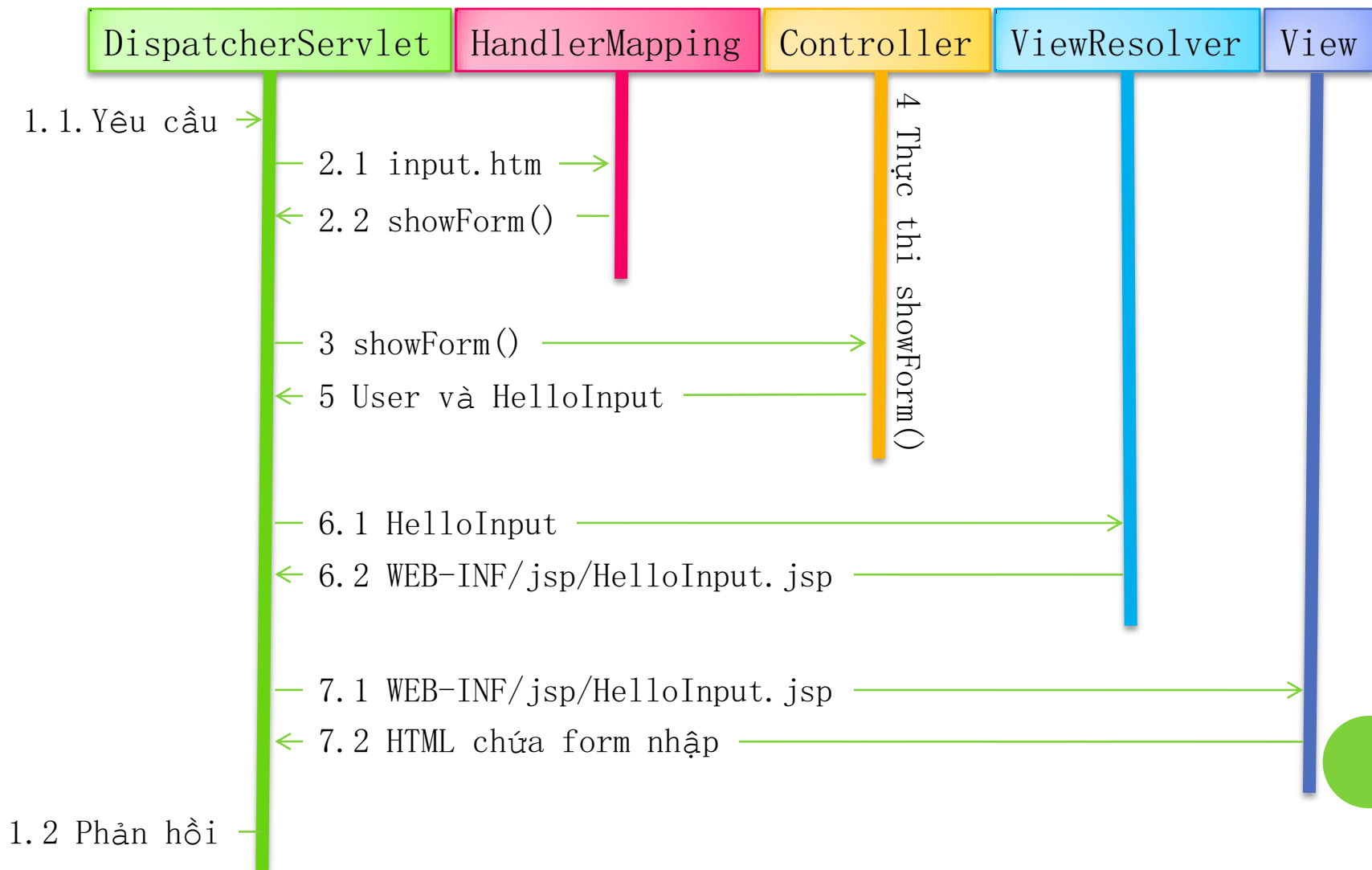
\${user.name}

public String getName() {}



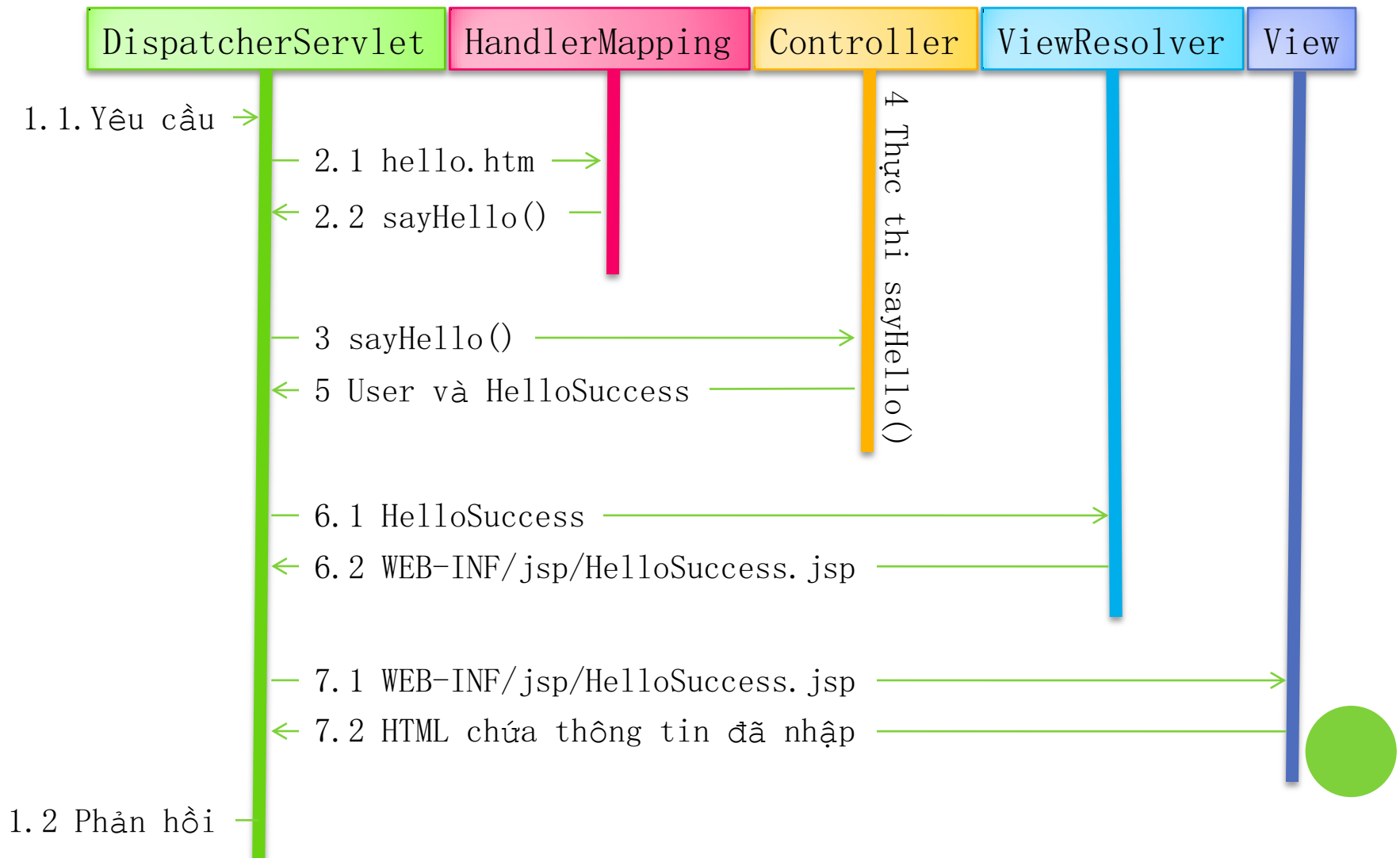
QUI TRÌNH HIỂN THỊ FORM NHẬP

<http://localhost:8080/SpringMVCHello/input.htm>



QUI TRÌNH XỬ LÝ NÚT SAY HELLO

Submit to hello.htm





CONTROLLER

- Tổ chức của một Controller
 - ❖ Khai báo 1 Action
 - ❖ Khai báo nhiều Action
- Các phương pháp tiếp nhận tham số

TỔ CHỨC CONTROLLER

○ @Controller

```
@Controller  
public class HelloController {
```

○ @RequestMapping

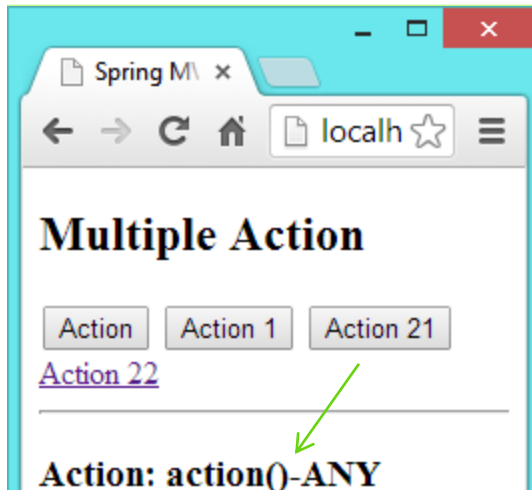
- ☆ Ánh xạ phương thức với 1 action

```
@RequestMapping(value="hello.htm")  
public String sayHello(  
  
@RequestMapping(value="insert.htm",  
    params="insert", method=RequestMethod.POST)  
public String insert(){
```

- ☆ Các thuộc tính quan trọng của @RequestMapping
 - **value**: tên action (**hello.htm**, input.htm...)
 - **method**: phương thức truyền (**POST**, GET...)
 - **params**: yêu cầu có chứa tham số ("**insert**")



FORM NHIỀU ACTION



Khởi động -> "action()-ANY"
Nhập [Action] -> "action()-ANY"
Nhập [Action 1] -> "action1-POST()"
Nhập [Action 21] -> "action21-POST()"
Nhập [Action 22] -> "action22-GET()"

```
<form:form action="myAction.htm" method="post">
  <input type="submit" value="Action">
  <input type="submit" value="Action 1" name="A1">
  <!--A2 POST-->
  <input type="submit" value="Action 21" name="A2">
</form:form>
<!--A2 GET-->
<a href="myAction.htm?A2">Action 22</a>
<hr>
<h3>Action: ${action}</h3>
```

TỔ CHỨC CONTROLLER NHIỀU ACTION

```
@RequestMapping("myAction")
public class MultiActionController {
    @RequestMapping()
    public String action(ModelMap model){
        model.addAttribute("action", "action()-ANY");
        return "MultiAction";
    }
    @RequestMapping(params="A1", method=RequestMethod.POST)
    public String action1(ModelMap model){
        model.addAttribute("action", "action1()-POST");
        return "MultiAction";
    }
    @RequestMapping(params="A2", method=RequestMethod.POST)
    public String action21(ModelMap model){
        model.addAttribute("action", "action21()-POST");
        return "MultiAction";
    }
    @RequestMapping(params="A2", method=RequestMethod.GET)
    public String action22(ModelMap model){
        model.addAttribute("action", "action22()-GET");
        return "MultiAction";
    }
}
```

TIẾP NHẬN THAM SỐ

- Sử dụng JavaBean **UserInfo user**

```
@RequestMapping(value="hello5.htm")  
public String sayHello5(UserInfo user) {
```

- Sử dụng **@RequestParam**

```
@RequestMapping(value="hello3.htm")  
public String sayHello3(@RequestParam("id") String id) {
```

- Sử dụng **@PathVariable**

```
@RequestMapping(value="hello4/{id}")  
public String sayHello4(@PathVariable("id") String id) {
```

- Sử dụng các phương thức của **HttpServletRequest**

- ☆ String **getParameter()**

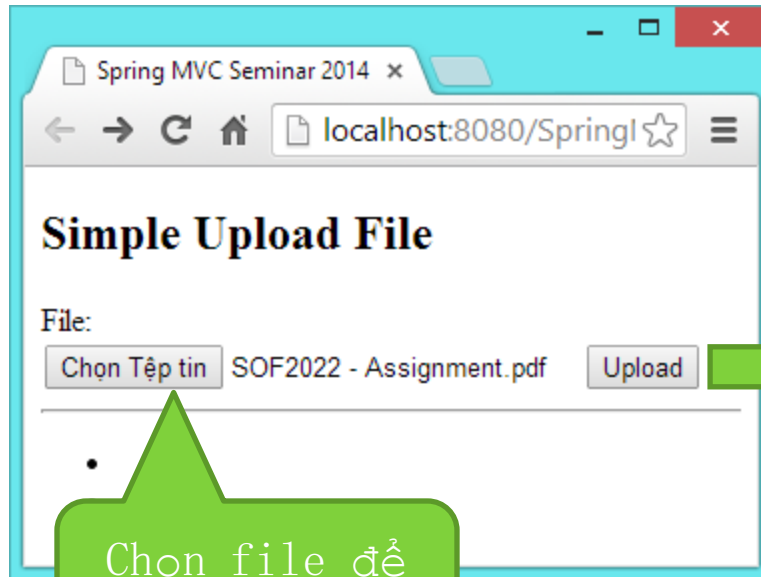
- ☆ String[] **getParameterValues()**

- ☆ Enumeration<String> **getParameterNames()**

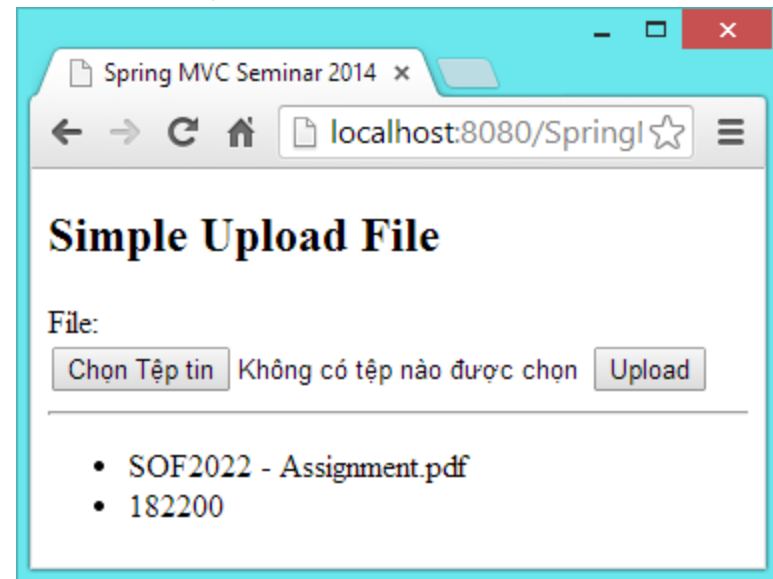
```
@RequestMapping(value="hello2.htm")  
public String sayHello2(HttpServletRequest request) {  
    String id = request.getParameter("id");
```



UPLOADFILE



Chọn file để upload



THỰC HIỆN UPLOAD FILE

- Nhận file upload từ máy khách là hành động tiếp nhận tham số đặc biệt.

- Bước 1: Cấu hình <bean>

```
<bean id="multipartResolver"
      class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
    <!-- maxUploadSize=10MB -->
    <property name="maxUploadSize" value="10485760"/>
</bean>
```

- Bước 2: Thiết kế form

```
<form:form action="single.htm"
           method="post" enctype="multipart/form-data">
```

- Bước 3: Nhận file upload

```
@RequestMapping(value="/single.htm", method=RequestMethod.POST)
public String singleUploadFile(ModelMap model,
    @RequestParam("document")MultipartFile file) throws Exception{
    if(!file.isEmpty()){
        String fileName = file.getOriginalFilename();
        long fileSize = file.getSize();
        file.transferTo(new File("c:/temp/" + fileName));
    }
}
```

UPLOADFILE FORM

```
<form:form action="single.htm"
  method="post" enctype="multipart/form-data">
  <div>File:</div>
  <input type="file" name="document">
  <input type="submit" value="Upload">
  <hr>
  <ul>
    <li>${filename}</li>
    <li>${filesize}</li>
  </ul>
</form:form>
```



UPLOADFILECONTROLLER

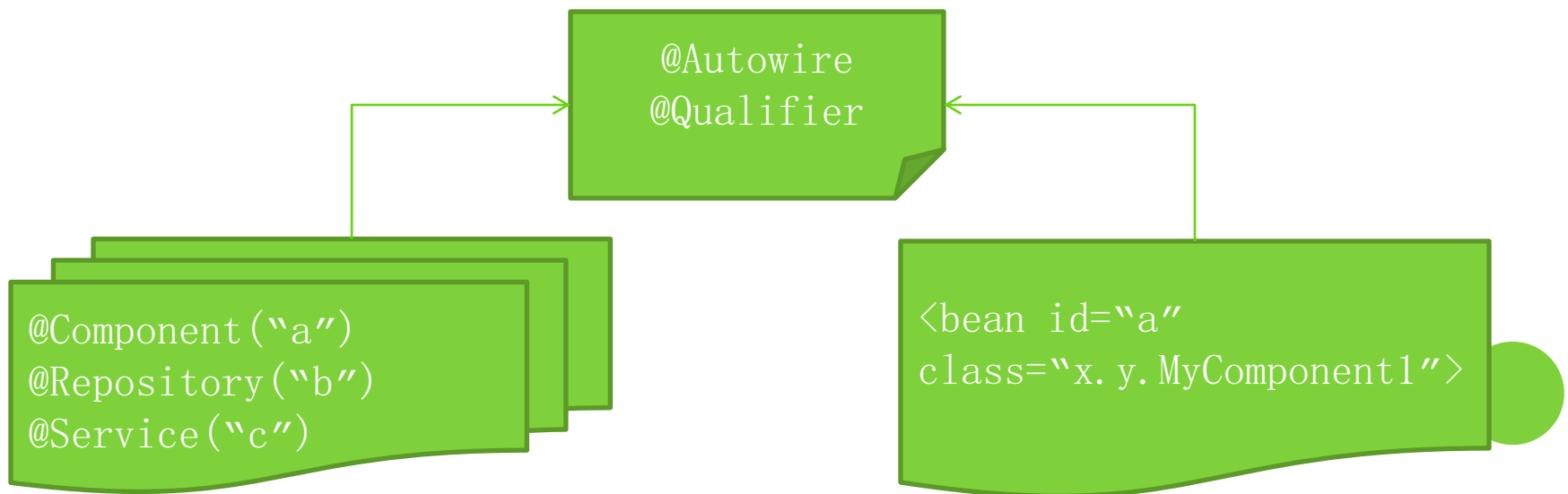
```
@RequestMapping(value="/single.htm", method=RequestMethod.GET)
public String singleUploadFile() {
    return "SingleUploadFile";
}

@RequestMapping(value="/single.htm", method=RequestMethod.POST)
public String singleUploadFile(ModelMap model,
    @RequestParam("document")MultipartFile file) throws Exception{
    if(!file.isEmpty()){
        String fileName = file.getOriginalFilename();
        long fileSize = file.getSize();
        file.transferTo(new File("c:/temp/" + fileName));

        model.addAttribute("filename", fileName);
        model.addAttribute("filesize", fileSize);
    }
    return "SingleUploadFile";
}
```

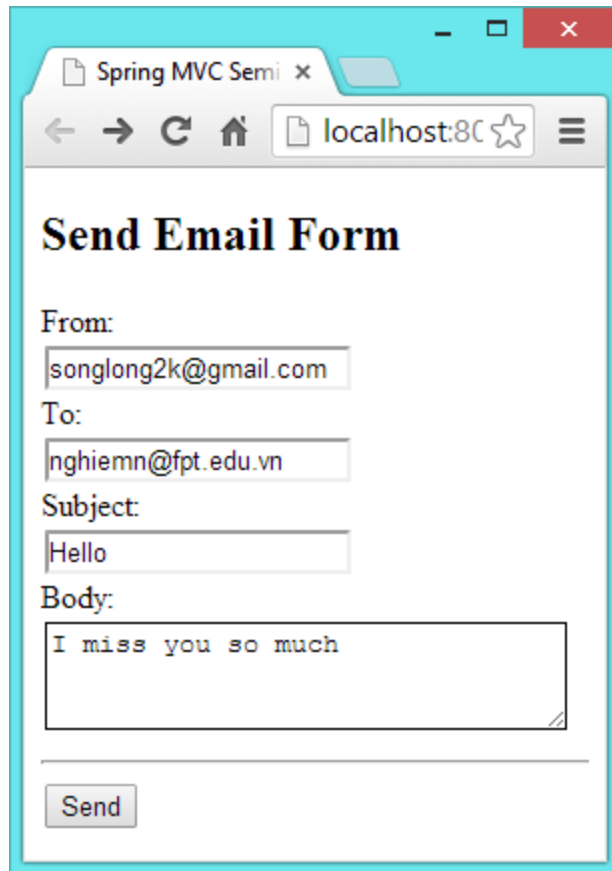
DEPENDENCY INJECTION

- Tạo và nhúng đối tượng vào chương trình để sử dụng khi cần thiết sử dụng các annotation `@Autowire` kết hợp với `@Qualifier`
- Các lớp có thể nhúng vào phải được định nghĩa với các annotation `@Component`, `@Repository` hay `@Service` hoặc khai báo `<bean>` trong file cấu hình.



ĐỀ MÔ: DEPENDENCY INJECTION

- Ứng dụng gửi email cần **JavaMailSender** của Spring. Chúng ta cần “tiêm” bean này vào Controller để thực hiện việc gửi email.



Spring MVC Semi x

localhost:80

Send Email Form

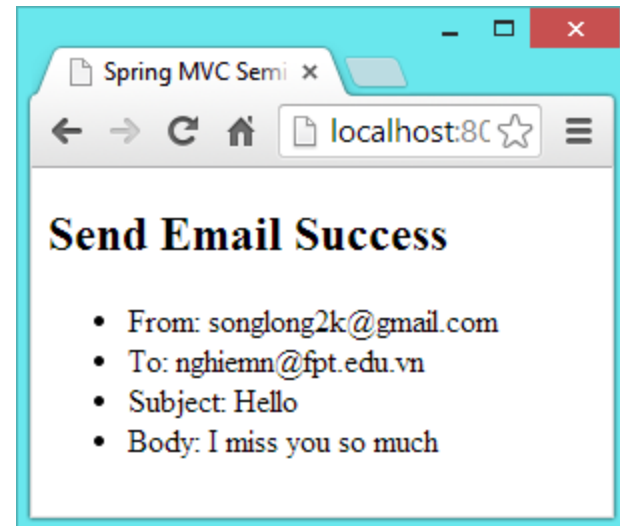
From:
songlong2k@gmail.com

To:
nghiemn@fpt.edu.vn

Subject:
Hello

Body:
I miss you so much

Send

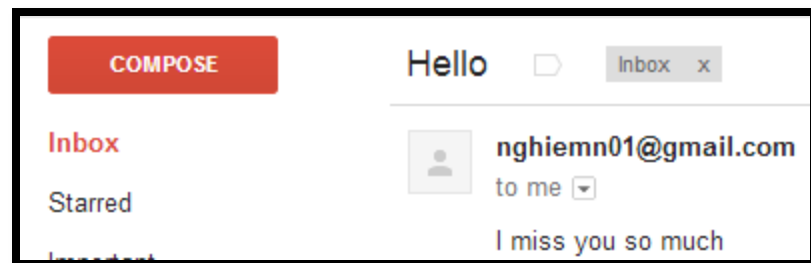


Spring MVC Semi x

localhost:80

Send Email Success

- From: songlong2k@gmail.com
- To: nghiemn@fpt.edu.vn
- Subject: Hello
- Body: I miss you so much



COMPOSE

Hello

Inbox x

Inbox

Starred

nghiemn01@gmail.com

to me

I miss you so much

CẤU HÌNH <BEAN>

- Sử dụng JavaMailSender được cung cấp bởi Spring để thực hiện gửi email thông qua tài khoản gmail.
- Công việc phải làm là khai báo <bean> và nhúng bean vào Controller để sử dụng.

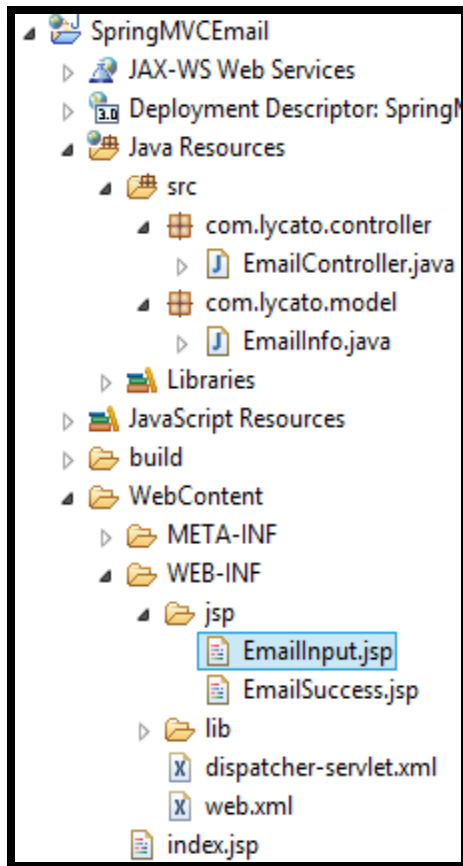
```
<bean id="mailSender"  
    class="org.springframework.mail.javamail.JavaMailSenderImpl">  
    <property name="host" value="smtp.gmail.com" />  
    <property name="port" value="465" />  
    <property name="username" value="nghiemn01@gmail.com" />  
    <property name="password" value="songlong" />  
</bean>
```



SỬ DỤNG <BEAN>

- JavaMailSender là một API được hỗ trợ sẵn trong Spring giúp bạn gửi mail thuận tiện

Injection



```
@Autowired
private JavaMailSender mailSender;

@RequestMapping(value="send.htm", method=RequestMethod.POST)
public String sendEmail(@ModelAttribute("mail") EmailInfo mail) {
    try{
        MimeMessage message = mailSender.createMimeMessage();

        MimeMessageHelper helper = new MimeMessageHelper(message, true);
        helper.setFrom(mail.getFrom());
        helper.setTo(mail.getTo());
        helper.setReplyTo(mail.getFrom());
        helper.setSubject(mail.getSubject());
        helper.setText(mail.getBody(), true);

        mailSender.send(message);
    }
}
```



JDBCTEMPLATE

- Giới thiệu JdbcTemplate
- Thao tác và truy vấn dữ liệu
- Xây dựng DAO làm việc với bảng

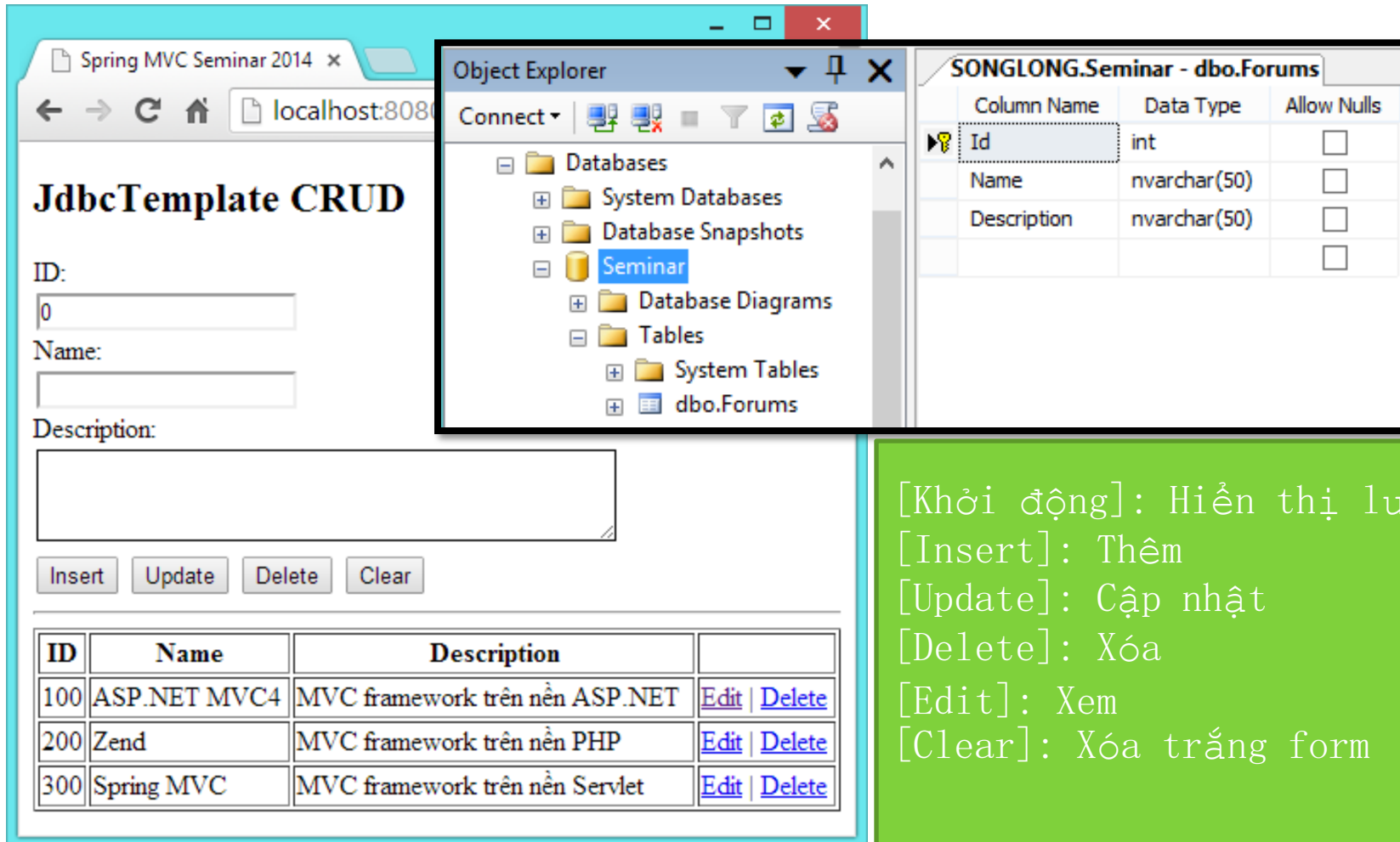
JDBCTEMPLATE

- Spring hỗ trợ JdbcTemplate giúp làm việc với CSDL đơn giản và hiệu quả.
- JdbcTemplate API đơn giản
 - ☆ **update()**
 - Thực thi câu lệnh thao tác dữ liệu (**insert**, **update** và **delete**)
 - ☆ **query()**
 - Truy vấn nhiều bản ghi
 - ☆ **queryForObject()**
 - Truy vấn 1 bản ghi
 - ☆ **queryForXYZ()**
 - Truy vấn một giá trị có kiểu là **XYZ** (Int, Long...)



CRUD VỚI JDBCTEMPLATE

○ Khám phá JdbcTemplate với ứng dụng CRUD



The screenshot displays a web application titled "JdbcTemplate CRUD" running on a browser at localhost:8080. The application has a form with fields for "ID:", "Name:", and "Description:". Below the form are buttons for "Insert", "Update", "Delete", and "Clear". A table below the form lists forum entries with columns "ID", "Name", "Description", and actions "Edit" and "Delete".

The Object Explorer shows the database structure:

- Databases
 - System Databases
 - Database Snapshots
 - Seminar**
 - Database Diagrams
 - Tables
 - System Tables
 - dbo.Forums**

The table structure for **SONGLONG.Seminar - dbo.Forums** is as follows:

Column Name	Data Type	Allow Nulls
Id	int	<input type="checkbox"/>
Name	nvarchar(50)	<input type="checkbox"/>
Description	nvarchar(50)	<input type="checkbox"/>
		<input type="checkbox"/>

The application data table shows the following entries:

ID	Name	Description	
100	ASP.NET MVC4	MVC framework trên nền ASP.NET	Edit Delete
200	Zend	MVC framework trên nền PHP	Edit Delete
300	Spring MVC	MVC framework trên nền Servlet	Edit Delete

[Khởi động]: Hiển thị lưới

[Insert]: Thêm

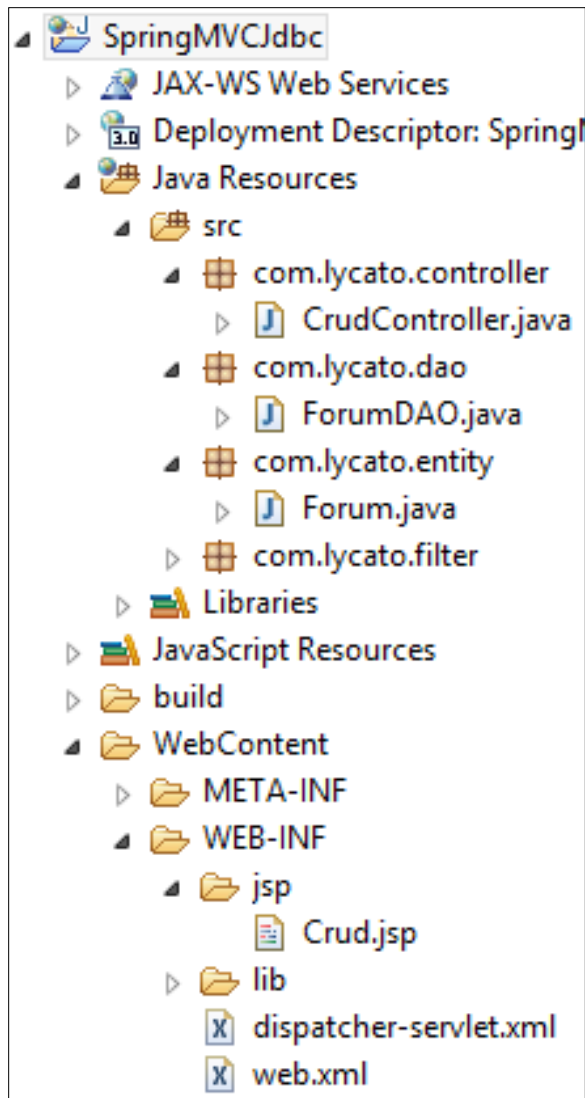
[Update]: Cập nhật

[Delete]: Xóa

[Edit]: Xem

[Clear]: Xóa trắng form

ĐỀ MÔ: CRUD QUẢN LÝ FORUM



- CrudController.java
 - ☆ Các action
- ForumDAO.java
 - ☆ Thao tác và truy vấn dữ liệu
- Forum.java
 - ☆ Thực thể mô tả thông tin bảng Forums
- Crud.jsp
 - ☆ Giao diện
- Dispatcher-servlet.xml
 - ☆ Khai báo cấu hình CSDL Seminar



CÁC BƯỚC THỰC HIỆN

○ Bước 1: Khai báo <bean> JdbcTemplate

```
<bean id="dataSource" destroy-method="close"  
      class="org.apache.commons.dbcp.BasicDataSource"../>  
  
<bean id="jdbcTemplate"  
      class="org.springframework.jdbc.core.JdbcTemplate">  
  <constructor-arg ref="dataSource"/>  
</bean>
```

○ Bước 2: Tiêm JdbcTemplate

```
@Autowired  
protected JdbcTemplate jdbc;
```



○ Bước 3: Thao tác và truy vấn

- ☆ jdbc.update()
- ☆ jdbc.query()
- ☆ jdbc.queryForXYZ()



CẤU HÌNH <BEAN>

- Bean DataSource cấu hình kết nối đến CSDL
- Bean JdbcTemplate sử dụng bean DataSource. Bean này sẽ được sử dụng trong lớp DAO để thực hiện các thao tác và truy vấn dữ liệu.

```
<bean id="dataSource" destroy-method="close"  
    class="org.apache.commons.dbcp.BasicDataSource"  
    p:driverClassName="com.microsoft.sqlserver.jdbc.SQLServerDriver"  
    p:url="jdbc:sqlserver://localhost:1433;DatabaseName=Seminar"  
    p:username="sa"  
    p:password="songlong"/>
```

```
<bean id="jdbcTemplate"  
    class="org.springframework.jdbc.core.JdbcTemplate">  
    <constructor-arg ref="dataSource"/>  
</bean>
```

DAO

```
<bean id="jdbcTemplate"  
      class="org.springframework.jdbc.core.JdbcTemplate">
```

@Repository

```
public class ForumDAO{
```

```
@Autowired
```

```
protected JdbcTemplate jdbc;
```

```
public void insert(Forum f) {
```

```
    String sql = "INSERT INTO Forums VALUES (?, ?, ?)";
```

```
    jdbc.update(sql, f.getId(), f.getName(), f.getDescription());
```

```
}
```

```
public Forum getById(Serializable id) {
```

```
    String sql = "SELECT * FROM Forums WHERE Id=?";
```

```
    return jdbc.queryForObject(sql, getRowMapper(), id);
```

```
}
```

```
protected List<Forum> getBySql(String sql) {
```

```
    return jdbc.query(sql, getRowMapper());
```

```
}
```

```
private RowMapper<Forum> getRowMapper() {
```

```
    return new BeanPropertyRowMapper<Forum>(Forum.class);
```

```
}
```


inject

Đọc bản ghi theo
các thuộc tính của
bean

THỰC THỂ FORUM

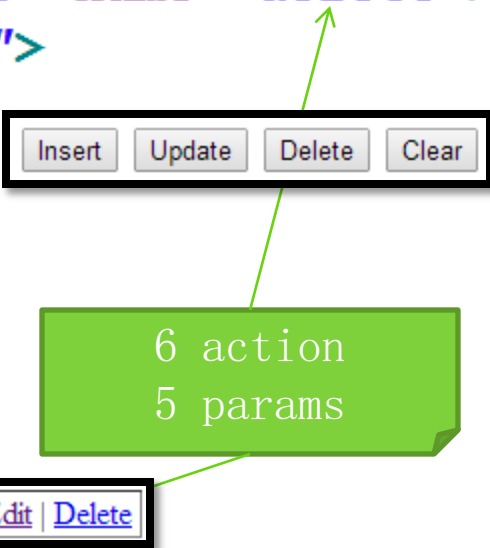
- Mô tả cấu trúc thông tin một bản ghi Forum

```
public class Forum {  
  
    private Integer id;  
    private String name, description;  
  
    public int getId() {..  
    public void setId(int id) {..  
  
    public String getName() {..  
    public void setName(String name) {..  
  
    public String getDescription() {..  
    public void setDescription(String description) {..  
}
```



CRUD.JSP FORM

```
<input type="submit" value="Insert" name="insert">
<input type="submit" value="Update" name="update">
<input type="submit" value="Delete" name="delete">
<input type="submit" value="Clear">
</form:form>
<hr>
<table border="1" style="width:100%">
<tr>
<c:forEach var="f" items="${forums}">
<tr>
<td>${f.id}</td>
<td>${f.name}</td>
<td>${f.description}</td>
<td>
<a href="?edit&id=${f.id}">Edit</a> |
<a href="?delete&id=${f.id}">Delete</a>
</td>
</tr>
</tr>
```



6 action
5 params

Edit | Delete

CRUDCONTROLLER

```
@Repository  
public class ForumDAO
```

```
@Controller  
@RequestMapping(value="/crud.htm")  
public class CrudController {  
  
    @Autowired  
    private ForumDAO dao;  
  
    public String clear(ModelMap model) {  
  
        @RequestMapping(params="insert", method = RequestMethod.P  
        public String insert(@ModelAttribute("forum") Forum forum  
            dao.insert(forum);  
            return clear(model);  
    }  
}
```

inject

ĐỀ MÔ: TRÌNH BÀY HÀNG HÓA

Gustaf flower



\$21.0



Tunnbr Korea



\$9.0



Singaporean Hokkien Fried Mee



\$14.0



Filo Mix



\$7.0



Gnocchi di nonna Alice



\$38.0



Ravioli Angelo



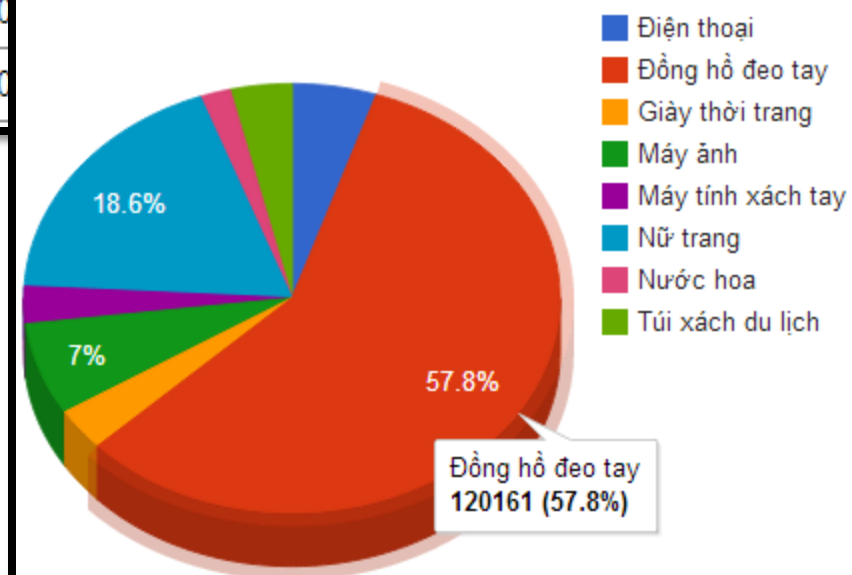
\$19.5



ĐỀ MÔ: THỐNG KÊ THÔNG TIN HÀNG HÓA

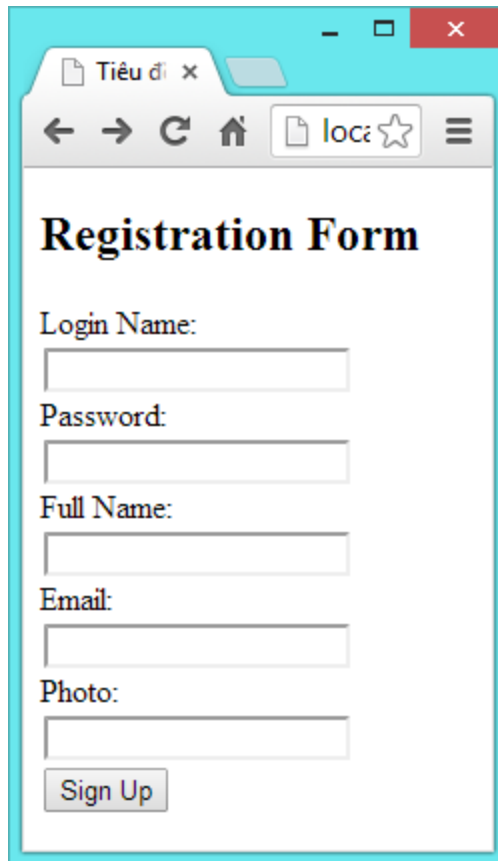
Loại	Số lượng hàng	Tổng giá trị	Giá thấp nhất	Giá cao nhất	Giá trung bình
Điện thoại	297	10,628.100	2.5	55.0	28.730
Đồng hồ đeo tay	758	120,161.000	4.5	263.5	51.050
Giày thời trang	166	6,645.950	10.0	53.0	32.370
Máy ảnh	365	14,521.260	9.2	81.0	23.962
Máy tính xách tay	196	5,837.000	10.0	43.9	23.218
Nữ trang	433	38,596.230	7.4	122.5	32.112
Nước hoa	145	3,831.750	7.0	26.5	26.396
Túi xách du lịch	251	7,745.850	6.0	31.0	30.860

Thống kê thông tin hàng hóa



ĐỀ MÔ: ĐĂNG KÝ THÀNH VIÊN

- Đăng ký = CustomerDAO.insert()
- Đăng nhập = CustomerDAO.getId()



A screenshot of a web browser window displaying a registration form. The browser's address bar shows a local file path. The form is titled "Registration Form" and contains five input fields: "Login Name:", "Password:", "Full Name:", "Email:", and "Photo:". A "Sign Up" button is located at the bottom of the form.



A screenshot of a web browser window displaying a login form. The browser's address bar shows a local file path. The form is titled "Login Form" and contains two input fields: "Login Name:" and "Password:". A "Sign In" button is located below the password field.

TỔNG KẾT

- Nguyên lý hoạt động của MVC
- Tổ chức Controller
- Các phương pháp nhận tham số yêu cầu
- Upload
- Dependency Injection
- Email
- JdbcTemplate
 - ☆ Crud
 - ☆ Product
 - ☆ Report & Google Chart
 - ☆ Login
 - ☆ Register



HỎI – ĐÁP



LỜI KẾT



THANK YOU

Trình bày:

- ✓ Nguyễn Nghiệm
- ✓ 0913. 745. 789
- ✓ nghiemn@fpt.edu.vn