

**BÀI TẬP MÔN LẬP TRÌNH WWW JAVA**  
**(ADVANCED WEB PROGRAMMING WITH JAVA)**  
**HỆ: ĐẠI HỌC - 2019**  
**(DÀNH CHO CHUYÊN NGÀNH: SE - SOFTWARE ENGINEERING)**

## Nội dung

TIÊU CHÍ ĐÁNH GIÁ CHO MÔN LẬP TRÌNH WWW/LẬP TRÌNH WEB NÂNG CAO (JAVA) ....	4
BÀI TẬP LỚN - DANH SÁCH ĐỀ TÀI .....	5
PHÂN BỐ THỜI GIAN THỰC HÀNH.....	7
<b>BÀI TẬP TUẦN 01-02 MÔN LẬP TRÌNH WEB NÂNG CAO (JAVA) .....</b>	<b>8</b>
<i>Bài 1. Phân tích yêu cầu của Bài tập lớn .....</i>	8
<i>Bài 2. Layout Bài tập lớn - Bài tập cá nhân .....</i>	11
<i>Bài 3. Cài đặt Tomcat Server .....</i>	12
<i>Bài 4. Java Servlet - Thao tác với doGet(), doPost() .....</i>	15
<i>Bài 5. Java Servlet - Filter .....</i>	16
<i>Bài 6. Java Servlet - Upload files.....</i>	16
<i>Bài 7. Java Servlet - Upload hình, lưu CSDL .....</i>	19
<i>Bài 8. Java Servlet - JavaMail API.....</i>	22
<b>BÀI TẬP TUẦN 03-04 MÔN LẬP TRÌNH WEB NÂNG CAO (JAVA) .....</b>	<b>25</b>
<i>Bài 1. JSPs - Thao tác với Form .....</i>	30
<i>Bài 2. JSPs - Model View Controller .....</i>	32
<i>Bài 3. JSPs - Xử lý đa ngôn ngữ.....</i>	33
<i>Bài 4. JSPs - Thao tác với JSP Session .....</i>	39
<i>Bài 5. JSPs - Thao tác với Shopping Cart.....</i>	51
<i>Bài 6. JSPs - Standard Tag Library và Expression Language .....</i>	54
<i>Bài 7. JSPs - Bài tập tổng hợp 1 .....</i>	59
<i>Bài 8. JSPs - Bài tập tổng hợp 2 .....</i>	60
<b>BÀI TẬP TUẦN 05-06-07 MÔN LẬP TRÌNH WEB NÂNG CAO (JAVA) .....</b>	<b>70</b>
<i>Bài 1. JSFs - Các tags căn bản .....</i>	86
<i>Bài 2. JSFs - Kiểm tra dữ liệu nhập dùng Regular Expression .....</i>	86
<i>Bài 3. JSFs - Page Navigation .....</i>	86
<i>Bài 4. JSFs - Bài tập Quản lý thông tin lớp học .....</i>	87
<i>Bài 5. JSFs - Thao tác với ComboBox .....</i>	96
<i>Bài 6. JSFs - Ajax căn bản .....</i>	97
<i>Bài 7. JSFs - Bài tập Quản lý danh sách công việc .....</i>	98
<i>Bài 8. JSFs - Thao tác với &lt;h:dataTable&gt; và &lt;ui:repeat&gt; .....</i>	100
<i>Bài 9. JSFs - Ajax .....</i>	105
<i>Bài 10. JSFs - Bài tập tổng hợp .....</i>	109

<i>Bài 11.</i>	<i>JSFs - Bài tập tổng hợp .....</i>	111
<i>Bài 12.</i>	<i>JSFs - Bài tập tổng hợp .....</i>	112
<b>BÀI TẬP TUẦN 08-09-10 MÔN LẬP TRÌNH WEB NÂNG CAO (JAVA).....</b>		114
<i>Bài 1.</i>	<i>Spring DI.....</i>	121
<i>Bài 2.</i>	<i>Spring AOP .....</i>	124
<i>Bài 3.</i>	<i>Spring Bean .....</i>	131
<i>Bài 4.</i>	<i>Spring MVC - Chương trình đơn giản Hello Word .....</i>	146
<i>Bài 5.</i>	<i>Spring MVC - Thao tác với Form.....</i>	149
<i>Bài 6.</i>	<i>Spring MVC - CRUD .....</i>	152
<i>Bài 7.</i>	<i>Spring MVC - Dùng Spring Tool Suite - Hello Word .....</i>	161
<i>Bài 8.</i>	<i>Spring MVC - Upload file .....</i>	173
<i>Bài 9.</i>	<i>Spring MVC - Exception Handling.....</i>	176
<i>Bài 10.</i>	<i>Spring MVC - Form Validation.....</i>	177
<i>Bài 11.</i>	<i>Spring MVC - Đa ngôn ngữ .....</i>	184
<i>Bài 12.</i>	<i>Spring MVC - JavaMail API .....</i>	193
<i>Bài 13.</i>	<i>Spring MVC - Thao tác với XML và JSON .....</i>	193
<i>Bài 14.</i>	<i>Spring MVC - Bài tập tổng hợp (1 bảng).....</i>	202
<i>Bài 15.</i>	<i>Spring MVC - Bài tập tổng hợp (1 bảng).....</i>	203
<i>Bài 16.</i>	<i>Spring MVC - Bài tập tổng hợp (2 bảng quan hệ 1-N).....</i>	210
<i>Bài 17.</i>	<i>Spring MVC - Bài tập tổng hợp (2 bảng quan hệ 1-N).....</i>	211

## TIÊU CHÍ ĐÁNH GIÁ CHO MÔN LẬP TRÌNH WWW/LẬP TRÌNH WEB NÂNG CAO (JAVA)

- CLO 1. Tham gia các hoạt động của nhóm trong quá trình hoàn thành công việc.
- CLO 2. Hoàn thành công việc của mình trong nhóm khi được phân công.
- CLO 3. Xác định được các mục tiêu công việc và các ràng buộc đi kèm của một ứng dụng Web.
- CLO 4. Trình bày và cài đặt được cách kết nối đến các nguồn dữ liệu (data sources) khi xây dựng ứng dụng web sử dụng JSPs/Servlets có kết nối đến cơ sở dữ liệu.
- CLO 5. Viết tài liệu báo cáo rõ ràng theo quy định (đúng cấu trúc, đầy đủ nội dung, trích dẫn, tài liệu tham khảo theo yêu cầu)
- CLO 6. Trình bày được các công nghệ trong phát triển ứng dụng Web. Trình bày được kiến trúc và cấu hình được môi trường thực thi của ứng dụng lựa chọn một trong 2 công nghệ cụ thể là JSF hoặc Spring MVC Framework.
- CLO 7. VỚI MỘT YÊU CẦU CỤ THỂ, CÓ THỂ CHỌN VÀ CÀI ĐẶT ĐƯỢC ỨNG DỤNG SỬ DỤNG MỘT TRONG 2 CÔNG NGHỆ JSF HOẶC SPRING FRAMEWORK.

## BÀI TẬP LỚN - DANH SÁCH ĐỀ TÀI

01. Website giới thiệu, bán thiết bị máy tính trực tuyến.
02. Website giới thiệu, bán xe máy và phụ kiện trực tuyến.
03. Website giới thiệu, bán mỹ phẩm trực tuyến.
04. Website giới thiệu, bán quần áo trực tuyến.
05. Website giới thiệu, bán văn phòng phẩm trực tuyến.
06. Website giới thiệu, bán dụng cụ thể thao trực tuyến.
07. Website giới thiệu, bán sách trực tuyến.
08. Website giới thiệu, bán giày dép trực tuyến.
09. Website giới thiệu, bán điện thoại trực tuyến.
10. Website giới thiệu, bán đồng hồ trực tuyến
11. Website giới thiệu, bán đồ gia dụng trực tuyến.
12. Website giới thiệu, bán máy PC trực tuyến.
13. Website giới thiệu, bán mắt kính trực tuyến
14. Website giới thiệu, bán thời trang nam, nữ trực tuyến
15. Website giới thiệu, bán quần áo trẻ em trực tuyến
16. Website giới thiệu, bán đồ chơi trẻ em trực tuyến
17. Website giới thiệu, bán thiết bị mạng trực tuyến
18. Website giới thiệu, bán máy in, máy scan trực tuyến
19. Website giới thiệu, bán phụ kiện điện thoại trực tuyến
20. Website giới thiệu, bán máy ảnh kỹ thuật số trực tuyến
21. Website giới thiệu, bán thiết bị gia đình trực tuyến
22. Website giới thiệu, bán đồ dùng trang trí nội thất trực tuyến
23. Website giới thiệu, bán thiết bị âm thanh trực tuyến
24. Website giới thiệu, bán nhạc cụ trực tuyến
25. Website giới thiệu, bán vali/túi du lịch trực tuyến
26. Website giới thiệu, bán thiết bị dụng cụ thể thao trực tuyến
27. Website giới thiệu, bán thiết bị sửa chữa nhà cửa trực tuyến
28. Website giới thiệu, bán ô tô, phụ kiện ô tô trực tuyến
29. Website giới thiệu, bán xe đạp điện trực tuyến

30. Website giới thiệu, bán văn phòng phẩm trực tuyến
31. Website giới thiệu, bán nước hoa trực tuyến
32. Website giới thiệu, bán thiết bị an ninh trực tuyến
33. Website giới thiệu, bán sách trực tuyến
34. Website giới thiệu, bán bánh ngọt trực tuyến
35. Website giới thiệu, bán bánh kẹo/socola trực tuyến
36. Website giới thiệu, bán vật phẩm phong thủy trực tuyến
37. Website giới thiệu, bán giày dép trực tuyến
38. Website giới thiệu, bán túi xách nữ trực tuyến
39. Website giới thiệu, bán thiệp trực tuyến
40. Website giới thiệu, bán quà tặng hàng thủ công trực tuyến
41. Website giới thiệu, bán vật tư y tế trực tuyến
42. Website giới thiệu, bán búp bê, thú bông trực tuyến
43. Website giới thiệu, bán máy cơ khí trực tuyến
44. Website giới thiệu, bán loa và phụ kiện âm thanh trực tuyến
45. Website giới thiệu, bán thức ăn chó mèo trực tuyến
46. Website giới thiệu, bán rau xanh trực tuyến
47. Website giới thiệu, bán trái cây trực tuyến
48. Website giới thiệu, bán sữa bột trực tuyến
49. Website giới thiệu, bán thực phẩm khô trực tuyến
50. Website giới thiệu, bán thực phẩm chức năng trực tuyến
51. Website giới thiệu báo giấy và cho phép đặt báo giấy trực tuyến.
52. Website giới thiệu tour du lịch và cho phép đăng ký tour trực tuyến.
53. Website giới thiệu và cho phép đăng ký dịch vụ bảo hiểm trực tuyến.
54. Website giới thiệu cơ sở vật chất của khách sạn và cho phép đặt phòng trực tuyến.

## PHÂN BỐ THỜI GIAN THỰC HÀNH

Thời gian: 10 tuần, 3 tiết/tuần

Lưu ý:

- Báo cáo bài tập lớn trình bày trên giấy khổ A4 theo đúng mẫu quy định
- Kiểu chữ Time New Roman 13

*Bài tập bắt buộc của Tuần XX GV sẽ công bố cuối giờ thực hành. Bài tập bắt buộc sẽ gửi vào email [thanhvan.fitse@gmail.com](mailto:thanhvan.fitse@gmail.com) với Subject và tập tin nén kèm theo: LTWWW\_JAVA\_DHKTPM12A\_TUANXX\_HOTENSINHVIEN hoặc LTWWW\_JAVA\_DHKTPM12ATT\_TUANXX\_HOTENSINHVIEN (trong đó XX sẽ là 01, 02 ... → 10). Sinh viên nộp sai yêu cầu bài tập Tuần nào xem như không nộp bài tập Tuần đó.*

- Tuần 01-02: Project bài tập lớn và Java Servlets
- Tuần 03-04: JavaServer Pages
- Tuần 05-06-07: JavaServer Faces
- Tuần 08-09-10: Spring Framework

# BÀI TẬP TUẦN 01-02 MÔN LẬP TRÌNH WEB NÂNG CAO (JAVA)

*Chương 1: Application Model - Web Application Architecture*

*Chương 2: Java Servlets*

Mục tiêu:

- (Review) Trình bày được mô hình ứng dụng Web các khái niệm liên quan.
- (Review) Thực hiện được layout của trang Web dùng HTML/CSS và thực hiện kiểm tra dữ liệu nhập phía Client dùng JavaScript.
- Hiểu được cấu trúc HTTP Request, HTTP Response.
- Phân tích yêu cầu theo đề tài bài tập lớn. Thực hiện các mô hình UML với yêu cầu tối thiểu.
- Cài đặt và cấu hình được Project với Tomcat Web Server.
- Hiểu được một cấu trúc ứng dụng Web Back-End với Java Servlets.
- Thực hiện các bài tập FormData, Session Tracking, Send Mail, Upload Files với Java Servlets.

Yêu cầu:

- Tất cả các bài tập lưu trong thư mục: T:\MaSV\_HoTen\Tuan01-02\
- Tạo Project **MaSV\_HoTen\_Tuan01-02** trong thư mục T:\MaSV\_HoTen\Tuan01-02\ trong Eclipse Java EE (Java Platform Enterprise Edition). Mỗi bài tập có thể lưu trong từng package riêng biệt.
- Cuối mỗi buổi thực hành, SV phải nén (.rar hoặc .zip) thư mục làm bài và nộp lại bài tập đã thực hiện trong buổi đó.

## Bài 1. Phân tích yêu cầu của Bài tập lớn

Yêu cầu của các đề tài 01 →50, chức năng tối thiểu:

- Website bao gồm 3 loại người dùng tương tác: người dùng không có tài khoản (guest), người dùng có tài khoản (customer), người quản trị hệ thống (admin).
- Người dùng không có tài khoản (guest) có các chức năng:
  - Xem danh sách sản phẩm (thiết bị máy tính, mỹ phẩm, quần áo ... tùy theo đề tài, danh sách này lấy từ CSDL)
  - Xem chi tiết của từng sản phẩm từ danh sách sản phẩm.
  - Chọn mua từng sản phẩm (có thể chọn mua từ trang Web danh sách sản phẩm hay từ trang Web chi tiết của từng sản phẩm), sản phẩm sau khi chọn mua sẽ được đưa vào trong giỏ hàng.
  - Xem giỏ hàng (danh sách sản phẩm đã chọn mua, thông tin này lưu trong biến Session, không cần cập nhật CSDL).
  - Khi xem giỏ hàng, có thể chỉnh sửa số lượng của từng sản phẩm trong giỏ hàng (nếu chỉnh sửa số lượng là 0 → bỏ sản phẩm đó ra khỏi giỏ hàng)
  - Có thể đăng ký tài khoản của website với các thông tin cần thiết (email không trùng với tài khoản khác), sau khi đăng ký thành công với thông tin hợp lệ, lưu trữ CSDL + gửi email + thông báo về tài khoản.

- Người dùng có tài khoản (customer) có thể thực hiện các chức năng của Người dùng không có tài khoản (guest), ngoài ra người dùng có tài khoản (customer) còn có thể:
  - Xử lý thanh toán (chức năng này thực hiện khi giỏ hàng đã có sản phẩm và người dùng đăng nhập thành công vào hệ thống): cập nhật thông tin vào CSDL + gửi email + thông báo đăng ký đặt hàng thành công với các thông tin kèm theo. Sau khi xử lý thành công, Session được xóa về null.
- Người quản trị hệ thống (admin) có thể thực hiện được chức năng như một người dùng có tài khoản (customer). Ngoài ra, chức năng khác dành cho người quản trị hệ thống (admin) - Phần Back-End:
  - Tìm kiếm thông tin về sản phẩm/loại sản phẩm, tài khoản người dùng, các đơn đặt sản phẩm.
  - Quản lý thông tin sản phẩm/loại sản phẩm:
    - Xem danh sách sản phẩm/loại sản phẩm.
    - Xem chi tiết từng sản phẩm/loại sản phẩm.
    - Xóa sản phẩm/loại sản phẩm trong trường hợp sản phẩm chưa có trong đơn hàng nào hoặc loại sản phẩm chưa có sản phẩm nào.
    - Thêm mới, cập nhật thông tin sản phẩm/loại sản phẩm.
  - Quản lý thông tin tài khoản người dùng:
    - Xem danh sách các tài khoản người dùng đã đăng ký.
    - Xem chi tiết từng tài khoản người dùng, không xem được password của người dùng.
    - Xóa tài khoản người dùng nếu người dùng chưa thực hiện đặt hàng online lần nào.
    - Cập nhật thông tin tài khoản người dùng.
  - Quản lý thông tin đơn hàng trực tuyến:
    - Xem danh sách các đơn hàng (sắp xếp theo ngày mua)
    - Xem chi tiết đơn hàng.
    - Cập nhật số lượng của mặt hàng trong đơn hàng trực tuyến
  - Lưu ý cho các chức năng quản lý thông tin:
    - Ràng buộc khi xóa dữ liệu
    - Trường hợp thêm hay cập nhật dữ liệu có thể kiểm tra phía Client bằng JavaScript/jQuery hoặc kiểm tra bằng Model phía Server, không dùng Functions/Check constraints/Stored Procedures trong hệ quản trị CSDL.

Yêu cầu của các đề tài 51 →54, chức năng tối thiểu:

- Website bao gồm 3 loại người dùng tương tác: người dùng không có tài khoản (guest), người dùng có tài khoản (customer), người quản trị hệ thống (admin).
- Người dùng không có tài khoản (guest) có các chức năng:
  - Xem danh sách các báo giấy/tour/phòng khách sạn/dịch vụ bảo hiểm (tùy theo đề tài, danh sách này lấy từ CSDL)
  - Xem chi tiết của từng báo giấy/tour/phòng khách sạn/dịch vụ bảo hiểm từ danh sách.

- Chọn từng tờ báo giấy/tour/phòng khách sạn/dịch vụ bảo hiểm (có thể chọn từ trang Web danh sách hay từ trang Web chi tiết), các tờ báo giấy/tour/phòng khách sạn/dịch vụ bảo hiểm sản phẩm sau khi chọn đặt sẽ được đưa vào trong giỏ hàng.
  - Xem giỏ hàng (danh sách đã chọn, thông tin này lưu trong biến Session, không cần cập nhật CSDL).
  - Khi xem giỏ hàng, có thể chỉnh sửa số lượng của từng thành phần trong giỏ hàng (nếu chỉnh sửa số lượng là 0 → bỏ sản phẩm đó ra khỏi giỏ hàng)
  - Có thể đăng ký tài khoản của website với các thông tin cần thiết (email không trùng với tài khoản khác), sau khi đăng ký thành công với thông tin hợp lệ, lưu trữ CSDL + gửi email + thông báo về tài khoản.
- Người dùng có tài khoản (customer) có thể thực hiện các chức năng của Người dùng không có tài khoản (guest), ngoài ra người dùng có tài khoản (customer) còn có thể:
- Xử lý thanh toán (chức năng này thực hiện khi giỏ hàng đã có thông tin và người dùng đăng nhập thành công vào hệ thống): cập nhật thông tin vào CSDL + gửi email + thông báo đăng ký đặt báo giấy/tour/phòng khách sạn/dịch vụ bảo hiểm thành công với các thông tin kèm theo. Sau khi xử lý thành công, Session được xóa về null.
- Người quản trị hệ thống (admin) có thể thực hiện được chức năng như một người dùng có tài khoản (customer). Ngoài ra, chức năng khác dành cho người quản trị hệ thống (admin) - Phần Back-End:
- Tìm kiếm thông tin về báo giấy/tour/phòng khách sạn/dịch vụ bảo hiểm và các loại dịch vụ, tài khoản người dùng, các đơn đặt sản phẩm.
  - Quản lý thông tin báo giấy/tour/phòng khách sạn/dịch vụ bảo hiểm/loại:
    - Xem danh sách báo giấy/tour/phòng khách sạn/dịch vụ bảo hiểm/loại.
    - Xem chi tiết từng báo giấy/tour/phòng khách sạn/dịch vụ bảo hiểm/loại.
    - Xóa báo giấy/tour/phòng khách sạn/dịch vụ bảo hiểm/loại trong trường hợp thông tin cần xóa có trong đơn hàng nào hoặc loại sản phẩm chưa có sản phẩm nào.
    - Thêm mới, cập nhật thông tin sản phẩm/loại sản phẩm.
  - Quản lý thông tin tài khoản người dùng:
    - Xem danh sách các tài khoản người dùng đã đăng ký.
    - Xem chi tiết từng tài khoản người dùng, không xem được password của người dùng.
    - Xóa tài khoản người dùng nếu người dùng chưa thực hiện đặt hàng online lần nào.
    - Cập nhật thông tin tài khoản người dùng.
  - Quản lý thông tin đơn hàng trực tuyến:
    - Xem danh sách các đơn hàng (sắp xếp theo ngày mua)
    - Xem chi tiết đơn hàng.
    - Cập nhật số lượng của mặt hàng trong đơn hàng trực tuyến
  - Lưu ý cho các chức năng quản lý thông tin:
    - Ràng buộc khi xóa dữ liệu
    - Trường hợp thêm hay cập nhật dữ liệu có thể kiểm tra phía Client bằng JavaScript/jQuery hoặc kiểm tra bằng Model phía Server, không dùng Functions/Check constraints/Stored Procedures trong hệ quản trị CSDL.

## Bài 2. Layout Bài tập lớn - Bài tập cá nhân

- Layout trang web dùng HTML/CSS dùng chung cho Project bài tập lớn (dùng cho nhóm)
- Layout trang web dùng HTML/CSS dùng chung cho tất cả các bài tập của môn học (dùng cho cá nhân).

Yêu cầu cho layout bài tập cá nhân hàng tuần:

<header>		
<menu>		
<nav>	Nội dung	<section>
<footer>		

Hoặc tương tự:

The screenshot shows a website layout for the Software Engineering Department at Industrial University of Ho Chi Minh City (IUH). The header contains the IUH logo and text. The main content area is yellow and empty. The footer includes copyright information and social media icons.

2018 © Software Engineering Department, Information Technology Faculty, IUH.

f

### Bài 3. Cài đặt Tomcat Server

Cài đặt Tomcat 9, mặc định Tomcat Server chạy ở Port 8080

Tạo user với Tomcat Server: **conf/tomcat-users.xml**

```

<tomcat-users xmlns="http://tomcat.apache.org/xml"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:schemaLocation="http://tomcat.apache.org/xml tomcat-users.xsd"
               version="1.0">

    <role rolename="tomcat"/>
    <role rolename="role1"/>
    <role rolename="admin"/>
<role rolename="manager-gui"/>
    <user username="admin" password="password" roles="standard,manager,admin"/>
    <user username="tomcat" password="" roles="tomcat"/>
<user username="both" password="" roles="tomcat,role1"/>
    <user username="role1" password="" roles="role1"/>
    <user username="admin" password="admin" roles="manager-gui"/>
</tomcat-users>

```

Thiết lập các đường dẫn PATH và CLASSPATH

*set PATH = C:\Program Files\Java\jdk-9.0.1\bin;%PATH%*

```

set JAVA_HOME = C:\Program Files\Java\jdk-9.0.1
set CATALINA = C:\Tomcat902
set CLASSPATH = %CATALINA%\common\lib\servlet-api.jar;%CLASSPATH%

```

Khi bổ sung thêm các thư viện .jar, thêm vào CLASSPATH

### Lưu ý phần triển khai Servlet

```

public class TênServlet extends javax.servlet.http.HttpServlet {
    // servlet code...
}

```

C1. Thực hiện trong file WEB-INF/web.xml

```

<servlet>
    <servlet-name>TênServlet</servlet-name>
    <servlet-class>TênLớpServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>TênServlet</servlet-name>
    <url-pattern>/TênPathofServlet</url-pattern>
</servlet-mapping>

```

Kiểm tra các triển khai trên Web Browser: <http://localhost:8080/TênPathofServlet>

C2. Dùng Annotation:

Lớp javax.servlet.annotation.WebServlet;

Cú pháp của annotation

```

@WebServlet(
    attribute1=value1,
    attribute2=value2,
    ...
)

```

Các thuộc tính

Tên thuộc tính	Loại dữ liệu	Thuộc tính bắt buộc hoặc tự chọn	Mô tả
value hoặc urlPatterns	<i>String[]</i>	Required	Specify one or more URL patterns of the servlet. Either of attribute can be used, but not both.
name	<i>String</i>	Optional	Name of the servlet
displayName	<i>String</i>	Optional	Display name of the servlet
description	<i>String</i>	Optional	Description of the servlet
asyncSupported	<i>boolean</i>	Optional	Specify whether the servlet supports asynchronous operation mode. Default is false.

initParams	<i>WebInitParam[]</i>	Optional	Specify one or more initialization parameters of the servlet. Each parameter is specified by <u>@WebInitParam</u> annotation type.
loadOnStartup	<i>int</i>	Optional	Specify load-on-startup order of the servlet.
smallIcon	<i>String</i>	Optional	Specify name of the small icon of the servlet.
largeIcon	<i>String</i>	Optional	Specify name of the large icon of the servlet.

```

import javax.servlet.annotation.WebServlet;
import javax.servlet.annotation.WebInitParam;
import javax.servlet.http.HttpServlet;

@WebServlet(
    name = "MyOwnServlet",
    description = "Mô tả ...",
    urlPatterns = {"/processServlet1", "/processServlet2"},
    initParams =
{
    @WebInitParam(name = "saveDir", value = "C:/FileUpload"),
    @WebInitParam(name = "allowedTypes", value = "jpg,jpeg,gif,png")
}
)
public class ServletName extends HttpServlet {
    // implement servlet doPost() and doGet()...
}

```

HTTP Status Code (Tham khảo: <http://www.restapitutorial.com/httpstatuscodes.html>)

Các HTTP status code có 3 chữ số XXX và được phân thành 5 loại chính dựa vào chữ số đầu tiên:

- 1xx: Các status code này dùng để đơn giản thông báo với client rằng server đã **nhận** được request.
- 2xx: Các status code loại này có ý nghĩa rằng request được server **nhận**, **hiểu** và **xử lý** thành công.
- 3xx: Các status code loại này có ý nghĩa rằng server sẽ chuyển tiếp request hiện tại sang một request khác và client cần thực hiện việc gửi request tiếp theo đó để có thể hoàn tất. Thông thường khi trình duyệt nhận được status code loại này nó sẽ tự động thực hiện việc gửi request tiếp theo để lấy về kết quả.
- 4xx: Các status code loại này có ý nghĩa rằng đã có lỗi từ phía *client* trong khi gửi request (như sai URL, sai HTTP method, không có quyền truy cập vào trang...)
- 5xx: Các status code loại này có ý nghĩa rằng server đã có lỗi từ phía *server* trong khi xử lý request (database lỗi hoặc server bị hết bộ nhớ)

#### Bài 4. Java Servlet - Thao tác với doGet(), doPost()

Bài tập dùng Servlet truyền dữ liệu thông qua Form Data. Tạo form đăng ký thông tin bao gồm các thành phần TextBox, CheckBox, ComboBox, TextArea.

- Khi thông tin có nhập đầy đủ, dữ liệu hiển thị thông tin mới nhập trừ password.
- Xem xét và so sánh HTTP Request message và HTTP Response message trường hợp method của form là GET/POST.

The screenshot shows a registration form titled "Register". The form fields are as follows:

- Name \*:  
First  Last
- Username \*:
- E-mail \*:
- Password \*:
- Facebook:  
Enter your Facebook profile URL:
- Short Bio:  
Share a little information about yourself.
- Submit

B1. Tạo form HTML, có <form method="get/post" action="**TênServlet**" name="registrationForm">

B2. Tạo Servlet

**@WebServlet("/TênServlet")**

```
public class RegistrationForm extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();

        String name=request.getParameter("Name");
        String email=request.getParameter("Email");
        String facebook=request.getParameter("Facebook");
        String shortbio=request.getParameter("ShortBio");
        String html=<br>                + "<html>"
```

```

        + "<head>"          + "<title>Result Page</title>"
        + "</head>"          + "<body>"
        + "First Name: "+ name + "<br>"
        + "Email: "+ email + "<br> Facebook URL: "+ facebook + "<br>"
        + "Short Bio: "+ shortbio + "<br>"
        + "</body>"          + "</html>";
    out.println(html);
}
}

```

## Bài 5. Java Servlet - Filter

(*Bài tập SV tự làm*)

- Dùng Filter trong Servlet để chặn các Requests từ Client trước khi Client gửi yêu cầu truy cập dữ liệu Back-End.
- Dùng Filter trong Servlet để thao tác các Response Server trước khi gửi trả lại tới Client.

### Hướng dẫn:

Các Filter thực hiện trong web.xml ánh xạ tới tên các Servlet và URL Pattern:

- Authentication Filters
- Data compression Filters
- Encryption Filters
- Filters kích hoạt các sự kiện (truy cập data source)
- Image Conversion Filters
- Logging and Auditing Filters
- MIME-TYPE Chain Filters.
- Tokenizing Filters .
- XSL/T Filter chuyển đổi nội dung XML

## Bài 6. Java Servlet - Upload files

Thực hiện upload nhiều file lưu trữ ở Server.

Upload multi-files

File #1:  No file chosen

File #2:  No file chosen

File #3:  No file chosen

File #4:  No file chosen

File #5:  No file chosen

### Hướng dẫn:

- B1. Dùng thư viện hỗ trợ upload file: commons-fileupload-1.2.2.jar và commons-io-2.3.jar
- B2. Tạo form cho phép tải lên Server nhiều tập tin.
- B3. Tạo Servlet xử lý việc upload nhiều files.

File Servlet tham khảo:

```
package se.iuh.edu.vn;

import java.io.File;
import java.io.IOException;
import java.util.Iterator;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.commons.fileupload.FileItem;
import org.apache.commons.fileupload.disk.DiskFileItemFactory;
import org.apache.commons.fileupload.servlet.ServletFileUpload;

public class MultiFilesUploadServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    private static final String UPLOAD_DIRECTORY = "upload";
    private static final int THRESHOLD_SIZE = 1024 * 1024 * 3; // 3MB
    private static final int MAX_FILE_SIZE = 1024 * 1024 * 10; // 10MB
    private static final int REQUEST_SIZE = 1024 * 1024 * 50; // 50MB

    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        if (!ServletFileUpload.isMultipartContent(request)) {
            response.getWriter().println("Does not support!");
            return;
        }

        // cấu hình tham số
        DiskFileItemFactory factory = new DiskFileItemFactory();
        factory.setSizeThreshold(THRESHOLD_SIZE);
        factory.setRepository(new File(System.getProperty("java.io.tmpdir")));

        ServletFileUpload upload = new ServletFileUpload(factory);
        upload.setFileSizeMax(MAX_FILE_SIZE);
```

```

upload.setSizeMax(REQUEST_SIZE);

// thư mục lưu trữ hình ảnh, chạy ngoài Eclipse
String uploadPath = getServletContext().getRealPath("") + File.separator +
UPLOAD_DIRECTORY;

// tạo thư mục nếu chưa tồn tại thư mục này
File uploadDir = new File(uploadPath);
if (!uploadDir.exists()) {
    uploadDir.mkdir();
}

try {
    List formItems = upload.parseRequest(request);
    Iterator iter = formItems.iterator();

    while (iter.hasNext()) {
        FileItem item = (FileItem) iter.next();
        if (!item.isFormField()) {
            String fileName = new File(item.getName()).getName();
            String filePath = uploadPath + File.separator + fileName;
            File storeFile = new File(filePath);

            item.write(storeFile); // lưu file vào thư mục
        }
    }
    request.setAttribute("message", "Upload has been done successfully!");
} catch (Exception ex) {
    request.setAttribute("message", "There was an error: " +
ex.getMessage());
    ex.printStackTrace();
}
getServletContext().getRequestDispatcher("MessageServlet").forward(request,
response);
}
}

```

## Bài 7. Java Servlet - Upload hình, lưu CSDL

Thực hiện trang Web cho phép upload hình ảnh và lưu dạng Binary trong CSDL SQL Server.

### File Upload to Database (multipart/form-data)

First Name:

Last Name:

Portrait Photo:  Choose File No file chosen

Hướng dẫn: (Sinh viên cần tách các lớp thao tác với CSDL, không dùng chung)

- CSDL:

```
CREATE DATABASE UploadFileServletDB;
CREATE TABLE contacts (
    contact_id int NOT NULL identity(1,1),
    first_name nvarchar(45) DEFAULT NULL,
    last_name nvarchar(45) DEFAULT NULL,
    photo image,
    PRIMARY KEY (contact_id)
)
```

B1. Tạo Servlet hoặc trang HTML form. Lưu ý thuộc tính của form enctype="multipart/form-data"

B2. Tạo Servlet xử lý:

- Đọc dữ liệu của form: dữ liệu text + dữ liệu file
- Kết nối CSDL
- Insert dữ liệu vào bảng gồm cả dữ liệu Text và dữ liệu Image

B3. Trả về kết quả sau khi đã thực hiện xong.

- Tham khảo code Form

```
<form method="post" action="FileUploadDBServlet" enctype="multipart/form-data">
    <table border="0">
        <tr>
            <td>First Name: </td>
            <td><input type="text" name="firstName" size="50"/></td>
        </tr>
        <tr>
            <td>Last Name: </td>
            <td><input type="text" name="lastName" size="50"/></td>
        </tr>
    </table>
</form>
```

```

        </tr>
        <tr>
            <td>Portrait Photo: </td>
            <td><input type="file" name="photo" size="50"/></td>
        </tr>
        <tr>
            <td colspan="2">
                <input type="submit" value="Save">
            </td>
        </tr>
    </table>
</form>

```

- Tham khảo code Servlet xử lý

```

package se.iuh.edu.vn;

import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.sql.Blob;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import javax.servlet.ServletException;
import javax.servlet.annotation.MultipartConfig;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.Part;

@WebServlet("/FileUploadDBServlet")
@MultipartConfig(maxFileSize = 16177215) // upload file's size up to 16MB
public class FileUploadDBServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    private static final int BUFFER_SIZE = 4096;
    private static final String SAVE_DIR = "images";

    private String dbURL =
"jdbc:sqlserver://localhost:1433;databaseName=UploadFileServletDB";
    private String dbUser = "sa";
    private String dbPass = "1234567890";

```

```

protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    String firstName = request.getParameter("firstName");
    String lastName = request.getParameter("lastName");

    InputStream inputStream = null; // luồng dữ liệu nhập của upload file

    // lấy thông tin tập tin upload trong form, form này gồm nhiều phần dữ liệu
text và file (multipart request)
    Part filePart = request.getPart("photo");
    String fileUploadName = "";
    if (filePart != null) {
        fileUploadName = filePart.getName(); // filePart.getSize());
        inputStream = filePart.getInputStream();
    }

    Connection conn = null;
    String message = null;

    String filePath = "E:/" + fileUploadName + ".jpg"; // lưu Image Field của
CSDL vào file này

    try {
        // connects to the database
        DriverManager.registerDriver(new
com.microsoft.sqlserver.jdbc.SQLServerDriver());
        conn = DriverManager.getConnection(dbURL, dbUser, dbPass);

        // chèn dữ liệu vào CSDL UploadFileServletDB, trường hợp này bằng
contacts (khóa tự động tăng)
        String sql = "INSERT INTO contacts (first_name, last_name, photo) values
(?, ?, ?)";
        PreparedStatement statement = conn.prepareStatement(sql);
        statement.setString(1, firstName);
        statement.setString(2, lastName);

        if (inputStream != null) {
            statement.setBlob(3, inputStream);
        }
        int row = statement.executeUpdate(); // thực hiện lưu thông tin vào CSDL
        if (row > 0) {
            message = "File uploaded and saved into database";
        }
    }

    // đọc CSDL lưu file
    String sql1 = "SELECT photo FROM contacts WHERE first_name=? AND
last_name=?";

```

```

        statement = conn.prepareStatement(sql1);
        statement.setString(1, firstName);
        statement.setString(2, lastName);

        ResultSet result = statement.executeQuery();
        if (result.next()) {
            Blob blob = result.getBlob("photo");
            inputStream = blob.getBinaryStream();
            OutputStream outputStream = new FileOutputStream(filePath);

            int bytesRead = -1;
            byte[] buffer = new byte[BUFFER_SIZE];

            while ((bytesRead = inputStream.read(buffer)) != -1) {
                outputStream.write(buffer, 0, bytesRead);
            }

            inputStream.close();
            outputStream.close();
        }
    } catch (SQLException ex) {
        message = "ERROR: " + ex.getMessage();
        ex.printStackTrace();
    } finally {
        if (conn != null) {
            try {
                conn.close();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
    }
    request.setAttribute("Message", message);

getServletContext().getRequestDispatcher("MessageServlet").forward(request,
response);
}
}
}

```

### Bài 8. Java Servlet - JavaMail API

Gửi mail trong Servlet dùng JavaMail API và Java Activation Framework (JAF). Form gửi mail bao gồm tiêu đề, người nhận, nội dung và file đính kèm.

#### JavaMail hỗ trợ nhận gửi mail dùng

- **SMTP (Simple Mail Transfer Protocol)** dùng để gửi email trên Internet.
- **POP3 (Post Office Protocol phiên bản 3)** dùng để nhận email từ Server.
- **IMAP (Internet Message Access Protocol)**

## Lớp trong JavaMail

- Session: đại diện cho một lần gửi nhận mail
- Message: gồm các phương thức hỗ trợ việc gửi nhận mail. Các thành phần message cơ bản bao gồm địa chỉ người gửi và địa chỉ người nhận, tiêu đề mail (subject) và nội dung (body gồm nhiều parts)
- Transport: hỗ trợ việc gửi nhận mail qua Internet
- Address: Internet Address là đối tượng thực thi tạo các địa chỉ cho việc gửi nhận email trên Internet.
- Transport: gửi message trên Internet thông qua thông tin được xác định trên session
- Authenticator: xác thực tài khoản

B1. Download thư viện mail.jar và activation.java (hoặc các version liên quan)

B2. Chuẩn bị thông tin tài khoản để gửi mail. VD dùng Google Mail

Incoming Mail (POP3) Server - requires SSL:	pop.gmail.com Use SSL: Yes Port: 995	1
Outgoing Mail (SMTP) Server - requires TLS:	smtp.gmail.com (use authentication) Use Authentication: Yes Use STARTTLS: Yes (some clients call this SSL) Port: 465 or 587	2
Account Name:	your full email address (including @gmail.com or @your_domain.com)	3
Email Address:	your email address (username@gmail.com or username@your_domain.com)	
Password:	your Gmail password	4

B3. Thực hiện gửi mail

```

String to = "abcd@gmail.com";
String from = "web@gmail.com";
String host = "localhost";

Properties properties = System.getProperties();
properties.setProperty("mail.smtp.host", host);

Session session = Session.getDefaultInstance(properties); // Lấy đối tượng mặc định.
try{

    MimeMessage message = new MimeMessage(session); // Tạo đối tượng mặc định MimeMessage.
    message.setFrom(new InternetAddress(from));
    message.addRecipient(Message.RecipientType.TO, InternetAddress(to));
}

```

```
message.setSubject("Subject Line!");

BodyPart messageBodyPart = new MimeBodyPart(); // tạo đối tượng BodyPart của email.

messageBodyPart.setText("Message body"); // Nội dung

Multipart multipart = new MimeMultipart(); // Email sẽ gồm 2 part (text, file attached)

multipart.addBodyPart(messageBodyPart); // Phần text

// Phần xử lý với file attached
messageBodyPart = new MimeBodyPart();
String filename = "file.txt";
DataSource source = new FileDataSource(filename);
messageBodyPart.setDataHandler(new DataHandler(source));
messageBodyPart.setFileName(filename);
multipart.addBodyPart(messageBodyPart);

message.setContent(multipart);

Transport.send(message); // Gửi mail
```

# BÀI TẬP TUẦN 03-04 MÔN LẬP TRÌNH WEB NÂNG CAO (JAVA)

## Chương 3: JavaServer Pages JSPs

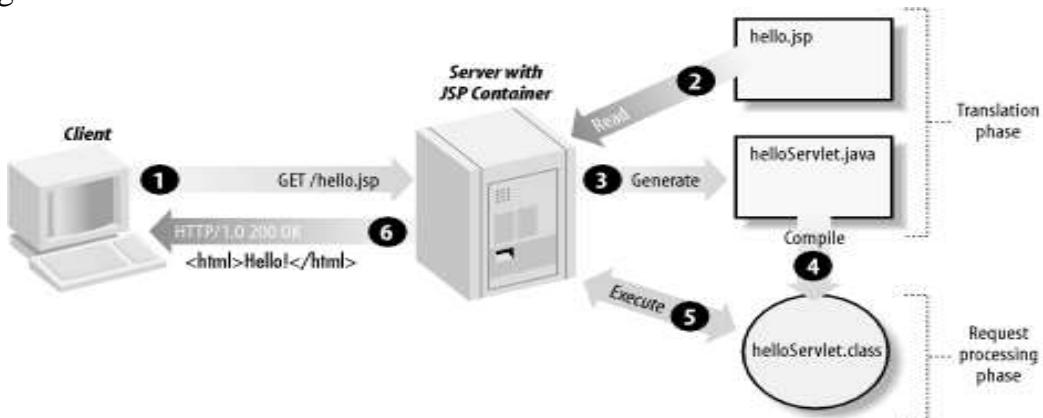
Mục tiêu:

- Hiểu và áp dụng được các cú pháp căn bản của JSPs trong việc xây dựng ứng dụng Web.
- Hiểu và áp dụng được các đối tượng ngầm định (implicit objects).
- Hiểu và áp dụng được JavaBean vào ứng dụng JSP.
- Hiểu và áp dụng được Expression Language vào ứng dụng Web.
- JSP Standard Tag Library (JSTL)

Yêu cầu:

- Tất cả các bài tập lưu trong thư mục: T:\MaSV\_HoTen\Tuan03-04\
- Tạo Project **MaSV\_HoTen\_Tuan03-04** trong thư mục T:\MaSV\_HoTen\Tuan03-04\ trong Eclipse Java EE (Java Platform Enterprise Edition). Mỗi bài tập có thể lưu trong từng package riêng biệt.
- Cuối mỗi buổi thực hành, SV phải nén (.rar hoặc .zip) thư mục làm bài và nộp lại bài tập đã thực hiện trong buổi đó.

JSP processing



1. Send an HTTP request to the Web server hello.jsp.
2. The web server recognizes that the HTTP request is for a JSP page *and forwards it to a JSP engine*. This is done by using the URL or JSP page which ends with **.jsp** instead of **.html**.
3. The **JSP engine** loads the JSP page from disk and converts it into a  *servlet content*. The code implements the corresponding dynamic behavior of the page.
4. The JSP engine compiles the servlet into an executable class and forwards the original request to a servlet engine.
5. The servlet engine loads the Servlet class and executes it. During execution, the servlet produces an output in HTML format. The output is further passed on to the web server by the servlet engine inside an HTTP response.
6. The web server forwards the HTTP response to your browser in terms of static HTML content. Finally, the web browser handles the dynamically-generated HTML page inside the HTTP response exactly as if it were a static page.

## Standard Tag Library (JSTL) trong JSP phân loại

- **Core Tags:** Nhóm thẻ cơ bản
- **Formatting tags:** Nhóm thẻ định dạng
- **SQL tags:** Nhóm thẻ SQL
- **XML tags:** Nhóm thẻ XML
- **JSTL Functions:** Nhóm hàm JSTL

### Core Tags (JSTL)

Cú pháp

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/html/core" %>
```

Thẻ	Miêu tả
<code>&lt;c:out&gt;</code>	Giống <code>&lt;%= ... %&gt;</code> , nhưng cho các Expression
<code>&lt;c:set&gt;</code>	Thiết lập kết quả của 1 Expression trong một 'scope'
<code>&lt;c:remove&gt;</code>	Gỡ bỏ một biến mục tiêu (tù một biến scope cụ thể, nếu đã xác định)
<code>&lt;c:catch&gt;</code>	Bắt bất kỳ Throwable
<code>&lt;c:if&gt;</code>	Thẻ điều kiện đơn giản.
<code>&lt;c:choose&gt;</code>	Thẻ điều kiện đơn giản mà thiết lập một context cho các hoạt động điều kiện loại trừ, được đánh dấu bởi <code>&lt;when&gt;</code> và <code>&lt;otherwise&gt;</code>
<code>&lt;c:when&gt;</code>	Thẻ phụ của <code>&lt;choose&gt;</code> mà điều kiện được ước lượng là true
<code>&lt;c:otherwise&gt;</code>	Thẻ phụ của <code>&lt;choose&gt;</code> mà theo sau thẻ <code>&lt;when&gt;</code> và chỉ chạy nếu tất cả điều kiện là 'false'
<code>&lt;c:import&gt;</code>	Dùng để import.
<code>&lt;c:forEach&gt;</code>	Thẻ lặp cơ bản.
<code>&lt;c:forTokens&gt;</code>	Lặp qua các token, được phân biệt bởi các dấu phân tách (delimiter) đã cung cấp
<code>&lt;c:param&gt;</code>	Thêm một parameter tới một URL của thẻ đang chứa 'import'
<code>&lt;c:redirect&gt;</code>	Redirect tới một URL mới
<code>&lt;c:url&gt;</code>	Tạo một URL với các tham số truy vấn tùy ý

### Formatting Tags (JSTL)

Cú pháp

```
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/html/fmt" %>
```

Thẻ	Miêu tả
<code>&lt;fmt:formatNumber&gt;</code>	Trả lại giá trị số với định dạng cụ thể
<code>&lt;fmt:parseNumber&gt;</code>	Parse biểu diễn chuỗi của một số, tiền tệ, phần trăm
<code>&lt;fmt:formatDate&gt;</code>	Định dạng một date/time bởi sử dụng Style và Pattern đã cho
<code>&lt;fmt:parseDate&gt;</code>	Parse biểu diễn chuỗi của một date/time
<code>&lt;fmt:bundle&gt;</code>	Gán một Resource Bundle để được sử dụng bởi phần thân thẻ

<b>&lt;fmt:setLocale&gt;</b>	Lưu giữ Locale đã cho trong biến cấu hình locale
<b>&lt;fmt:setBundle&gt;</b>	Gán một Resource Bundle và lưu giữ trong biến scope đã đặt tên hoặc biến cấu hình bundle
<b>&lt;fmt:timeZone&gt;</b>	Xác định timezone cho bất kỳ định dạng time nào hoặc parse các action được lặp trong phần thân của tag.
<b>&lt;fmt:setTimeZone&gt;</b>	Gán timezone đã cung cấp biến cấu hình time zone đó
<b>&lt;fmt:message&gt;</b>	Hiển thị một thông báo đa ngôn ngữ
<b>&lt;fmt:requestEncoding&gt;</b>	Thiết lập mã hóa ký tự cho request

## SQL Tags (JSTL)

Cú pháp

```
<%@ taglib prefix="sql" uri="http://java.sun.com/jsp/html/sql" %>
```

Thẻ	Miêu tả
<b>&lt;sql:setDataSource&gt;</b>	Tạo một DataSource kết nối vào hệ quản trị cơ sở dữ liệu.
<b>&lt;sql:query&gt;</b>	Thực thi SQL query được định nghĩa trong tag đóng/mở hoặc thông qua thuộc tính sql
<b>&lt;sql:update&gt;</b>	Thực thi SQL update được định nghĩa trong tag đóng/mở hoặc thông qua thuộc tính sql
<b>&lt;sql:param&gt;</b>	Thiết lập một parameter trong một lệnh SQL
<b>&lt;sql:dateParam&gt;</b>	Thiết lập một parameter trong một lệnh SQL với giá trị java.util.Date đã xác định
<b>&lt;sql:transaction&gt;</b>	Cung cấp các phần tử database action được lặp với một Connection đã chia sẻ, thiết lập để thực thi tất cả các lệnh

## XML Tags (JSTL)

Cú pháp

```
<%@ taglib prefix="x" uri="http://java.sun.com/jsp/html/xml" %>
```

Thư viện

- **XercesImpl.jar:** Tải từ <http://www.apache.org/dist/xerces/j/>
- **xalan.jar:** Tải từ <http://xml.apache.org/xalan-j/index.html>

Thẻ	Miêu tả
<b>&lt;x:out&gt;</b>	Giống <%= ... >, nhưng dành cho các XPath Expression

<b>&lt;x:parse&gt;</b>	Sử dụng để parse XML data được xác định hoặc thông qua một thuộc tính hoặc trong phần thân thẻ
<b>&lt;x:set &gt;</b>	Thiết lập một biến tới giá trị của một XPath expression
<b>&lt;x:if &gt;</b>	Ước lượng XPath Expression và nếu nó là true, thì xử lý phần thân thẻ. Nếu điều kiện là false, phần thân bị bỏ qua
<b>&lt;x:forEach&gt;</b>	Lặp qua các node trong một tài liệu XML
<b>&lt;x:choose&gt;</b>	Điều kiện đơn giản mà thiết lập một context cho hoạt động điều kiện loại trừ, được đánh dấu bởi <when> và <otherwise> trong JSTL
<b>&lt;x:when &gt;</b>	Thẻ phụ của <choose>.
<b>&lt;x:otherwise &gt;</b>	Thẻ phụ của <choose>.
<b>&lt;x:transform &gt;</b>	Áp dụng một phép biến đổi XSL trên một tài liệu XML
<b>&lt;x:param &gt;</b>	Sử dụng cùng với thẻ transform để thiết lập một parameter trong XSLT stylesheets

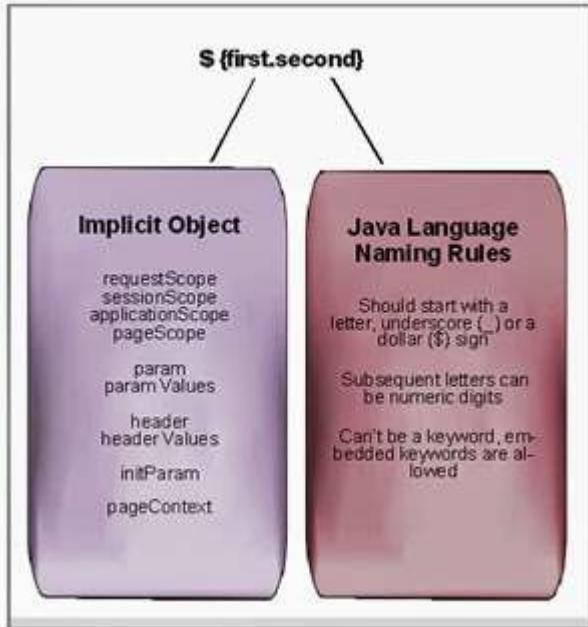
### JSTL Functions

Cú pháp

```
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/html/functions" %>
```

Hàm	Miêu tả
<b>Hàm fn:contains()</b>	Kiểm tra nếu một chuỗi input chứa chuỗi phụ đã cho
<b>Hàm fn:containsIgnoreCase()</b>	Kiểm tra nếu một chuỗi input chứa chuỗi phụ đã cho trong trường hợp không phân biệt kiểu chữ
<b>Hàm fn:endsWith()</b>	Kiểm tra nếu một chuỗi input kết thúc với suffix đã cho
<b>Hàm fn:escapeXml()</b>	Các ký tự thoát mà có thể được phiên dịch như XML markup
<b>Hàm fn:indexOf()</b>	Trả về index bên trong một chuỗi về sự xuất hiện đầu tiên của chuỗi phụ
<b>Hàm fn:join()</b>	Kết hợp tất cả phần tử trong một mảng thành một chuỗi
<b>Hàm fn:length()</b>	Trả về số item trong một tập hợp, hoặc số ký tự trong một chuỗi
<b>Hàm fn:replace()</b>	Trả về một chuỗi là kết quả của việc thay thế một chuỗi input với một chuỗi đã cho
<b>Hàm fn:split()</b>	Chia một chuỗi thành một mảng các chuỗi phụ
<b>Hàm fn:startsWith()</b>	Kiểm tra nếu một chuỗi input bắt đầu với prefix đã cho
<b>Hàm fn:substring()</b>	Trả về một tập con của một chuỗi
<b>Hàm fn:substringAfter()</b>	Trả về một tập con của một chuỗi ở sau một chuỗi phụ đã cho
<b>Hàm fn:substringBefore()</b>	Trả về một tập con của một chuỗi ở trước một chuỗi phụ đã cho
<b>Hàm fn:toLowerCase()</b>	Biến đổi tất cả ký tự của một chuỗi thành chữ thường
<b>Hàm fn:toUpperCase()</b>	Biến đổi tất cả ký tự của một chuỗi thành chữ hoa
<b>Hàm fn:trim()</b>	Gỡ bỏ các khoảng trắng từ hai đầu của một chuỗi

JSP Expression Language (EL là một phần của đặc tả của JSTL 1.0, đi kèm với JSP 1.2 và có thể được sử dụng như các attribute của thẻ JSTL)



Cú pháp: \${expression}

## Bài 1. JSPs - Thao tác với Form

Các bài tập Java Servlets thực hiện bằng JSPs.

Lưu ý dùng các actions khi cần thiết:

- jsp:forward, jsp:include
- jsp:useBean, jsp:setProperty, jsp:getProperty

Thực hiện form đăng ký khóa học cho sinh viên:

First name  (max 30 characters a-z and A-Z)

Last name  (max 30 characters a-z and A-Z)

Date of birth  Day:  Month:  Year:

Email

Mobile number  (10 digit number)

Gender Male  Female

Address

City  (max 30 characters a-z and A-Z)

Pin code  (6 digit number)

State  (max 30 characters a-z and A-Z)

Country  India

Hobbies Drawing  Singing  Dancing  Sketching  Others

Qualification

Sl.No.	Examination	Board	Percentage	Year of Passing
1	Class X	<input type="text"/>	<input type="text"/>	<input type="text"/>
2	Class XII	<input type="text"/>	<input type="text"/>	<input type="text"/>
3	Graduation	<input type="text"/>	<input type="text"/>	<input type="text"/>
4	Masters	<input type="text"/>	<input type="text"/>	<input type="text"/>

(10 char max) (upto 2 decimal)

Course applies for BCA  B.Com  B.Sc  B.A

B1. Tạo lớp Student bao gồm các thuộc tính mô tả trên form, các thuộc tính khai báo private kèm theo các phương thức get/set + constructors.

B2. Tạo form đăng ký <form action="RegistrationForm" name="formDangKy" method="GET">

B3. Tạo Servlet xử lý thao tác nhập dữ liệu của form:

```
@WebServlet("/RegistrationForm")
public class RegistrationForm extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter dt = response.getWriter();

    String fname = request.getParameter("txtFName");
    String lname = request.getParameter("txtLName");
    String day = request.getParameter("day");
    String month = request.getParameter("month");
    String year = request.getParameter("year");
    String email = request.getParameter("txtEmail");
    String mobileNum = request.getParameter("txtMobileNumber");
    String gender = request.getParameter("Gender");
    String address = request.getParameter("taAddress");
    String city = request.getParameter("txtCity");
    String pinCode = request.getParameter("txtPinCode");
    String state = request.getParameter("txtState");
    String country = request.getParameter("txtCountry");
    String hobbies = request.getParameter("ckHobbies");
    String course = request.getParameter("txtrCourse");

    String birthdate = day + " " + month + " " + year;

    Student sv = new Student();
    sv.setFirstName(fname);
    sv.setLastName(lname);
    sv.setEmail(email);
    sv.setGender(gender);
    sv.setBirthday(birthdate);

    request.setAttribute("student", sv);

    RequestDispatcher rd = request.getRequestDispatcher("ResultForm.jsp");
    rd.forward(request, response);
}

protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    doGet(request, response);
}
```

B4. Tạo trang JSP hiển thị kết quả

```

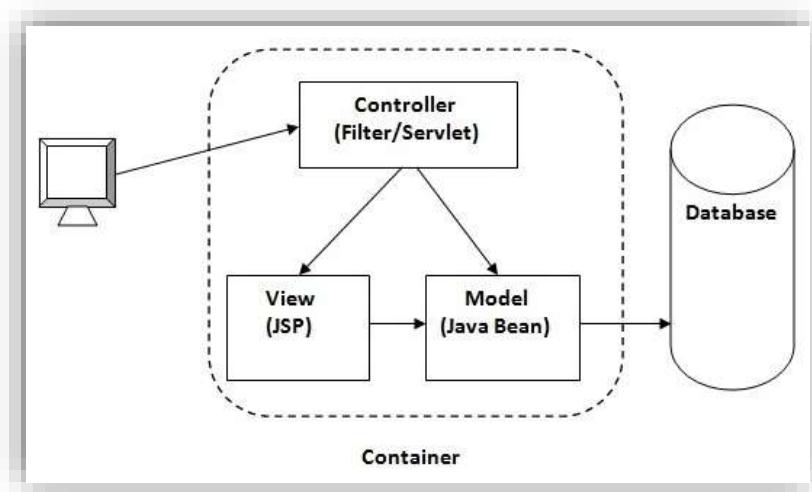
<%@page import="se.iuh.edu.vn.Student"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Result submit</title>
</head>

<body>
<%
Student svt= new Student();
svt = (Student)request.getAttribute("student");
out.println("First name:" + svt.getFirstName()
+ "<br/> Last name: " + svt.getLastName()
+ "<br/> Email : " + svt.getEmail()
+ "<br/> Gender: " + svt.getGender()
+ "<br/> Hobbies: " + svt.getHobbies()
+ "<br/> Birthday: " + svt.getBirthday()
);
%>
</body>
</html>

```

## Bài 2. JSPs - Model View Controller

Thực hiện form đăng ký tài khoản với JSPs. Sau khi đăng ký thành công cập nhật danh sách tài khoản (không hiển thị password), Thực hiện theo mô hình MVC.



## User Registration Form

First Name	Last Name	
Your Email		
Re-enter Email		
New Password		
Birthday		
Month	Day	Year
<input type="radio"/> Female	<input type="radio"/> Male	
Sign Up		

### Bài 3. JSPs - Xử lý đa ngôn ngữ

Thực hiện ứng dụng đa ngôn ngữ với JSPs.

Please select your language	<input type="radio"/> Vietnamese	<input checked="" type="radio"/> English	Choose
Username	<input type="text"/>		
Password	<input type="password"/>		
<input type="button" value="Login"/>			

Chọn ngôn ngữ	<input type="radio"/> Tiếng Việt	<input type="radio"/> Tiếng Anh	Chọn
Username	<input type="text"/>		
Mật khẩu	<input type="password"/>		
<input type="button" value="Đăng nhập"/>			

<http://www.oracle.com/technetwork/articles/javase/multilingualjsp-142726.html>

Danh sách Language/Country/Region

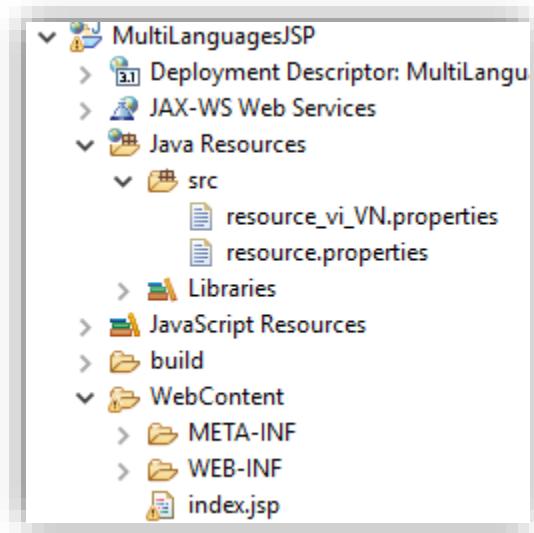
<b>Code</b>	<b>Language – Country/Region</b>	<b>Code</b>	<b>Language – Country/Region</b>
<b>af</b>	Afrikaans	hu-HU	Hungarian – Hungary
<b>af-ZA</b>	Afrikaans – South Africa	is	Icelandic
<b>sq</b>	Albanian	is-IS	Icelandic – Iceland
<b>sq-AL</b>	Albanian – Albania	id	Indonesian
<b>ar</b>	Arabic	id-ID	Indonesian – Indonesia
<b>ar-DZ</b>	Arabic – Algeria	it	Italian
<b>ar-BH</b>	Arabic – Bahrain	it-IT	Italian – Italy
<b>ar-EG</b>	Arabic – Egypt	it-CH	Italian – Switzerland
<b>ar-IQ</b>	Arabic – Iraq	ja	Japanese
<b>ar-JO</b>	Arabic – Jordan	ja-JP	Japanese – Japan
<b>ar-KW</b>	Arabic – Kuwait	kn	Kannada
<b>ar-LB</b>	Arabic – Lebanon	kn-IN	Kannada – India
<b>ar-LY</b>	Arabic – Libya	kk	Kazakh
<b>ar-MA</b>	Arabic – Morocco	kk-KZ	Kazakh – Kazakhstan
<b>ar-OM</b>	Arabic – Oman	kok	Konkani
<b>ar-QA</b>	Arabic – Qatar	kok-IN	Konkani – India
<b>ar-SA</b>	Arabic – Saudi Arabia	ko	Korean
<b>ar-SY</b>	Arabic – Syria	ko-KR	Korean – Korea
<b>ar-TN</b>	Arabic – Tunisia	ky	Kyrgyz
<b>ar-AE</b>	Arabic – United Arab Emirates	ky-KG	Kyrgyz – Kyrgyzstan
<b>ar-YE</b>	Arabic – Yemen	lv	Latvian
<b>hy</b>	Armenian	lv-LV	Latvian – Latvia
<b>hy-AM</b>	Armenian – Armenia	lt	Lithuanian
<b>az</b>	Azeri	lt-LT	Lithuanian – Lithuania
<b>az-AZ-Cyril</b>	Azeri (Cyrillic) – Azerbaijan	mk	Macedonian
<b>az-AZ-Latin</b>	Azeri (Latin) – Azerbaijan	mk-MK	Macedonian – Former Yugoslav Republic of Macedonia
<b>eu</b>	Basque	ms	Malay
<b>eu-ES</b>	Basque – Basque	ms-BN	Malay – Brunei
<b>be</b>	Belarusian	ms-MY	Malay – Malaysia
<b>be-BY</b>	Belarusian – Belarus	mr	Marathi
<b>bg</b>	Bulgarian	mr-IN	Marathi – India
<b>bg-BG</b>	Bulgarian – Bulgaria	mn	Mongolian
<b>ca</b>	Catalan	mn-MN	Mongolian – Mongolia
<b>ca-ES</b>	Catalan – Catalan	no	Norwegian
<b>zh-HK</b>	Chinese – Hong Kong SAR	nb-NO	Norwegian (Bokmål) – Norway
<b>zh-MO</b>	Chinese – Macao SAR	nn-NO	Norwegian (Nynorsk) – Norway
<b>zh-CN</b>	Chinese – China	pl	Polish
<b>zh-CHS</b>	Chinese (Simplified)	pl-PL	Polish – Poland
<b>zh-SG</b>	Chinese – Singapore	pt	Portuguese
<b>zh-TW</b>	Chinese – Taiwan	pt-BR	Portuguese – Brazil
<b>zh-CHT</b>	Chinese (Traditional)	pt-PT	Portuguese – Portugal

<b>hr</b>	Croatian	pa	Punjabi
<b>hr-HR</b>	Croatian – Croatia	pa-IN	Punjabi – India
<b>cs</b>	Czech	ro	Romanian
<b>cs-CZ</b>	Czech – Czech Republic	ro-RO	Romanian – Romania
<b>da</b>	Danish	ru	Russian
<b>da-DK</b>	Danish – Denmark	ru-RU	Russian – Russia
<b>div</b>	Dhivehi	sa	Sanskrit
<b>div-MV</b>	Dhivehi – Maldives	sa-IN	Sanskrit – India
<b>nl</b>	Dutch	sr-SP-Cyril	Serbian (Cyrillic) – Serbia
<b>nl-BE</b>	Dutch – Belgium	sr-SP-Latn	Serbian (Latin) – Serbia
<b>nl-NL</b>	Dutch – The Netherlands	sk	Slovak
<b>en</b>	English	sk-SK	Slovak – Slovakia
<b>en-AU</b>	English – Australia	sl	Slovenian
<b>en-BZ</b>	English – Belize	sl-SI	Slovenian – Slovenia
<b>en-CA</b>	English – Canada	es	Spanish
<b>en-CB</b>	English – Caribbean	es-AR	Spanish – Argentina
<b>en-IE</b>	English – Ireland	es-BO	Spanish – Bolivia
<b>en-JM</b>	English – Jamaica	es-CL	Spanish – Chile
<b>en-NZ</b>	English – New Zealand	es-CO	Spanish – Colombia
<b>en-PH</b>	English – Philippines	es-CR	Spanish – Costa Rica
<b>en-ZA</b>	English – South Africa	es-DO	Spanish – Dominican Republic
<b>en-TT</b>	English – Trinidad and Tobago	es-EC	Spanish – Ecuador
<b>en-GB</b>	English – United Kingdom	es-SV	Spanish – El Salvador
<b>en-US</b>	English – United States	es-GT	Spanish – Guatemala
<b>en-ZW</b>	English – Zimbabwe	es-HN	Spanish – Honduras
<b>et</b>	Estonian	es-MX	Spanish – Mexico
<b>et-EE</b>	Estonian – Estonia	es-NI	Spanish – Nicaragua
<b>fo</b>	Faroese	es-PA	Spanish – Panama
<b>fo-FO</b>	Faroese – Faroe Islands	es-PY	Spanish – Paraguay
<b>fa</b>	Farsi	es-PE	Spanish – Peru
<b>fa-IR</b>	Farsi – Iran	es-PR	Spanish – Puerto Rico
<b>fi</b>	Finnish	es-ES	Spanish – Spain
<b>fi-FI</b>	Finnish – Finland	es-UY	Spanish – Uruguay
<b>fr</b>	French	es-VE	Spanish – Venezuela
<b>fr-BE</b>	French – Belgium	sw	Swahili
<b>fr-CA</b>	French – Canada	sw-KE	Swahili – Kenya
<b>fr-FR</b>	French – France	sv	Swedish
<b>fr-LU</b>	French – Luxembourg	sv-FI	Swedish – Finland
<b>fr-MC</b>	French – Monaco	sv-SE	Swedish – Sweden
<b>fr-CH</b>	French – Switzerland	syr	Syriac
<b>gl</b>	Galician	syr-SY	Syriac – Syria
<b>gl-ES</b>	Galician – Galician	ta	Tamil
<b>ka</b>	Georgian	ta-IN	Tamil – India
<b>ka-GE</b>	Georgian – Georgia	tt	Tatar

<b>de</b>	German	tt-RU	Tatar – Russia
<b>de-AT</b>	German – Austria	te	Telugu
<b>de-DE</b>	German – Germany	te-IN	Telugu – India
<b>de-LI</b>	German – Liechtenstein	th	Thai
<b>de-LU</b>	German – Luxembourg	th-TH	Thai – Thailand
<b>de-CH</b>	German – Switzerland	Tr	Turkish
<b>el</b>	Greek	tr-TR	Turkish – Turkey
<b>el-GR</b>	Greek – Greece	uk	Ukrainian
<b>gu</b>	Gujarati	uk-UA	Ukrainian – Ukraine
<b>gu-IN</b>	Gujarati – India	ur	Urdu
<b>he</b>	Hebrew	ur-PK	Urdu – Pakistan
<b>he-IL</b>	Hebrew – Israel	uz	Uzbek
<b>hi</b>	Hindi	uz-UZ-Cyrl	Uzbek (Cyrillic) – Uzbekistan
<b>hi-IN</b>	Hindi – India	uz-UZ-Latn	Uzbek (Latin) – Uzbekistan
<b>hu</b>	Hungarian	vi	Vietnamese

## Tham khảo hướng dẫn

Project gồm các file resource cho từng ngôn ngữ, cần khai báo thư viện cho JSTL.



### B1. Tạo các files liên quan đến ngôn ngữ

*resource\_vi\_VN.properties (Language vi, Country/Region VN)*

```
languagemessase = Please select your language
vn = Vietnamese
en = English
choosebutton = Choose
username = Username
pass = Password
login = Login
```

*resource.properties*

```
languagemessase = Chọn ngôn ngữ
vn = Tiếng Việt
en = Tiếng Anh
choosebutton = Chọn
username = Username
pass = Mật khẩu
login = Đăng nhập
```

### B2. Thực hiện đa ngôn ngữ trong file JSP.

Khai báo các taglib

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
%@taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="f"%>
```

Dùng `<f:setLocale value="${mangonngu}" scope="session" />` lấy mã ngôn ngữ cần xử lý, gán file resource liên quan đến ngôn ngữ: `<f:setBundle basename="resource" scope="session" />`

```

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="f"%>
<%@page import="java.sql.*"%>

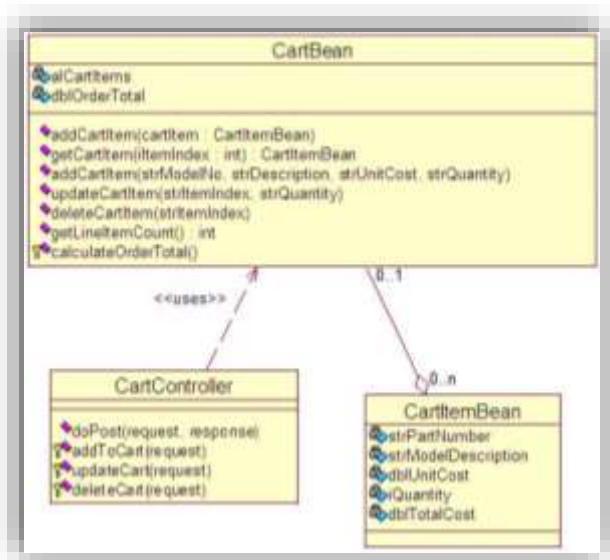
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
<body>
    <c:set var="mangonngu" value="${param.radio_mangonngu}" />
    <c:if test="${not empty mangonngu}">
        <f:setLocale value="${mangonngu}" scope="session" />
    </c:if>
    <br>
    <f:setBundle basename="resource" scope="session" />
    <form action="index.jsp" method="post">
        <br>
        <f:message key="Langagemessase" />
        <input type="radio" name="radio_mangonngu" value="vi-VN"
               <c:if test="${mangonngu=='vi-VN'}">checked</c:if> />
        <f:message key="vn" />
        <input type="radio" name="radio_mangonngu" value="en-US"
               <c:if test="${mangonngu=='en-US'}">checked</c:if> />
        <f:message key="en" />
        <input type="submit" name="submit" value=<f:message key='choosebutton' /> /> <br>
        <hr />

        <table border="0">
            <tr>
                <td><f:message key="username" /></td>
                <td><input type="text" name="txtusername" value="" /></td>
            </tr>
            <tr>
                <td><f:message key="pass" /></td>
                <td><input type="text" name="txtpassword" value="" /> <br></td>
            </tr>
            <tr>
                <td colspan="2" > <input type="submit" name="submit" value=<f:message
key='Login' /> /> </td>
            </tr>
        </table>
    </form>
</body>
</html>

```

#### Bài 4. JSPs - Thao tác với JSP Session

Thực hiện website mua bán sản phẩm đơn giản dùng JSP Session.



Hiển thị danh sách sản phẩm và cho thêm sản phẩm vào giỏ hàng

The screenshot shows a web page titled "View Cart" displaying a grid of six mobile phone products. Each product card includes an image, name, price, a quantity input field set to 1, and an "Add To Cart" button.

Product	Image	Price	Quantity	Action
Nokia Lumia		99000.0	1	Add To Cart
BlackBerry Passport		48000.0	1	Add To Cart
Sony Xperia Z5		52000.0	1	Add To Cart
HTC One M9		83000.0	1	Add To Cart
Samsung Galaxy Note 5		71000.0	1	Add To Cart
iPhone 7 jet-black Plus		120000.0	1	Add To Cart

Khi giỏ hàng không có gì, hiển thị thông báo

Model Description	Quantity	Unit Price	Total
Cart is currently empty -			
			Subtotal: \$0.0

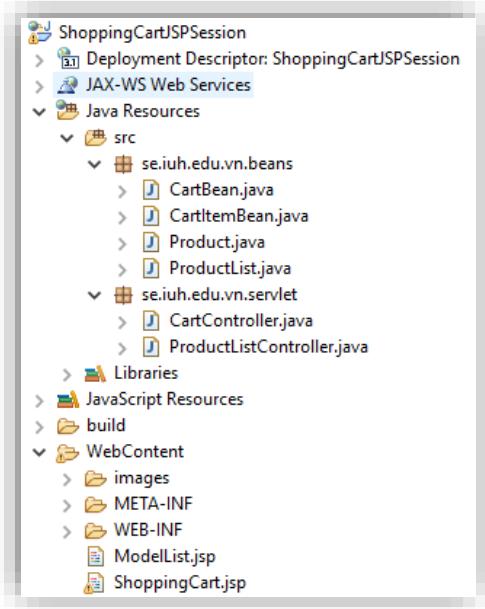
Khi giỏ hàng có sản phẩm, cho phép xóa, sửa các sản phẩm.

Model Description	Quantity	Unit Price	Total
PRO02 BlackBerry Passport	1	<input type="button" value="Update"/> <input type="button" value="Delete"/>	\$48000.0 \$48000.0
PRO06 iPhone 7 jet-black Plus	1	<input type="button" value="Update"/> <input type="button" value="Delete"/>	\$120000.0 \$120000.0
			Subtotal: \$168000.0

### Tham khảo hướng dẫn

B1. Tạo Dynamic Web Project gồm các phần:

- Model: Product (sản phẩm), CartBean (giỏ hàng) và CartBeanItem(các sản phẩm trong giỏ hàng) và ProductList (danh sách các sản phẩm, lưu trong collection hoặc truy vấn từ CSDL)
- View: ModelList (danh sách sản phẩm) và ShoppingCart (giỏ hàng)
- Controller: CartController dùng để xử lý thao tác nghiệp vụ đối với giỏ hàng và ProductListController dùng để xử lý việc hiển thị danh sách sản phẩm.



## B2. Tạo Model

Lớp *Product.java*

```
package se.iuh.edu.vn.beans;
public class Product {
    private String id;
    private String model;
    private String description;
    private int quantity;
    private double price;
    private String imgURL;

    public String getImgURL() {..}

    public void setImgURL(String imgURL) {..}

    public String getId() {..}

    public void setId(String id) {..}

    public String getDescription() {..}

    public void setDescription(String description) {..}

    public int getQuantity() {..}

    public void setQuantity(int quantity) {..}

    public double getPrice() {..}

    public void setPrice(double price) {..}

    public String getModel() {..}

    public void setModel(String model) {..}

}
```

## Lớp CartItemBean, lưu thông tin từng thành phần trong giỏ hàng

```
package se.iuh.edu.vn.beans;
public class CartItemBean {
    private String strPartNumber;
    private String strModelDescription;
    private double dblUnitCost;
    private int iQuantity;
    private double dblTotalCost;

    public String getPartNumber() {
        return strPartNumber;
    }

    public void setPartNumber(String strPartNumber) {
        this.strPartNumber = strPartNumber;
    }

    public String getModelDescription() {
        return strModelDescription;
    }

    public void setModelDescription(String strModelDescription) {
        this.strModelDescription = strModelDescription;
    }

    public double getUnitCost() {
        return dblUnitCost;
    }

    public void setUnitCost(double dblUnitCost) {
        this.dblUnitCost = dblUnitCost;
    }

    public int getQuantity() {
        return iQuantity;
    }

    public void setQuantity(int quantity) {
        iQuantity = quantity;
    }

    public double getTotalCost() {
        return dblTotalCost;
    }

    public void setTotalCost(double dblTotalCost) {
        this.dblTotalCost = dblTotalCost;
    }
}
```

B3. Xử lý các thao tác liên quan đến giỏ hàng: thêm sản phẩm vào giỏ hàng, xóa sản phẩm, cập nhật số lượng: CartBean.java

```
package se.iuh.edu.vn.beans;

import java.util.ArrayList;

public class CartBean {
    private ArrayList<CartItemBean> alCartItems = new ArrayList<CartItemBean>();
    private double dblOrderTotal;

    public int getLineItemCount() { }

    public void deleteCartItem(String strItemIndex) { }

    public void updateCartItem(String strItemIndex, String strQuantity) { }

    public void addCartItem(String strModelNo, String strDescription, String strUnitCost, String strQuantity) { }

    public void addCartItem(CartItemBean cartItem) { }

    public CartItemBean getCartItem(int iItemIndex) { }

    public ArrayList<CartItemBean> getCartItems() { }

    public void setCartItems(ArrayList<CartItemBean> alCartItems) { }

    public double getOrderTotal() { }

    public void setOrderTotal(double dblOrderTotal) { }

    protected void calculateOrderTotal() { }

}
```

Các hàm xử lý trong CartBean:

```
public int getLineItemCount() {
    return alCartItems.size();
}

public void deleteCartItem(String strItemIndex) {
    int iItemIndex = 0;
    try {
        iItemIndex = Integer.parseInt(strItemIndex);
        alCartItems.remove(iItemIndex - 1);
        calculateOrderTotal();
    } catch (NumberFormatException nfe) {
        System.out.println("Error while deleting cart item: " + nfe.getMessage());
        nfe.printStackTrace();
    }
}
```

```

public void updateCartItem(String strItemIndex, String strQuantity) {
    double dblTotalCost = 0.0;
    double dblUnitCost = 0.0;
    int iQuantity = 0;
    int iItemIndex = 0;
    CartItemBean cartItem = null;
    try {
        iItemIndex = Integer.parseInt(strItemIndex);
        iQuantity = Integer.parseInt(strQuantity);
        if (iQuantity > 0) {
            cartItem = (CartItemBean) alCartItems.get(iItemIndex - 1);
            dblUnitCost = cartItem.getUnitCost();
            dblTotalCost = dblUnitCost * iQuantity;
            cartItem.setQuantity(iQuantity);
            cartItem.setTotalCost(dblTotalCost);
            calculateOrderTotal();
        }
    } catch (NumberFormatException nfe) {
        System.out.println("Error while updating cart: " + nfe.getMessage());
        nfe.printStackTrace();
    }
}

```

```

public CartItemBean getCartItem(int iItemIndex) {
    CartItemBean cartItem = null;
    if (alCartItems.size() > iItemIndex) {
        cartItem = (CartItemBean) alCartItems.get(iItemIndex);
    }
    return cartItem;
}

public ArrayList<CartItemBean> getCartItems() {
    return alCartItems;
}

public void setCartItems(ArrayList<CartItemBean> alCartItems) {
    this.alCartItems = alCartItems;
}

public double getOrderTotal() {
    return dblOrderTotal;
}

public void setOrderTotal(double dblOrderTotal) {
    this.dblOrderTotal = dblOrderTotal;
}

protected void calculateOrderTotal() {
    double dblTotal = 0;
    for (int counter = 0; counter < alCartItems.size(); counter++) {
        CartItemBean cartItem = (CartItemBean) alCartItems.get(counter);
        dblTotal += cartItem.getTotalCost();
    }
    setOrderTotal(dblTotal);
}

```

```
public void addCartItem(String strModelNo, String strDescription, String strUnitCost, String strQuantity)
{
    double dblTotalCost = 0.0;
    double dblUnitCost = 0.0;
    int iQuantity = 0;
    CartItemBean cartItem = new CartItemBean();
    try {
        dblUnitCost = Double.parseDouble(strUnitCost);
        iQuantity = Integer.parseInt(strQuantity);
        if (iQuantity > 0) {
            dblTotalCost = dblUnitCost * iQuantity;
            cartItem.setPartNumber(strModelNo);
            cartItem.setModelDescription(strDescription);
            cartItem.setUnitCost(dblUnitCost);
            cartItem.setQuantity(iQuantity);
            cartItem.setTotalCost(dblTotalCost);
            alCartItems.add(cartItem);
            calculateOrderTotal();
        }
    } catch (NumberFormatException nfe) {
        System.out.println("Error while parsing from String to primitive types: " + nfe.getMessage());
        nfe.printStackTrace();
    }
}

public void addCartItem(CartItemBean cartItem) {
    alCartItems.add(cartItem);
}
```

#### B4. Tạo danh sách sản phẩm (dùng Collection, có thể lấy từ CSDL cũ thế khác)

```
package se.iuh.edu.vn.beans;

import java.util.ArrayList;
import java.util.List;

public class ProductList {

    private static final List<Product> ds = new ArrayList<Product>();
    static {
        initData();
    }

    public static List<Product> queryProducts() {
        return ds;
    }

    private static void initData() {
        Product sp = new Product();
        sp.setId("PRO01");
        sp.setDescription("");
        sp.setPrice(99000);
        ds.add(sp);

        sp = new Product();
        sp.setId("PRO02");
        sp.setDescription("");
        sp.setPrice(48000);
        ds.add(sp);

        sp = new Product();
        sp.setId("PRO03");
        sp.setDescription("");
        sp.setPrice(52000);
        ds.add(sp);

        sp = new Product();
        sp.setId("PRO04");
        sp.setDescription("");
        sp.setPrice(83000);
        ds.add(sp);

        sp = new Product();
        sp.setId("PRO05");
        sp.setDescription("");
        sp.setPrice(71000);
        ds.add(sp);
    }
}
```

## B5. Tạo Servlet nhận yêu cầu xử lý request/response của Client: CartController.java

```
package se.iuh.edu.vn.servlet;

import se.iuh.edu.vn.beans.CartBean;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebServlet("/CartController")
public class CartController extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public void doPost(HttpServletRequest request, HttpServletResponse response) {}

    protected void deleteCart(HttpServletRequest request) {}

    protected void updateCart(HttpServletRequest request) {}

    protected void addToCart(HttpServletRequest request) {}

}
```

### Các hàm xử lý trong CartController.java

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    String strAction = request.getParameter("action");

    if (strAction != null && !strAction.equals("")) {
        if (strAction.equals("add")) {
            addToCart(request);
        } else if (strAction.equals("Update")) {
            updateCart(request);
        } else if (strAction.equals("Delete")) {
            deleteCart(request);
        }
    }
    response.sendRedirect("/ShoppingCartJSPSession/ShoppingCart.jsp");
}

protected void deleteCart(HttpServletRequest request) {
    HttpSession session = request.getSession();
    String strItemIndex = request.getParameter("itemIndex");
    CartBean cartBean = null;

    Object objCartBean = session.getAttribute("cart");
    if (objCartBean != null) {
        cartBean = (CartBean) objCartBean;
    } else {
        cartBean = new CartBean();
    }
    cartBean.deleteCartItem(strItemIndex);
}
```

```

protected void updateCart(HttpServletRequest request) {
    HttpSession session = request.getSession();
    String strQuantity = request.getParameter("quantity");
    String strItemIndex = request.getParameter("itemIndex");

    CartBean cartBean = null;

    Object objCartBean = session.getAttribute("cart");
    if (objCartBean != null) {
        cartBean = (CartBean) objCartBean;
    } else {
        cartBean = new CartBean();
    }
    cartBean.updateCartItem(strItemIndex, strQuantity);
}

protected void addToCart(HttpServletRequest request) {
    HttpSession session = request.getSession();
    String strModelNo = request.getParameter("modelNo");
    String strDescription = request.getParameter("description");
    String strPrice = request.getParameter("price");
    String strQuantity = request.getParameter("quantity");

    CartBean cartBean = null;

    Object objCartBean = session.getAttribute("cart");

    if (objCartBean != null) {
        cartBean = (CartBean) objCartBean;
    } else {
        cartBean = new CartBean();
        session.setAttribute("cart", cartBean);
    }

    cartBean.addCartItem(strModelNo, strDescription, strPrice, strQuantity);
}

```

## B5. Tạo Servlet hiển thị danh sách: ProductListController.java

```
package se.iuh.edu.vn.servlet;

import java.io.IOException;

@WebServlet("/DSSP")
public class ProductListController extends HttpServlet {
    private static final long serialVersionUID = 1L;

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        List<Product> list = ProductList.queryProducts();
        request.setAttribute("ds", list);
        RequestDispatcher dispatcher = getServletContext().getRequestDispatcher("/ModelList.jsp");
        dispatcher.forward(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        doGet(request, response);
    }
}
```

## B6. Tạo giao diện hiển thị danh sách và cho phép thêm sản phẩm vào giỏ hàng (View): ModelList.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Product List</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<style>
.a {
    width: 160px; height: 200px; border: 1px solid black;
    padding: 5px; margin: 10px;
    float: left; text-align: center;
}

.hinh {
    width: 80px; height: 100px;
}
</style>
</head>

<body>
    <p><a href="/ShoppingCartJSPSession/ShoppingCart.jsp">View Cart</a></p>

    <c:forEach items="${ds}" var="sp">
        <div class="a">
            <form name="modelDetail" method="POST"
                  action="/ShoppingCartJSPSession/CartController">
                ${sp.model} <br />
                 <br />
                Price: ${sp.price} <br />
                <input type="text" size="2" value="1" name="quantity"><br>
                <input type="hidden" name="modelNo" value="${sp.id}">
                <input type="hidden" name="price" value="${sp.price}">
                <input type="hidden" name="description" value="${sp.model}">
                <input type="hidden" name="action" value="add">
                <input type="submit" name="addToCart" value="Add To Cart">
            </form>
        </div>
    </c:forEach>
</body>
</html>
```

## B7: Tạo giao diện hỗ trợ xử lý trong giỏ hàng: JSP ShoppingCart.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<jsp:useBean id="cart" scope="session" class="se.iuh.edu.vn.beans.CartBean" />

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head><title>Shopping Cart</title>
</head>
<body>
    <p><a href="/ShoppingCartJSPSession/DSSP">Product List</a></p>
    <table width="100%" border="1">
        <tr bgcolor="#CCCCCC">
            <td>Model Description</td> <td>Quantity</td>
            <td>Unit Price</td> <td>Total</td>
        </tr>

        <c:if test="${cart.lineItemCount==0}">
            <tr>
                <td colspan="4">Cart is currently empty -<br />
            </tr>
        </c:if>
        <c:forEach var="cartItem" items="${cart.cartItems}" varStatus="counter">
            <form name="item" method="POST" action="/ShoppingCartJSPSession/CartController">
                <tr>
                    <td>
                        <c:out value="${cartItem.partNumber}" /></b><br />
                        <c:out value="${cartItem.modelDescription}" /></td>
                    <td>
                        <input type='hidden' name='itemIndex' value='<c:out
value="${counter.count}" />'>
                        <input type='text' name="quantity" value='<c:out
value="${cartItem.quantity}" />' size='2'>
                        <input type="submit" name="action" value="Update">
                        <input type="submit" name="action" value="Delete">
                    </td>
                    <td><c:out value="${cartItem.unitCost}" /></td>
                    <td><c:out value="${cartItem.totalCost}" /></td>
                </tr>
            </form>
        </c:forEach>
        <tr>
            <td colspan="2"></td>
            <td></td>
            <td>Subtotal: <c:out value="${cart.orderTotal}" /></td>
        </tr>
    </table>
</body>
</html>
```

## Bài 5. JSPs - Thao tác với Shopping Cart

Thực hiện Shopping Cart với Session trong JSP, chỉ lưu CSDL khi thanh toán.

Thực hiện các trang: *DanhSachSP.jsp*

The screenshot shows a website for "IUH BOOKSTORE". The top navigation bar includes links for HOME, EXAMPLES, SERVICES, PRODUCTS, and CONTACT. On the left, there's a "ABOUT US" section with a placeholder for "About us information will be here....." and a "Read More" link. Below it is a "SEARCH SITE" input field. The main content area displays six book products in a 2x3 grid:

- Sô tay viết văn - Tác giả: Tô Hoài**  

Price: 99000  
Quantity: 10  
[Product details](#)  
[Add to cart](#)
- Nhật ký ma cà rồng 03 - CÔN THỊNH NỘ**  

Price: 48000  
Quantity: 10  
[Product details](#)  
[Add to cart](#)
- Billy và Beth - 3 bí quyết của người thành đạt**  

Price: 52000  
Quantity: 10  
[Product details](#)  
[Add to cart](#)
- Bí ẩn một cái tên - Dịch giả: Nguyễn Hồng Dung**  

Price: 83000  
Quantity: 10  
[Product details](#)  
[Add to cart](#)
- Kẻ trộm sách - Tác giả: Markus Zusak**  

Price: 71000  
Quantity: 10  
[Product details](#)  
[Add to cart](#)
- Cơm & Phở - Tác giả: Xuân Sách**  

Price: 120000  
Quantity: 10  
[Product details](#)  
[Add to cart](#)

Trang *DanhSachSP.jsp* khi đã có mua sản phẩm trong giỏ hàng

## ABOUT US

About us information will be here..... [Read More »](#)

## SEARCH SITE

[Shopping cart \(4\)](#)

Sô tay viết văn - Tác giả: Tô Hoài



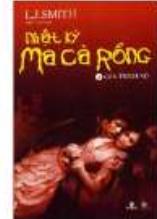
Price: 99000

Quantity: 10

[Product details](#)

[Add to cart](#)

Nhật ký ma cà rồng 03 - CƠN THỊNH NỘ



Price: 48000

Quantity: 10

[Product details](#)

[Add to cart](#)

Billy và Beth - 3 bí quyết của người thành đạt



Price: 52000

Quantity: 10

[Product details](#)

[Add to cart](#)

Bí ẩn một cái tên - Dịch giả: Nguyễn Hồng Dung



Price: 83000

Quantity: 10

[Product details](#)

[Add to cart](#)

Ké trộm sách - Tác giả: Markus Zusak



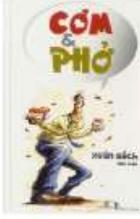
Price: 71000

Quantity: 10

[Product details](#)

[Add to cart](#)

Cơm & Phở - Tác giả: Xuân Sách.



Price: 120000

Quantity: 10

[Product details](#)

[Add to cart](#)

## ChiTietSP.jsp

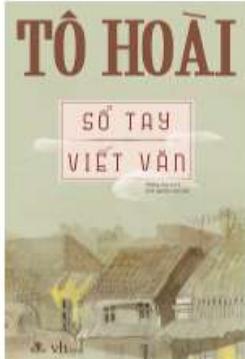
**IUH BOOKSTORE**

HOME EXAMPLES SERVICES PRODUCTS CONTACT

ABOUT US  
About us information will be here..... [Read More »](#)

SEARCH SITE

Product details: Sô tay viết văn - Tác giả: Tô Hoài



Price (VND): 99000  
Quantity: 10

[Back to Product List](#)

© IUH Bookstore 2016. All rights reversed.

## Trang XuLyMuaHang.jsp

**IUH BOOKSTORE**

HOME EXAMPLES SERVICES PRODUCTS CONTACT

ABOUT US  
About us information will be here..... [Read More »](#)

SEARCH SITE

[Shopping cart \(4\)](#)

YOUR SHOPPING CART						
Product ID	Product name	Price	Qty	Total	Remove	
pro01	Sô tay viết văn - Tác giả: Tô Hoài	99000	1	99000	<a href="#">Remove</a>	
pro02	Nhật ký ma cà rồng 03 - CƠN THỊNH NỘ	48000	1	48000	<a href="#">Remove</a>	
pro03	Billy và Beth - 3 bí quyết của người thành đạt	52000	3	156000	<a href="#">Remove</a>	
pro06	Com & Phò - Tác giả: Xuân Sách.	120000	1	120000	<a href="#">Remove</a>	

Total price (VND) 423000

[Checkout](#) [Continue shopping](#)

© IUH Bookstore 2016. All rights reversed.

## Trang XuLyThanhToan.jsp

## ABOUT US

About us information will be here..... [Read More »](#)

## SEARCH SITE

[Shopping cart \(4\)](#)

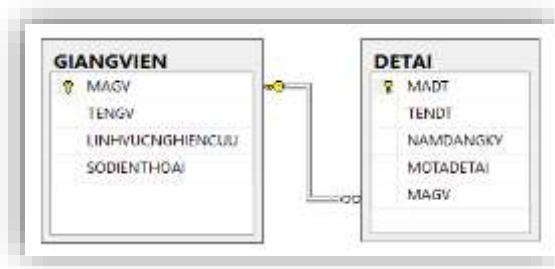
Checkout - Already registered? ...

Fullname:	<input type="text"/>
Shipping address:	<input type="text"/>
Total price:	423000
Payment method	<input type="radio"/> Paypal <input type="radio"/> ATM Debit <input type="radio"/> Visa/Master card
<input type="button" value="Save"/> <input type="button" value="Cancel"/>	

© IUH Bookstore 2016. All rights reversed.

Bài 6. JSPs - Standard Tag Library và Expression Language

Quản lý thông tin về đè tài và giảng viên ra đè tài tốt nghiệp cho sinh viên ngành công nghệ thông tin được mô tả như sau:



Dùng JSP Standard Tag Library và Expression Language thực hiện việc:

- Truy vấn hiển thị thông tin
- Thêm mới thông tin giảng viên/đè tài

Faculty ID	Full Name	Research Area	Telephone Number
GV0001	Nguyễn Bùi Thanh Hòa <u>Danh sách đề tài:</u> • Tìm hiểu Support Vector Machine	Lĩnh vực về Khoa học cơ bản	0909123456
GV0002	Hồ Huỳnh Thùy An <u>Danh sách đề tài:</u> • Tìm hiểu Mikrotik Router	Lĩnh vực về Công nghệ thông tin	0918456894
GV0003	Lê Hoàng Hóa <u>Danh sách đề tài:</u> • Xây dựng hệ thống quản lý hệ thống siêu thị Coop Mart	Lĩnh vực về Kỹ thuật phần mềm	0988345576
GV0004	Trần Bảo Châu <u>Danh sách đề tài:</u> • Xây dựng website thương mại điện tử • Xây dựng phần mềm quản lý nhân sự, tiền lương • Xây dựng ứng dụng quản lý thư viện	Lĩnh vực về Hệ thống thông tin	0914324578
GV0005	Nguyễn Lê Giao <u>Danh sách đề tài:</u> • Tìm hiểu BigData với AWS	Lĩnh vực về Khoa học máy tính	0978123453
GV0006	Hoàng Thị Ngọc Ánh <u>Danh sách đề tài:</u> • Web Development with Machine Learning	Điện tử công nghiệp	0988765434

### Tham khảo hướng dẫn

B1. Tạo CSDL và các bảng liên quan

```

create database QuanLyGiangVienDeTai

create table GIANGVIEN(
    MAGV nvarchar(20) primary key,
    TENGV nvarchar(100) not null,
    LINHVUCNGHIECUU nvarchar(250),
    SODIENTHOAI nvarchar(50)
)

create table DETAI(
    MADT nvarchar(20) primary key,
    TENDT nvarchar(100) not null,
    NAMDANGKY int,
    MOTADETAI nvarchar(250),
    MAGV nvarchar(20) references GIANGVIEN(MAGV) on delete cascade on update cascade
)

insert into GIANGVIEN values(N'GV0001',N'Nguyễn Bùi Thanh Hòa', N'Lĩnh vực về Khoa học cơ bản', '0909123456')
insert into GIANGVIEN values(N'GV0002',N'Hồ Huỳnh Thùy An', N'Lĩnh vực về Công nghệ thông tin', '0918456894')
insert into GIANGVIEN values(N'GV0003',N'Lê Hoàng Hóa', N'Lĩnh vực về Kỹ thuật phần mềm', '0988345576')
insert into GIANGVIEN values(N'GV0004',N'Trần Bảo Châu', N'Lĩnh vực về Hệ thống thông tin', '0914324578')
insert into GIANGVIEN values(N'GV0005',N'Nguyễn Lê Giao', N'Lĩnh vực về Khoa học máy tính', '0978123453')
insert into GIANGVIEN values(N'GV0006',N'Hoàng Thị Ngọc Ánh', N'Điện tử công nghiệp', '0988765434')

insert into DETAI values(N'DT0001',N'Tìm hiểu Mikrotik Router',2016, N'Tìm hiểu Mikrotik Router và xây dựng Demo hệ thống Hotspot Gateway cho dịch vụ Internet LAN Wifi có chứng thực trình bày các nội dung về các khái quát về công nghệ Wireless LAN',N'GV0002')
insert into DETAI values(N'DT0002',N'Xây dựng website thương mại điện tử',2015, N'Thiết kế và xây dựng website thương mại điện tử cho công ty X',N'gv0004')
insert into DETAI values(N'DT0003',N'Xây dựng hệ thống quản lý hệ thống siêu thị Coop Mart',2015, N'Nhiệm vụ của đề tài là trình bày, phân tích và thiết kế phần mềm về tổ chức siêu thị, xác định mô hình hoạt động, xác định hiện trạng và phạm vi của hệ thống.',N'GV0003')
insert into DETAI values(N'DT0004',N'Xây dựng phần mềm quản lý nhân sự, tiền lương',2015, N'Phân tích, thiết kế, xây dựng phần mềm quản lý nhân sự, tiền lương cho công ty Y',N'GV0004')
insert into DETAI values(N'DT0005',N'Xây dựng ứng dụng quản lý thư viện',2015, N'Phân tích, thiết kế, xây dựng phần mềm quản lý thư viện cho trường cao đẳng N',N'GV0004')

```

## B2. Import các thư viện liên quan

```

<!-- http://mvnrepository.com/artifact/org.apache.taglibs/taglibs-standard-spec -->
<dependency>
    <groupId>org.apache.taglibs</groupId>
    <artifactId>taglibs-standard-spec</artifactId>
    <version>1.2.5</version>
</dependency>
<!-- http://mvnrepository.com/artifact/org.apache.taglibs/taglibs-standard-impl -->
<dependency>
    <groupId>org.apache.taglibs</groupId>
    <artifactId>taglibs-standard-impl</artifactId>
    <version>1.2.5</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.microsoft.sqlserver/sqljdbc4 -->
<dependency>
    <groupId>com.microsoft.sqlserver</groupId>
    <artifactId>sqljdbc4</artifactId>
    <version>4.0</version>
    <scope>test</scope>
</dependency>

```

## B3. Thực hiện truy vấn dữ liệu dùng JSTL

Lấy dữ liệu <sql:query> </sql:query>

```

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ page import="javax.servlet.http.*,javax.servlet.*"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head> <title>Giảng viên - Đề tài</title></head>
<body>
    <sql:setDataSource var="snapshot" user="sa" password="1234567890"
        driver="com.microsoft.sqlserver.jdbc.SQLServerDriver"
        url="jdbc:sqlserver://localhost:1433;databaseName=QuanLyGiangVienDeTai" />
    <sql:query dataSource="${snapshot}" var="result">
        SELECT * from GIANGVIEN;
    </sql:query>
    <table border="1" width="100%">
        <tr> <th>Faculty ID</th><th>Full Name</th><th>Research Area</th> <th>Telephone Number</th></tr>
        <c:forEach var="row" items="${result.rows}">
            <tr>
                <td><c:out value="${row.MaGV}" /></td>

                <td><c:out value="${row.TENGV}" /> <br/><u><i>Danh sách đề tài:</i></u>
                    <sql:query dataSource="${snapshot}" var="detaigv">
                        SELECT * from DETAI WHERE MAGV=?;
                        <sql:param value="${row.MaGV}" />
                    </sql:query>
                    <ul>
                        <c:forEach var="rowdt" items="${detaigv.rows}">
                            <li><c:out value="${rowdt.TENDT}" /></li>
                        </c:forEach>
                    </ul></td>
                <td><c:out value="${row.LINHVUCNGHIENCUU}" /></td>
                <td><c:out value="${row.SODIENTHOAI}" /></td>
            </tr>
        </c:forEach>
    </table>
</body>
</html>

```

Cập nhật dữ liệu dùng <sql:update></sql:update>

```

<sql:setDataSource var="snapshot" user="sa" password="1234567890"
    driver="com.microsoft.sqlserver.jdbc.SQLServerDriver"
    url="jdbc:sqlserver://localhost:1433;databaseName=QuanLyGiangVienDeTai" />

    <sql:update dataSource = "${snapshot}" var = "result">
        INSERT INTO GIANGVIEN VALUES ('GV104', 'Thanh Van', 'Lĩnh vực', '0011223344');
    </sql:update>

```

Xóa dữ liệu dùng <sql:update></sql:update>

```

<sql:setDataSource var="snapshot" user="sa" password="1234567890"
    driver="com.microsoft.sqlserver.jdbc.SQLServerDriver"

```

```

url="jdbc:sqlserver://localhost:1433;databaseName=QuanLyGiangVienDeTai" />
<c:set var = "gvId" value = "GV104"/>
<sql:update dataSource = "${snapshot}" var = "count">
    DELETE FROM GIANGVIEN WHERE MAGV = ?
    <sql:param value = "${gvId}" />
</sql:update>

```

Cập nhật dữ liệu dùng <sql:update></sql:update>

```

<sql:setDataSource var="snapshot" user="sa" password="1234567890"
    driver="com.microsoft.sqlserver.jdbc.SQLServerDriver"
    url="jdbc:sqlserver://localhost:1433;databaseName=QuanLyGiangVienDeTai" />
<c:set var = "gvId" value = "GV104"/>
<sql:update dataSource = "${snapshot}" var = "count">
    UPDATE GIANGVIEN SET TENGV = 'Van' WHERE MAGV=?;
    <sql:param value = "${gvId}" />
</sql:update>

```

## Bài 7. JSPs - Bài tập tổng hợp I

Xây dựng ứng dụng quản lý tin tức cho một website trực tuyến, các trang Web cần thực hiện các chức năng: *xem tin tức theo từng danh mục, thêm mới tin tức và xóa tin tức không cần thiết*. Một phần của cơ sở dữ liệu được mô tả như sau:

DANHMUC(MADM, TENDANHMUC, NGUOIQUANLY, GHICHU)

TINTUC(MATT, TIEUDE, NOIDUNGTT, LIENKET, MADM)

Xây dựng ứng dụng dùng Servlet/JSPs kết hợp với SQL Server và Web server Tomcat thực hiện các yêu cầu với mô hình MVC (Các Servlet đóng vai trò Controller):

- Tạo cơ sở dữ liệu với các quan hệ như mô tả trên và nhập dữ liệu cần thiết để kiểm tra chương trình. Cơ sở dữ liệu trong SQL Server lưu tên: QUANLYDANHMUC
- Dùng HTML/CSS tạo Layout chung cho các trang Web quản lý tin tức trực tuyến



- Tạo lớp mô tả thông tin *TinTuc.java* (bao gồm các get/set cho các thuộc tính và các constructors) dùng để thao tác với tin tức trực tuyến.
- Tạo lớp *DanhSachTinTucQuanLy.java* (bao gồm các phương thức thao tác với CSDL: lấy tin tức theo danh mục tin tức, thêm mới tin tức và xóa tin tức).
- Tạo giao diện Web cho phép thực hiện thao tác liệt kê dữ liệu danh sách tin tức theo từng danh mục.
- Tạo giao diện Web cho phép thực hiện giao diện thao tác với việc thêm dữ liệu
  - Tạo Form cho phép thêm nội dung tin tức với đầy đủ thông tin phù hợp với CSDL (*TinTucFormServlet.java*, *TinTucForm.jsp*)
  - Kiểm tra dữ liệu nhập phía Client trên Form:
    - Mã TT, Tiêu đề, Liên kết, Nội dung, Thông tin danh mục của tin là bắt buộc nhập.
    - Liên kết bắt đầu bởi “*http://*” (dùng RegularExpression).
    - Nội dung không quá 255 ký tự (dùng RegularExpression).

- Thông tin nhập vào hợp lệ sau khi nhấn nút “Thêm” sẽ được thêm vào cơ sở dữ liệu và hiển thị dữ liệu trên màn hình (*KetQua.jsp* hoặc trả về *DanhSachTinTuc.jsp*).
- Tạo giao diện Web cho phép thực hiện chức năng quản lý: thao tác hủy dữ liệu. Khi chọn chức năng này, hiển thị danh sách và kèm theo chức năng hủy tin tức ra khỏi cơ sở dữ liệu (*QuanLyFormServlet.java*, *QuanLyForm.jsp*).

### Bài 8. JSPs - Bài tập tổng hợp 2

Xây dựng ứng dụng quản lý thông tin sản phẩm cho một công ty mua bán điện thoại trực tuyến, các trang Web cần thực hiện các chức năng: *xem danh sách sản phẩm theo nhà cung cấp*, *thêm mới sản phẩm* và *xóa sản phẩm không cần thiết*. Một phần của cơ sở dữ liệu được mô tả như sau:

NHACUNGCAP(MANCC, TENNHACC, DIACHI, SODIENTHOAI)

DIENTHOAI(MADT, TENDT, NAMSANXUAT, CAUHINH, MANCC)

Xây dựng ứng dụng dùng Servlet/JSPs kết hợp với SQL Server và Web server Tomcat thực hiện các công việc sau:

- Tạo cơ sở dữ liệu với các quan hệ như mô tả trên và nhập dữ liệu cần thiết để kiểm tra chương trình. Cơ sở dữ liệu trong SQL Server lưu tên: QUANLYDIENTHOAI
- Dùng HTML/CSS tạo Layout chung cho chương trình tương tự như sau:



- Tạo lớp mô tả thông tin *DienThoai.java*, *NhaCungCap.java* (bao gồm các get/set cho các thuộc tính và các constructors) dùng để thao tác với thông tin sản phẩm điện thoại.
- Tạo lớp *DanhSachDienThoaiQuanLy.java* (bao gồm các phương thức thao tác với CSDL: lấy sản phẩm theo mã (hoặc tên) nhà cung cấp, thêm mới và xóa sản phẩm điện thoại).

- Tạo giao diện Web cho phép thực hiện thao tác liệt kê dữ liệu dùng JSPs: (*DanhSachDienThoaiNCCServlet.java*, *DanhSachDienThoaiNCC.jsp*) Liệt kê dữ liệu danh sách sản phẩm điện thoại theo từng nhà cung cấp.
- Tạo giao diện Web cho phép thực hiện giao diện thao tác với việc thêm dữ liệu
  - Tạo Form thêm thông tin điện thoại với đầy đủ thông tin phù hợp với CSDL (*DienThoaiFormServlet.java*, *DienThoaiForm.jsp*)
  - Kiểm tra dữ liệu nhập phía Client trên Form:
    - Mã ĐT, Tên điện thoại, Năm sản xuất, Thông tin cấu hình: các thông tin này bắt buộc nhập.
    - Năm sản xuất là số nguyên 4 chữ số (dùng RegularExpression).
    - Thông tin cấu hình không quá 255 ký tự (dùng RegularExpression).
- Thông tin nhập vào hợp lệ sau khi nhấn nút “Thêm” sẽ được thêm vào cơ sở dữ liệu và hiển thị dữ liệu trên màn hình (*KetQua.jsp* hoặc *DanhSachDienThoaiNCC.jsp*).
- Tạo giao diện Web cho phép thực hiện chức năng quản lý: thao tác hủy dữ liệu. Khi chọn chức năng này, hiển thị danh sách và kèm theo chức năng hủy thông tin điện thoại ra khỏi cơ sở dữ liệu (*QuanLyFormServlet.java*, *QuanLyForm.jsp*).

Lưu ý có thể dùng trường hợp dùng TagLib kết nối trực tiếp:

Trường hợp dùng <sql:setDataSource>, tạo file JSP dùng chung cho việc xử lý, tương tự như file **JSPDataController.jsp**

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
    <sql:setDataSource var = "snapshot"
        driver = "com.microsoft.sqlserver.jdbc.SQLServerDriver"
        url = "jdbc:sqlserver://localhost:1433;databaseName=QuanLyNCCDienThoai"
        user = "sa" password = "1234567890" />

    <sql:query var = "dsncc" dataSource = "${snapshot }">
        select * from NHACUNGCAP;
    </sql:query>

    <c:if test = "${pageContext.request.method=='POST' }">
        <c:if test="${param.action=='Them'}">
            <% int nam = Integer.parseInt(request.getParameter("txtNamSanXuat"));%>
        </c:if>
    </c:if>
</body>
</html>
```

```

<c:catch var = "exception">
    <sql:update var="result" dataSource="${snapshot }">
        insert into DIENTHOAI(MADT,TENDT,CAUHINH,MANCC,NAMSANXUAT) values
        (?, ?, ?, ?, <%=nam%>)
        <sql:param value = "${param.txtMaDT}"></sql:param>
        <sql:param value = "${param.txtTenDT}"></sql:param>
        <sql:param value = "${param.txtCauHinh}"></sql:param>
        <sql:param value = "${param.sltNCC}"></sql:param>
    </sql:update>
    <c:if test = "${result}>=1 ">
        <font color="green"> Thêm thành công!</font>
    </c:if>
</c:catch>
<c:if test = "${exception!=null }">
    <font color="red">Lỗi</font>
</c:if>
</c:if>
</c:if>
</body>
</html>

```

## File JSP hiển thị danh sách

```

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql "%>
<%@ include file="JSPDataController.jsp"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Danh sách điện thoại nhà cung cấp</title>
<style>
* {
    padding: 0;
    margin: 0;
}

header {
    background-image: url("Images/203Logo.png");
    height: auto;
    border: 1px solid black;
}

footer {
    height: 50px;
    text-align: center;
    border: 1px solid black;
}

#main {

```

```

        height: auto;
    }

#menu {
    height: 50px;
    border: 1px solid black;
    text-align: center;
}

ul {
    list-style: none;
}

```

</style>

</head>

<body>

- <header> </header>
- <div id="menu"> <a href="DanhSachDienThoaiNCC.jsp">Danh sách sản phẩm</a> | <a href="DienThoaiForm.jsp">Thêm mới sản phẩm</a> | <a href="TrangQuanLy.jsp">Chức năng quản lý</a>
- </div>
- <div id="main">
- <form method="POST">
- <h1>Danh sách điện thoại</h1>
- <div>
- <table border=1 style="border-collapse: collapse;">
- <tr>
- <th>Tên nhà cung cấp</th>
- <th>Mã sản phẩm:</th>
- <th>Tên sản phẩm:</th>
- <th>Năm sản xuất</th>
- <th style="width: 300px;">Cấu hình</th>
- </tr>
- <c:forEach var="ncc" items="\${dsncc.rows}">
- <tr>
- <td><c:out value="\${ncc.TENNHHACC}"></c:out></td>
- <sql:query var="dtncc" dataSource="\${snapshot}"> SELECT DIENTHOAI.MADT, DIENTHOAI.TENDT, DIENTHOAI.NAMSANXUAT, DIENTHOAI.CAUHINH, DIENTHOAI.MANCC, NHACUNGCAP.TENNHHACC FROM DIENTHOAI INNER JOIN NHACUNGCAP ON DIENTHOAI.MANCC = NHACUNGCAP.MANCC WHERE NHACUNGCAP.MANCC=?
- <sql:param value="\${ncc.MANCC}"></sql:param>
- </sql:query>
- <td>
- <ul>
- <c:forEach var="dt" items="\${dtncc.rows}">
- <li><c:out value="\${dt.MADT}"></c:out> <input type="hidden" value="\${dt.MADT}" name="LblMadt"></li>

```

                </c:forEach>
            </ul>
        </td>
        <td>
            <ul>
                <c:forEach var="dt"
items="${dtncc.rows }">
                    <li><c:out
value="${dt.TENDT}"></c:out></li>
                <c:forEach var="dt"
items="${dtncc.rows }">
                    <li><c:out
value="${dt.NAMSANXUAT}"></c:out></li>
                <c:forEach var="dt"
items="${dtncc.rows }">
                    <li><c:out
value="${dt.CAUHINH}"></c:out></li>
                <c:forEach var="ncc" items="${dsncc.rows }">
                    <option value="${ncc.MANCC}"><c:out
value="${ncc.TENNHAAC}></c:out></option>
                </c:forEach>
            </select>
        </div>
    </div>
    <footer> Họ tên SV - Lớp - Phòng máy </footer>
</body>
</html>

```

### Trường hợp theo yêu cầu tạo Servlet xử lý các yêu cầu:

Các lớp Entity: DienThoai.java, DienThoaiNCC.java và NhaCungCap.java để trong package: entities  
Lớp kết nối CSDL

```
package controller;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

import entities.DienThoai;
import entities.NhaCungCap;

public class DanhSachDienThoaiQuanLy {
    private Connection con = null;

    public ConnectToDB() {
        String user = "sa";
        String password = "1234567890";
        String url =
"jdbc:sqlserver://localhost:1433;databaseName=QuanLyNCCDienThoai";
        try {
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
            Con = DriverManager.getConnection(url, user, password);
        } catch (Exception e) {
            // TODO: handle exception
            e.printStackTrace();
        }
    }

    public List<DienThoai> laySPTheoMa(String maNCC){
        String sql = "select * from DIENTHOAI where MANCC=' "+maNCC+" '";
        List<DienThoai> ds = new ArrayList<>();
        try {
            Statement st = con.createStatement();
            ResultSet rs = st.executeQuery(sql);
            while(rs.next()) {
                String maDT = rs.getString(1);
                String tenDT = rs.getString(2);
                String cauHinh = rs.getString(4);
                int namSX = rs.getInt(3);

                ds.add(new DienThoai(maDT, tenDT, namSX, cauHinh, maNCC));
            }
        } catch (Exception e) {
            // TODO: handle exception
            e.printStackTrace();
        }
    }
}
```

```

        return ds;
    }
    public List<NhaCungCap> layMaNCC(){
        String sql = "select MANCC,TENNHCAC FROM NHACUNGCAP";
        List<NhaCungCap> ds = new ArrayList<>();
        try {
            Statement st = con.createStatement();
            ResultSet rs = st.executeQuery(sql);
            while(rs.next()) {
                String maNCC = rs.getString(1);
                String tenNCC = rs.getString(2);
                ds.add(new NhaCungCap(tenNCC, maNCC));
            }
        } catch (Exception e) {
            // TODO: handle exception
            e.printStackTrace();
        }
        return ds;
    }
    public void themDienThoai(DienThoai dt) {
        String sql = "insert into DIENTHOAI values(?,?,?,?,?,?)";
        try {
            PreparedStatement st = con.prepareStatement(sql);
            st.setString(1, dt.getMaDT());
            st.setString(2, dt.getTenDT());
            st.setInt(3, dt.getNamSX());
            st.setString(4, dt.getCauHinh());
            st.setString(5, dt.getMaNCC());
            st.execute();
        } catch (Exception e) {
            // TODO: handle exception
            e.printStackTrace();
        }
    }
    public void xoaDienThoai(String madt) {
        try {
            String sql = "delete from DIENTHOAI where MADT=?";
            PreparedStatement pre = con.prepareStatement(sql);
            pre.setString(1, madt);
            pre.execute();
        } catch (Exception e) {
            // TODO: handle exception
            e.printStackTrace();
        }
    }
}

```

Tập tin Servlet thực hiện các yêu cầu liệt kê danh sách, thêm, xóa dữ liệu.

```

package controller;

import java.io.IOException;

```

```

import java.util.ArrayList;
import java.util.List;
import java.util.Set;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import entities.DienThoai;
import entities.DienThoaiNCC;
import entities.NhaCungCap;

@WebServlet("/DanhSachDienThoaiServlet")
public class DanhSachDienThoaiServlet extends HttpServlet {

    DanhSachDienThoaiQuanLy ql = new DanhSachDienThoaiQuanLy ();

    public DanhSachDienThoaiServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
     *      response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // TODO Auto-generated method stub
        response.getWriter().append("Served at: ").append(request.getContextPath());
        String action = request.getParameter("action");
        if (action != null) {
            if (action.equalsIgnoreCase("Them")) {
                System.out.println(ql.layMaNCC());
                request.setAttribute("dsNCC", ql.layMaNCC());

                getServletContext().getRequestDispatcher("/DienThoaiForm.jsp").forward(request,
                response);
                return;
            } else if (action.equalsIgnoreCase("Luu")) {
                String maDT = request.getParameter("maDT");
                String tenDT = request.getParameter("tenDT");
                String cauHinh = request.getParameter("cauHinh");
                String maNCC = request.getParameter("maNCC");
                int namSX = Integer.parseInt(request.getParameter("namSX"));
                DienThoai dt = new DienThoai(maDT, tenDT, namSX, cauHinh, maNCC);
                System.out.println(dt);
                themMoi(dt);
                request.setAttribute("dt", dt);
            }
        }
    }
}

```

```

getServletContext().getRequestDispatcher("/KetQua.jsp").forward(request, response);
        return;
    }
}

List<DienThoaiNCC> ds = layThongTin();
request.setAttribute("ds", ds);

getServletContext().getRequestDispatcher("/DanhSachDienThoaiNCC.jsp").forward(request, response);
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
 *      response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    // TODO Auto-generated method stub
    doGet(request, response);
}

public List<DienThoai> laySPTheoMa(String maNCC) {
    List<DienThoai> ds = ql.laySPTheoMa(maNCC);
    return ds;
}

public List<DienThoaiNCC> layThongTin() {
    List<DienThoaiNCC> ds = new ArrayList<>();
    List<NhaCungCap> dsNCC = ql.layMaNCC();
    for (NhaCungCap nhaCungCap : dsNCC) {

        String maNCC = nhaCungCap.getMaNCC();
        String tenNCC = nhaCungCap.getTenNCC();

        List<DienThoai> dsDienThoai = laySPTheoMa(maNCC);
        ds.add(new DienThoaiNCC(maNCC, tenNCC, dsDienThoai));
    }
    return ds;
}

public void themMoi(DienThoai dt) {
    ql.themDienThoai(dt);
}

public void xoaSP() {
}
}

```

File JSP

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
```

```

pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Danh sách</title>
</head>
<body>
    <c:forEach items="${ds }" var="item">
        <form method="post" action="DanhSachDienThoaiServlet">
            <tr>
                <td><c:out value="${item.tenNCC }"></c:out></td>

                <td><i>Danh Sach Dien Thoai SX</i>
                    <ul>
                        <c:forEach items="${item.ds }" var="item2">
                            <li><c:out value="${item2.tenDT }"></c:out></li>
                        </c:forEach>
                    </ul>
                </td>
            </tr>
        </form>
    </c:forEach>
</body>
</html>

```

Tự thao tác với file KetQua.jsp và các file còn lại

# BÀI TẬP TUẦN 05-06-07 MÔN LẬP TRÌNH WEB NÂNG CAO (JAVA)

## Chương 4: JavaServer Faces (JSFs)

Mục tiêu:

- Hiểu được kiến trúc của JavaServer Faces - JSF Architecture
- Hiểu và làm việc được với faces-config.xml - Page Navigation
- Thực hiện được ứng dụng với JSF Tags
- Thực hiện được ứng dụng với JSF - Composite Components
- Thực hiện được ứng dụng với JSF – Ajax
- Thực hiện được ứng dụng với JSF Event Handling

Yêu cầu:

- Tắt cả các bài tập lưu trong thư mục: T:\MaSV\_HoTen\Tuan05-06-07\
- Tạo Project **MaSV\_HoTen\_Tuan05-06-07** trong thư mục T:\MaSV\_HoTen\Tuan05-06-07\ trong Eclipse Java EE (Java Platform Enterprise Edition). Mỗi bài tập có thể lưu trong từng package riêng biệt.
- Cuối mỗi buổi thực hành, SV phải nén (.rar hoặc .zip) thư mục làm bài và nộp lại bài tập đã thực hiện trong buổi đó.

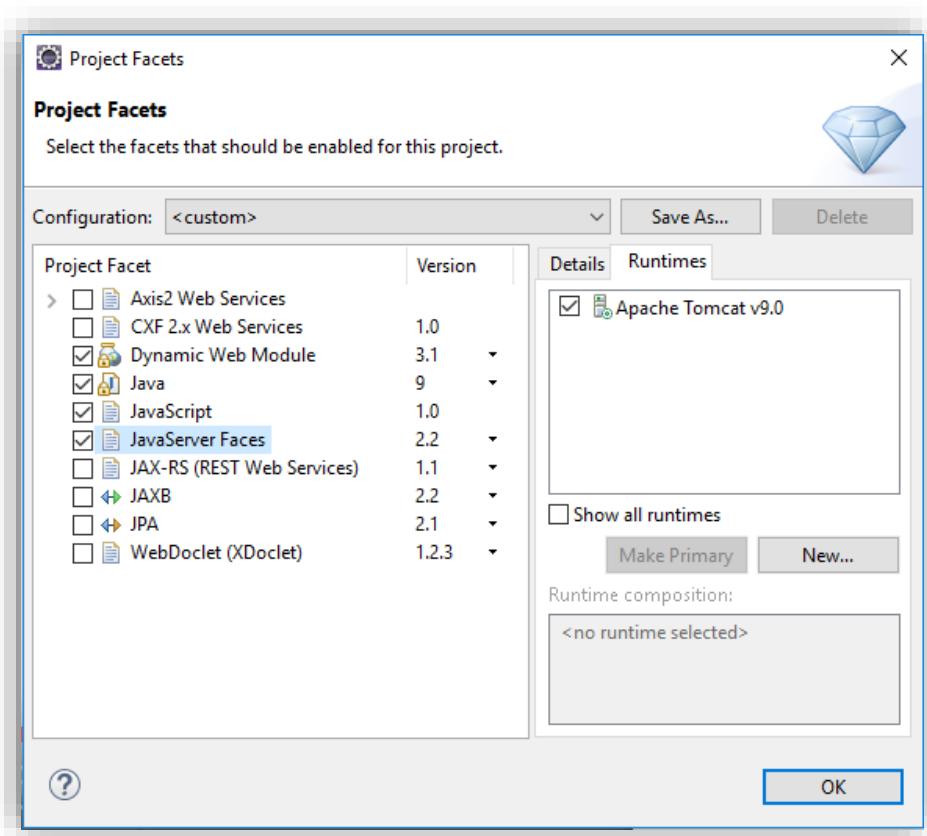
### Một số lưu ý trong JavaServer Faces

Yêu cầu cho ứng dụng sử dụng JavaServer Faces:

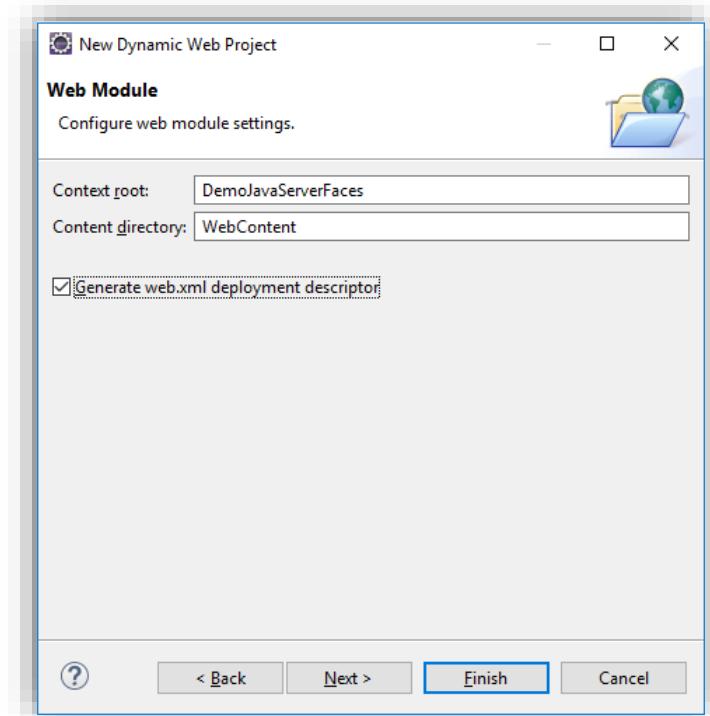
- JDK 8 hoặc 9
- Eclipse Oxygen (hoặc phiên bản khác)
- Tomcat 9 (hoặc phiên bản khác) được tích hợp chung với Eclipse
- Thư viện JSF 2.1, 2.2 (Mojarra 2.2.0), 2.3

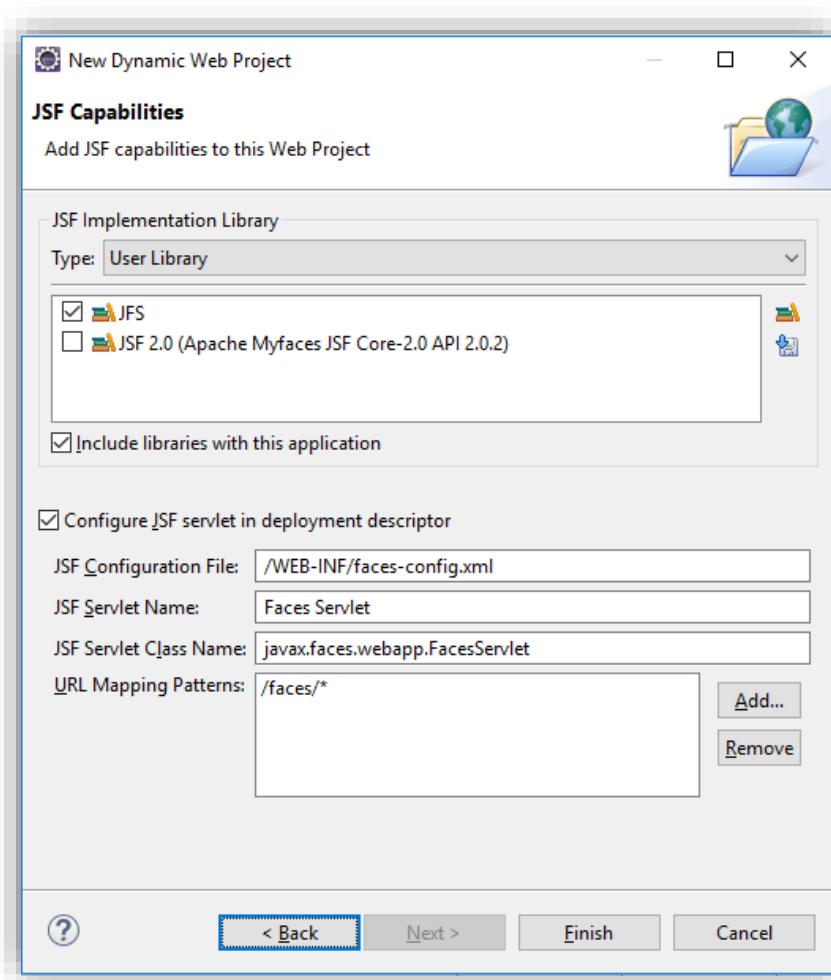
Tạo Project JavaServer Faces:

- Tạo Dynamic Web Project, thay đổi Configuration:



Lưu ý phát sinh web.xml cho Project đổi với Eclipse Oxygen

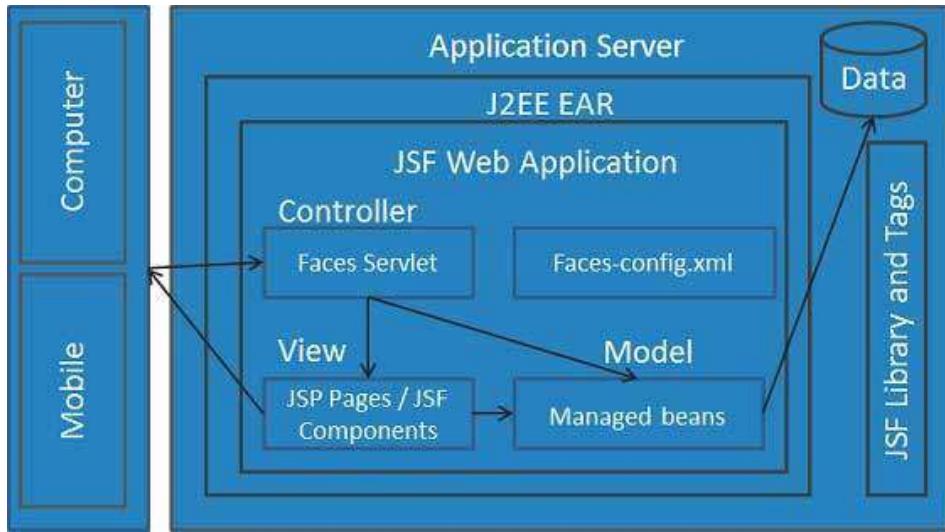




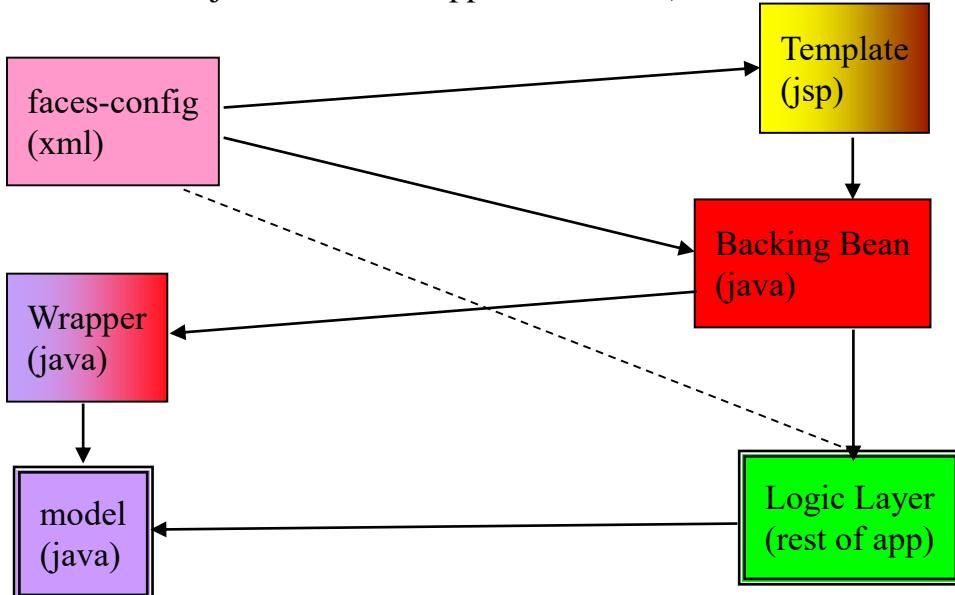
## 1. Tính năng của JSFs

- Component Based Framework
- Facelets Technology
- Expression Language
- HTML5
- Ease and Rapid Web Development
- Support Internationalization
- Exception Handling
- AJAX Support
- Security

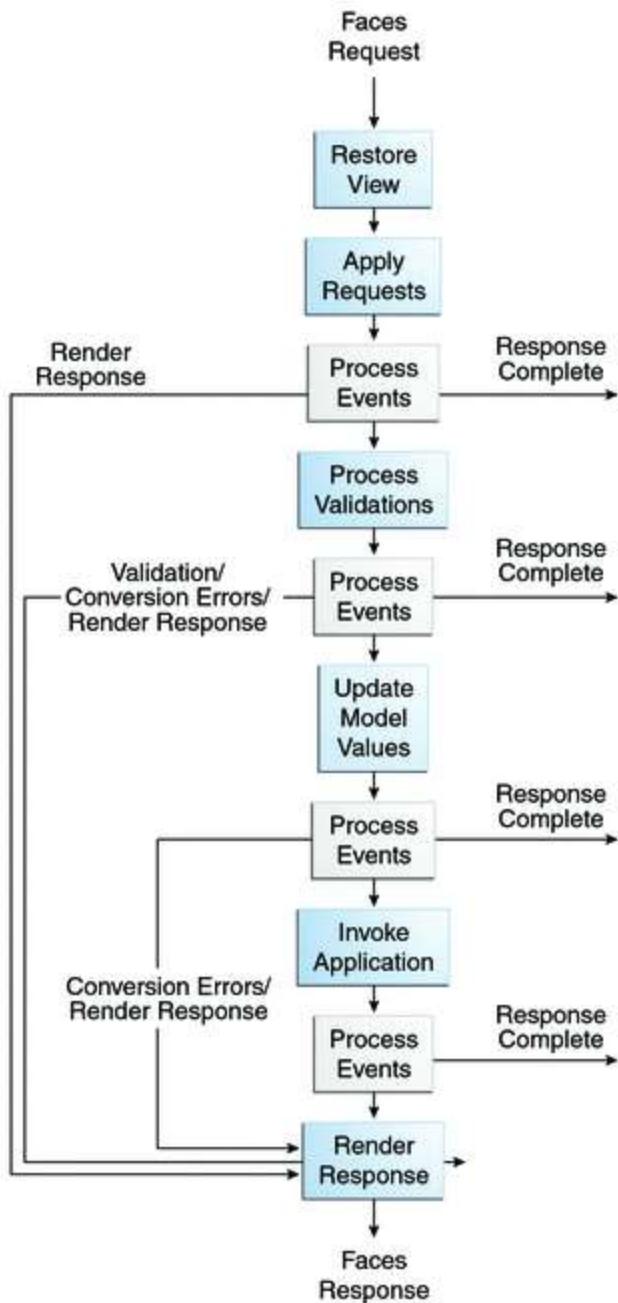
## 2. JSFs Framework



Faces Servlet (javax.faces.webapp.FacesServlet)



## 3. JSF Life Cycle



*Giai đoạn Restore View:* Khi có request từ client đến Web server, JSF thực hiện đầu tiên Restore View

- Thực hiện xây dựng các thành phần giao diện từ form yêu cầu của client
- Các UI Component được tổ chức theo dạng tree, và trạng thái của các control, trạng thái kích hoạt được thu thập, tổng hợp và đưa vào Faces Servlet

*Giai đoạn Apply Request Value:* Sau khi đã hoàn tất giai đoạn Restore View, JSF chuyển sang giai đoạn Apply Request Value

- Từ thông tin thu thập được, JSF chuyển qua ép kiểu tương ứng thành kiểu dữ liệu khai báo từ trong ManagedBean hay tương ứng với loại control của form.
- Nếu ép kiểu dữ liệu không thành công thì JSF thông báo lỗi và response cho Faces Context hay Faces Servlet để Faces Servlet thông báo cho client. Nếu thành công, chuyển sang giai đoạn kế tiếp.

**Giai đoạn Process Validation:** Dữ liệu tiếp tục được kiểm tra ràng buộc tương ứng với yêu cầu trong thành phần Validators, trong ManagedBean. Nếu dữ liệu không hợp lệ theo yêu cầu thì JSF thông báo lỗi và response cho Faces Context hay Faces Servlet để Faces Servlet thông báo cho client. Nếu đúng, dữ liệu này được chuyển vào ManagedBean tương ứng.

**Giai đoạn Update Model Value:** Cập nhật giá trị tương ứng vào các thuộc tính trong ManagedBeans

**Giai đoạn Invoke Application:** Khi cập nhật thành công, Faces Servlet hay Faces Context dựa trên event, thực hiện gọi (invoke) phương thức xử lý tương ứng (such as submitting a form/linking to another page), xác định trang JSF output.

**Giai đoạn Render Response:** Dữ liệu output khi xử lý xong sẽ được chuyển đổi tương ứng nạp vào form, cập nhật giá trị các control UI. Render để trình bày kết quả trên form và đưa output cho Faces Servlet hay Faces Context để response cho client kết quả thực hiện.

#### 4. ManagedBean

Managed Bean: là các Java Bean được đăng ký với JSFs. Managed Bean là Java Bean quản lý bởi JSF Framework.

Managed bean thường thực hiện các chức năng:

- Validate dữ liệu.
- Xử lý các sự kiện gửi tới.
- Thực thi tiến trình để xác định trang tiếp theo mà ứng dụng cần chuyển tới.

C1: Cấu hình Managed Bean bên trong file **faces-config.xml**

```
<managed-bean>
    <managed-bean-name>user</managed-bean-name>
    <managed-bean-class>User</managed-bean-class>
    <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```

C2: Dùng Annotation khi khai báo lớp ManagedBean.

```
@ManagedBean // Using ManagedBean annotation
```

```
@ApplicationScoped // Using Scope annotation
```

```
public class User {
    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

Các Scope của ManagedBean

- *Application* (dùng annotation kèm theo định nghĩa ManagedBean `@ApplicationScoped`): Bean tồn tại trong suốt thời gian hoạt động của ứng dụng web. Bean này được tạo khi có một HTTP request đầu tiên gửi tới bean (hoặc lúc ứng dụng trên Web Server được khởi tạo và thuộc tính `eager=true`) và chỉ bị mất đi khi ứng dụng Web shutdown.

- *Session* (dùng annotation kèm theo định nghĩa ManagedBean @SessionScoped): Bean tồn tại tương ứng với HTTP Session. Bean được tạo khi có một HTTP request đầu tiên gửi tới bean và bị mất đi khi HTTP Session bị hủy.
- *View* (dùng annotation kèm theo định nghĩa ManagedBean @ViewScoped): Bean tồn tại tương ứng với quá trình giao tiếp với cùng một View của JSF. Bean được tạo khi có một HTTP request được gửi và bị mất đi khi chuyển tới một View khác.
- *Request* (dùng annotation kèm theo định nghĩa ManagedBean @RequestScoped): Bean tồn tại tương ứng với chu trình HTTP request – response. Được tạo khi một request được gửi tới và mất đi khi có một response tương ứng trả về.
- *None* (dùng annotation kèm theo định nghĩa ManagedBean @NoneScoped): Biểu thị scope không được định nghĩa, dùng với Expression Language (EL). Được tạo và hủy ngay trước và sau khi sử dụng EL
- *Custom* (dùng annotation kèm theo định nghĩa ManagedBean @CustomScoped): Đây là không phải là một scope chuẩn. Được định nghĩa bởi người dùng.

Mặc định các ManagedBean là lazy (chỉ được tạo khi có một request gửi đến). Có thể tạo thể hiện và đặt bean trong application scope ngay khi ứng dụng bắt đầu chạy bằng khai báo @ManagedBean (eager=true)

```
@ManagedBean // Using ManagedBean annotation
@ApplicationScoped // Using Scope annotation
public class User {
    private String name;

    @ManagedProperty
    private Address add;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Address getAdd () {
        return name;
    }

    public void setAdd (Address a) {
        this.add = a;
    }
}
```

JSF là một Framework Dependency Injection (DI) đơn giản. Với việc sử dụng annotation @ManagedProperty, giá trị của managed bean có thể được inject vào bên trong một bean khác.

### Trường hợp sử dụng @ManagedProperty

Tạo ManagedBean dùng 2 thuộc tính là @ManagedProperty, giá trị 2 thuộc tính này inject thông qua tham số truyền vào

```
package jsf.managedproperty;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.ManagedProperty;
import javax.faces.bean.RequestScoped;

@ManagedBean
@RequestScoped
public class UserBean {
    @ManagedProperty(value = "#{param.userName}")
    private String userName;

    @ManagedProperty(value = "#{param.userAddress}")
    private String userAddress;

    public String submit() {
        return "result";
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getUserAddress() {
        return userAddress;
    }

    public void setUserAddress(String userAddress) {
        this.userAddress = userAddress;
    }
}
```

Tạo Template hiển thị + kết quả

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">
<h:body>
    <h:form>
        <h:commandButton value="Send" action="#{userBean.submit()}">
```

```

        <f:param id="userName" name="userName" value="vttvan"/>
        <f:param id="userAddress" name="userAddress" value="address"/>
    </h:commandButton>
</h:form>
</h:body>
</html>

```

```

<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
<h:body>
    <p>#{userBean.userName}</p>
    <p>#{userBean.userAddress}</p>
</h:body>
</html>

```

### 5. Dependency cho ứng dụng dùng JSFs (Project Maven)

```

<dependencies>
    <dependency>
        <groupId>com.sun.faces</groupId>
        <artifactId>jsf-api</artifactId>
        <version>2.2.9</version>
    </dependency>
    <dependency>
        <groupId>com.sun.faces</groupId>
        <artifactId>jsf-impl</artifactId>
        <version>2.2.9</version>
    </dependency>
</dependencies>

```

### 6. File cấu hình web.xml của Project JSFs

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns="http://xmlns.jcp.org/xml/ns/javaee"
          xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
          http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" id="WebApp_ID" version="3.1">
    <display-name>TestJSFsDemo</display-name>
    <welcome-file-list>
        <welcome-file>index.xhtml</welcome-file>
    </welcome-file-list>
    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>

```

```

<servlet-mapping>
    < servlet-name>Faces Servlet</servlet-name>
    < url-pattern>/faces/*</url-pattern>
</servlet-mapping>
<context-param>
    <description>State saving method: 'client' or 'server' (=default). See JSF Specification 2.5.2</description>
    <param-name>javax.faces.STATE_SAVING_METHOD</param-name>
    <param-value>client</param-value>
</context-param>
<context-param>
    <param-name>javax.servlet.jsp.jstl.fmt.localizationContext</param-name>
    <param-value>resources.application</param-value>
</context-param>
<listener>
    <listener-class>com.sun.faces.config.ConfigureListener</listener-class>
</listener>
</web-app>

```

## 7. Cấu hình file faces-config.xml

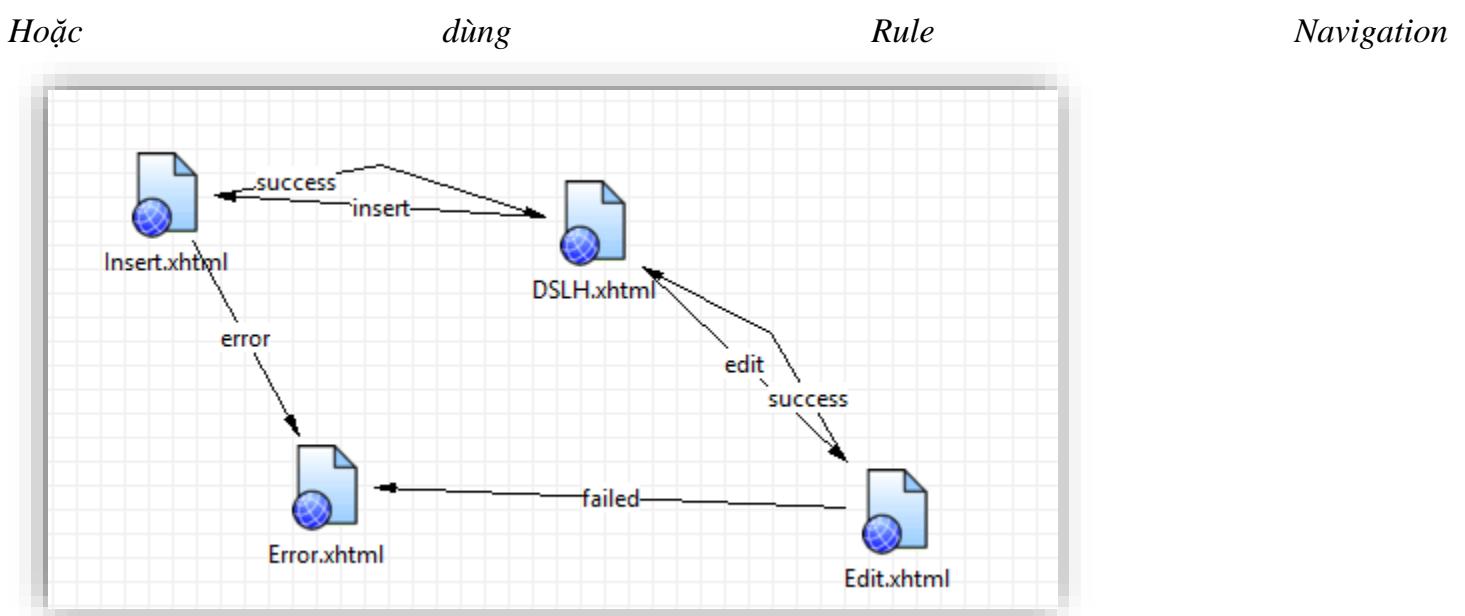
- Khai báo các ManagedBean
- Khai báo các Navigation Rules

```

<?xml version="1.0" encoding="UTF-8"?>
<faces-config xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-facesconfig_2_2.xsd" version="2.2">
    <managed-bean>
        <managed-bean-name>lophoc</managed-bean-name>
        <managed-bean-class>se.iuh.edu.vn.LopHocBean</managed-bean-class>
        <managed-bean-scope>session</managed-bean-scope>
    </managed-bean>
    <navigation-rule>
        <from-view-id>/DSLH.xhtml</from-view-id>
        <navigation-case>
            <from-outcome>insert</from-outcome>
            <to-view-id>/Insert.xhtml</to-view-id>
        </navigation-case>
        <navigation-case>
            <from-outcome>edit</from-outcome>
            <to-view-id>/Edit.xhtml</to-view-id>
        </navigation-case>
    </navigation-rule>
    <navigation-rule>
        <from-view-id>/Insert.xhtml</from-view-id>
        <navigation-case>
            <from-outcome>success</from-outcome>

```

```
        <to-view-id>/DSLH.xhtml</to-view-id>
    </navigation-case>
    <navigation-case>
        <from-outcome>error</from-outcome>
        <to-view-id>/Error.xhtml</to-view-id>
    </navigation-case>
</navigation-rule>
<navigation-rule>
    <from-view-id>/Edit.xhtml</from-view-id>
    <navigation-case>
        <from-outcome>success</from-outcome>
        <to-view-id>/DSLH.xhtml</to-view-id>
    </navigation-case>
    <navigation-case>
        <from-outcome>failed</from-outcome>
        <to-view-id>/Error.xhtml</to-view-id>
    </navigation-case>
</navigation-rule>
</faces-config>
```



#### *8. Các namespace và 1 số tags sử dụng cho Facelet*

## Namespace:

```
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
```

**h:inputText** Hiển thị HTML input type = “text”

**h:inputSecret** Hiển thị HTML input type = “password”

**h:inputTextarea** Hiển thị HTML textarea

**h:inputHidden** Hiển thi HTML input type = “hidden”

**h:selectBooleanCheckbox** Hiển thị HTML check box  
**h:selectManyCheckbox** Hiển thị một nhóm HTML check box  
**h:selectOneRadio** Hiển thị HTML radio button  
**h:selectOneListbox** Hiển thị một HTML danh sách item cho phép chọn 1.  
**h:selectManyListbox** Hiển thị một HTML danh sách item cho phép chọn nhiều giá trị.  
**h:selectOneMenu** Hiển thị HTML select box  
**h:outputText** Hiển thị text  
**h:outputFormat** Hiển thị text theo định dạng  
**h:graphicImage** Hiển thị ảnh.  
**h:outputStylesheet** Hiển thị CSS.  
**h:outputScript** Include script vào HTML  
**h:commandButton** Hiển thị submit button  
**h:link** Hiển thị link  
**h:commandLink** Hiển thị link.  
**h:outputLink** Hiển thị link.  
**h:panelGrid** Hiển thị layout dạng grid  
**h:message** Hiển thị message cho JSF UI component.  
**h:messages** Hiển thị tất cả message của các JSF UI component  
**f:param** Truyền tham số vào cho JSF UI component  
**f:attribute** Truyền thuộc tính vào cho JSF UI component  
**f:setPropertyActionListener** Thiết lập thuộc tính cho managed bean.

Thuộc tính chung các tags:

- **styleClass:** tương đương tham số class trong các thẻ HTML
- **required:** Đánh dấu trường này là bắt buộc.
- **converter:** Chuyển đổi dữ liệu hiển thị.

Ngoài ra tra cứu thêm

- <https://docs.oracle.com/javaee/7/tutorial/jsf-page003.htm>
- <https://docs.oracle.com/javaee/7/tutorial/jsf-page002.htm>
- <https://docs.oracle.com/javaee/7/jaserver-faces-2-2/vdldocs-facelets/f/tld-frame.html>

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
<h:head>
    <title>JSFs</title>
</h:head>
<h:body>
    <h:form>
        <table>
            <tr>
                <td>Name</td>
```

```

        <td>: <h:inputText value="#{userBean.name}" />
    </td>
</tr>
<tr>
    <td>Password</td>
    <td>: <h:inputSecret value="#{userBean.password}" />
    </td>
</tr>
<tr>
    <td>Gender</td>
    <td>: <h:selectOneRadio value="#{userBean.gender}">
        <f:selectItem itemValue="male" itemLabel="male"
/>
        <f:selectItem itemValue="female"
itemLabel="female" />
    </h:selectOneRadio>
    </td>
</tr>
<tr>
    <td>Favorites</td>
    <td>: <h:selectManyCheckbox value="#{userBean.favorites}">
        <f:selectItem itemValue="Travelling"
itemLabel="Travelling" />
        <f:selectItem itemValue="Design"
itemLabel="Design" />
        <f:selectItem itemValue="Coding"
itemLabel="Coding" />
        <f:selectItem itemValue="Music" itemLabel="Music"
/>
        <f:selectItem itemValue="Food" itemLabel="Food"
/>
        <f:selectItem itemValue="Sport" itemLabel="Sport"
/>
    </h:selectManyCheckbox></td>
</tr>
<tr>
    <td>Skills</td>
    <td>: <h:selectManyListbox value="#{userBean.skills}">
        <f:selectItem itemValue="Java" itemLabel="Java"
/>
        <f:selectItem itemValue="SQL" itemLabel="SQL" />
        <f:selectItem itemValue="AWS" itemLabel="AWS" />
        <f:selectItem itemValue="C#" itemLabel="C#" />
        <f:selectItem itemValue="Python"
itemLabel="Python" />
    </h:selectManyListbox></td>
</tr>
<tr>

```

```

        <td>Title</td>
        <td>: <h:selectOneMenu value="#{userBean.title}">
            <f:selectItem itemValue="Developer" itemLabel="Developer" />
            <f:selectItem itemValue="QA" itemLabel="QA" />
            <f:selectItem itemValue="Tester" itemLabel="Tester" />
        </h:selectOneMenu></td>
    </tr>
    <tr>
        <td colspan="2" action="#{userBean.submit()}" value="Submit" /></td>
    </tr>
</table>
</h:form>
</h:body>
</html>
```

## 9. Các JSFs tags hỗ trợ layout

Dùng namespace:

```
<html xmlns = "http://www.w3.org/1999/xhtml" xmlns:ui = "http://java.sun.com/jsf/facelets">
```

```
<?xml version = "1.0" encoding = "UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:h="http://java.sun.com/jsf/html"
xmlns:ui="http://java.sun.com/jsf/facelets">
```

```
<h:head></h:head>
<h:body>
    <div style="border: solid 2px">
        <ui:insert name="header">
            <ui:include src="header.xhtml" />
        </ui:insert>
    </div>
    <br />
```

```
<div style="border: solid 2px">
    <ui:insert name="content">
        <ui:include src="contents.xhtml" />
    </ui:insert>
</div>
<br />
```

```

<div style="border: solid 2px">
    <ui:insert name="footer">
        <ui:include src="footer.xhtml" />
    </ui:insert>
</div>

```

```

</h:body>
</html>

```

Tra cứu thêm: <https://docs.oracle.com/javaee/7/javaserver-faces-2-2/vdldocs-facelets/ui/tld-frame.html>

## *10. Tags hỗ trợ validation trong JSF Framework.*

Khai báo namespace:

```
<html xmlns = "http://www.w3.org/1999/xhtml" xmlns:f = "http://java.sun.com/jsf/core">
```

- **f:validateLength** (kiểm tra độ dài chuỗi)
- **f:validateLongRange** (giá trị số trong 1 vùng từ min → max)
- **f:validateDoubleRange** (giá trị số thực trong 1 vùng từ min → max)
- **f:validateRegex** (validate theo mẫu pattern)
- Có thể tự tạo các user-defined validator tag

```

<h:inputText id="input-name" value="#{userBean.name}" label="name" required="true"
requiredMessage="" validatorMessage="">
    <f:validateLength minimum="5" maximum="8" />
</h:inputText>
<h:message for="input-name" style="color: red"/>

<h:inputText id="input-age" value="#{userBean.age}" label="age" required="true">
    <f:validateLongRange minimum="1" maximum="50" />
</h:inputText>
<h:message for="input-age" style="color: red" /><br/>

<h:inputText id="input-rank" value="#{userBean.rank}" label="rank" required="true">
    <f:validateDoubleRange minimum="1.0" maximum="10.0" />
</h:inputText>
<h:message for="input-rank" style="color: red" /><br/>

<h:inputText id="input-email" value="#{helloBean.email}" label="email" required="true">
    <f:validateRegex
        pattern="^[_A-Za-z0-9-]+(\.[_A-Za-z0-9-]+)*@[A-Za-z0-9-]+(\.[A-Za-z]{2,})+$" />
</h:inputText>
<h:message for="input-email" style="color: red" /><br/>

```

Ngoài ra có thể dùng JSF Validate Bean (javax.validation Annotations)

Dùng dependency trong Maven Project:

```
<!-- java validation API and Reference Implementation -->
<dependency>
    <groupId>javax.validation</groupId>
    <artifactId>validation-api</artifactId>
    <version>1.1.0.Final</version>
</dependency>
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>5.1.3.Final</version>
</dependency>
```

```
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.validation.constraints.*;

@ManagedBean
@RequestScoped
public class UserBean {

    @Size(min = 3, message = "Name should at least be 3 characters long")
    private String name;

    @Pattern(regexp = "^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\\.[a-zA-Z0-9-.]+$", message =
    "This is not a valid email")
    private String email;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}
```

### Bài 1. JSFs - Các tags căn bản

JSF - Basic Tags, Facelets Tags, JSF - Convertor Tags, JSF - Validator Tags

Thực hiện form nhập dữ liệu và kiểm tra dữ liệu nhập là bắt buộc. Trong form tối thiểu bao gồm các thành phần:

- Text Boxes nhập liệu
- Checkboxes
- Dropdown List

Name :

Password :

Gender :

male  female

Favorites :

Travelling  Design  Coding  Music  Food  Sport

Skills :

Java  
SQL  
AWS  
C#  
Python

Title :

Submit

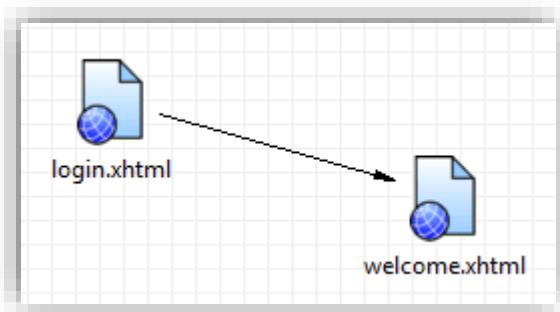
### Bài 2. JSFs - Kiểm tra dữ liệu nhập dùng Regular Expression

Kiểm tra dữ liệu nhập trên form dùng Regular Expression của JSF Validation Tags gồm:

- User name: 8 ký tự
- Password: 8-12 ký tự có số
- Email: [yourname@yourcompany.com](mailto:yourname@yourcompany.com)

### Bài 3. JSFs - Page Navigation

Thao tác với JSF - Page Navigation – faces-config.xml cho việc login vào ứng dụng Web



```

<?xml version="1.0" encoding="UTF-8"?>
<faces-config xmlns="http://xmlns.jcp.org/xml/ns/javaee">
  
```

```

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-facesconfig_2_2.xsd"
version="2.2">
<managed-bean>
    <managed-bean-name>loginBean</managed-bean-name>
    <managed-bean-class>se.iuh.edu.vn.loginBean</managed-bean-class>
    <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
<navigation-rule>
    <display-name>login.xhtml</display-name>
    <from-view-id>/login.xhtml</from-view-id>
    <navigation-case>
        <to-view-id>/welcome.xhtml</to-view-id>
        <redirect />
    </navigation-case>
</navigation-rule>

</faces-config>

```

Chỉnh sửa trong trường hợp lỗi, hiển thị error.xhtml

#### Bài 4. JSFs - Bài tập Quản lý thông tin lớp học

Quản lý thông tin lớp học. Cho phép hiển thị danh sách, cập nhật chi tiết từng lớp học, xóa thông tin lớp học. Thông tin lớp học gồm mã lớp học, tên lớp học và tên GV chủ nhiệm lớp đó.

## DANH SÁCH LỚP

Mã lớp	Tên lớp	GV chủ nhiệm	<a href="#">Insert New</a>
DHCNTT11A	Đại học Công nghệ thông tin 11A	Phạm Thái Khanh	<a href="#">Edit</a> - <a href="#">Delete</a>
DHHTTT11A	Đại học Hệ thống thông tin 11A	Trần Thị Kim Chi	<a href="#">Edit</a> - <a href="#">Delete</a>
DHKHMT11A	Đại học Khoa học máy tính 11A	Bùi Công Danh	<a href="#">Edit</a> - <a href="#">Delete</a>
DHKTPM11A	Đại học Kỹ thuật phần mềm 11A	Trần Thị Anh Thi	<a href="#">Edit</a> - <a href="#">Delete</a>

# Sửa thông tin lớp học

Mã lớp

DHKTPM11A

Tên lớp

Đại học Kỹ thuật phần mềm 11A

Tên Giảng viên chủ nhiệm

Trần Thị Anh Thi

Update

## Tham khảo hướng dẫn:

B1. Tạo CSDL Quản lý thông tin sinh viên, bảng LopHoc(MaLop, TenLop, TenGVCN)

B2. Tạo Project JavaServer Faces

B3. Tạo Managed Bean với scope là session, dùng faces-config.xml hoặc dùng Annotation

```
package se.iuh.edu.vn;
```

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import javax.faces.component.UIComponent;
import javax.faces.component.html.HtmlDataTable;
import javax.faces.event.ActionEvent;

public class LopHocBean {

    private String msLop;
    private String tenLop;
    private String tenGVCN;

    public LopHocBean() {
    }

    public LopHocBean(String msLop, String tenLop, String tenGVCN) {
        this.msLop = msLop;
        this.tenLop = tenLop;
        this.tenGVCN = tenGVCN;
    }

    public String getMsLop() {
        return msLop;
    }

    public void setMsLop(String msLop) {
        this.msLop = msLop;
    }

    public String getTenGVCN() {
```

```

        return tenGVCN;
    }

    public void setTenGVCN(String tenGVCN) {
        this.tenGVCN = tenGVCN;
    }

    public String getTenLop() {
        return tenLop;
    }

    public void setTenLop(String tenLop) {
        this.tenLop = tenLop;
    }

    public Connection getConnectionDB() {
        Connection c = null;
        try {
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
            c =
DriverManager.getConnection("jdbc:sqlserver://localhost:1433;databaseName=QLSV", "sa",
"1234567890");

        } catch (Exception e) {
            e.printStackTrace();
        }
        return c;
    }

    // Lấy ds lớp học
    public ArrayList<LopHocBean> getAllLophoc() {
        ArrayList<LopHocBean> lst = new ArrayList<LopHocBean>();
        try {
            Connection con = getConnectionDB();
            ResultSet rs = con.createStatement().executeQuery("select * from
LopHoc");
            while (rs.next()) {
                LopHocBean lh = new LopHocBean(rs.getString(1), rs.getString(2),
rs.getString(3));
                lst.add(lh);
            }
            con.close();
        } catch (Exception e) {
            e.printStackTrace();
        }

        return lst;
    }

    public String insertNew() {
        try {
            Connection con = getConnectionDB();

```

```

        String sql = "insert into LopHoc values(?, ?, ?)";
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setString(1, msLop);
        ps.setString(2, tenLop);
        ps.setString(3, tenGVCN);
        int x = ps.executeUpdate();
        if (x > 0) {
            return "success";
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return "failed";
}

public void deleteAction(ActionEvent evt) {

    HtmlDataTable table = getParentDatatable((UIComponent) evt.getSource()); // đổi tương xử lý sự kiện: bảng

    Object o = table.getRowData(); // Đổi tương dòng chọn.
    LopHocBean sellh = (LopHocBean) o;
    try {
        Connection con = getConnectionDB();
        String sql = "delete LopHoc where msLop=?";
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setString(1, sellh.msLop);
        ps.executeUpdate();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public void editLopHoc(ActionEvent evt) {

    HtmlDataTable table = getParentDatatable((UIComponent) evt.getSource());
    Object o = table.getRowData();
    LopHocBean lh = (LopHocBean) o;
    this.msLop = lh.msLop;
    this.tenLop = lh.tenLop;
    this.tenGVCN = lh.tenGVCN;
}

// Method to get the HtmlDataTable.
private HtmlDataTable getParentDatatable(UIComponent compo) {
    if (compo == null) {
        return null;
    }
    if (compo instanceof HtmlDataTable) {
        return (HtmlDataTable) compo;
    }
    return getParentDatatable(compo.getParent());
}

```

```

    }

    public String update() {
        try {
            Connection con = getConnectionDB();
            String sql = "update lophoc set tenLop=?, tenGVCN=? where msLop=?";
            PreparedStatement ps = con.prepareStatement(sql);
            ps.setString(1, tenLop);
            ps.setString(2, tenGVCN);
            ps.setString(3, msLop);
            int x = ps.executeUpdate();
            if (x > 0) {
                return "success";
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return "failed";
    }
}

```

Hoặc thêm trong file faces-config.xml

```

<managed-bean>
    <managed-bean-name>lophoc</managed-bean-name>
    <managed-bean-class>se.iuh.edu.vn.LopHocBean</managed-bean-class>
    <managed-bean-scope>session</managed-bean-scope>
</managed-bean>

```

Hoặc dùng annotation @ManagedBean

```

package se.iuh.edu.vn;

import ...;
import javax.faces.bean.ManagedBean;

@ManagedBean
public class LopHocBean { ...}

```

B4. Tạo giao diện liệt kê danh sách bằng 1 trong 2 cách B4.1 hoặc B4.2

B4.1. Dùng DSLH.jsp file với taglib

```

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="f" uri="http://java.sun.com/jsf/core"%>
<%@taglib prefix="h" uri="http://java.sun.com/jsf/html"%>

<f:view>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>JSP Page</title>
</head>
<body>
    <h1 align="center">DANH SÁCH LỚP</h1>
    <h:form id="frm">

```

```

<h: dataTable width="100%" border="1" var="Lh" id="table"
    value="#{lophoc.allLophoc}">
    <h:column id="ms" headerClass="headertext">
        <f:facet name="header">
            <h:outputText style="headerText" value="Mã Lớp" />
        </f:facet>
        <h:outputText id="mL" value="#{lh.msLop}" />
    </h:column>
    <h:column id="ten" headerClass="headertext">
        <f:facet name="header">
            <h:outputText style="headerText" value="Tên Lớp" />
        </f:facet>
        <h:outputText id="tL" value="#{lh.tenLop}" />
    </h:column>
    <h:column id="gv" headerClass="headertext">
        <f:facet name="header">
            <h:outputText style="headerText" value="GV chủ nhiệm" />
        </f:facet>
        <h:outputText id="gvcn" value="#{lh.tenGVCN}" />
    </h:column>
    <h:column id="actions" headerClass="headertext">
        <f:facet name="header">
            <h:commandLink value="Insert New" action="insert"
immediate="true" />
        </f:facet>
        <h:commandLink value="Edit" immediate="true"
            actionListener="#{lophoc.editLopHoc}" action="edit" /> -
        <h:commandLink value="Delete"
            actionListener="#{lophoc.deleteAction}" />
    </h:column>
</h: dataTable>
</h:form>
</body>
</html>
</f:view>

```

#### B4.2. Dùng DSLH.xhtml file

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:ui="http://xmlns.jcp.org/jsf/facellets"
    xmlns:h="http://xmlns.jcp.org/jsf/html"
    xmlns:f="http://xmlns.jcp.org/jsf/core">

<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>JSP Page</title>
</head>
<body>
    <h1 align="center">DANH SÁCH LỚP</h1>
    <h:form id="frm">
        <h: dataTable width="100%" border="1" var="Lh" id="table"

```

```

        value="#{Lophoc.allLophoc}">
        <h:column id="ms" headerClass="headertext">
            <f:facet name="header">
                <h:outputText style="headerText" value="Mã Lớp" />
            </f:facet>
            <h:outputText id="mL" value="#{Lh.msLop}" />
        </h:column>
        <h:column id="ten" headerClass="headertext">
            <f:facet name="header">
                <h:outputText style="headerText" value="Tên Lớp" />
            </f:facet>
            <h:outputText id="tL" value="#{Lh.tenLop}" />
        </h:column>
        <h:column id="gv" headerClass="headertext">
            <f:facet name="header">
                <h:outputText style="headerText" value="GV chủ nhiệm" />
            </f:facet>
            <h:outputText id="gvcn" value="#{Lh.tenGVCN}" />
        </h:column>
        <h:column id="actions" headerClass="headertext">
            <f:facet name="header">
                <h:commandLink value="Insert New" action="insert"
immediate="true" />
            </f:facet>
            <h:commandLink value="Edit" immediate="true"
actionListener="#{Lophoc.editLopHoc}" action="edit" /> -
            <h:commandLink value="Delete"
actionListener="#{Lophoc.deleteAction}" />
        </h:column>

    </h:dataTable>
</h:form>
</body>
</html>

```

B5. Tạo giao diện thao tác với việc thêm mới lớp học vào CSDL: Insert.jsp hoặc Insert.xhtml

```

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="f" uri="http://java.sun.com/jsf/core"%>
<%@taglib prefix="h" uri="http://java.sun.com/jsf/html"%>

<f:view>
    <html>
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
            <title>JSP Page</title>
        </head>
        <body>
            <h1 align="center"><h:outputText value="Thêm Lớp học mới"/></h1>
            <h:form>
                <h:panelGrid columns="2">

```

```

        <h:outputLabel for="ms" value="Mã số Lớp" />
        <h:inputText id="ms" value="#{lophoc.msLop}" size="40"
maxlength="256" />
        <h:outputLabel for="ht" value="Tên Lớp"/>
        <h:inputText id="ht" value="#{lophoc.tenLop}" size="60"
maxlength="256" />
        <h:outputLabel for="gv" value="Tên GVCN"/>
        <h:inputText id="gv" value="#{lophoc.tenGVCN}" size="60"
maxlength="256" />
    </h:panelGrid>
    <h:commandButton value="Insert" action="#{lophoc.insertNew}" />
</h:form>
</body>
</html>
</f:view>
```

#### B6. Tạo giao diện thao tác với việc sửa thông tin: Edit.jsp hoặc Edit.xhtml

```

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<%@taglib prefix="f" uri="http://java.sun.com/jsf/core"%>
<%@taglib prefix="h" uri="http://java.sun.com/jsf/html"%>

<f:view>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>JSP Page</title>
</head>
<body>
    <h1 align="center">
        <h:outputText value="Sửa thông tin Lớp học" />
    </h1>
    <h:form>
        <h:panelGrid columns="2">
            <h:outputLabel for="ms" value="Mã Lớp" />
            <h:inputText id="ms" value="#{lophoc.msLop}" readonly="true" />

            <h:outputLabel for="ht" value="Tên Lớp" />
            <h:inputText id="ht" value="#{lophoc.tenLop}" />

            <h:outputLabel for="gv" value="Tên GVCN" />
            <h:inputText id="gv" value="#{lophoc.tenGVCN}" />
        </h:panelGrid>
        <h:commandButton value="Update" action="#{lophoc.update}" />
    </h:form>
</body>
</html>
</f:view>
```

Trường hợp lỗi Error.jsp hoặc Error.xhtml:

```

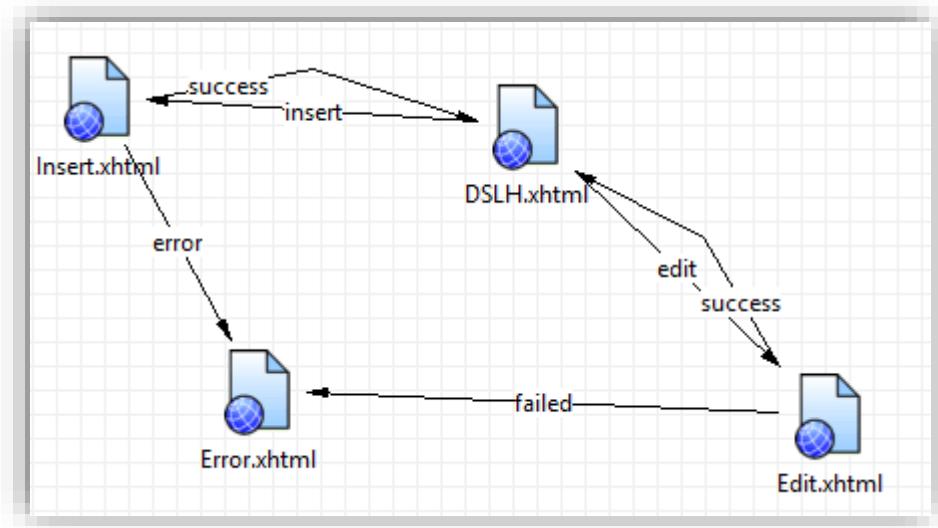
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="f" uri="http://java.sun.com/jsf/core"%>
<%@taglib prefix="h" uri="http://java.sun.com/jsf/html"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">

<f:view>
    <html>
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
            <title>JSP Page</title>
        </head>
        <body>
            <h1><h:outputText value="Error"/></h1>
        </body>
    </html>
</f:view>

```

Cấu hình faces-config.xml



Hoặc

```

<?xml version="1.0" encoding="UTF-8"?>
<faces-config xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/web-facesconfig_2_2.xsd"
    version="2.2">
    <managed-bean>
        <managed-bean-name>lophoc</managed-bean-name>
        <managed-bean-class>se.iuh.edu.vn.LopHocBean</managed-bean-class>
        <managed-bean-scope>session</managed-bean-scope>
    </managed-bean>
    <navigation-rule>

```

```

<from-view-id>/DSLH.xhtml</from-view-id>
<navigation-case>
    <from-outcome>insert</from-outcome>
    <to-view-id>/Insert.xhtml</to-view-id>
</navigation-case>
<navigation-case>
    <from-outcome>edit</from-outcome>
    <to-view-id>/Edit.xhtml</to-view-id>
</navigation-case>
</navigation-rule>
<navigation-rule>
    <from-view-id>/Insert.xhtml</from-view-id>
    <navigation-case>
        <from-outcome>success</from-outcome>
        <to-view-id>/DSLH.xhtml</to-view-id>
    </navigation-case>
    <navigation-case>
        <from-outcome>error</from-outcome>
        <to-view-id>/Error.xhtml</to-view-id>
    </navigation-case>
</navigation-rule>
<navigation-rule>
    <from-view-id>/Edit.xhtml</from-view-id>
    <navigation-case>
        <from-outcome>success</from-outcome>
        <to-view-id>/DSLH.xhtml</to-view-id>
    </navigation-case>
    <navigation-case>
        <from-outcome>failed</from-outcome>
        <to-view-id>/Error.xhtml</to-view-id>
    </navigation-case>
</navigation-rule>
</faces-config>

```

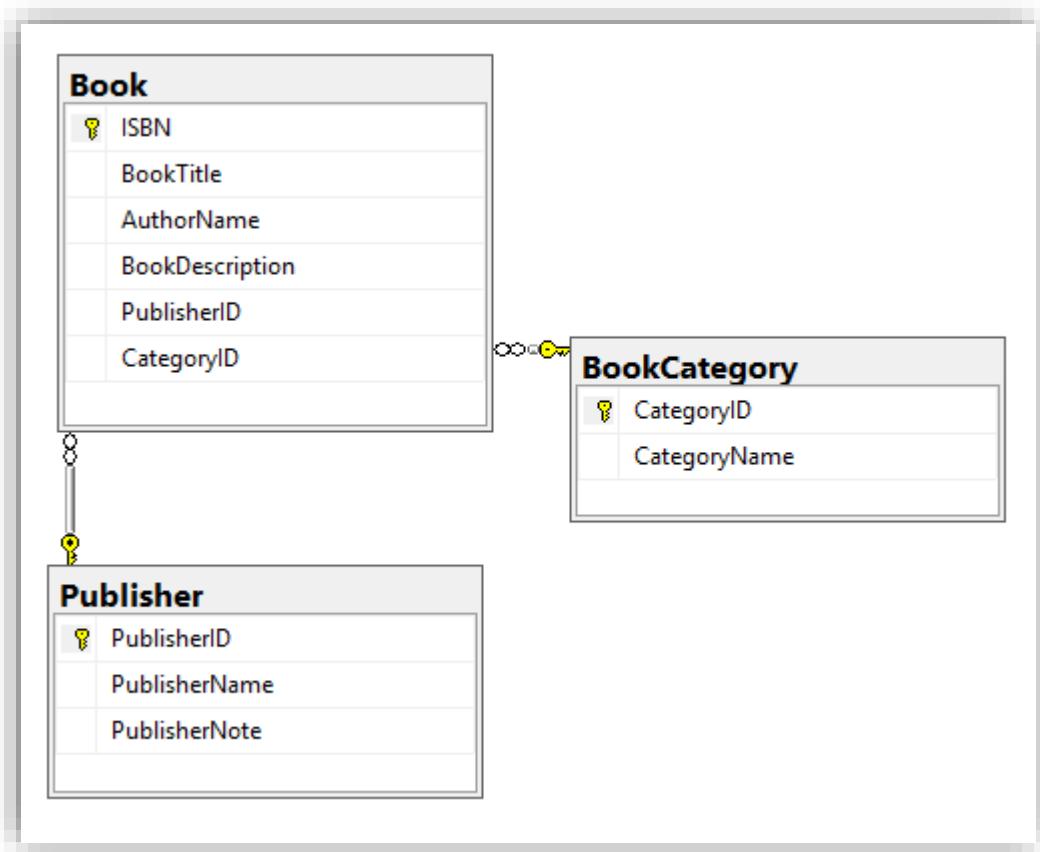
### Bài 5. JSFs - Thao tác với ComboBox

Dùng JSF thực hiện hiển thị dữ liệu trên ComboBox

Thực hiện giao diện liệt kê danh sách các sách theo từng nhà xuất bản.

Thực hiện giao diện liệt kê danh sách các sách theo từng loại.

Thực hiện form giao diện cho phép nhập thông tin sách, trong đó loại sách và danh sách nhà xuất bản được hiển thị theo ComboBox (dữ liệu này lấy từ CSDL)



```

CREATE TABLE Publisher(
    PublisherID int NOT NULL primary key,
    PublisherName nvarchar(150) NOT NULL,
    PublisherNote nvarchar(150)
)
go
CREATE TABLE BookCategory(
    CategoryID int NOT NULL primary key,
    CategoryName nvarchar(250) NOT NULL
)
go
CREATE TABLE Book(
    ISBN bigint NOT NULL primary key,
    BookTitle nvarchar(250) NOT NULL,
    AuthorName nvarchar(250) NOT NULL,
    BookDescription ntext NOT NULL,
    PublisherID int references Publisher(PublisherID) on delete cascade on update cascade,
    CategoryID int references BookCategory(CategoryID) on delete cascade on update cascade
)

```

## Bài 6. JSFs - Ajax căn bản

Thực hiện Ajax với JSF

<https://docs.oracle.com/javaee/7/javaserver-faces-2-2/vdldocs-facelets/toc.htm>

### Please enter the following information

Name	<input type="text"/>	Name cannot be empty.
User ID	<input type="text"/>	User ID cannot be empty.
Password	<input type="password"/>	Password must be >= 8 characters.

### Bài 7. JSFs - Bài tập Quản lý danh sách công việc

Tạo Todo list, cho phép thêm mới và xóa danh sách các công việc. Tách các lớp theo MVC, phần View có thể dùng .JSP hoặc .xhtml.

#### Todo list

Title

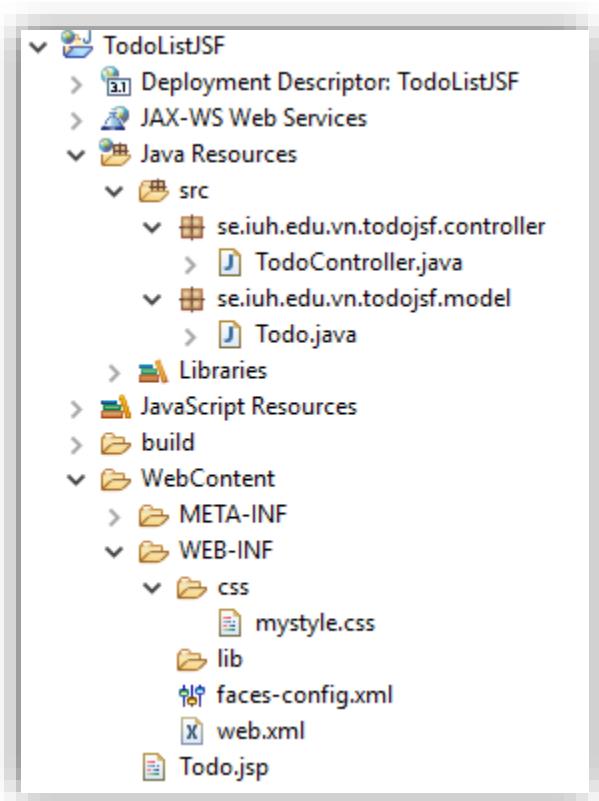
Description

Priority

[Hide Data Table](#) [Show Data Table](#)

Priority	Title	Description	Actions
1	Drinking coffee	At The Coffee House.	<a href="#">Delete</a>
3	Dinner with friends	At BongSen Restaurant.	<a href="#">Delete</a>
3	Learning Spring MVC Working with final Project	Working with final Project	<a href="#">Delete</a>
3	Learning Database	Oracle Database Foundations	<a href="#">Delete</a>

Project trong Eclipse



## Bài 8. JSFs - Thao tác với <h:dataTable> và <ui:repeat>

Dùng <h:dataTable> và <ui:repeat> hiển thị dữ liệu từ ManagedBean

<https://docs.oracle.com/javaee/7/javaserver-faces-2-2/vdldocs-facelets/h/dataTable.html>

<https://docs.oracle.com/javaee/7/javaserver-faces-2-2/vdldocs-facelets/ui/repeat.html>

Order No	Product Name	Price	Quantity	
A0001	Intel CPU	700000.00	3	<a href="#">Delete</a>
A0002	Harddisk 10TB	500000.00	2	<a href="#">Delete</a>
A0003	Dell Laptop	1160000.00	4	<a href="#">Delete</a>
A0004	Samsung LCD	520000.00	3	<a href="#">Delete</a>
A0005	A4Tech Mouse	120000.00	5	<a href="#">Delete</a>

B1. Cấu hình Dynamic Web Project với JSFs

B2. Tạo lớp Order và OrderBean (ManagedBean) hỗ trợ các phương thức trả về danh sách, xóa phần tử của các Order

Lớp Order:

```
package se.iuh.edu.vn.jsf.htable;
import java.math.BigDecimal;
public class Order{

    String orderNo;
    String productName;
    BigDecimal price;
    int qty;

    public Order(String orderNo, String productName,
                BigDecimal price, int qty) {
        this.orderNo = orderNo;
        this.productName = productName;
        this.price = price;
        this.qty = qty;
    }

    public String getOrderNo() {
        return orderNo;
    }

    public void setOrderNo(String orderNo) {
        this.orderNo = orderNo;
    }
}
```

```

    }

    public String getProductName() {
        return productName;
    }

    public void setProductName(String productName) {
        this.productName = productName;
    }

    public BigDecimal getPrice() {
        return price;
    }

    public void setPrice(BigDecimal price) {
        this.price = price;
    }

    public int getQty() {
        return qty;
    }

    public void setQty(int qty) {
        this.qty = qty;
    }

}

```

### Lớp OrderBean

```

package se.iuh.edu.vn.jsf.htable;

import java.io.Serializable;
import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.Arrays;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;

@ManagedBean
@SessionScoped
public class OrderBean implements Serializable {

    private static final long serialVersionUID = 1L;

```

```

private static final ArrayList<Order> orderList = new
ArrayList<Order>(Arrays.asList(
    new Order("A0001", "Intel CPU", new BigDecimal("700000.00"), 3),
    new Order("A0002", "Harddisk 10TB", new BigDecimal("500000.00")),
2),
    new Order("A0003", "Dell Laptop", new BigDecimal("1160000.00")),
4),
    new Order("A0004", "Samsung LCD", new BigDecimal("520000.00"), 3),
    new Order("A0005", "A4Tech Mouse", new BigDecimal("120000.00")),
5)));
}

public ArrayList<Order> getOrderList() { // hiển thị danh sách
    return orderList;
}

public String deleteAction(Order order) { // xóa phần tử
    orderList.remove(order);
    return null;
}

public String addAction(Order order) {
    orderList.add(order);
    return null;
}
}

```

B3. Tạo Template hiển thị dùng <h:tableData>. Tạo thư mục *resource/css* trong WebContent

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:ui="http://java.sun.com/jsf/facelets">
<h:head>
    <h:outputStylesheet library="css" name="table-style.css" />
</h:head>
<h:body>
    <h1>Dùng tag h:Table</h1>
    <h: dataTable value="#{orderBean.orderList}" var="o"
                  styleClass="order-table" headerClass="order-table-header"
                  rowClasses="order-table-odd-row,order-table-even-row">

        <h:column>
            <f:facet name="header">Order No</f:facet>
            #{o.orderNo}
    
```

```

</h:column>
    <h:column>
        <f:facet name="header">Product Name</f:facet>
        #{o.productName}
    </h:column>
    <h:column>
        <f:facet name="header">Price</f:facet>
        #{o.price}
    </h:column>
    <h:column>
        <f:facet name="header">Quantity</f:facet>
        #{o.qty}
    </h:column>
    <h:column>
        <f:facet name="header">Delete</f:facet>
        <h:form>
            <h:commandLink id="remove" value="Delete"
                action="#{orderBean.deleteAction(o)}"
                onclick="return confirm('Are you sure?')">
            </h:commandLink>
        </h:form>
    </h:column>
</h:dataTable>
</h:body>
</html>

```

Tạo Template hiển thị dùng <ui:repeat>

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:c="http://java.sun.com/jsp/jstl/core">
<h:head>
    <h:outputStylesheet library="css" name="table-style.css" />
</h:head>
<h:body>
    <h1>Dùng tag ui:Repeat</h1>
    <table class="order-table">
        <tr>
            <th class="order-table-header">Order No</th>
            <th class="order-table-header">Product Name</th>
            <th class="order-table-header">Price</th>
            <th class="order-table-header">Quantity</th>

```

```

        <th class="order-table-header"></th>
    </tr>
    <tbody>
        <ui:repeat var="o" value="#{orderBean.orderList}"
varStatus="status">
            <h:panelGroup rendered="#{status.even}">
                <tr>
                    <td class="order-table-even-row">#{o.orderNo}</td>
                    <td class="order-table-even-row">#{o.productName}</td>
                    <td class="order-table-even-row">#{o.price}</td>
                    <td class="order-table-even-row">#{o.qty}</td>
                    <td class="order-table-even-row"><h:form>
                        <h:commandLink id="remove"
value="Delete"
action="#{orderBean.deleteAction(o)}"
onclick="return confirm('Are you
sure?')">
                            </h:commandLink>
                        </h:form></td>
                </tr>
            </h:panelGroup>
            <h:panelGroup rendered="#{status.odd}">
                <tr>
                    <td class="order-table-odd-row">#{o.orderNo}</td>
                    <td class="order-table-odd-row">#{o.productName}</td>
                    <td class="order-table-odd-row">#{o.price}</td>
                    <td class="order-table-odd-row">#{o.qty}</td>
                    <td class="order-table-odd-row"><h:form>
                        <h:commandLink id="remove"
value="Delete"
action="#{orderBean.deleteAction(o)}"
onclick="return confirm('Are you
sure?')">
                            </h:commandLink>
                        </h:form></td>
                </tr>
            </h:panelGroup>
        </ui:repeat>
    </tbody>
</table>
</h:body>
</html>

```

## Bài 9. JSFs - Ajax

Thực hiện Todo List Management dùng JSFs kết hợp Ajax

**Todo List Management**

To do Exercise 1-10 on page 2	<a href="#">Edit</a> <a href="#">Delete</a>
Working with Assignment documentation	<a href="#">Edit</a> <a href="#">Delete</a>
Watching a movie	<a href="#">Edit</a> <a href="#">Delete</a>
Listen to country music	<a href="#">Edit</a> <a href="#">Delete</a>

**Add todo to table**

Todo:  [Add record](#)

**Todo List Management**

To do Exercise 1-10 on p	<a href="#">Save</a> <a href="#">Cancel</a> <a href="#">Delete</a>
Working with Assignment documentation	<a href="#">Edit</a> <a href="#">Delete</a>
Watching a movie	<a href="#">Edit</a> <a href="#">Delete</a>
Listen to country music	<a href="#">Edit</a> <a href="#">Delete</a>

**Add todo to table**

Todo:  [Add record](#)

**AJAX** (Asynchronous JavaScript and XML): là một kỹ thuật sử dụng JavaScript để gửi dữ liệu tới Server và nhận dữ liệu về một cách không đồng bộ. Khi sử dụng Ajax, JavaScript code sẽ trao đổi dữ liệu với Server và cập nhật vào một phần của trang Web mà không cần phải load lại nguyên trang Web.

JSFs cung cấp sự hỗ trợ cho Ajax. Tag `<f:ajax ..>` để xử lý gọi Ajax. Các attributes:

- **disabled**: nếu `disabled=true` Ajax không có tác dụng
- **event**: sự kiện sẽ thực thi gửi Ajax request: ‘click’, ‘change’, ‘blur’...
- **execute**: list các id của các component sẽ bao gồm trong Ajax request
- **immediate**: nếu `immediate=true`, sự kiện Ajax sẽ được gọi trong *apply request values phase*
- **listener**: khai báo method ở bean được gọi để xử lý Ajax request
- **onerror**: tên của function JavaScript được thực thi nếu xảy ra lỗi khi gọi Ajax
- **onevent**: tên của function JavaScript được thực thi để xử lý các sự kiện UI
- **render**: danh sách id của các component sẽ được cập nhật sau khi gọi Ajax

Tham khảo thêm phần JSFs Ajax: <https://docs.oracle.com/javaee/7/tutorial/jsf-ajax.htm>

- B1. Tạo Dynamic Web Project với JSFs  
 B2. Tạo các lớp Todo và ManagedBean TodoBean

```
package se.iuh.jsfs.ajax;

public class Todo {
    private String description;
    private transient boolean editable; // editable được xác định not
    serialization

    public Todo(String description) {
        this.description = description;
        this.editable = false;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public boolean isEditable() {
        return editable;
    }

    public void setEditable(boolean editable) {
        this.editable = editable;
    }
}
```

```
package se.iuh.jsfs.ajax;

import javax.annotation.PostConstruct;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

@ManagedBean
@SessionScoped
public class TodoBean implements Serializable {
```

```

private List<Todo> todoList;
private String description;

@PostConstruct
public void init() { // lấy từ collection hoặc CSDL
    todoList = new ArrayList<Todo>();
    todoList.add(new Todo("To do Exercise 1-10 on page 2"));
    todoList.add(new Todo("Working with Assignment documentation"));
    todoList.add(new Todo("Watching a movie"));
    todoList.add(new Todo("Listen to country music"));
}

public List<Todo> getTodoList() {
    return todoList;
}

public void setTodoList(List<Todo> todoList) {
    this.todoList = todoList;
}

public String getDescription() {
    return description;
}

public void setDescription(String description) {
    this.description = description;
}

// Thao tác với todoList
public void add(){ // thêm
    todoList.add(new Todo(description));
}

public void save(Todo todo){
    todoList.set(todoList.indexOf(todo), todo);
    cancelEdit(todo);
}

public void edit(Todo todo){
    for (Todo existing : getTodoList()){
        existing.setEditable(false);
    }
    todo.setEditable(true);
}

public void cancelEdit(Todo todo){
    todo.setEditable(false);
}

```

```

    }

    public void remove(Todo todo){ // xóa
        todoList.remove(todo);
    }

}

```

### B3. Tạo View kèm theo tag <f:ajax>

```

<%@xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core">
<h:head>

</h:head>
<h:body>
    <h:form id="todoForm">
        <h:dataTable id="todoTable" value="#{todoBean.todoList}" var="todo">
            <h:column>
                <f:facet name="header">
                    <h:outputText value="Todo List Management" />
                </f:facet>
                <h:inputText value="#{todo.description}">
                    <f:ajax render="todoForm:todoTable" />
                <h:outputText value="#{todo.description}">
                    rendered="#{not todo.editable}" />
                    <f:ajax render="todoForm:todoTable" />
                <h:commandLink value="Edit" action="#{todoBean.edit(todo)}">
                    rendered="#{not todo.editable}" />
                    <f:ajax render="todoForm:todoTable" />
                </h:commandLink>
                <h:commandLink value="Save" action="#{todoBean.save(todo)}">
                    rendered="#{todo.editable}" />
                    <f:ajax execute="@form" render="todoForm:todoTable" />
                </h:commandLink>
            </h:column>
            <h:column>
                <h:commandLink value="Cancel" />
                <f:ajax render="todoForm:todoTable" />
            </h:column>
        </h:dataTable>
    </h:form>
</h:body>

```

```

        </h:column>
        <h:column>
            <h:commandLink id="remove" value="Delete"
                action="#{todoBean.remove(todo)}"
                onclick="return confirm('Are you sure?')"
                <f:ajax render="todoForm:todoTable" />
            </h:commandLink>
        </h:column>
    </h:dataTable>

    <h4>Add todo to table</h4>
    <h:outputLabel value="Todo: ">
        <h:inputText id="todo" value="#{todoBean.description}" />
    </h:outputLabel>

    <h:commandButton value="Add record" action="#{todoBean.add}">
        <f:ajax execute="todoForm" render="todoForm:todoTable" />
    </h:commandButton>
</h:form>
</h:body>
</html>

```

### Bài 10. JSFs - Bài tập tổng hợp

Xây dựng ứng dụng quản lý thông tin đồng hồ cho cửa hàng bán đồng hồ khuyến mãi online, các trang Web cần thực hiện các chức năng: *xem danh sách đồng hồ, thêm mới đồng hồ và xóa đồng hồ không còn bán khuyến mãi*. Một phần của cơ sở dữ liệu được mô tả như sau:

SaleWatch(WatchID, Model, RetailPrice, SalePrice, WatchDescription, ImageURL)

Xây dựng ứng dụng dùng JavaServer Faces Framework kết hợp với SQL Server và Web server Tomcat thực hiện các công việc sau:

- Dùng HTML/CSS tạo Layout chung cho chương trình tương tự (layout này dùng chung cho tất cả các giao diện còn lại):

# Mother's Day SALE

Extended Returns Through June 13th


 "IN STOCK" ITEMS  
 SHIP IN 24 HRS

## CONNECT WITH US

 Join our mailing list today  
 and be the first to hear all  
 about our latest updates,  
 offers and promotions!

Email address

[Your privacy is respected](#)

-  Like us on Facebook
-  Follow on Instagram
-  Add us on LinkedIn
-  Follow us on Tumblr
-  Check Out AW Blog
-  Add us on Google+
-  Subscribe to YouTube
-  Follow us on Twitter


**Rolex Lady-Datejust 26  
179174**

 Retail Price: \$9,200.00  
**\$7,775.00**

**Rolex Lady-Datejust 26  
179173**

 Retail Price: \$10,650.00  
**\$8,375.00**

**Rolex Submariner Date  
116610LV**

 Retail Price: \$10,950.00  
**\$10,950.00**

**Rolex Lady-Datejust 26  
179160**

 Retail Price: \$6,100.00  
**\$5,075.00**

**Rolex Submariner Date  
116610**

 Retail Price: \$9,175.00  
**\$9,175.00**

**Rolex Yacht-Master II 116680**

 Retail Price: \$18,750.00  
**\$15,475.00**

- Viết lớp kết nối CSDL và lớp thực hiện thao tác với việc truy xuất cơ sở dữ liệu (bao gồm các phương thức: liệt kê danh sách đồng hồ khuyến mãi, thêm mới, xóa thông tin đồng hồ).
- Tạo giao diện Web cho phép thực hiện thao tác liệt kê dữ liệu dùng các tags của JSFs: (*DanhSach.xhtml*) liệt kê dữ liệu danh sách các đồng hồ khuyến mãi online từ cơ sở dữ liệu.
- Dùng các Facelet tạo giao diện Web cho phép thực hiện giao diện thao tác với việc thêm dữ liệu (*ThemMoiForm.xhtml* – nhấn vào link “Add new”). Dữ liệu phải được kiểm tra dùng JSFs Validation trước khi lưu vào cơ sở dữ liệu:
  - Tất cả dữ liệu *WatchID*, *Model*, *RetailPrice*, *SalePrice*, *WatchDescription*, *ImageURL* đều bắt buộc.
  - *RetailPrice*, *SalePrice* là số thực

- Tên đồng hồ (*Model*): Các ký tự đầu của mỗi từ là ký tự hoa, không có số hay ký tự đặc biệt trong tên.

Sau khi thực hiện thêm thành công, hiển thị danh sách có đồng hồ mới.

- Tạo giao diện Web cho phép thực hiện chức năng quản lý (*QuanLyUI.xhtml – nhấn vào link “Manage watches”*): thao tác hủy dữ liệu. Trước khi hủy hỏi người dùng có muốn hủy không, nếu muốn hủy dùng Ajax thực hiện chức năng này.

### Bài 11. JSFs - Bài tập tổng hợp

Xây dựng ứng dụng quản lý thông tin sách cho website [readanybook.com](http://readanybook.com), các trang Web cần thực hiện các chức năng: *xem danh sách các quyển sách, thêm mới sách và xóa sách không cần thiết*. Một phần của cơ sở dữ liệu được mô tả như sau:

BookOnline(BookID, BookName, Author, PublishYear, Rating, ImageURL)

Xây dựng ứng dụng dùng JavaServer Faces Framework kết hợp với SQL Server và Web server Tomcat thực hiện các công việc sau:

- Tạo cơ sở dữ liệu với các quan hệ như mô tả trên và nhập dữ liệu cần thiết để kiểm tra chương trình. Cơ sở dữ liệu trong SQL Server lưu tên: HOTEN\_MASOSINHVIEN
- Dùng HTML/CSS tạo Layout chung cho chương trình tương tự:

The screenshot shows a web application interface for managing books. At the top, there is a logo for "read ANY book . com". Below the logo, there is a navigation bar with icons for books and a search bar. On the right side of the header, there are links for "Manage books" and "Add new book". The main content area displays a list of books with their details and ratings. Each book entry includes a thumbnail image, the book's title, the author's name, the publish year, the rating (e.g., 9.78 / 10), and a small yellow progress bar icon. The books listed are: "The Silmarillion" by J.R.R. Tolkien (rating 9.78 / 10), "Disobedience" by Naomi Alderman (rating 7.45 / 10), "A Wrinkle in Time" by Madeleine L'engle (rating 8.36 / 10), "The Leisure Seeker" by Michael Zadoorian (rating 6.77 / 10), "Ready Player One" by Ernest Cline (rating 8.02 / 10), and "Arielle Immortal Awakening" by Lilian Roberts (rating 8.36 / 10). On the far right, there is a sidebar titled "Links" containing the "Manage books" and "Add new book" links, along with a placeholder text "@Copyright StudentName - ID".

Book Name	Author	Publish Year	Rating
The Silmarillion	J.R.R. Tolkien	9.78 / 10	
Disobedience	Naomi Alderman	7.45 / 10	
A Wrinkle in Time	Madeleine L'engle	8.36 / 10	
The Leisure Seeker	Michael Zadoorian	6.77 / 10	
Ready Player One	Ernest Cline	8.02 / 10	
Arielle Immortal Awakening	Lilian Roberts	8.36 / 10	

- Viết lớp kết nối CSDL và lớp thực hiện thao tác với việc truy xuất cơ sở dữ liệu (bao gồm các phương thức: liệt kê danh sách sách, thêm mới, xóa thông tin sách).
- Tạo giao diện Web cho phép thực hiện thao tác liệt kê dữ liệu dùng các tags của JSFs: (DanhSach.xhtml) liệt kê dữ liệu danh sách các quyền sách online từ cơ sở dữ liệu.
- Dùng các Facelet tạo giao diện Web cho phép thực hiện giao diện thao tác với việc thêm dữ liệu (*ThemMoiForm.xhtml* – nhấn vào link “Add new book”). Dữ liệu phải được kiểm tra dùng JSFs Validation trước khi lưu vào cơ sở dữ liệu:
  - Tất cả dữ liệu *BookID*, *BookName*, *Author*, *PublishYear*, *Rating*, *ImageURL* đều bắt buộc.
  - Năm xuất bản (*PublishYear*): từ 1900 – 2018
  - Tên sách (*BookName*): Các ký tự đầu của mỗi từ là ký tự hoa, không có số hay ký tự đặc biệt trong tên.

Sau khi thực hiện thêm thành công, hiển thị danh sách mới.

- Tạo giao diện Web cho phép thực hiện chức năng quản lý (*QuanLyUI.xhtml* – nhấn vào link “Manage books”): thao tác hủy dữ liệu. Trước khi hủy hỏi người dùng có muốn hủy không, nếu muốn hủy dùng Ajax thực hiện chức năng này.

### Bài 12. JSFs - Bài tập tổng hợp

Xây dựng ứng dụng quản lý thông tin xe website [reversidecarstore.net](http://reversidecarstore.net), các trang Web cần thực hiện các chức năng: *xem danh sách các xe đang được bán trực tuyến, thêm mới xe và xóa xe đã hết trong kho*. Một phần của cơ sở dữ liệu được mô tả như sau:

RiversideCar(CarID, CarName, Price, ModelYear, CarDescription, ImageURL)

Xây dựng ứng dụng dùng JavaServer Faces Framework kết hợp với SQL Server và Web server Tomcat thực hiện các công việc sau:

- Tạo cơ sở dữ liệu với các quan hệ như mô tả trên và nhập dữ liệu cần thiết để kiểm tra chương trình. Cơ sở dữ liệu trong SQL Server lưu tên: HOTEN\_MASOSINHVIEN
- Dùng HTML/CSS tạo Layout chung cho chương trình tương tự:

The screenshot shows a website for "RIVERSIDE CAR STORE". The header features the store's name in large, bold, red and white letters. Below the header, there's a sidebar with links for "Manage Cars" and "Add new Car". A copyright notice at the bottom left reads "@Copyright 2018 StudentID\_StudentName". The main content area displays six car listings in a 2x3 grid. Each listing includes a small image of the car, its model and year, contact information, price, and a "View Details" button.

Car Model	Year	Price
2015 Dodge Challenger Wide Body	2015	\$34,999
2014 Scion tC	2014	\$13,999
2013 Dodge Challenger	2013	\$18,999
2011 Kia Forte Koup	2011	\$7,999
2010 Infiniti G37 Sedan	2010	\$9,999
2010 Scion xB Release series	2010	\$7,999

- Viết lớp kết nối CSDL và lớp thực hiện thao tác với việc truy xuất cơ sở dữ liệu (bao gồm các phương thức: liệt kê danh sách xe, thêm mới xe, xóa thông tin xe).
- Tạo giao diện Web cho phép thực hiện thao tác liệt kê dữ liệu dùng các tags của JSFs: (*DanhSach.xhtml*) liệt kê dữ liệu danh sách thông tin các xe lấy từ cơ sở dữ liệu.
- Dùng các Facelet tạo giao diện Web cho phép thực hiện giao diện thao tác với việc thêm dữ liệu (*ThemMoiForm.xhtml* – nhấn vào link “*Add new Car*”). Dữ liệu phải được kiểm tra dùng JSFs Validation trước khi lưu vào cơ sở dữ liệu:
  - Tất cả dữ liệu *CarID*, *CarName*, *Price*, *ModelYear*, *CarDescription*, *ImageURL* đều bắt buộc.
  - Năm sản xuất (*ModelYear*): từ 1900 – 2018
  - Tên xe (*CarName*): Các ký tự đầu của mỗi từ là ký tự hoa, không có số hay ký tự đặc biệt trong tên.
  - Giá (*Price*): số thực.
- Sau khi thực hiện thêm thành công, hiển thị danh sách mới.
- Tạo giao diện Web cho phép thực hiện chức năng quản lý (*QuanLyUI.xhtml* – nhấn vào link “*Manage Cars* ”): thao tác hủy dữ liệu. Trước khi hủy hỏi người dùng có muốn hủy không, nếu muốn hủy dùng Ajax thực hiện chức năng này.

# BÀI TẬP TUẦN 08-09-10 MÔN LẬP TRÌNH WEB NÂNG CAO (JAVA)

## Chương 5: Spring MVC

Mục tiêu:

- Hiểu và hiện thực được Spring IoC và DI
- Hiểu và hiện thực được Spring Bean
- Hiểu và hiện thực được Spring AOP Transaction
- Hiểu và hiện thực được Handful of Services
- Hiểu và hiện thực được Spring Web MVC

Yêu cầu:

- Tất cả các bài tập lưu trong thư mục: T:\MaSV\_HoTen\Tuan08-09-10\
- Tạo Project **MaSV\_HoTen\_Tuan08-09-10** trong thư mục T:\MaSV\_HoTen\Tuan08-09-10\ trong Eclipse Java EE (Java Platform Enterprise Edition). Mỗi bài tập có thể lưu trong từng package riêng biệt.
- Cuối mỗi buổi thực hành, SV phải nén (.rar hoặc .zip) thư mục làm bài và nộp lại bài tập đã thực hiện trong buổi đó.

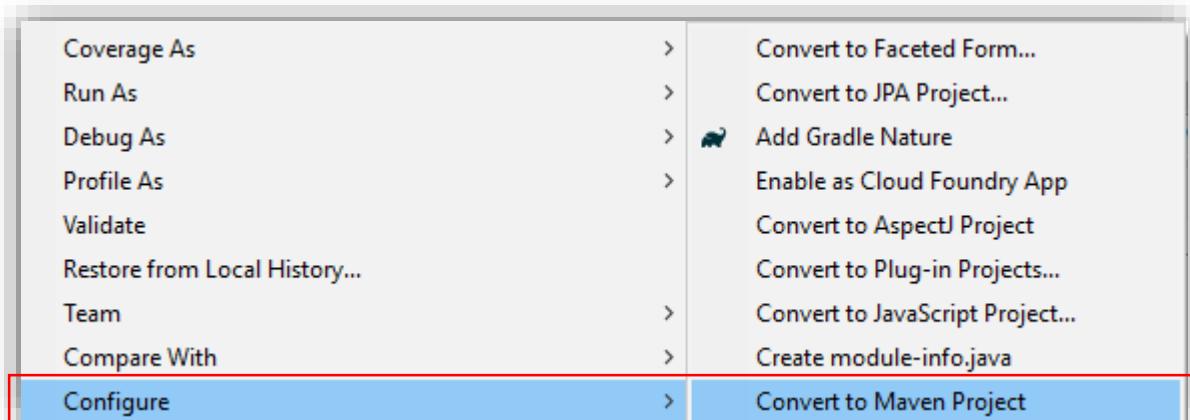
## Phần 1. Spring Core

Lưu ý:

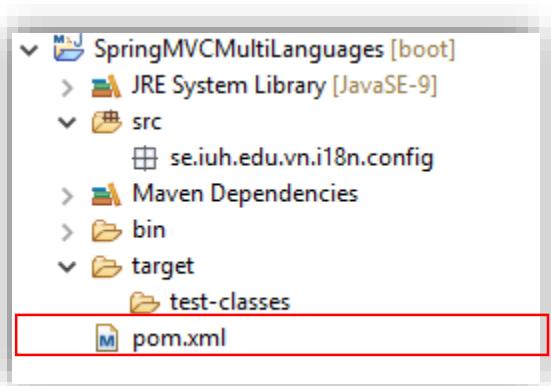
Sử dụng Maven

- Phiên bản Eclipse Oxygen đã có sẵn Maven <http://maven.apache.org/>
- Các dependencies tra cứu trong <https://mvnrepository.com/>

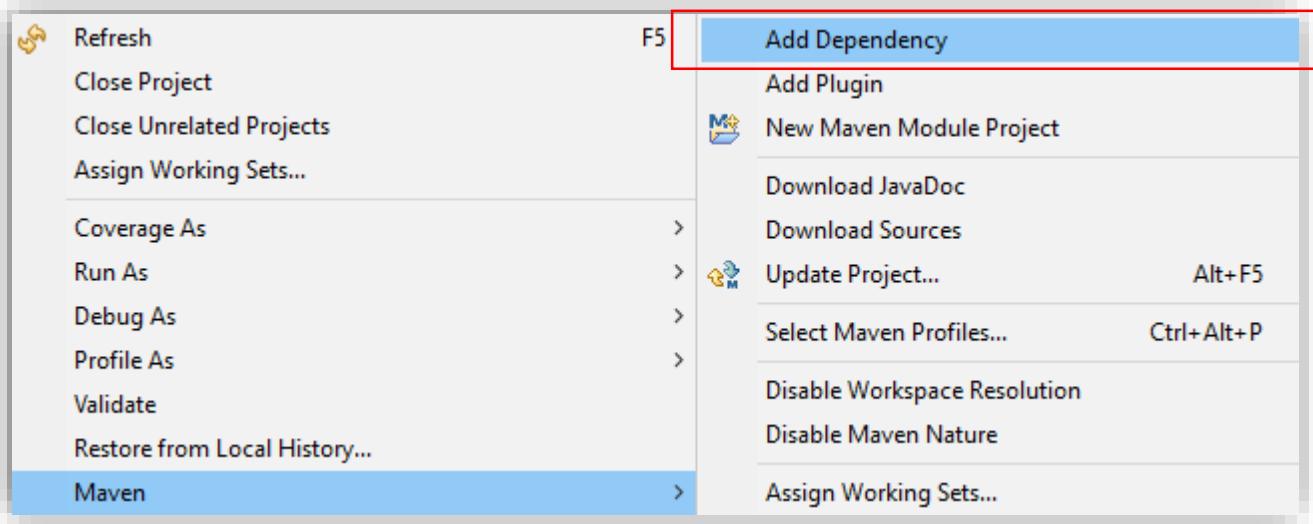
Project của Eclipse nhấn chuột phải → Chọn Configure → Chọn Convert to Maven Project



Các thay đổi về dependencies sẽ thay đổi trong file pom.xml



Thao tác Add Dependency bằng tool.



Hoặc có thể thay đổi trực tiếp trong file pom.xml.

Các dependencies tra cứu trong <https://mvnrepository.com/> , khi tra cứu lưu ý các groupId, artifactId, version cần dùng.

```

<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>SpringMVCMultiLanguages</groupId>
  <artifactId>SpringMVCMultiLanguages</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <build>
    <sourceDirectory>src</sourceDirectory>
    <plugins>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.7.0</version>
        <configuration>
          <release>9</release>
        </configuration>
      </plugin>
    </plugins>
  </build>

```

```

        </plugin>
    </plugins>
</build>
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>3.8.1</version>
        <scope>test</scope>
    </dependency>

    <!-- Spring Core -->
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-core</artifactId>
        <version>5.0.2.RELEASE</version>
    </dependency>

    <!-- Spring Context -->
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
    >
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.0.2.RELEASE</version>
    </dependency>

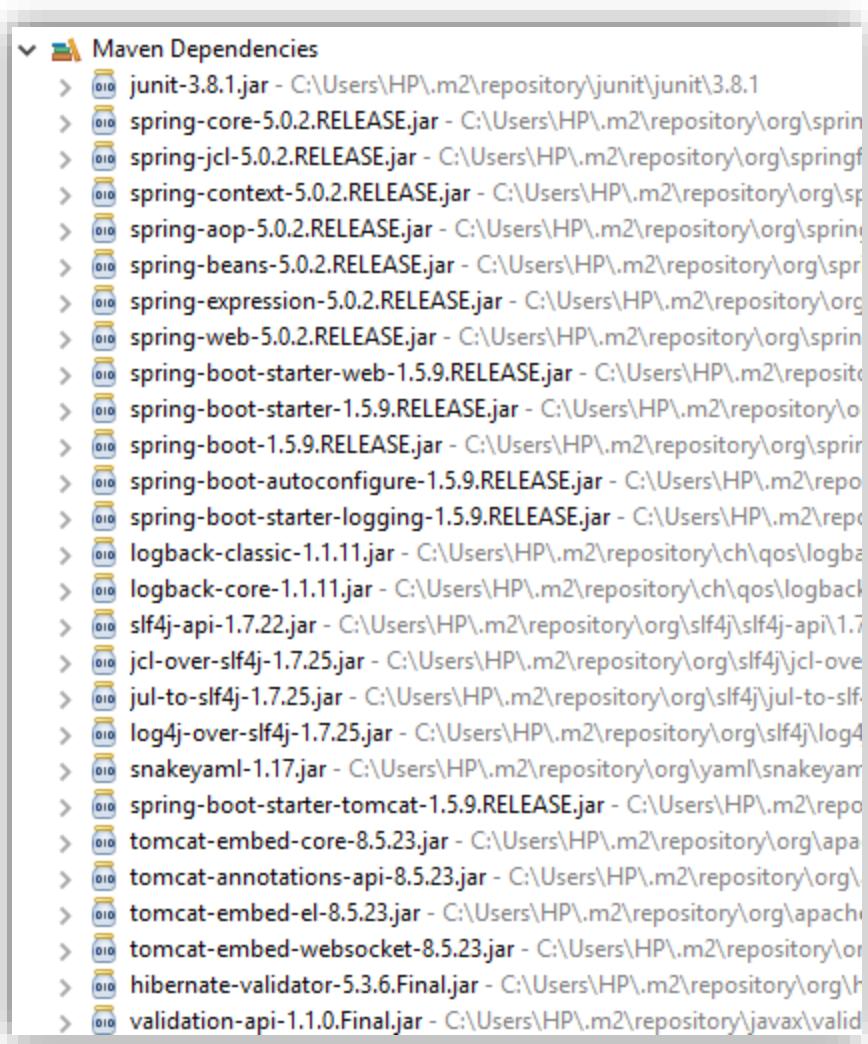
    <!-- Spring Web -->
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-web -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-web</artifactId>
        <version>5.0.2.RELEASE</version>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-web -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
        <version>1.5.9.RELEASE</version>
    </dependency>

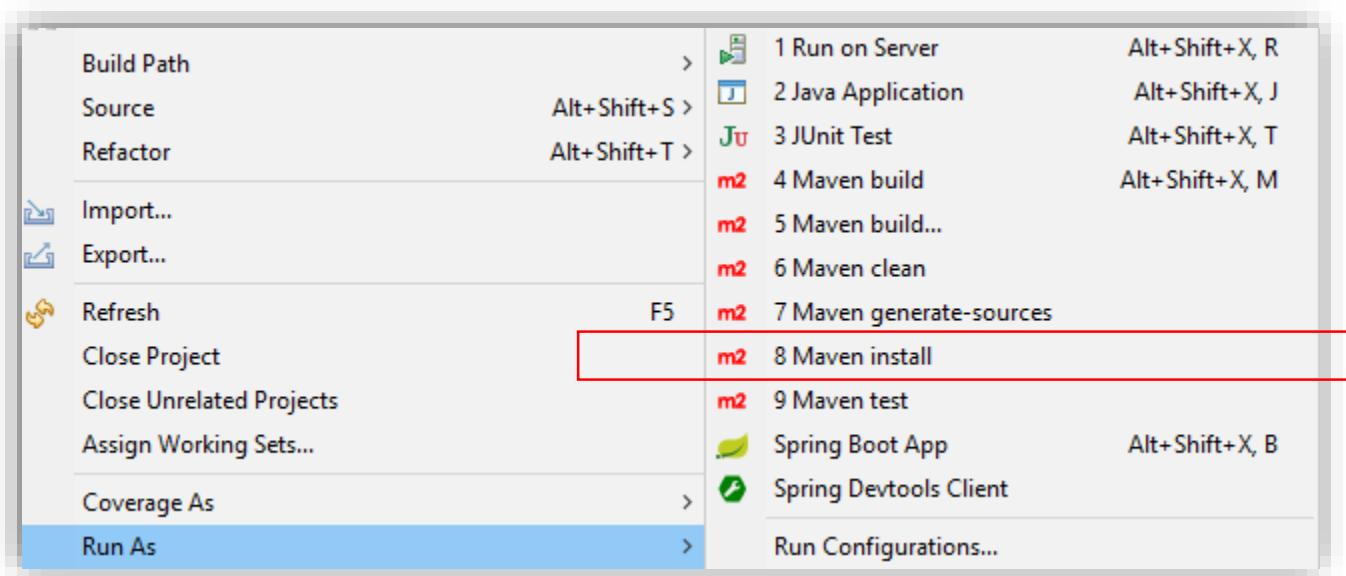
    </dependencies>
</project>

```

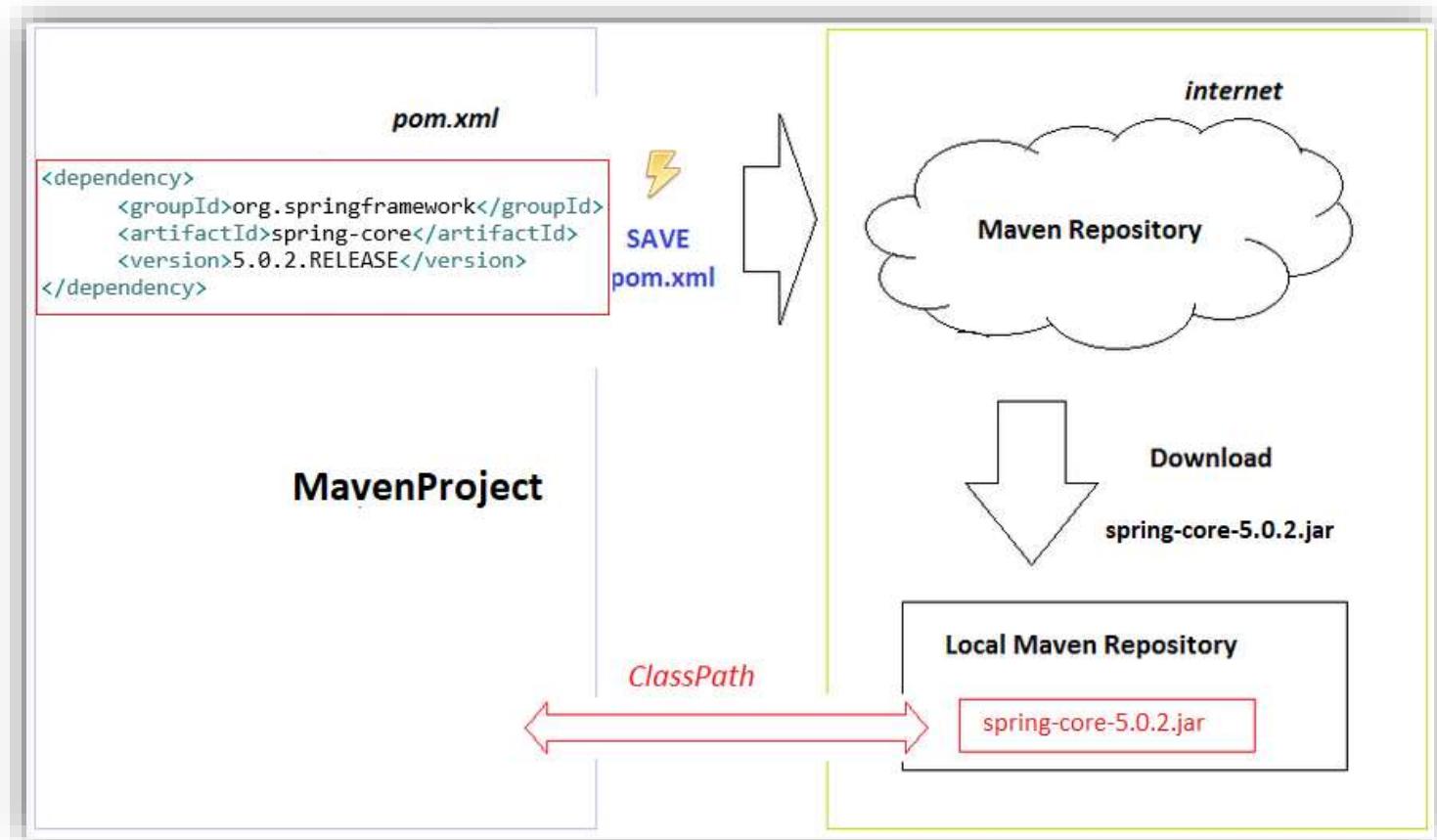
Các Maven dependencies



Tạo file jar từ Maven: **SpringMVCMultiLanguages0.0.1-SNAPSHOT.jar**

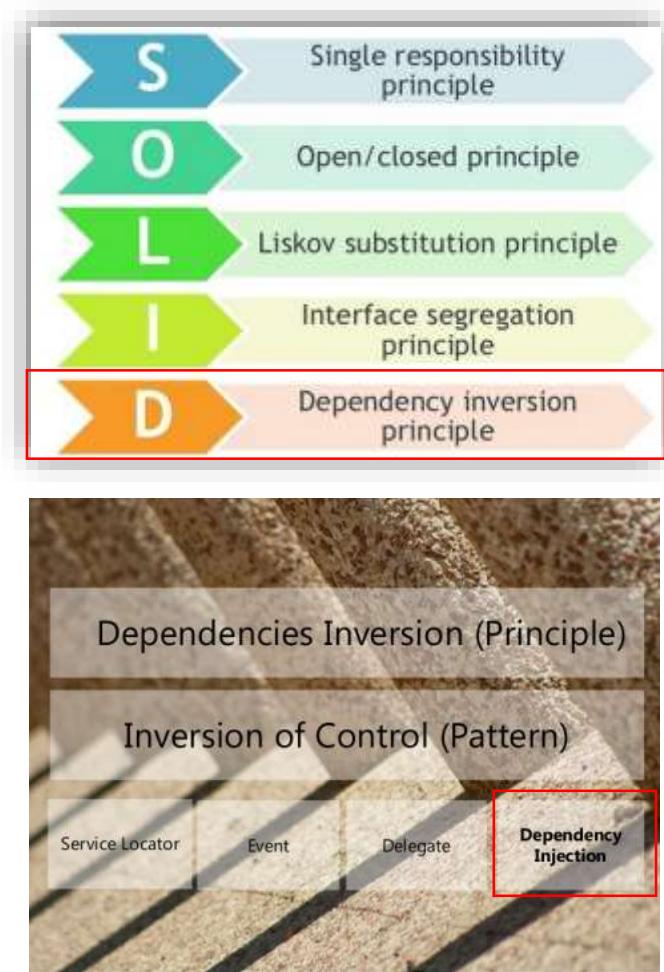


## Nguyên tắc hoạt động của Maven



- Khai báo trên tập tin **pom.xml** các dependencies phụ thuộc vào các thư viện có version rõ ràng.
- Khi lưu file pom.xml, Maven kiểm tra các thư viện này đã tồn tại ở máy tính (Local Repository) chưa. Nếu chưa Maven sẽ download về từ kho dữ liệu từ Internet (Repository).
- Maven sẽ tự động khai báo CLASSPATH cho Project tới vị trí file jar được download.

## SOLID Princiles



**IoC Container trong Spring** là core của Spring Framework. IoC Container, IoC tạo ra các đối tượng, nối các đối tượng lại với nhau, cấu hình, và quản lý life cycle (từ khi tạo ra đến khi bị hủy). IoC Container sử dụng DI để quản lý các thành phần tạo nên một ứng dụng. Các đối tượng này được gọi là Spring Bean.

IoC Container được cung cấp thông tin từ các tập tin XML.

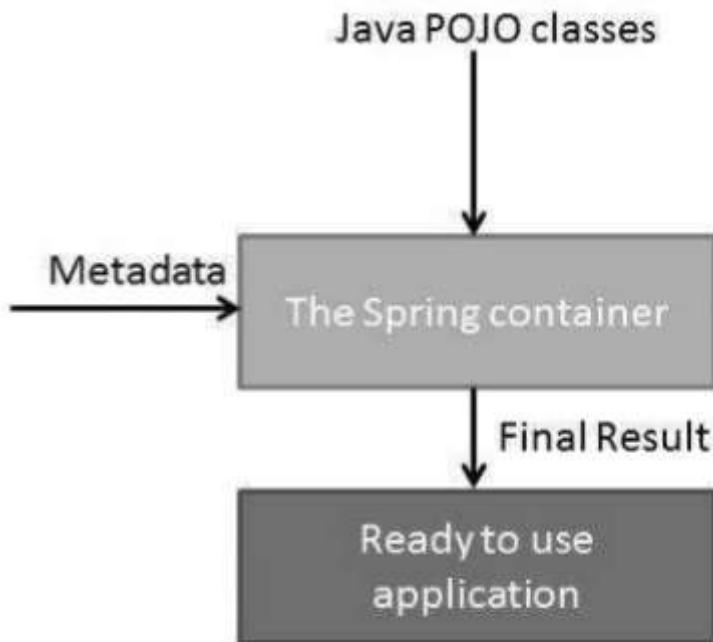
Hai loại IoC containers:

1. **BeanFactory**
2. **ApplicationContext**

- ApplicationContext (*org.springframework.context.ApplicationContext* interface) cũng tương tự như BeanFactory (*org.springframework.beans.factory.BeanFactory*), sử dụng để mô tả Spring Container. ApplicationContext được xây dựng trên interface BeanFactory.
- BeanFactory cung cấp các chức năng cơ bản trong khi ApplicationContext cung cấp các tính năng mở rộng hơn cho các ứng dụng Spring nên ApplicationContext phù hợp với các ứng dụng J2EE hơn. Lưu ý ApplicationContext bao gồm tất cả các chức năng của BeanFactory.
- Đối tượng của interface ApplicationContext như là một khung chứa của Spring để gọi đến đối tượng cần lấy bằng cách sử dụng phương thức *getBean()*.

```
ApplicationContext context = new ClassPathXmlApplicationContext("Beans.xml");
Student obj = (Student) context.getBean("studentbean");
obj.displayInfo();
```

```
Resource resource=new ClassPathResource("Beans.xml");
BeanFactory factory=new XmlBeanFactory(resource);
Student obj =(Student)factory.getBean("studentbean");
obj.displayInfo();
```



### Dependency Injection:

- Các module không giao tiếp trực tiếp với nhau, mà thông qua interface. Module cấp thấp sẽ implement interface, module cấp cao sẽ gọi module cấp thấp.

Để giao tiếp với database, sử dụng interface *IDatabase*, các module cấp thấp là *XMLDatabase*, *SQLDatabase*. Module cấp cao là *CustomerBusiness* sẽ sử dụng interface *IDatabase*.

- Việc khởi tạo các module cấp thấp sẽ do DI Container thực hiện.

Trong module *CustomerBusiness*, không khởi tạo *IDatabase db = new XMLDatabase()*, việc này sẽ do DI Container thực hiện. Module *CustomerBusiness* sẽ không biết gì về module *XMLDatabase* hay *SQLDatabase*.

- Các Module kết hợp với các interface cụ thể sẽ được config trong code hoặc trong file XML.
- DI được dùng để làm giảm sự phụ thuộc giữa các module, dễ dàng trong việc thay đổi module, bảo trì code và testing.

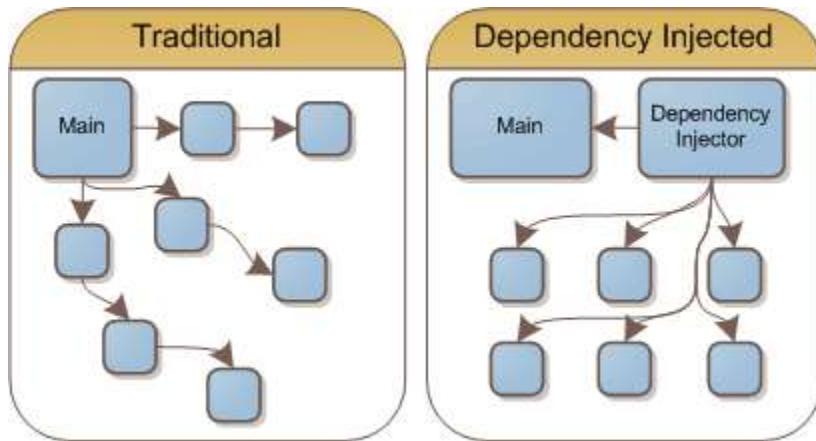
### Thực hiện Dependency Injection theo 3 cách:

- Constructor Injection:** Các dependency sẽ được container truyền/inject vào 1 class thông qua constructor của class.
- Setter Injection:** Các dependency sẽ được truyền vào 1 class thông qua các hàm setter.
- Interface Injection:** Class cần truyền/inject sẽ implement 1 interface. Interface này có chứa 1 hàm tên *Inject*. Container sẽ injection dependency vào 1 class thông qua việc gọi hàm *Inject* của interface.

### Bài 1. Spring DI

- Làm việc với Dependency Injection
- Thao tác Spring Dependency Injection

IoC trong Spring và phương pháp tiếp cận truyền thống.



Một nhân viên trong công ty sẽ có các thông tin như họ tên, địa chỉ .... Thiết kế đối tượng Student và Address dùng Dependency Injection (DI) trong Spring để loại bỏ sự phụ thuộc giữa các mã chương trình.

### Tham khảo hướng dẫn:

- Tạo Project Java
- Configure Project → Convert to Maven Project
- Cấu hình các dependency của Spring trong pom.xml

```

<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>02_DependencyInjection</groupId>
  <artifactId>02_DependencyInjection</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <build>
    <sourceDirectory>src</sourceDirectory>
  
```

```

<plugins>
    <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.7.0</version>
        <configuration>
            <release>9</release>
        </configuration>
    </plugin>
</plugins>
</build>
<dependencies>
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-core</artifactId>
        <version>5.0.3.RELEASE</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-beans -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-beans</artifactId>
        <version>5.0.3.RELEASE</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-context-->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.0.3.RELEASE</version>
    </dependency>
</dependencies>
</project>

```

#### B4. Tạo lớp Address

```

package se.iuh.edu.vn.di;

public class Address {
    private String city;
    private String state;
    private String country;

    public Address(String city, String state, String country) {
        super();
        this.city = city;
        this.state = state;
        this.country = country;
    }

    public String toString() {
        return city + " " + state + " " + country;
    }
}

```

B5. Tạo lớp Employee. Làm giảm sự phụ thuộc giữa Employee và Address, tạo lớp Address riêng và truyền/inject vào Employee thông qua Constructor (hoặc phương thức Setter).

Dùng Contructor

```
package se.iuh.edu.vn.di;

public class Employee {
    private int id;
    private String name;
    private Address address;// Aggregation (Employee, Address)

    public Employee() {

    }

    public Employee(int id, String name, Address address) {
        super();
        this.id = id;
        this.name = name;
        this.address = address;
    }

    void show() {
        System.out.println(id + " " + name);
        System.out.println(address.toString());
    }
}
```

Dùng Setter:

```
public class Employee {
    private Address address;

    // phương thức setter được sử dụng để truyền dependency
    public void setAddress(Address address) {
        this.address = address;
    }

    // phương thức getter trả về đối tượng Address
    public Address getAddress() {
        return address;
    }
}
```

B6. Cấu hình file bean *applicationContext.xml*. File lưu trong thư mục src

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:p="http://www.springframework.org/schema/p"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans-4.3.xsd">
```

```

<bean id="a1" class="se.iuh.edu.vn.di.Address">
    <constructor-arg value="HCMCity"></constructor-arg>
    <constructor-arg value="HCM"></constructor-arg>
    <constructor-arg value="Viet Nam"></constructor-arg>
</bean>

<bean id="e" class="se.iuh.edu.vn.di.Employee">
    <constructor-arg value="12" type="int"></constructor-arg>
    <constructor-arg value="Van"></constructor-arg>
    <constructor-arg>
        <ref bean="a1" />
    </constructor-arg>
</bean>

</beans>

```

## B7. Test chương trình

```

package se.iuh.edu.vn.di;

import org.springframework.beans.factory.support.DefaultListableBeanFactory;
import org.springframework.beans.factory.xml.XmlBeanDefinitionReader;

import org.springframework.core.io.*;

public class Test {
    public static void main(String[] args) {

        DefaultListableBeanFactory factory = new DefaultListableBeanFactory();
        XmlBeanDefinitionReader reader = new XmlBeanDefinitionReader(factory);
        reader.loadBeanDefinitions(new ClassPathResource("applicationContext.xml"));
        Employee e = (Employee) factory.getBean("e");

        e.show();

    }
}

```

## Bài 2. Spring AOP

- Làm việc với Aspect-Oriented Programming (tham khảo <http://spring.io>)
- Thao tác Spring AOP

Aspect Oriented Programming (AOP) là một kỹ thuật lập trình tách chương trình thành các module riêng, phân biệt, không phụ thuộc nhau. Khi hoạt động, chương trình sẽ kết hợp các module lại để thực hiện các chức năng, tuy nhiên khi có chỉnh sửa yêu cầu, chức năng thì chỉ cần sửa module cần thiết, không thay đổi cấu trúc.

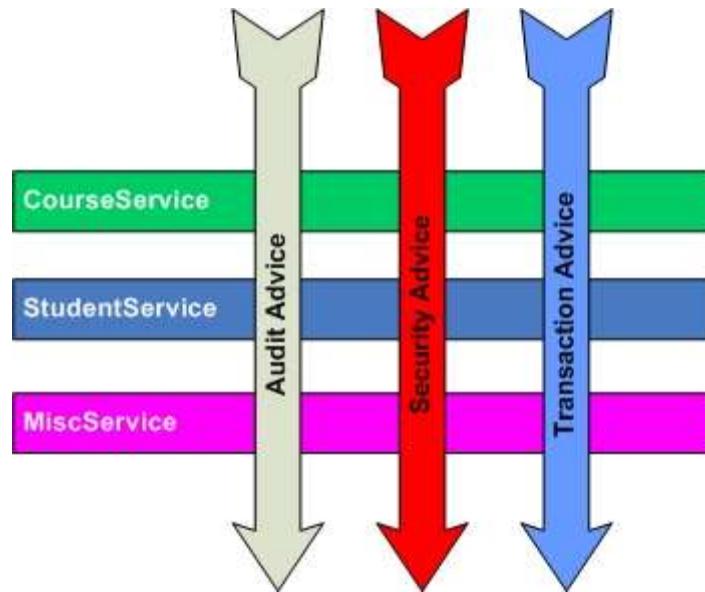
Yêu cầu bài toán	→ Module 1 (Aspect 1) → Kết hợp các Module → Ứng dụng thực tế
	→ Module 2 (Aspect 2) (Weaving)
	→ Module 3 (Aspect 3)
	→ Module 4 (Aspect 4)

## Các thuật ngữ trong AOP

- Core concerns: hàm chính của chương trình (các method cần thực hiện)
- Crosscutting concerns: những chức năng khác của chương trình: (chức năng)
- Join points: một điểm của chương trình, là nơi có thể chèn những “custom action”
- Pointcut: có nhiều cách để xác định joinpoint, những cách như thế được gọi là pointcut. Pointcut là expression language giúp khớp nối với joint point
- Advice: là những xử lý phụ được thêm vào xử lý chính, code để thực hiện các xử lý đó được gọi Advice. Advice cũng là một hành động thực hiện ở joint point
- Introduction: Cho phép introduce các new interface tới bất kì object advised nào.
- Target Object: Object sẽ được advised
- Aspect: là một class bao gồm các Advice và các Joint point
- Interceptor: là một aspect chỉ có duy nhất một advice
- AOP Proxy: dùng để cài đặt các Aspect
- Weaving: là tiến trình nối các aspect với các object, types để tạo nên advised object.

Aspect Oriented Programming (AOP) là kỹ thuật lập trình dùng để tách logic chương trình thành các phần riêng biệt... Trong Spring AOP, có 4 loại advice được hỗ trợ:

- Before advice: chạy trước khi method được thực thi
- After returning advice: chạy sau khi method trả về một kết quả
- After throwing advice: chạy khi method ném ra một exception
- Around advice: chạy khi method được thực thi (gồm cả 3 loại advice trên)



AspectJ là một thư viện, đặc tả trong Java để thực hiện AOP. AspectJ cung cấp các annotation, sử dụng đơn giản hơn.

Spring AOP định nghĩa trong Bean, AspectJ cung cấp các annotation để định nghĩa các pointcut, advice, target object... Các Annotation:

- @Before: Chạy trước khi method được thực thi.
- @After: Chạy sau khi method trả về một kết quả.
- @AfterReturning: Chạy sau khi method trả về một kết quả, lọc lấy kết quả trả về.
- @AfterThrowing: Chạy khi method xảy ra exception.
- @Around: Gồm các tất cả các Advice Before, After, AfterReturning, AfterThrowing.
- @Aspect: Đánh dấu đây là 1 Aspect.

Các dependencies hỗ trợ AspectJ

```
<!-- AspectJ -->
<dependency>
    <groupId>org.aspectj</groupId>
    <artifactId>aspectjrt</artifactId>
    <version>1.8.13</version>
</dependency>
<dependency>
    <groupId>org.aspectj</groupId>
    <artifactId>aspectjweaver</artifactId>
    <version>1.8.13</version>
</dependency>
```

## Thực hiện chương trình đơn giản minh họa AOP

Các dependencies để thực hiện AOP

```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-beans -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-beans</artifactId>
    <version>5.0.3.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.0.3.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-aop -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-aop</artifactId>
    <version>5.0.3.RELEASE</version>
</dependency>
```

Cấu hình Bean trong Spring Framework: applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:p="http://www.springframework.org/schema/p"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

    <bean id="hello" class="se.iuh.springaop.HelloAOP" />
    <bean id="demoBeforeAdvice" class="se.iuh.springaop.BeforeAdvice" />
    <bean id="demoAfterAdvice" class="se.iuh.springaop.AfterAdvice" />
    <bean id="demoThrowAdvice" class="se.iuh.springaop.ThrowAdvice" />
    <bean id="demoAroundAdvice" class="se.iuh.springaop.AroundAdvice" />

    <bean id="helloProxy" class="org.springframework.aop.framework.ProxyFactoryBean">
        <property name="target" ref="hello" />
        <property name="interceptorNames">
            <list>
                <value>demoBeforeAdvice</value>
                <value>demoAfterAdvice</value>
                <value>demoThrowAdvice</value>
                <value>demoAroundAdvice</value>
            </list>
        </property>
    </bean>

</beans>
```

## Lớp HelloAOP (lớp ứng dụng)

```
package se.iuh.springaop;

public class HelloAOP {
    public void method1() {
        System.out.println("+++++++++++++");
        System.out.println("method 1");
    }

    public void method2() {
        System.out.println("+++++++++++++");
        System.out.println("method 2");
    }

    public void method3() {
        System.out.println("+++++++++++++");
        System.out.println("method 3");
        throw new IllegalArgumentException();
    }
}
```

## Lớp BeforeAdvice, proxy thực hiện trước.

```
package se.iuh.springaop;

import java.lang.reflect.Method;
import org.springframework.aop.MethodBeforeAdvice;

public class BeforeAdvice implements MethodBeforeAdvice {

    public void before(Method method, Object[] args, Object target) throws Throwable {
        System.out.println("before method: " + method.getName());
    }
}
```

## Lớp AfterAdvice

```
package se.iuh.springaop;

import java.lang.reflect.Method;
import org.springframework.aop.AfterReturningAdvice;

public class AfterAdvice implements AfterReturningAdvice {

    public void afterReturning(Object returnValue, Method method, Object[] args, Object target) throws Throwable {
        System.out.println("after method: " + method.getName());
    }
}
```

## Lớp ThrowAdvice

```
package se.iuh.springaop;

import org.springframework.aop.ThrowsAdvice;

public class ThrowAdvice implements ThrowsAdvice {

    public void afterThrowing(IllegalArgumentException e) throws Throwable {
        System.out.println("throw advice method: " );
    }

}
```

## Lớp AroundAdvice

```
package se.iuh.springaop;

import org.aopalliance.intercept.MethodInterceptor;
import org.aopalliance.intercept.MethodInvocation;

public class AroundAdvice implements MethodInterceptor {

    public Object invoke(MethodInvocation invocation) throws Throwable {
        // same with MethodBeforeAdvice
        System.out.println("around - before method: " +
invocation.getMethod().getName());

        try {
            // proceed to original method call
            Object result = invocation.proceed();

            // same with AfterReturningAdvice
            System.out.println("around - before method: " +
invocation.getMethod().getName());

            return result;
        } catch (IllegalArgumentException e) {
            // same with ThrowsAdvice
            System.out.println("around - throw advice method: " +
invocation.getMethod().getName());
            throw e;
        }
    }
}
```

## Lớp kiểm tra AOP chạy:

```
package se.iuh.springaop;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
```

```
public class MainApp {  
    public static void main(String[] args) throws Exception {  
        ApplicationContext context = new  
ClassPathXmlApplicationContext("applicationContext.xml");  
        HelloAOP hello = (HelloAOP) context.getBean("helloProxy");  
        hello.method1();  
        System.out.println("\n");  
        hello.method2();  
        System.out.println("\n");  
        hello.method3();  
    }  
}
```

### Bài 3. Spring Bean

Làm việc với Spring Bean

- Auto-wiring ‘no’: chế độ auto-wiring mặc định, cần wire (nối) bean thông qua thuộc tính ‘ref’

```
<!-- Inject by setter -->
<bean id="person" class="se.iuh.springdiobject.Person">
    <property name="name" value="se.iuh"></property>
    <property name="age" value="25"></property>
    <property name="address" ref="address"></property>
</bean>

<bean id="address" class="se.iuh.springdiobject.Address">
    <property name="country" value="Viet Nam"></property>
    <property name="province" value="Ho Chi Minh"></property>
    <property name="district" value="District 6"></property>
</bean>
```

- Auto-wiring ‘byName’: Spring container sẽ tìm bean có id trùng với attribute Address trong class Person và thực thi auto wired thông qua method setter. public void setAddress(Address address)

```
public class Person {
    private String name;
    private int age;
    private Address address;

    public Person() {
    }

    public Person(String name, int age, Address address) {
        this.name = name;
        this.age = age;
        this.address = address;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Address getAddress() {
        return address;
    }

    public void setAddress(Address address) {
        this.address = address;
    }
}
```

Tìm kiếm bean có  
id=address

```

public int getAge() {
    return age;
}

public void setAge(int age) {
    this.age = age;
}

public void print() {
    System.out.println("Person: " + this.name + " Age: "
        + this.age + " Address: "
        + (this.address == null ? "null" : this.address.toString()));
}

}

```

Lúc đó cần khai báo cấu hình

```

<!-- Inject by setter -->
<bean id="person" class="se.iuh.springdiobject.Person" autowire="byName">
    <property name="name" value="se.iuh"></property>
    <property name="age" value="25"></property>
</bean>

<bean id="address" class="se.iuh.springdiobject.Address">
    <property name="country" value="Viet Nam"></property>
    <property name="province" value="Ho Chi Minh"></property>
    <property name="district" value="District 6"></property>
</bean>

```

- **Auto-wiring ‘byType’:** Spring container sẽ tìm bean có type là Address và thực thi auto wired thông qua method setter. public void setAddress(Address address)

```

public class Person {
    private String name;
    private int age;
    private Address address;

    public Person() {
    }

    public Person(String name, int age, Address address) {
        this.name = name;
        this.age = age;
        this.address = address;
    }

    public String getName() {
        return name;
    }
}

```

Tìm kiếm bean có  
type=address

```

    }

    public void setName(String name) {
        this.name = name;
    }

    public Address getAddress() {
        return address;
    }

    public void setAddress(Address address) {
        this.address = address;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public void print() {
        System.out.println("Person: " + this.name + " Age: "
            + this.age + " Address: "
            + (this.address == null ? "null" : this.address.toString()));
    }
}

```

Lúc đó cần khai báo cấu hình

```

<!-- Inject by setter -->
<bean id="person" class="se.iuh.springdiobject.Person" autowire="byType">
    <property name="name" value="se.iuh"></property>
    <property name="age" value="25"></property>
</bean>

<bean id="address" class="se.iuh.springdiobject.Address">
    <property name="country" value="Viet Nam"></property>
    <property name="province" value="Ho Chi Minh"></property>
    <property name="district" value="District 6"></property>
</bean>

```

- **Auto-Wiring ‘constructor’**: Spring container sẽ tìm bean có type giống với type của address trong method constructor và thực hiện auto wired thông qua method constructor – public Person(Address address)

```

public class Person {
    private String name;
    private int age;
    private Address address;

    public Person() {}

    public Person(Address address) {
        this.address = address;
    }

    public Person(String name, int age, Address address) {
        this.name = name;
        this.age = age;
        this.address = address;
    }

    ...
}

```

Tìm kiếm bean có type=address

Lúc đó cần khai báo cấu hình

```

<!-- Inject by setter -->
<bean id="person" class="se.iuh.springdiobject.Person" autowire="constructor">
    <property name="name" value="se.iuh"></property>
    <property name="age" value="25"></property>
</bean>

<bean id="address" class="se.iuh.springdiobject.Address">
    <property name="country" value="Viet Nam"></property>
    <property name="province" value="Ho Chi Minh"></property>
    <property name="district" value="District 6"></property>
</bean>

```

- Auto-Wiring ‘autodetect’ dùng đối với Spring phiên bản < 3. Spring Container sẽ thử với auto wired byConstructor, nếu không được thì chuyển sang auto wired byType.
- Annotation @Autowired trong Spring

Annotation @Autowired cho phép các thuộc tính sẽ được auto wired. Auto wired byType khai báo @Autowired ở trước phần khai báo thuộc tính hoặc trước method setter:

```

@Autowired(required = false)
private Address address;

```

//hoặc

```
@Autowired(required = false)
public void setAddress(Address address) {
    this.address = address;
}
```

Để auto wired byConstructor khai báo @Autowired ở trước method Constructor:

```
@Autowired(required=true)
public Person(Address address) {
    this.address = address;
}
```

Lưu ý:

Thuộc tính required của annotation @Autowired mặc định là true.

- required = true, nếu Spring container không tìm thấy bean address để inject vào thì sẽ báo lỗi
- required = false, nếu Spring container không tìm thấy bean address để inject vào thì sẽ inject null

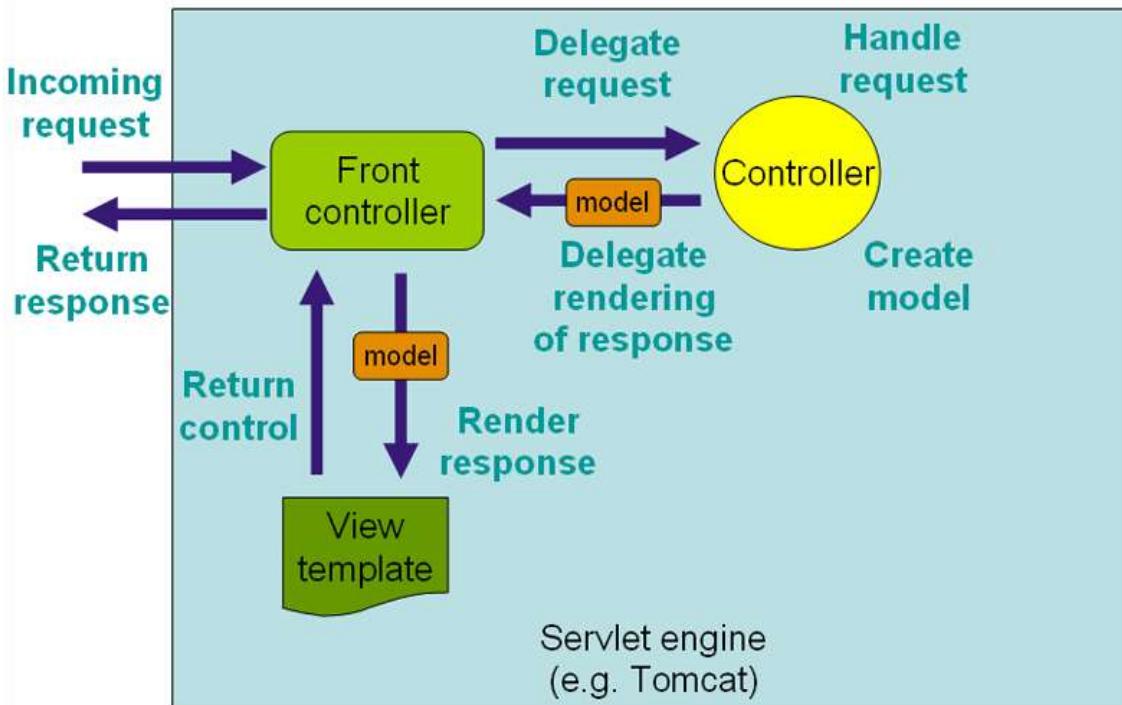
@Autowired là một annotation config của Spring, để sử dụng cần phải khai báo trong file cấu hình Spring: <context:annotation-config /> kèm theo namespace context:

```
<beans:bean xmlns="http://www.springframework.org/schema/mvc"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:beans="http://www.springframework.org/schema/beans"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/mvc
    http://www.springframework.org/schema/mvc/spring-mvc.xsd
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd">
...
</beans:bean>
```

## Phân 2. Spring MVC

### 0. Mô hình Spring MVC trong môi trường Web

- Front Controller tiếp cận tất cả các request gửi tới, chuyển tới Controller cần xử lý.
- Controller trả về các Model cần thiết cho Front Controller và chọn View phù hợp.
- Front Controller cũng trả về kết quả cho người dùng.



Với Java, *Front Controller* thường là đối tượng implement interface **javax.servlet.Servlet**. Trong Spring MVC thì Front Controller là **org.springframework.web.servlet.DispatcherServlet**.

#### Các bước thực hiện:

- B1. Request tới ứng dụng Web, *DispatcherServlet* là đối tượng nhận tất cả các request.
- B2. Tìm và chuyển hướng request tới Request Handler phù hợp là các Controllers trong ứng dụng Web.
- B3. Các Controller xử lý request yêu cầu.
- B4. Chuẩn bị Model và chọn View hiển thị.
- B5. Trả về kết quả xử lý request cho *DispatcherServlet*.
- B6. *DispatcherServlet* sẽ gọi View Template phù hợp để xử lý việc hiển thị trên giao diện bằng cách sử dụng Model.
- B7. View Template trả kết quả về cho *DispatcherServlet*.
- B8. Trả Response.

1. Cấu hình Front Controller **org.springframework.web.servlet.DispatcherServlet**: thường để trong **web.xml** (Phiên bản Servlet 3.0 trở lên có thể dùng **web-fragment.xml** hoặc dùng **javax.servlet.ServletContainerInitializer**)

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">

    <!-- The definition of the Root Spring Container shared by all Servlets and Filters
-->
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/spring/root-context.xml</param-value>
    </context-param>

    <!-- Creates the Spring Container shared by all Servlets and Filters -->
    <listener>
        <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
        </listener>
    
```

Đường dẫn đến tập tin cấu hình xml

```

        <!-- Processes application requests -->
        <servlet>
            <servlet-name>appServlet</servlet-name>
            <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
            <init-param>
                <param-name>contextConfigLocation</param-name>
                <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-
value>
            </init-param>
            <load-on-startup>1</load-on-startup>
        </servlet>
    
```

```

        <servlet-mapping>
            <servlet-name>appServlet</servlet-name>
            <url-pattern>/</url-pattern>
        </servlet-mapping>
    
```

</web-app>

Mặc định khi *DispatcherServlet* được khởi tạo, đối tượng này sẽ khởi tạo một đối tượng *org.springframework.web.context.WebApplicationContext* với hiện thực là đối tượng *org.springframework.web.context.support.XmlWebApplicationContext*.

Đối tượng *XmlWebApplicationContext* chứa cấu hình tất cả các beans mà ứng dụng định nghĩa trong khung chứa (container) của Spring. Đối tượng *XmlWebApplicationContext* này sẽ dùng một tập tin cấu hình của Spring với tên gọi mặc định là **[tên-servlet]-servlet.xml** nằm trong thư mục */src/webapp/WEB-INF* (trong Spring Tool Suite STS) hoặc */WEB-INF* với ứng dụng SpringMVC tạo bằng Dynamic Web Project trong Eclipse Oxygen hoặc STS.

## 2. Khai báo các beans trong root-servlet

### Cấu hình root Servlet

Trường hợp định nghĩa nhiều Servlet trong một tập tin *web.xml*, mỗi servlet có thể được khởi tạo, ánh xạ đến các URL khác nhau. Dẫn đến trường hợp này sẽ có nhiều Spring container tương ứng với từng Servlet. Trong trường hợp đó, một số định nghĩa bean có thể lặp đi lặp lại trong nhiều Spring container.

Đối tượng *org.springframework.web.context.ContextLoaderListener* được tạo ra để giải quyết vấn đề lặp đi lặp lại này bằng cách **tạo ra một root *org.springframework.web.context.WebApplicationContext* sử dụng chung cho tất cả các Servlet**. *ContextLoaderListener* sẽ sử dụng tập tin được định nghĩa trong *contextConfigLocation* của tag *<context-param>*.

Thường trong tập tin *root-context.xml*, định nghĩa các bean, các thuộc tính dùng chung giữa các container của Spring, mỗi container Spring trong mỗi Servlet sẽ sử dụng các bean.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:p="http://www.springframework.org/schema/p"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.0.xsd">

    <!-- Khai báo package chứa các thành phần xử lý Controller -->
    <context:component-scan base-
package="se.iuh.edu.vn.controller"></context:component-scan>

    <!-- Khai báo ViewResolver (xử lý View), các View là tập tin JSP được ánh
        xa từ thư mục /WEB-INF/jsp/ -->
    <bean
        class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/jsp/"></property>
        <property name="suffix" value=".jsp"></property>
    </bean>
    <bean id="ds"
        class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName"
value="com.microsoft.sqlserver.jdbc.SQLServerDriver"></property>
        <property name="url"
value="jdbc:sqlserver://localhost:1433;databaseName=UploadFileServletDB"></property>
        <property name="username" value="sa"></property>
        <property name="password" value="1234567890"></property>
    </bean>

    <bean id="jt" class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource" ref="ds"></property>
    </bean>
```

```

</bean>

<bean id="dao" class="se.iuh.edu.vn.dao.EmployeeDao">
    <property name="template" ref="jt"></property>
</bean>
</beans>

```

*InternalResourceViewResolver* sẽ map tên View mà trả về trong Controller với một tên file nằm trong một thư mục cụ thể.

Khai báo bean *InternalResourceViewResolver* sẽ map tất cả các tên View với các file có phần mở rộng .jsp nằm trong thư mục /src/main/WEB-INF/views để render giao diện.

- Định nghĩa request “/” với tên View trả về trong method home() là “home”, tương ứng với file /src/main/WEB-INF/views/home.jsp.
- Định nghĩa request “/login” với tên View trả về trong method home() là “home”, tương ứng với file /src/main/WEB-INF/views/login.jsp.
- Định nghĩa request “/home” với tên View trả về trong method home() là “home”, tương ứng với file /src/main/WEB-INF/views/user.jsp.

```

@Controller
public class HomeController {

    @RequestMapping(value = "/", method = RequestMethod.GET)
    public String home(Locale locale, Model model) {
        ...
        return "home";
    }

    @RequestMapping(value = "/login", method = RequestMethod.GET)
    public String loginPage(Locale locale, Model model) {
        return "login";
    }

    @RequestMapping(value = "/home", method = RequestMethod.POST)
    public String login(@Validated User user, Model model) {
        model.addAttribute("userName", user.getUserName());
        return "user";
    }
}

```

Cách ánh xạ giữa View và tập tin View:

- home → /WEB-INF/views/home.jsp
- admin/login → /WEB-INF/views/admin/login.jsp

## 2. Tạo Controller trong Spring MVC

- Annotation `@Controller` khai báo cùng với định nghĩa của lớp **XXXController**.

- Annotation **@RequestMapping** khai báo cùng với định nghĩa của phương thức xử lý nào đó **xulyData()**. Với annotation **@RequestMapping** có `@RequestMapping(value = "/", method = RequestMethod.GET)`
  - thuộc tính **value** để định nghĩa request URL
  - thuộc tính **method** để định nghĩa HTTP request method.

```

import java.text.DateFormat;
import java.util.Date;
import java.util.Locale;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
public class HomeController {

    private static final Logger logger = LoggerFactory.getLogger(HomeController.class);
    @RequestMapping(value = "/", method = RequestMethod.GET)
    public String home(Locale locale, Model model) {
        logger.info("Welcome home! The client locale is {}.", locale);

        Date date = new Date();
        DateFormat dateFormat = DateFormat.getDateInstance(DateFormat.LONG,
        DateFormat.LONG, locale);

        String formattedDate = dateFormat.format(date);

        model.addAttribute("serverTime", formattedDate);

        return "home";
    }
}

```

Spring MVC có thể định nghĩa một Controller sử dụng interface `org.springframework.web.servlet.mvc.Controller`, hoặc annotation `org.springframework.stereotype.Controller`.

Lưu ý:

- Trường hợp phương thức xử lý nhiều request:

```

@RequestMapping(value = {" /home", " /login", " /logout"}, method = RequestMethod.GET)
public String home(Locale locale, Model model) {

```

- Trong trường hợp cả class và phương thức đều được khai báo với annotation **@RequestMapping** cùng với request URL thì request URL mà người dùng sử dụng để truy cập đến một trang nào đó phải là request URL của lớp cộng với request URL của phương thức.

```

import java.text.DateFormat;
import java.util.Date;
import java.util.Locale;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
@RequestMapping("test")
public class HomeController {

    private static final Logger logger = LoggerFactory.getLogger(HomeController.class);

    @RequestMapping(value = "home", method = RequestMethod.GET)
    public String home(Locale locale, Model model) {
        logger.info("Welcome home! The client locale is {}.", locale);

        Date date = new Date();
        DateFormat dateFormat = DateFormat.getDateTimeInstance(DateFormat.LONG,
        DateFormat.LONG, locale);

        String formattedDate = dateFormat.format(date);

        model.addAttribute("serverTime", formattedDate);

        return "home";
    }
}

```

Truy cập: <http://localhost:8080/.../test/home>

- Tạo Controller để kiểm soát handle request từ client gửi tới dạng QueryString: /login?username=xxx&password=xxx

```

import javax.servlet.http.HttpServletRequest;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class LoginController {

    @RequestMapping(value = "/login")
    public void login(HttpServletRequest request) {

```

```

        String username = request.getParameter("username");
        String password = request.getParameter("password");
    }
}

```

Hoặc dùng @RequestParam annotation

```

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
public class LoginController {

    @RequestMapping(value = "/login")
    public void login(@RequestParam String username, @RequestParam String password) {
    }
}

```

Nếu biến trong QueryString và biến phương thức khác tên thì thay đổi:

```
public void login(@RequestParam(value = "username") String user, @RequestParam String password) {}
```

Nếu bắt buộc phải có giá trị (khác null):

```
public void login(@RequestParam(value = "username", required = false) String user,
    @RequestParam String password) {}
```

Nếu muốn gán giá trị mặc định:

```
public void login(@RequestParam(value = "username", required = false, defaultValue =
"xxxx") String user, @RequestParam String password) {}
```

Muốn dữ liệu trả về ở định dạng JSON: dùng thuộc tính produces trong annotation @RequestMapping với giá trị là “application/json”:

```

import org.springframework.http.MediaType;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HomeController {

    @RequestMapping(value = "/hello", produces = MediaType.APPLICATION_JSON_VALUE)
    public String hello() {
        return "Hello World!";
    }
}

```

Trong đó `@RestController` annotation để tạo Controller phát triển các ứng dụng RESTful Web Service. Trả về dữ liệu dạng JSON, XML,... không cần View. `@RestController` thay thế cho `@Controller` và `@ResponseBody`.

```
import org.springframework.http.MediaType;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@Controller
public class HomeController {

    @RequestMapping(value = "/hello", produces = MediaType.APPLICATION_JSON_VALUE)
    @ResponseBody
    public String hello() {
        return "Hello World!";
    }
}
```

Nếu dữ liệu gửi tới từ Request tới Controller dưới dạng JSON, và Controller chỉ chấp nhận Request này là JSON dùng thuộc tính `consumes`

```
import org.springframework.http.MediaType;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloController {

    @RequestMapping(value = "/hello", consumes = MediaType.APPLICATION_JSON_VALUE)
    public String hello() {
        return "Hello World!";
    }
}
```

### 3. Đối tượng HttpServletRequest và HttpServletResponse trong Controller của Spring MVC

```
import java.text.DateFormat;
import java.util.Date;
import java.util.Locale;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```
@Controller
public class HomeController {

    private static final Logger logger = LoggerFactory.getLogger(HomeController.class);
    @RequestMapping(value = "/", method = RequestMethod.GET)
    public String home(Locale locale, Model model, HttpServletRequest request,
HttpServletResponse response) {
        logger.info("Welcome home! The client locale is {}.", locale);
        logger.info(request.getContextPath());

        Date date = new Date();
        DateFormat dateFormat = DateFormat.getDateInstance(DateFormat.LONG,
DateFormat.LONG, locale);

        String formattedDate = dateFormat.format(date);
        model.addAttribute("serverTime", formattedDate);

        return "home";
    }
}
```

### 4. Truyền dữ liệu vào Model trong Controller (Spring MVC)

- C1. Dùng đối tượng `org.springframework.ui.Model`, và phương thức `addAttribute("Tên thuộc tính", giá trị)`;

```
@RequestMapping(value = "/", method = RequestMethod.GET)
public String home(Locale locale, Model model) {
    ...
    model.addAttribute("Tên thuộc tính", giá trị);
}
```

- C2. Dùng *java.util.Map*

```
@RequestMapping(value = "/", method = RequestMethod.GET)
public String home(Locale locale, Map<String, Object> model) {
    ...
    model.put("Tên thuộc tính", giá trị);
}
```

- C3. Dùng *org.springframework.ui.ModelMap*

```
@RequestMapping(value = "/", method = RequestMethod.GET)
public String home(Locale locale, ModelMap model) {
    ...
    model.addAttribute("Tên thuộc tính", giá trị)
}
```

Chuyển hướng (send redirect) request trong controller của Spring MVC

```
import java.text.DateFormat;
import java.util.Date;
import java.util.Locale;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
public class HomeController {
    @RequestMapping(value = "/", method = RequestMethod.GET)
    public String home(Locale locale, Model model) {
        return "redirect:/home";
    }
}
```

## 5. Đối tượng Enum: *org.springframework.web.bind.annotation.RequestMethod* gồm các request method

- RequestMethod.GET
- RequestMethod.HEAD
- RequestMethod.POST
- RequestMethod.PUT
- RequestMethod.PATCH
- RequestMethod.DELETE
- RequestMethod.OPTIONS
- RequestMethod TRACE

## Bài 4. Spring MVC - Chương trình đơn giản Hello Word

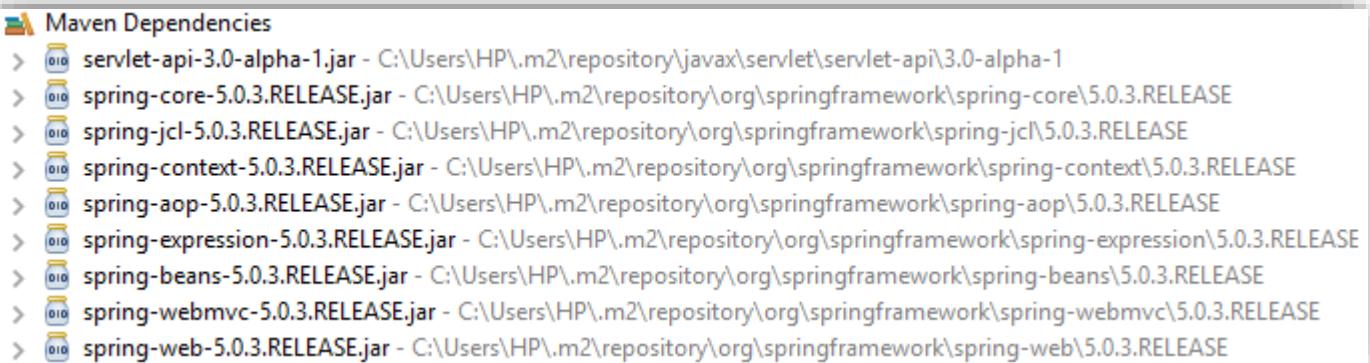
Thực hiện ứng dụng Spring MVC Hello World

B1. Dùng Spring Tool Suite (STS) tạo **Dynamic Web Project**, khai báo các thông tin liên quan

- **web.xml**: Khai báo cấu hình cần thiết của ứng dụng Web (để trong thư mục WebContent)
- **WebContent/dispatcher-servlet.xml**: Khai báo cấu hình tối thiểu về Spring MVC
- **index.jsp**: Trang khởi đầu mặc định của ứng dụng Web
- **se.iuh.edu.vn.controller.HelloController**: Controller đơn giản
- **WEB-INF/jsp/hellopage.jsp**: View đơn giản
- **WEB-INF/lib/\*.jar**: Các tập tin thư viện cần thiết
- **pom.xml**: Khai báo các dependencies (Nếu khai báo dependencies trong **pom.xml** thì không copy thư viện vào **WEB-INF/lib/\*.jar**)

B2. Khai báo các dependencies

```
<dependencies>
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId> servlet-api</artifactId>
        <version>3.0-alpha-1</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId> spring-core</artifactId>
        <version>5.0.3.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId> spring-context</artifactId>
        <version>5.0.3.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId> spring-beans</artifactId>
        <version>5.0.3.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId> spring-webmvc</artifactId>
        <version>5.0.3.RELEASE</version>
    </dependency>
</dependencies>
```



### B3. Khai báo các cấu hình cần thiết cho ứng dụng web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
        http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
    id="WebApp_ID" version="3.1">
    <display-name>HelloMVC</display-name>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>

    <servlet>
        <servlet-name>spring</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>spring</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>

    <!-- Các request đến ứng dụng Web đều được Encoding UTF-8 -->
    <filter>
        <filter-name>encodingFilter</filter-name>
        <filter-class>
            org.springframework.web.filter.CharacterEncodingFilter
        </filter-class>
        <init-param>
            <param-name>encoding</param-name>
            <param-value>UTF-8</param-value>
        </init-param>
        <init-param>
            <param-name>forceEncoding</param-name>
            <param-value>true</param-value>
        </init-param>
    </filter>
    <filter-mapping>
        <filter-name>encodingFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
</web-app>
```

### B4. Cấu hình cho phép sử dụng Spring MVC Annotation dispatcher-servlet.xml

- Khai báo package chứa các thành phần xử lý Controller: Các component (@Controller, @Service, @Repository, @Component...) sẽ được tìm trong se.iuh.edu.vn.controller.
- Khai báo ViewResolver (xử lý View), các View là tập tin JSP được ánh xạ từ thư mục /WEB-INF/jsp/. Các View đặt trong /WEB-INF/jsp/ và kèm theo phần loại tập tin cho View là .jsp
- Cho phép sử dụng annotation trong ứng dụng Spring MVC.

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:p="http://www.springframework.org/schema/p"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-3.0.xsd">

    <!-- Cấu hình cho phép sử dụng Spring MVC Annotation <mvc:annotation-driven/>
        <context:annotation-config /> -->

    <!-- Khai báo package chứa các thành phần xử lý Controller -->
    <context:component-scan base-package="se.iuh.edu.vn.controller"></context:component-scan>

    <!-- Khai báo ViewResolver (xử lý View),
        các View là tập tin JSP được ánh xạ từ thư mục /WEB-INF/jsp/ -->
    <bean
        class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/jsp/"></property>
        <property name="suffix" value=".jsp"></property>
    </bean>

</beans>

```

## B5. Trang chủ index.jsp

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Hello MVC</title>
</head>
<body>

    <%= java.util.Calendar.getInstance().getTime() %>
    <a href="hello">click here</a>
</body>
</html>

```

## B6. Tạo HelloController

HelloController định nghĩa action có tên hello (mặc định gọi theo phương thức GET). Action hello sẽ gọi tới vào model một thuộc tính msg="Hello World" và chuyển về view hellopage.jsp đặt tại /WEBINF/jsp/

```

package se.iuh.edu.vn.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

@Controller
public class HelloController {

    @RequestMapping("/hello")
    public ModelAndView mymethod(){
        return new ModelAndView("hellopage","msg","Hello World!");
    }
}

```

### B8. Tạo *hellopage.jsp*

Hiển thị giá trị của thuộc tính msg được tạo trong controller trước đó.

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Hello</title>
</head>
<body>

    Message is: ${msg}
</body>
</html>

```

## Bài 5. Spring MVC - Thao tác với Form

Kiểm soát dữ liệu truyền qua Form dùng Spring MVC

B1. Dùng Spring Tool Suite (STS) tạo Dynamic Web Project, khai báo các thông tin liên quan

- *web.xml*: Khai báo cấu hình cần thiết của ứng dụng Web (để trong thư mục WebContent)
- *WebContent/dispatcher-servlet.xml*: Khai báo cấu hình tối thiểu về Spring MVC
- *index.jsp*: Trang khởi đầu mặc định của ứng dụng Web
- *se.iuh.edu.vn.beans.Employee*: Model.
- *se.iuh.edu.vn.controller.EmployeeController*: Controller bao gồm action hiển thị form và action hiển thị danh sách nhân viên
- *WEB-INF/jsp/empform.jsp*, *WEB-INF/jsp/viewemp.jsp*: View,
- (Nếu khai báo dependencies trong *pom.xml* thì không copy thư viện vào *WEB-INF/lib/\*.jar*)

B2. Khai báo các dependencies: [Tương tự Bài 5](#)

B3. Khai báo các cấu hình cần thiết cho ứng dụng *web.xml*: [Tương tự Bài 5](#)

B4. Cấu hình cho phép sử dụng Spring MVC Annotation *dispatcher-servlet.xml*: [Tương tự Bài 5](#)

## B5. Trang chủ index.jsp

```
<a href="empform">Add Employee</a> <a href="viewemp">View Employees</a>
```

## B6. Tạo Model: se.iuh.edu.vn.beans.Employee

```
package se.iuh.edu.vn.beans;

public class Employee {
    private int id;
    private String name;
    private float salary;
    private String designation;

    public Employee() {
    }

    public Employee(int id, String name, float salary, String designation) {
        super();
        this.id = id;
        this.name = name;
        this.salary = salary;
        this.designation = designation;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public float getSalary() {
        return salary;
    }

    public void setSalary(float salary) {
        this.salary = salary;
    }

    public String getDesignation() {
        return designation;
    }

    public void setDesignation(String designation) {
        this.designation = designation;
    }
}
```

## B7. Tạo Controller: *se.iuh.edu.vn.controller.EmployeeController*

```
package se.iuh.edu.vn.controller;

import java.util.ArrayList;
import java.util.List;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;

import se.iuh.edu.vn.beans.Employee;

@Controller
public class EmployeeController {
    List<Employee> list; // Danh sách Employee
    int count = 4;
    public EmployeeController()
    {
        list=new ArrayList<Employee>(); // Khởi tạo giá trị mặc định
        list.add(new Employee(1,"rahul",35000f,"S.Engineer"));
        list.add(new Employee(2,"aditya",25000f,"IT Manager"));
        list.add(new Employee(3,"sachin",55000f,"Care Taker"));
    }

    @RequestMapping("/empform")
    public ModelAndView showform(){
        // Hiển thị form, mặc định khởi tạo new Employee()
        return new ModelAndView("empform","command",new Employee());
    }
    @RequestMapping(value="/save",method = RequestMethod.POST)
    public ModelAndView save(@ModelAttribute("emp") Employee emp){
        // Thêm Employee mới vào danh sách list
        list.add(new Employee(count++, emp.getName(),
            emp.getSalary(), emp.getDesignation()));

        // Hiển thị dữ liệu vừa thêm
        // return new ModelAndView("emptorm","command",emp);
        // chuyển qua request mapping viewemp
        return new ModelAndView("redirect:/viewemp");
    }

    @RequestMapping("/viewemp")
    public ModelAndView viewemp(){
        // Hiển thị danh sách
        return new ModelAndView("viewemp","list",list);
    }
}
```

## B8. Tạo *empform.jsp*

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<form:form method="post" action="save">
    <table>
        <tr>
            <td>Name :</td>
            <td><form:input path="name" /></td>
        </tr>
        <tr>
            <td>Salary :</td>
            <td><form:input path="salary" /></td>
        </tr>
        <tr>
            <td>Designation :</td>
            <td><form:input path="designation" /></td>
        </tr>
        <tr>
            <td colspan="2"><input type="submit" value="Save" /></td>
        </tr>
    </table>
</form:form>

```

Tạo viewemp.jsp

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<table border="2" width="70%" cellpadding="2">
    <tr>
        <th>Id</th>
        <th>Name</th>
        <th>Salary</th>
        <th>Designation</th>
    </tr>
    <c:forEach var="emp" items="${list}">
        <tr>
            <td>${emp.id}</td>
            <td>${emp.name}</td>
            <td>${emp.salary}</td>
            <td>${emp.designation}</td>
        </tr>
    </c:forEach>
</table>

```

## Bài 6. Spring MVC - CRUD

Spring MVC với các thao tác CRUD thêm, xóa, sửa với cơ sở dữ liệu.

Thực hiện thao tác thêm xóa sửa với CSDL quản lý thông tin nhân viên. CSDL quản lý bao gồm bảng Employee (ID, NAME, SALARY, DESIGNATION), trong đó mã nhân viên ID tự động tăng.

# Employees List

<b>Id</b>	<b>Name</b>	<b>Salary</b>	<b>Designation</b>	<b>Edit</b>	<b>Delete</b>
1	Thanh Van	2400000.0	Tester	<a href="#">Edit</a>	<a href="#">Delete</a>
3	Hoàng Giang	2300000.0	Developer	<a href="#">Edit</a>	<a href="#">Delete</a>
4	Hồng Minh	2400000.0	QA	<a href="#">Edit</a>	<a href="#">Delete</a>
5	Hoàng Khánh	2500000.0	Developer	<a href="#">Edit</a>	<a href="#">Delete</a>
6	Ngọc Dung	2200000.0	QA	<a href="#">Edit</a>	<a href="#">Delete</a>

[Add New Employee](#)

```
CREATE TABLE Emp99(
    [ID] [numeric](18, 0) NOT NULL identity(1, 1),
    [NAME] [nvarchar](50) NULL,
    [SALARY] [numeric](18, 0) NULL,
    [DESIGNATION] [nvarchar](50) NULL,
    PRIMARY KEY (ID)
)
```

Thực hiện các chức năng:

- Khi chương trình chạy cho phép hiển thị danh sách nhân viên.
- Chọn “Edit” cho phép sửa thông tin của từng nhân viên, chọn “Delete” cho phép xóa thông tin của nhân viên và cập nhật CSDL..
- Chọn “Add New Employee” hiển thị form cho phép thêm nhân viên mới vào CSDL.

Tham khảo hướng dẫn:

B1. Dùng Spring Tool Suite (STS) tạo Dynamic Web Project, khai báo các thông tin liên quan

B2. Khai báo các dependencies trong pom.xml:

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
    <version>5.0.3.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.0.3.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-beans</artifactId>
    <version>5.0.3.RELEASE</version>
</dependency>
```

```

<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>5.0.3.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>5.0.3.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.microsoft.sqlserver/sqljdbc4 -->
<dependency>
    <groupId>com.microsoft.sqlserver</groupId>
    <artifactId>sqljdbc4</artifactId>
    <version>4.0</version>
    <scope>runtime</scope>
</dependency>
<!-- https://mvnrepository.com/artifact/jstl/jstl -->
<dependency>
    <groupId>jstl</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
</dependency>

```

B3. Khai báo các cấu hình cần thiết cho ứng dụng *web.xml*:

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" id="WebApp_ID" version="3.1">
    <display-name>SpringMVC_CRUD_SQLServer</display-name>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
    <servlet>
        <servlet-name>spring</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>spring</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
</web-app>

```

B4. Cấu hình cho phép sử dụng Spring MVC Annotation *spring-servlet.xml*:

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:p="http://www.springframework.org/schema/p"

```

```

xmlns:context="http://www.springframework.org/schema/context"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.0.xsd">

<!-- Khai báo package chứa các thành phần xử lý Controller -->
<context:component-scan base-
package="se.iuh.edu.vn.controller"></context:component-scan>

<!-- Khai báo ViewResolver (xử lý View), các View là tập tin JSP được ánh
xạ từ thư mục /WEB-INF/jsp/ -->
<bean
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix" value="/WEB-INF/jsp/"></property>
    <property name="suffix" value=".jsp"></property>
</bean>
<bean id="ds"
    class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName"
value="com.microsoft.sqlserver.jdbc.SQLServerDriver"></property>
    <property name="url"
value="jdbc:sqlserver://localhost:1433;databaseName=EmployeeDB"></property>
    <property name="username" value="sa"></property>
    <property name="password" value="1234567890"></property>
</bean>

<bean id="jt" class="org.springframework.jdbc.core.JdbcTemplate">
    <property name="dataSource" ref="ds"></property>
</bean>

<bean id="dao" class="se.iuh.edu.vn.dao.EmployeeDao">
    <property name="template" ref="jt"></property>
</bean>
</beans>

```

- *ds*: là bean cấu hình các thông số kết nối CSDL
- *JdbcTemplate*: *jt* là bean được khai báo để làm việc với CSDL được inject vào và sử dụng sau này trong ứng dụng

## B5. Lớp mô tả dữ liệu (Entity)

```

package se.iuh.edu.vn.beans;

public class Employee {
    private int id;
    private String name;
    private float salary;
    private String designation;

    + public int getId() {..}

    + public void setId(int id) {..}

    + public String getName() {..}

    + public void setName(String name) {..}

    + public float getSalary() {..}

    + public void setSalary(float salary) {..}

    + public String getDesignation() {..}

    + public void setDesignation(String designation) {..}

}

```

B6. Lớp truy xuất dữ liệu (DAO): Lớp này chứa các phương thức thao tác dữ liệu (thêm, sửa, xóa) và truy vấn dữ liệu.

- insert(): thêm
- update(): sửa
- delete(): xóa
- getXyz(): truy vấn

Lớp nào được chú thích bởi `@Repository` để có thể inject vào Controller trong ứng dụng bởi `@Autowired` để sử dụng sau này. Khai báo này có thể để trong lớp EmployeeDao hoặc khai báo bên Controller.

```

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.BeanPropertyRowMapper;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.stereotype.Repository;
...

@Repository
public class EmployeeDao {
    @Autowired
    JdbcTemplate template;
    ...
}

```

```

package se.iuh.edu.vn.dao;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.List;
import org.springframework.jdbc.core.BeanPropertyRowMapper;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;
import se.iuh.edu.vn.beans.Employee;

public class EmployeeDao {
    JdbcTemplate template;

    public void setTemplate(JdbcTemplate template) {
        this.template = template;
    }

    public int save(Employee p) {
        String sql = "insert into Empployee(name,salary,designation) values('"
            + p.getName() + "','" + p.getSalary()
            + "','" + p.getDesignation() + "')";
        return template.update(sql);
    }

    public int update(Employee p) {
        String sql = "update Empployee set name='" + p.getName() + "', salary="
            + p.getSalary() + ", designation='"
            + p.getDesignation() + "' where id=" + p.getId() + "";
        return template.update(sql);
    }

    public int delete(int id) {
        String sql = "delete from Empployee where id=" + id + "";
        return template.update(sql);
    }

    public Employee getEmpById(int id) {
        String sql = "select * from Empployee where id=?";
        return template.queryForObject(sql, new Object[] { id },
            new BeanPropertyRowMapper<Employee>(Employee.class));
    }

    public List<Employee> getEmployees() {
        return template.query("select * from Empployee", new RowMapper<Employee>() {
            public Employee mapRow(ResultSet rs, int row) throws SQLException {
                Employee e = new Employee();
                e.setId(rs.getInt(1));
                e.setName(rs.getString(2));
                e.setSalary(rs.getFloat(3));
                e.setDesignation(rs.getString(4));
                return e;
            }
        });
    }
}

```

Lưu ý tra cứu Interface RowMapper.

## B7. Lớp xử lý (Controller)

```
package se.iuh.edu.vn.controller;

import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;
import se.iuh.edu.vn.beans.Employee;
import se.iuh.edu.vn.dao.EmployeeDao;

@Controller
public class EmployeeController {
    @Autowired
    EmployeeDao dao; // sẽ inject dao khai báo trong file spring-servlet.xml

    /* Hiển thị form để nhập dữ liệu, lệnh "command" là reserved request attribute
     * dùng để hiển thị đối tượng dữ liệu vào form */
    @RequestMapping("/empform")
    public ModelAndView showform() {
        return new ModelAndView("empform", "command", new Employee());
    }

    /* Lưu đối tượng vào CSDL. @ModelAttribute dùng để đưa dữ liệu vào đối
     * tượng model. Lưu ý cần chỉ rõ phương thức truyền là POST: RequestMethod.POST,
     * mặc định GET request */
    @RequestMapping(value = "/save", method = RequestMethod.POST)
    public ModelAndView save(@ModelAttribute("emp") Employee emp) {
        dao.save(emp);
        // sẽ chuyển hướng sang viewemp request mapping
        return new ModelAndView("redirect:/viewemp");
    }

    /* Trả về danh sách nhân viên */
    @RequestMapping("/viewemp")
    public ModelAndView viewemp() {
        List<Employee> list = dao.getEmployees();
        return new ModelAndView("viewemp", "list", list);
    }

    /* Hiển thị đối tượng dữ liệu vào form với id cụ thể. @PathVariable đưa
     * dữ liệu URL vào biến. */
    @RequestMapping(value = "/editemp/{id}")
    public ModelAndView edit(@PathVariable int id) {
        Employee emp = dao.getEmpById(id);
        return new ModelAndView("empeditform", "command", emp);
    }

    /* Cập nhật đối tượng model. */
    @RequestMapping(value = "/editsave", method = RequestMethod.POST)
    public ModelAndView editsave(@ModelAttribute("emp") Employee emp) {
        dao.update(emp);
        return new ModelAndView("redirect:/viewemp");
    }

    /* Xóa mẫu tin với id trên URL và chuyển sang RequestMapping /viewemp */
    @RequestMapping(value = "/deleteemp/{id}", method = RequestMethod.GET)
    public ModelAndView delete(@PathVariable int id) {
        dao.delete(id);
        return new ModelAndView("redirect:/viewemp");
    }
}
```

B8. Các giao diện xử lý xem danh sách, thêm nhân viên mới, cập nhật nhân viên đã tồn tại trong CSDL.  
*viewemp.jsp*

```
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<h1>Employees List</h1>
<table border="2" width="70%" cellpadding="2">
    <tr>
        <th>Id</th>
        <th>Name</th>
        <th>Salary</th>
        <th>Designation</th>
        <th>Edit</th>
        <th>Delete</th>
    </tr>
    <c:forEach var="emp" items="${list}">
        <tr>
            <td>${emp.id}</td>
            <td>${emp.name}</td>
            <td>${emp.salary}</td>
            <td>${emp.designation}</td>
            <td><a href="editemp/${emp.id}">Edit</a></td>
            <td><a href="deleteemp/${emp.id}">Delete</a></td>
        </tr>
    </c:forEach>
</table>
<br />
<a href="empform">Add New Employee</a>
```

*empform.jsp*

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

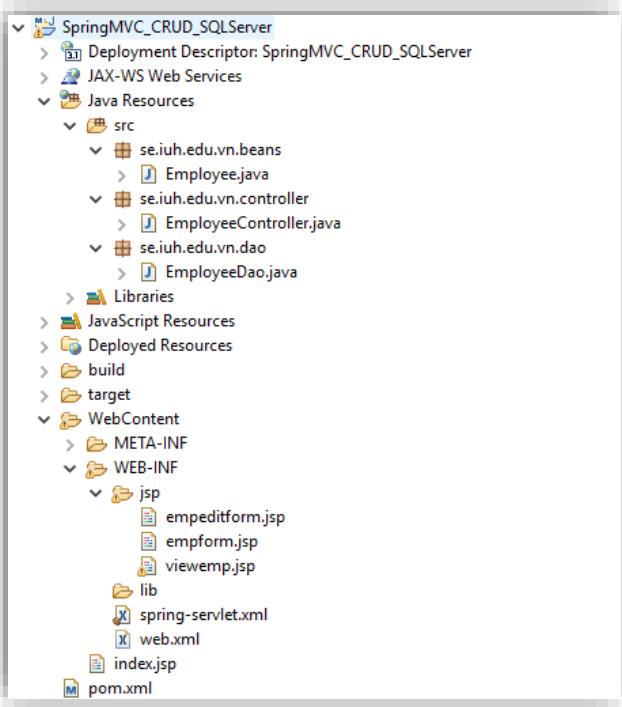
<form:form method="post" action="save">
    <table>
        <tr>
            <td>Name :</td>
            <td><form:input path="name" /></td>
        </tr>
        <tr>
            <td>Salary :</td>
            <td><form:input path="salary" /></td>
        </tr>
        <tr>
            <td>Designation :</td>
            <td><form:input path="designation" /></td>
        </tr>
        <tr>
            <td colspan="2"><input type="submit" value="Save" /></td>
        </tr>
    </table>
</form:form>
```

## *empeditform.jsp*

```
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<h1>Edit Employee</h1>
<form:form method="POST" action="/SpringMVC_CRUD_SQLServer/editsave">
    <table>
        <tr>
            <td></td>
            <td><form:hidden path="id" /></td>
        </tr>
        <tr>
            <td>Name :</td>
            <td><form:input path="name" /></td>
        </tr>
        <tr>
            <td>Salary :</td>
            <td><form:input path="salary" /></td>
        </tr>
        <tr>
            <td>Designation :</td>
            <td><form:input path="designation" /></td>
        </tr>
        <tr>
            <td></td>
            <td><input type="submit" value="Edit Save" /></td>
        </tr>
    </table>
</form:form>
```

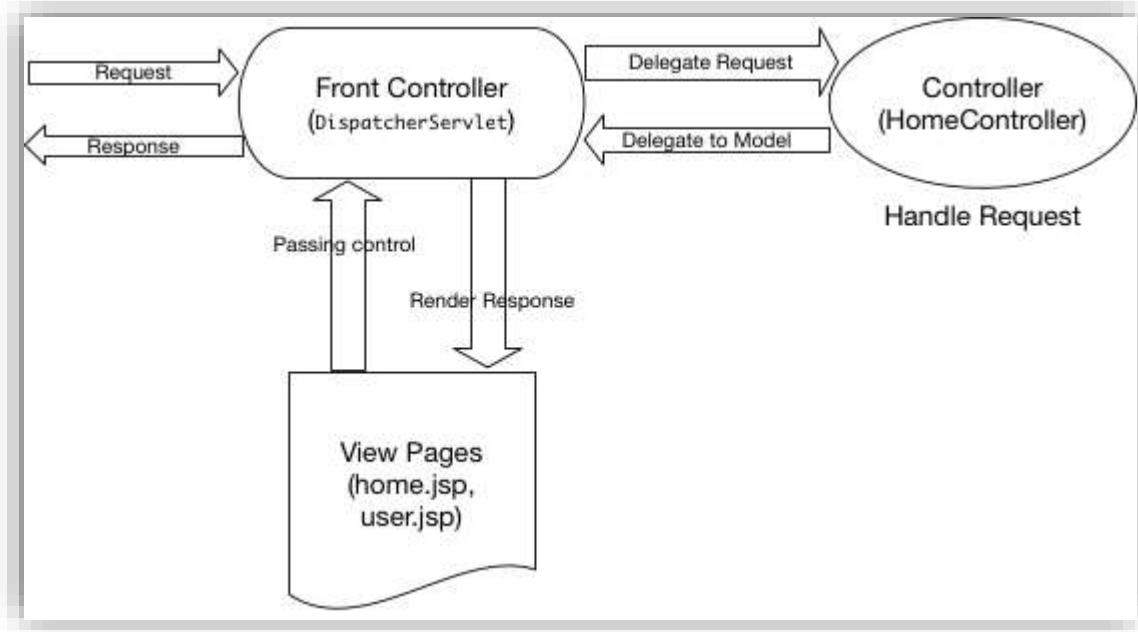
## *Phần tham khảo project cấu trúc:*



## Bài 7. Spring MVC - Dùng Spring Tool Suite - Hello Word

Thực hiện chương trình HelloWorld dùng **Spring Tool Suite (STS 3.9.2.RELEASE)**

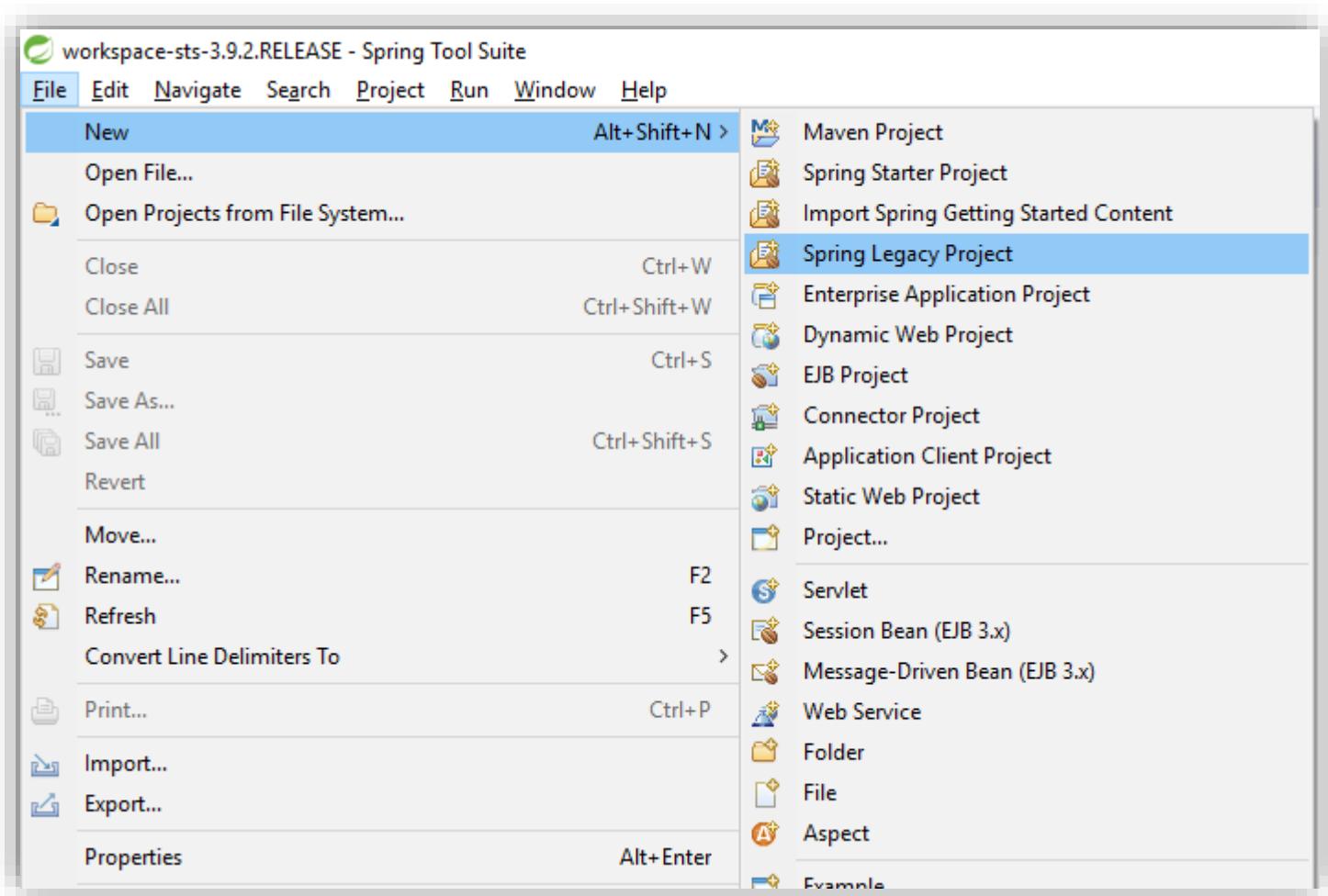
### B1. Tạo Project Spring MVC

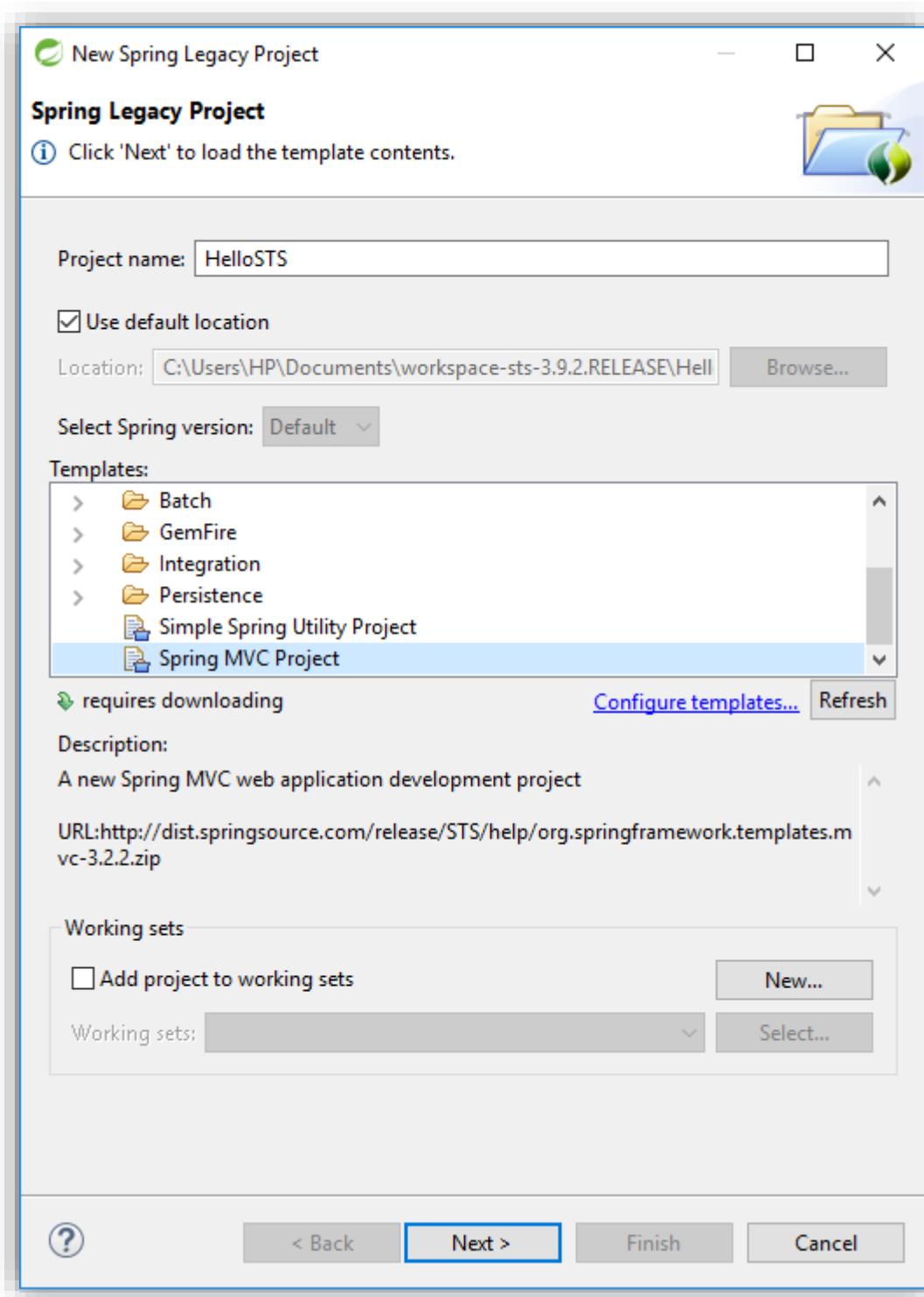


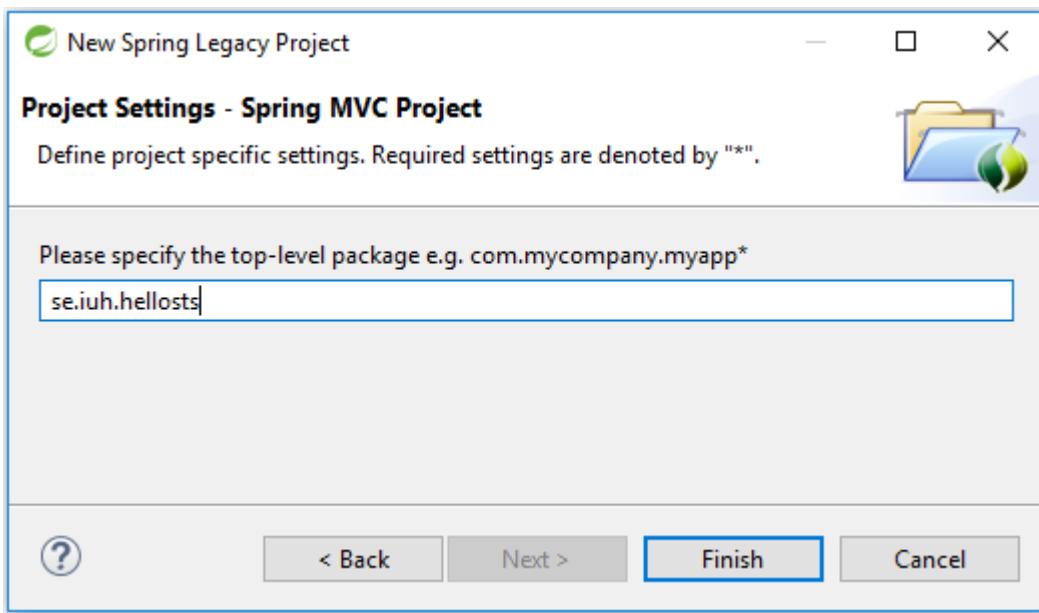
Trong đó:

- DispatcherServlet là lớp front controller nhận tất cả các yêu cầu và bắt đầu xử lý các yêu cầu đó. Cần cấu hình trong tập tin web.xml. Front controller có nhiệm vụ chuyển yêu cầu đến các lớp Controller cần thiết, và gửi lại phản hồi khi các trang View được phát sinh.
- HomeController.java là lớp Controller trong ứng dụng Spring MVC.
- home.jsp, user.jsp là các View trong ứng dụng Spring MVC.
- User.java là lớp Model trong ứng dụng Spring MVC.

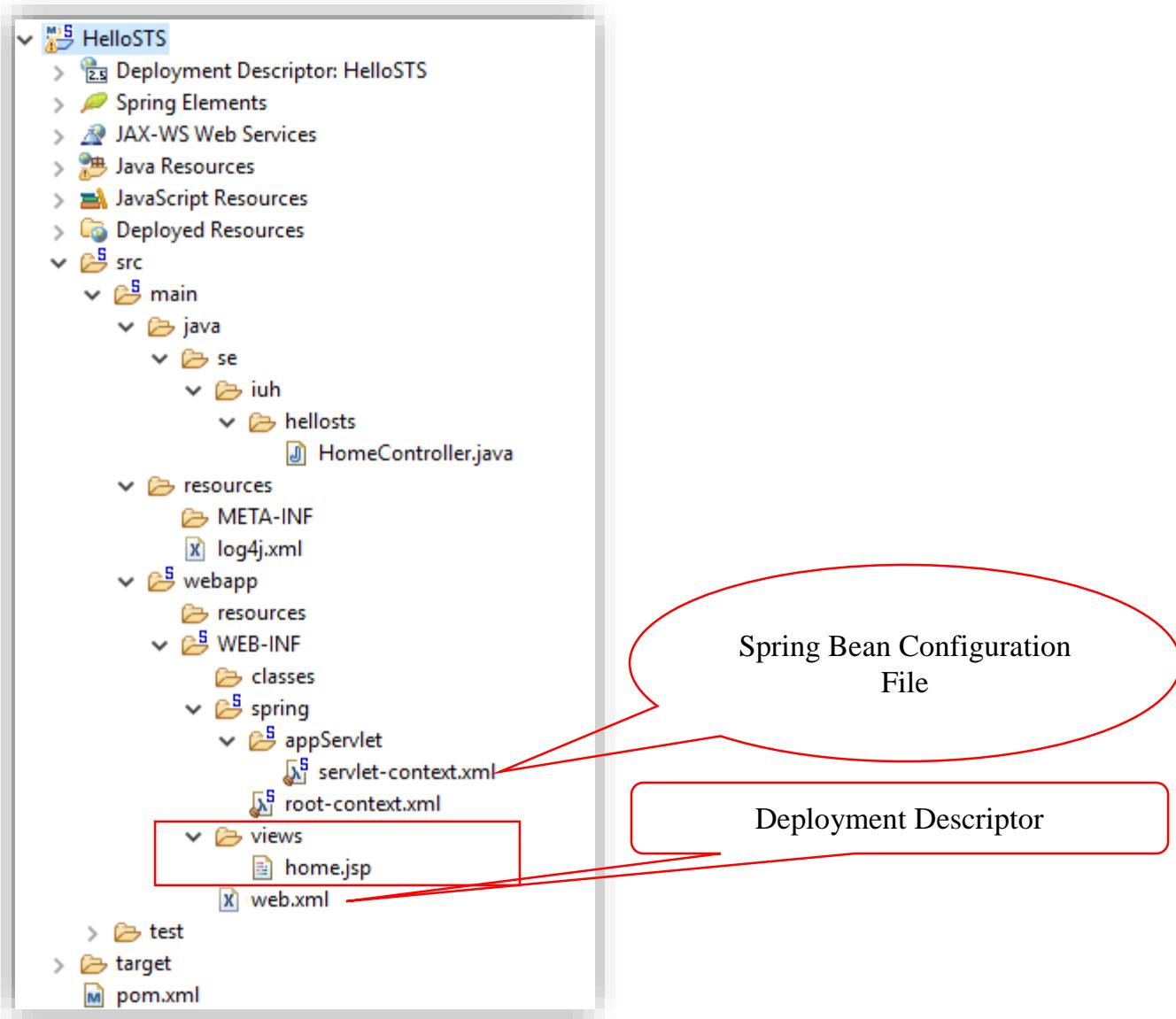
### B1. Tạo Spring MVC Project







B2. Sau khi tạo, Project HelloSTS có cấu trúc tương tự như sau:



Trong đó

- Thư mục *src/main/webapp/WEB-INF/spring* chứa tất cả các cấu hình của Spring.
- Thư mục *src/main/webapp/WEB-INF/views*. Các tập tin jsp để trong này.

B3. Chính sửa version phù hợp cho file *pom.xml*

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>se.iuh</groupId>
  <artifactId>hellosts</artifactId>
  <name>HelloSTS</name>
```

```

<packaging>war</packaging>
<version>1.0.0-BUILD-SNAPSHOT</version>
<properties>
    <java-version>1.9</java-version>
    <org.springframework-version>5.0.3.RELEASE</org.springframework-version>
    <org.aspectj-version>1.8.13</org.aspectj-version>
    <org.slf4j-version>1.7.25</org.slf4j-version>
</properties>
<dependencies>
    <!-- Spring -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>${org.springframework-version}</version>
        <exclusions>
            <!-- Exclude Commons Logging in favor of SLF4j -->
            <exclusion>
                <groupId>commons-logging</groupId>
                <artifactId>commons-logging</artifactId>
            </exclusion>
        </exclusions>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>${org.springframework-version}</version>
    </dependency>

    <!-- AspectJ -->
    <dependency>
        <groupId>org.aspectj</groupId>
        <artifactId>aspectjrt</artifactId>
        <version>${org.aspectj-version}</version>
    </dependency>

    <!-- Logging -->
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-api</artifactId>
        <version>${org.slf4j-version}</version>
    </dependency>
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>jcl-over-slf4j</artifactId>
        <version>${org.slf4j-version}</version>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-log4j12</artifactId>
        <version>${org.slf4j-version}</version>
        <scope>runtime</scope>
    </dependency>

```

```

        </dependency>
<dependency>
    <groupId>log4j</groupId>
    <artifactId>log4j</artifactId>
    <version>1.2.15</version>
    <exclusions>
        <exclusion>
            <groupId>javax.mail</groupId>
            <artifactId>mail</artifactId>
        </exclusion>
        <exclusion>
            <groupId>javax.jms</groupId>
            <artifactId>jms</artifactId>
        </exclusion>
        <exclusion>
            <groupId>com.sun.jdmk</groupId>
            <artifactId>jmxtools</artifactId>
        </exclusion>
        <exclusion>
            <groupId>com.sun.jmx</groupId>
            <artifactId>jmxri</artifactId>
        </exclusion>
    </exclusions>
    <scope>runtime</scope>
</dependency>

<!-- @Inject -->
<dependency>
    <groupId>javax.inject</groupId>
    <artifactId>javax.inject</artifactId>
    <version>1</version>
</dependency>

<!-- Servlet -->
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId> servlet-api</artifactId>
    <version>2.5</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>javax.servlet.jsp</groupId>
    <artifactId> jsp-api</artifactId>
    <version>2.1</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId> jstl</artifactId>
    <version>1.2</version>
</dependency>

```

```

<!-- Test -->
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.7</version>
    <scope>test</scope>
</dependency>
</dependencies>
<build>
    <plugins>
        <plugin>
            <artifactId>maven-eclipse-plugin</artifactId>
            <version>2.9</version>
            <configuration>
                <additionalProjectnatures>

<projectnature>org.springframework.ide.eclipse.core.springnature</projectnature>
                </additionalProjectnatures>
                <additionalBuildcommands>

<buildcommand>org.springframework.ide.eclipse.core.springbuilder</buildcommand>
                </additionalBuildcommands>
                <downloadSources>true</downloadSources>
                <downloadJavadocs>true</downloadJavadocs>
            </configuration>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>2.5.1</version>
            <configuration>
                <source>1.6</source>
                <target>1.6</target>
                <compilerArgument>-Xlint:all</compilerArgument>
                <showWarnings>true</showWarnings>
                <showDeprecation>true</showDeprecation>
            </configuration>
        </plugin>
        <plugin>
            <groupId>org.codehaus.mojo</groupId>
            <artifactId>exec-maven-plugin</artifactId>
            <version>1.2.1</version>
            <configuration>
                <mainClass>org.test.int1.Main</mainClass>
            </configuration>
        </plugin>
    </plugins>
</build>
</project>

```



#### B4. servlet-context.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/mvc"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:beans="http://www.springframework.org/schema/beans"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/mvc
    http://www.springframework.org/schema/mvc/spring-mvc.xsd
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd">

    <!-- DispatcherServlet Context: defines this servlet's request-processing
    infrastructure -->

    <!-- Enables the Spring MVC @Controller programming model -->
    <annotation-driven />

    <!-- Handles HTTP GET requests for /resources/** by efficiently serving up static
    resources in the ${webappRoot}/resources directory -->
    <resources mapping="/resources/**" location="/resources/" />

    <!-- Resolves views selected for rendering by @Controllers to .jsp resources in the
    /WEB-INF/views directory -->
    <beans:bean
        class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <beans:property name="prefix" value="/WEB-INF/views/" />

```

```

        <beans:property name="suffix" value=".jsp" />
    </beans:bean>

    <context:component-scan base-package="se.iuh.hellosts" />

</beans:beans>

```

## B5. web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">

    <!-- The definition of the Root Spring Container shared by all Servlets and Filters
-->
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/spring/root-context.xml</param-value>
    </context-param>

    <!-- Creates the Spring Container shared by all Servlets and Filters -->
    <listener>
        <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>

    <!-- Processes application requests -->
    <servlet>
        <servlet-name>appServlet</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-
value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>appServlet</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>

</web-app>

```

## B6. Thêm vào HomeController

- Một Controller cần có annotation **@Controller** khi khai báo lớp **XXXController (HomeController)**.
- Annotation **@RequestMapping** được khai báo cùng với định nghĩa của phương thức **home()**, **loginPage()**, **login()**. Trong annotation, thuộc tính **value** định nghĩa request URL và thuộc tính **method** để định nghĩa HTTP request method.
- Phương thức **home()** trong lớp **HomeController** đang trả về một chuỗi “**home**”, phương thức **loginPage()** trả về chuỗi “**login**”, phương thức **login()** trả về chuỗi “**user**”, các chuỗi này định nghĩa tên View sẽ được dùng để hiển thị kết quả của request cho người dùng.
- Trong phương thức **home()** có biến **model** lưu trữ những dữ liệu cần thiết cho view “**home**” sử dụng, phương thức **login()** có biến **model** User.

```
package se.iuh.hellosts;

import java.text.DateFormat;
import java.util.Date;
import java.util.Locale;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import se.iuh.hellosts.User;

/**
 * Handles requests for the application home page.
 */
@Controller
public class HomeController {

    private static final Logger logger = LoggerFactory.getLogger(HomeController.class);

    /**
     * Simply selects the home view to render by returning its name.
     */
    @RequestMapping(value = "/", method = RequestMethod.GET)
    public String home(Locale locale, Model model) {
        logger.info("Welcome home! The client locale is {}.", locale);

        Date date = new Date();
        DateFormat dateFormat = DateFormat.getDateInstance(DateFormat.LONG,
        DateFormat.LONG, locale);

        String formattedDate = dateFormat.format(date);

        model.addAttribute("serverTime", formattedDate);
    }
}
```

```

        return "home";
    }

    @RequestMapping(value = "/login", method = RequestMethod.GET)
    public String loginPage(Locale locale, Model model) {
        return "login";
    }

    @RequestMapping(value = "/home", method = RequestMethod.POST)
    public String login(@Validated User user, Model model) {
        model.addAttribute("userName", user.getUserName());
        return "user";
    }
}

```

## B7. Các View:

*home.jsp*

```

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ page session="false"%>
<html>
<head>
<title>Home</title>
</head>
<body>
    <h1>Hello world!</h1>
    <P>The time on the server is ${serverTime}.</P>
</body>
</html>

```

*login.jsp*

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
       pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Login Page</title>
</head>
<body>
    <form action="home" method="post">
        Please enter your name: <input type="text" name="userName"><br>
        <input type="submit" value="Send">
    </form>
</body>
</html>

```

*user.jsp*

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
       pageEncoding="UTF-8"%>

```

```

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>User Home Page</title>
</head>
<body>
    <h3>Hi ${userName}</h3>
</body>
</html>

```

## Bài 8. Spring MVC - Upload file

Thực hiện ứng dụng Spring MVC Upload File

### Tham khảo hướng dẫn:

B1. Khai báo Maven pom.xml:

Ngoài những thư viện của Spring, khai báo thêm các dependencies:

```

<!-- https://mvnrepository.com/artifact/jstl/jstl -->
<dependency>
    <groupId>jstl</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
</dependency>
<!-- https://mvnrepository.com/artifact/commons-io/commons-io -->
<dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>2.6</version>
</dependency>
<!-- https://mvnrepository.com/artifact/commons-fileupload/commons-fileupload -->
<dependency>
    <groupId>commons-fileupload</groupId>
    <artifactId>commons-fileupload</artifactId>
    <version>1.3.3</version>
</dependency>

```

B2. Cấu hình Spring spring-servlet.xml

Ngoài các khai báo Khai báo package chứa các thành phần xử lý Controller, khai báo bean ViewResolver, khai báo thêm bean org.springframework.web.multipart.commons.CommonsMultipartResolver:

```

<bean id="multipartResolver"
      class="org.springframework.web.multipart.commons.CommonsMultipartResolver" />

```

B3. Cấu hình Project web.xml

Khai báo FrontController: DispatcherServlet

#### B4. Tạo UploadController

```
package se.iuh.edu.vn.controller;

import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import javax.servlet.ServletContext;

import javax.servlet.http.HttpSession;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.multipart.commons.CommonsMultipartFile;
import org.springframework.web.servlet.ModelAndView;

@Controller
public class UploadController {
    private static final String UPLOAD_DIRECTORY = "/images";

    @RequestMapping("uploadform")
    public ModelAndView uploadForm() {
        return new ModelAndView("uploadform");
    }

    @RequestMapping(value = "savefile", method = RequestMethod.POST)
    public ModelAndView saveimage(@RequestParam CommonsMultipartFile file,
                                  HttpSession session) throws Exception {

        ServletContext context = session.getServletContext();
        String path = context.getRealPath(UPLOAD_DIRECTORY);
        String filename = file.getOriginalFilename();

        System.out.println(path + File.separator + filename);

        byte[] bytes = file.getBytes();
        BufferedOutputStream stream = new BufferedOutputStream(
            new FileOutputStream(new File(path + File.separator + filename)));
        stream.write(bytes);
        stream.flush();
        stream.close();

        return new ModelAndView("uploadform", "filesuccess",
                               "File successfully saved!");
    }
}
```

#### B5. Tạo View

```

<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>

<!DOCTYPE html>
<html>
<head>
<title>Image File Upload</title>
<style>
    #dvPreview1 {
        height: 100px;
        width: 100px;
        color:blue;
    }
</style>
<script src="jquery-1.10.2.js"></script>

<script>
    //đoạn script hiển thị hình khi người dùng chọn file hình
    $(function () {
        $("#fileToUpload").change(function () {
            //kiểm tra xem có phải hình không? dùng RegularExpression
            var regex = /^[a-zA-Z0-9\s\.\.\-:]+(\.jpg|\.jpeg|\.gif|\.png|\.bmp)$/;
            if (regex.test($("#this").val().toLowerCase())) {
                if (typeof (FileReader) != "undefined") {
                    var reader = new FileReader();
                    reader.onload = function (e) {
                        $("#dvPreview1").attr("src", e.target.result);
                    }
                    reader.readAsDataURL($("#this")[0].files[0]);
                } else {
                    alert("This browser does not support FileReader.");
                }
            } else {
                alert("Please upload a valid image file.");
            }
        });
    });
</script>
</head>
<body>
    <h1>File Upload Example</h1>

    <h3 style="color: red">${filesuccess}</h3>
    <form:form method="post" action="savefile"
        enctype="multipart/form-data">
        <p>
            <label for="image">Choose Image</label>
        </p>
        <p>
            <input name="file" id="fileToUpload" type="file" />
        </p>
        <p>
            <input type="submit" value="Upload">
        </p>
    </form:form>
    <div id="#dvPreview1"> </div>
</body>
</html>

```

## Bài 9. Spring MVC - Exception Handling

Thực hiện ứng dụng Spring MVC Exception Handling

- a. Xử lý Exception dùng HttpStatus Code
- b. Xử lý Exception bằng Controller dùng @ExceptionHandler

### a. Dùng @ResponseStatus

B1. Lớp xử lý ngoại lệ:

```
@ResponseStatus(value = HttpStatus.UNAUTHORIZED, reason = "User not logged in") // 401
public class UnauthorizedException extends RuntimeException {
    ...
}

@ResponseStatus(value = HttpStatus.NOT_FOUND, reason = "File not found") // 404
public class FileNotFoundException extends RuntimeException {
    ...
}
```

B2. Cấu hình bổ sung xử lý trang lỗi trong web.xml

```
<error-page>
    <error-code>401</error-code>
    <location>/WEB-INF/error-pages/401.jsp</location>
</error-page>

<error-page>
    <error-code>404</error-code>
    <location>/WEB-INF/error-pages/404.jsp</location>
</error-page>
```

B3. File Controller

```
@RequestMapping(value = "/unauthorizedException")
public String testUnauthorizedException() throws IOException {
    throw new UnauthorizedException(); // chuyển sang xử lý lỗi 401
}

@RequestMapping(value = "/fileNotFoundException")
public String testFileNotFoundException() throws IOException {
    throw new FileNotFoundException(); // chuyển sang xử lý lỗi 404
}
```

B4. Tạo trang xử lý lỗi /WEB-INF/error-pages/401.jsp

### b. Xử lý Exception bằng Controller dùng @ExceptionHandler

```
import java.io.IOException;

import org.springframework.validation.BindException;
import org.springframework.validation.ObjectError;
```

```

import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.servlet.ModelAndView;

@ControllerAdvice
public class GlobalExceptionHandling {

    @ExceptionHandler(org.springframework.validation.BindException.class)
    private ModelAndView processInvalidData(BindException ex) {
        StringBuilder error = new StringBuilder();
        for (ObjectError objectError: ex.getAllErrors()) {
            error.append(objectError.getDefaultMessage());
            error.append("<br/>");
        }
        ModelAndView model = new ModelAndView("error");
        model.addObject("error", error);
        return model;
    }

    @ExceptionHandler(IOException.class)
    private ModelAndView processIOException(IOException ex) {
        ModelAndView model = new ModelAndView("error");
        model.addObject("error", ex.getMessage());
        return model;
    }

}

```

## Bài 10. Spring MVC - Form Validation

Thực hiện ứng dụng Spring MVC Form Validator

The screenshot shows a form with five input fields and their corresponding validation messages:

- Id :** [Empty] **Id can not be null**
- Name :** [Empty] **length must be between 5 and 10**
- Email :** **van** **Email is wrong format**
- Phone Number :** [Empty] **Phone number is not valid**
- Date Of Birth :** [Empty] **Date of birth can not be null**

Below the form is a **Submit** button.

Dùng STS tạo Spring MVC Project với các cấu hình mặc định.

B1. File cấu hình các thư viện/dependencies *pom.xml*.

Ngoài các thư viện mặc định phát sinh bởi STS, cần khai báo thêm các dependencies:

```
<!-- https://mvnrepository.com/artifact/org.hibernate.validator/hibernate-validator -->
<dependency>
    <groupId>org.hibernate.validator</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>6.0.7.Final</version>
</dependency>
```

Một số trường hợp cần khai báo thêm thư viện Joda-time, cho phép thao tác với kiểu date nhiều và phức tạp

```
<!-- https://mvnrepository.com/artifact/joda-time/joda-time -->
<dependency>
    <groupId>joda-time</groupId>
    <artifactId>joda-time</artifactId>
    <version>2.9.9</version>
</dependency>
```

B2. File cấu hình Project *web.xml* (Khai báo FrontController).

B3. Tạo lớp chứa các thông điệp báo lỗi dữ liệu trên form message\_en.properties  
*/src/main/webapp/i18n/message\_en.properties*

```
NotNull.user.id=Id can not be null
NotBlank.user.name=Name can not be empty
NotBlank.user.email=Email can not be null
NotNull.user.dateOfBirth=Date of birth can not be null
Past.user.dateOfBirth=Must be in the past
Email.user.email=Email is wrong format
Phone=Phone number is not valid
```

B4. File cấu hình liên quan đến Spring */WEB-INF/spring/appServlet/servlet-context.xml*.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/mvc"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:beans="http://www.springframework.org/schema/beans"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/mvc
    http://www.springframework.org/schema/mvc/spring-mvc.xsd
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd">

    <!-- DispatcherServlet Context: defines this servlet's request-processing
    infrastructure -->

    <!-- Enables the Spring MVC @Controller programming model -->
    <annotation-driven />
```

```

<!-- Handles HTTP GET requests for /resources/** by efficiently serving up static
resources in the ${webappRoot}/resources directory -->
<resources mapping="/resources/**" location="/resources/" />

<!-- Resolves views selected for rendering by @Controllers to .jsp resources in the
/WEB-INF/views directory -->
<beans:bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <beans:property name="prefix" value="/WEB-INF/views/" />
    <beans:property name="suffix" value=".jsp" />
</beans:bean>

<context:component-scan base-package="se.iuh.formvalidator" />

<beans:bean id="messageSource"
class="org.springframework.context.support.ReloadableResourceBundleMessageSource">
    <beans:property name="basename" value="/i18n/message" />
</beans:bean>
</beans:beans>

```

## B5. Tạo lớp kiểm tra (kiểm tra số điện thoại – user-defined validator)

```

package se.iuh.formvalidator.validator;

import static java.lang.annotation.ElementType.FIELD;
import static java.lang.annotation.ElementType.METHOD;
import static java.lang.annotation.RetentionPolicy.RUNTIME;

import java.lang.annotation.Documented;
import java.lang.annotation.Retention;
import java.lang.annotation.Target;

import javax.validation.Constraint;
import javax.validation.Payload;

@Documented
@Constraint(validatedBy = PhoneValidator.class)
@Retention(RUNTIME)
@Target({ FIELD, METHOD })
public @interface Phone {
    String message() default "{Phone}";

    Class<?>[] groups() default {};

    Class<? extends Payload>[] payload() default {};
}

```

```

package se.iuh.formvalidator.validator;

import javax.validation.ConstraintValidator;
import javax.validation.ConstraintValidatorContext;

public class PhoneValidator implements ConstraintValidator<Phone, String> {

    public void initialize(Phone paramA) {
    }

    public boolean isValid(String phoneNo, ConstraintValidatorContext ctx) {
        if (phoneNo == null) {
            return false;
        }
        return phoneNo.matches("\\d{10}");
    }

}

```

## B6. Tạo Model với các ràng buộc kiểm tra dữ liệu (Hibernate).

```

package se.iuh.formvalidator.entities;

import java.util.Date;

import javax.validation.constraints.NotNull;
import javax.validation.constraints.Past;

import org.hibernate.validator.constraints.Email;
import org.hibernate.validator.constraints.Length;
import org.hibernate.validator.constraints.NotBlank;
import org.springframework.format.annotation.DateTimeFormat;

import se.iuh.formvalidator.validator.Phone;

public class User {

    @NotNull(message = "Id may not be null")
    private Integer id;

    @NotNull
    @Length(min = 5, max = 10)
    private String name;

    @NotNull
    @Email
    private String email;

    @NotNull
    @DateTimeFormat(pattern = "dd/MM/yyyy")
    @Past
    private Date dateOfBirth;
}

```

```

@Phone(message = "Phone Number is invalid")
private String phoneNumber;

public Integer getId() {
    return id;
}

public void setId(Integer id) {
    this.id = id;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public Date getDateOfBirth() {
    return dateOfBirth;
}

public void setDateOfBirth(Date dateOfBirth) {
    this.dateOfBirth = dateOfBirth;
}

public String getPhoneNumber() {
    return phoneNumber;
}

public void setPhoneNumber(String phoneNumber) {
    this.phoneNumber = phoneNumber;
}
}

```

## B7. Tạo Controller

```

package se.iuh.formvalidator.controller;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.validation.ObjectError;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotationModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import se.iuh.formvalidator.entities.User;

@Controller
public class UserController {

    @RequestMapping(value = "/addUser", method = RequestMethod.GET)
    public String doGetAddUser(Model model) {
        if (!model.containsAttribute("user")) {
            model.addAttribute("user", new User());
        }
        return "add-user";
    }

    @RequestMapping(value = "/addUser", method = RequestMethod.POST)
    public String doPostAddUser(@ModelAttribute("user") @Validated User user,
                                BindingResult result) {
        if (result.hasErrors()) {
            for (ObjectError objectError: result.getAllErrors()) {
                System.out.println(objectError);
                System.out.println(objectError.getCode());
            }
            return "add-user";
        }
        return "view-user";
    }
}

```

## B8. Tạo View

```

<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>
<html>
<head>
<style>
.error {
    color: red;
}
</style>
</head>

<body>
    <h2>Add New User</h2>
    <form:form action="addUser" method="POST" modelAttribute="user">
        Id: <form:input path="id" />
        <form:errors path="id" cssClass="error" />
        <br />
        <br />
        Name: <form:input path="name" />
        <form:errors path="name" cssClass="error" />
        <br />
        <br />
        Email: <form:input path="email" />
        <form:errors path="email" cssClass="error" />
        <br />
        <br />
        Phone Number: <form:input path="phoneNumber" />
        <form:errors path="phoneNumber" cssClass="error" />
        <br />
        <br />
        Date Of Birth: <form:input path="dateOfBirth" />
        <form:errors path="dateOfBirth" cssClass="error" />
        <br />
        <button type="submit">Submit</button>
    </form:form>
</body>
</html>

```

```

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
<html>
<head>
<title>Form Validation</title>
</head>

<body>
    <h2>User information</h2>
    Id: ${user.id}
    <br /> Name: ${user.name}
    <br /> Email: ${user.email}
    <br /> Phone Number: ${user.phoneNumber}
    <br /> Date Of Birth:
    <fmt:formatDate pattern="dd/MM/yyyy" value="${user.dateOfBirth}" />
</body>
</html>
</html>

```

## Bài 11. Spring MVC - Đa ngôn ngữ

Thực hiện ứng dụng Spring MVC đa ngôn ngữ - Internationalization and Localization

- **Internationalization** (i18n) (Toàn cầu hóa), **Localization** (L10n) (Địa phương hóa hóa)
- Spring Framework hỗ trợ cho việc thiết kế website đa ngôn ngữ (i18n) dùng **Spring Interceptors, Locale Resolvers** và **Resource Bundles**.

Thực hiện trang Web đa ngôn ngữ (tiếng Anh, tiếng Pháp, tiếng Nhật và tiếng Việt) gồm trang đăng nhập như sau:

### Login Page

Language : [English](#) [Viet Nam](#) [Japanese](#) [France](#)

Username

Password

### Trang đăng nhập

Ngôn ngữ: [English](#) [Vietnamese](#) [Japanese](#) [France](#)

Tên người dùng

Mật khẩu

### ログインページ

Language: [English](#) [Vietnamese](#) [Japanese](#) [France](#)

ユーザー名

パスワード

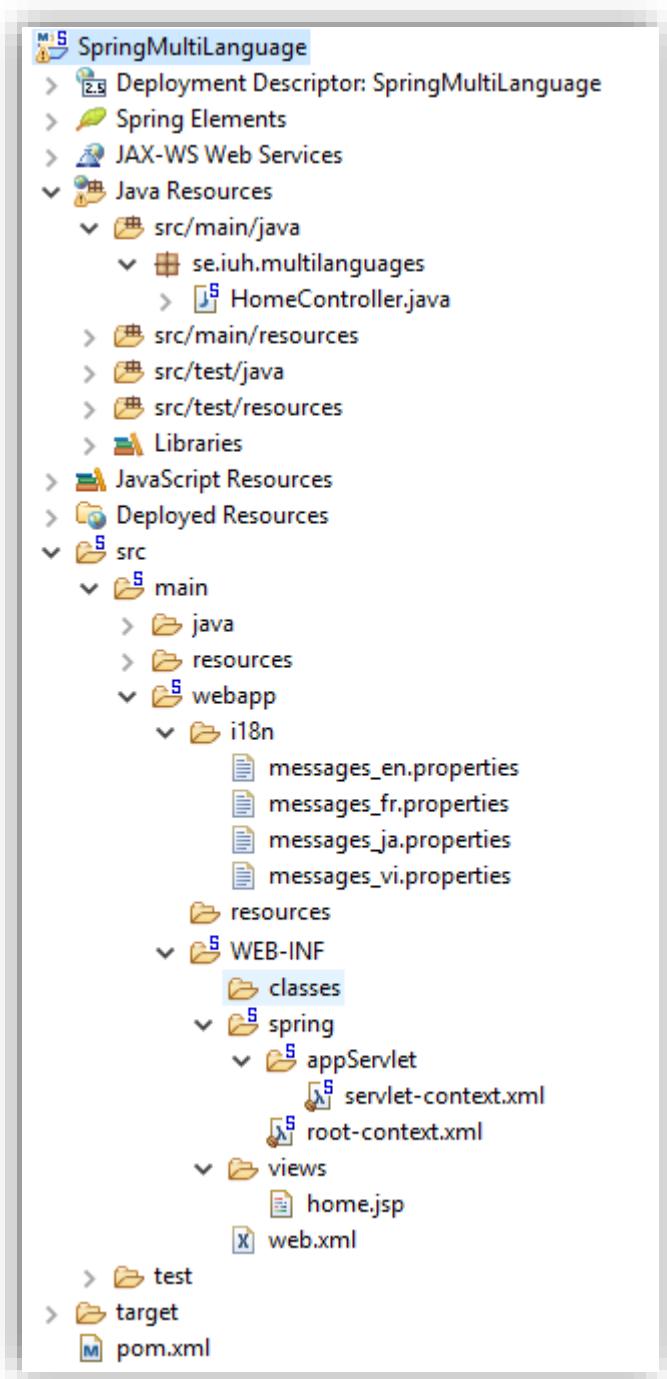
### Page de connexion

La langue: [English](#) [Vietnamese](#) [Japanese](#) [France](#)

Nom d'utilisateur

Mot de passe

## Tham khảo hướng dẫn:



B1. Dùng STS tạo Spring MVC Project

B2. Tạo các tập tin properties cho từng ngôn ngữ (en, fr, vi, ja)

Tạo thư mục lưu trữ file properties của các ngôn ngữ trong WEB-INF/i18n

**messages\_en.properties**

label.login.title=Login Page

label.login.username=Username

label.login.password=Password  
label.login.submit=Login  
label.login.language=Language

#### **messages\_vn.properties**

label.login.title=Thông tin đăng nhập  
label.login.username=Tên tài khoản  
label.login.password=Mật khẩu  
label.login.submit=Đăng nhập  
label.login.language=Ngôn ngữ

#### **messages\_fr.properties**

label.login.title= Page de connexion  
label.login.username= Nom d'utilisateur  
label.login.password= Mot de passe  
label.login.submit= Entrer  
label.login.language=La Langue

#### **messages\_fr.properties**

label.login.title= ログインページ  
label.login.username= ユーザー名  
label.login.password= パスワード  
label.login.submit= ログイン  
label.login.language=Language

B3. File cấu hình dependencies Maven: *pom.xml* (*mặc định phát sinh, chỉnh sửa version của Java và Spring*)

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>se.iuh</groupId>
  <artifactId>multilanguages</artifactId>
  <name>SpringMultiLanguage</name>
  <packaging>war</packaging>
  <version>1.0.0-BUILD-SNAPSHOT</version>
  <properties>
    <java-version>1.9</java-version>
    <org.springframework-version>5.0.3.RELEASE</org.springframework-version>
    <org.aspectj-version>1.8.13</org.aspectj-version>
    <org.slf4j-version>1.7.25</org.slf4j-version>
  </properties>
  <dependencies>
    <!-- Spring -->
```

```

<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>${org.springframework-version}</version>
    <exclusions>
        <!-- Exclude Commons Logging in favor of SLF4j -->
        <exclusion>
            <groupId>commons-logging</groupId>
            <artifactId>commons-logging</artifactId>
        </exclusion>
    </exclusions>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>${org.springframework-version}</version>
</dependency>

<!-- AspectJ -->
<dependency>
    <groupId>org.aspectj</groupId>
    <artifactId>aspectjrt</artifactId>
    <version>${org.aspectj-version}</version>
</dependency>

<!-- Logging -->
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>${org.slf4j-version}</version>
</dependency>
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>jcl-over-slf4j</artifactId>
    <version>${org.slf4j-version}</version>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-log4j12</artifactId>
    <version>${org.slf4j-version}</version>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>log4j</groupId>
    <artifactId>log4j</artifactId>
    <version>1.2.15</version>
    <exclusions>
        <exclusion>
            <groupId>javax.mail</groupId>
            <artifactId>mail</artifactId>
        </exclusion>
    </exclusions>
</dependency>

```

```

<exclusion>
    <groupId>javax.jms</groupId>
    <artifactId>jms</artifactId>
</exclusion>
<exclusion>
    <groupId>com.sun.jdmk</groupId>
    <artifactId>jmxtools</artifactId>
</exclusion>
<exclusion>
    <groupId>com.sun.jmx</groupId>
    <artifactId>jmxri</artifactId>
</exclusion>
</exclusions>
<scope>runtime</scope>
</dependency>

<!-- @Inject -->
<dependency>
    <groupId>javax.inject</groupId>
    <artifactId>javax.inject</artifactId>
    <version>1</version>
</dependency>

<!-- Servlet -->
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>servlet-api</artifactId>
    <version>2.5</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>javax.servlet.jsp</groupId>
    <artifactId>jsp-api</artifactId>
    <version>2.1</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
</dependency>

<!-- Test -->
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.7</version>
    <scope>test</scope>
</dependency>
</dependencies>
<build>
    <plugins>

```

```

<plugin>
    <artifactId>maven-eclipse-plugin</artifactId>
    <version>2.9</version>
    <configuration>
        <additionalProjectnatures>

<projectnature>org.springframework.ide.eclipse.core.springnature</projectnature>
        </additionalProjectnatures>
        <additionalBuildcommands>

<buildcommand>org.springframework.ide.eclipse.core.springbuilder</buildcommand>
        </additionalBuildcommands>
        <downloadSources>true</downloadSources>
        <downloadJavadocs>true</downloadJavadocs>
    </configuration>
</plugin>
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>2.5.1</version>
    <configuration>
        <source>1.6</source>
        <target>1.6</target>
        <compilerArgument>-Xlint:all</compilerArgument>
        <showWarnings>true</showWarnings>
        <showDeprecation>true</showDeprecation>
    </configuration>
</plugin>
<plugin>
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>exec-maven-plugin</artifactId>
    <version>1.2.1</version>
    <configuration>
        <mainClass>org.test.int1.Main</mainClass>
    </configuration>
</plugin>
</plugins>
</build>
</project>

```

#### B4. File cấu hình Project web.xml (cấu hình FrontController – dùng file phát sinh bởi STS)

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">

    <!-- The definition of the Root Spring Container shared by all Servlets
        and Filters -->
    <context-param>
        <param-name>contextConfigLocation</param-name>

```

```

<param-value>/WEB-INF/spring/root-context.xml</param-value>
</context-param>

<!-- Creates the Spring Container shared by all Servlets and Filters -->
<listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>

<!-- Processes application requests -->
<servlet>
    <servlet-name>appServlet</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-
value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
    <servlet-name>appServlet</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>

<filter>
    <filter-name>encodingFilter</filter-name>
    <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-
class>
    <init-param>
        <param-name>encoding</param-name>
        <param-value>UTF-8</param-value>
    </init-param>
    <init-param>
        <param-name>forceEncoding</param-name>
        <param-value>true</param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>encodingFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
</web-app>

```

## B5. File cấu hình Spring servlet-context.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/mvc"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:beans="http://www.springframework.org/schema/beans"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc.xsd
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd">

    <!-- DispatcherServlet Context: defines this servlet's request-processing
        infrastructure -->

    <!-- Enables the Spring MVC @Controller programming model -->
    <annotation-driven />

    <!-- Handles HTTP GET requests for /resources/** by efficiently serving
        up static resources in the ${webappRoot}/resources directory -->
    <resources mapping="/resources/**" location="/resources/" />

    <!-- Resolves views selected for rendering by @Controllers to .jsr resources
        in the /WEB-INF/views directory -->
    <beans:bean
        class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <beans:property name="prefix" value="/WEB-INF/views/" />
        <beans:property name="suffix" value=".jsr" />
    </beans:bean>

    <context:component-scan base-package="se.iuh.multilanguages" />

    <beans:bean id="messageSource"
        class="org.springframework.context.support.ReloadableResourceBundleMessageSource">
        <beans:property name="basename" value="/i18n/messages" />
        <beans:property name="defaultEncoding" value="UTF-8" />
    <beans:bean id="LocaleResolver"
        class="org.springframework.web.servlet.i18n.CookieLocaleResolver">
        <beans:property name="defaultLocale" value="fr" />
        <beans:property name="cookieName" value="myAppLocaleCookie"/></beans:property>
        <beans:property name="cookieMaxAge" value="3600"/></beans:property>
    </beans:bean>

    <interceptors>
        <beans:bean
            class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor">
            <beans:property name="paramName" value="language" />
        </beans:bean>
    </interceptors>
</beans:beans>

```

## B6. Tạo Controller

```
package se.iuh.multilanguages;  
  
import org.springframework.stereotype.Controller;  
  
@Controller  
public class HomeController {  
  
    @RequestMapping("/")  
    public String home() {  
        return "home";  
    }  
}
```

## B7. Tạo View

```
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>  
<html>  
<body>  
    <h1><spring:message code="label.login.title" text="default text" /></h1>  
  
    <spring:message code="label.login.language" text="default text" />  
    <a href="?language=en">English</a>|  
    <a href="?language=vi_VN">Vietnamese</a>|  
    <a href="?language=ja_JP">Japanese</a>|  
    <a href="?language=fr_FR">France</a>  
    <form>  
        <table>  
            <tr>  
                <td><spring:message code="label.login.username"  
                    text="default text" /></td>  
                <td><input type="text" /></td>  
            </tr>  
            <tr>  
                <td><spring:message code="label.login.password"  
                    text="default text" /></td>  
                <td><input type="text" /></td>  
            </tr>  
            <tr>  
                <td colspan="2"><input type="submit"  
                    value=<spring:message code="label.login.submit"  
                    text="default text" />" /></td>  
            </tr>  
        </table>  
    </form>  
  
    <%-- Current Locale : ${pageContext.response.locale} --%>  
    </body>  
    </html>
```

## Bài 12. Spring MVC - JavaMail API

Gửi mail kèm attach file dùng Spring MVC

- Bước 1: Khai báo thư viện/dependencies
- Bước 2: Các file cấu hình
- Bước 3: Form gửi email AttachEmailInput.jsp
- Bước 4: EmailController.java và EmailInfo.java
- Bước 5: Hiển thị kết quả AttachEmailSuccess.jsp

## Bài 13. Spring MVC - Thao tác với XML và JSON

Trả về dữ liệu XML và JSON

Yêu cầu: Cơ sở dữ liệu quản lý User, viết chương trình Spring MVC trả dữ liệu của bảng User về ở dạng JSON.

### *Trường hợp dữ liệu trả về là JSON*

- Khai báo dependencies hỗ trợ JSON (hoặc thêm trong RequestMapping thuộc tính produces = MediaType.APPLICATION\_JSON\_VALUE):

```
<!-- JSON -->
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.9.4</version>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-core</artifactId>
    <version>2.9.4</version>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-annotations</artifactId>
    <version>2.9.4</version>
</dependency>
```

- Tạo lớp Model

```
package se.iuh.xmlandjson.model;

public class User {
    private int id;
    private String name;
    private String email;

    public User() { }

    public User(int id, String name, String email) {
        this.id = id;
```

```

        this.name = name;
        this.email = email;
    }

    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
}

```

- Phương thức trong Controller khai báo **@ResponseBody** → định dạng đối tượng và trả về dữ liệu dạng JSON

```

@RequestMapping(value = "/jsonvalue")
@ResponseBody
public User returnJSON() { // thay đổi code, kết nối vào CSDL để lấy dữ liệu
    User user = new User(1, "thanhvan", "thanhvan.fit@gmail.com");
    return user;
}

@RequestMapping(value = "/json-list")
@ResponseBody
public List<User> returnListJSON() { // thay đổi code, kết nối vào CSDL để lấy dữ liệu
    List<User> listofUsers = new ArrayList<User>();
    listofUsers.add(new User(1, "thanhvan", "thanhvan.fit@gmail.com"));
    listofUsers.add(new User(2, "vttvan", "vttvan@iu.edu.vn"));
    return listofUsers;
}

```

## Dữ liệu XML

- `@XmlRootElement` định nghĩa tag root của file XML
- `@XmlType(propOrder = { "tt1", "tt2", "tt3", .... })` thứ tự cho các thuộc tính
- `@XmlElement` các element cho file XML

```
package se.iuh.xmldata.model;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

@XmlRootElement(name = "user")
@XmlType(propOrder = { "id", "name", "email" })
public class User {
    private int id;
    private String name;
    private String email;

    public User() {
    }

    public User(int id, String name, String email) {
        this.id = id;
        this.name = name;
        this.email = email;
    }

    public String getName() {
        return name;
    }

    @XmlElement
    public void setName(String name) {
        this.name = name;
    }

    public int getId() {
        return id;
    }

    @XmlElement
    public void setId(int id) {
        this.id = id;
    }

    public String getEmail() {
        return email;
    }
}
```

```

@XmlElement
public void setEmail(String email) {
    this.email = email;
}

}

```

Kết quả file XML:

```

<user>
    <id> ... </id>
    <name> ... </name>
    <email> ... </email>
    ...
    ...
</user>

```

### Bài 1. Thao tác với RESTful Web services

B1. Dùng *pom.xml* mặc định của Project Spring MVC kèm theo dependencies liên quan đến JSON. (Tham khảo dependencies Bài 14). Các cấu hình *web.xml*, *context-servlet.xml* dùng mặc định của Project Spring MVC trong STS.

### B2. Model

```

package se.iuh.restful.model;

public class User {
    private int id;
    private String name;
    private String email;
    private String address;

    public User() {
    }

    public User(int id, String name, String email, String address) {
        this.id = id;
        this.name = name;
        this.email = email;
        this.address = address;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }
}

```

```

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public String getAddress() {
    return address;
}

public void setAddress(String address) {
    this.address = address;
}

}

```

### B3. Tạo Controller thực hiện với CREATE, POST, PUT, DELETE

```

package se.iuh.restful.controller;
import java.util.ArrayList;
import java.util.List;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import se.iuh.restful.model.User;

@RestController
public class UserRestController {

    public static List<User> listUsers = new ArrayList<User>();
    static {
        listUsers.add(new User(1, "Steve", "steve.john@gmail.com", "Texas - USA"));
        listUsers.add(new User(2, "Nick", "nick.belinger@gmail.com", "New York - USA"));
    }
}

```

```

        ListUsers.add(new User(3, "Eric", "eric.hall@gmail.com", "London - England"));
        ListUsers.add(new User(4, "Cherry", "herry@yahoo.com", "Tokyo - Japan"));
        ListUsers.add(new User(4, "Snow", "snow.italia@yahoo.com", "Bangkok - Thailand"));
    }

    // Lấy danh sách người dùng HttpStatus GET
    @RequestMapping(value = "/users", method = RequestMethod.GET)
    public ResponseEntity<List<User>> getAllUser() {
        List<User> listUser = ListUsers;
        return new ResponseEntity<List<User>>(listUser, HttpStatus.OK);
    }

    /// Lấy thông tin người dùng theo id GET
    @RequestMapping(value = "/users/{id}", method = RequestMethod.GET)
    public ResponseEntity<Object> getUserById(@PathVariable int id) {
        User user = ListUsers.get(id);
        if (user != null) {
            return new ResponseEntity<Object>(user, HttpStatus.OK);
        }
        return new ResponseEntity<Object>("Not Found User", HttpStatus.NO_CONTENT);
    }

    // Tạo mới User POST/CONFLICT
    @RequestMapping(value = "/users", method = RequestMethod.POST)
    public ResponseEntity<String> createUser(@RequestBody User user) {
        if (ListUsers.contains(user)) {
            return new ResponseEntity<String>("User Already Exist!", HttpStatus.CONFLICT);
        }
        ListUsers.add(user);
        return new ResponseEntity<String>("Created!", HttpStatus.CREATED);
    }

    // Xóa người dùng theo id DELETE
    @RequestMapping(value = "/users/{id}", method = RequestMethod.DELETE)
    public ResponseEntity<String> deleteUserById(@PathVariable int id) {
        User user = ListUsers.get(id);
        if (user == null) {
            return new ResponseEntity<String>("Not Found User", HttpStatus.OK);
        }

        ListUsers.remove(id);
        return new ResponseEntity<String>("Deleted!", HttpStatus.OK);
    }

    // Cập nhật người dùng theo id PUT
    @RequestMapping(value = "/users", method = RequestMethod.PUT)
    public ResponseEntity<String> updateUser(@RequestBody User user) {
        User oldUser = ListUsers.get(user.getId());
        if (oldUser == null) {

```

```

        return new ResponseEntity<String>("Not Found User",
HttpStatus.NO_CONTENT);
    }
    listUsers.add(user);
    return new ResponseEntity<String>("Updated!", HttpStatus.OK);
}

```

B4. Test chương trình dùng Postman (<https://www.getpostman.com/>) hoặc HttpRequester (<https://sourceforge.net/projects/httprequester/>)

**JSON (Javascript Object Notation)** là một dạng dữ liệu mà hầu hết các ngôn ngữ lập trình hiện nay đều có thể đọc được. JSON có định dạng đơn giản, dễ dàng sử dụng và truy vấn hơn.

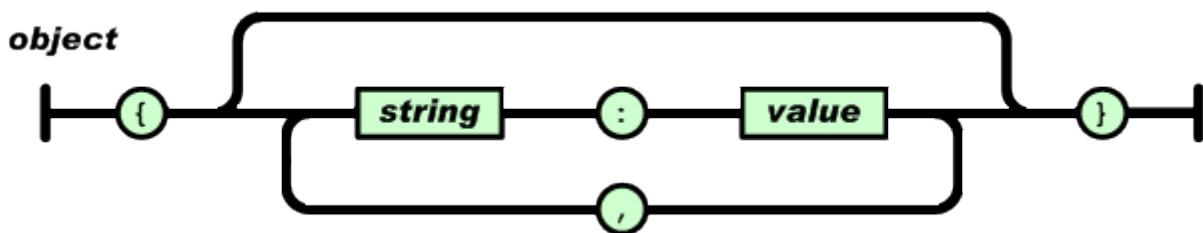
[<https://www.json.org/json-vi.html>] JSON được xây dựng trên 2 cấu trúc:

- Là tập hợp của các cặp tên và giá trị name-value. Trong những ngôn ngữ khác nhau, đây được nhận thấy như là 1 đối tượng (object), sự ghi (record), cấu trúc (struct), từ điển (dictionary), bảng băm (hash table), danh sách khoá (keyed list), hay mảng liên hợp.
- Là 1 tập hợp các giá trị đã được sắp xếp. Trong hầu hết các ngôn ngữ, this được nhận thấy như là 1 mảng, véc tơ, tập hợp hay là 1 dãy sequence.

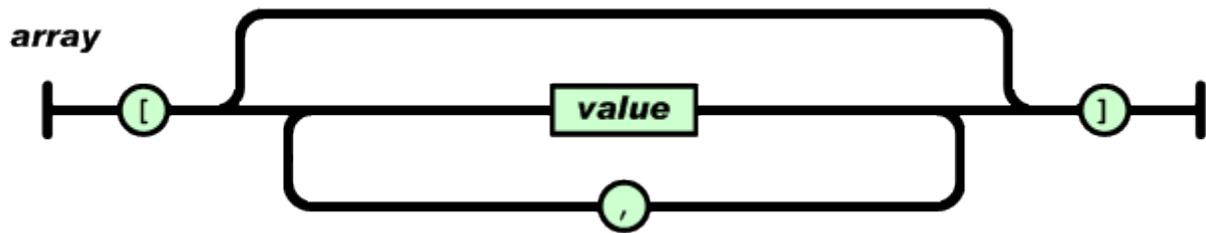
Đây là 1 cấu trúc dữ liệu phổ dụng. Hầu như tất cả các ngôn ngữ lập trình hiện đại đều hỗ trợ chúng trong 1 hình thức nào đó. Chúng tạo nên ý nghĩa của 1 định dạng hoán vị dữ liệu với các ngôn ngữ lập trình cũng đã được cơ sở hoá trên cấu trúc này.

Trong JSON, chúng có những thứ trên các định dạng:

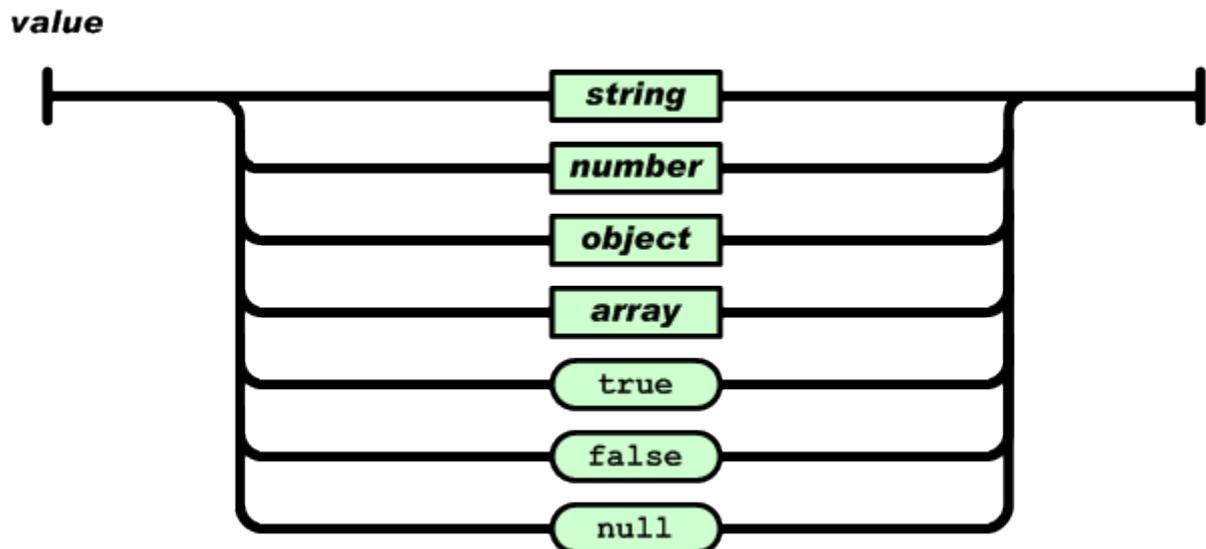
- 1 đối tượng là 1 tập các cặp tên và giá trị. 1 đối tượng bắt đầu bởi dấu ngoặc đơn trái { và kết thúc với dấu ngoặc đơn phải }. Từng tên được theo sau bởi dấu 2 chấm : và các cặp tên/giá trị được tách ra bởi dấu phẩy ,.



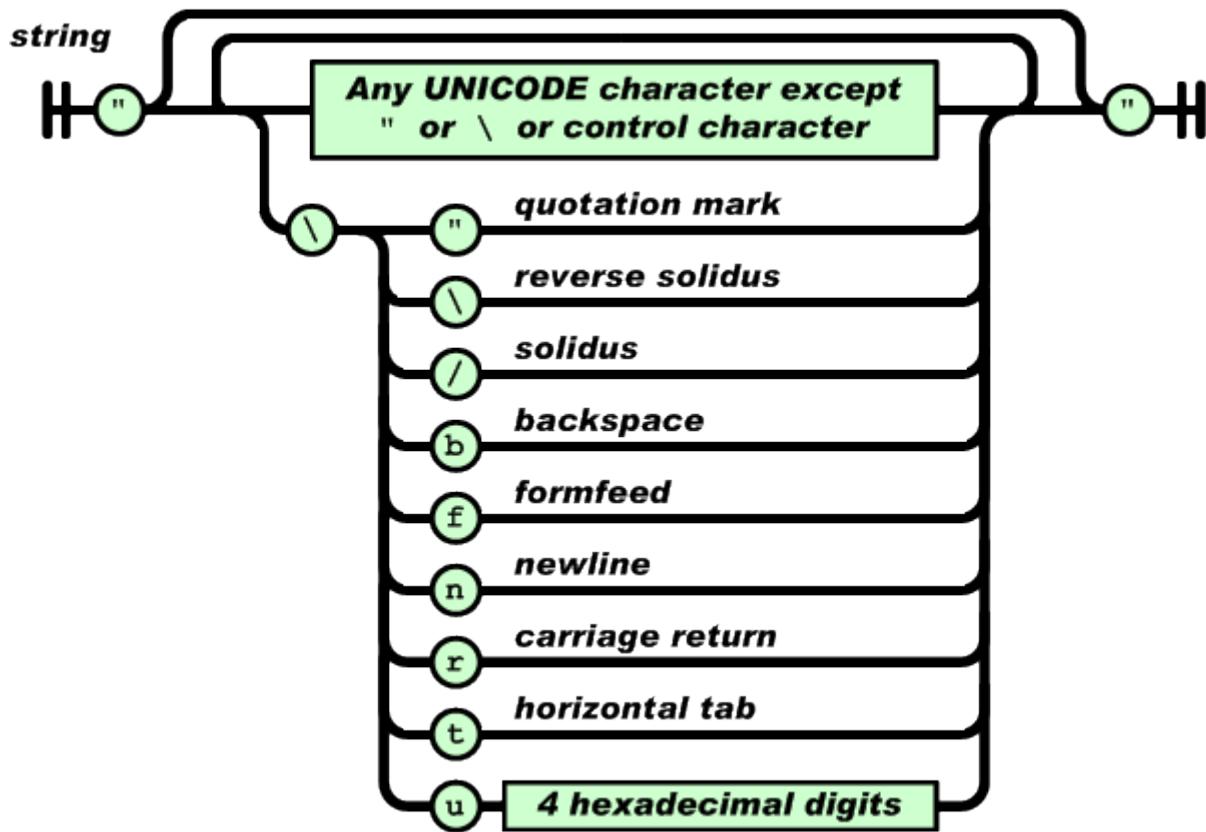
- 1 mảng là 1 tập hợp các giá trị đã được sắp xếp. 1 mảng bắt đầu bởi dấu mở ngoặc vuông [ và kết thúc với dấu ngoặc vuông phải ]. Các giá trị được cách nhau bởi dấu phẩy ,.



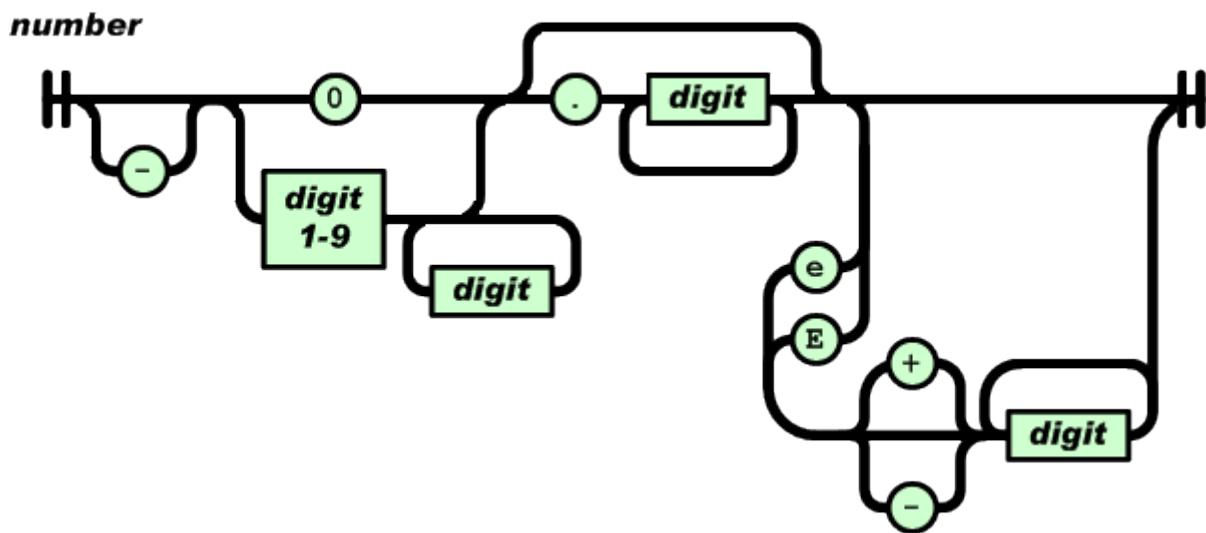
- 1 giá trị có thể là 1 chuỗi string trong những trích dẫn kép hay là 1 số, hay true hay false hay null, hay là 1 đối tượng hay là 1 mảng. Những cấu trúc này có thể đã được lồng vào nhau.



- 1 chuỗi string là 1 tập hợp của zero hay ngay cả mẫu tự Unicode, được bao bọc trong các dấu trích dẫn kép (""), dùng để thoát ra dấu chéo ngược. 1 ký tự đã được hiển thị như là 1 chuỗi ký tự đơn độc. 1 chuỗi string rất giống như là chuỗi string C hay là Java.



- 1 số rất giống 1 số C và Java, trừ định dạng oct và hex là không thể dùng.



Khoảng trống có thể chèn vào giữa bất cứ các cặp của các dấu hiệu. Trừ các chi tiết 1 vài mã hoá, mà hoàn toàn mô tả ngôn ngữ.

#### Bài 14. Spring MVC - Bài tập tổng hợp (1 bảng)

Xây dựng ứng dụng quản lý thông tin sách cho website [readanybook.com](http://readanybook.com), các trang Web cần thực hiện các chức năng: xem danh sách các quyền sách, thêm mới sách và xóa sách không cần thiết. Một phần của cơ sở dữ liệu được mô tả như sau:

BookOnline(BookID, BookName, Author, PublishYear, Rating, ImageURL)

Xây dựng ứng dụng dùng Spring MVC Framework kết hợp với SQL Server và Web server Tomcat thực hiện các công việc sau:

- Tạo cơ sở dữ liệu với các quan hệ như mô tả trên và nhập dữ liệu cần thiết để kiểm tra chương trình.
- Dùng HTML/CSS tạo Layout chung cho chương trình tương tự:

The screenshot shows a web page with a dark blue header containing the logo "read ANY book . com" and several book icons. Below the header is a list of six books arranged in two columns. Each book entry includes the book cover, title, author, rating, and a "Manage" link. To the right of the list is a sidebar titled "Links" with links to "Manage books", "Add new book", and "@Copyright StudentName - ID".

Book Cover	Title	Author	Rating
	The Silmarillion	by J.R.R. Tolkien	9.78 / 10
	A Wrinkle in Time	by Madeleine L'engle	8.36 / 10
	Ready Player One	by Ernest Cline	8.02 / 10
	Disobedience	by Naomi Alderman	7.45 / 10
	The Leisure Seeker	by Michael Zadoorian	6.77 / 10
	Arielle Immortal Awakening	by Lilian Roberts	8.36 / 10

Hình 1. Layout của trang Web quản lý thông tin sách cho website [readanybook.com](http://readanybook.com)

- Viết lớp kết nối CSDL và lớp thực hiện thao tác với việc truy xuất cơ sở dữ liệu (bao gồm các phương thức: liệt kê danh sách sách, thêm mới, xóa thông tin sách).
- Tạo Controller cho phép hiển thị giao diện Web với danh sách các quyền sách online lấy từ cơ sở dữ liệu.
- Tạo Controller cho phép hiển thị giao diện thao tác với việc thêm dữ liệu (*nhấn vào link “Add new book”*). Dữ liệu phải được kiểm tra trước khi lưu vào cơ sở dữ liệu:
  - Tất cả dữ liệu *BookID*, *BookName*, *Author*, *PublishYear*, *Rating*, *ImageURL* đều bắt buộc.
  - Năm xuất bản (*PublishYear*): từ 1900 – 2018

- Tên sách (*BookName*): Các ký tự đầu của mỗi từ là ký tự hoa, không có số hay ký tự đặc biệt trong tên.

Sau khi thực hiện thêm thành công, hiển thị danh sách mới.

- Tạo Controller cho phép hiển thị giao diện Web cho phép thực hiện chức năng quản lý (*nhấn vào link “Manage books”*): thao tác hủy dữ liệu. Trước khi hủy hỏi người dùng có muốn hủy không.

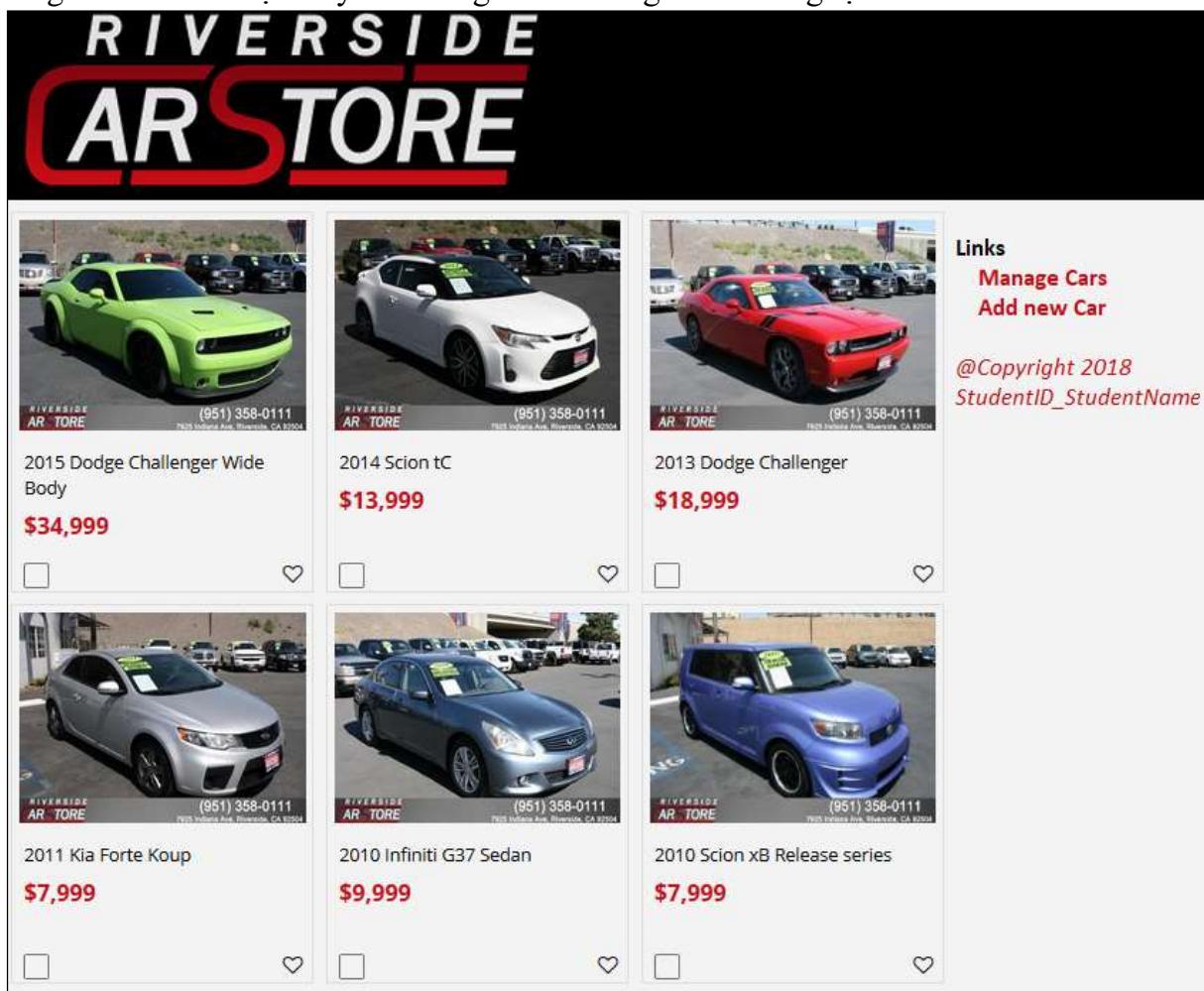
### Bài 15. Spring MVC - Bài tập tổng hợp (1 bảng)

Xây dựng ứng dụng quản lý thông tin xe website [reversidecarstore.net](http://reversidecarstore.net), các trang Web cần thực hiện các chức năng: *xem danh sách các xe đang được bán trực tuyến, thêm mới xe và xóa xe đã hết trong kho.* Một phần của cơ sở dữ liệu được mô tả như sau:

**RiversideCar(CarID, CarName, Price, ModelYear, CarDescription, ImageURL)**

Xây dựng ứng dụng dùng Spring MVC Framework kết hợp với SQL Server và Web server Tomcat thực hiện các công việc sau:

- Tạo cơ sở dữ liệu với các quan hệ như mô tả trên và nhập dữ liệu cần thiết để kiểm tra chương trình.
- Dùng HTML/CSS tạo Layout chung cho chương trình tương tự:



The screenshot shows a website for "RIVERSIDE CAR STORE". The header features the store's name in large, bold, white and red letters. Below the header is a grid of six car listings, each with a small image, the car's name, its price in red, and two interaction icons (a square and a heart). To the right of the grid is a sidebar with links for "Manage Cars" and "Add new Car", and copyright information.

Car Model	Price
2015 Dodge Challenger Wide Body	\$34,999
2014 Scion tC	\$13,999
2013 Dodge Challenger	\$18,999
2011 Kia Forte Koup	\$7,999
2010 Infiniti G37 Sedan	\$9,999
2010 Scion xB Release series	\$7,999

- Viết lớp kết nối CSDL và lớp thực hiện thao tác với việc truy xuất cơ sở dữ liệu (bao gồm các phương thức: liệt kê danh sách xe, thêm mới xe, xóa thông tin xe).
- Tạo Controller cho phép hiển thị giao diện Web liệt kê danh sách thông tin các xe lấy từ cơ sở dữ liệu.
- Tạo Controller cho phép hiển thị giao diện Web thao tác với việc thêm dữ liệu (*nhấn vào link “Add new Car”*). Dữ liệu phải được kiểm tra trước khi lưu vào cơ sở dữ liệu:
  - Tất cả dữ liệu *CarID*, *CarName*, *Price*, *ModelYear*, *CarDescription*, *ImageURL* đều bắt buộc.
  - Năm sản xuất (*ModelYear*): từ 1900 – 2018
  - Tên xe (*CarName*): Các ký tự đầu của mỗi từ là ký tự hoa, không có số hay ký tự đặc biệt trong tên.
  - Giá (*Price*): số thực.

Sau khi thực hiện thêm thành công, hiển thị danh sách mới.

- Tạo Controller cho phép hiển thị giao diện Web cho phép thực hiện chức năng quản lý (*nhấn vào link “Manage Cars”*): thao tác hủy dữ liệu. Trước khi hủy hỏi người dùng có muốn hủy không.

### Tham khảo một số thao tác:

#### B1. Câu hình dependencies:

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.0.3.RELEASE <version>
    <exclusions>
        <!-- Exclude Commons Logging in favor of SLF4j -->
        <exclusion>
            <groupId>commons-logging</groupId>
            <artifactId>commons-logging</artifactId>
        </exclusion>
    </exclusions>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>${org.springframework-version}</version>
</dependency>

<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>5.0.3.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-tx -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-tx</artifactId>
    <version>5.0.3.RELEASE</version>
```

```

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.commons/commons-dbcP2 -->
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-dbcP2</artifactId>
    <version>2.2.0</version>
</dependency>

```

B2. Khai báo các Beans trong cấu hình của Spring servlet-context.xml:

```

<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/mvc"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:beans="http://www.springframework.org/schema/beans"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/mvc
    http://www.springframework.org/schema/mvc/spring-mvc.xsd
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd">

    <!-- DispatcherServlet Context: defines this servlet's request-processing
    infrastructure -->

    <!-- Enables the Spring MVC @Controller programming model -->
    <annotation-driven />

    <!-- Handles HTTP GET requests for /resources/** by efficiently serving up static
    resources in the ${webappRoot}/resources directory -->
    <resources mapping="/resources/**" location="/resources/" />

    <!-- Resolves views selected for rendering by @Controllers to .jsp resources in the
    /WEB-INF/views directory -->
    <beans:bean
        class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <beans:property name="prefix" value="/WEB-INF/views/" />
        <beans:property name="suffix" value=".jsp" />
    </beans:bean>

    <context:component-scan base-package="se.iuh.springcarapp" />

    <beans:bean id="dataSource"
        class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <beans:property name="driverClassName"
            value="com.microsoft.sqlserver.jdbc.SQLServerDriver"/></beans:property>
        <beans:property name="url"
            value="jdbc:sqlserver://localhost:1433;databaseName=CarOnLineDB"/></beans:property>
        <beans:property name="username" value="sa"/></beans:property>
        <beans:property name="password" value="1234567890"/></beans:property>
    </beans:bean>

```

```

<beans:bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
    <beans:property name="dataSource" ref="dataSource"></beans:property>
</beans:bean>

<beans:bean id="carDAO" class="se.iuh.springcarapp.dao.CarDAO">
    <beans:property name="template" ref="jdbcTemplate"></beans:property>
</beans:bean>
</beans:beans>

```

### B3. Viết lớp Entity (*se.iuh.springcarapp.model.Car*) + Lớp thao tác với CSDL

```

package se.iuh.springcarapp.dao;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;

import se.iuh.springcarapp.model.Car;

public class CarDAO {
    JdbcTemplate template;

    public JdbcTemplate getTemplate() {
        return template;
    }

    public void setTemplate(JdbcTemplate template) {
        this.template = template;
    }

    public List<Car> getAllCars(){
        List<Car> carList = new ArrayList<Car>();
        String sql = "SELECT * FROM ReversideCar";
        carList = template.query(sql, new RowMapper<Car>() {
            @Override
            public Car mapRow(ResultSet rs, int rowNum) throws SQLException {
                String carID = rs.getString(1);
                String carName = rs.getString(2);
                double price = rs.getDouble(3);
                int modelYear = rs.getInt(4);
                String carDescription = rs.getString(5);
                String imageURL = rs.getString(6);
                Car car = new Car(carID, carName, price, modelYear,
carDescription, imageURL);
                return car ;
            }
        });
    }
}

```

```

        return carList ;
    }

    public boolean addNewCar(Car car) {
        String sql = "INSERT INTO ReversideCar VALUES(?,?,?,?,?,?)";
        int result = template.update(sql
            , new Object[] {
                car.getCarID()
                ,car.getCarName()
                ,car.getPrice()
                ,car.getModelYear()
                ,car.getCarDescription()
                ,car.getImageURL()});
        return result > 0;
    }

    public boolean deleteCar(String carID) {
        String sql = "DELETE FROM ReversideCar WHERE carid =?";
        int result = template.update(sql , new Object[] {carID});
        return result > 0;
    }
}

```

#### B4. Controller

```

package se.iuh.springcarapp;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.multipart.MultipartFile;

import se.iuh.springcarapp.dao.CarDAO;
import se.iuh.springcarapp.model.Car;

@Controller
public class CarController {

    @Autowired
    private CarDAO carDAO;
    @RequestMapping(value={"/home","/"},method=RequestMethod.GET)
    public String showCars(Model model) {
        List<Car> carList = carDAO.getAllCars();
        model.addAttribute("danhsach",carList);
        return "danhsach";
    }

    @RequestMapping(value={"/add-new-car"},method=RequestMethod.GET)

```

```

public String showAddNewCarForm(Model model) {
    model.addAttribute("car", new Car());
    return "addNewCarForm";
}

@RequestMapping(value={"/addCar"},method=RequestMethod.POST)
public String addCar(@ModelAttribute(name="car") Car car) {
    // .....

    return "danhsach";
}

```

### B5. View:

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
<link href="resources/css/style.css" type="text/css" rel="stylesheet" />
<style>
body{
    margin:0;
    padding:0;
    background-color:whiteSmoke;
}
aside{
    width:30%;
    float:right;
    height: 100%;
}
header{
    background-image: url("../images/riverside_cars_2057.PNG");
    background-repeat: no-repeat;
    background-color:black;
    min-height: 120px;
}
section{
    width:70%;
    float:left;
    height: auto;
}
.navigation{
    list-style-type: none;
}
.navigation li a{
    color: gray;

```

```

        text-decoration: none;
    }
    .card-car{
        border:1px solid black;
        height: 180px;
        width: 150px;
        display: inline-table;
        margin: 10px;
        padding:5px;
    }
    .btn{
        background-color: white;
        border: 1px solid black;
        border-radius: 5px;
        color:black;
        font-size: 16pt;
        transition:0.2s;
    }
    .btn:hover{
        background-color: green;
        border: 1px solid white;
        color:white;
    }

```

</style>

</head>

<body>

- <header></header>
- <section>
- <c:forEach items="\${danh sach }" var="car">
- <div class="card-car">
- 
- \${car.carName} <br>
- <b>\$ \${car.price}</b>
- </div>
- </c:forEach>
- </section>
- <aside>
- <h3>Links</h3>
- <ul class="navigation">
- <li><a href="manage-cars">Manage Cars</a></li>
- <li><a href="add-new-car">Add new Car</a></li>
- </ul>
- <p>Copyright 2018 - Name - Student ID </p>
- </aside>

</body>

</html>

## Bài 16. Spring MVC - Bài tập tổng hợp (2 bảng quan hệ 1-N)

Bài tập ôn lập trình Web nâng cao (Java) – Thời gian 120 phút

Quản lý thông tin về đề tài và giảng viên ra đề tài tốt nghiệp cho sinh viên ngành công nghệ thông tin được mô tả như sau:

Giảng viên bao gồm các thông tin như *mã giảng viên, tên giảng viên, lĩnh vực nghiên cứu, số điện thoại và email của giảng viên*. Một giảng viên sẽ thực hiện nhiều đề tài khác nhau, mỗi một đề tài có các thông tin như *mã đề tài, tên đề tài, năm đăng ký thực hiện, hình ảnh minh họa, và mô tả của đề tài*.

GIANGVIEN(MAGV, TENGV, LINHVUCNGHIENU, DIENTHOAI, EMAIL)

DETAI(MADETAI, TENDETAI, NAM, MOTADETAI, URLHINH, MAGV)

Xây dựng ứng dụng dùng mô hình MVC với công nghệ Java (*Spring Web MVC hoặc JavaServer Faces*) kết hợp với SQL Server và Web server Tomcat thực hiện các công việc sau:

- Tạo cơ sở dữ liệu với các quan hệ như mô tả trên và nhập dữ liệu cần thiết để kiểm tra chương trình.

- Dùng HTML/CSS tạo Layout chung cho chương trình. Layout tối thiểu:

Hình ảnh Logo + Menu chức năng (Danh sách   Thêm   Chức năng quản lý)
Nội dung
@Copywrite 2018

- |                 |
|-----------------|
| @Copywrite 2018 |
|-----------------|
- Xây dựng các lớp Entity bao gồm các get/set cho các thuộc tính, constructors, và các annotation kiểm tra dữ liệu (dùng Hibernate và Java Validator)
    - Lớp GiangVien: Mã, tên giảng viên là bắt buộc, số điện thoại từ 9-11 số và email phải đúng chuẩn email [yourname@yourcompany.com/org.net.vn](mailto:yourname@yourcompany.com/org.net.vn).
    - Lớp DeTai: Tên đề tài, mã đề tài, năm bắt đầu là NOT NULL. Tên đề tài mỗi từ bắt đầu bởi ký tự hoa, có số nhưng không có ký tự đặc biệt và không quá 100 ký tự (dùng RegularExpression).
  - Thực hiện các cấu hình cho Project, cấu hình cho Spring/JavaServer Faces.
  - Tạo lớp (trong Controller cần @Autowired đến lớp DAO này) cho phép thao tác với CSDL, lớp này bao gồm các phương thức: *lấy danh sách giảng viên, lấy danh sách đề tài theo mã giảng viên, lấy thông tin chi tiết đề tài theo mã đề tài, thêm đề tài mới, thêm giảng viên mới, xóa đề tài theo mã đề tài*.
  - Tạo giao diện Web cho phép thực hiện thao tác liệt kê dữ liệu: Liệt kê dữ liệu danh sách các đề tài theo các giảng viên. Có thể thực hiện theo 1 trong 2 cách sau:

- Ban đầu hiển thị ComboBox danh sách giảng viên khi chọn giảng viên cụ thể, hiển thị danh sách tên các đề tài.
- Hiển thị danh sách GV, mỗi giảng viên có từng danh sách con bao gồm tên các đề tài.
- Nhấn vào link của tên đề tài, cho phép xem chi tiết thông tin của từng đề tài, thông tin này lấy từ CSDL.
- Tạo giao diện Web cho phép thực hiện giao diện thao tác với việc thêm dữ liệu
  - a. Tạo Form thêm thông tin đề tài mới với đầy đủ thông tin phù hợp với CSDL. Nếu người dùng không chọn hình ảnh, lấy hình ảnh mặc định, danh sách GV phải được chọn từ ComboBox. Kiểm tra dữ liệu nhập phía Server dùng Hibernate và Java Validation trên Form: Tên đề tài mỗi từ bắt đầu bởi ký tự hoa, có số nhưng không có ký tự đặc biệt và không quá 100 ký tự (dùng RegularExpression). Thông tin nhập vào hợp lệ sau khi nhấn nút “Lưu” sẽ được thêm vào cơ sở dữ liệu và hiển thị trang danh sách với kết quả mới.
  - b. **Upload hình khi thêm đề tài mới (optional).**
  - c. Form thêm giảng viên mới. Mã, tên giảng viên là bắt buộc, số điện thoại từ 9-11 số và email phải đúng chuẩn email [yourname@yourcompany.com/org/net.vn](mailto:yourname@yourcompany.com/org/net.vn). Thông tin nhập vào hợp lệ sau khi nhấn nút “Lưu” sẽ được thêm vào cơ sở dữ liệu và hiển thị trang danh sách với kết quả mới.

Tạo giao diện Web dùng Layout chung cho phép thực hiện chức năng quản lý: thao tác hủy dữ liệu liên quan đến đề tài. Khi chọn chức năng này, hiển thị danh sách và kèm theo chức năng hủy thông tin đề tài khỏi cơ sở dữ liệu.

#### Bài 17. Spring MVC - Bài tập tổng hợp (2 bảng quan hệ 1-N)

Ứng dụng quản lý thông tin tác giả, tác phẩm cho nhà sách Nhã Nam [www.nhanam.vn](http://www.nhanam.vn) mua bán sách trực tuyến, các trang Web cần thực hiện các chức năng: *xem danh sách tác phẩm theo tác giả, xem chi tiết từng tác phẩm, thêm mới tác phẩm và quản lý (xóa) tác phẩm*. Một phần của cơ sở dữ liệu được mô tả như sau:

TACGIA(MATACGIA, HOTEN, LINHVUC, TIEUSU)

TACPHAM(MATP, TENTP, LOAITP, SOTRANG, GIA, TOMTAT, HINH, MATG)

Khi có một tác phẩm của một tác giả được nhập về kho, thông tin tác phẩm này cần được nhập vào hệ thống trực tuyến của nhà xuất bản. Thông tin của một tác phẩm bao gồm *mã tác phẩm, tên, loại tác phẩm, số trang, giá bán, nội dung tóm tắt, và hình mẫu*. Mỗi tác phẩm này sẽ thuộc về một tác giả. Thông tin tác giả được mô tả: *mã tác giả, họ tên, lĩnh vực và tiêu sứ* của tác giả đó.

Xây dựng ứng dụng dùng mô hình MVC với công nghệ Java (Spring Web MVC hoặc JavaServer Faces) kết hợp với SQL Server và Web server Tomcat thực hiện các công việc sau:

- Tạo cơ sở dữ liệu với các quan hệ như mô tả trên và nhập dữ liệu cần thiết để kiểm tra chương trình.
- Dùng HTML/CSS tạo Layout chung cho chương trình.



*Bởi vì sách là thế giới.*

[Danh sách](#) | [Thêm tác phẩm mới](#) | [Chức năng quản lý](#)

©2018 Nhà sách Nhã Nam. All Rights Reserved.

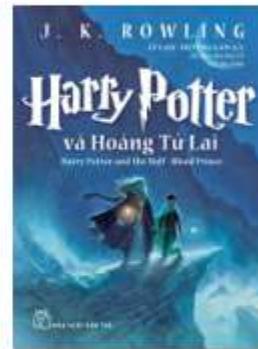
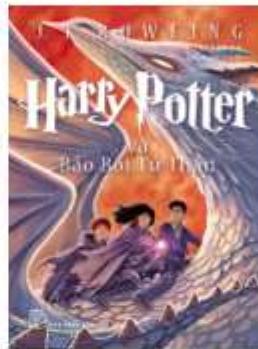
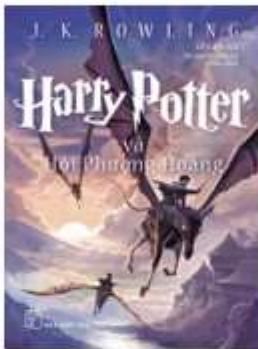
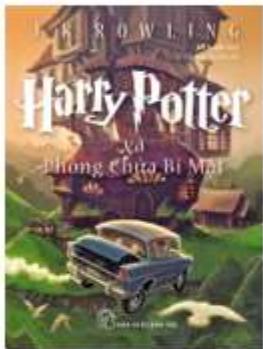
- Tạo lớp mô tả thông tin cho CSDL mô tả ở trên *TacGia.java*, *TacPham.java* (bao gồm các get/set cho các thuộc tính và các phương thức equals, constructors).
- Tạo lớp *TacPhamDAO.java* cho phép thao tác với CSDL, lớp này bao gồm các phương thức: *lấy danh sách tác giả*, *lấy tác phẩm theo mã tác giả*, *lấy thông tin chi tiết tác phẩm theo mã tác phẩm*, *thêm phẩm mới*, *xóa tác phẩm theo mã tác phẩm*.
- Tạo giao diện Web cho phép thực hiện thao tác liệt kê dữ liệu: Liệt kê dữ liệu danh sách các tác phẩm theo từng tác giả.



Bởi vì sách là thế giới.

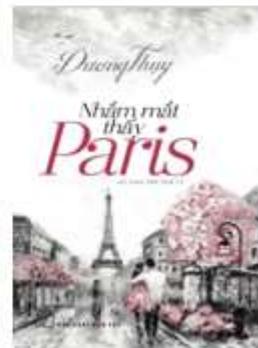
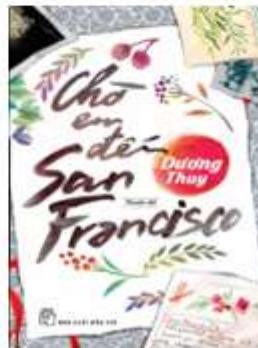
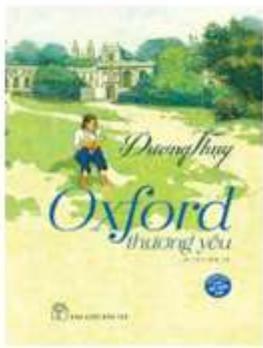
[Danh sách](#) | [Thêm tác phẩm mới](#) | [Chức năng quản lý](#)

### Tác giả: J.K.Rowling



>>

### Tác giả: Dương Thụy



>>

©2018 Nhà sách Nhã Nam. All Rights Reserved.

- Nhấn vào hình của các tác phẩm trên website, cho phép xem chi tiết thông tin của từng tác phẩm, thông tin này lấy từ CSDL.
- Tạo giao diện Web cho phép thực hiện chức năng quản lý: thao tác hủy dữ liệu. Khi chọn chức năng này, hiển thị danh sách và kèm theo chức năng hủy thông tin tác phẩm khỏi cơ sở dữ liệu.
- Tạo giao diện Web cho phép thực hiện giao diện thao tác với việc thêm dữ liệu
  - a. Tạo Form thêm thông tin tác phẩm mới với đầy đủ thông tin phù hợp với CSDL.
  - b. Kiểm tra dữ liệu nhập phía Server dùng Hibernate và Java Validation trên Form:
    - Mã tác phẩm, Tên tác phẩm, Giá, Số trang, Loại tác phẩm và Tóm tắt: các thông tin này bắt buộc nhập.
    - Số trang là số nguyên, Giá là số thực (dùng RegularExpression).
    - Thông tin loại tác phẩm cách nhau bởi dấu phẩy và không quá 100 ký tự (dùng RegularExpression).
  - c. Thông tin nhập vào hợp lệ sau khi nhấn nút “Lưu” sẽ được thêm vào cơ sở dữ liệu và hiển thị trang danh sách với kết quả mới.



Bởi vì sách là thế giới.

[Danh sách](#) | [Thêm tác phẩm mới](#) | [Chức năng quản lý](#)

### Thêm thông tin tác phẩm mới

Mã tác phẩm

Loại tác phẩm

Tên tác phẩm

Nội dung tóm tắt

Số trang

Giá

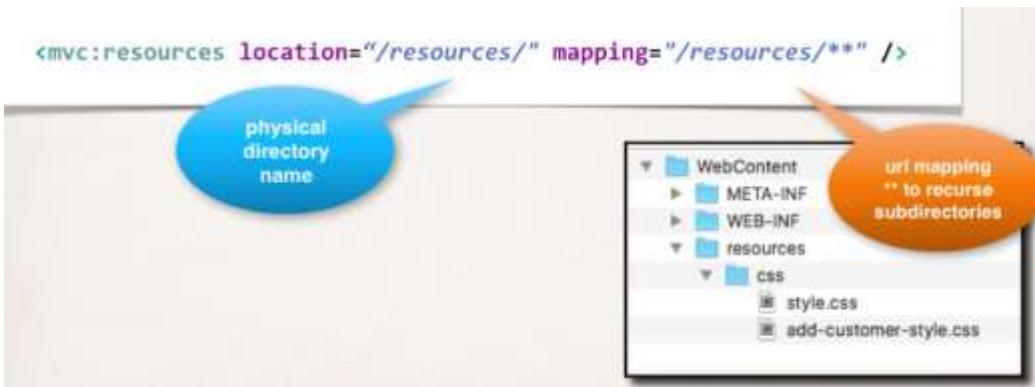
URL hình ảnh



Lưu

©2018 Nhà sách Nhã Nam. All Rights Reserved.

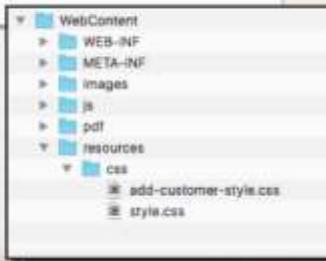
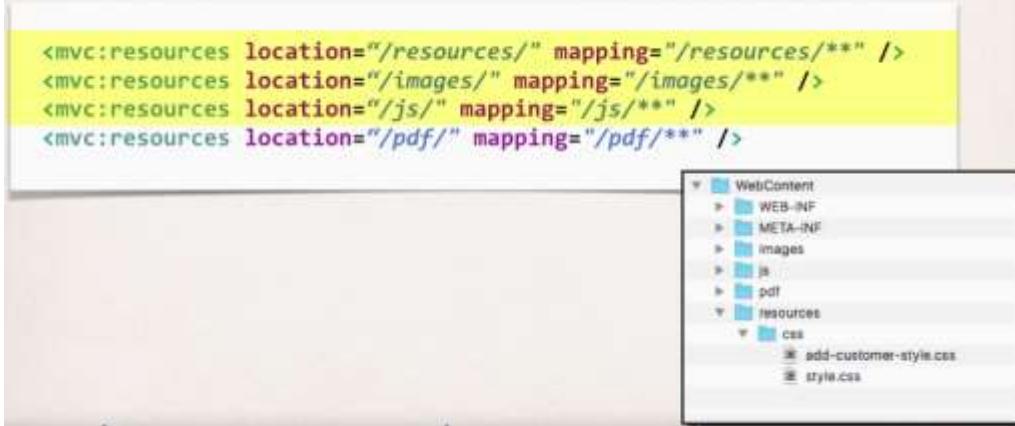
Cấu hình thư mục lưu các tập tin tài nguyên (HTML, CSS, JS, ...) trong tập tin cấu hình của Spring. Tương tự cho cả 2 Dynamic Web Project và Spring Legacy Project. Đối với Spring Legacy Project thư mục được tạo trong webapp.



Dùng Expression Language lấy tên thực của ứng dụng, chỉ rõ đường dẫn đến tập tin tài nguyên.



Trường hợp tách các thư mục riêng thì trong file cấu hình Spring



Xử lý Request dạng GET/POST theo thông thường

```
@RequestMapping(path="/processForm", method=RequestMethod.GET)
public String processForm(...) {
    ...
}

@RequestMapping(path="/processForm", method=RequestMethod.POST)
public String processForm(...) {
    ...
}
```

Xử lý Request dạng GET/POST dùng @GetMapping và @PostMapping (từ Spring 4.3)

```
@GetMapping("/processForm")
public String processForm(...) {
    ...
}

@PostMapping("/processForm")
public String processForm(...) {
    ...
}
```