TÀI LIỆU KHÓA HỌC “Java Spring MVC”

Tác giả: Hỏi Dân IT & Eric

Version: 4.0

Note cập nhật:

- Cập nhật source code của cả khóa học video #2.2 (và theo từng video)
- Update video #9.1, bổ sung logic check Khoảng trắng (v2.0)
- Upload link download mysql workbench (v4.0)

Chapter 1: Bắt buộc xem
Hướng dẫn sử dụng khóa học hiệu quả

#1. Hướng dẫn sử dụng khóa học này hiệu quả

Bạn vui lòng "xem video lần lượt" theo trình tự. Vì khóa học như 1 dòng chảy, video sau sẽ kế thừa lại kết quả của video trước đó.

1. Dành cho học viên "có ít thời gian"
Nếu bạn vội, cần học nhanh, hoặc "bạn đã biết rồi", thì "vẫn xem video, cơ mà không cần code theo".

Lưu ý: vẫn xem qua tài liệu khóa học để biết "video hướng dẫn gì".
Đã "Không xem video", thì cần "đọc giáo án".

Có như vậy mới biết khóa học nó làm cái gì.

2. Dành cho học viên "thông thường"
Nguyên tắc:
- Xem video lần lượt
- Xem video kết hợp với giáo án. Bạn không cần take note, vì những điều quan trọng đã có trong giáo án

- **Bạn vui lòng code theo video.**
Nếu bạn "code theo ý bạn", vui lòng "không hỏi khi có bugs".

Câu chuyện này giống như việc bạn đi khám bệnh, nhưng không tin lời bác sĩ
=> Nếu bạn giỏi, bạn làm luôn bác sĩ, còn đi khám bệnh làm gì.

- Bạn có thể "code theo ý bạn muốn", sau khi "đã kết thúc khóa học"

- Nếu bạn có thắc mắc (hoặc có ý tưởng/nhận thấy bugs), take note lại, bạn hỏi, rồi mình giải đáp.
Chứ không phải là "tự ý làm theo điều các bạn muốn".
Vì đa phần, các bugs trong khóa học mình đã fix hết rồi.
Nên là yên tâm để học theo bạn nhé.

3. Về cách code.

Bạn vui lòng code theo video, từ cách đặt tên biến, hàm. Vì mình đã tuân theo "convention tối thiểu" khi bạn đi làm đấy

4. Về bài tập thực hành

Đối với bài tập thực hành, bạn cứ code theo cách bạn hiểu, và kết hợp với "search Google, stackoverflow..."

KHÔNG DÙNG CHATGPT. Đây giống kiểu chưa học "phép tính", mà đã "dùng máy tính".

Nên nhớ 1 điều, trước 2023, không có chat gpt, thì mình học như thế nào ?

Khi bạn đã đi làm, bạn có quyền dùng cái gì bạn thích, còn với beginner, hãy biết say NO với CHAT GPT.

tương tự bạn dạy con bạn:
học lớp cấp 1: không chịu học tính nhẩm => đưa luôn máy tính cho nó. Rời máy tính ra, là không biết làm phép tính

còn với học sinh cấp 2, 3 : dùng máy tính tùy thích

5. Về source của cả khóa học

Source code của cả khóa học KHÔNG ĐƯỢC CUNG CẤP.
Học viên cần code theo video để có source code.

Chỉ có videos mình nói "cho source code", source code sẽ có trong tài liệu khóa học.

#2.1 Tài liệu của khóa học

//Link download tài liệu khóa học

#2.2 Hướng Dẫn Sử Dụng Source Code của Khóa Học

Đây là khóa học thực hành, mình "KHÔNG khuyến khích" sử dụng source code.

Vì nếu các bạn "không tự code", kiến thức không bao giờ (never) là của các bạn.

Đi làm, ai code hộ bạn ?

Link project final, xem tại đây

//Có thể xem full source code, hoặc xem code theo từng video (check theo commit)

#3. Demo kết quả đạt được
Link video demo: https://youtu.be/V6LA6MlTHs0

- Dự án: xây dựng website bán laptop với Java Spring

Lưu ý: đây là dự án thiên hướng "fullstack", tức rằng bạn sẽ code từ A tới Z, cả frontend lẫn backend :v

1. Công nghệ sử dụng
Backend: (Java)
- **Spring Boot** : bootstrapping project
- **Spring MVC** : mô hình model-view-controller
- **Spring Security**: bảo vệ route với role (dạng basic)
- **Spring Session:** quản lý phiên đăng nhập của người dùng
- **Spring Data** (Hibernate/JPA): sử dụng ORM (object relational mapping) để mô hình hóa Model
    + có học cách tư duy phân tích database (ràng buộc relationship)

- Build tool: **Maven**

Frontend: HTML, CSS và Javascript
- View Engine: JSP
- AJAX để gọi APIs (không cần reload page)
Database: MySQL (phần mềm MySQL WorkBench)

2. Triển khai dự án
Dự án được chạy tại localhost và không triển khai lên hosting, vì:
- rất ít hosting FREE hỗ trợ java + mysql
- hosting FREE không lưu trữ ảnh upload

Tuy nhiên, trong khóa học có hướng dẫn build với Docker
=> nếu bạn muốn triển khai thực tế, mua vps, cài docker và triển thôi

Tham khảo (hướng triển khai với docker): https://hoidanit.vn/khoa-hoc/ultimate-guide-to-deploy-react-nodejs-640bee82f7099c369b3bc6a4.html

3. Học viên nào có thể học ?
Để học được khóa học này, học viên cần:
+ Biết cú pháp của Java và tư duy lập trình hướng đối tượng

Tham khảo (nếu bạn chưa biết về Java):
https://www.youtube.com/playlist?list=PLncHg6Kn2JT5EVkhKoJmzOytHY39Mrf_o

+ Biết cú pháp cơ bản của HTML, CSS và Javascript

Khóa học này dành cho những bạn:
- Muốn 1 khóa học (và chỉ 1), có thể làm ra 1 website với java
- Muốn tìm hiểu về Java Spring sử dụng mô hình MVC
- Khóa học mua 1 lần, học mãi mãi và nhận update free

4. Về source của cả khóa học

Source code của cả khóa học ĐƯỢC CUNG CẤP.

Yên tâm 1 điều là:
- Nếu bạn code theo video, 100% sẽ có được thành quả
- Source code được update theo thời gian. 1 khi bugs được học viên report
=> sẽ được fix ngay lập tức

5. Về testing

Khóa học này không hướng dẫn viết testcase

#4. Yêu cầu để học được khóa học này

- Bạn cần biết cú pháp java và java 8 (lamda):
https://www.youtube.com/playlist?list=PLncHg6Kn2JT5EVkhKoJmzOytHY39Mrf_o

- Bạn cần biết cú pháp cơ bản của html/css/js

- Về source code của cả khóa học => không share

#5. Về Tác giả

Về tác giả:

Mọi thông tin về Tác giả Hỏi Dân IT, các bạn có thể tìm kiếm tại đây:

Website chính thức: https://hoidanit.vn/

Youtube “Hỏi Dân IT” : https://www.youtube.com/@hoidanit

Tiktok “Hỏi Dân IT” : https://www.tiktok.com/@hoidanit

Fanpage “Hỏi Dân IT” : https://www.facebook.com/askITwithERIC/

Udemy Hỏi Dân IT: https://www.udemy.com/user/eric-7039/

Chapter 2: Setup Environment
Cài đặt môi trường thực hiện dự án

#6. Cài Java version 17
Môi trường Java trong dự án này sử dụng version 17. Bạn vui lòng sử dụng version này để hạn chế tối đa lỗi có thể xảy ra.

Link tải Java 17 và IDE sử dụng trong video (windows):
https://drive.google.com/drive/folders/11_KIENr8SJebcQMGmFJSDgr1Sli0Hv0J?usp=sharing

- Lưu ý: cài đặt java 17
1. JDK (java development kit)
https://www.oracle.com/java/technologies/downloads/archive/
https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html

download và cài đặt java 17 (8, 11, 17, 21 -> LST: long term support)

- kiểm tra version: **java --version**

- kiểm tra java path (if needed)
https://www.w3schools.com/java/java_getstarted.asp

#7. Cài đặt Visual Studio Code

Công cụ code trong dự án sử dụng VSCode, 1 IDE hoàn toàn miễn phí

Link download:
https://code.visualstudio.com/download

#8. Cấu hình Visual Studio Code

1. Format Code

Setup Format on Save

Mục đích: Mỗi lần nhấn Ctrl + S , code sẽ được auto format trông cho đẹp/dễ nhìn

2. Cài đặt Extensions

Lưu ý: off các extension như eslint, prettier … để tránh xung đột

Fact: đi làm, người ta cấu hình eslint, prettier..thông qua code, vì mỗi 1 dự án (1 khách hàng 1 yêu cầu), cài global qua extension thì cái nào cũng giống cái nào

Đồng thời, với rule trên sẽ đảm bảo mọi thành viên trong team sẽ có cấu hình giống nhau

Các extensions cài đặt thêm:

- code spell checker : hỗ trợ check chính tả khi đặt tên tiếng anh
- extension pack for java : hỗ trợ code/debug java
- spring boot Extension Pack : hỗ trợ code java spring

#9.1 Lưu ý quan trọng về "Dấu Cách"

- Trailing Spaces : cái này dùng để check file có thừa khoảng trắng hay không, ae cài đặt thêm nhé. Dùng extension để check file, ví như file **application.properties (**các bạn copy thừa khoảng trắng là nó bị toang)
https://marketplace.visualstudio.com/items?itemName=shardulm94.trailing-spaces

Cấu hình chỉ ignore (check) một vài file cụ thể:
https://github.com/shardulm94/vscode-trailingspaces?tab=readme-ov-file#ignore-syntax

"trailing-spaces.syntaxIgnore": ["html", "java", "javascript", "markdown", "sql", "typescript", "typescriptreact", "xml", "xsl", "yaml"]

#9. Tại sao mình dùng VScode ?
Có 2 IDE nổi tiếng nhất để code java
- Eclipse/Spring Tool Suite (miễn phí)
- IntelliJ IDEA (trả phí), dùng miễn phí bản Community

IDE thực chất là công cụ code, giúp gợi ý code và phát hiện lỗi
=> dùng công cụ code nào mà bạn "thoải mái nhất"

Mình chọn VScode vì:
- miễn phí (nếu bạn code Frontend thì chắc chắn bạn sẽ thích)
- có gợi ý code và phát hiện lỗi
- theme dark :v
- hỗ trợ git out-of-the-box
- support mạnh mẽ cho "frontend" => tức là 1 IDE cho cả frontend/backend

Trong khóa học này, chỉ cần bạn code giống mình, dùng IDE nào không quan trọng, điều quan trọng, chính là cách chúng ta code ra làm sao :v

Recommend: dùng VScode, để đảm bảo bạn và mình giống nhau 100%, từ coding cho tới debug :v

Yên tâm 1 điều là: điều quan trọng nhất chính là khả năng bạn “tư duy” (mindset), bạn có thể dùng những điều học được, để áp dụng sang IDE bạn thích.

Fact: mình đã từng rơi vào công ty (khá to), cơ mà không mua license bản quyền phần mềm (trong khi không cho dùng crack)
=> bắt buộc phải dùng các công cụ free @@

#10. Cài đặt và sử dụng Git

- Nếu bạn chưa biết gì về Git, xem nhanh tại đây:
https://www.youtube.com/playlist?list=PLncHg6Kn2JT6nWS9MRjSnt6Z-9Rj0pAlo

- Sử dụng Git theo nguyên tắc:

1. Học xong video nào, commit đẩy lên Github/Gitlab
=> tạo cơ hội để thực hành câu lệnh của Git, ví dụ:
git add
git commit
git push...

2. Git là công cụ "mặc định bạn phải biết" khi đi làm phần mềm
=> điều 1 ở trên giúp bạn thực hành

3. Thói quen học xong video nào, đẩy code lên Git, giúp bạn tạo ra bản "backup" cho project của bạn

Ví dụ máy tính bạn bị hỏng đột xuất/bị mất
=> vẫn còn code, chỉ cần pull về code tiếp mà không phải code từ đầu.

Nên nhớ, khóa học này mình không share source code => chỉ bạn giúp được bạn

4. Trong trường hợp bạn bị bug
=> bạn có thể gửi link github/gitlab cho mình xem => support fix bug

=> Mục đích sử dụng git ở đây là : backup code + thực hành công cụ đi làm mà bạn "phải biết" nếu muốn đi thực tập/đi làm.

#11. Cài đặt Google Chrome

Lưu ý: sử dụng version tiếng anh
=> change language

Mục tiêu:

- Sử dụng Google Chrome để chạy ứng dụng web
- Ngôn ngữ hiển thị là Tiếng Anh

#12. Cài đặt Postman

Download Postman: https://www.postman.com/downloads/

Mục đích: dùng Postman để test backend Java

Chapter 3: Hello World với Spring
Làm quen với Java Spring Framework

#13. Java Spring là gì ?
1. What ?
Java Spring là Java "mùa xuân" :v.

Java Spring là một "framework" sử dụng ngôn ngữ Java.
=> để học Spring, chúng ta "bắt buộc" cần biết "cú pháp của java"

Khi nó là framework, nó thưởng "huge" (to lớn), và cung cấp bộ khung/hạ tầng để chúng ta tạo dự án/code nhanh hơn.

2. Why ?
Các công ty lựa chọn Spring vì:
- Sử dụng ngôn ngữ Java: an toàn (security) và hiệu năng (performance)
- Spring là framework => tiết kiệm thời gian phát triển dự án

- Spring là framework được phát triển từ 2002 => thời gian đã chứng minh sự hiệu quả của nó

- Ngôn ngữ Java được ông lớn Oracle chống lưng
=> sản phẩm phổ biến sẽ khó mà die :v

3. When ?
Khi nào nên sử dụng Java Spring ?

- Khi bạn đã biết ngôn ngữ java, muốn sử dụng framework để tiết kiệm thời gian

- Khi bạn muốn code website với Java => spring làm website là đa số :v

- Khi bạn đã dùng "hệ sinh thái của Oracle" (các cty lĩnh vực tài chính/ngân hàng dùng rất nhiều)

- Khi bạn muốn 1 công cụ "đã được kiểm chứng" theo thời gian (security/performance)

Khi nào không nên sử dụng Java Spring:
- Bạn không thích code Java, bạn thích Javascript, Python, PHP...

- Bạn không thích cú pháp dài dòng, ràng buộc chặt chẽ qua kế thừa, oop...

- Bạn cần 1 giải pháp nhanh, gọn và tiết kiệm thời gian cho công ty.
Dự án bạn phát triển chỉ cần code chạy được, về yếu tố security/performance "nó là tương đối"

=> đây là lý do tại sao JavaScript, Python nó vượt trội về "độ nổi tiếng ngày nay"
Javascript: code cả frontend lẫn backend
Python: hỗ trợ mạnh mẽ các thư viện về AI (trí tuệ nhân tạo), ML (học máy) và Big data (xử lý dữ liệu lớn)

Java có thể làm được, cơ mà không hiệu quả bằng :v

#14. Việc làm về Java Spring ?

1. Nhu cầu của thị trường

Khi bạn đã dùng "hệ sinh thái của Oracle" (các cty lĩnh vực tài chính/ngân hàng dùng rất nhiều)
=> chắc chắn bạn sẽ dùng Java Spring để phát triển website

Để biết nhu cầu "việc làm" như thế nào, cần nghiên cứu qua các website việc làm, hoặc đơn giản, gõ trực tiếp vào google.

=> Việc làm thì "luôn có", điều quan trọng, là bạn có "apply" được hay không ?

Và để có thể "apply", việc đầu tiên cần làm, "là học kiến thức".

Không hành động, làm sao có kiến thức, và mãi mãi là beginner :v

2. Tại sao ít việc làm Java Spring dành cho beginner ?
- Java Spring là công nghệ lâu đời, ít có sự đột phá về các lĩnh vực mới mẻ/trending bây giờ: AI, Big data..
=> Số lượng việc làm "có thể ít hơn" các ngôn ngữ trending

- Học/sử dụng Java tốn thời gian hơn một vài ngôn ngữ khác (như Javascript/Python)
=> không phải công ty nào "cũng bắt buộc dùng Java"

- Jobs về Java Spring vẫn có, tuy nhiên nó "đòi kinh nghiệm"

=> **Giải pháp cho beginners là :**
- Học kiến thức
- Đi thực tập
- Xin fresher

=> kiên trì theo đuổi là có kết quả bạn nhé. (nên nhớ là lương dev Java nó không có thấp.kk)

#15. Setup Dự Án Thực Hành

Lưu ý: môi trường thực hành là Java version 17.

1. Việc cần làm

Bước 1: Clone project
https://gitlab.com/public-starter-projects1/000-java/01-java-spring-mvc/01-java-spring-laptopshop-starter

Bước 2: Loading project với VSCode, chờ xíu để nó tự cài thư viện

2. Behind the scenes (XEM VÀ KHÔNG CẦN LÀM THEO)

Lưu ý:
- Để đảm bảo code của mình và code của bạn là giống nhau nhất có thể (hạn chế tối đa bugs), bạn vui lòng không thực hiện mục 2 này.

- Bạn chỉ thực hiện theo mục 2 này "SAU KHI BẠN ĐÃ HỌC XONG KHÓA HỌC", và muốn "tự tạo dự án từ đầu".

Tóm lại: mục này sinh ra là để giải đáp sự "tò mò" của nhiều bạn về câu hỏi, mình tạo dự án như thế nào.
Xem và BẠN KHÔNG CẦN LÀM THEO, OK ?

Bước 1: truy cập https://start.spring.io/

Bước 2: Cấu hình dự án theo cách bạn muốn

Bước 3: Generate source code (download)

Tại sao mình không làm bước này, vì đơn giản version Java, version Spring thay đổi theo thời gian.

=> Việc bạn clone code sẽ đảm bảo code giống nhau (từ version java, version Spring...)

#16. Cách đẩy dự án lên Github/Gitlab của chính bạn
- Nếu bạn chưa biết gì về Git, học ngay và luôn:
https://www.youtube.com/playlist?list=PLncHg6Kn2JT6nWS9MRjSnt6Z-9Rj0pAlo

Bước 1: Đảm bảo rằng bạn đã clone code từ video trước (đã có dự án cần quản lý với Git)

Bước 2: Tạo repository trên github/gitlab

Bước 3: Đẩy dự án lên repository vừa tạo
git init
git add .
git commit -m "init project"
git add origin .... (copy paste :v)

với github:
git push origin master

với gitlab:
git push origin main

Nếu muốn đẩy nhánh master:
git checkout -b master
git push origin master

Bước 4: Minh họa khi code xong 1 video
git add
git commit
git push …

#17. Hello world with Spring Boot
Thêm component cho spring web:

1. Setup Spring for web
Update file pom.xml:

```
<dependency>
	<groupId>org.springframework.boot</groupId>
	<artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

2. Viết Hello World

Tài liệu: https://spring.io/guides/gs/spring-boot/

Bước 1: Tạo file HelloController.java (cùng cấp với file main)

Bước 2: update nội dung như sau:

```
@RestController
public class HelloController {

	@GetMapping("/")
	public String index() {
		return "Hello World from Spring Boot!";
	}

}
```

Bước 3: chạy project
Truy cập: http://localhost:8080/

```
//properties:
server.port = 8081
```

#18. Spring Boot là gì ?

Spring Boot = Spring + Boot
Boot === Bootstrapping app (chạy dự án lên/start app)

1. Spring là gì ?
- là 1 framework rất nổi tiếng để build Java application

Vấn đề tồn đọng:
- Phát triển một ứng dụng Java Spring truyền thống mất rất nhiều công sức do setup phức tạp, ví dụ:
+ lựa chọn thư viện nào (Java dependencies)
+ config như thế nào: sử dụng .xml hay java config file
+ setup server như thế nào: tomcat, jboss...
...

2. Spring Boot
- Làm đơn giản hóa quá trình xây dựng Java app sử dụng Spring framework
- Giảm thiểu tối đa quá trình "cấu hình" cho dự án:
+ đơn giản hóa quá trình cài đặt thư viện, quản lý cấu hình thông qua maven...

- cung cấp sẵn server để chạy ứng dụng (embedded HTTP server)
(không cần phải cài đặt server riêng lẻ)

3. Spring Boot vs Spring
Behind the scenes, Spring Boot sử dụng "Spring"
Spring Boot làm "đơn giản hóa" quá trình sử dụng "Spring"

4. Một vài câu hỏi thường gặp:

Q1: Spring Boot thay thế Spring MVC, Spring REST...

No.

Spring Boot có thể sử dụng các công nghệ trên . 1 ứng dụng Java Spring thực tế, sẽ bao gồm:

+ Spring Boot
+ Spring MVC, Spring REST...

có nghĩa rằng, chúng ta cần Spring Boot để có thể setup nhanh chóng 1 ứng dụng Java Spring, tạo nền tảng cho công nghệ Spring MVC, Spring REST… phát triển

Hiểu đơn giản:

- Spring Boot là cái móng nhà (setup cấu hình để có 1 dự án Spring)
- Spring MVC, Spring REST... là gạch, tường, nền nhà…, thứ tạo nên vẻ đẹp và tạo ra ngôi nhà hoàn chỉnh.

=> Spring Boot sống chung với các công nghệ như Spring MVC, Spring REST…

Q2: Spring Boot chạy code nhanh hơn Spring truyền thống ?

No.

- Spring Boot sử dụng Spring
- Spring Boot giúp đơn giản hóa cấu hình cho dự án sử dụng Java Spring

Tradeoff:

+ Đơn giản hóa quá trình setup
+ Hiệu năng thay đổi không đáng kể

Q3: Có cần thiết phải sử dụng IDE nào đặc biệt để code Java Spring không ?

No.

Sử dụng IDE nào mà bạn thấy "thoải mái" với nó

Cá nhân mình: VSCode, vì code cả Frontend lẫn Backend trên cùng 1 IDE :v

#19. Tài liệu của Java Spring ?
Lưu ý về cách học
Về tài liệu chính thống của Java Spring, tham khảo tại:
https://spring.io/projects

Cách đọc & học 1 dự án Spring:
Bước 1: Chọn project

Bước 2: Đọc tài liệu liên quan tới project

Bước 3: Thực hành : code, google, github…

Fact: Tài liệu Spring ít dành cho beginner, đa phần bạn sẽ cần đọc code (source code của thư viện, của người khác), thực hành để biến thành kinh nghiệm của bản thân

#20. Quá nhiều khái niệm liên quan tới Spring ?

Question: Anh ơi, em muốn học Java Spring, vì em nghe nói công nghệ đấy, sao anh đề cập nhiều cái liên quan tới Spring quá,

như là Spring Framework, Spring Boot, Spring MVC, Spring Security, Hibernate...

Trả lời:
Bản chất, Java Spring là 1 hệ sinh thái, được liệt kê thông qua các projects của nó: https://spring.io/projects/

Tương tự với câu hỏi: bạn có biết Vin không ?
- VinHome => làm bất động sản
- VinMart => siêu thị
- VinFast => làm ô tô
...

=> Mỗi 1 công cụ "có chữ Spring" là nằm trong hệ sinh thái của Spring, và phục vụ 1 mục đích khác nhau

Spring Boot: làm đơn giản hóa quá trình cấu hình và chạy dự án Spring

Spring MVC: áp dụng mô hình MVC (model-view-controller)

Spring Data (Hibernate/JPA): thao tác với dữ liệu của database hiệu quả

Spring Security: tăng tính security cho dự án của bạn
....

=> việc học Java Spring, là cách chúng ta "học lần lượt các công cụ" trong hệ sinh thái của nó.

Cái khó, là chúng ta sắp xếp chúng như thế nào, và ứng dụng nó ra làm sao trong dự án thực tế, đấy chính là lý do khóa học này ra đời :v

Chapter 4: Maven & Cấu trúc dự án Spring Boot

Tác dụng của Maven và cấu trúc dự án Spring Boot

#21. Các thư mục viết code

- **src** : thư mục chứa code "chính" của dự án

- **resources** : chứa nguồn tài nguyên của dự án như frontend (html, css, js, images), file cấu hình…

- **test**: thư mục viết code test

- **target**: mã nguồn java sau khi được dịch (JVM sẽ chạy thư mục target)

- **JRE system library**: các thư viện đã được cài đặt sẵn khi cài Java

- **maven dependencies**: các thư viện chúng ta tự cài đặt, và dùng maven để quản lý

#22. Spring Build Tool và Maven

1. Why Maven ?

Có rất nhiều cách để "chạy" ứng dụng java thực tế, sử dụng công cụ như:
Gradle - Groovy
Gradle - Kotlin
Maven
Ant...

Maven được sử dụng rộng rãi và có rất nhiều dự án "đã sử dụng Maven"
=> khóa học này dùng Maven, các khóa tiếp theo sẽ dùng Gradle :v

2. Maven là gì ?

Tài liệu: https://maven.apache.org/what-is-maven.html#mavens-objectives

- Mục tiêu của Maven:
+ Làm đơn giản hóa quá trình build dự án (build để deploy production)
+ Cung cấp 1 cấu trúc build thống nhất, rành mạch, rõ ràng => nâng cao chất lượng cho dự án
+ Tạo ra 1 bộ khung để developer follow (best practice)

Maven KHÔNG PHẢI LÀ trang "tài liệu" cho các thư viện

=> hiểu 1 cách đơn giản: Maven giúp build dự án cho java.

#23. Maven files
Bạn không cần thiết phải "cài đặt maven" cho dự án Spring.

Nếu bạn muốn sử dụng "maven" global (tại bất cứ nơi nào trên máy tính) => cần cài đặt

Trong dự án của spring có 2 file:
mvnw => dành cho macos
mvnw.cmd => dành cho windows

=> nhờ có 2 file này, chúng ta có thể sử dụng các câu lệnh của maven mà không cần cài đặt thủ công

//demo chạy dự án:
./mvnw spring-boot:run

#24. The POM file
Tài liệu:
https://maven.apache.org/guides/introduction/introduction-to-the-pom.html

POM (Project Object Model), hông phải porn ha :v
=> dịch nghĩa ra không có tác dụng, điều quan trọng chúng ta hiểu nó dùng để làm gì.

Với dự án Spring, khi khởi tạo đã có sẵn file pom.xml

tiền tố .xml là cú pháp để gõ code, tương tự như html

POM dùng để làm gì ?
- POM được maven sử dụng để xử lý project (compile/build & run)

- **Nếu bạn dùng nodejs, POM tương tự như file package.json**, giúp chúng ta:
+ định nghĩa các thư viện sử dụng

+ cách build project...

#25. Cú pháp đặt tên thư viện/project với Spring

groupId, artifactId => unique project all across the world
If:

groupId = com.example => unique id (reverse domain name)
artifactId=myservice => project's name
version=1.0

Then in the repository you can expect to see the following directory layout:

com/example/myservice/1.0/myservice-1.0.jar (and Pom.xml)

Chapter 5: Spring và Inversion of Control (IoC)
Áp dụng IoC và Dependencies Injection cho dự án Spring

#26. My Goal ?
Tại sao là My goal , mà không phải 'your goal' ?

Demo dự án jhipster (fullstack):
https://gitlab.com/public-starter-projects1/000-java/01-java-spring-mvc/02-java-react-with-jhipster

#27. Kiến trúc của Spring Framework

Tài liệu: https://docs.spring.io/spring-framework/docs/3.0.x/spring-framework-reference/html/overview.html

spring container:
- part of the core Spring framework
- responsible for managing all the beans
- perform dependency injection

#28. Spring Beans

Spring beans: an instance of a class managed by the Spring container

print list of beans:

```
ApplicationContext abc = SpringApplication.run(Demo1Application.class, args);
for(String s: abc.getBeanDefinitionNames())
{
        printf(s)
}
```

#29. Khái niệm Component và Component Scan

//todo

#30. Inversion of Control và Dependency Injection
Injecting dependencies (IoC)

Dependency injection (DI) : nghe có vẻ "phức tạp" và thiên hướng kỹ thuật/design-pattern.

Bằng cách áp dụng DI, code sẽ đơn giản hơn, dễ hiểu và dễ test

DI hoạt động như thế nào ?
Tất cả các ứng dụng phức tạp hơn "Hello World", đều được tạo nên từ 2 or nhiều class khác nhau.

Các object có thiên hướng phụ thuộc vào nhau (tạo ra sự ràng buộc) => khó để test

#31. Viết code “tightly coupled” và “loosely coupled”

Tài liệu:
https://stackoverflow.com/questions/226977/what-is-loose-coupling-please-provide-examples

#32. Injector

//todo

Giải thích về cơ chế của spring

Chapter 6: Spring Security
Cấu hình security cho dự án Spring với Spring Security

#33. Spring Boot Devtool
Vấn đề đang tồn đọng:
Mỗi lần muốn code xong, chúng ta phải làm "thủ công" (manually) hai bước:
Bước 1: Nhấn nút lưu source code (Ctrl + S)
Bước 2: Nhấn nút restart server để cập nhật source code mới nhất

=> sử dụng Devtool để mỗi lần lưu code => project tự động chạy lại (đỡ phải nhấn nút restart :v)

Cài đặt:
https://docs.spring.io/spring-boot/docs/3.2.2/reference/html/using.html#using.devtools

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <optional>true</optional>
</dependency>
```

=> cần restart lại server để nhận cấu hình mới

//todo
- Tạo route /user
- Tạo route /admin

#34. Lựa chọn database nào ?
Database là nơi lưu trữ dữ liệu cho dự án.

Lưu ý, bạn nên có kiến thức nền tảng về database SQL trước khi học khóa học này, đây là kiến thức mà trường đại học đã truyền đạt cho bạn (ví dụ môn học cơ sở dữ liệu)

Nếu chưa biết SQL là gì, học nhanh tại đây: https://www.w3schools.com/sql/

1. Phân loại database
Có 2 loại database chính hay gặp: SQL và NoSQL

Loại 1: SQL, hay còn gọi là cơ sở dữ liệu quan hệ. (relational database)
Các đại diện tiêu biểu như:
- MySQL : sử dụng qua MySQL Workbench, XAMPP
- **Postgres**
- Oracle database
- SQL Server: Microsoft SQL Server..

Loại 2: NoSQL, còn được gọi là cơ sở dữ liệu phi quan hệ (non-relational database)
Đại diện tiêu biểu:
- MongoDB
- Redis..

2. Sử dụng loại database nào
Đa phần các trường đại học sẽ "tập trung" vào SQL (cơ sở dữ liệu quan hệ) và "giới thiệu" về NoSQL

Hầu hết các bài toán trong thực tế được giải quyết với SQL (mà không phải NoSQL)

Chỉ nên dùng NoSQL khi bạn "chưa xác định được" data bạn sẽ lưu.

Recommend: sử dụng Java với SQL

#35. Setup Mysql Workbench

Link tài liệu hướng dẫn:

- Windows: https://dev.mysql.com/doc/refman/5.7/en/windows-installation.html
- MacOS: https://dev.mysql.com/doc/refman/5.7/en/macos-installation.html

Link hướng dẫn dành cho MacOS: https://www.youtube.com/watch?v=2cvH0HRjZF8

Link tải file cài đặt:
https://dev.mysql.com/downloads/installer/

Download file cài đặt sử dụng trong video (windows):
https://drive.google.com/file/d/1RvOJDRoSEqrxD_w_OLwxJ2c4h6Y6kRl6/view

Lưu ý: version MySQL mình cài đặt (link drive) là 8.0.37

Khi cài đặt, các bạn chọn full options (cài đặt full), nó hơi khác với video (mình quay video là version 8.0.35).

Trong quá trình cài đặt (nếu có lỗi xảy ra), ví dụ như không initial database..., các bạn fix bằng cách chủ động tải version mới nhất về xem có bị lỗi không nhé.

#36. Setup Mysql với Spring

Tài liệu:
https://spring.io/guides/gs/accessing-data-mysql/

1.Setup dependencies

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
   <groupId>mysql</groupId>
   <artifactId>mysql-connector-java</artifactId>
   <version>8.0.33</version>
</dependency>
```

2. Setup properties

```
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://${MYSQL_HOST:localhost}:3306/db_example
spring.datasource.username=springuser
spring.datasource.password=ThePassword
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
#spring.jpa.show-sql: true
```

3. Check kết nối tới database

Lưu ý: mỗi lần chạy dự án, cần chạy database trước. Và tập thói quen đọc lỗi khi không chạy được project

#37. Tổng quan về các kiến thức sẽ học
1.Các khái niệm liên quan tới security
- **Authentication**

- **Authorization**

2. Spring Security Filter Chain (tương tự middleware, can thiệp vào req, res)

3. Các dạng security:
- Form authentication
- Basic authentication
- jwt authentication

4. Các lỗi security thường gặp
- CSRF, CORS...

#38. Authentication vs Authorization

Ứng dụng Facebook, ví dụ chức năng newfeed (bản tin) có:
- Nguồn tài nguyên (resources):
+ bài post của bạn, của friends, của group, của page...
+ comment
+ reactions: like, angry, haha...

- Muốn thao tác với 1 chức năng nhất định, chúng ta "bị kiểm soát" quyền hạn.
Ví dụ: bạn không thể xóa/edit comment "của friends" (comment không phải của bạn)

Bất cứ hệ thống nào, khi liên quan tới vấn đề "bảo vệ tài nguyên (data)", đều cần trả lời 2 câu hỏi:

1. Authentication

Trả lời cho câu hỏi: Who are you ? Bạn là ai ?
Đối với ứng dụng Facebook, bạn cần "đăng nhập" thì mới có thể "định danh" bạn là ai.

Nếu bạn không đăng nhập, bạn là "guess" (khách), ai cũng giống nhau, không thể biết bạn là ai.

Vì vậy, quá trình "authentication" thường gắn liền với bước đăng nhập vào hệ thống.

Các cách authentication hay gặp:
- sử dụng **username/password** (bạn dùng email/số điện thoại + mật khẩu để đăng nhập facebook)
- sử dụng biometric: xác thực vân tay, vẽ pattern, xác thực khuôn mặt...

2. Authorization

Sau khi bạn hệ thống đã biết bạn là ai, câu hỏi tiếp theo cần trả lời, là What can I do ? (bạn có thể/không thể làm gì)
Ví dụ, bạn đã đăng nhập thành công vào tài khoản facebook (authentication):
+ Bạn có thể tạo mới/update/delete bài post/comments của chính bạn
+ Bạn có thể xem (read) post/comments của friends
+ Bạn "không thể" update/delete post/comments của friends

=> Quá trình xác định "bạn có thể/không thể" làm gì, gọi là authorization.

Thông thường, quá trình này cũng gắn liền với việc login vào hệ thống.

Khi bạn đã "login thành công", hệ thống đã biết được bạn "được phép" làm gì :v

#39. Nguyên tắc thiết kế Security cho hệ thống web

1. Trust nothing

Bạn không nên "tin bất cứ ai" với thế giới internet
- validate every request : luôn luôn xác thực tất cả request
- luôn xác thực "data" gửi vào hệ thống

2. Xác định quyền hạn cho hệ thống

- Bất cứ hệ thống nào muốn "an toàn", cần được chú ý tới security đầu tiên

- Trước khi bắt tay vào "code" hệ thống, bạn cần xác định hệ thống có những actor nào, và mỗi actor có thể làm gì ?
=> chức năng login cần được code đầu tiên trong hệ thống (nếu như hệ thống có phân quyền người dùng)

Ngoài ra, có 1 vài nguyên tắc sau này bạn có thể tuân theo: (khi làm thực tế)

- Nên có nhiều lớp security
+ firewall, middleware...

- Kiến trúc của security nên đơn giản => dễ maintain
+ hệ thống càng đơn giản thì càng dễ bảo vệ

#40. Setup Spring Security
Tài liệu: https://docs.spring.io/spring-security/reference/getting-spring-security.html

Cài đặt:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

Có thể kiểm tra = cách xem file jars đã được download về chưa ?

Xem logs:
Mặc định, app được bảo vệ với **DefaultSecurityFilterChain**, bao gồm:

```
[
   security.web.session.DisableEncodeUrlFilter@629984eb,
   security.web.context.request.async.WebAsyncManagerIntegrationFilter@7b1e5e55,
   security.web.context.SecurityContextHolderFilter@69aa7d76,
   org.springframework.security.web.header.HeaderWriterFilter@5b5b59,
   org.springframework.web.filter.CorsFilter@4c1d59cd,
   org.springframework.security.web.csrf.CsrfFilter@318c68d5,
   org.springframework.security.web.authentication.logout.LogoutFilter@4733f6f5,
   security.web.authentication.UsernamePasswordAuthenticationFilter@56cc9f29,
  security.web.authentication.ui.DefaultLoginPageGeneratingFilter@41ccb3b9,
  security.web.authentication.ui.DefaultLogoutPageGeneratingFilter@76cf841,
security.web.authentication.www.BasicAuthenticationFilter@661fe025,
 security.web.savedrequest.RequestCacheAwareFilter@f1266c6,
   security.web.servletapi.SecurityContextHolderAwareRequestFilter@3913f206,
  security.web.authentication.AnonymousAuthenticationFilter@297c9a9b,
   org.springframework.security.web.access.ExceptionTranslationFilter@6baf25d7,
   org.springframework.security.web.access.intercept.AuthorizationFilter@415ef4d8
]
```

Sau khi cài đặt, auto có:
- /login , /logout (mọi thứ được bảo vệ mặc định)
- password printf in console

#41. Spring Security Overview
Spring security:
- dùng để bảo vệ (protect) web applications, REST API, microservices.

- Có thể gặp khó khăn khi bắt đầu, vì đề cập tới nhiều khái niệm mới:
+ filter chain
+ authentication managers
+ authentication providers

Đổi lại:
- Spring Security dễ sửa đổi (flexible)
- Everything is protected (mặc định, mọi resources trong hệ thống được bảo vệ bởi spring security)

tham khảo: https://backendstory.com/spring-security-authentication-architecture-explained-in-depth/

1. Spring MVC hoạt động như thế nào ?

Request => Dispatcher Servlet => Controllers... => Response
(client)

Dispatcher Servlet : là 1 lớp "đứng giữa", phụ trách nhiệm vụ trỏ tới đúng controller/method để xử lý request

2. Spring MVC + Spring Security

Request => Spring Security => Dispatcher Servlet => Controllers... => Response

- Tất cả request cần đi qua Spring Security đầu tiên

#42. Spring Security hoạt động như thế nào ?

Nếu bạn đã học backend Nodejs, Spring Security chính là "middleware" của hệ thống.

Spring Security bảo vệ hệ thống thông qua các method "đặc biệt", gọi là filter.

Các chức năng Spring Security cung cấp:

- Authentication : Is a valid user (ví dụ: BasicAuthenticationFilter)
- Authorization: User có thể làm gì (ví dụ: Authorization Filter)

- Các tính năng khác:
+ Bảo vệ CORS (Cross-Origin Resource Sharing): CorsFilter

+ Bảo vệ CSRF (Cross Site Request Forgery): CsrfFilter

+ Login Page, Logout Page được cung cấp sẵn

...

Thứ tự của Filter là "quan trọng". Bạn hình dung nó gọi là chainFilter, có nghĩa là được thực hiện tuần tự, ưu tiên như sau:

1. Check bảo mật với các tấn công hay gặp
Basic Check Filter: CORS, CSRF...

2. Xác thực người dùng là ai ?
Authentication Filter

3. Xác định người dùng có thể làm gì ?
Authorization Filter
....

Chapter 7: Spring MVC
Áp dụng mô hình MVC cho dự án Spring

#43. Mô hình MVC là gì ?
Format XML cài thêm extension: **XML Tools**

Lưu ý về fix mysql và version dependencies:
https://stackoverflow.com/questions/76305054/mysql-connector-j-vs-mysql-connector-java-maven-dependency-differences

MVC là viết tắt của Model - View - Controller, là một mô hình sử dụng rất phổ biến trong lập trình website

1. Why ?
Một dự án thực tế, cần tổ chức code theo một cấu trúc nhất định, đảm bảo các nguyên tắc:
- Có khả năng mở rộng (code thêm tính năng)
- Có khả năng bảo trì (maintain)

2. What ?

View: chịu trách nhiệm render (hiển thị) giao diện website (cái người dùng nhìn thấy)

Model: các đối tượng sử dụng hệ thống được mô hình hóa qua OOP (lập trình hướng đối tượng). Với Spring, Models bao gồm các tables tại databases

Controller: chịu trách nhiệm xử lý data. Lấy dữ liệu từ Model, xử lý, và đưa cho View hiển thị

3. How ?
Với Spring Framework, mô hình MVC được tích hợp thông qua “web modules”

#44. Java Annotation

1.Khái niệm Annotation

Về annotation : https://stackoverflow.com/questions/2798181/what-are-annotations-and-how-do-they-actually-work-for-frameworks-like-spring

Annotation (chú thích), là cách chúng ta "trang trí" class của Java. (decorator)
Mục đích là "tăng thêm sức mạnh" cho class đấy (meta-data)

Khi sử dụng Java Spring, thông thường, rất hiếm khi chúng ta định nghĩa thêm annotation, đa phần là sử dụng lại các annotation mà framework đã định nghĩa sẵn.

Khi sử dụng annotation, code của chúng ta ngắn hơn, từng minh và rõ ràng (dấu toàn bộ logic/magic phía sau annotation)

2. Cấu hình Username/Password

https://docs.spring.io/spring-boot/docs/current/reference/html/application-properties.html#appendix.application-properties.security

//update file : application.properties

```
#config spring security
spring.security.user.name=hoidanit
spring.security.user.password=123456
```

Disable Security:

Lưu ý: chỉ disable tạm thời để không cần nhập username/password

Update file LaptopshopApplication.java, cập nhập annotation như sau:

Cách 1:

```
@SpringBootApplication(exclude =
org.springframework.boot.autoconfigure.security.servlet.SecurityAutoConfiguration.class)
```

Nếu cách 1 không hoạt động, ae thử cách này nhé:

```
@SpringBootApplication(exclude = {

        org.springframework.boot.autoconfigure.security.servlet.SecurityAutoConfiguration.class,

        org.springframework.boot.actuate.autoconfigure.security.servlet.ManagementWebSecurityAutoConfiguration.class
})
```

Tham khảo: https://stackoverflow.com/a/63540383

Và, đặc biệt quan trọng, không thêm dependencies lung tung vào file pom.xml (ví dụ như khi vscode gợi ý chẳng hạn)

// không tự động thêm như này nhé (xóa ngay và luôn, để đảm bảo code của bạn và video giống nhau)

```
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>
```

#45. Áp dụng mô hình MVC

Tài liệu:
https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html

https://gitlab.com/public-starter-projects1/000-java/01-java-spring-mvc/02-java-react-with-jhipster

1. Tạo Controller và Services

Về các annotation hay dùng với MVC: https://www.baeldung.com/spring-mvc-annotations

@Controller

@Services

#46. Render view HTML
Tài liệu:
https://docs.spring.io/spring-framework/reference/web/webmvc/mvc-config/static-resources.html

1. Khái niệm file "tĩnh"

Static: tĩnh
Dynamic: động

File "tĩnh" là các file lưu trong hệ thống "ít thay đổi", và được "public" để truy cập từ bên ngoài

Có nghĩa rằng, cho dù bạn lưu trữ file đấy bên trong source code => người dùng internet vẫn có thể truy cập.

File static hay gặp: images, html, css, js

2. Sử dụng static file

Bước 1: sử dụng **@Controller** để báo hiệu đây là một class Controller (trong mô hình MVC)

Bước 2: Tạo View
View là file html dùng để hiển thị dữ liệu

Tạo thư mục : src/main/resources/**static**

Tạo file html tương ứng trong thư mục static, ví dụ : hello.html

Bước 3: Controller return ra tên của view muốn hiển thị

Lưu ý: do là file "tĩnh" => có thể truy cập từ ngoài internet
http://localhost:8080/hello.html

#47. View Engine là gì

1.Vấn đề còn tồn đọng

Đối với "**static**" file, nội dung luôn không đổi. Chúng ta cần code từ A tới Z
(static là file tĩnh)

Điều chúng ta muốn:

- Code cần ít đi (ngắn lại), code chay HTML "full page" rất dài và khổ
- Dữ liệu trang html cần "**dynamic**", vì đôi khi data sẽ thay đổi

2. View Engine là gì

View Engine là "công cụ" giúp chúng ta code ít đi, và hiệu quả hơn so với việc code HTML truyền thống.

View engine giúp giải quyết bài toán "dynamic" page.

Ví dụ: chúng ta muốn:

- Truyền data từ controller sang view (html) để render
- Sử dụng câu điều kiện, sử dụng vòng lặp… bên trong view

Với Spring, chúng ta có : Spring View Technologies, bao gồm:
https://docs.spring.io/spring-framework/reference/web/webmvc-view.html

Java Server Pages (JSP), Thymeleaf, Groovy, FreeMarker, Jade…

Trong khóa học này, mình sử dụng JSP vì: cách tư duy nó tương tự một vài công cụ khác, ví dụ:
EJS (của Nodejs)
Blade (của Laravel)

Và, khi bạn đã học được 1 view engine, các công cụ khác sẽ học tương tự (quan trọng nhất là cách tư duy)

Sau khi đã làm quen với JSP, dự án kế tiếp, mình gợi ý bạn học Thymeleaf nhé :v
https://www.thymeleaf.org/doc/articles/thvsjsp.html

#48. Setup JSP

Tài liệu:
https://howtodoinjava.com/spring-boot/spring-boot-jsp-view-example/

https://www.baeldung.com/spring-boot-jsp

1.Setup Dependencies

https://docs.spring.io/spring-boot/docs/current/reference/html/dependency-versions.html

```
<dependency>
   <groupId>org.apache.tomcat.embed</groupId>
   <artifactId>tomcat-embed-jasper</artifactId>
</dependency>

<dependency>
  <groupId>jakarta.servlet.jsp.jstl</groupId>
  <artifactId>jakarta.servlet.jsp.jstl-api</artifactId>
</dependency>

<dependency>
  <groupId>org.glassfish.web</groupId>
  <artifactId>jakarta.servlet.jsp.jstl</artifactId>
</dependency>
```

2. Config view folder

```
spring.mvc.view.prefix=/WEB-INF/view/
spring.mvc.view.suffix=.jsp
```

Tạo folder **config**
Ý tưởng: https://gitlab.com/public-starter-projects1/000-java/01-java-spring-mvc/02-java-react-with-jhipster

```
@Configuration
@EnableWebMvc
public class WebMvcConfig implements WebMvcConfigurer {

  @Bean
  public ViewResolver viewResolver() {
    final InternalResourceViewResolver bean = new InternalResourceViewResolver();
    bean.setViewClass(JstlView.class);
    bean.setPrefix("/WEB-INF/view/");
    bean.setSuffix(".jsp");
    return bean;
  }

  @Override
  public void configureViewResolvers(ViewResolverRegistry registry) {
    registry.viewResolver(viewResolver());
  }
}
```

#49. JSTL trong View
Tài liệu:
https://docs.spring.io/spring-framework/reference/web/webmvc-view/mvc-jsp.html

1.Về cú pháp của JSP
Tham khảo:
https://www.tutorialspoint.com/jsp/jsp_syntax.htm

Định nghĩa model:
https://stackoverflow.com/questions/18486660/what-are-the-differences-between-model-modelmap-and-modelandview

2.Về JSTL

JSTL là viết tắt của **JSP Standard Tag Library**

Chúng ta cần những tag này để "code java" bên trong file HTML, ví dụ như sử dụng câu điều kiện, vòng lặp…
=> cung cấp thêm sức mạnh cho cú pháp của JSP

Bắt đầu file html:
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

A page directive (<%@page ... %>) sets the content type returned by the page.

Tag library directives (<%@taglib ... %>) import custom tag libraries.

#50. Tích hợp Bootstrap và JQuery

1. Sử dụng online

```
<!-- Latest compiled and minified CSS -->
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet">

<!-- Latest compiled JavaScript -->
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></
script>


<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
```

2. Sử dụng CSS/JS

https://stackoverflow.com/questions/25098413/whats-the-diff-btween-src-main-resources-and-src-main-webapp-resources

Tạo webapp/**resources**

https://gitlab.com/public-starter-projects1/000-java/01-java-spring-mvc/02-java-react-with-jhipster

Tham khảo: https://www.baeldung.com/spring-mvc-static-resources

```
  @Override
  public void addResourceHandlers(ResourceHandlerRegistry registry) {
     registry.addResourceHandler("/css/**").addResourceLocations("/resources/css/");
  }
```

#51. Bài Tập Giao diện Tạo Mới (CREATE) User

1. Yêu cầu

- Tạo đối tượng User, bao gồm các thông tin sau:
  id: long
  email: String
  password: String
  fullName: String
  address: String;
  phone: String

  Viết **getter**, **setter**, **toString**

- Tạo **view** create new user, sử dụng Bootstrap và có các thông tin sau:
  email, password, fullName, address, phone

2. Gợi ý cách làm

Tạo thêm folder, đặt tên là **domain -> User.java**

Với view, chia thành 2 folder: client/admin

Folder admin: lưu giao diện xử lý tại admin
Folder client: lưu giao diện xử lý tại user (client)

Tạo thêm view **admin/user/create.jsp**

Tạo thêm url **/admin/user**

Về giao diện form, sử dụng:
https://getbootstrap.com/docs/5.3/forms/overview/#overview

Chia layout nhanh với: Container, Row, Col của bootstrap
https://getbootstrap.com/docs/5.3/layout/breakpoints/

#52. Chữa Bài Tập Giao diện Tạo Mới (CREATE) User

Source code video này: (#51)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=sharing

#53. Send data với HTML Form (Submit Form)

Bước 1: Tạo form html
https://www.w3schools.com/html/html_forms_attributes.asp

HTML form gồm 2 attributes: **method** và **action**

Bước 2: Xử lý method POST
@RequestMapping(value = "/admin/user", method = RequestMethod.POST)

Bước 3: Tạo form với JSTL
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>

Tham khảo: https://www.baeldung.com/spring-mvc-form-tutorial

Lưu ý về **@ModelAttribute** Annotation : là cách chúng ta convert dữ liệu từ **View** trả cho **controller** xử lý

Chapter 8: Spring Data với JPA và Hibernate
Thao tác xử lý data với ORM (Object Relational Mapping)

#54. Spring Data
Tài liệu: https://spring.io/projects/spring-data

1. Về Spring Data
Spring Data là công cụ giúp xử lý data (dữ liệu của ứng dụng) trong hệ sinh thái của Spring
Mục tiêu của **Spring Data là làm đơn giản hóa quá trình truy cập và sử dụng data**

- Hỗ trợ đa dạng loại database: SQL (MySQL, Postgres…) , NoSQL (MongoDB)... https://spring.io/projects/spring-data

- Cung cấp đối tượng "Repository" để truy cập database
  Lập trình viên không cần biết câu lệnh truy vấn, vẫn thao tác được với database

2. Spring Data JPA
JPA (Java Persistence API) : https://docs.spring.io/spring-data/jpa/reference/index.html

Cài đặt:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

https://github.com/spring-projects/spring-boot/blob/main/spring-boot-project/spring-boot-starters/spring-boot-starter-data-jpa/build.gradle

#55. Cách Read/Write Data
Với database, chúng ta có 2 thao tác : READ (đọc dữ liệu) và WRITE (update/insert/delete), tức là ghi/sửa đổi dữ liệu

Java là một ngôn ngữ hướng đối tượng, nên mình lựa chọn cơ sở dữ liệu quan hệ (relational database)
Về cơ sở dữ liệu phi quan hệ (non-relational database), tham khảo lộ trình node.js: https://hoidanit.vn/khoa-hoc?type=nodejs

Để thực hiện điều trên, chúng ta có 2 cách:

1.Sử dụng JDBC
JDBC: Java Database Connectivity

Là cách chúng ta **kiểm soát việc sử dụng database thông qua câu lệnh truy vấn**, ví dụ truy vấn SQL

Eg: all Users = **select * from users**
All Users with name === Eric: **select * from users where name = 'Eric'**

2. Sử dụng ORM/ODM
ORM: Object Relational Mapping

Là cách chúng ta **truy cập/sử dụng database thông qua các object định nghĩa trong code**

Eg: all Users = **findAll( )**
All Users with name === Eric: findAllByName('Eric')

Ưu điểm:
- Bạn không cần biết câu lệnh truy vấn SQL
- Gần với ngôn ngữ tự nhiên

Nhược điểm: hiệu năng không phải là 100%. Nếu bạn muốn kiểm soát tối đa hiệu năng, hãy dùng câu lệnh SQL

#56. Hibernate vs Spring Data JPA
Tham khảo: https://www.youtube.com/watch?app=desktop&v=4Py9RTVWyvE

https://docs.jboss.org/hibernate/orm/6.4/introduction/html_single/Hibernate_Introduction.html#introduction

#57. Hướng Dẫn Kỹ Thuật Debug Java với VSCode
//todo

#58. Entity
Tài liệu:
https://www.baeldung.com/learn-jpa-hibernate
https://spring.io/guides/gs/accessing-data-jpa

1. Ý tưởng
Lập trình hướng đối tượng xoay quanh “objects”

Vậy có cách nào để thao tác với database, thông qua “đối tượng” và không cần phải sử dụng câu lệnh truy vấn ?

2. Entity
https://www.baeldung.com/jpa-entities

Entity : thực thể ~ actor (các tác nhân tham gia vào hệ thống)

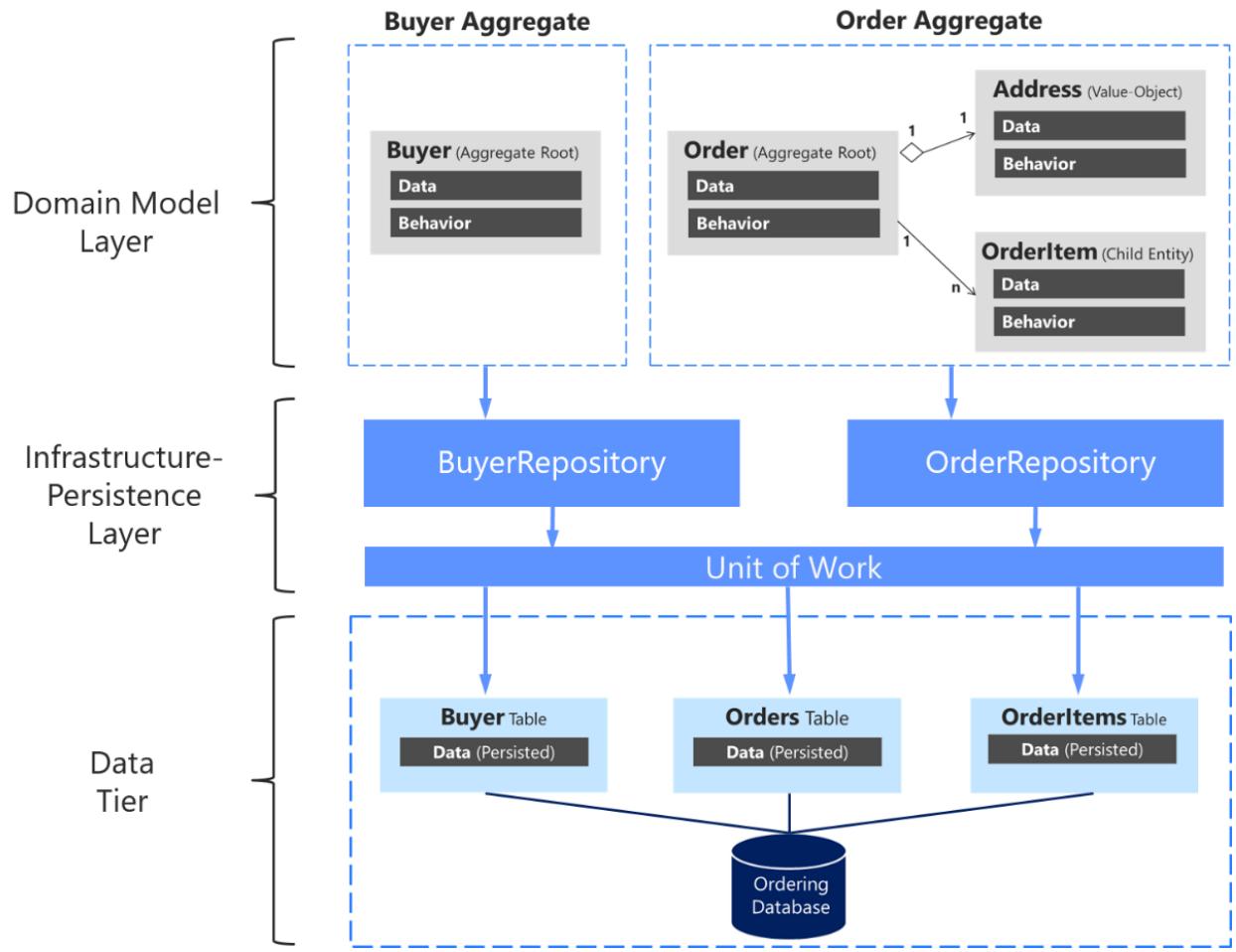
Bản chất Entity là 1 class của Java, chỉ có điều khác biệt, là nó đại diện cho 1 table tương ứng trong database

=> Entity chính là model trong mô hình MVC

#59. Mô hình MVC áp dụng với Spring Data

1.Domain-driven design
Tài liệu: https://en.wikipedia.org/wiki/Domain-driven_design
Ứng dụng chúng ta xoay quanh các đối tượng (object/model)
=> đây là lý do tại sao đặt tên thư mục là **domain**

2. Repository Pattern
Tài liệu: https://learn.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/infrastructure-persistence-layer-design



#60. Repository
Tài liệu: https://docs.spring.io/spring-data/jpa/reference/jpa/getting-started.html

Hoàn thiện tính năng tạo mới User (lưu ý chưa xử lý password)
Sử dụng Service và viết code theo Dependency Injection

Source code video này: (#60)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=sharing

#61. Định nghĩa Repository Query

Tài liệu:
https://docs.spring.io/spring-data/jpa/reference/repositories/query-keywords-reference.html
https://docs.spring.io/spring-data/jpa/reference/repositories/query-methods-details.html

Mục tiêu: Lấy danh sách tất cả users

Ví dụ:

```
//repository
@Repository
public interface UserRepository extends CrudRepository<User, Long> {
    Optional<User> findOneByActivationKey(String activationKey);

    List<User> findAllByActivatedIsFalseAndCreatedDateBefore(Instant dateTime);

    Optional<User> findOneByResetKey(String resetKey);

    Page<User> findAllByIdNotNullAndActivatedIsTrue(Pageable pageable);

}
```

Bước 1: định nghĩa tên method theo các quy tắc

Bước 2: Spring sẽ tự động chuyển "tên method" thành câu lệnh truy vấn database

Bước 3: nhận kết quả trả ra từ Repository

Lưu ý: không có sự khác biệt giữa findOne hay findBy hay findAll
https://www.baeldung.com/spring-data-jpa-findby-vs-findoneby

Sự khác biệt đến từ kiểu trả về của method

#62. Bài tập design table Users

Yêu cầu:

Sửa đổi thành 2 URL:

http://localhost:8080/admin/user/create
=> hiển thị form tạo mới User

http://localhost:8080/admin/user
=> hiển thị table Users

Gợi ý:

Sử dụng table:
https://getbootstrap.com/docs/4.0/content/tables/#bordered-table

#63. Chữa bài tập design table users

Source code video này: (#62)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=sharing

#64. Chức năng hiển thị danh sách User

```
<c:forEach var="weather" items="${weathers}">
  <tr>
    <td>${weather[0]}</td>
    <td>${weather[1]}</td>
    <td style="text-align: center">${weather[2]}℃</td>
    <td style="text-align: center">${weather[3]}℃</td>
  </tr>
</c:forEach>
```

#65. Tổng kết về mô hình MVC với Spring Data

Các thành phần tham gia vào mô hình MVC với Spring :

M: model, trong dự án này, chúng ta **code tại thư mục domain** (domain-driven design) Model sẽ chịu trách nhiệm quản lý các "đối tượng/tác nhân" tham gia vào hệ thống, chính là các tables lưu trữ tại database

V : view, **là các file jsp** (html), tạo ra giao diện cho người dùng nhìn thấy

C : controller, **code tại thư mục controller**, có tác dụng điều hướng hoạt động của website

Mối quan hệ giữa các thành phần:

Bước 1: người dùng truy cập vào 1 route (url tại website), ví dụ localhost:8081**/abc**
abc là url người dùng "muốn truy cập"

Bước 2: Server java nhận được yêu cầu của người dùng, sẽ quét qua tất cả controller để tìm ra controller nào phụ trách xử lý yêu cầu trên. Nếu có (chuyển bước 3), còn không sẽ trả ra lỗi 404 not found

Bước 3: Nếu cần lấy dữ liệu, **controller sẽ gọi -> service -> gọi repository** để lấy data

Bước 4: Controller truyền data sang view

Bước 5: View render (tạo ra giao diện) và hiển thị kết quả cho người dùng

#66. Design giao diện xem chi tiết User

Ý tưởng: cần lấy được id của User -> tạo route động ứng với từng id người dùng

Tài liệu: https://www.baeldung.com/spring-pathvariable

https://getbootstrap.com/docs/5.0/components/card/#list-groups

```
<div class="card" style="width: 18rem;">
  <div class="card-header">
    Featured
  </div>
  <ul class="list-group list-group-flush">
    <li class="list-group-item">An item</li>
    <li class="list-group-item">A second item</li>
    <li class="list-group-item">A third item</li>
  </ul>
</div>
```

#67. Bài tập chức năng xem chi tiết User

//Todo

Gợi ý cách làm:

Hiện tại, controller đã lấy động được id của người dùng (ứng với url)

Việc cần làm:

- Lấy data của user, thông qua id
- Truyền data này qua view để hiển thị

Controller -> gọi tới Service -> gọi tới Repository

Bây giờ cần viết 1 method lấy động thông tin user ???

(#61. Định nghĩa Repository Query)

#68. Bài tập design giao diện Update User

Gợi ý cách làm:

Lưu ý: không cập nhật password.

Nếu muốn cập nhật password, sau này sẽ làm tính năng “quên mật khẩu”.
Ở đây, chỉ cập nhập thông tin cơ bản của người dùng

- Về giao diện, copy y nguyên giao diện tạo mới người dùng,
  tuy nhiên, **thay password = id**

- Tạo các route “động” khi nhấn vào view update (lấy được id của người dùng)

#69. Chức năng Update User

Cách làm:

- Lấy user by Id
- Set các thuộc tính cho user
- Lưu user

Phân biệt save vs create (upsert) ?

Upsert = update / insert

#70. Bài tập chức năng chức năng Delete User

Gợi ý cách làm:

- Tạo view xác nhận xóa user ?
  https://getbootstrap.com/docs/5.0/components/alerts/

- Viết method xóa user by id ?
- Nếu xóa thành công, redirect về table list users

Chapter 9: Project thực hành

Phân tích, thiết kế database cho dự án thực hành để hiểu sâu hơn về Spring MVC

#71. Nhìn lại các kiến thức đã học

1. Các công nghệ đã dùng

- Sử dụng Spring Boot để chạy dự án

- Sử dụng Spring MVC để thực hiện mô hình MVC:
    + Model (Entity)
    + View (jsp)
    + Controller

- Sử dụng Spring Data (Hibernate/JPA) để mô hình hóa model và kết nối tới database:
    + Domain-driven design
    + Repository pattern

2. Làm sao để phát triển thêm dự án

- Định nghĩa thêm model (Spring Data)

- Viết logic theo mô hình MVC (Spring MVC)

#72. Yêu cầu dự án thực hành

Đề bài: tạo một website bán laptop

Lưu ý: đây là website thực hành dùng để "học kiến thức", website thực tế sẽ cần phải cải thiện nhiều hơn nữa, tránh hiện tượng ảo tưởng sức mạnh =))

Ý tưởng: có một trang web để hiển thị sản phẩm cho người dùng lựa chọn, các tính năng chính :

- Hiển thị danh sách sản phẩm (trang chủ /homepage)
- Xem chi tiết 1 sản phẩm
- Tìm kiếm sản phẩm theo tiêu chí (giá cả, số lượng, nhà sản xuất...)
- Giỏ hàng
- Thông báo mua thành công

Tham khảo: https://fptshop.com.vn/may-tinh-xach-tay

#73. Phân Tích Model cho dự án

Một dự án thực tế, luôn bắt đầu bằng việc xác định rõ, hệ thống có bao nhiêu loại người dùng sử dụng (actor)

Với dự án thực hành, chia thành 2 loại chính:

1. Người dùng chưa đăng nhập (guest)

Có thể :

- Xem danh sách sản phẩm, xem chi tiết sản phẩm
- Tìm kiếm sản phẩm theo tiêu chí

Không thể:

- Đặt hàng (place order). Vì chúng ta không có thông tin người dùng, hoặc thông tin chưa xác minh. Nếu muốn làm tính năng "lịch sử mua hàng" là không thể

- Thêm/sửa/xóa sản phẩm và thông tin trên website

2. Người dùng đã đăng nhập (user)

Phân chia theo vai trò của người dùng

Vai trò người dùng thông thường: (user)

- Kế thừa lại các tính năng của Guest (người dùng chưa đăng nhập)
- Có thể đặt hàng

- Không có quyền hạn thêm/sửa/xóa sản phẩm và thông tin trên website

Vai trò người dùng quản trị: (admin)

- Kế thừa lại các tính năng của User
- Full quyền trong hệ thống

Lưu ý: một website thực tế sẽ có cơ cấu phân quyền phức tạp hơn, ví dụ chia theo phòng ban, chia theo tổ chức: marketing, sale, it...

Tuy nhiên, đấy là 1 team code nên website đấy (và đôi khi là 1 công ty). Chúng ta đang solo và đặt mục tiêu "học hỏi", vì vậy, chỉ chia cấu trúc đơn giản nhất

#74. Phân Tích Thiết Kế Database

File excel tài liệu trong video:
https://drive.google.com/file/d/1IhPePMNGQiY76cAoMbbkjEOJ4f-kgtWA/view?usp=drive_link

- user
- role
- product
- order

#75. Design Models cho database

File excel tài liệu trong video:
https://drive.google.com/file/d/1IhPePMNGQiY76cAoMbbkjEOJ4f-kgtWA/view?usp=drive_link

#76. Mô hình hóa với Spring Data JPA

Source code video này (#76):
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=sharing

Lưu ý: tải về, update thêm User như video ae nhé :v

Hiện tại, chưa cần chạy code, vì nếu chạy, sẽ có lỗi không tạo được table Order. Lý do tại sao, các bạn cứ tiếp tục xem video, mình sẽ fix tại chương tiếp theo.

Chapter 10: Spring Data JPA Relationships
Áp dụng các mối quan hệ của table database với Entity

#77. Quan Hệ Cho Model - Relationships
Tài liệu:
https://docs.jboss.org/hibernate/orm/6.4/introduction/html_single/Hibernate_Introduction.html#associations

Relationship : mối quan hệ

Association: sự kết hợp/kết nối với nhau

1. Fix lỗi tạo table Order
@Table(name = "Orders")
https://stackoverflow.com/questions/3599803/jpa-hibernate-cant-create-entity-called-order

https://www.w3schools.com/sql/sql_orderby.asp

2. Associations

Có 3 loại quan hệ chính:
one-to-one : quan hệ 1 - 1
one-to-many và **many-to-one** : quan hệ 1 - Nhiều
many-to-many : quan hệ Nhiều - Nhiều

Chia thành 2 hình thức:
Unidirectional (quan hệ 1 chiều /unique) : là quan hệ giữa 2 table, mà trong đó

Bi-directional : quan hệ 2 chiều

The **owning** side : bên sở hữu - bên có foreign key (khóa ngoại)
Reverse side

#78. One-to-Many Relationship

Tài liệu:

https://docs.jboss.org/hibernate/orm/6.4/introduction/html_single/Hibernate_Introduction.html#associations

#79. Áp dụng One-to-Many Relationship

Tài liệu: https://stackoverflow.com/a/66241669

Với quan hệ 2 chiều (bi-directional), cần tuân theo quy luật:
1.The **inverse side** cần sử dụng **mappedBy**
-> the **owner side** không được sử dụng mappedBy (tức là table có FK - khóa ngoại)

Mối quan hệ Users và Roles: N -1

Bước 1: phân tích mối quan hệ
1 user có 1 role và **table user chứa role_id** (owner side)
User -> ManyToOne - lưu là Role
@ManyToOne
private Role role;

1 role có nhiều user -> Role - OneToMany - lưu là List<User>
@OneToMany
List<User> users;

Bước 2: chỉ rõ column mapping

Sử dụng **mappedBy** cho inverse side:
@OneToMany(mappedBy = “role”)
List<User> users;

Sử dụng **joinColumns** cho owner side:
@**Many**ToOne
@JoinColumn(name = "role_id")
private Role role;

Mối quan hệ User và Order: 1 - N

Sử dụng **mappedBy** cho inverse side:
@OneToMany(mappedBy = “user”)
List<Order> orders;

Sử dụng **joinColumns** cho owner side:
@ManyToOne
@JoinColumn(name = "user_id")
private User user;

#80. Many-to-Many Relationship
Tài liệu:
https://www.bezkoder.com/jpa-many-to-many/

https://docs.jboss.org/hibernate/orm/6.4/introduction/html_single/Hibernate_Introduction.html#many-to-many

Quan hệ N - N = N - 1 + 1 - N

#81. Áp dụng Many-to-Many Relationship
Order_Detail:
id: long
order_id: long
product_id: long
quantity: long
price: double

Quan hệ giữa Order và Order_detail
1 order hasMany order_detail (1 - N)
@ManyToOne
@JoinColumn(name = “order_id”)
private Order order;

@OneToMany(mappedBy = “order”)
List<OrderDetail> orderDetails;

Quan hệ giữa Product và Order_detail
1 product có thể thuộc nhiều order_detail (1 - N)
@ManyToOne
@JoinColumn(name = “product_id”)
private Product product;

#82. One-to-One Relationship

Tài liệu: https://vladmihalcea.com/the-best-way-to-map-a-onetoone-relationship-with-jpa-and-hibernate/

Chapter 11 : Module Upload File
Upload file với Java Spring MVC

#83. Design Giao Diện Admin
Source code video này (#83):
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

Template ban đầu:
https://gitlab.com/public-starter-projects1/000-java/01-java-spring-mvc/03-admin-template

Cài thêm extension vscode live server : **Live Server**
Mục đích: mỗi lần nhấn lưu file html, không cần refresh website, browser sẽ tự động update

Các bước test template:
Bước 1: clone source code ở trên

Bước 2: đảm bảo code đang ở nhánh (branch) **master**

Bước 3: chạy file index.html

Với template dùng cho dự án, checkout sang nhánh code mới:
git checkout hoidanit

Việc cần làm:
- Tạo file admin.jsp

#84. Chia Layout Admin

Source code của video này (#84):
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

Tái sử dụng các thành phần:

- Header
- Sidebar
- Footer

Sử dụng jsp để tái sử dụng code:
https://stackoverflow.com/a/60957169

#85. Hoàn thiện Layout Admin

Source code của video này (#85):
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

#86. Bài Tập Design Upload File ?

Source code video này: (#86)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

1. Mục tiêu

Thêm input upload file và select Role

Gợi ý:
https://getbootstrap.com/docs/5.0/forms/form-control/#file-input

https://getbootstrap.com/docs/5.0/forms/select/

Design layout: https://getbootstrap.com/docs/5.0/forms/layout/#gutters

#87. Image với Preview

Tạo roles

INSERT INTO laptopshop.roles (name, description)
VALUES
("ADMIN", "Admin thì full quyền"),
("USER", "User thông thường");

Sử dụng jquery:
https://www.geeksforgeeks.org/preview-an-image-before-uploading-using-jquery/

Lưu ý: cần import Jquery ở đầu file:

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>

```
<script>
    $(document).ready(() => {
        const avatarFile = $("#avatarFile");
        avatarFile.change(function (e) {
            const imgURL = URL.createObjectURL(e.target.files[0]);
            $("#avatarPreview").attr("src", imgURL);
            $("#avatarPreview").css({ "display": "block" });
        });
    });
</script>
```

#88. Nơi nào để lưu trữ file ?

1.Hình thức upload file

Có 2 hình thức upload file : **single** (upload 1 file/lần) và **multiple** (upload nhiều file/lần)

Khi upload multiple files:

- Frontend cần xử lý để truyền lên "multiple" files
- Backend cần xử lý để lưu nhiều file 1 lúc
- Vấn đề sẽ phát sinh nếu quan trọng thứ tự file upload

Khi upload 1 file: frontend gửi 1 file và backend chỉ cần xử lý đúng file đấy (1 file tại 1 thời điểm)

=> **Cách dễ nhất** là upload 1 file. Khi cần upload nhiều file, dùng vòng lặp để upload

2. Nơi nào để lưu trữ file

Hình thức 1: **lưu trữ qua các dịch vụ online,** ví dụ S3 (amazon), cloudflare (R2)
Ưu điểm: truy cập mọi lúc, mọi nơi. Tự động backup (sao lưu dữ liệu)

Nhược điểm: pay as you go (dùng bao nhiêu trả bấy nhiêu, hoặc mua theo gói cước)

Fun fact: Nếu dịch vụ cloud mà rẻ (khi có nhiều data), thì zalo đã không lưu trữ dữ liệu local (tại máy người dùng)

Hình thức 2: **lưu trữ tại database**
Với database mysql , hỗ trợ định dạng blob, clob (khóa học sern youtube)
https://www.youtube.com/watch?v=VvvXhNbFWKY&list=PLncHg6Kn2JT6E38Z3kit9Hnif1xC_9VqI

Ưu điểm: hoàn toàn miễn phí, có thể lưu tối đa khoảng 4GB/file

Nhược điểm: tốc độ IO chậm (read/write). Phải tự backup

Hình thức 3: **lưu trữ tại máy chủ Server (nơi viết code)**
Ưu điểm: miễn phí, tốc độ IO nhanh hơn database (vì lưu raw file)
Nhược điểm: phải tự backup dữ liệu và tối ưu hóa IO

=> khóa học này sử dụng hình thức 3 (vì đơn giản nó tiện & miễn phí)
Bonus: đi làm, sẽ sử dụng ftp server

#89. Upload file với Spring

1. Nguyên tắc khi upload file

Với hình thức lưu trữ file "tại server" (tức là lưu trữ trong folder chứa source code/hoặc folder trong máy tính)

Bước 1: client gửi file lên server

Bước 2: server sẽ lưu file vào thư mục quy định trước

Bước 3: kiểm tra xem client có thể truy cập được file đã upload hay không

Chúng ta cần kiểm tra, trước khi tiến hành code. Vì nếu mọi thứ chạy bình thường, thì chỉ cần code làm sao để file lưu vào đúng folder đấy là được.

2. Tìm hiểu upload file với JSP

Multiple part

Tham khảo:
https://www.digitalocean.com/community/tutorials/spring-mvc-file-upload-example-single-multiple-files

https://spring.io/guides/gs/uploading-files

#90. Hoàn thiện tính năng Upload file (Part 1)
Source code video này: (#90)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

Lưu ý:
Chỉ lưu tên file vào database

Bước 1:
Với role, cần update để mapping với model: (update setter, getter)
https://docs.spring.io/spring-framework/reference/web/webmvc-view/mvc-jsp.html#mvc-view-jsp-formtaglib-selecttag

```
<form:select class="form-select" path="role.name">
        <form:option value="ADMIN">ADMIN</form:option>
        <form:option value="USER">USER</form:option>
</form:select>
```

Bước 2: thêm tag cho form để upload file : **enctype="multipart/form-data"**
<form method="POST" **enctype="multipart/form-data"** action="/">

Bước 3: sử dụng input với **type=”file”, thêm thuộc tính (attribute) name**

```
 <input
        class="form-control"
        type="file"
        id="avatarFile"
        name="hoidanitFile"
        accept=".png, .jpg, .jpeg"
/>
```

Bước 4: Lấy file từ controller
@RequestParam("**hoidanitFile**") MultipartFile file

Bước 5: Lưu file
Input đầu vào: MultipartFile file

Update giới hạn upload file:
https://docs.spring.io/spring-boot/docs/current/reference/html/application-properties.html#appendix.application-properties.web

//application.properties
#default = 1MB
spring.servlet.multipart.max-file-size=50MB

#default = 10 MB (form data)
spring.servlet.multipart.max-request-size=50MB

```
//  private final ServletContext servletContext;
        byte[] bytes = file.getBytes();
        String rootPath = this.servletContext.getRealPath("/resources/images");

        File dir = new File(rootPath + File.separator + "avatar");
        if (!dir.exists())
           dir.mkdirs();

        // Create the file on server
        File serverFile = new File(dir.getAbsolutePath() + File.separator +
              +System.currentTimeMillis() + "-" + file.getOriginalFilename());

        BufferedOutputStream stream = new BufferedOutputStream(
              new FileOutputStream(serverFile));
        stream.write(bytes);
        stream.close();
```

Bonus khi upload nhiều file:

```
 <input
        class="form-control"
        type="file"
        id="avatarFile"
        name="hoidanitFile"
        multiple //thêm cái này
        accept=".png, .jpg, .jpeg"
/>

@RequestParam("hoidanitFile") MultipartFile[ ] files
```

#91. Hoàn thiện tính năng Upload file (Part 2)

Source code video này: (#91)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

#92. Các hình thức lưu trữ Data (Encoding, Hashing, Encryption)

Phân biệt Tương tự: Encode, Hash, Encrypt

Không nên dịch nghĩa ra, thay vào đấy là hiểu tác dụng nó dùng để làm gì.
Nếu dịch, cả 3 keyword trên = mã hóa. Điểm khác biệt là cách thức hoạt động của nó

"Mã hóa" là cách chúng ta che dấu đi dữ liệu (data) ban đầu, để hạn chế tối đa việc bị lộ thông tin. But How ?

1. Encode
Encode ><Decode (mã hóa ><giải mã)

Encoding là cách chúng ta "biến đổi" data từ dạng này sang dạng khác.
ví dụ: data1 -> data2

input: "hello world"
output: 10100001101000001111

+ Không sử dụng key/password để mã hóa

+ Có thể giải mã chuỗi đã được encoding

+ Thông thường, encoding không dùng để "bảo mật" thông tin (data)

+ Được dùng để: compression (nén dữ liệu), Streaming
Các định dạng thực tế sử dụng encode data: Base64, mp3 (audio), mp4 (video)

2. Hashing
ví dụ:
input: "hello world"
output: 3338be694f50c5f338814986cdf0686453a888b84f424d792af4b9202398f392

Hashing là cách chúng ta "băm" data thành 1 chuỗi string đặc biệt (hash)
+ Đây là quá trình 1 chiều, bạn không thể "giải mã" 1 chuỗi đã được hash để lấy dữ liệu ban đầu

bạn có chuỗi 'output', bạn không thể băm ngược lại để có chuỗi input ban đầu

use case:
- lưu password hoặc data bạn muốn

cách làm:
Bước 1: hash input A => Store hash A
Bước 2: cung cấp input B. Nếu input B có hash B = hash A thì match

bạn tạo tài khoản (register) -> lưu data vào database
bạn login -> hash password input và so sánh với hash ở database
nếu matching => ok

3. Encryption

Encrypt là cách chúng ta mã hóa dữ liệu sử dụng key/password. Nó là 1 dạng khác của "encode", tuy nhiên có mức bảo mật cao hơn.
Ví dụ: sử dụng thuật toán RSA để encrypt/decrypt data (khóa nextjs)

encoding: lưu trữ data hiệu quả (ví dụ nén dữ liệu)

encryption: bảo vệ data

#93. Hash User Password

Source code video này: (#93)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

Tài liệu:
https://docs.spring.io/spring-security/reference/features/authentication/password-storage.html

https://www.baeldung.com/spring-security-registration-password-encoding-bcrypt

private BCryptPasswordEncoder bCryptPasswordEncoder;

//todo: hoàn thiện tính năng create user (lưu avatar dưới dạng String)

Lưu ý: chưa validate data

#94. Hoàn thiện tính năng CRUD User

Update table user (hiển thị role)

Lưu ý: chưa validate data

Chapter 12: Module Product
Thực hành module CRUD sản phẩm

#95. Design Giao Diện Trang Chủ
Source code video này: (#95)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

Template ban đầu:
https://gitlab.com/public-starter-projects1/000-java/01-java-spring-mvc/04-client-template

//todo
Design giao diện homepage (convert từ html sang jsp)

#96. Chia layout Client
Source code video này: (#96)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

#97. Bài tập Design view detail product

Source code video này: (#97)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

Yêu cầu:

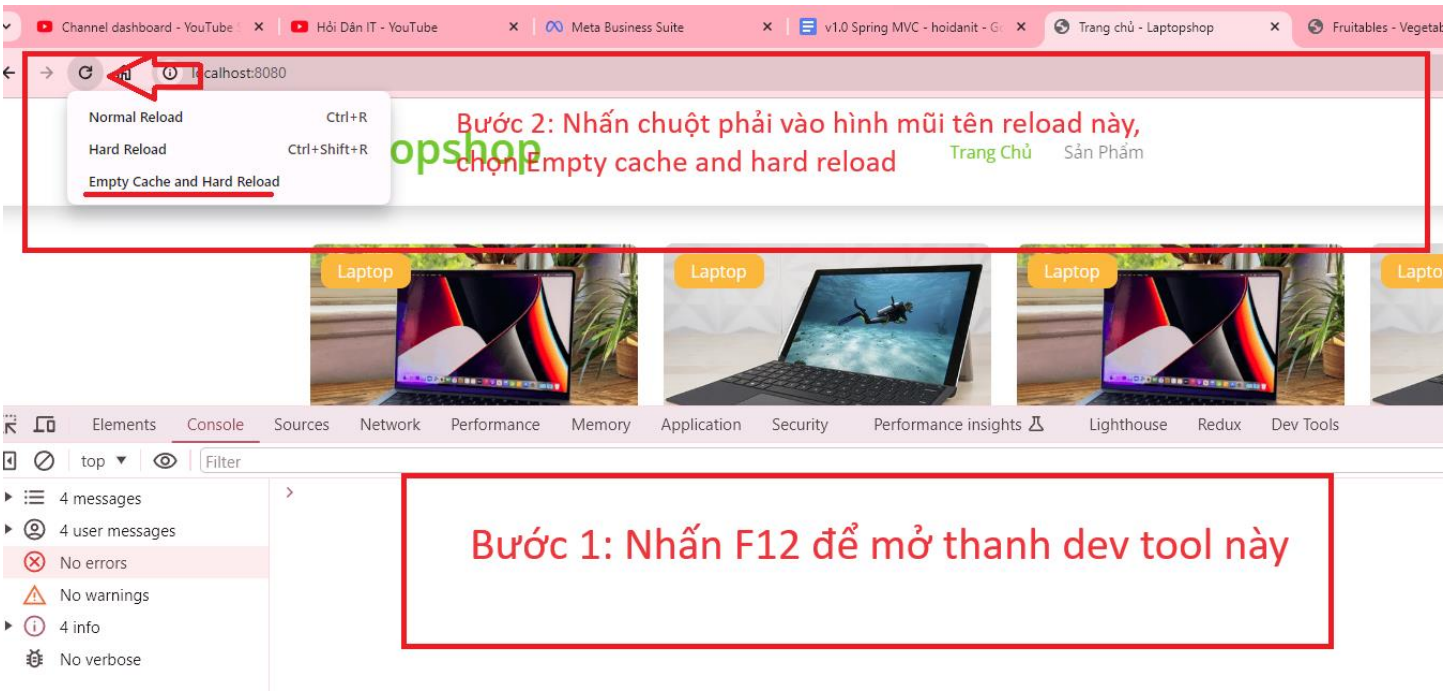
- Tạo url ứng với id
- Tạo view (detail.jsp) ứng với product

Lưu ý: không đặt bean cùng tên

#98. Hoàn thiện layout Client

Source code video này: (#98)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

Nếu sau khi update hình ảnh, tuy nhiên website không cập nhật, cần clear "cache" để load hình ảnh mới



#99. Bài tập Design Giao Diện Thêm mới Product

Source code video này: (#99)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

Yêu cầu:

Tạo form để thêm mới Product, gồm các trường thông tin: (model Product)

- Tên sản phẩm: **name**
- Giá: **price**
- Mô tả chỉ tiết: **detailDesc (dùng textarea cho trường này)**
- Mô tả ngắn: **shortDesc**
- Số lượng: **quantity**
- Nhà sản xuất: **factory**
- Mục đích sử dụng: **target**
- Ảnh sản phẩm: **image**

#100. Validate Form Input

1. Validate dữ liệu là gì ?

Validate là cách chúng ta **kiểm tra tính "hợp lệ" của dữ liệu** trước khi thao tác

Ví dụ: muốn có tài khoản born hub, bạn cần xác nhận >= 18 tuổi

Tác dụng: hạn chế rác dữ liệu (dữ liệu không hợp lệ được lưu vào database)

2. Validate dữ liệu tại đâu ? Frontend hay Backend ?

Nếu validate dữ liệu tại frontend, bạn sẽ cần sử dụng html và javascript

Ví dụ:

<input type="text" id="username" name="username" **required**>

https://getbootstrap.com/docs/5.0/forms/validation/#custom-styles

Nếu validate dữ liệu tại backend, bạn sử dụng chính ngôn ngữ code lên backend. Ví dụ như khóa học này là sử dụng Java

Trong thực tế, chúng ta cần validate dữ liệu tại frontend hay backend là đủ ?

Với những trường hợp đơn giản, dữ liệu không quan trọng (cho phép sai số), chỉ cần validate tại frontend.

Tuy nhiên, để cho chắc chắn 100%, bắt buộc validate tại backend (không thể hack được)

3. Validate form input với Spring

Có 2 cách hay dùng để validate dữ liệu:

Cách 1: Sử dụng Jakarta.validation (Jakarta Bean Validation)
https://beanvalidation.org/

Ví dụ:
https://spring.io/guides/gs/validating-form-input

Cách 2: Sử dụng Hibernate Validator
https://docs.jboss.org/hibernate/stable/validator/reference/en-US/html_single/#preface

Ví dụ: https://www.geeksforgeeks.org/hibernate-validator-with-example/

Cần cài đặt thêm trong file pom.xml

```
<dependency>
   <groupId>org.hibernate.validator</groupId>
   <artifactId>hibernate-validator</artifactId>
   <version>8.0.1.Final</version>
</dependency>
```

Nên sử dụng cái nào ? (tương tự như câu hỏi lựa chọn công nghệ nào là phù hợp)

Lựa chọn cái nào cũng được, miễn sao nó giải quyết tốt vấn đề của bạn.
Khi nào nó không giải quyết được nữa, switch qua cái còn lại (hoặc tìm công nghệ khác)

#101. Validate Model
Tài liệu: https://spring.io/guides/gs/validating-form-input

Bước 1: Decorate Model với annotation
https://jakarta.ee/specifications/bean-validation/3.0/jakarta-bean-validation-spec-3.0.html#builtinconstraints

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-validation</artifactId>
</dependency>
```

Bước 2: Validate tại controller
Ádfaf
@Valid

Bước 3: Get message lỗi

@Binding Result
https://stackoverflow.com/a/7384449

```
  List<FieldError> errors = bindingResult.getFieldErrors();
   for (FieldError error : errors ) {
      System.out.println (error.getObjectName() + " - " + error.getDefaultMessage());
   }
```

#102. Hiển thị thông báo lỗi

Source code video này: (#102)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link
Validate email:
https://stackoverflow.com/a/65371615

```
@Email(message = "Email is not valid", regexp = "^[a-zA-Z0-9_!#$%&'*+/=?`{|}~^.-]+@[a-zA-Z0-9.-]+$")
@NotEmpty(message = "Email cannot be empty")
private String email;
```

Về error tags:
https://docs.spring.io/spring-framework/reference/web/webmvc-view/mvc-jsp.html#mvc-view-jsp-formtaglib-errorstag

https://docs.spring.io/spring-framework/docs/6.1.5/javadoc-api/org/springframework/web/servlet/tags/form/ErrorsTag.html

Validate với Bootstrap 5:
https://getbootstrap.com/docs/5.0/forms/validation/#server-side
Đối với span, thêm class "**invalid-feedback**"
Đối với input, thêm class "**is-invalid**"

Check theo điều kiện có valid hay không ?
https://mkyong.com/spring-mvc/spring-mvc-form-check-if-a-field-has-an-error/

Bước 1:

```
<%@taglib prefix="spring" uri="http://www.springframework.org/tags" %>
```

Bước 2:

```
<spring:bind path="newUser.password">
        <form:input type="password"
            class="form-control ${status.error ? 'is-invalid' : ''}"
            path="password" />
</spring:bind>
<form:errors path="password" cssClass="invalid-feedback" />
```

Hoặc sử dụng câu điều kiện:

```
<c:set var="nameHasBindError">
    <form:errors path="password" />
</c:set>


<form:input type="password"
    class="form-control ${not empty nameHasBindError? 'is-invalid':''}"
    path="password" />

<form:errors path="password" cssClass="invalid-feedback" />
```

//todo : làm tương tự cho tính năng create/update user có validate

#103. Bài tập Thêm mới Product

Source code video này: (#103)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

Yêu cầu:

- Có validate dữ liệu và hiển thị thông báo lỗi
- Thêm mới product thành công
- Hiển thị table danh sách product

Lưu ý: upload file sẽ lưu vào thư mục “product”

#104. Bài Tập Update/Delete Product

Source code video này: (#104)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

//todo

#105. Load Động Data Product cho HomePage

Source code video này: (#105)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

1.Lưu dữ liệu text với MySQL

Với field "**detailDesc**", khi khai báo là kiểu String và không định nghĩa **type**
=> database đang sử dụng varchar 255 (lưu tối đa 255 ký tự)

Về kiểu dữ liệu text: https://stackoverflow.com/a/13932834

Bước 1: xóa tables
Xóa order_detail → xóa product

Bước 2: update type

```
@NotNull
@NotEmpty(message = "detailDesc không được để trống")
@Column(columnDefinition = "MEDIUMTEXT")
private String detailDesc;
```

https://www.baeldung.com/jpa-annotation-postgresql-text-type

Bước 3: restart server

2.Tạo fake data

Link download hình ảnh sản phẩm sử dụng trong video (và file sql)
https://drive.google.com/drive/folders/1WOwvjI_hi3fmLbgPI492k8hMTm90d3by?usp=sharing

//todo

#106. Xem Chi Tiết Product

Source code video này: (#106)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

Format currency:
https://stackoverflow.com/questions/51198011/how-to-prevent-e-when-displaying-double-number-in-jsp

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

Chapter 13: Module Auth (Session và Spring Security)
Xử lý authentication/authorization với login/register và Spring Security

#107. Bài Tập Design giao diện Register
Source code video này: (#107)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

#108. DTO - Data Transfer Object
Tài liệu:
https://www.baeldung.com/java-dto-pattern

1.DTO Design Pattern

DTO (data transfer object) là cách chúng ta biến đổi object này sang object khác.

Ví dụ: từ cát (sand) -> chúng ta biến đổi thành "thủy tinh" (transfer ở đây có nghĩa "biến đổi hình thù/định dạng")

Transfer data (giống việc copy data từ usb vào máy tính)

Đổi với lập trình, chúng ta cần DTO để biến đổi class này thành class khác.

Mục đích : tăng thêm/hoặc giảm bớt các thuộc tính của class

Ứng dụng:

Ví dụ, tại server, chúng ta thao tác với User object -> có thông tin password

Tuy nhiên, trước khi ném data về client, chúng ta "nên dấu đi/chém bớt đi" trường password
-> đây chính là lúc DTO ra tay.

Tác dụng:

- Giúp code gọn hơn (con giao 2 lưỡi, giống luật việt vị của bóng đá :v)

//todo: tạo userDTO

Source code video này: (#108)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

#109. Mapper Class
Tham khảo:
https://mapstruct.org/

Source code video này: (#109)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

//todo

#110. Giới Thiệu Custom Validator

Tham khảo:

https://www.bezkoder.com/spring-boot-custom-validation/

https://www.baeldung.com/spring-mvc-custom-validator

https://docs.spring.io/spring-framework/reference/core/validation/beanvalidation.html

Source code phần validator:

https://drive.google.com/file/d/1l9Sh4Tb2xN24oSQJzvsx3ipI7jyRTbYQ/view?usp=drive_link

#111. Hoàn thiện chức năng Register

Source code video này: (#111)

https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

#112. Bài Tập Design giao diện Login

Source code video này: (#112)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

#113. Tổng Quan Về Spring Security

//fix bug register: email không đúng định dạng/ko nhập email

//demo (checkout code hết video #45)
git checkout hash_commit

Ví dụ: git checkout f8edb3708cc09e0634297042028e2af5a14e3d7f
Sau khi checkout, cần enable Spring security

Lưu ý: đây là Spring Security ứng với version 6.x

1. Luồng hoạt động của spring security

Default login với user lưu tại memory. Memory tức là lưu tại RAM của máy tính. Mỗi lần restart server (hoặc máy tính), sẽ cần login lại
https://docs.spring.io/spring-security/reference/servlet/authentication/passwords/in-memory.html

Luồng login gồm các bước sau:

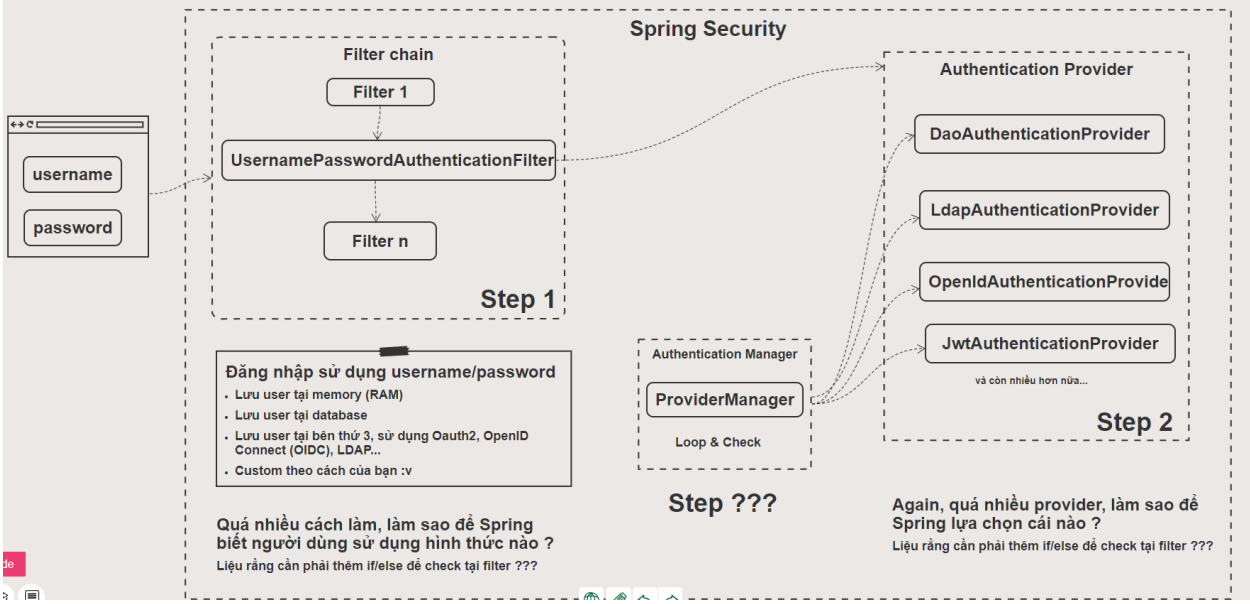
Bước 1: Truy cập page Login, và nhấn nút submit.
Các thông tin gửi lên server bao gồm các fields sau:
username : đây là tên đăng nhập (có thể là email, số điện thoại, cccd…).
Tuy nhiên path = username

password : mật khẩu của user (plain password)

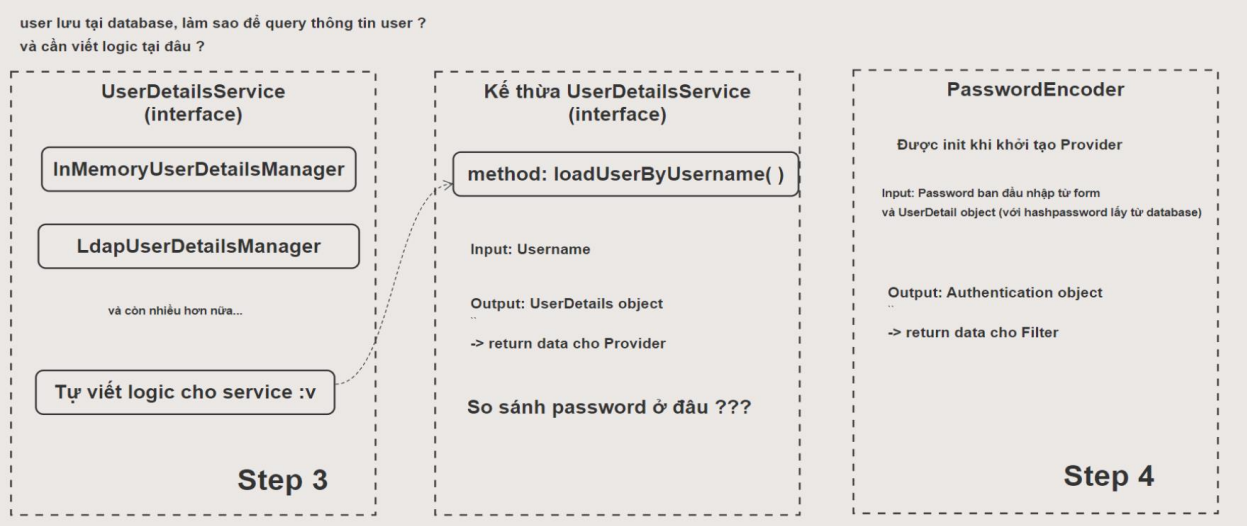
=> user cung cấp credentials (thông tin xác thực) cho server

Bước 2: Spring Security chạy Filter để lọc request

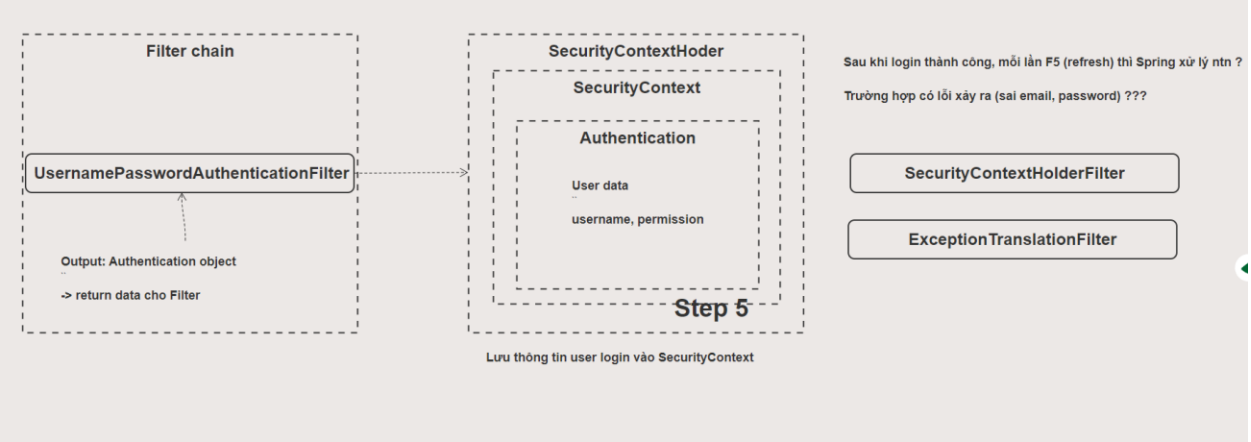
- Chọn provider, mặc định khi sử dụng username/password, Spring sử dụng **DaoAuthticationProvider**



Sử dụng UserDetailsService để lấy thông tin user:
https://docs.spring.io/spring-security/reference/servlet/authentication/passwords/user-details-service.html



Bước 3: Nạp user vào Context



#114. Debug Spring Security (Part 1)

Lưu ý: bạn nào muốn debug như trong video thì cần back code lại #45 (và enabled Spring Security). Còn với code hiện tại chúng ta không debug được, vì đang lỗi tại phần compare password, dẫn tới không login được.

Nếu vẫn muốn debug, thì đợi sau khi làm xong luồng login, bạn debug project này bình thường

https://github.com/redhat-developer/vscode-java/issues/204

Mở nhanh file (đã compiled) với VSCode:
nhấn **Ctrl + T**

Thứ tự các file cần debug:
UsernamePasswordAuthenticationFilter

AuthenticationManager

ProviderManager

DaoAuthenticationProvider

InMemoryUserDetailsManager

SecurityContextHolderFilter

Watch variable:
deferredContext.get( )

#115. Spring Security loadUserByUsername
Source code video này: (#115)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

Bước 1: fake data
//application.properties
Remove this #config spring security
spring.security.user.name=hoidanit
spring.security.user.password=123456

Tạo account: (từ trang register)
Email: hoidanit@gmail.com
Password: 123456

Bước 2: Enable Spring security
//update file main
@SpringBootApplication
public class LaptopshopApplication { //...}

Bước 3: Viết Service
Input: username

Output: UserDetail object

import org.springframework.security.core.userdetails.**UserDetails**;
import org.springframework.security.core.userdetails.**UserDetailsService**;
import org.springframework.security.core.userdetails.User

//import package with same name
https://stackoverflow.com/a/2079832

```
public class CustomUserDetailsService implements UserDetailsService {
    @Override
   public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
      // TODO Auto-generated method stub
      vn.hoidanit.laptopshop.domain.User user =
this.userService.getUserByUsername(username);
      if (user == null) {
         throw new UsernameNotFoundException("User not found");
      }

      return new User(
            user.getEmail(),
            user.getPassword(),
            Collections.singletonList(new SimpleGrantedAuthority("ROLE_USER")));
   }
```

Bước 4: Overwrite Spring config với Bean

Từ version 5.8, Spring Security không dùng WebSecurityConfigurerAdapter

https://spring.io/blog/2022/02/21/spring-security-without-the-websecurityconfigureradapter

```
   @Bean
   public UserDetailsService userDetailsService(UserService userService) {
      return new CustomUserDetailsService(userService);
   }

   @Bean
   public AuthenticationManager authenticationManager(HttpSecurity http,
PasswordEncoder passwordEncoder,
         UserDetailsService userDetailsService) throws Exception {
      AuthenticationManagerBuilder authenticationManagerBuilder = http
            .getSharedObject(AuthenticationManagerBuilder.class);
      authenticationManagerBuilder
            .userDetailsService(userDetailsService)
            .passwordEncoder(passwordEncoder);
      return authenticationManagerBuilder.build();
   }
```

#116. Debug Spring Security (Part 2)
//bonus: debug xem code chạy (compare password)
DaoAuthenticationProvider
protected void additionalAuthenticationChecks

AbstractUserDetailsAuthenticationProvider

SecurityContextHolderFilter

Watch variable:
deferredContext.get( )

https://stackoverflow.com/questions/43007763/spring-security-encoded-password-gives-me-bad-credentials

https://stackoverflow.com/a/62279149

//application.properties
logging.level.org.springframework.security=DEBUG

```
@Bean
public DaoAuthenticationProvider authProvider(
        PasswordEncoder passwordEncoder,
        UserDetailsService userDetailsService) {

    DaoAuthenticationProvider authProvider = new DaoAuthenticationProvider();
    authProvider.setUserDetailsService(userDetailsService);
    authProvider.setPasswordEncoder(passwordEncoder);
    authProvider.setHideUserNotFoundExceptions(false);

    return authProvider;
}
```

#117. Spring Security Custom Login Page

1.CSRF Attack

Csrf token: token ứng với lượt truy cập web của user. Cần token này để phòng tránh CSRF (Cross Site Request Forgery) . Hiểu 1 cách đơn giản, là tăng độ an toàn cho website của bạn
https://www.youtube.com/watch?v=m0EHlfTgGUU

Csrf token:
https://docs.spring.io/spring-security/reference/servlet/exploits/csrf.html#csrf-integration-form

<input type="hidden" name="${_csrf.parameterName}" value="${_csrf.token}"/>

2. Cấu hình mặc định của Spring
SecurityFilterChainConfiguration

Tài liệu: https://docs.spring.io/spring-security/reference/servlet/configuration/java.html#jc-httpsecurity

https://docs.spring.io/spring-security/reference/servlet/authorization/authorize-http-requests.html

https://spring.io/blog/2019/11/21/spring-security-lambda-dsl

Bước 1: update login.jsp
Lưu ý: với custom login page, cần có form action với các thuộc tính thỏa mãn
https://docs.spring.io/spring-security/reference/servlet/authentication/passwords/form.html
Username, password và csrf token

Bước 2: update security config

```
@Bean
SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
    http
        .authorizeHttpRequests(authorize -> authorize
          .anyRequest().permitAll())

        .formLogin(formLogin -> formLogin
            .loginPage("/login")
            .failureUrl("/login?error")
            .permitAll());

    return http.build();
}
```

//Lưu ý: từ v6.0, Spring auto "validate" forward request, có nghĩa là với spring MVC, các request tới view (jsp) sẽ bị check.

https://stackoverflow.com/a/75223368
https://docs.spring.io/spring-security/reference/5.8/migration/servlet/authorization.html#_permit_forward_when_using_spring_mvc

Not working (infinity loop):

```
        .authorizeHttpRequests(authorize -> authorize
            .requestMatchers("/login", "/client/**", "/css/**", "/js/**",
"/images/**").permitAll()
            .anyRequest().authenticated())
```

Working:

```
    http
        .authorizeHttpRequests(authorize -> authorize
            .dispatcherTypeMatchers(DispatcherType.FORWARD,
                DispatcherType.INCLUDE) .permitAll()

            .requestMatchers("/login", "/client/**", "/css/**", "/js/**",
"/images/**").permitAll()
            .anyRequest().authenticated())
```

Bước 3: Thông báo lỗi nếu login failed

```
<c:if test="${param.error != null}">
    <div class="my-2" style="color: red;">Invalid email or password.</div>
</c:if>
```

Source code video này: (#117)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

#118. Authorize với Spring
https://docs.spring.io/spring-security/reference/servlet/authorization/index.html

1.Setup login with role
Mặc định, spring sẽ thêm tiền tố (Prefix) https://docs.spring.io/spring-security/reference/servlet/authorization/architecture.html#authz-authorities

Tạo 2 account:
user@gmail.com với role = USER
admin@gmail.com vơi role = ADMIN
=> spring sẽ lưu trong context là ROLE_USER VÀ ROLE_ADMIN

//CustomUserDetailsService.java
=> update code

2. Các hình thức kiểm tra "quyền hạn"
Ở mức cơ bản, có 2 hình thức chính:

Cách 1: Method Security
https://docs.spring.io/spring-security/reference/servlet/authorization/method-security.html#using-authorization-expression-fields-and-methods

Cách 2: HTTP Request
https://docs.spring.io/spring-security/reference/servlet/authorization/authorize-http-requests.html#authorizing-endpoints

#119. Authorize by Role

Source code video này: (#119)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

Phân tích Yêu cầu:

Yêu cầu 1: Với giao diện client

- Không phân biệt role, ai cũng có thể truy cập trang homepage, sản phẩm và xem chi tiết sản phẩm
  URL = / /product

- Người dùng cần đăng nhập (role = USER/ADMIN) để truy cập trang giỏ hàng và đơn hàng
  URL = /cart /order

Yêu cầu 2: với giao diện admin

- (/admin) => chỉ role = ADMIN

Yêu cầu 3: xử lý khi login success (**optional >< previous url**)

- Khi login thành công, nếu role = USER -> redirect về "/"
  Nếu role = ADMIN -> redirect về /admin

Tham khảo: https://www.baeldung.com/spring-redirect-after-login

https://www.baeldung.com/spring-security-redirect-login

https://github.com/spring-projects/spring-security/blob/main/web/src/main/java/org/springframework/security/web/authentication/SimpleUrlAuthenticationSuccessHandler.java#L60

#120. Chức năng Logout

Bước 1: Update Header

- Thêm thông tin user login và chức năng logout

Yêu cầu: khi chưa login, không hiện icon giỏ hàng và user. Xóa modal search (and icons search)

Sử dụng component dropdown:
https://getbootstrap.com/docs/5.0/components/dropdowns/

```
 <div class="dropdown my-auto">
                        <a href="#" class="dropdown" role="button" id="dropdownMenuLink"
                           data-bs-toggle="dropdown" aria-expanded="false" data-bs-toggle="dropdown"
                           aria-expanded="false">
                           <i class="fas fa-user fa-2x"></i>
                        </a>

                        <ul class="dropdown-menu dropdown-menu-end p-4" aria-
labelledby="dropdownMenuLink">
                           <li class="d-flex align-items-center flex-column" style="min-width: 300px;">
                              <img style="width: 150px; height: 150px; border-radius: 50%; overflow: hidden;"
                                 src="/images/product/1711078092373-asus-01.png" />
                              <div class="text-center my-3">
                                 Hỏi Dân IT bla bla bla
                              </div>
                           </li>

                           <li><a class="dropdown-item" href="#">Quản lý tài khoản</a></li>

                           <li><a class="dropdown-item" href="#">Lịch sử mua hàng</a></li>
                           <li>
                              <hr class="dropdown-divider">
                           </li>
                           <li><a class="dropdown-item" href="#">Đăng xuất</a></li>
                        </ul>
</div>
```

Bước 2: Display username login
https://stackoverflow.com/a/21336269

Bước 3: Logout

POST /logout
https://docs.spring.io/spring-security/reference/servlet/authentication/logout.html

Bước 4: Custom page for unauthorize
https://www.baeldung.com/spring-security-custom-access-denied-page

Source code video này: (#120)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

Lưu ý: source code cung cấp gồm 2 file header.jsp (1 file cho client và 1 file cho admin).
Cần 2 file, vì đều làm chức năng logout tại 2 nơi này

#121. Cơ chế Session và Remember Me
//bonus: giải thích phần redirect tại #119

1. Remember me ?
//close browser?
https://docs.spring.io/spring-security/reference/servlet/authentication/rememberme.html

https://www.baeldung.com/spring-security-remember-me

2.Session là gì ?
https://www.baeldung.com/cs/web-sessions

Hiểu 1 cách đơn giản, Session là việc server:

- Lưu thông tin user vào Memory (RAM), với id để xác định user đấy là ai
- Mỗi request client gửi lên server, sẽ cần đính kèm "key"/id
- Server sẽ dựa vào key client gửi lên, để biết client đấy là ai

3. Cơ chế Session mặc định của Spring
- Được lưu tại client thông qua cookies
- Được lưu tại Memory => **restart lại sẽ mất thông tin**

Giải pháp hay dùng:
https://spring.io/projects/spring-session

4. Cơ chế của Spring Session
https://docs.spring.io/spring-session/reference/guides/boot-jdbc.html#httpsession-jdbc-boot-servlet-configuration

Spring-Session sẽ cung cấp **springSessionRepositoryFilter** (đây là Filter)

Filter trên sẽ overwrite (ghi đè) thông tin HttpSession .

Hiểu 1 cách đơn giản, thông tin về Session sẽ do Spring-Session phụ trách. Còn việc nó hoạt động như thế nào, làm sao để chạy với Spring thì nó đã tự lo :v

#122. Spring Session

1.Tích hợp Spring-Session

Bước 1: add dependencies

```
 <dependency>
   <groupId>org.springframework.session</groupId>
   <artifactId>spring-session-jdbc</artifactId>
 </dependency>
```

Bước 2: cấu hình với spring

//application.properties

https://docs.spring.io/spring-boot/docs/current/reference/html/application-properties.html#appendix.application-properties.security

```
spring.session.store-type=jdbc
spring.session.timeout=30m

#spring.session.jdbc.initialize-schema=always
#server.servlet.session.timeout
#spring.session.jdbc.table-name=SPRING_SESSION
```

Bước 3: setup với Spring security

https://docs.spring.io/spring-session/reference/3.0/spring-security.html#spring-security-concurrent-sessions

```
.sessionManagement((sessionManagement) -> sessionManagement
                        .sessionCreationPolicy(SessionCreationPolicy.ALWAYS)
        .invalidSessionUrl("/logout?expired")
                                .maximumSessions(1)
        .maxSessionsPreventsLogin(false))

        //.logout(logout-
>logout.deleteCookies("JSESSIONID").invalidateHttpSession(true))
```

2. Cơ chế hoạt động của Session

Bước 1: Mỗi lần vào website, Server sẽ tạo 1 cookie lưu tại client

Bước 2: Nếu server restart, client vẫn lưu cookies => không bị mất dữ liệu

Nếu client close browser (đóng session). Cookies sẽ mất => bị mất dữ liệu

=> tính bảo mật cao (ví dụ banking app)

3. Tích hợp "Remember-me"

```
@Bean
SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
	http
		// ... additional configuration ...
		.rememberMe((rememberMe) -> rememberMe
			.rememberMeServices(rememberMeServices())
		);
}

@Bean
public SpringSessionRememberMeServices rememberMeServices() {
	SpringSessionRememberMeServices rememberMeServices =
			new SpringSessionRememberMeServices();
	// optionally customize
	rememberMeServices.setAlwaysRemember(true);
	return rememberMeServices;
}
```

Cấu hình cookies:
https://docs.spring.io/spring-session/reference/3.1/api.html#api-cookieserializer

Cơ chế hoạt động:
https://docs.spring.io/spring-session/reference/3.1/spring-security.html#spring-security-rememberme

Source code video này: (#122)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

#123. Test Session

1. Lấy data user từ session or security context ?

//set user vào session ?

Restart server ?

Expired ?

Login 1 session ?

Source code video này: (#123)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

Chapter 14: Module Cart/Order
Hoàn thiện tính năng giỏ hàng và thanh toán sản phẩm

#124. Phân tích chức năng Giỏ Hàng
Source code video này: (#124)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

//model **Cart** (giỏ hàng)
id
user id //user id
cart_detail_id
sum //tổng số lượng sản phẩm trong giỏ hàng

//model **CartDetail**
id
cart_id
product_id
quantity //số lượng của sản phẩm
price //giá cả của sản phẩm

Lưu ý về 1 quan hệ:
User - Cart: 1 -1

Cart - CartDetail (tương tự Order và OrderDetail): 1 - N

#125. Thêm sản phẩm vào giỏ hàng

Source code video này: (#125)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

1.Lưu ý về CRSF token

<input type="hidden" name="${_csrf.parameterName}" value="${_csrf.token}" />

2.Phân tích logic thêm sản phẩm vào giỏ hàng

//model Cart
user id
cart_detail_id

//model CartDetail
cart_id
product_id

Bước 1: Request gửi lên server
Gồm các thông tin:

- Sản phẩm : product_id

Số lượng : mặc định = 1
user_id: lấy từ spring security/hoặc session

Bước 2: Xử lý tại server
Model Cart: check xem đã từng có giỏ hàng chưa. Nếu chưa có, cần tạo mới.
(lưu ý tính sum product)

Từ model Cart này, sẽ lưu thông tin vào CartDetail, ứng với sản phẩm mua.
Mỗi 1 sản phẩm mua, tăng số lượng sum của table Cart lên 1 đơn vị

//Mỗi lần lưu cart thành công, tiến hành update Session (tổng số sản phẩm đã mua để render lại view)

#126. Design giao diện chi tiết giỏ hàng
Source code video này: (#126)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

Bug: xử lý trường hợp thêm cùng 1 sản phẩm

- Count số lượng sản phẩm của 1 giỏ hàng: nếu đã tồn tại giỏ hàng, sum = sum + 1

//sau khi tạo cart_detail => update data "số lượng sản phẩm" vào session

//khi login thành công . user -> getCart. set data "số lượng sản phẩm" vào session

#127. Bài tập Chức năng chi tiết Giỏ hàng
Source code video này: (#127)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

Fix bug nếu chưa từng có giỏ hàng -> hiển thị = 0

Yêu cầu:

- Lấy động danh sách sản phẩm trong giỏ hàng, hiển thị lên view

#128. Xử lý tăng/giảm Product trong Cart

Source code video này: (#128)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

Fix bug: khi giỏ hàng = 0, cần check điều kiện (lý do cần optional :v)
//nếu chưa thêm vào giỏ hàng, hiển thị message tại view (bài tập tự làm)

- Xử lý tăng/giảm giỏ hàng

#129. Bài Tập Xóa Product từ Cart

Source code video này: (#129)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

Gợi ý cách làm:

Bước 1: Tạo form tại view

- Xóa sản phẩm khỏi giỏ hàng: (cần csrf token cho form)
- Truyền thêm id của cart-detail để biết xóa bản ghi nào
  <form method="post" action="/delete-cart-product/**${cartDetail.id}**">

Bước 2: xử lý tại controller/service

Lưu ý: do ràng buộc cha/con, cần xóa Cart-detail, rồi mới xóa cart
Xóa cart-detail. Và update sum/hoặc xóa cart

- Tìm cart-detail theo id (id từ view gửi lên)

- Từ cart-detail này, lấy ra đối tượng cart (giỏ hàng) tương ứng

- Xóa cart-detail

- Kiểm tra điều kiện:
  - Nếu Cart có sum > 1 => update trừ đi 1 đơn vị (update cart)
  - Nếu Cart có sum = 1 => xóa cart

Bước 3: update session để hiển thị số lượng giỏ hàng

- Nếu Cart có sum > 1 => update trừ đi 1 đơn vị -> set session = sum
- Nếu Cart có sum = 1 => xóa cart -> set session = 0

#130. Bài tập Design Giao Diện Thanh Toán (Checkout)

Source code video này: (#130)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

Yêu cầu:

- Trước khi nhấn nút "xác nhận đơn hàng", cần cập nhật cart_detail ứng với số lượng hiển thị trên màn hình

- Design giao diện checkout

- Update model Order:
  receiverName: tên người nhận
  receiverAddress: địa chỉ người nhận
  receiverPhone: số điện thoại người nhận
  status: trạng thái đơn hàng (PENDING)

- Submit form, lấy data từ view gửi lên controller

#131. Chức năng Đặt Hàng (Place Order)

Source code video này: (#131)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

Logic xử lý khi đặt hàng:

- Tạo order
- Tạo order-detail

//case đơn giản (thanh toán tất cả sản phẩm)

- Xóa cart tương ứng Xóa Cart_detail.g
- Cập nhật giỏ hàng (session) = 0

//case phức tạp (chọn sản phẩm để thanh toán)

- Xử lý cart-detail:
  + Nếu số lượng sản phẩm >= sản phẩm trong cart-detail -> xóa
  + Nếu nhỏ hơn -> update

- Xử lý cart:
  + Count cart_detail -> update sum
    Nếu cart_detail = 0 -> xóa cart

#132. Bài tập Quản lý Order tại Admin

Source code video này: (#132)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

Design thankyou page

Fix bug tại table order:

- Lưu trạng thái đơn hàng là PENDING
- Lưu tổng số tiền của đơn hàng

CRUD Order tại Admin:

- Không tạo đơn hàng từ Admin
- Khi nhấn xem chi tiết đơn hàng, query Order_detail

#133. Xây dựng dashboard

Source code video này: (#133)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

#134. Bài tập Chức năng Lịch Sử Mua Hàng

Source code video này: (#134)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

- Người admin có thể thay đổi trạng thái đơn hàng
- Chức năng thêm sản phẩm vào giỏ hàng (từ trang xem chi tiết sản phẩm)
- Chức năng lịch sử mua hàng

#135. Tổng kết các kiến thức đã học (Basic)

End game kiến thức cơ bản:
Các kiến thức đã học:

- **Spring Boot** : cấu hình và chạy dự án Spring một cách nhanh chóng

- **Spring JPA:** viết query database theo **ORM** (object relational mapping)
  Cách viết code theo mô hình : Controller - Service - Repository

  Cách viết code theo mô hình Domain Driven Design : Định nghĩa domain (model)

  Cách validate dữ liệu với package hỗ trợ sẵn của java

  Cách tư duy và thiết kế database, đồng thời ràng buộc mối quan hệ giữa các model (OneToOne, OneToMany, ManytoMany)

  Cách viết code theo chuẩn DI (dependency injection)

- **Spring MVC:** viết code theo mô hình MVC với view là JSP, sử dụng JSTL (Jakarta Standard Tag Library)

- **Spring Security:** authentication (người dùng đã đăng nhập hay chưa ?) và authorization (người dùng có quyền làm gì) cho Spring

- **Spring Session** : duy trì phiên đăng nhập của người dùng

Ngoài ra còn có:

- Chức năng upload file
- Kỹ năng sinh tồn Debug xem code chạy :v

Nhược điểm của cách làm ở trên:

- Chưa tối ưu hóa hiệu năng. Phần này sẽ được làm tiếp tục tại 2 chapter tiếp theo.

- Code frontend khổ vãi shit :v . Mình chia sẻ thật đấy. Nên là sau này, cố gắng học thêm frontend để làm cho nó chuyên nghiệp các bạn nhé :v
  Reload every time :v

Chapter 15: Pagination Query

Tối ưu hóa fetching data với việc phân trang dữ liệu

#136.Tại sao cần phân trang (Pagination) Data ?

Vấn đề tồn đọng: fetch tất cả data (paginate)

Nếu dữ liệu nhỏ (vài chục, vài trăm rows), không vấn đề gì. Tuy nhiên, vài ngàn, vài trăm ngàn, vài triệu … sẽ là vấn đề.

Why ? lấy càng nhiều data, càng tốn nhiều băng thông (tốc độ truyền tải của mạng internet) và thời gian chờ đợi càng lâu, làm giảm trải nghiệm của người dùng.

Giải pháp đề ra:
Chỉ lấy số lượng data vừa đủ. Câu chuyện này tương tự :
Muốn download file nhanh -> tải file nhẹ thôi (file càng nặng, tải càng lâu, thời gian chờ đợi càng lớn)

Muốn chờ đợi ít -> fetch ít data thôi nhé (và chỉ lấy data cần thiết, không dư thừa)

#137. Khái Niệm Offset/Limit
Tài liệu:
https://www.sqltutorial.org/sql-limit/
https://www.w3schools.com/php/php_mysql_select_limit.asp

Limit: giới hạn số phần tử muốn lấy

Offset: bỏ đi bao nhiêu phần tử , rồi mới lấy data

N1, N2, N3, …. N98, N99, N100 (LIMIT = 10)

PAGE 1: N1, N2, … N10 -> OFFSET = 0
PAGE 2: N11, N12, … N20 -> OFFSET = 10
PAGE 3: N21, N22,... N30 -> OFFSET = 20
…
PAGE 9: N81, N82.. N90
PAGE 10: N91, N92… N100 -> OFFSET = 90

#138. Khái niệm Query String
https://en.wikipedia.org/wiki/Query_string

https://example.com/over/there ?**name=ferret**

https://example.com/path/to/page**?name=ferret&color=purple**

NAME = FERRET

COLOR = PURPLE

https://example.com/path/to/search**?page=1&limit=10**

#139. Design Pagination

Tài liệu: https://getbootstrap.com/docs/5.0/components/pagination/

Source code video này: (#139)

https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

#140. Spring Pagination

Tài liệu:

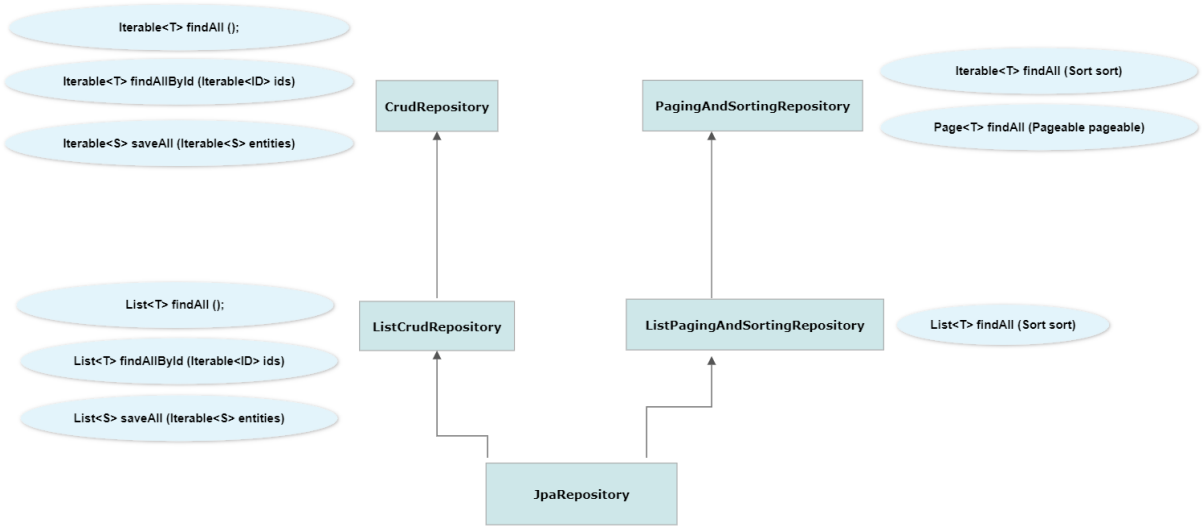
https://docs.spring.io/spring-data/rest/reference/paging-and-sorting.html

https://docs.spring.io/spring-data/rest/docs/2.0.0.M1/reference/html/paging-chapter.html

1. Các loại Repository hay dùng với Spring

- CRUDRepository
- JpaRepository
- PagingAndSortingRepository
  … and more =))

Tham khảo: https://stackoverflow.com/a/74512964



2. Cách xử lý Pagination với Spring

- Kế thừa **JpaRepository**
- Sử dụng **Pageable**

#141. Update Fetch Product với Pagination
Source code video này: (#141)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

Tham khảo: https://stackoverflow.com/questions/56240870/paging-with-spring-mvc-jpa-and-datatables

Bước 1: Tạo Pageable Object
Sử dụng @**RequestParam** để lấy tham số **page** từ client gửi lên

Bước 2: findAll with Pageable

Bước 3: Truyền data qua view
Lấy data với **getContent**

Cần tính **tổng số trang**

Xử lý next, prev

#142. Hoàn thiện Fetch Product

Source code video này: (#142)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

Cần tính **tổng số trang**

Xử lý **next**, **prev.**

Logic xử lý:

- Từ controller, truyền qua view: **currentPage**, **totalPages**

Sử dụng totalPages để dùng **vòng lặp** render ra phần pagination.
https://stackoverflow.com/a/6099110

Nếu index = currentPage -> thêm class **active**

Nếu currentPage = 1 => **disable** button “previous”.
Nếu currentPage = length của vòng lặp => disable button “next”

#143. Bài tập Pagination

Source code video này: (#143)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

Fix bug:

- Không truyền tham số page lên url
  https://stackoverflow.com/questions/22373696/requestparam-in-spring-mvc-handling-optional-parameters

- gán giá trị mặc định cho page = 1 (nếu không truyền page lên url)

//todo
Cập nhật quản lý user, order với pagination

#144. Chức Năng Product

Source code video này: (#144)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

Tài liệu:
https://getbootstrap.com/docs/5.3/forms/checks-radios/

//todo design Product page
//lưu ý design luôn filter

Chapter 16: Filter Query với JPA Specifications
Query data theo tiêu chí chọn trước sử dụng JPA Specifications

#145. Filter dữ liệu với Spring
Tham khảo query của jhipster:
https://gitlab.com/public-starter-projects1/000-java/01-java-spring-mvc/02-java-react-with-jhipster

branch: **filter**
git pull
git checkout filter

https://reflectoring.io/spring-data-specifications/
1.Cách Query truyền thống

//sử dụng **Query Method** : định nghĩa câu query với Spring JPA

//sử dụng **JPA Criteria query (v2.0)**
https://www.mastertheboss.com/hibernate-jpa/criteria-query/hibernate-criteria-example-code/

//sử dụng thư viện (third-party), ví dụ: querydsl
https://github.com/querydsl/querydsl

2. Specification
https://docs.spring.io/spring-data/jpa/reference/jpa/specifications.html

Ý tưởng: tương tự như việc phân trang, repository kế thừa **JpaSpecificationExecutor**

Truyền specification vào hàm findAll:
List<T> findAll(Specification<T> spec);

#146. Giới Thiệu Java Predicate
Tài liệu: https://docs.oracle.com/javase/8/docs/api/java/util/function/Predicate.html

https://www.geeksforgeeks.org/java-8-predicate-with-examples/

Predicate là function trả ra boolean (true/false)

Khi nào cần Predicate:
https://stackoverflow.com/a/38278857

Sử dụng function với 1 tham số đầu vào, và trả ra true/false (hay được dùng với cú pháp lamda)

(arg) - > return true/false

#147. Giới Thiệu JPA Criteria MetaModel
https://vladmihalcea.com/jpa-criteria-metamodel/

Cài đặt thư viện:

```
<!-- https://mvnrepository.com/artifact/org.hibernate.orm/hibernate-jpamodelgen -->
<dependency>
    <groupId>org.hibernate.orm</groupId>
    <artifactId>hibernate-jpamodelgen</artifactId>
    <version>6.4.1.Final</version>
    <scope>provided</scope>
</dependency>
```

Mục đích sử dụng: hạn chế “hardcode tối đa”. Phần còn lại, thư viện lo :v

#148. Create Specification

Source code video này: (#148)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

Tài liệu: https://docs.spring.io/spring-data/jpa/reference/jpa/specifications.html

https://reflectoring.io/spring-data-specifications/

Specification: cách tạo câu điều kiện Where (dynamic)

Root: đại diện table muốn truy vấn, được dùng để truy cập entity và fields của nó

CriteriaQuery: tạo ra cấu trúc tổng quan của query, dùng để modify the select, join, group by, order by, etc. (ít dùng)

CriteriaBuilder: sử dụng predicates, để build ra điều kiện của câu query

Ví dụ chức năng: tìm sản phẩm theo tên (có thể ứng dụng làm tính năng search)

Bước 1: Định nghĩa Specification

```
private Specification<Product> nameLike(String name){
  return (root, query, criteriaBuilder)
     -> criteriaBuilder.like(root.get(Product_.NAME), "%"+name+"%");
}
```

Bước 2: Repository kế thừa **JpaSpecificationExecutor**

Bước 3: Sử dụng findAll( )

//test = cách truyền name=abc lên url , sử dụng tính năng /products tại client

#149. Cách Tạo Predicate với Criteria Builder

Source code video này: (#149)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

https://spring.io/blog/2011/04/26/advanced-spring-data-jpa-specifications-and-querydsl

//todo: tách riêng function search by Name để không lỗi project
//hướng dẫn cách tạo fake data để test

Bước 1: Trang bị kiến thức về SQL
Cần có kiến thức cơ bản về sử dụng điều kiện với SQL
https://www.w3schools.com/sql/

Bước 2: Xác định câu lệnh phù hợp
Các câu lệnh hỗ trợ:
https://docs.oracle.com/javaee%2F7%2Fapi%2F%2F/javax/persistence/criteria/CriteriaBuilder.html

Bước 3: Thực hành test

#150. Bài tập về Specification

Mục đích: Tạo filter cho các tiêu chí
factory : hãng sản xuất
target : mục đích sử dụng
price: giá cả

Cách tư duy:
Query hãng sản xuất: dùng điều kiện **in**

Query mục đích sử dụng: dùng điều kiện **in**

Query giá cả: dùng điện kiện **so sánh : equal, lessThan, greaterThan…**
-> tìm kiếm theo range -> sử dụng **and (sử dụng nhiều Predicate)**

Lưu ý: gộp specification vào 1 class để code cho gọn (đã tạo tại video #149)

Cụ thể:
Tham khảo: https://fptshop.com.vn/may-tinh-xach-tay

Yêu cầu 1: http://localhost:8080/products?**min-price**=1000
Lấy ra tất cả sản phẩm có giá cả tối thiểu là 1000 (vnd)

Yêu cầu 2: http://localhost:8080/products?**max-price**=100000
Lấy ra tất cả sản phẩm có giá cả tối đa là 100000 (vnd)

Yêu cầu 3: http://localhost:8080/products?factory=**APPLE**
Lấy ra tất cả sản phẩm có hãng sản xuất = APPLE

Yêu cầu 4: http://localhost:8080/products?factory=**APPLE,DELL**
Lấy ra tất cả sản phẩm có hãng sản xuất = APPLE hoặc DELL . Truyền nhiều điều kiện, ngăn cách các giá trị bởi dấu phẩy (điều kiện IN)

Yêu cầu 5: http://localhost:8080/products?**price**=10-toi-15-trieu
Lấy ra tất cả sản phẩm theo range (khoảng giá). 10 triệu <= price <= 15 triệu

Yêu cầu 6: http://localhost:8080/products?**price**=10-toi-15-trieu,16-toi-20trieu
Lấy ra tất cả sản phẩm theo range (khoảng giá). 10 triệu <= price <= 15 triệu và
16 triệu <= price <= 20 triệu

#151. Chữa Bài tập về Specification (Part 1)

Source code video này: (#151)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

Các câu lệnh hỗ trợ:
https://docs.oracle.com/javaee%2F7%2Fapi%2F%2F/javax/persistence/criteria/CriteriaBuilder.htm

#152. Chữa Bài tập về Specification (Part 2)

//continue part 1

#153. Xử lý Javascript truyền động URL Filter

Source code video này: (#153)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

Mục tiêu:

- Khi nhấn button filter, cần lấy được các tiêu chí filter, và truyền động lên URL

- Khi reload page, nếu có data trên đường link, cần checkbox filter

Lưu ý:
Ví dụ: http://localhost:8080/products?page=1&price=duoi-10-trieu**%2C**10-15-trieu&sort=gia-tang-dan&factory=APPLE**%2C**ACER&target=GAMING%2CTHIET-KE-DO-HOA%2CMONG-NHE

Trên url có những ký tự "%2C", đây là dấu phẩy ae nhé. Chẳng qua là browser nó encode thành như vậy (đối với ký tự đặc biệt)

Vấn đề trên không ảnh hưởng gì tới hệ thống của chúng ta, vì đa phần các ngôn ngữ server sẽ tự động decode (chuyển cái %2C thành dấu phẩy)

Tham khảo về url encoding: https://en.wikipedia.org/wiki/Percent-encoding

#154. Add Filter Criteria

Source code video này: (#154)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

Viết object để customize request params

#155. Multiple Specification

Source code video này: (#155)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/domain/Specification.html

#156. Add Sort

Source code video này: (#156)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

//todo

Chapter 17: Tổng kết
Tổng kết các kiến thức đã học

#157. Roadmap Spring ?
1. Những kiến thức đã học
Spring mvc là gì ?
Là cách chúng ta viết code theo mô hình Model-View-Controller, sử dụng server side rendering (tức là giao diện được tạo từ phía server)

- Spring boot
- Spring MVC
- Spring JPA
- Spring Security
- Spring Session

2. Chúng ta đang đứng ở đâu ?
Tham khảo: https://roadmap.sh/spring-boot

File pdf road map:
https://drive.google.com/file/d/19jxMycYHJaPJy3M6fqRwBWbxDyoT3sFY/view?usp=sharing

#158. Nhận xét về dự án thực hành

1.Về nghiệp vụ (requirement)

Về nghiệp vụ: mỗi order thành công, cần trừ đi số lượng của product (quản lý tồn kho :v)

Điểm nào chưa hợp lý, cứ sửa theo logic các bạn muốn (lưu ý là sau khi xem hết tất cả video của khóa học hãy làm nhé :v)

2. Về cách code

- Về code frontend: khổ vl . reload every time

#159. Sử Dụng Ajax (bonus)

Source code video này: (#159)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

1. Check giao diện khi chưa đăng nhập

Khi chưa đăng nhập, không thể thêm sản phẩm vào giỏ hàng và bị đá về trang login
=> do cơ chế mặc định của Spring security (chúng ta đang authorize tất cả request)

//fix bug filter sản phẩm dưới 10 triệu (set min # 0)
//fix bug disable next/previous button (set css class)

2. Add to cart mà không reload page

Dùng AJAX để ko cần reload page
Cứ reload đi. Ko sao cả. Học thêm frontend và viết api sẽ giải quyết được vấn đề trên.
Tham khảo: https://github.com/kamranahmedse/jquery-toast-plugin

#160. Phong Cách Code Dự Án Spring

1.Sử dụng DI - Dependency Injection

Tham khảo: https://odrotbohm.de/2013/11/why-field-injection-is-evil/

Có 2 cách hay dùng nhất để sử dụng DI với Spring.

Cách 1: Field Injection với **@Autowired** annotation

Cách 2: Constructor Injection (cách làm trong khóa học)

Field injection (@Autowired)
++ less code to write
-- unsafe code
- more complicated to test

Constructor injection:
++ safe code
- more code to write (see the hint to Lombok)
+ easy to test

2. Sử dụng interface

Tham khảo:
https://stackoverflow.com/questions/53941713/best-practice-to-organize-service-service-implementation-and-repository-in-a-spr

Viết seviceImp

3. Relationship giữa các model

Tối ưu hóa hiệu năng với Fetch mode eager/lazy

Tham khảo: https://www.baeldung.com/hibernate-lazy-eager-loading

#161. Quy Trình Tự Code 1 Website với Spring

Cách code dự án cho chính bạn với Spring Framework (tự thực hành bằng cách clone lại các website thực tế)
Bước 1: Tạo dự án Spring với Spring Boot
Truy cập : https://start.spring.io/

Cần lưu ý về version Java cũng như version của Spring Boot

Bước 2: Thêm dependency
Có thể thêm các thư viện cần thiết ngay tại bước 1, hoặc là khi nào cần dùng thì thêm sau.
Recommend: thêm thư viện ngay từ bước 1 ở trên

Các thư viện khuyến khích cài đặt: (mức cơ bản)
spring-boot-starter-web

spring-boot-devtools

spring-boot-starter-data-jpa

mysql-connector-j

spring-boot-starter-security

spring-boot-starter-validation

Bước 3: Tiến trình code dự án
Tổ chức code theo mô hình MVC và các cấu trúc thư mục như dự án đã học.

Do chúng ta cài đặt Spring-security, nên mặc định website sẽ được bảo vệ bởi Spring (yêu cầu đăng nhập, chặn url…)

Bước 3.1: Disable tạm thời Security (như video #44)
Làm tính năng CRUD users để có tài khoản đăng nhập hệ thống.
Sử dụng Spring-security để hash password

Tạo base các tính năng cơ bản của dự án

Bước 3.2: Cấu hình Security và Session
Làm tính năng Login/register cũng như cấu hình Security để truy cập website

Lưu ý: với method post cần csrf token. Nếu bạn không muốn truyền token tại form, có thể google để disable nó đi

Bước 4: Xử lý các tính năng trong dự án của bạn
Thực hành CRUD

#162. Suy Nghĩ Về Level Fresher

Q: Anh ơi, học xong khóa học này có đủ trình độ để làm fresher hay không ?

Q: Anh ơi, khóa học có cung cấp đủ kiến thức cho fresher hay không ?

A: Khóa học của mình chỉ chịu trách nhiệm chia sẻ kiến thức, đủ hay không, cần đi phỏng vấn, đi thực tập tại công ty mới biết được

Chia sẻ chi tiết:
Với **Java Spring, nó là một hệ sinh thái đa dạng với nhiều kiến thức** của 22 năm phát triển (tính từ 2002 tới 2024)

Hiểu được bản chất của Spring, bạn sẽ tự trả lời được các câu hỏi ở trên.

Cá nhân mình không muốn trả lời Yes/No cho câu hỏi ban đầu, vì bản thân mình KHÔNG LÀ CÔNG TY tuyển dụng bạn

Mỗi 1 công ty, một yêu cầu khác nhau. Chỉ có vào tận nơi mới rõ kết quả.

Điều bạn học được từ khóa học, chính là khả năng tư duy để giải quyết vấn đề

Công nghệ nó luôn thay đổi theo thời gian, chỉ có 1 cái giúp bạn sinh tồn, chính là khả năng tư duy (code như nào, search như nào, debug như nào, đọc tài liệu như nào…)

Level Fresher tại các công ty, luôn là: giao công việc (không chỉ dẫn) và yêu cầu có kết quả. Như vậy, bạn cần có các kỹ năng sinh tồn để có thể vượt qua.

Và hãy nhớ, mình đang cho bạn cần câu cơm, còn câu được gì (cá, cơm…) là phụ thuộc vào bạn (khả năng hiểu và vận dụng), chứ không phải mình bạn nhé.

#163. What's Next

- Chưa sử dụng date
- Sử dụng api
- Tối ưu hóa các kiến thức đã học
- Xem lại roadmap của spring

Một khóa học không thể bao quát tất cả kiến thức. Vì nếu làm vậy, khóa học sẽ rất dài, trong khi học viên gen Z thường lười học @@

Việc tiếp theo nên làm:
Bước 1: Tự thực hành **tối thiểu 01** dự án website với Java Spring MVC (tự làm từ a tới z). Trong quá trình tự làm, không cần nhớ code. Chỗ nào quên thì google

Điều quan trọng là bạn phải hiểu bạn đang làm gì (không yêu cầu bạn hiểu từng chữ một)

Bước 2: học thêm kỹ năng về frontend để code frontend cho nhàn
Recommend các framework/library như React, Angular, Vue

Cách chúng ta dùng Jquery trong khóa học là cách làm đầu những năm 2000 - 2015
Còn từ 2015 trở đi, chúng ta sử dụng Framework/library chuyên dành cho Frontend (đa phần các công ty làm cách này)

Bước 3: Tìm hiểu về API và sử dụng REST API với Spring
Có thể tham khảo Jhipster

Bước 4: Keep learning
Tìm hiểu các mô hình như Spring Cloud, Spring Microservices…

#164. Build Spring với Docker (bonus)

Yêu cầu để thực hiện video này: bạn cần có kiến thức cơ bản về docker
Nếu chưa biết gì, tham khảo:
https://www.youtube.com/playlist?list=PLncHg6Kn2JT4kLKJ_7uy0x4AdNrCHbe0n

Tại sao không deploy local ?

Source code video này: (#164)
https://drive.google.com/drive/folders/1Yq7--94RyTV72BziDCXbCW-fWOIiE3Ja?usp=drive_link

Lời Kết

Như vậy là chúng ta đã cùng nhau trải qua hơn 165+ video về sử dụng framework Spring MVC dành cho backend Java

Tất cả các kiến thức mình chia sẻ, đều được lấy từ kinh nghiệm đi làm của mình và… các trang tài liệu về Spring.

Dĩ nhiên rằng, trong quá trình quá trình thực hiện khóa học này, mình sẽ không thể tránh khỏi những sai sót (vì nếu không có sai số thì mình làm member của team Spring rồi :v).

Vì vậy, nếu thấy sai sót, các bạn cứ thoải mái đóng góp qua Fanpage Hỏi Dân IT nhé.
https://www.facebook.com/askITwithERIC

Nếu bạn thấy khóa học này hữu ích, đừng quên Review đánh giá trên Udemy để nhận được ưu đãi (giảm giá) cho các khóa tiếp theo nhé ^^

Hẹn gặp lại các bạn ở các khóa học tiếp theo ….
Hỏi Dân IT (Eric)