TÀI LIỆU KHÓA HỌC
Java Spring RESTful APIs - Xây Dựng Backend với Spring Boot

PROJECT X

Tác giả: Hỏi Dân IT & Eric

Version: 2.0

Cập nhật:

- Project X (~ 90 videos) : dự án full-stack Spring + React (basic)

X - Chapter 1: Bắt buộc xem - Không bỏ qua chương học này
Hướng dẫn sử dụng khóa học hiệu quả, đạt chất lượng cao nhất

#1. Hướng Dẫn Sử Dụng Khóa Học 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..."

Ưu tiên cách học thông qua các trang tài liệu chính thức (documents), search Google và đọc stackoverflow.
(đây là cách làm mình hướng dẫn trong khóa học)

Trường hợp đã thử mọi cách không có kết quả, đây là lúc sử dụng các công cụ AI:

- Hạn chế việc, copy/paste solution (tức là hỏi, lấy đáp án, và không hiểu logic thực hiện) từ các ứng dụng AI, ví dụ như ChatGPT

- Với beginners, cần phải xây nền móng thật vững trước khi sử dụng các ứng dụng AI. Nên nhớ, copy/paste nhưng phải hiểu bạn đang làm gì.

- **Nếu bạn sử dụng AI, nên thông qua việc hỏi - phản biện**, giống như việc bạn nhờ AI để thực hiện search/deep-research.

- Nên tự đặt câu hỏi là làm sao có thể code được như vậy, cần học những kiến thức gì, và quan trọng nhất, đừng quên bước thực hành.

Vì không có thực hành, tất cả chỉ là lý thuyết.

#2+3+4. Tài liệu khóa học
//todo

#5. Demo kết quả đạt được

Mục tiêu của Project X :

- Hiểu rõ về Spring và định hướng các kiến thức trọng tâm cần học
- Viết code Spring theo chuẩn Restful API
- Thực hành dự án fullstack siêu cơ bản từ A tới Z : Backend Java Spring và Frontend React (TypeScript)
- Hướng dẫn viết testcase (unit test và integration test cho Restful API)

//demo giao diện + demo source code

Lưu ý: dự án Project X là nền tảng hỗ trợ những bạn hổng kiến thức về Java Spring, có cơ hội được ôn tập lại kiến thức trọng tâm trước khi chuyển qua thực hành Project Y

#6. Về Quyền Tác Giả

#7. Cách Dùng Udemy - Hỗ Trợ Hỏi Đáp Q&A

Lưu ý: không bỏ qua video này. Xem để biết cách sử dụng Udemy, cũng như cách đặt Q/A khi cần hỗ trợ (support)

1. Sử dụng trên máy tính

Xem hướng dẫn tài liệu chi tiết tại đây

- Cách bắt đầu sử dụng khóa học (bắt buộc xem)
- Cách đặt câu hỏi cho khóa học (bắt buộc xem)
    - Hướng dẫn cách sử dụng Q&A
    - Hướng dẫn cách liên hệ Instructor qua Message
- Cách sử dụng phím tắt
- Take note trực tiếp trên video đang xem

2. Sử dụng trên điện thoại

Udemy có hỗ trợ ứng dụng trên điện thoại Android/IOS

Xem hướng dẫn tài liệu chi tiết tại đây

#8. Thông Tin Tác Giả Hỏi Dân IT

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/

Nếu bạn muốn nói chuyện với mình (giao lưu trao đổi võ công :), có thể xem mình livestream trực tiếp tối thứ 2 & thứ 5 hàng tuần trên Youtube Hỏi Dân IT

X - Chapter 2: Setup Environment
Cài đặt & chuẩn bị môi trường thực hiện dự án

#9. Chuyện Cài Đặt Công Cụ (Bắt Buộc Xem)

1.Mục đích
Chương học này sẽ hướng dẫn chi tiết cách cài đặt các công cụ cần thiết để phục vụ cho khóa học. Vì vậy, bạn vui lòng không bỏ qua video nào, xem lần lượt theo thứ tự

Có 2 sai lầm mà các bạn hay gặp phải, đặc biệt là những bạn "đã biết 1 chút"

Sai lầm 1: Bỏ qua các video cài đặt công cụ vì bạn cho rằng bạn "đã biết rồi"

Hãy nhớ rằng, **cài công cụ là 1 phần, đang còn phải "cấu hình" nó nữa**.

Khóa học được sinh ra, và đã tối ưu. Bạn chỉ dành 5 tới 10 phút để xem video, đổi lại tiết kiệm cho bạn cả giờ đồng hồ ngồi mò mẫm.

Sai lầm 2: Không quan tâm tới version của phần mềm

Khi thực hiện khóa học, bạn vui lòng download và cài đặt version phần mềm giống như video. **Điều này sẽ đảm bảo môi trường thực thi code là giống nhau. (hạn chế tối đa bug có thể xảy ra)**

2. Version Phần Mềm theo thời gian

Bạn đừng sợ phần mềm (công cụ) nó thay đổi version, hay thậm chí là "chê bai" version cũ. Vì vốn dĩ, công nghệ nó là vậy, luôn thay đổi theo thời gian.

Bạn yêu cầu "version mới nhất" cho cái bạn học, mình đã làm điều đó tại thời điểm quay video khóa học, tuy nhiên, sẽ là cố định 1 version.

Lý do: công nghệ sẽ cập nhật theo thời gian. Cho dù bạn muốn hay không, hoặc thậm chí lắp tên lửa vào đít, đuổi cũng không kịp.
Những cái ngày hôm nay, bạn cho là mới nhất, qua ngày mai, nó đã là cũ.

Điều bạn cần làm là: học 1 version, và quan trọng hơn, là bạn học, bạn cần hiểu nó

Sau đấy, nếu cần thiết, bạn học version mới hơn. Điểm khác biệt ở đây, là khi học version mới, bạn không phải là người bắt đầu từ số 0 (do đã có base từ version cũ)

Chỉ không học version cũ khi và chỉ khi: sản phẩm của version cũ không dùng được

Đấy là lý do tại sao các khóa học của mình, khi học xong, mình mới hướng dẫn nâng cấp version.

#10. Cài Đặt & Cấu Hình Java

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 (sử dụng trong video) tại đây

- 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, 23 -> LST: long term support)

- kiểm tra java path: JAVA_HOME
https://www.baeldung.com/java-home-on-windows-mac-os-x-linux

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

2. Trường hợp dùng nhiều version Java

Nếu như bạn **bắt buộc phải dùng nhiều version của Java** trên cùng một máy tính, **có thể (xác suất nhỏ) là xảy ra lỗi** trong quá trình thực hiện dự án.

Ngược lại, nếu không "bắt buộc" phải dùng nhiều version Java, bạn nên dùng một version thôi. Như vậy nó sẽ đảm bảo môi trường thực thi code của bạn và mình là giống hệt nhau, hạn chế tối đa lỗi có thể xảy ra.

#11. Cài Đặt IDE Code Java Spring

IDE (Integrated Development Environment), là phần mềm cung cấp môi trường để phát triển sản phẩm: từ việc coding, debug, compile...

Tiêu chí ưu tiên: hỗ trợ tốt cho việc coding (gợi ý code, phát hiện lỗi, debug), và chi phí là tối thiểu, hỗ trợ tốt cho Windows, Macos

1.Các công cụ phổ biến để code Java (Spring):

Dùng miễn phí:

- Eclipse : cần cài thêm plugin để code Java Spring
- VSCode: cần cài thêm extension để code Java Spring

- IntelliJ IDEA Community : hỗ trợ Java Spring nhưng không đầy đủ, chỉ basic.

- Spring Tool Suite (STS): chuyên dùng cho Java Spring (phát triển từ Eclipse)

Dùng trả phí: IntelliJ IDEA Ultimate

Trong khóa học này, mình sử dụng Spring Tool Suite (và VSCode, nếu có), vì miễn phí, hỗ trợ tốt cho việc code Java Spring

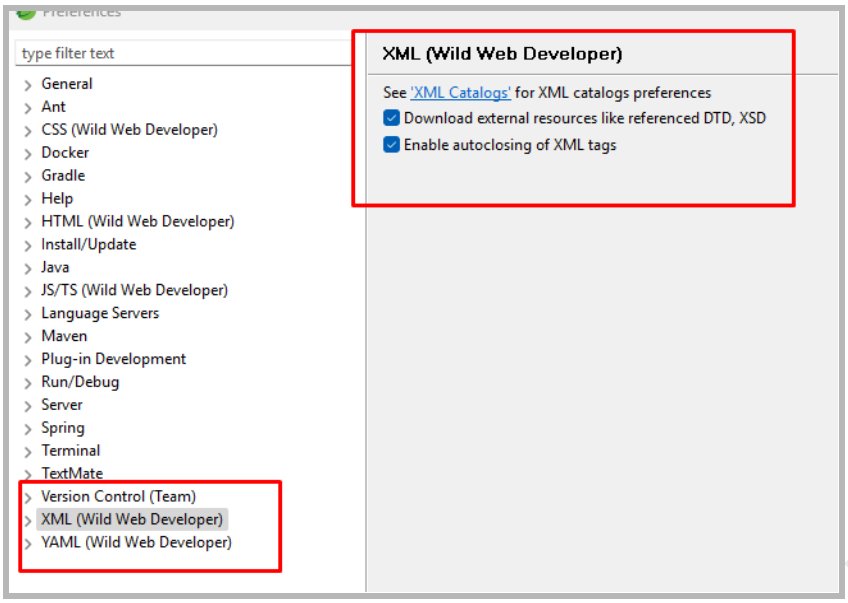
2. Cài đặt Spring Tool Suite (STS)
Tải version mới nhất tại đây

Tải tất cả version tại đây

3. Cấu Hình Spring Tool Suite
//cấu hình download pom.xml

Open the Eclipse preferences window (**Window > Preferences**), Go to **XML**(Wild Web Developer).

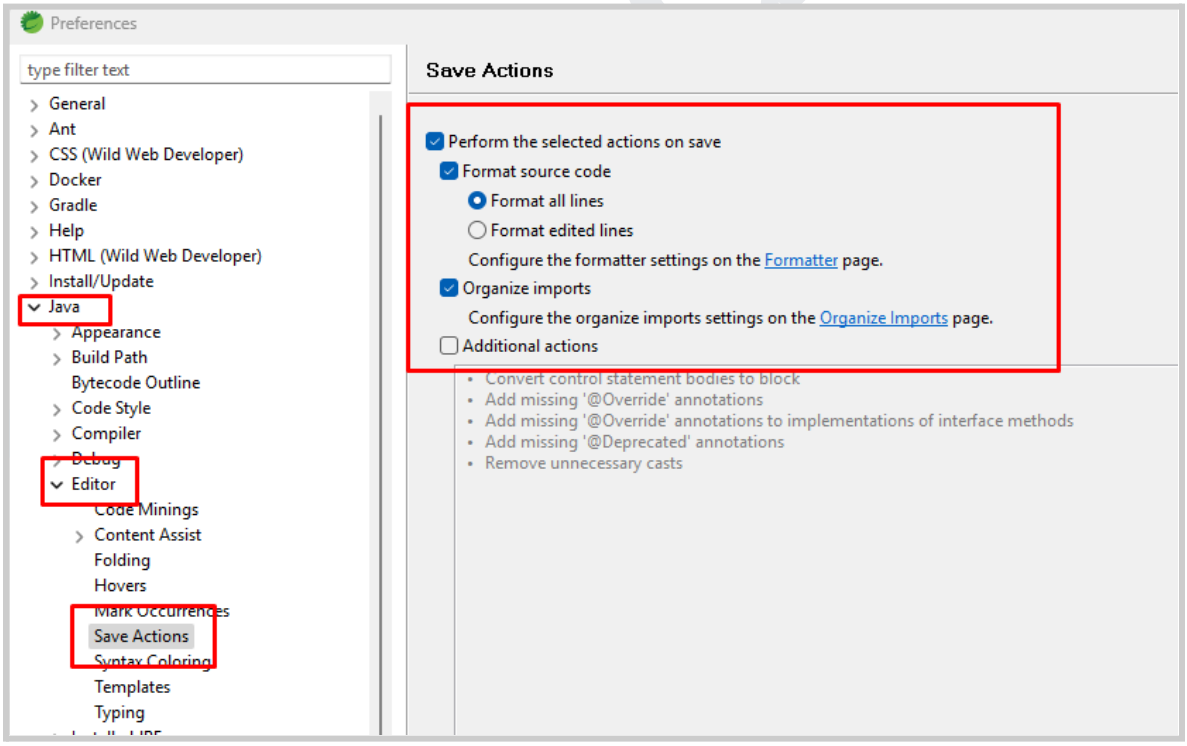
check the "Download external resources like referenced DTD,XSD", then Apply.



//cấu hình format on save

Go to Window > Preferences (or Eclipse > Preferences on macOS).

Expand Java > Editor > Save Actions.



#12. Cài đặt VSCode

Mục đích dùng VSCode code giao diện frontend (nếu có)

1.Cài đặt VSCode

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

2.Cấu hình VSCode

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

Các extensions cài đặt thêm:
- Code Spell Checker : hỗ trợ check chính tả khi đặt tên tiếng anh
- Auto Complete Tag : hỗ trợ code nhanh HTML

#13. Cài đặt Node.JS

Mục đích: môi trường để chạy code Javascript ở phía frontend

Tài liệu: https://nodejs.org/en

1.Nodejs là gì ?
Nodejs không phải là thư viện (library), không phải framework của JavaScript.

Nodejs là môi trường để bạn thực thi code javascript, tại browser và server.

Bạn học Node.js, về bản chất, là học các thư viện/framework (viết bằng JavaScript), nên bạn cần cài đặt môi trường Nodejs để có thể thực thi code JavaScript

Điều này tương tự với:

Bạn học cách sử dụng Microsoft Excel (javascript)
Bạn cần cài hệ điều hành Windows để có thể học nó (nodejs)

2. Cài đặt Nodejs

Sai lầm của beginners, là không quan tới tới version của phần mềm. Nên nhớ, công nghệ nó thay đổi theo thời gian, vì vậy, để hạn chế tối đa lỗi tối đa, bạn nên dùng version phần mềm như khóa học hướng dẫn.

Điều này tương tự với:
Bạn đang chơi 1 con game rất ngon trên Windows 7, bạn vác lên Windows 10 để chạy, có điều gì để đảm bảo rằng "sẽ không có lỗi xảy ra" ?

Trong khóa học này, mình sử dụng **version Node.js là 22.13.0**

Vì vậy, để hạn chế tối đa lỗi có thể xảy ra, bạn vui lòng cài đặt chính xác version nodejs ở trên

Khi code giống nhau, môi trường thực thi code giống nhau (version nodejs), thì rất hiếm khi lỗi xảy ra.

Nếu đây là lần đầu tiên bạn học (coding) một dự án với Node.js, mình khuyến khích sử dụng duy nhất 01 version Node.js (dễ quản lý)

Chỉ sử dụng nhiều version Node.js, khi và chỉ khi, trên bạn có nhiều dự án Node.js, và mỗi dự án yêu cầu một version Node.js khác nhau. (hướng dẫn tại mục 3 bên dưới)

Link tải nodejs v22.13.0:
https://nodejs.org/download/release/v22.13.0/

Sau khi cài đặt xong, kiểm tra bằng cách gõ câu lệnh:

node -v

3. Trường hợp dùng nhiều version Nodejs

Lưu ý: bạn cần gỡ nodejs trước khi cài nvm

//áp dụng cho windows
https://github.com/coreybutler/nvm-windows

//áp dụng cho macos
Video hướng dẫn cài nvm cho mac, xem tại đây

https://dev.to/ajeetraina/how-to-install-and-configure-nvm-on-mac-os-5fgi

#14. Cài đặt Postman
Mục đích: công cụ để test ứng dụng backend

Link download: https://www.postman.com/downloads/

#15. Cài đặt Git
Mục đích: công cụ để quản lý mã nguồn và sử dụng source code của dự án (tránh tình trạng máy tính hư bị mất hết code)

Nếu bạn chưa biết gì về Git, xem nhanh tại đây (miễn phí)
Khóa học Git trả phí, tham khảo tại đây

- 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.

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.

#16. Cài đặt Google Chrome

Mục đích: testing giao diện frontend (nếu có)

Ở đây, sử dụng google chrome vì nó là ứng dụng phổ biến nhất (trình duyệt web được dùng nhiều nhất)

Bạn nên dùng Google Chrome (thay vì Firefox/Edge...) để đảm bảo rằng thao tác sử dụng giữa bạn và mình là giống nhau (tránh gây khó khăn không cần thiết)

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
- Set default app là google chrome (nếu nó mở app, thì chạy với google chrome)

Xem hướng dẫn setup default app cho windows tại đây

X - Chapter 3: Hello World với Spring

Cung cấp một bức tranh tổng quát về môi trường và cách học dự án với Java Spring. Viết chương trình Hello World đầu tiên

#17. Tổng quan về chapter

- Demo kết quả đạt được
- Hệ thống lại kiến thức, tránh tình trạng bị hổng kiến thức

#18. Việc Làm của Spring (Java)

Lưu ý: Chỉ áp dụng với thị trường Việt Nam

Để nắm rõ nhu cầu của thị trường việc làm về công nghệ Java như thế nào, **bạn cần dành thời gian nghiên cứu qua các trang tuyển dụng việc làm** (làm khảo sát - research)

Các tiêu chí khi làm khảo sát, bao gồm:

- **Vị trí (location) bạn sinh sống**. Việc làm tại Hà Nội, Hồ Chí Minh, sẽ khác với Đà Nẵng, Huế...

- **Yêu cầu của công việc**, như bằng cấp, số năm kinh nghiệm làm việc

- **Thời điểm bạn làm khảo sát**: thông thường thời điểm từ tháng 3 tới tháng 9 có số lượng công việc nhiều nhất (quý 2 và quý 3).

  Cuối năm (quý 4) và đầu năm (quý 1) số lượng việc làm sẽ ít hơn (do sự thay đổi nhân sự của công ty là không đáng kể/nhảy việc)

- Nguồn khảo sát, bao gồm mạng xã hội, trang web chính thức của công ty và các trang web tuyển dụng việc làm, có thể kể tới như (ví dụ minh họa):

  https://itviec.com/
  https://topdev.vn/
  https://www.topcv.vn/
  https://www.vietnamworks.com/
  https://www.linkedin.com/

#19. Học Spring Framework hay Spring Boot ?
https://spring.io/

1.Spring vs Spring Boot

Spring Framework: Là một framework lớn, cung cấp rất nhiều module (Spring Core, Spring MVC, Spring Data, Spring Security, v.v.). Tuy nhiên, nó đòi hỏi bạn phải cấu hình các modules thủ công (lắp ghép để cho hoạt động).

Spring Boot: Là một dự án mở rộng từ Spring, giúp đơn giản hóa việc phát triển ứng dụng bằng cách cung cấp cấu hình tự động (auto-configuration) và tích hợp sẵn các thành phần cần thiết, giúp bạn viết ít code hơn.

2. Quy trình từng bước để học Spring cho người mới bắt đầu

1. Học Java Core thật chắc

- Lập trình hướng đối tượng (OOP).
- Generics, Collections, Streams API.
- Exception Handling (try-catch-finally).
- Multi-threading (cơ bản).
- Lambda Expressions & Functional Programming (Java 8+).

2. Học Spring Core (Nền tảng của Spring)

- IoC (Inversion of Control): Spring quản lý đối tượng thay vì bạn tạo thủ công.
- DI (Dependency Injection): Truyền dependencies thay vì khởi tạo trực tiếp.
- Bean & ApplicationContext: Cách Spring quản lý Bean.
- Annotation-based configuration: @Component, @Service, @Repository, @Autowired.
- Properties & Environment: Cấu hình ứng dụng trong application.properties.

3. Học Spring Boot (Dùng để làm dự án thực tế) (khóa học đang đứng tại đây)
   - Tạo project Spring Boot với Spring Initializr.
   - Spring Boot Starter (spring-boot-starter-web, spring-boot-starter-data-jpa…).
   - Spring Boot Auto-configuration.
   - Cấu trúc dự án Spring Boot chuẩn (Controller, Service, Repository).
   - REST API với Spring Boot: @RestController, @RequestMapping, @GetMapping, @PostMapping…
   - Làm việc với JSON (Jackson).

4. Học Spring Data JPA (Làm việc với Database)
   - Spring Boot + JPA (Hibernate) để thao tác database.
   - Entity & Repository: @Entity, @Id, @GeneratedValue.
   - CRUD với database: findById(), save(), delete()...
   - Pagination & Sorting với Spring Data JPA.

5. Học Spring Security (Bảo mật API & Web App)
   - Spring Security Basic Auth (Username & Password).
   - JWT Authentication để bảo vệ API.
   - Role-based Authorization (Admin/User).
   - OAuth2 (Google, Facebook Login).

6. Học Spring Boot nâng cao (Microservices, Cloud, DevOps)
   - Spring Cloud (Eureka, API Gateway, Load Balancing).
   - Spring Kafka & RabbitMQ (Message Queue).
   - Spring Boot với Docker & Kubernetes.
   - Triển khai ứng dụng Spring Boot trên AWS, GCP, Azure.

#20. Tài liệu của Spring

1.Tài liệu chính thức

- Spring Documentation
  Link: https://spring.io/projects/spring-framework

- Spring Boot Documentation
  Link: https://spring.io/projects/spring-boot

- Spring Security Documentation
  Link: https://spring.io/projects/spring-security

2. Websites
https://www.baeldung.com/

Đọc từ Github, stackoverflow

3. Sử dụng công cụ AI

Ưu tiên cách làm này sau cùng, sau khi đã thực hiện cách 1 và 2 chưa có kết quả.

Khi hỏi, cần hỏi theo hướng phản biện, và hỏi các kiến thức nền tảng/cơ bản để giải quyết vấn đề.

Và, cần phải thực hành.

#21. My Goal ???

//Demo với jhipster
https://www.jhipster.tech/

//Tham khảo về dự án jhipster tại đây

#22. Setup Dự Án Thực Hành

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

1. Việc cần làm

Bước 1: Clone/Download project tại đây

Bước 2: Loading project với Spring Tool Suite, chờ xíu để nó tự cài thư viện
//todo

2.Giải thích về cấu trúc dự án thực hành

//todo

Lưu ý về bản quyền source code (license) khi sử dụng ?

#23. Cách mình base dự án thực hành (Extra)

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 theo video này.

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

Mục đích của video này là để giải đáp sự "tò mò" của nhiều bạn về câu hỏi, mình tạo dự án như thế nào.

Bạn xem video 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...)

#24. Hello World với Spring

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 with @hoidanit";
	}
}
```

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

#25. Setup 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)

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

Update file pom.xml:

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

=> cần restart lại server để nhận cấu hình mới

X - Chapter 4: Spring Data JPA (Ôn Tập)

Cấu hình database và ôn tập lại các kiến thức trọng tâm của Spring Data JPA, giúp thao tác với dữ liệu của ứng dụng

#26. Tổng quan về chapter

//todo

#27. Spring Data JPA là gì ?

https://docs.spring.io/spring-data/jpa/reference/jpa/getting-started.html

1. Spring Data JPA là gì

Spring Data JPA là một phần của Spring Framework, cung cấp lớp trừu tượng giúp làm việc với JPA (Java Persistence API) dễ dàng hơn.

- JPA (Java Persistence API) là một tiêu chuẩn giúp ánh xạ dữ liệu từ database vào Java Objects.

- Spring Data JPA giúp đơn giản hóa JPA, giúp bạn không cần phải viết nhiều code khi làm việc với cơ sở dữ liệu.

2. Cài đặt Spring Data JPA

//pom.xml

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

#28. Cài Đặt MySQL Workbench
//todo

#29. Kết nối Spring với Database
1. Cài đặt connector cho mysql
https://mvnrepository.com/artifact/mysql/mysql-connector-java

//update file pom.xml

```
<dependency>
   <groupId>com.mysql</groupId>
   <artifactId>mysql-connector-j</artifactId>
   <version>9.2.0</version>
</dependency>
```

2. Setup properties
//application.properties
Lưu ý về dấu cách

```
spring.datasource.url=jdbc:mysql://localhost:3306/yourDB
spring.datasource.username=yourUser
spring.datasource.password=ThePassword
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.hibernate.ddl-auto=update
#spring.jpa.show-sql: true
```

https://docs.spring.io/spring-boot/appendix/application-properties/index.html

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

#30. Tạo Entity Todo

Tài liệu:
https://spring.io/guides/gs/accessing-data-jpa
Mục đích:
Kết nối xuống database và tạo được table với entity.

Sử dụng
@Entity
@Table
@Id

#31. Kiến trúc phân lớp (Layered Architecture)

1.Cấu trúc thư mục code

```
📂 src/main/java/com/example/app
 ├── 📂 controller    # 🌍 Lớp điều khiển (Controller Layer)
 │    ├── UserController.java
 │
 ├── 📂 service       # 📌 Lớp xử lý nghiệp vụ (Service Layer)
 │    ├── UserService.java
 │
 ├── 📂 repository    # 🛠️ Lớp truy vấn dữ liệu (Repository Layer)
 │    ├── UserRepository.java
 │
 ├── 📂 entity        # 🏗️ Định nghĩa đối tượng (Entity Layer)
 │    ├── User.java
```

2.Demo về cách code
//todo

#32. Repository (Extra)

1.Repository là gì ?

Trong Java Spring (đặc biệt là Spring Data JPA), **repository là một interface** dùng để tương tác với cơ sở dữ liệu (CRUD database) một cách dễ dàng và tự động, mà không cần viết SQL thủ công.

Tại sao repository là interface, không phải là class ?

- Dùng interface, bạn không cần viết code thực hiện bên trong.
  => Spring Data JPA sẽ tự động tạo ra một class thực thi (implementation) khi ứng dụng chạy.

- Bạn tập trung vào định nghĩa hành vi, không cần lo phần implementation.

- Code dễ đọc, dễ test.

2. Interface kế thừa

JpaRepository, CrudRepository, PagingAndSortingRepository

CrudRepository<T, ID> : Dùng khi bạn chỉ cần các thao tác CRUD cơ bản

PagingAndSortingRepository<T, ID> : Dùng khi bạn cần CRUD + phân trang và sắp xếp

JpaRepository<T, ID> : Phổ biến nhất, gồm tất cả từ CrudRepository + PagingAndSortingRepository, và còn thêm các method khác

=> sử dụng repository kế thừa từ : **JpaRepository<T, ID>**

3. Quy luật khi viết code sử dụng repository

- Đặt tên method theo quy ước của Spring Data

- Dùng @Query khi cần viết SQL tùy chỉnh
  ví dụ:
  @Query("SELECT u FROM User u WHERE u.name LIKE %:keyword%")
  List<User> searchByName(@Param("keyword") String keyword);

4. Hướng dẫn cách đặt tên method theo quy ước của Spring Data
Tham khảo:
https://docs.spring.io/spring-data/jpa/reference/jpa/query-methods.html

https://docs.spring.io/spring-data/jpa/reference/repositories/query-keywords-reference.html

Download file cheatsheet tại đây

#33. Tạo Todo (Create)

Lưu ý: viết code theo DI (dependency injection)

Bước 1: Tạo controller
//todo

Bước 2: Tạo service
//todo

Bước 3: Tạo repository
//todo

- Repository là interface, kết thừa **JpaRepository<T, ID>**

#34. Lấy danh sách Todos (Read)
//todo

#35. Cập nhật Todo (Update)
//todo

#36. Xóa Todo (Delete)
//todo

X - Chapter 5: Restful APIs

Tìm hiểu về cách sử dụng Spring để viết Restful APIs

#37. Tổng quan về chapter

//todo

#38. API là gì ?

1.Bài toán đặt ra

Frontend dùng React, Vue, Angular… làm giao diện website

Backend dùng Spring Boot, Node.js, Laravel… xử lý logic và giao tiếp với database

➡️Làm sao để hai phần này giao tiếp với nhau tạo nên 1 website hoàn chỉnh ?

2. API là gì ?

API viết tắt của **Application Programming Interface,** nghĩa là: Giao diện lập trình ứng dụng.

Nói đơn giản, API là cầu nối giúp các phần mềm, hệ thống, hoặc dịch vụ nói chuyện được với nhau.

Ví dụ thực tế: Gọi món ở quán cà phê = API
Bạn đến quán, ngồi vào bàn (bạn là client).

Bạn gọi: “Cho mình 1 ly trà sữa ít đá, thêm trân châu!” (gửi yêu cầu).

Nhân viên ghi order → chuyển cho bếp (nhân viên là API).

Bếp làm xong → giao đồ uống lại cho bạn (phản hồi).

✅ Bạn không cần biết quầy bếp làm thế nào, chỉ cần nhận đúng đồ uống mình muốn.

➡️ API = Nhân viên order, nhận yêu cầu từ bạn và trả lại kết quả từ "hệ thống bên trong".

💻 Ví dụ lập trình: Gọi API để lấy danh sách người dùng

Client (trình duyệt, mobile app) gửi request:
GET https://myapp.com/api/users

Server phản hồi lại:
[
{ "id": 1, "name": "Alice" },
{ "id": 2, "name": "Bob" }
]

😎 **Lập trình viên thì thấy:**
GET /api/users là một API endpoint.

Gọi API = giống như hỏi: "Cho tôi danh sách người dùng"

Server trả kết quả về = dữ liệu bạn cần.

=> Về bản chất, **API là một URL được tạo nên bởi backend**.
Url này (endpoint sẽ được frontend sử dụng để lấy dữ liệu (data)

#39. Restful API là gì ?

RESTful API là một kiểu API được thiết kế theo kiến trúc REST (Representational State Transfer), giúp client (web, mobile...) và server giao tiếp với nhau **thông qua giao thức HTTP** một cách rõ ràng, logic và hiệu quả.

Chúng ta học Java Spring (backend) viết API (restful api) là viết code tại server (java), phục vụ cho client (web, mobile)...

Ví dụ trong thực tế
https://jsonplaceholder.typicode.com/

Các thành phần cấu thành Restful API:

1. Endpoint (URL)

- Là địa chỉ cụ thể mà client gửi yêu cầu đến.
- Mỗi endpoint tương ứng với một chức năng cụ thể.
- **Công việc tạo ra URL (API) này xảy ra tại backend**

Ví dụ:
GET /users → Lấy danh sách người dùng
POST /users → Thêm người dùng mới
GET /users/1 → Xem thông tin người dùng có ID = 1
DELETE /users/1 → Xoá người dùng có ID = 1

2.HTTP Method (Phương thức)

Method	Ý nghĩa	Ví dụ
GET	Lấy dữ liệu	Lấy danh sách người dùng
POST	Thêm mới dữ liệu	Thêm mới sản phẩm
PUT	Cập nhật toàn bộ	Cập nhật thông tin user
PATCH	Cập nhật một phần	Cập nhật một trường nhỏ
DELETE	Xóa dữ liệu	Xóa user

3. Request (Yêu cầu)

Là dữ liệu client gửi đến server, gồm:

- URL: Địa chỉ API
- Method: Phương thức HTTP (**mục 2 bên trên)**
- Headers: (tùy chọn) – chứa thông tin như loại dữ liệu, token xác thực...
- Body: (chỉ với POST, PUT, PATCH) – chứa dữ liệu gửi lên (**thường là JSON**)

Ví dụ JSON trong request body khi tạo mới user:

```
{
  "name": "Hỏi Dân IT",
  "email": "admin@hoidanit.vn"
}
```

4. Response (Phản hồi)

Khi server xử lý xong, nó trả kết quả lại cho client:

- Status code (200, 201, 400, 404, 500…)
  https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status
- Body (dữ liệu JSON, text…)
- Headers

Ví dụ Response JSON khi tạo mới user

```
{
  "id": 1,
  "name": "Hỏi Dân IT",
  "status": "created"
}
```

#40. Phân biệt @Controller và @RestController

@Controller

- Đây là annotation truyền thống của Spring MVC
- Trả về tên của một view template (HTML, JSP...)
- Thường dùng trong website phát triển với mô hình SSR (server side rendering)

Ví dụ:

```
@Controller
public class HomeController {
   @GetMapping("/")
   public String home() {
      return "home"; // render file home.html (thường nằm trong templates)
   }
}
```

@RestController

- Đây là @Controller + @ResponseBody
- Trả về dữ liệu JSON hoặc XML, không cần @ResponseBody nữa
- Dùng để tạo RESTful API

Ví dụ:

```
@RestController
public class ApiController {
   @GetMapping("/api/user")
   public User getUser() {
      return new User("John", 25); // tự động convert sang JSON
   }
}
```

Khi nào dùng cái nào?

👉 Dùng @RestController khi bạn xây dựng API cho frontend (React, Angular, mobile...)

👉 Dùng @Controller khi bạn xây dựng web truyền thống có HTML hiển thị từ backend

@RestController = @Controller + @ResponseBody

#41. GET Method - @GetMapping

GET Method
Khi bạn truy cập vào một trang web, trình duyệt của bạn đang gửi một **HTTP GET** request.
===
@**GetMapping** là một annotation trong Spring dùng để xử lý HTTP GET request.

Nó là cách viết ngắn gọn cho: @RequestMapping(method = RequestMethod.GET)

Dùng khi nào?
- Khi bạn muốn lấy dữ liệu từ server (GET)

- Khi tạo REST API cho frontend hoặc mobile.

- Khi bạn chỉ muốn đọc thông tin, không làm thay đổi dữ liệu. (READ)

Ví dụ về GetMapping
//todo

#42. ResponseEntity

Tài liệu:
https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/http/ResponseEntity.html

ResponseEntity là một đối tượng dùng để tùy chỉnh toàn bộ phản hồi (response) từ controller trả về client trong Spring Boot.

Nó cho phép bạn:

- Thiết lập status code (200, 201, 404, 500…)
- Trả về body (JSON, text, object…)
- Thêm headers nếu cần (Authorization, Set-Cookie…)

=> Phản hồi chuẩn REST, giúp frontend dễ xử lý

#43. Get All Todos API

Hướng dẫn sử dụng postman để test API
//todo

#44. Quy tắc đặt tên URL trong RESTful API

Mục tiêu: Việc đặt tên URL trong RESTful API đúng chuẩn giúp code dễ hiểu, dễ bảo trì và phù hợp với quy tắc REST.

RESTful API là gì ?
Là phong cách thiết kế API mà trong đó:

- Mỗi URL đại diện cho một tài nguyên (resource).

- Sử dụng đúng HTTP method (GET, POST, PUT, DELETE...).

1. Sử dụng danh từ số nhiều (plural nouns)

GET /**users** → Lấy danh sách người dùng
GET /**users**/1 → Lấy thông tin người dùng có ID = 1
POST /**users** → Tạo người dùng mới
PUT /**users**/1 → Cập nhật người dùng ID = 1
DELETE /**users**/1 → Xoá người dùng ID = 1

❌ Không dùng động từ trong URL (cách dùng sai):
❌ /getUserById/1
❌ /deleteUser/1

2. Không dùng CamelCase hay dấu gạch dưới
❌ /getAllUsers, /user_list

✅ /users
✅ REST **ưu tiên viết thường, dấu gạch ngang và dấu gạch chéo**
/articles/2024/latest-news

3. Dùng - (dấu gạch ngang) để phân cách từ
GET /user-accounts

4. Sử dụng query parameters cho filter, sort, pagination

GET /products**?category=phone&sort=price&page=2&limit=10**

#45. Get a Todo API - @PathVariable

Tạo url: **/todos**/todo-id

Ví dụ: /todos/1
/todos/2

1.PathVariable
https://www.baeldung.com/spring-pathvariable

@PathVariable là một annotation dùng để lấy dữ liệu từ URL.
Nó được dùng để trích xuất biến trong path URL và truyền vào phương thức controller.

Ví dụ:

```
@GetMapping("/users/{id}")
public String getUserById(@PathVariable Long id) {
   return "User ID là: " + id;
}
```

#46. POST Method - @PostMapping

1.Định nghĩa

@PostMapping là một annotation trong Spring MVC dùng để xử lý HTTP POST request.

👉 Dùng khi bạn muốn gửi dữ liệu từ client (giao diện, frontend) lên server để:

- Tạo mới dữ liệu (Create)
- Gửi form, đăng ký, đăng nhập
- Gửi JSON từ frontend

@RequestBody: lấy dữ liệu JSON từ request và convert thành object Java

2. Tại sao lại dùng POST, mà không truyền data lên url (GET) ?

1. Không an toàn (Security)

- URL có thể bị lưu lại trong lịch sử trình duyệt, file log, cache,...
- Password và dữ liệu nhạy cảm bị lộ dễ dàng!

  => Không nên gửi mật khẩu, số thẻ, token,... qua URL.

2. Giới hạn độ dài URL

- URL bị giới hạn (~2000 ký tự ở một số trình duyệt như Internet Explorer).
- Gửi dữ liệu lớn (nội dung bài viết, JSON, v.v) là không thể qua URL.

3. Không gửi được body (payload)

GET request không có phần body → không gửi JSON hoặc form-data được.

POST thì có thể gửi mọi định dạng: text, JSON, file, form,...

4. Không chuẩn REST khi tạo dữ liệu

#47. Create a Todo API - @RequestBody

//todo

#48. PUT Method - @PutMapping

@PutMapping là một annotation trong Spring được dùng để xử lý các HTTP PUT requests.

👉 Thường được dùng để cập nhật toàn bộ một tài nguyên đã tồn tại — ví dụ như cập nhật thông tin người dùng, sản phẩm, bài viết…

#49. Update a Todo API

//todo

#50. Phân biệt PUT và PATCH

Khác biệt	Giải thích
RESTful logic	PUT = yêu cầu gửi toàn bộ object để ghi đè PATCH = chỉ gửi các field cần thay đổi
Code xử lý khác nhau	PUT thường map toàn bộ DTO vào entity PATCH phải kiểm tra từng field (nếu có)
Frontend truyền khác nhau	PUT gửi đầy đủ mọi field PATCH chỉ gửi 1-2 field cần sửa

//todo

Minh họa với code Jhipster (update **Employee**)

#51. DELETE Method - @DeleteMapping

@DeleteMapping là một annotation trong Spring được dùng để xử lý HTTP DELETE request.
👉 Mục đích chính là để xóa một tài nguyên dựa trên ID hoặc một điều kiện nào đó.

#52. Delete a Todo API

//todo

X - Chapter 6: Testing với Spring

Hướng dẫn viết testcase cơ bản nhất (Unit Test, Integration Test) để test API với Spring

#53. Tổng quan về chapter

//todo

#54. Tại sao viết Code cần phải Test ?

1.Code cho chạy được, cơ mà cần trả ra đúng format mà frontend mong muốn

Khi viết API backend (như Spring Boot), có 2 mức độ:

✅ **Mức 1: Code chạy được**

- API không lỗi
- Trả về HTTP 200
- Có dữ liệu

Nhưng…

🚫 Nếu trả về sai format, frontend không dùng được!

✅ **Mức 2: Code chạy đúng + trả đúng format**

Frontend thường kỳ vọng:

Yêu cầu	Ví dụ
Đúng status code	201 khi tạo, 400 khi lỗi dữ liệu, 404 khi không tìm thấy
Đúng format JSON	{ "data": ..., "message": ..., "status": ... }
Có field rõ ràng	id, name, email,... không được thiếu
Có error message cụ thể	"email is already used"

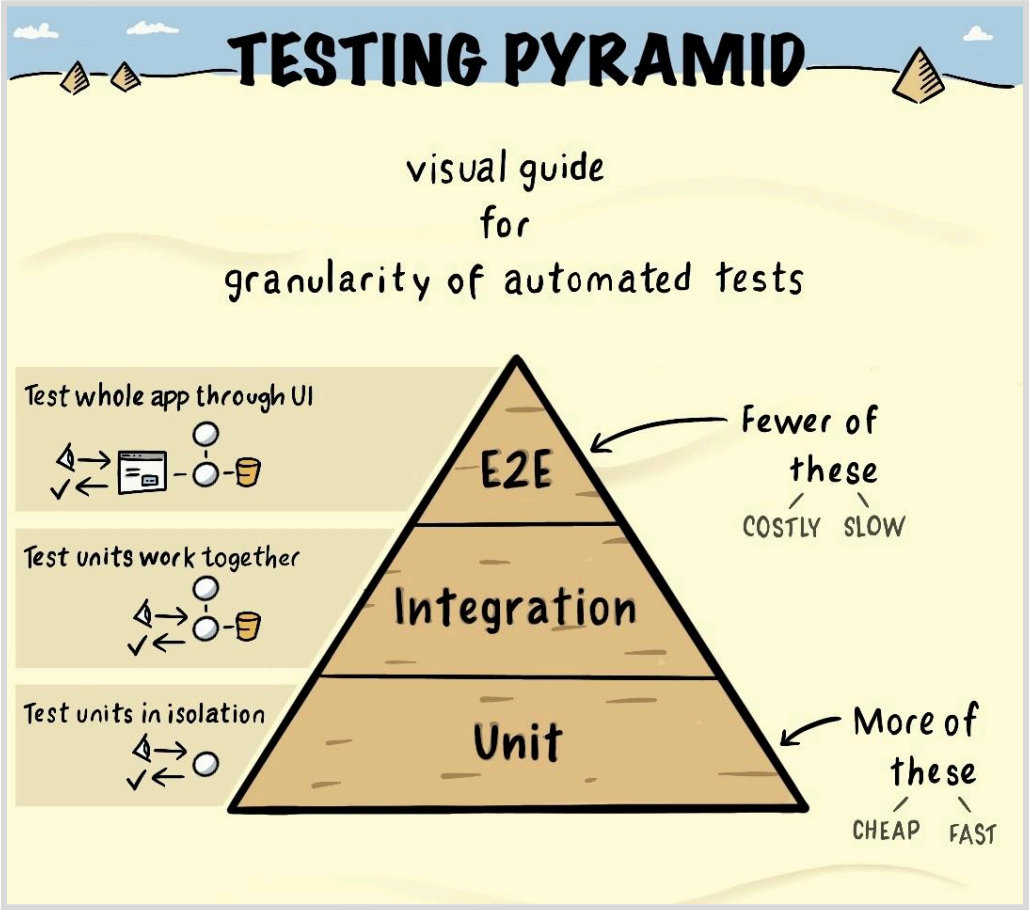
Vì sao cần test?

Vì mắt người không thấy được sai format nhỏ như:

- Thiếu trường status
- Gửi về null thay vì [ ]
- Dữ liệu không đúng kiểu (String thay vì int)
- Status HTTP sai: trả 200 thay vì 400

==> Viết test để kiểm tra response trả về có đúng như frontend mong muốn không ?

#55. Có bao nhiêu loại test ?



Kim tự tháp test được chia làm 3 tầng:

- Tầng đáy (nhiều nhất): **Unit Test** – nhanh, rẻ, đáng tin cậy
- Tầng giữa: **Integration Test** – test các thành phần hoạt động cùng nhau
- Tầng đỉnh (ít nhất): **UI/E2E (end to end) Test** – kiểm tra toàn bộ hệ thống, chậm và tốn kém

#56. Demo Test với ứng dụng JHipster
//todo

#57. Viết Unit Test Cơ Bản

Mục tiêu:

- Giúp bạn hiểu rõ cách viết Unit Test cơ bản.
- Không dùng Spring Boot context, không dùng database.

https://docs.spring.io/spring-boot/reference/testing/index.html

Ví dụ 1: Tính chiết khấu khi mua hàng

- Nếu tổng tiền < 100 → không giảm giá
- Nếu tổng tiền từ 100–499 → giảm 10%
- Nếu từ 500 trở lên → giảm 20%

Class cần test:

```
public class DiscountCalculator {

   public double calculateDiscount(double totalAmount) {
      if (totalAmount < 100) {
         return 0;
      } else if (totalAmount < 500) {
         return totalAmount * 0.10;
      } else {
         return totalAmount * 0.20;
      }
   }
}
```

Bước 1: Tạo class Test

//todo

Bước 2:

@Test

@assertEquals

#58. Quy trình viết Unit Test

1.Nên test thành phần nào ?

Controller => service => repository
Không bắt buộc phải test tất cả thành phần. Ưu tiên test những thành phần quan trọng, có nhiều logic xử lý

2.Cách viết code test AAA (Arrange – Act – Assert)

Giai đoạn	Ý nghĩa	Cách nhớ
Arrange	Chuẩn bị dữ liệu, mock, input	"Sắp xếp, chuẩn bị dữ liệu"
Act	Gọi hàm cần kiểm tra	"Hành động" – gọi hàm cần test
Assert	Kiểm tra kết quả có đúng không	"Khẳng định" – kiểm tra kết quả có đúng không

Tác dụng khi áp dụng mô hình AAA:

- Tập trung rõ mục tiêu: bạn nhìn vào test là biết đang test cái gì
- Dễ debug khi fail: biết lỗi ở bước nào: setup hay logic
- Đọc test code dễ hiểu hơn

3. Khái niệm Mock

Mock data là quá trình tạo ra dữ liệu giả lập, dùng để thay thế cho dữ liệu thật (Database, API, file...)
→ Giúp bạn test nhanh, an toàn, kiểm soát được kết quả.

Vì sao bạn cần mock data?

1. Test độc lập – Không phụ thuộc dữ liệu thật
Bạn test class UserService, nó gọi UserRepository, nếu không mock:

- Bạn phải có database thật
- Dữ liệu có thể thay đổi bất ngờ
- Test có thể thất bại không do lỗi code, mà do lỗi dữ liệu

Mock = giả lập phản hồi → không cần database thật

2. Test nhanh hơn
Gọi DB thật: 50–200ms/request

Mock: gần như 0ms

Test 1000 lần → tiết kiệm thời gian đáng kể!

3. Kiểm soát được kết quả đầu vào / đầu ra
Bạn tự định nghĩa dữ liệu:
when(userRepository.findById(1L)).thenReturn(Optional.of(mockUser));
→ Bạn biết trước kết quả, nên dễ assert, dễ phát hiện lỗi.

4. Không làm hỏng dữ liệu thật
Khi test, nếu không mock:
Gọi .save(...) vào DB thật → ghi dữ liệu rác

Gọi .delete(...) → xoá dữ liệu thật

Mock giúp an toàn tuyệt đối, không động vào môi trường thật.

5. Dễ tạo các tình huống kiểm thử đặc biệt
Ví dụ: Trả về null, empty, hoặc ném exception

4.Sử dụng Mock như thế nào ?

- Chỉ cần mock các thành phần phụ thuộc (dependencies)

- Không cần mock những gì đơn giản, an toàn, hoặc không gây side effect.

5. Chuẩn bị môi trường test
Download files sử dụng trong video tại đây
Mục tiêu: CRUD User

//todo: test api với postman

#59. Unit Test với JUnit và Mockito (Part 1)

1. Công cụ

Giới thiệu các annotation thường gặp và ý nghĩa của nó
https://site.mockito.org/

@Mock
@InjectMocks
@ExtendWith(MockitoExtension.class)

Quy tắc đặt tên:
File test: <Tên class>Test.java
Tên test method: <methodBeingTested>_should<expected>_when<condition>()

Ví dụ:
login_shouldReturnToken_whenCredentialsAreValid
updateUser_shouldThrowException_whenUserNotFound
deleteUser_shouldDoNothing_whenUserDoesNotExist

2. Thực hành
//todo

#60. Unit Test với JUnit và Mockito (Part 2)

//todo viết unit test cho service trong dự án

#61. Integration Test

1. Tổng kết về unit test

Tham khảo video #55

Vì sao viết unit test cho service ?
Service là nơi xử lý logic nghiệp vụ (business logic), và cần được test riêng biệt, nhanh, rõ ràng, dễ bảo trì:

- Tốc độ nhanh: không cần Spring context, không load database, chạy cực nhanh
- Cô lập logic nghiệp vụ: test chính xác từng rule, không bị nhiễu bởi controller, repository
- Dễ viết và bảo trì: không phụ thuộc environment, config
- Giúp phát hiện lỗi sớm: vì test từng phần nhỏ, khi logic thay đổi là phát hiện ngay

2. Integration Test

Integration Test = kiểm tra nhiều thành phần của hệ thống kết hợp với nhau, để đảm bảo luồng xử lý thật hoạt động đúng như mong đợi.

Tiêu chí	Unit Test	Integration Test
Kiểm thử cái gì?	1 class hoặc 1 method riêng lẻ (hoạt động độc lập)	Nhiều class/tầng kết hợp (controller → service → repo → DB)
Dùng thật hay mock?	Mock tất cả phụ thuộc	Dùng thật (hoặc 1 phần mock)
Nhanh/chậm	Rất nhanh	Chậm hơn (vì chạy Spring context, DB,...)
Độ chính xác	Kiểm tra logic chi tiết	Kiểm tra hoạt động của toàn bộ hệ thống
Dễ viết/bảo trì	Dễ	Phức tạp hơn

Khi nào dùng unit test, khi nào dùng integration test ?

//todo: demo về integration Test với jhipster ?

#62. Setup Test Database với Spring Profile

Mục tiêu: tạo profile cho môi trường test (tránh ảnh hưởng tới môi trường dev)

//todo: test container; h2

1. Về profile:
https://www.baeldung.com/spring-profiles#profiles-in-spring-boot

Tạo profile cho test
src/main/resources/**application-test.properties**
application-test.properties

spring.datasource.url=jdbc:mysql://localhost:3306/test_db
spring.datasource.username=test_user
spring.datasource.password=test_pass

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true

@**ActiveProfiles**("test")

#63. Quy trình viết Integration Test

1.Các bước thực hiện

Nên viết theo mô hình AAA (Arrange – Act – Assert) để mã test rõ ràng, dễ đọc, dễ bảo trì.

Quá trình test cho API, bao gồm:

- Tạo dữ liệu
- Gửi request với MockMvc
- Kiểm tra status HTTP
- Đọc response JSON bằng ObjectMapper (nếu có)
- Kiểm tra dữ liệu trong DB
- Rollback tự động bằng @Transactional (optional)

2. Giới thiệu các annotation hay dùng

@SpringBootTest: Khởi chạy toàn bộ Spring context

@AutoConnfigureMockMvc: Cấu hình MockMvc để giả lập gọi HTTP

https://docs.spring.io/spring-framework/reference/testing/mockmvc.html

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

//todo: test api = postman (đảm bảo api hoạt động bình thường)

#64. Jackson và Object Mapper (Extra)

Spring có thể chuyển đổi (convert) giữa JSON và Object (Java POJO) **một cách tự động** trong RESTful API là nhờ vào cơ chế serialization và deserialization **do thư viện Jackson đảm nhiệm.**

✅ Từ JSON (client) → Object khi server nhận request
✅ Từ Object (server) → JSON khi trả response cho client

1.Spring làm điều đó như thế nào?
Jackson là thư viện mặc định của Spring Boot
Khi bạn thêm Spring Web (spring-boot-starter-web), Jackson sẽ được tự động import.
//todo: minh họa với maven

2. Deserialization (JSON → Object)
Khi bạn có một endpoint nhận dữ liệu JSON như sau:

```
@PostMapping("/users")
public String createUser(@RequestBody User user) {
    // Spring tự động chuyển JSON từ request thành đối tượng User
    return "User: " + user.getName();
}
```

Dữ liệu từ client gửi lên (dạng JSON) sẽ được Jackson chuyển đổi thành đối tượng User nhờ @RequestBody.

3. Serialization (Object → JSON)
Ngược lại, nếu bạn return một Object từ controller:

```
@GetMapping("/users")
public User getUser() {
    return new User("hoidanit", 25);
}
```

Spring sẽ tự động chuyển đối tượng User thành JSON trước khi gửi về client.

4. Object Mapper

ObjectMapper là class chính của Jackson
(nằm trong package com.fasterxml.jackson.databind) dùng để:

- **Chuyển từ JSON → Java Object (read):**
  https://www.baeldung.com/java-text-blocks (java 15 trở lên)

```
String json = """
  {
    "name": "eric",
    "email": "hoidanit@gmail.com"
  }
  """;
```

  readValue(jsonString, Class<T>)

- **Chuyển từ Java Object → JSON (write):**

```
String json = objectMapper.writeValueAsString(user);
System.out.println(json);

byte[] jsonBytes = objectMapper.writeValueAsBytes(user);
```

Khi làm việc với API (MockMvc), sử dụng **writeValueAsBytes thay vì writeValueAsString**

Bởi vì Spring sử dụng byte để xử lý request. Nếu dùng String, Spring sẽ cần chuyển đổi từ String sang byte (có thể gặp lỗi nếu không xử lý chính xác ký tự đặc biệt)

#65. Viết Integration Test (Part 1)

//todo

https://spring.io/guides/gs/testing-web

#66. Viết Integration Test (Part 2)

//todo

https://stackoverflow.com/a/44589454

#67. Tối ưu Integration Test
Nguồn tài nguyên tham khảo: https://www.youtube.com/watch?v=u5foQULTxHM

1.Các annotation sử dụng thêm
//sử dụng transactional
//beforeEach

2. Nếu có nhiều controller phải test ?
//reuse annotation
//**IntegrationTest.java**
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@SpringBootTest()
@ActiveProfiles("test")
public @interface **IntegrationTest** {
}

//cách để chạy tất cả controller một lúc

./mvnw test

#68. Các loại test khác có thể gặp

Ngoài Unit Test và Integration Test, với backend developer, cần quan tâm thêm:

1.Security Test

- Kiểm tra role-based access (phân quyền)
- Check JWT, CSRF, token, session, login…

Ví dụ:

```
@WithMockUser(username = "admin", roles = {"ADMIN"})
@Test
void adminCanDeleteUser() throws Exception {
   mockMvc.perform(delete("/users/1"))
      .andExpect(status().isNoContent());
}

@Test
void anonymousShouldBeUnauthorized() throws Exception {
   mockMvc.perform(delete("/users/1"))
      .andExpect(status().isUnauthorized());
}
```

2. Performance Test

- Đảm bảo API phản hồi nhanh, ổn định khi có tải cao
- Phát hiện bottleneck, timeout

Công cụ:

- JMeter, Gatling, k6 → gửi nhiều request song song
- Spring Actuator + Micrometer để đo thời gian phản hồi

X - Chapter 7: Project thực hành 01

Thực hành dự án fullstack 01 (sử dụng Java Spring và React)

#69. Tổng quan về chapter

//todo

#70. Viết code Spring theo Implements

Mô hình hiện tại: Controller => Service => Repository

1.Ví dụ demo

//todo

2. Quy tắc đặt tên

Service là interface, dùng để định nghĩa hành vi

Class sẽ implement lại interface, định nghĩa cụ thể hành vi.

Hay viết tắt là Impl : ví dụ UserServiceImpl

3.Ưu, nhược điểm

Nên viết code sử dụng interface (implements) khi bạn muốn:

- Tách biệt rõ logic & định nghĩa
- Dễ mock, dễ test
- Mở rộng linh hoạt

Không bắt buộc sử dụng interface (implements) khi:

- Dự án nhỏ
- Logic đơn giản, chỉ dùng nội bộ
- Không cần mock, test nhiều

#71. Thực hành viết Service với Implements
//todo

#72. Format Response
Việc định dạng response một cách nhất quán là rất quan trọng để đảm bảo tính dễ đọc, dễ bảo trì và khả năng tương thích với các client.

Không bắt buộc phải format response, phụ thuộc vào yêu cầu và mong muốn của bạn.
1. Sử dụng cấu trúc JSON nhất quán
Một cấu trúc phổ biến bao gồm:

- Dữ liệu chính (**data**): Chứa thông tin mà client yêu cầu.
- Trạng thái (**status**): Mô tả trạng thái của request (thành công, lỗi, v.v.).
- Thông báo (**message**): Cung cấp thông tin bổ sung (nếu cần).
- Mã lỗi (**errorCode**): Nếu có lỗi xảy ra.

Ví dụ khi thành công:
```
{
  "status": "success",
  "message": "Dữ liệu đã được lấy thành công",
  "data": {
    "id": 1,
    "name": "Product A",
    "price": 100.0
  },
  "errorCode": null
}
```

Ví dụ khi có lỗi:
```
{
  "status": "error",
  "message": "Không tìm thấy tài nguyên",
  "errorCode": "RESOURCE_NOT_FOUND",
  "data": null
}
```

2. Tạo Response Wrapper

Download file Response wrapper tại đây

//todo:

- Xử lý trường hợp thành công
- Xử lý trường hợp exception

#73. Xử lý Exception

Mục tiêu:

✅Trả về lỗi có format chuẩn cho frontend

✅ Giao tiếp rõ ràng: message, errorCode, status, path…

✅ Không lộ stacktrace cho người dùng

✅ Tập trung lỗi về một chỗ (@ControllerAdvice, @ExceptionHandler)

1.Cách xử lý exception trong Spring

- **Try-catch cục bộ trong controller:** với lỗi đơn giản, nhỏ
- **@ExceptionHandler** (local): trong 1 controller cụ thể
- **@ControllerAdvice** (global): xử lý lỗi toàn hệ thống

@RestControllerAdvice = @ControllerAdvice + @ResponseBody

//todo: sửa lại controller sử dụng @RestController

2. Sử dụng Exception Handler

Ví dụ:

```
 @GetMapping("/users/{id}")
 public User getUser(@PathVariable Long id) {
     throw new EntityNotFoundException("Not found");
  }

  @ExceptionHandler(EntityNotFoundException.class)
  public ResponseEntity<?> handleNotFound(EntityNotFoundException ex) {
   //todo
  }
```

3. Global Exception Handler

//todo

#74. Xử lý Validation
Tài liệu: https://spring.io/guides/gs/validating-form-input
https://beanvalidation.org/

https://docs.jboss.org/hibernate/validator/8.0/reference/en-US/html_single/#section-builtin-constraints

Trong Spring, validation (xác thực dữ liệu) là một phần quan trọng để đảm bảo rằng dữ liệu đầu vào từ client (ví dụ: request body, query parameters) hợp lệ trước khi xử lý logic nghiệp vụ.

Spring hỗ trợ validation thông qua Bean Validation API với triển khai phổ biến là Hibernate Validator.

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

1.Sử dụng Annotation Validation
Spring sử dụng các annotation từ Bean Validation API để xác thực dữ liệu

//todo: phân biệt blank, empty, null, tham khảo tại đây

Validate email, tham khảo tại đây

2. Áp dụng Validation trong Controller

Sử dụng **@Valid** để kích hoạt validation trên tham số đầu vào.

Nếu validation thất bại, Spring sẽ ném ngoại lệ **MethodArgumentNotValidException.**

3. Xử lý Validation với @RestControllerAdvice

```
@ExceptionHandler(MethodArgumentNotValidException.class)
public ResponseEntity<ApiResponse<Object>>
handleValidationExceptions(MethodArgumentNotValidException ex) {
    List<String> errorList = ex.getBindingResult().getFieldErrors().stream()
            .map(error -> error.getField() + ": " + error.getDefaultMessage())
            .collect(Collectors.toList());
    String errors = String.join("; ", errorList);

    ApiResponse<Object> response = new ApiResponse<>(HttpStatus.BAD_REQUEST,
errors, null, "VALIDATION_ERROR");
    return new ResponseEntity<>(response, HttpStatus.BAD_REQUEST);
}
```

#75. Test APIs với Integration Test

Mục tiêu: Đảm bảo api chạy được, và cần phải trả ra đúng format mà frontend trông chờ

Download file test tại đây

#76. Test thành quả đạt được (full frontend + backend)

1.Trước khi thực hành

- Bạn đã xem và chạy Unit Test tại video #75, và đã pass tất cả test case.
  Điều này sẽ đảm bảo backend của bạn chạy được, và hoạt động đúng như yêu cầu của frontend mong muốn

- Bạn đã xem và cài đặt Node.js như hướng dẫn tại video #13, như vậy máy tính bạn sẽ có môi trường Node.js để chạy code Frontend

- Bạn không bắt buộc phải biết code Frontend, mình sẽ cung cấp sẵn full source code frontend để bạn test full dự án

  Các video tiếp theo sẽ hướng dẫn code frontend theo từng tính năng cụ thể

2.Thực hành

Bước 1: Download source code frontend (final) tại đây

Bước 2: Build dự án frontend

- Update file .env (cập nhật url backend)
- Cài đặt thư viện: **npm i**
- Build dự án: **npm run build**
- Chạy dự án frontend tại chế độ build: **npm run preview**

Kết thúc bước 2 này, sẽ bị lỗi CORS (Lỗi này sẽ được giải thích và fix triệt để tại các video tiếp theo)

Bước 3: Fix lỗi CORS

//update code backend
@CrossOrigin(origins = "*")

Fix phần **Global exception**s, được làm tại video #85

```
	/*
	 * handle the rest exceptions
	 */
	@ExceptionHandler(Exception.class)
	public ResponseEntity<ApiResponse<?>> handleAllException(Exception ex) {
		var result = new ApiResponse<>(HttpStatus.INTERNAL_SERVER_ERROR,
ex.getMessage(), null,
				"INTERNAL_SERVER_ERROR");

		return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(result);
	}
```

Bước 4: Test full Frontend/Backend
//todo

Bước 5: Rollback code

Nếu bạn code theo cách video tiếp theo, hãy xóa đi phần thay đổi tại **bước 3**, như vậy nó sẽ đảm bảo là code của bạn và code trong video hướng dẫn tại các video tiếp theo, là giống nhau

#77. Frontend: Setup dự án thực hành

1.Môi trường thực hiện
Đảm bảo rằng máy tính bạn đã cài đặt Node.js v**22.13.0**

Nếu bạn chưa cài đặt Node.js, tham khảo video #13

2.Yêu cầu trước khi thực hành

Cần phải có hiểu biết cơ bản về React và TypeScript.

Nếu bạn là beginners, chưa từng học React, có thể tham khảo:

- Khóa học React Ultimate tại đây
- Khóa học React Portfolio tại đây
- Lộ trình học React từ a tới z tại đây

3.Dự án thực hành

Download/Clone dự án thực hành tại đây

4.Giải thích về cấu trúc dự án thực hành
//todo

Lưu ý về bản quyền source code (license) khi sử dụng ?

#78. Frontend: Chia Layout

1.Cài đặt thư viện

Sử dụng chính xác:
npm i --save-exact antd@5.24.6 react-router@7.5.0

- Làm UI với antd: https://ant.design/
- Điều hướng trang (router) với React router: https://reactrouter.com/home

2.Chia base layout

Lưu ý: sử dụng antd v5 và react-router v7

Mục tiêu: /trang chủ và /users

Bước 1: chia base component theo routes
https://reactrouter.com/start/data/routing

//todo

Bước 2: tạo UI
//todo
https://ant.design/components/menu

npm i --save-exact @ant-design/icons@6.0.0

#79. Frontend: Tạo Table Users

Mục tiêu:

- Tạo layout (tái sử dụng header/footer)
  https://reactrouter.com/7.5.0/start/declarative/routing

- Tạo table user
  https://ant.design/components/table/

#80. Frontend: Cách gọi API của Backend

1.Gọi APIs từ phía Frontend

- AJAX (Jquery)
- Fetch (nodejs)
- Các thư viện khác: ví dụ axios

2.Axios
https://axios-http.com/docs/intro
https://www.npmjs.com/package/axios

npm i --save-exact axios@1.8.4

//todo: gọi API với axios

#81. Frontend: CORS là gì ?

1.Nguồn gốc (Origin) là gì?

Một origin được định nghĩa bởi sự kết hợp của **protocol** (giao thức như http, https), **domain** (tên miền như example.com), và **port** (cổng như 80, 3000).
Ví dụ:
http://example.com và **https://example.com** là hai origin khác nhau (khác protocol).

http://example.com:3000 và **http://example.com:4000** là hai origin khác nhau (khác port).

http://example.com và **http://sub.example.com** cũng là hai origin khác nhau (khác domain).

Khi một trang web ở một origin (ví dụ: **http://frontend.com**) gửi yêu cầu tới một origin khác (ví dụ: **http://api.com**), trình duyệt sẽ áp dụng chính sách **Same-Origin Policy** (chỉ cho phép yêu cầu trong cùng origin).

CORS là cách để nới lỏng chính sách này một cách an toàn.

2.CORS là gì ?

CORS (Cross-Origin Resource Sharing) là **một cơ chế bảo mật được tích hợp trong các trình duyệt web,** cho phép hoặc hạn chế các yêu cầu HTTP từ một nguồn gốc (origin) này tới một nguồn gốc khác.

Nó được thiết kế để bảo vệ người dùng khỏi các cuộc tấn công nguy hiểm như truy cập trái phép dữ liệu từ các trang web không đáng tin cậy, đồng thời vẫn cho phép các ứng dụng web hiện đại hoạt động linh hoạt.

Ví dụ về tác dụng của CORS:

Ví dụ : Bảo vệ người dùng khỏi tấn công CSRF (Cross-Site Request Forgery)

Bối cảnh:
Bạn đăng nhập vào trang ngân hàng **https://bank.com** và có một cookie phiên (session cookie) để xác thực.

Một trang web độc hại **http://evil.com** cố gắng gửi yêu cầu đến **https://bank.com/transfer?amount=1000&to=attacker** để chuyển tiền từ tài khoản của bạn.

Trường hợp 1: Không có CORS:
Trình duyệt tự động gửi cookie của **bank.com** cùng với yêu cầu từ **evil.com** vì cookie được lưu trong trình duyệt.

Nếu **bank.com** không kiểm soát origin, yêu cầu này có thể thành công, dẫn đến việc kẻ tấn công đánh cắp tiền của bạn mà bạn không hề hay biết.

Trường hợp 2: Với CORS:
Trình duyệt gửi **header Origin: http://evil.com** trong yêu cầu tới **bank.com.**

Server của **bank.com** kiểm tra và thấy **http://evil.com** không nằm trong danh sách Access-Control-Allow-Origin (chỉ cho phép **https://bank.com**).

Server từ chối yêu cầu bằng cách không trả về header Access-Control-Allow-Origin phù hợp hoặc trả về lỗi.

Kết quả: Yêu cầu bị chặn, dữ liệu của bạn được bảo vệ.

=> Muốn fix lỗi CORS, cần thực hiện tại backend (frontend chỉ dùng và không cần quan tâm tới lỗi này)

#82. Backend: Fix Lỗi CORS

Tài liệu:
https://docs.spring.io/spring-framework/reference/web/webmvc-cors.html
https://www.baeldung.com/spring-cors

1.Sử dụng @CrossOrigin

Sử dụng **@CrossOrigin** trên Controller hoặc Method

@CrossOrigin(origins = {"http://localhost:3000", "https://domain1.com"})

2.Sử dụng WebMvcConfigurer

```
@Configuration
public class WebConfig implements WebMvcConfigurer {

    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping("/**") // Áp dụng cho tất cả các endpoint
                .allowedOrigins("http://localhost:3000", "https://yourdomain.com") // Danh sách origins được phép
                .allowedMethods("GET", "POST", "PUT", "DELETE", "OPTIONS") // Các phương thức HTTP được phép
                .allowedHeaders("*") // Cho phép tất cả headers
                .allowCredentials(true) // Cho phép gửi cookie hoặc thông tin xác thực
                .maxAge(3600); // Thời gian cache CORS preflight request (giây)
    }
}

//todo: minh họa với JHipster
```

#83. Frontend: Hiển thị danh sách Users
//todo

#84. Frontend: Modal Create User

Mục tiêu:
- Sử dụng Modal
- Khi submit form, lấy được data user

https://ant.design/components/modal

#85. Frontend: Tạo Mới User
//todo

#86. Frontend: Update User
//todo

#87. Frontend: Delete User
//todo

#88. Nhận xét về dự án thực hành 01
//todo