Chương 1: TỔNG QUAN ĐỀ TÀI

1. Giới thiệu tổng quan và lý do chọn đề tài

* Thời trang là một trong nhưng điều không thể thiếu đối với mỗi người hiện đại ngày nay, nó không phải chỉ còn là một nhu yêu cầu thiết yếu đối với mỗi người mà còn là một loại trang sức, một vật trang trí thể hiện cá tính, vè đẹp,.. hoặc có thể che đi khuyết điểm của mỗi người. Với việc xã hội ngày nay ngày càng hiện đại hóa và con người dần có xu hướng làm đẹp nhiều hơn,thì thời trang là một ngành công nghiệp béo bở mà bất cứ một doanh nghiệp nào cũng mong muốn được phát triển. Từ đó các hàng thời trang lớn nhỏ cũng mọc lên như nấm, tiêu biểu các thương hiệu thời trang lớn với doanh thu khổng lồ trên thế giới như: Louis Vuitton(33.6 tỷ USD), Hermes(15.3 tỷ USD), Gucci(14.9 tỷ USD), Rolex(9.3 tỷ USD). Việt Nam là một trong những nước đang phát triển với tốc độ nhanh và thời trang cũng đang dần hội nhập với các nước phát triển, vì vậy nước là thị trường thời trang lớn và tiềm năng đối với các hãng thời trang lớn trên thế giới.
* Quần áo, phụ kiện, trang sức luôn là một chủ đề lớn trên các trang thương mại điện tử lớn cả trong và ngoài nước: tiki, shoppee, amazon… Cộng với việc người tiêu dùng đang ngày càng có xu hướng mua sắm online cũng như ảnh hưởng của đại dịch covid-19, nên việc nhiều hãng thời trang chuyển qua hình thức kinh doanh online cũng là điều tất yếu.
* Vì những lý do trên, em quyết định chọn đề tài phát triển website thời trang cho đồ án của mình

1. Mục tiêu và phương pháp phát triển đồ án
   1. Mục tiêu phát triển đồ án

* Tích lũy kinh nghiệm, kiến thức trong quá trình phát xây dựng đồ án
* Phát triển đầy đủ tính năng 1 trang web thời trang, có thể ứng dụng trang web vào các dự án thực tế
* Project có thể đáp ứng yêu cầu của nhà tuyển dụng khi phỏng vấn tìm việc làm
* Cố gắng tích hợp các công nghệ mới vào project: thanh toán online, kết nối với facebook, gmail, …
  1. Phương pháp phát triển đồ án
* Lựa chọn các các công nghệ mới và nổi bật để phát triển trang web.
* Chọn lọc dữ liệu từ các trang web thời trang khác để thao tác database.
* Thu thập ý kiến của thầy cô, bạn bè để tạo nội dung cho trang web.
* Tham khảo các chức năng, UI từ các trang web uy tính để ứng dụng vào trang web.

1. Đối tượng và phạm vi phát triển đồ án

* Cào dữ liệu thông tin: tên sản phẩm, chất liệu, giá thành, hình ảnh, màu sắc, kích thước, … các sản phẩm thuộc lĩnh vự thời trang: quần áo, đồng hồ, giày, áo khoác, …
* Continue

1. Bố cục báo cáo

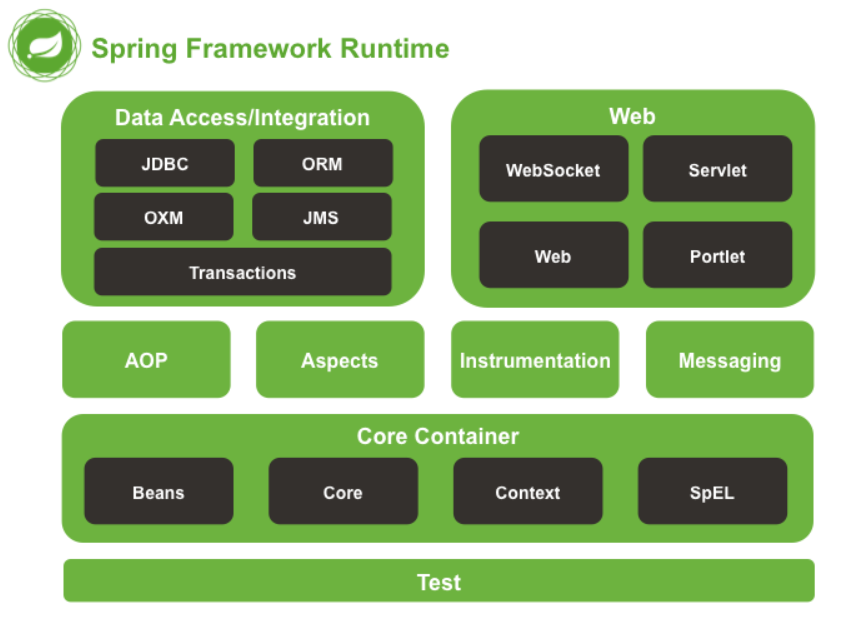
Chương 2: CƠ SỞ LÝ THUYẾT

1. Giới thiệu

* Chương này sẽ giới thiệu tổng quan về các c công nghệ được áp dụng để phát triển trang web bao gồm:
  + Back End: Spring boot
  + Front End: Angular
  + Database: Mysql

1. Cơ sở lý thuyết
   1. Spring boot
      1. Spring Framework

* Spring framework được viết bởi Rod Johnson vào năm 2000 và lần đầu tiên được ra mắt vào tháng 6 năm 2003 bởi Rod Johnson .
  + Spring là framework mã nguồn mở để xây dựng các ứng dụng phần mềm ứng dụng java.
  + Spring được phát triển dựa trên nền tảng Java EE và nó là framework phổ biến nhất trong cộng đồng java.
  + Spring nhẹ, trong suốt và hoạt động dựa trên 2 nguyên tắc design là: Dependency Injection và Aspect Oriented Programing.
  + Lợi ích của Spring framework:
    - Spring cho phép chúng ta sử dụng POJOS. Với POJOS chúng ta không cần phải làm việc với EJB. Bên cạnh đó việc cấu hình môi trường và các luồn chạy cũng đơn giản hơn rất nhiều.
    - Spring hỗ trợ sử dụng nhiều công nghệ: ORM Framework, logging framework,…
    - Spring được tổ chức theo kiểu module.
    - Sẽ rất dễ dàng để kiểm thử chương trình viết bằng Sping.
    - Web framework của spring là một Web MVC framework có thiết kế tốt.
    - …
  + Kiến trúc của Spring: Spring được chia làm 20 module khác nhau và các module này được phân nhóm thành các module lớn: Core Container, Instrumentation, Data Access/Integration, Messaging, Test, Web, AOP.



Hình 1. Cấu trúc Spring Framework

* Test: Tầng này cung cấp khả năng, môi trưởng hỗ trợ kiểm thử với Junit và TestNG.
* Spring Core Container: bao gồm các module spring core, beans, expression language
  + Spring core, bean thực hiện chức năng IOC và Dependency Injection.
  + Spring Context hỗ trợ đa ngôn ngữ, các tính năng Java EE như EJB, JMX.
  + Expression Language hỗ trợ việc getting/setting các giá trị, method, cho phép truy cập collections, toán tử logic…
* AOP, Aspects, Instrumentation, Messaging: hỗ trợ cài đặt lập trình hướng khía cạnh(Aspect Oriented Programming), hỗ trợ tích hợp với AspectJ.
* Data Access/ Integration: hỗ trợ giao tiếp với Database và nhóm này bao gồm JDBC, ORM, OXM, JMS và module Transaction.
* Web: bao gồm Web, Web-Servlet và nhóm này hỗ trợ việc tạo ứng dụng web hay còn được gọi là Spring MVC.
  + 1. Spring IoC Container, Dependency Injection
* IoC (Inversion of Control) container và DI (Dependency Injection), Bean được coi là cốt lỗi của Spring framework.
* Ioc giúp cho lập trình viên quản lý quá trình thực hiện DI trong ứng dụng một cách tự động.
* Còn đối tượng tạo nên xương sống của ứng dụng sử dụng spring là Bean và nó được quản lý vở Spring IoC Container dựa trên nguyên lý DI.
* Dependency Injection(DI):
  + Dependency Injection là 1 kỹ thuật, 1 design pattern cho phép giảm,xóa bỏ sự phụ thuộc giữ các denpendency với nhay. Và thay vào đó chúng ta sẽ có thêm 1 nhiệm vụ là tiêm sự phụ thuộc(Injection dependency)
  + VD: Ta có 1 class Person, người này là người rất thích giày. Vì vậy trong class Person ta tạo 1 object class là Shoes.





Vào 1 ngày đẹp trời người này không thích mang giày nữa mà chuyển qua thích mang dép thì ta phải làm sao. Ta tạo một Person mới với object Class Slipper, tuy nhiên khi sử dụng dependency injection, chúng ta có thể thay đổi giày hoặc dép một cách dễ dàng bằng cách tạo object interface ShoeOrSlipper cho class Person và cho 2 class Slipper, Shoes implement interface ShoeOrSlipper. Để mổi khi t muốn thay đổi việc mang giày hay dép ta chỉ cần inject dependency(Shoes hoặc Slipper) vào. Vì vậy đối tượng person không còn phụ thuộc vào class Shoes hay Slipper nữa mà chỉ phụ thuộc vào ShoeOrSlipper và ta có thể tùy chình việc sử dụng shoe hoặc slipper cho Person









* + Có 3 loại Dependency Injection:
    - Constructor Injection: các dependenncy được cung cấp thông qua constructor.
    - Setter Injection: tạo ra 1 setter method để các class khác có thể sử dụng để cấp dependency.
    - Interface Injection: mỗi class muốn inject gì thì chỉ cần implement interface chứa hàn inject.
  + Lợi ích của Dependency Injection:
    - Dễ viết Unit test hơn.
    - Mở rộng dự án dễ dàng hơn.
    - Tạo liên kết lỏng(loose coupling) giữa các thành phần trong dự án.
* Inversion of Control: IoC giúp chúng ta quản lý và inject các dependency cho chúng ta
  + Spring framework là một framework ra đời để thực hiện ý tưởng Inversion of Control. Tuy nhiên, theo thời gian, Spring lớn mạnh và trở thành một hệ sinh thất lớn với rất nhiều chức năng trên nền tàng IoC này
  + IoC giống như một cái kho để chứa toàn bộ dependency, bean mà chúng ta định nghĩa trước đó, và khi ta muốn inject một đối tượng nào đó, thì IoC sẽ giúp chúng ta tìm trong cái kho đó rồi lấy ra cho chúng ta sử dụng.
  1. Spring Boot
* Spring boot là một dự án phát triển dựa trên ngôn ngữ Java nổi bật trong hệ sinh thái Spring Framework.
* Là framework được tạo ra để đơn giản việc cấu hình, bằng cách tự cấu hình tất cả bằng cách cung cấp các hành vi cụ thể.
* Nó đơn giản hóa việc deploy bằng cách đóng gói app thàng gói jar để dễ dàng tích hợp vào các web container.
* Các lợi ích của Spring Boot:
  + Dễ dàng tạo ra một Rest service độc lập.
  + Dễ dàng phát triển các ứng dụng dựa trên Spring.
  + Giảm thiểu thời gian phát triển và tăng năng xuất.
  + Dễ dàng tương tác các ứng dụng Spring Boot với các hệ sinh thái .của Spring như JDBC, ORM, Spring Data JPA, Spring Security.
  + Nó cung cấp nhiều plugin để phát triển và test ứng dụng Spring Boot nhanh chóng sử dụng các công cụ Build như: Maven và Gradle.
  + Nó cung cấp nhiêu plugin để làm việc với các cơ sở dữ liệu nhúng và cơ sở dữ liệu trên bộ nhớ một cách dễ dàng.
    1. Annotation
* Annotation trong Spring là các chú thích được đánh dấu trên mỗi class, mỗi method hoặc là trên mỗi field, để đánh dấu chức năng cho class hoặc method đó, annotation được biểu diễn như sau: @ + tên annotation, ví dụ: @Bean, @Service…
* Annotation được sử dụng ở 3 dạng:
  + Chú thích cho trình biên dịch.
  + Chú thích cho quá trình build.
  + Chú thích qua quá trình chạy chương trình.
    1. Restful API
* Restful API là tiêu chuẩn được dùng trong việc thiết kế API có các ứng dụng web hay còn gọi là thiết kế Web services để tiện cho việc quản lý các resources.
* API (Application Programming Interface): là một tập các quy tắc và cơ chế mà một ứng dụng hay là một thành phần tương tác với ứng dụng và thành phần khác hoặc là tương tác với database. Và kết quả dữ liệu trả về thường là kiểu dữ liệu JSON hặc XML.
* REST (Representational State Transfer): được giới thiệu vào năm 2000 bởi tiến sẽ Roy Thomas Fielding. Là một dạng chuyển đổi cấu trúc dữ liệu, kiểu kiến trúc để viết API. Nó sử dụng phương thức HTTP để tạo ra sự giao tiếp giữa các máy, các chương trình. REST quy định cách sử dụng các HTTP method: GET, POST, PUT, DELETE… và cách định dạng URL cho ứng dụng web để quản lý các resource. Nhờ vào việc biết được URL và HTTP method đó các lập trình viên cũng cảm thấy dễ dàng hơn trong việt quản lý resource.
* Các HTTP method phổ biến:
  + GET (SELECT): trả về một resource hoặc danh sách các resource.
  + POST (CREATE): tạo mới một resource.
  + PUT (UPDATE): cập nhập resource.
  + DELETE (DELETE): xóa một resource.
* Restfull API trong Spring Boot (Routing):
  + Trước khi các nghệ về Restfull API chưa được biết đến, lập trình viên thường phải trả về cả 1 trang view khi thực hiện 1 request nào đó đến application. Và trong Spring công nghệ để làm điều này được gọi là Spring MVC.
  + VD:
  + Nhưng với sự ra đời của Restfull API, Spring đã cung cấp tính năng, công nghệ để thực hiện Restfull Web Service, đó là Spring Boot.
  + Và để phân biệt được 2 loại công nghệ này trong Spring thì ta phải đánh dấu một Annotation cho mỗi loại công nghệ. Đối với Spring MVC ta phải đánh dấu class chứa URL request là @Controller, còn đối với Spring Boot ta phải đánh dấu là @RestController để phân biệt.
  + VD:
  + Tương ứng với 4 HTTP method GET, POST, PUT, DELETE. Spring boot cũng tạo ra 4 annotation hay còn gọi là Restfull API CRUD để thể hiện cho 4 loại HTTP method này là: @GetMapping, @PostMapping, @PutMapping, @DeleteMapping.
  + Và để trả về kết quả dưới dạng JSON với 4 HTTP method trên thì mỗi method ta có thể trả về kiểu dữ liệu ResponseEntity hoặc là JSONObject, 2 kiểu dữ liệu trả về này sẽ giúp chúng ta tự động parse JSON cho chúng ta mà không cần sử dụng một thư viện parse JSON nào khác.
  + @RequestMapping: là annotation có mục đích sử dụng rộng nhất, được sử dụng trong trường hợp ta muốn tất cả các method đều có thể dùng chung một cách xử lý.
  + @GetMapping: là annotation với mục đích đánh dấu hàm xử lý GET request trong controller. Và tất nhiên để trả về resource mà client yêu cầu thì client thường phải gửi cho Server một Request nào đó, ví dụ: mã người dùng, tên người dùng,… Và để phương thức chứa annotation @GetMapping hứng được các request từ client thì thường sẽ có 2 annotation để làm được điều này là: @PathVariable(“tên request param”) và @RequestParam(“tên request param”). Mặc dù 2 annotation trên đều chung nhiệm vụ là lấy request từ phía client nhưng cách thể hiện API URL và ý nghĩa có chút khác nhau.
    - @PathVariable: thường chỉ phù hợp với các phương thức GET có 1 request param và tên request param không cần tường minh, ví dụ như ID product. Và nó được thể hiện trên API URL sau dấu ‘/’. Trong Spring để lấy request bằng @PathVariable ta cần phải định nghĩa ở 2 vị trí đó là tên request bên trong thuộc tính name của @GetMapping và @PathVariable.
      * VD:

@GetMapping(“/get/find\_product\_by\_id/{id}”)

public ResponseEntity<Object> findProductById(@PathVariable(“id”) Long id)

* Ở đây ta thấy id định nghĩa trong @GetMapping phải được bọc trong cặp dấu ngoặc nhọn và tên request trong @GetMapping và @PathVariable là phải giống nhau.
* @RequestParam: phù hợp với các phương thức GET có nhiều request param. Và được thể hiện trên API URL sau dấu ‘?’, ‘&’.
  + VD:

@GetMapping(“/get/find\_product\_by\_id\_or\_name)

public ResponseEntity<Object> findProductByIdOrName(@RequestParam(“id”) Long id, @RequestParam(“name”) String name)

* + Đối với @RequestParam thì ta không cần phải định nghĩa tên request trong name API URL như @PathVariable.
  + @PostMapping: là annotation với mục đích đánh dấu hàm xử lý POST request trong controller. Không giống như @GetMapping sẽ để lộ thông tin các request params trên URL, @PostMapping sẽ không để lộ các thông tin request params, vì vậy ngoài mục đích tạo mới một resource, các phương thức POST còn thường được dùng để làm việc với các tác vụ cần bảo vệ thông tin như: đăng nhập, cập nhập thông tin... Request param phương thức POST được gửi tử Client lên Server thường có định dạng JSON và để Server nhận được các request params đó ta phải gắn annotation @RequestBody phía trước parameter của phương thức POST để parse dữ liệu kiểu JSON sang kiểu dữ liệu thường. Và các param của request gửi lên server thường phải được truyền vào bằng collection Map để lưu trữ theo kiểu key value giống với JSON hoặc là ta phải tạo một class request trên Server và phải đảm bảo tên của các field trong class request đó phải giống với tên KEY của dữ liệu JSON truyền từ client và phải có phương thức Getter, Setter cho mỗi field để đảm bảo việc convert dữ liệu từ request param sang các field của request class.
    - VD:

@PostMapping(“/post/create\_new\_product”)

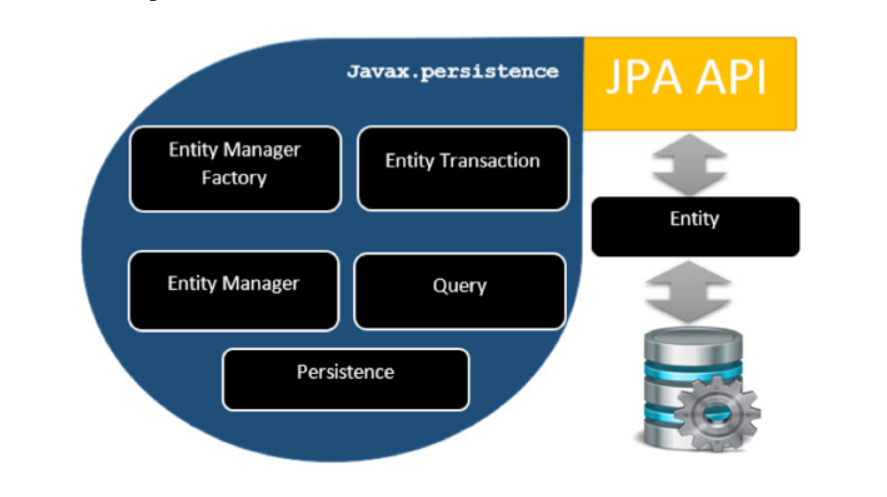
public ResponseEntity<?> createNewProduct(@RequestBody Map<String, String> requestParam){}

hoặc

@PostMapping(“/post/create\_new\_product”)

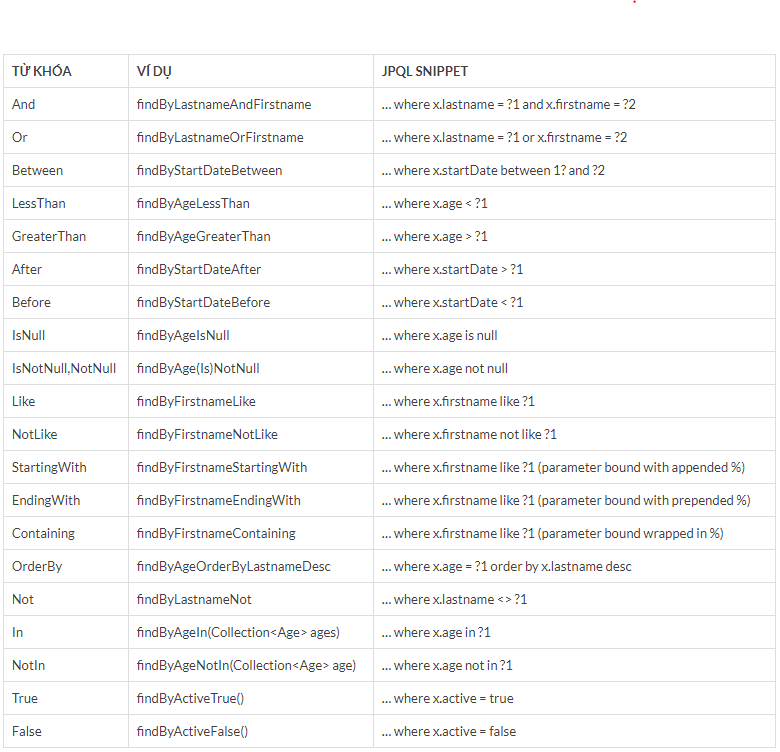
public JSONObject createNewProduct(@RequestBody CreateNewProductRequest request){}

với CreateNewProductRequest là class chứa các field tương ứng với KEY của dữ liệu JSON trong request.

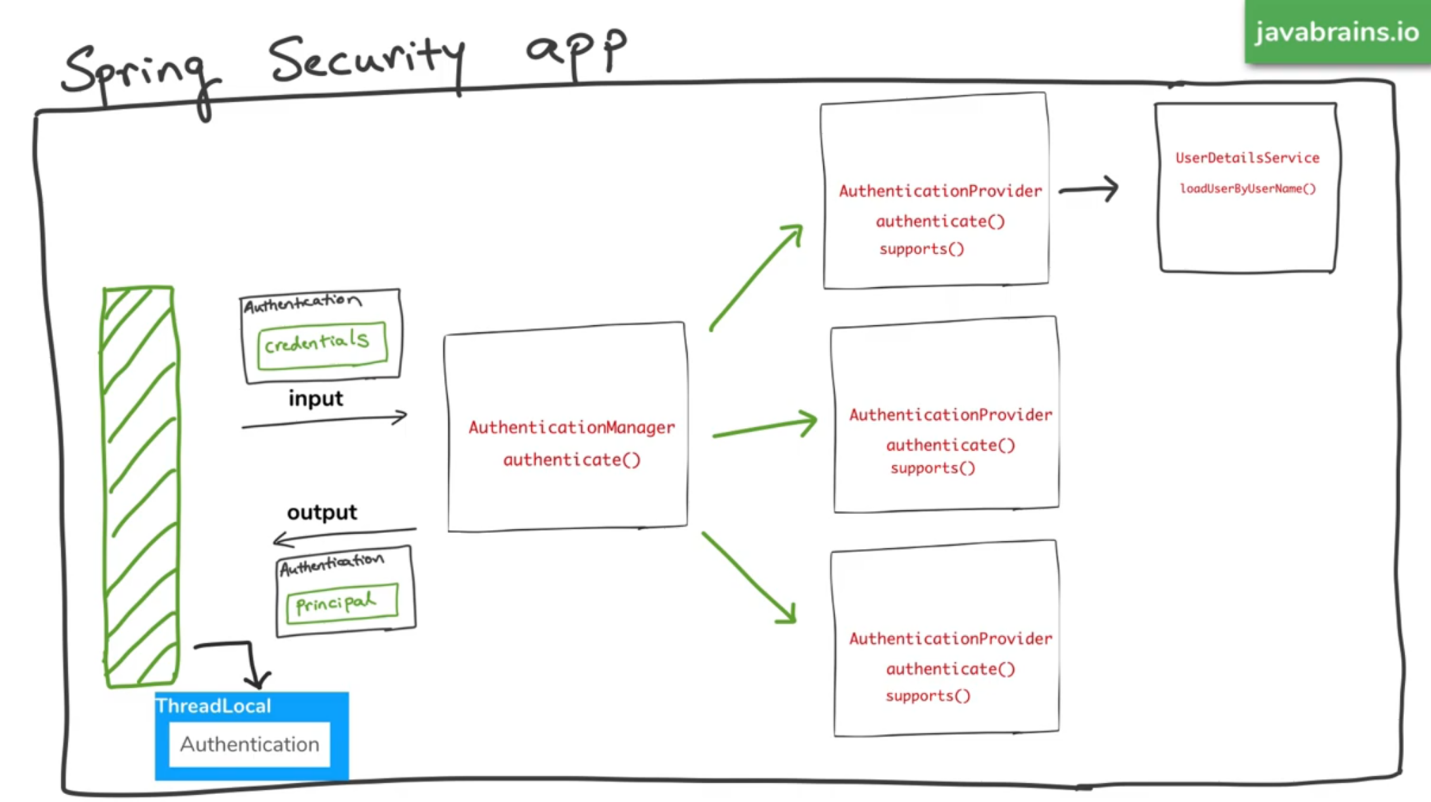
* + @PutMapping: là annotation với mục đích đánh dấu hàm xử lý PUT request trong controller. Tương tự như @PostMapping, nó cũng sử dụng @RequestBody để nhận request từ client.
  + @DeleteMapping: là annotation với mục đích đánh dấu hàm xử lý DELETE request trong controller. Tương tự như @GetMapping, nó cũng sử dụng 2 annotation là @Pathvariable và @RequestParam để nhận request từ client.
  1. Spring Data JPA vs MySQL
* Với bất kỳ một application nào thì việc thực hiện các thao tác query đến database là điều bắt buộc phải có và trong các java application việc này phải viết rất nhiều code để làm được.
* Trước đây, với Java, ta có công nghệ JDBC để làm việc với Database. Nhưng việc sử dụng JDBC có rất nhiều khuyết điểm từ quy trình đến hiệu năng. Với JDBC, ta phải thực hiện rất nhiều việc: tạo kết nối Database, tạo các Statement, ResultSet,… và sau khi thực hiện xong ta phải đóng tất cả những thứ này lại, dẫn đến code chở nên rất cồng kềnh và khó để có thể bảo trì và mở rộng. Và còn nhược điểm lớn khác nữa của JDBC là nếu phải thay đổi Database của ứng dụng thì ta cũng phải chỉnh sửa lại các các SQL để phù hợp với Database mới, điều này sẽ tốn rất nhiều thời gian, chi phí và nhiều rủi ro khác. Và để khắc phục được điều này, nhiều Framework đã ra đời với mục đích xóa đi vấn đề tương thích giữa các Database và giảm thiểu code SQL và giúp lập trình viên trập trung vào việc xử lý nghiệp vụ. Spring Data JPA là 1 trong những framework được ra đời để xử lý vấn đề này.
  + 1. JPA
* JPA (Java Persistence API) là một tập các tiêu chuẩn của Java để làm việc với cơ sở dữ liệu quan hệ một các đơn giản và nhanh chóng.
* JPA là một đặc tả cho việc ánh xạ giữa các dối tượng java các cơ sở dữ liệu quan hệ sử dụng công nghệ ORM. Nó có thể dễ dàng persist một đối tượng Java vào trong cơ sở dữ liệu hoặc ánh xạ dữ liệu từ cơ sở dữ liệu và ánh xạ vào các đối tượng Java. Nó hoạt động như một cầu nối giữa các table trong Database với các class trong Java. Từ đó mỗi khi muốn tuy vấn vào các table hoặc column của table đó thì ta chỉ cần gọi các thương thức, các field của class mà không cần quan tâm đến việc đang dùng loại Database nào.
* Kiến Trúc JPA: JPA bao gồm 3 phần chính là Entity, EntityManager và EntityManagerFactory và một số thành phần khác như EntityTransaction, Persistence, Query.

Hình ?. Kiến trúc JPA

* Entity: là các đối tượng thể hiện, ánh xạ 1 table trong cơ sở dữ liệu. Entity thường là các class POJO đơn giản, chỉ gồm các field và phương thức getter, setter.
  + Đặc điểm entity:
    - Entity có thể tương tác với cơ sở dữ liệu quan hệ.
    - Được xác định thông qua id, tương được với khóa chính trong table.
    - Hỗ trợ transaction
    - Có thể kế thừa giống class bình thường.
* EntityManager: là một interface cung cấp các API để tương tác với các Entity.
  + Một số chúc năng cơ bản:
    - Persist: dùng để lưu thực thể mới vào cơ sở dữ liệu.
    - Merge: dùng để cập nhập trạng thái vào cơ sở dữ liệu.
    - Remove: xóa một instance của entity.
* EntityManagerFactory: dùng để tạo ra instance của EntityManager.
* Persistence: định nghĩa một tập hợp các Entity được quản lý bởi 1 instance của EntityManager trong ứng dụng.
* EntityTransaction: để quản lý thao tác và thực hiện Transaction với mỗi thao tác database.
* Query: là một interface được mỗi nhà cung cấp JPA implement để có được các đối tượng quan hệ đáp ứng các tiêu chí truy vấn.
  + 1. ORM
* ORM (Object Relational Mapping) là một công nghệ, một kỹ thuật lập trình để chuyển đổi dữ liệu từ ngôn ngữ hướng đối tượng sang cơ sở dữ liệu quan hệ và ngược lại.
* ORM có khả năng xử lý thực hiện nhiều loại câu lệnh query khác nhau mà không cần quan tâm đến loại Database nào đang được sử dụng.
  + 1. Spring Data JPA
* Việc kết hợp JPA với Spring Framework tạo ra Spring Data JPA.
* Spring Data JPA là một module trong một project gọi là Spring Data project. Mục tiêu của Spring Data project là giảm thiệu việc lặp đi lặp lại các thao tác với hệ thống quản trị data khi phát triển ứng dụng bằng Spring frameword.
  + - 1. Entity trong Spring Data JPA
* Để chú thích 1 class là 1 Entity thì ta sử dụng annotation @Entity cho class đó. Nhưng nhiệm vị chính của Entity là biến đổi một class thông thường thành POJO (chỉ chứa các field và phương thức getter, setter) để chúng ta có thể thao tác với database. Còn việc ánh xạ Entity này với table nào đó trong Database thì ta phải sử dụng annotation @Table.
* Để chú thích 1 field ánh xạ với 1 column của Database ta sử dụng annotation @Column
* Để chú thích 1 filed ánh xạ với column ID trong table ta có thể sử dụng annotation @Id hoặc sử dụng kiểu dữ liệu UUID cho field khóa chính. Với annotation @Id, ta thường đi kèm với annotation @GeneratedValue để tạo giá trị ID tự động tăng lên 1 khi tạo mới một thực thể vào Database.
  + - 1. Quan hệ dữ liệu giữa các Entity trong Spring Data JPA
* Có 3 loại quan hệ trong cơ sở dữ liệu quan hệ: One to Many, Many To One, One To One. Và ứng với mỗi lại quan hệ trên Spring Boot cũng cung cấp các annotation tương ứng để thể hiện quan hệ giữa các entity ánh xạ với quan hệ của các table trong Database.
* @OneToMany và @ManyToOne: thể hiện quan hệ 1 nhiều giữa 2 Entity. Annotation @OneToMany được chú thích với một Collection field của entity có quan hệ một, @ManyToOne được chú thích với field của entity có quan hệ nhiều.
* @OneToOne: thể hiện quan hệ 1 1 giữa 2 Entity.
* @ManyToMany: Thể hiện quan hệ nhiều nhiều giữa 2 Entity, nhưng annotation này chỉ được sử dụng trong trường hợp table trung gian chỉ chứa khóa chính của của 2 table có quan hệ many to many. Trong trường hợp ta muốn khai báo thêm các thuộc tính khác trong table trung gian, thì ta phải tạo một Entity trung gian và khai báo các thuộc tính khác trong Entity trung gian này bằng các annotation @Column, và các khóa chính của các Entity có quan hệ many to many phải được khai báo trong một class khác có chú thích annotation @Embeddable và sau đó khai báo trong Entity trung gian với annotation @EmbeddedId.
* Một số tính năng trong các Annotation quan hệ dữ liệu:
  + mappedBy: là tính năng của @OneToMany. Để liên kết với đối tượng của lớp Entity có quan hệ @OneToMany, được khai báo bên Entity có quan hệ @ManyToOne. MappedBy còn được sử dụng cho @ManyToMany.
  + cascade: là tính năng của @OneToMany, giúp quản lý trạng thái của các đối tượng trong quan hệ một cách tự động. VD: Khi ta xóa dữ liệu của đối tượng có quan hệ @OneToMany thì toàn bộ dữ liệu của đối tướng có quan hệ @ManyToOne cũng sẽ xóa theo.
    - 1. Repository
* Repositoy là một trong những thứ cốt lỗi của Spring Data JPA, nơi chúng ta sẽ thực hiện truy vấn database, nhưng thay vì chúng là phải viết các câu lệnh SQL trên database thì ta chỉ cần viết các phương thức bằng ngôn ngữ Java dựa trên các Entitiy theo quy tắc của Spring Data JPA.
* Spring Data JPA cung cấp cho chúng ta 3 interface repository là: CrudRepository, PagingAndSortingRepository, JpaRepository. Tất cả 3 interface này đều extends từ một thằng interface chúng là Repository interface và mỗi interface này sẽ có thêm các tính năng khác nhau để phù hợp với mục đích mà nó hướng tới.
* Theo thứ tự từ lớn tới bé thì: JpaRepository interface extends PagingAndSortingRepository interface và PagingAndSortingRepository interface extends CrudRepository.
* CrudRepository interface cung cấp các phương thức CRUD cơ bản, PagingAndSortingRepository cung cấp các phương thức về việc phân trang và sắp xếp, JpaRepository interface có mọi phương thức trong PagingAndSortingRepository và CrudRepository và một số phương thức cơ bản khác.
  + - 1. Query Creation
* Query Creation là một phương pháp tạo các câu lệnh truy vấn dữ liệu theo tên method trong Spring Data JPA.
* Bạn chỉ cần đặt tên đúng chuẩn theo quy tắc của Spring Data JPA thì nó sẽ tự động sinh ra các cậu query tương ứng.
* Trong interface CrudRepository hay JpaRepository, chỉ cung cấp cho ta 1 số phương thức cơ bản và được sử dụng phổ biến như: findAll(), findById()… Tuy nhiên trong trường hợp bạn muốn nhiều field và điều kiện khác nhau thì Query Creation sẽ giúp chúng ta giải quyết vấn đề. Và tất cả phương thức Query Creation phải được khai báo trong các interface được kế thừa thừ 1 trong 3 Repository interface trên.
* Các phương thức Query Creation trong Spring Data JPA phải được đặt tên theo quy tắc camelCase(viết hoa chữ cái đầu tiên của mỗi từ trừ từ đầu tiên). Và các field được sử dụng để tạo các phương phức Query Creation là các biến được khai báo trong Entity được gán annotation @Column, @Id…
* Trong phương thức có thể có một số biệt thức kết hợp như: AND hay OR, các biểu thức so sánh Between, Like, In,… Sắp xếp kết quả bằng biểu thức OrderBy.
* VD: findByFirstName(String name), findByIdAndLastName(int id, String lastName);

Hình. Các biểu thức trong Query Creation

* 1. Spring Security
* Spring security cũng là một trong những dự án nổi bật của hệ sinh thái Spring. Nó cung cấp các dịch vụ bảo mật toàn diện cho các ứng dụng Spring có nền tảng jave EE.
* Spring security có 2 cơ chế cơ bản là : Authentication và Authorization.
  + Authentication (xác thực): là tiến trình xác thực người dùng (principal) khi đăng nhập vào hệ thống.
  + Authorization (phân quyền): là tiến trình xác thực quyền hạn người dùng. Nó quyết định xem một principle có được phép truy cập, sử dụng tài nguyên hệ thống hay không. Quá trình Authorization được thực hiện sau khi xác thực đăng nhập authentication.
    1. SecurityContext
* Là interface cốt lỗi của Spring Security, nó lưu trữ tất cả các thông tin chi tiết liên quan đến bảo mật trong ứng dụng. Khi chạy chương trình Spring Security thì SecurityContext cũng được kích hoạt theo.
* Tuy nhiên chúng ta không thể truy cập trực tiếp vào SecurityContext, mà thay vào đó ta sẽ sử dụng lớp SecurityContextHolder. Lớp này lưu trữ security context của ứng dụng, bao gồm tất cả thông tin người dùng (principal) của ứng dụng.
  + 1. UserDetail và UserDetailService
* UserDetail cũng là một trong những interface cốt lỗi của Spring Security. Nó đại diện cho principal và cung cấp cho ta các phương thức để làm việc với principal:
  + getUsername(): trả về username đã thực hiện authentication.
  + getPassword(): trả về password đã thực hiện authentication.
  + getAuthorities(): trả về danh sách các quyền của principal.
  + isEnabled(): trả về true nếu người dùng thực hiện authentication.
  + isAccountNonLocked(): trả về true nếu người dùng chưa bị khóa.
  + isAccountNonExpired(): trả về tru nếu principal chưa hết hạn.
  + isCredentialsNonExpired(): trả về true nếu mật khẩu của principal hết hạn.
* Như ta thấy ở trên, UserDetails chỉ cung cấp cho ta các phương thức, các thuộc tính về thông tin cơ bản của người dùng. Để có thể mở rộng thêm các thông tin cho người dùng, ta có thể tạo ra một Class implemets org.springframework.security.userdetails.UserDetails interface, và thêm các thông tin khác tùy ý.
* UserDetails chỉ là interface cung cấp thông tin người dùng, còn để thực hiện việc load dữ liệu của UserDetails ta phải cung tạo ra implementation của UserDetails. Và đó là lúc chúng ta sử dụng UserDetailsService interface. UserDetailsService chỉ cung cấp một phương thức duy nhất là: “UserDetailsDetails loadUserByUsername(String username) throws UserNameNotFoundException”.
  + Phương thức này nhận tham số truyền vào là username, sau đó nó sẽ tìm kiếm username đó trong Database. Nếu không tìm thấy, nó sẽ ném ngoại lệ UserNameNotFoundException.
  + Nếu username được tìm thấy thì phương thức loadUserByUsername() sẽ trả về một implementation của UserDetails. Và implementation này có thể là: ‘org.springframework.security.core.userdetails.User’ hoặc là class implement ‘org.springframework.security.userdetail.UserDetails’.
* Và để có thể thực hiện được phương thức ‘loadUserByUsername(String username)’ ta phải tạo ra 1 Clsas implements UserDetailsService.
  + 1. Authentication trong Spring Security
* Để có thể thực hiện cơ chế Authentication trong Spring Security ta phải tiến hành config bằng cách tạo ra 1 Class kế thứa abstract Class WebSecurityConfigurerAdapter và đánh dấu annotation @EnableWebSecutity và @Configuration cho class đó.
* Inject các dependency cần thiết như: UserDetailService, AuthenticationManager…
* Override một số phương thức của class WebSecurityConfigurer
* protected void configure(AuthenticationManagerBuilder auth): xác thực username và password bằng cách lấy UserDetail từ phương thức loadUserByUsername() của UserDetailsService interface và mã hõa mật khẩu nếu có.
* public PasswordEncode passwordEncoder(): để mã khóa mật khẩu ta phải tạo ra một bean mã hóa mật khẩu. Để tạo ra bean, ta tạo một phương thức trả về PasswordEncoder được đánh dấu annotation @Bean.
* public AuthenticationManager authenticationManagerbean(): kích hoạt Authentication, và được quản lý bởi AuthenticationManager interface.
* Quá trình Authentication được quản lý bởi AuthenticationManager interface. Và để thực hiện xác thực, ta sử dụng phương thức ‘authenticate(new UsernamePasswordAuthenticationToken(String username,String password))’. Phương thức sẽ throw DisabledException nếu phương thức isEnabled() là false, và throw BadCredentialsException nếu Username hoặc Password không đúng.

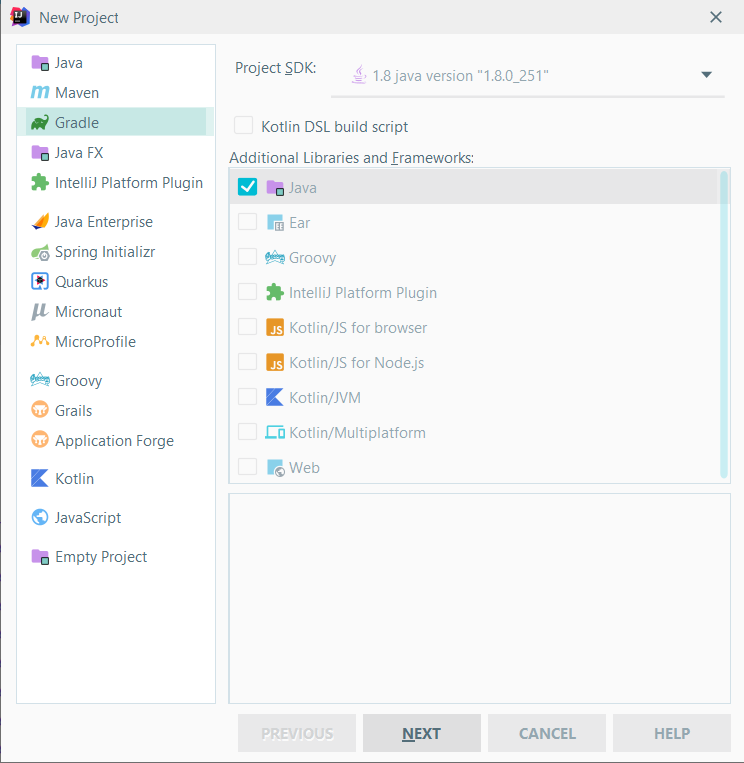


Quá trình Authentication.

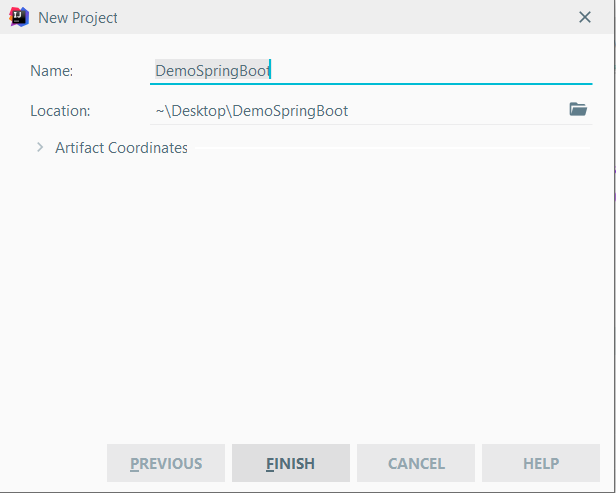
* + 1. JWT
* JWT (JSON Web Token) là phương tiện yêu cầu chuyển giao thông tin dạng JSON được định nghĩa theo chuẩn RFC 7519 giữa Client và Server. JWT gồm 3 phần: header, payplad, signature.
  + Header: chứa kiểu dữ liệu, và thuật toán sử dụng để mã hóa ra chuỗi JWT.
  + Payload: chứa các thông tin mà bạn muốn đưa vào Token.
  + Signature: được tạo ra bằng cấp mã hóa header, payload và 1 chuỗi secret.
* Tất cả 3 phần của JWT Token phải được mã hóa bằng một thuật toán nào đó và được gộp lại với nhau theo cấu trúc: header.payload.signature.
* JWT token được sử dụng sau khi cơ chế Authentication được thực hiện thành công. Sau đó, dựa trên thông tin User đó để đưa nội dụng cần thiết vào phần payload của JWT Token . Một phần không thể thiếu khi đưa thông tin của User vào Token và Authorities để xác thực quyền người dùng trong cơ chế Authorization. Và thông tin token sau khi được mã hóa thường sẽ được đưa vào header response khi gửi cho client để đảm bảo tính bảo mật.
* VD:
  + JWT sau khi được mã hóa: eyJhbGciOiJIUzUxMiJ9.eyJhdXRob3JpdGllcyI6W3siYXV0aG9yaXR5IjoiU1VQRVJfQURNSU4ifV0sInN1YiI6InF1YW5ndGluaCIsImlhdCI6MTYwNDEzMzI1OCwiZXhwIjoxNzYxODQzNjAwfQ.T-kVc3YKGA0yemyD0mGZvNMhW9ZzaIxuiZ5KwiuRdMzRWeAx3TGz1bNL2nNgioWyXyyiJ2s3SHMhakM\_uimsPw
  + JWT ở dạng JSON:

Hình JWT dạng JSON

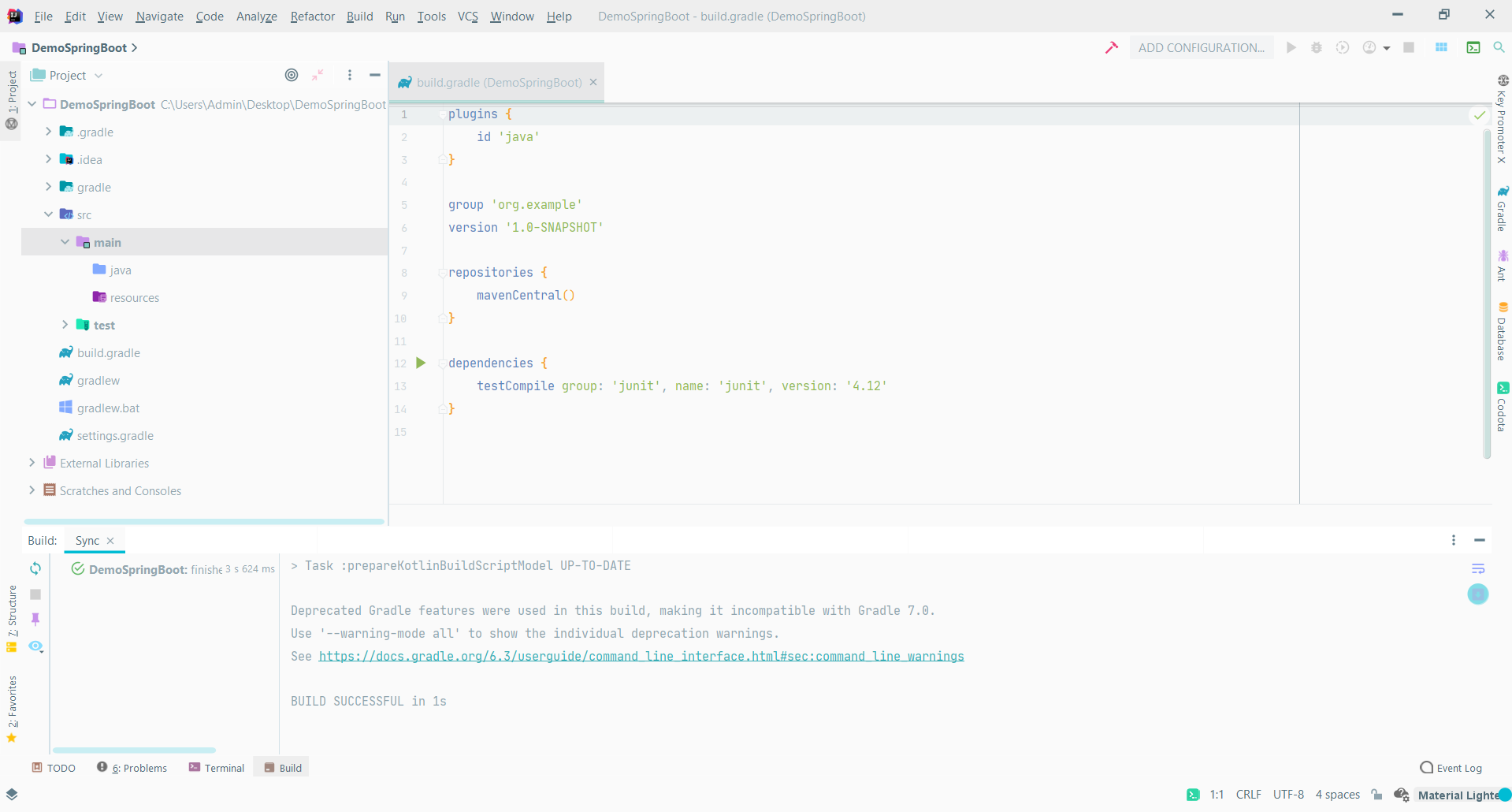
* + 1. Authorization trong Spring Security
* Authorization trong Spring Security là cơ chế xác thực người dùng dựa trên authrities của người dùng sau khi thực hiện cơ chế Authentication. Và để sử dụng được cơ chế Authorization ta cũng phải config, và việc config có thể thực hiện chung với Class config của cơ chế Authentication.
* Override phương thức: ‘protected void configure(HttpSecurity http)’ để tiến thực hiền cơ chế xác thực người dùng.
* Một số phương thức của HttpSecurity được sử dụng cho cơ chế Authorization:
  + cors(): enable cơ chế bảo mật Cross-Origin Resource Sharing.
  + addFilter(): lọc request trước khi xác thực quyền người dùng.
  + antMatchers(): danh sách các routing name cần để xác thực Authorization.
  + permetAll(): cho phép tất cả các user sử dụng routing được khai báo trong antMatchers().
  + hasAnyAuthority(): cho phép các user có authority được khai báo trong phương thức này sử dụng routing được khai báo trong anyMatchers().
* Mỗi request từ client lên server để lấy API đều phải được xác thực Authorization để đảm bảo tính bảo mật của trang web. Và để có thể duyệt qua tất cả các request từ client lên server ta phải tạo ra 1 class kế thừa abstract class OncePerRequestFilter. Ví dụ trong trường hợp này ta tạo ra class RequestFilter.
* Trong class RequestFilter ta cần phải thực hiện Override phương thức doFilterInternal() và inject vào phương thức addFilter() ở trên để có thể lấy được Authorities để tiến hành thực hiện cơ chế Authorization.
* Trong phương thức doFilterInternal(), ta sẽ lấy được authorities bằng cách extract token trong header từ HttpServletRequest của phương thức doFilterInternal() và lưu vào SecurityContextHolder. Từ các Authorities này ta sẽ đối chiếu với các Authorities được khai báo trong phương thức hasAnyAutohrity() như đã đề cập ở trên.
  1. Gradle
* Gradle là công nghệ tự động hóa việc build, quản lý thư viện; biên dịch, đóng gói chương trình. Nó là sự kết hợp, cải thiện những điểm yếu của 2 công nghệ cùng ngành khác là: Apache Ant và Apache Maven.
* Gradle hỗ trợ build cho nhiều ngôn ngữ khác nhau như: Java, Python, Android, IOS…
* Ưu điểm:
  + Gradle thừa hượng được của của 2 công nghệ cùng ngành khác là Ant và Maven
  + Sử dụng Groovy trong việc giải quyết bài toán đống gói nên việc build script trở nên ngắn gọn và dễ đọc hơn
  + Gradle hỗ trợ nhiều tính năng để cải tiện tính đóng gói cũng như tăng năng suất cho nhà phát triển.
  + Quá trình build nhanh hơn nhiều so với Maven, thì Gradle không cần phải xóa toàn bộ artiface và đóng gói lại từ đầu như Maven mà chỉ cần chạy lại những thay đổi trong source code.
  + Tích hợp nhiều test services như: AppThwack, Manymo
  + …
  1. Demo
* Tạo ứng dụng Spring Boot sử dụng Gradle bằng ItelliJ.



* Click NEXT



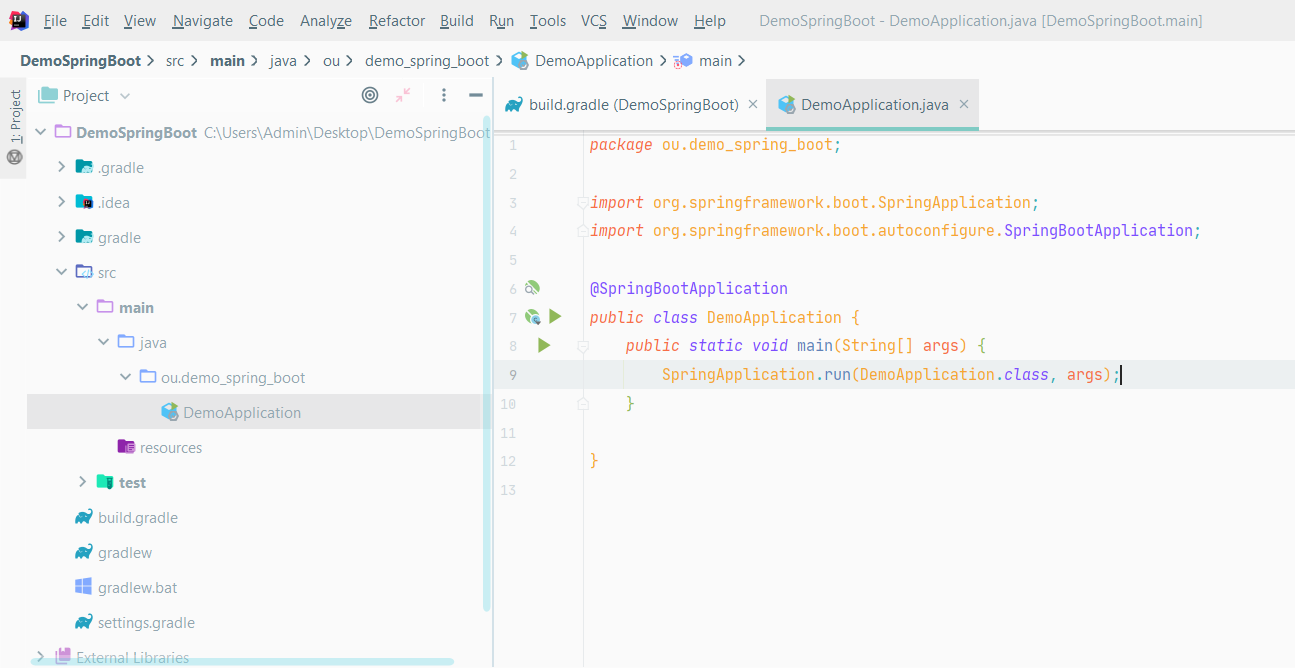
* Click FINISH
* Giao diện sau khi được tạo project thành công bằng ItelliJ.



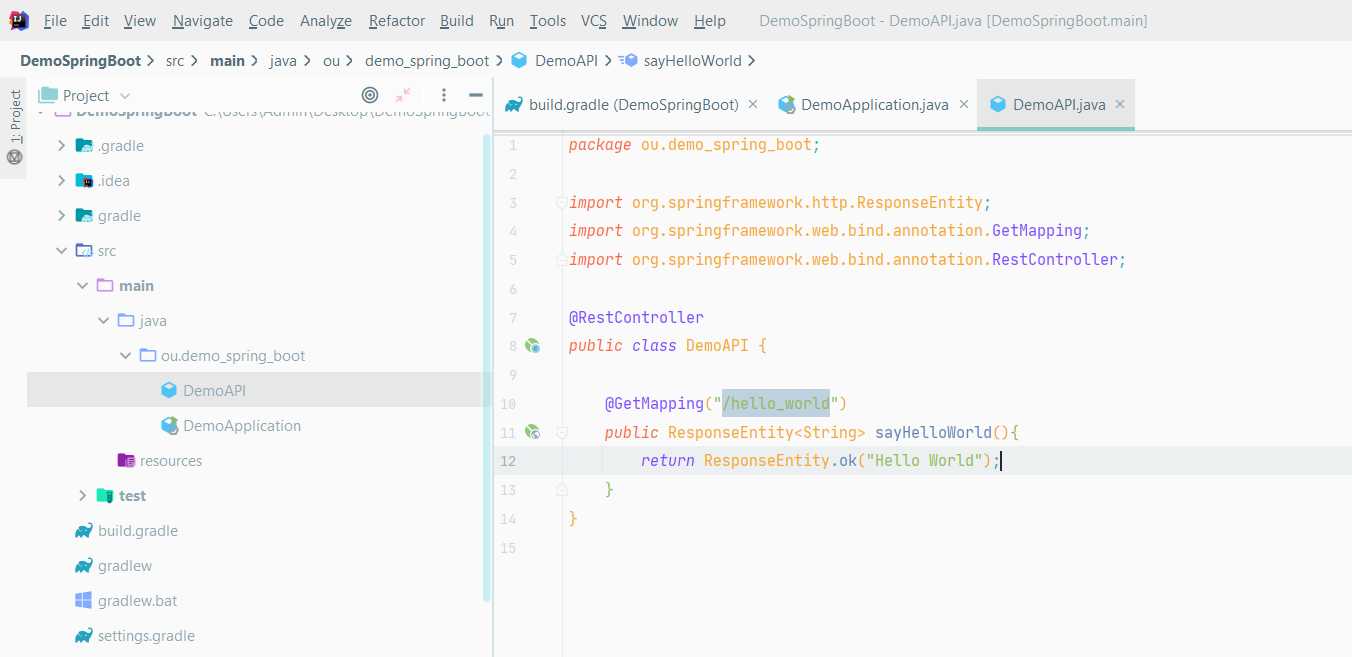
* Để có thể chạy được ứng dụng spring boot ta phải thêm 1 thư viện vảo file build.gradle là: spring-boot-starter-web



* Tạo Class DemoApplication với annotation @SpringBootApplication trong DemoSpringBoot/src/main/java/ou.demo\_spring\_boot để ghi phương thức main để chạy chương trình.



* Tới bước này ta đã có thể chạy được 1 ứng dụng Spring boot với url: <http://localhost:8080>.
* Và để tạo ra một API thì ta tạo Class DemoAPI với annotation @RestController với các phương thức có annotation có thể là: @GetMapping, @PostMapping, @GetMapping, @DeleteMapping.



* Chạy chương trình và tiến hành test API bằng Postman.