

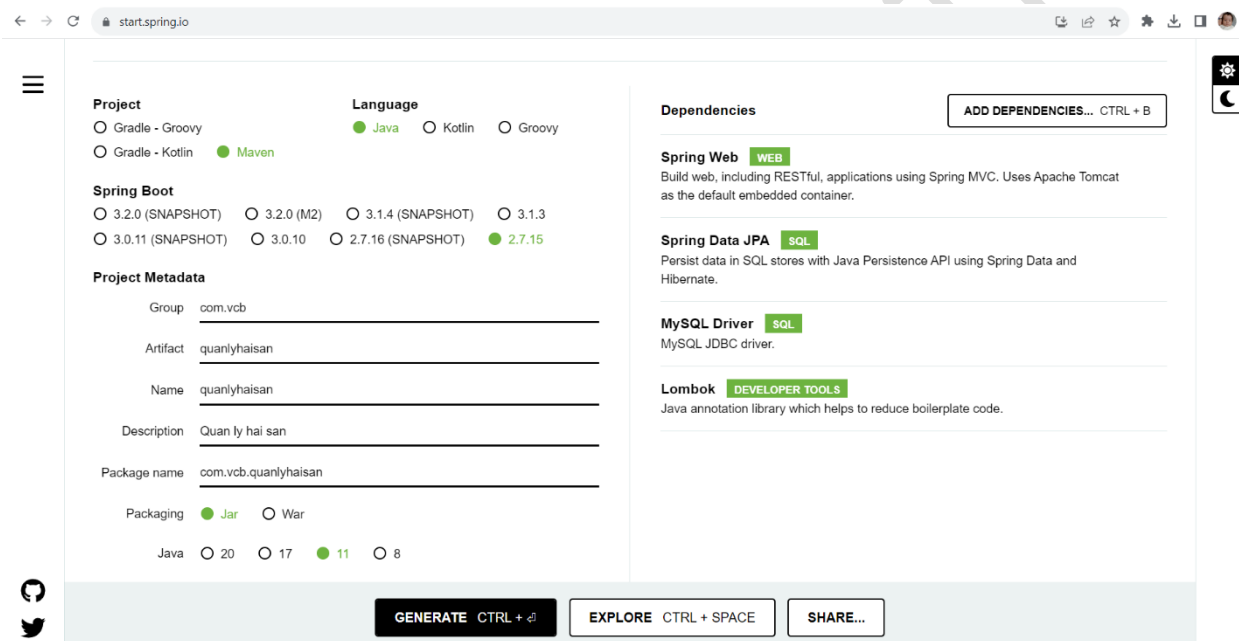
Hướng dẫn phát triển ứng dụng QLHS FullStack – Phần I API GetAll

Giáo viên: Nguyễn Hùng Cường

Bước 1: Tạo ứng dụng SpringBoot và database

Vào trang start.spring.io, đây là trang cho phép tạo nhanh các project SpringBoot. Tiếp theo ta tạo mới một Project SpringBoot, sau đó add các dependencies như hình bên dưới.

Sau đó ta mở MySQL, tạo mới một cơ sở dữ liệu với tên là quanlyhaisan và tạo bảng seafood gồm 4 cột: id, name, age, address.

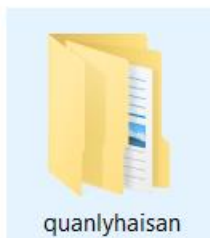


The screenshot shows the Spring Boot Start page with the following configuration:

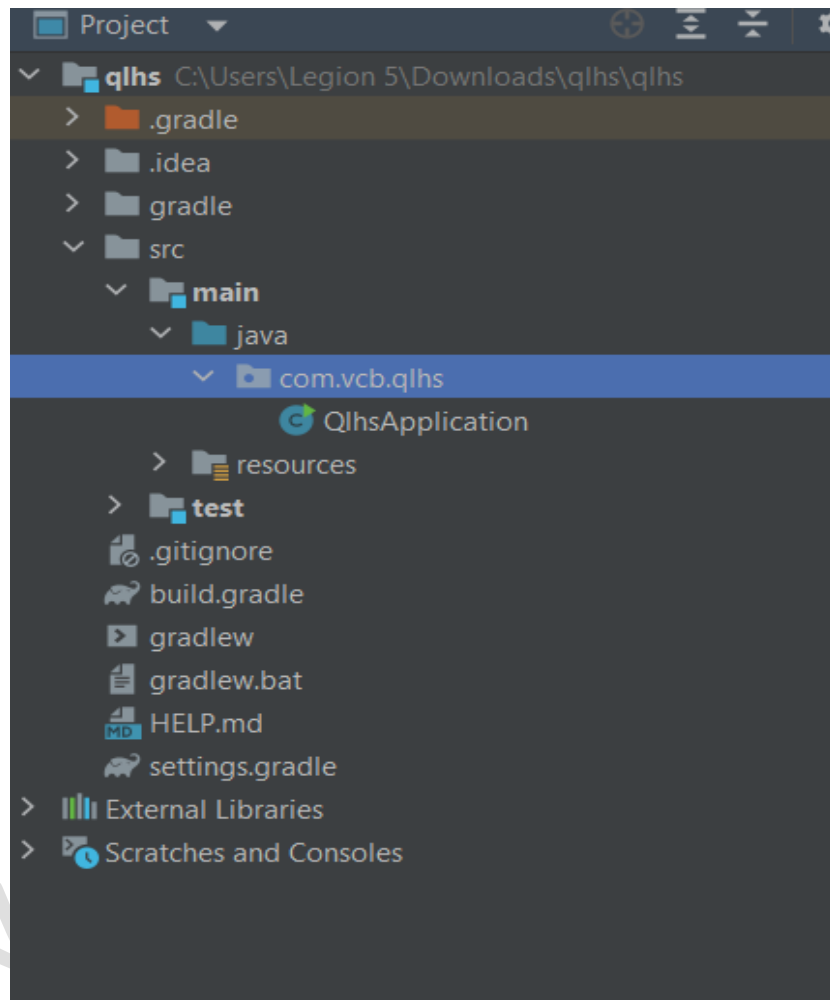
- Project:** Gradle - Groovy, Gradle - Kotlin, **Maven**
- Language:** **Java**, Kotlin, Groovy
- Spring Boot:** 3.2.0 (SNAPSHOT), 3.2.0 (M2), 3.1.4 (SNAPSHOT), 3.1.3, 3.0.11 (SNAPSHOT), 3.0.10, 2.7.16 (SNAPSHOT), **2.7.15**
- Project Metadata:**
 - Group: com.vcb
 - Artifact: quanlyhaisan
 - Name: quanlyhaisan
 - Description: Quan ly hai san
 - Package name: com.vcb.quanlyhaisan
 - Packaging: **Jar**, War
 - Java: 20, 17, **11**, 8
- Dependencies:**
 - Spring Web** (WEB): Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
 - Spring Data JPA** (SQL): Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.
 - MySQL Driver** (SQL): MySQL JDBC driver.
 - Lombok** (DEVELOPER TOOLS): Java annotation library which helps to reduce boilerplate code.

Buttons at the bottom: **GENERATE** (CTRL + G), **EXPLORE** (CTRL + SPACE), **SHARE...**

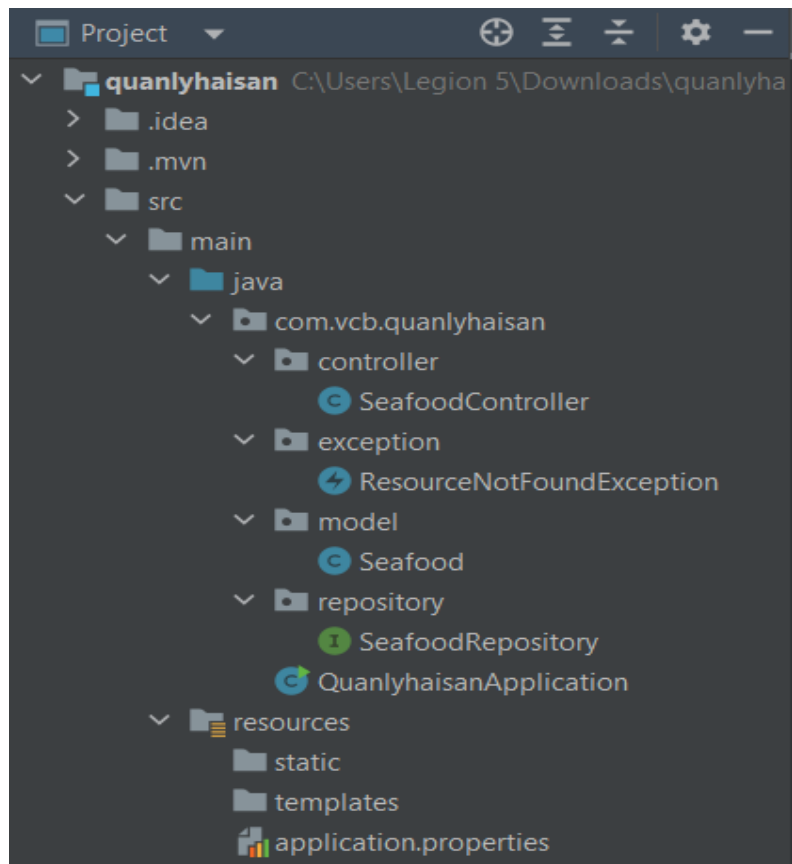
Sau đó nhấn Generate, ta sẽ thấy project đã được tải về máy mình. Ta giải nén ra và sẽ được thư mục có chứa cấu trúc project.



Ta có thể mở ứng dụng ra, và sẽ thấy cấu trúc thư mục của project đã được hiển thị như hình bên dưới.



Tiếp theo, ta tạo các package và các file Java trong cấu trúc thư mục của project SpringBoot như sau.



Bước 2: Viết mã SpringBoot

Tiếp theo, ta mở file config của ứng dụng SpringBoot ra, file có tên là application.properties nằm trong thư mục resources. Sau đó ta cấu hình cho project như sau, đây là phần cấu hình kết nối đến CSDL MySQL quanlyhaisan tại máy cục bộ, với tài khoản là root và mật khẩu rỗng.

```
1 spring.datasource.url=jdbc:mysql://localhost:3306/quanlyhaisan?useSSL=false
2 spring.datasource.username=root
3 spring.datasource.password=
4
5 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL57Dialect
6 spring.jpa.hibernate.ddl-auto=update
```

Tiếp theo, ta định nghĩa class Seafood nằm bên trong package model để thể hiện các bản ghi Seafood như sau.

```

Seafood.java x
7  import lombok.Setter;
8
9  @Setter
10 @Getter
11 @AllArgsConstructor
12 @NoArgsConstructor
13 @Entity
14 @Table(name="seafood")
15 public class Seafood {
16     @Id
17     @GeneratedValue(strategy = GenerationType.IDENTITY)
18     private long id;
19     @Column(name = "name")
20     private String name;
21     @Column(name = "price")
22     private int price;
23     @Column(name = "address")
24     private String address;
25 }

```

Sau đó ta tạo tiếp interface SeafoodRepository để cho phép thực hiện các thao tác CRUD dữ liệu với bảng seafood trong database.

```

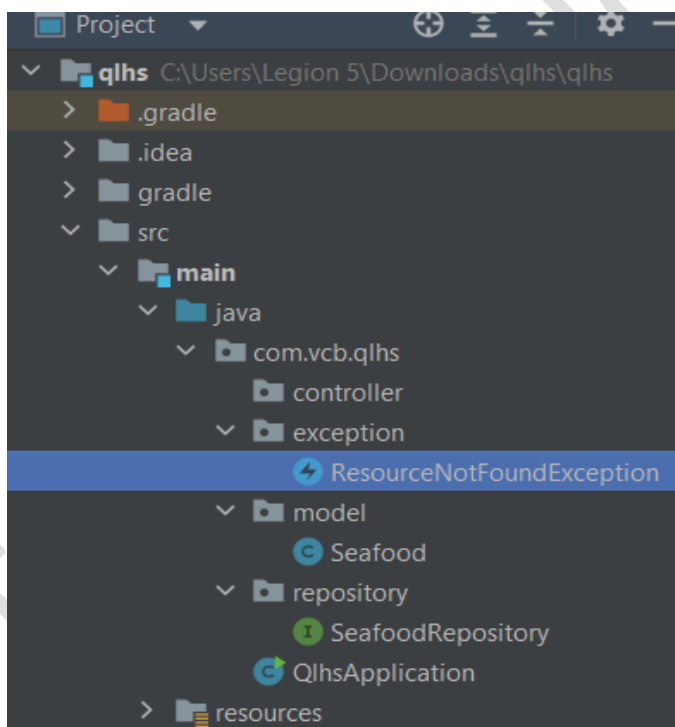
SeafoodRepository.java x
1  package com.vcb.qlhs.repository;
2
3  import com.vcb.qlhs.model.Seafood;
4  import org.springframework.data.jpa.repository.JpaRepository;
5
6  no usages
7  public interface SeafoodRepository extends JpaRepository <Seafood, Long> {
8  }

```

Ta cũng định nghĩa một class để thể hiện các Exception có thể được ném ra trong quá trình thao tác như sau.

```
ResourceNotFoundException.java x
1 package com.vcb.qlhs.exception;
2
3 import org.springframework.http.HttpStatus;
4 import org.springframework.web.bind.annotation.ResponseStatus;
5
6 no usages
7 @ResponseStatus(value = HttpStatus.NOT_FOUND)
8 public class ResourceNotFoundException extends RuntimeException {
9     no usages
10     public ResourceNotFoundException(String message) {
11         super(message);
12     }
13 }
```

Dưới đây là cấu trúc thư mục của project sau khi đã tạo ra các class.



Tiếp theo, ta định nghĩa controller, trong đó có cung cấp method getAllSeafoods(), method này sẽ lấy ra toàn bộ dữ liệu trông bảng seafood trong database, rồi trả về dữ liệu thông qua api như sau.

```
SeafoodController.java x
1 package com.vcb.quanlyhaisan.controller;
2
3 import com.vcb.quanlyhaisan.model.Seafood;
4 import com.vcb.quanlyhaisan.repository.SeafoodRepository;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.web.bind.annotation.RequestMapping;
7 import org.springframework.web.bind.annotation.RestController;
8 import java.util.List;
9
10 //controller cung cap api cho phép thao tác với bảng seafood
11 no usages
12 @RestController
13 public class SeafoodController {
14     1 usage
15     @Autowired
16     private SeafoodRepository seafoodRepository;
17
18     //method trả về tất cả nhân viên (seafood)
19     no usages
20     @RequestMapping("/api/seafoods")
21     public List<Seafood> getAllSeafoods() { return seafoodRepository.findAll(); }
```

Bước tiếp theo, ta mở file chứa method main, rồi viết thêm mã để khai báo đối tượng SeafoodRepository như hình bên dưới.

```
QuanlyhaisanApplication.java x
1 package com.vcb.quanlyhaisan;
2
3 import com.vcb.quanlyhaisan.model.Seafood;
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.boot.CommandLineRunner;
6 import org.springframework.boot.SpringApplication;
7 import org.springframework.boot.autoconfigure.SpringBootApplication;
8
9 @SpringBootApplication
10 public class QuanlyhaisanApplication implements CommandLineRunner {
11
12     3 usages
13     @Autowired
14     private com.vcb.quanlyhaisan.repository.SeafoodRepository seafoodRepository;
15
16     public static void main(String[] args) { SpringApplication.run(QuanlyhaisanApplication.class, args); }
```

Ta định nghĩa tiếp method run(). Method này mỗi khi được thực thi sẽ thêm mới một số bản ghi vào trong bảng, để bảng có sẵn dữ liệu nhằm phục vụ cho thao tác GET dữ liệu về từ server.

```
QuanlyhaisanApplication.java x
18
19     @Override
20     public void run(String ... args) throws Exception {
21         Seafood obj1 = new Seafood();
22         obj1.setName("Tieu Vy");
23         obj1.setPrice(4500);
24         obj1.setAddress("Settle");
25         seafoodRepository.save(obj1);
26
27         Seafood obj2 = new Seafood();
28         obj2.setName("Mai Ngoc");
29         obj2.setPrice(4000);
30         obj2.setAddress("Hawaii");
31         seafoodRepository.save(obj2);
32
33         Seafood obj3 = new Seafood();
34         obj3.setName("Phuong Oanh");
35         obj3.setPrice(3500);
36         obj3.setAddress("Paris");
37         seafoodRepository.save(obj3);
38     }
39 }
```

Bước 3: Thực thi ứng dụng và xem kết quả

Sau khi đã viết mã xong, hãy thực thi ứng dụng và xem kết quả. Mỗi khi thực thi ứng dụng thì method main() sẽ gọi method run() và sẽ insert 3 bản ghi vào bảng seafood.

Để kiểm tra dữ liệu với Postman, ta mở Postman ra, rồi gõ url đã được khai báo ở trong file controller ở trên. Ta sẽ thấy api trả về danh sách nhân viên chính là dữ liệu trong bảng seafood đúng như mong muốn.

