

ĐINH HOÀNG GIA

# HƯỚNG DẪN



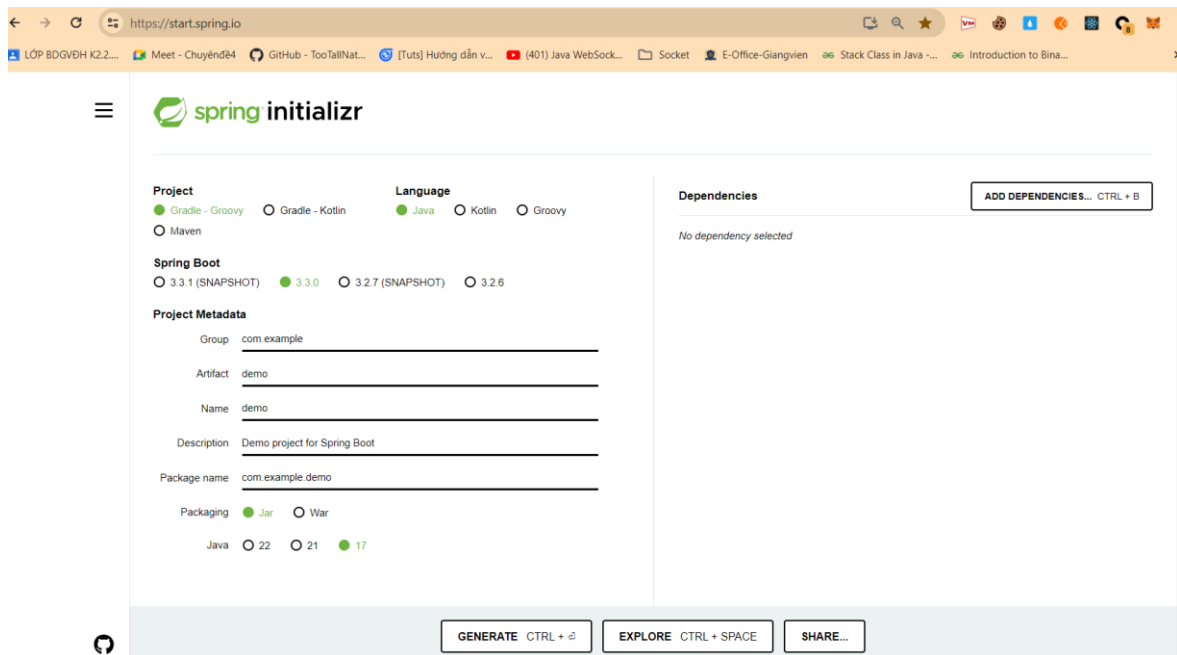
## MỤC LỤC

PHẦN 1. TẠO PROJECT SPRING BOOT.....	2
PHẦN 2. LẬP TRÌNH SPRING BOOT TRÊN ECLIPSE.....	8
2.1. Tạo controller .....	8
2.2. Tạo model.....	11
2.3. Tạo interface MyDB khai báo sử dụng JpaRepository framework để truy vấn.....	18
2.4. Sử dụng câu truy vấn JpaRepository.....	19
2.5. Chạy thử dự án .....	22
2.6. Cấu hình swagger để testing backend.....	22
2.7. Viết hàm khởi tạo dữ liệu nhiều bản ghi phục vụ cho việc kiểm thử.	27
2.8. Kết nối cơ sở dữ liệu trong backend .....	30
PHẦN 3. KẾT NỐI VÀ SỬ DỤNG SQL SERVER TRONG SPRING BOOT .....	35

## PHẦN 1. TẠO PROJECT SPRING BOOT

Để tạo 1 project dạng spring boot (web cho Java) bằng một công cụ online tên là **Spring Initializr**

**Bước 1:** Vào trình duyệt gõ đường dẫn <https://start.spring.io/>



The screenshot shows the Spring Initializr web application interface. The browser's address bar displays <https://start.spring.io>. The page features a sidebar with the Spring logo and a hamburger menu icon. The main content area is divided into several sections:

- Project:** Includes radio buttons for **Gradle - Groovy** (selected), **Gradle - Kotlin**, and **Maven**.
- Language:** Includes radio buttons for **Java** (selected), **Kotlin**, and **Groovy**.
- Spring Boot:** Includes radio buttons for **3.3.1 (SNAPSHOT)**, **3.3.0** (selected), **3.2.7 (SNAPSHOT)**, and **3.2.6**.
- Project Metadata:** Contains input fields for **Group** (com.example), **Artifact** (demo), **Name** (demo), **Description** (Demo project for Spring Boot), and **Package name** (com.example.demo).
- Packaging:** Includes radio buttons for **Jar** (selected) and **War**.
- Java:** Includes radio buttons for **22**, **21**, and **17** (selected).
- Dependencies:** A section with the text "No dependency selected" and a button labeled **ADD DEPENDENCIES... CTRL + B**.

At the bottom of the interface, there are three buttons: **GENERATE CTRL + G**, **EXPLORE CTRL + SPACE**, and **SHARE...**.

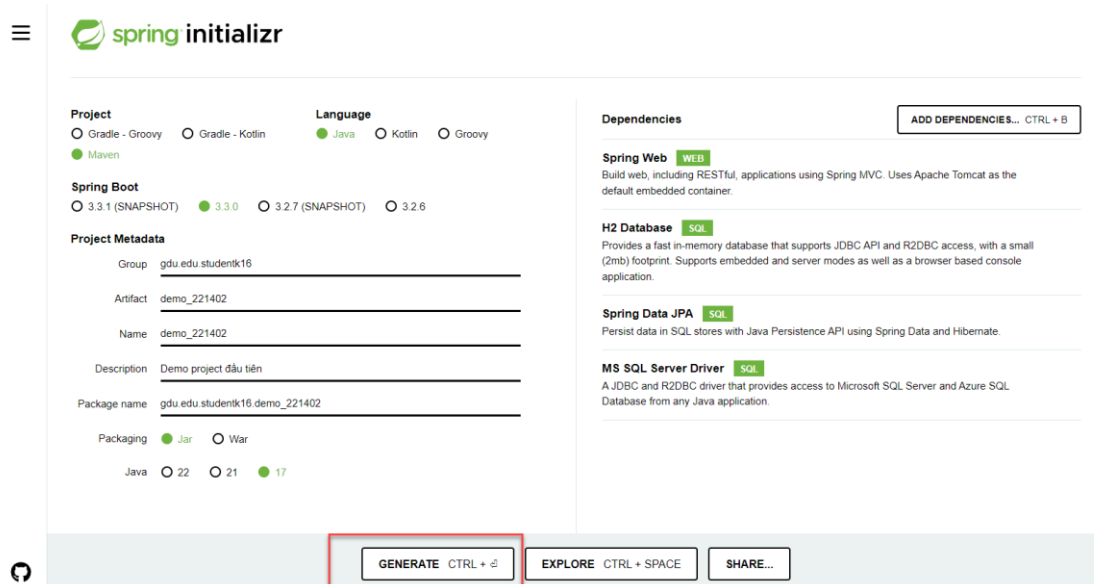
**Bước 2:** Sửa một số thông tin về project mà bạn cần tạo

The screenshot shows the Spring Initializr web interface. The 'Project' section has 'Maven' selected. The 'Language' section has 'Java' selected. The 'Spring Boot' section has '3.3.0' selected. The 'Project Metadata' section has 'Group' as 'gdu.edu.studentk16', 'Artifact' as 'demo\_221402', 'Name' as 'demo\_221402', 'Description' as 'Demo project đầu tiên', and 'Package name' as 'gdu.edu.studentk16.demo\_221402'. The 'Packaging' section has 'Jar' selected. The 'Java' section has '17' selected. The 'Dependencies' section is empty. Handwritten annotations in red include: 'Mẫu project là Maven' pointing to the Maven selection; 'Tên project định tạo' pointing to the Artifact field; 'Parkge cho project' (likely 'Package') pointing to the Jar selection; and 'Dùng phiên bản jdk-17 trở lên tùy bản java cài trên máy bạn' pointing to the Java version selection.

### Bước 3: Chọn một số thư viện

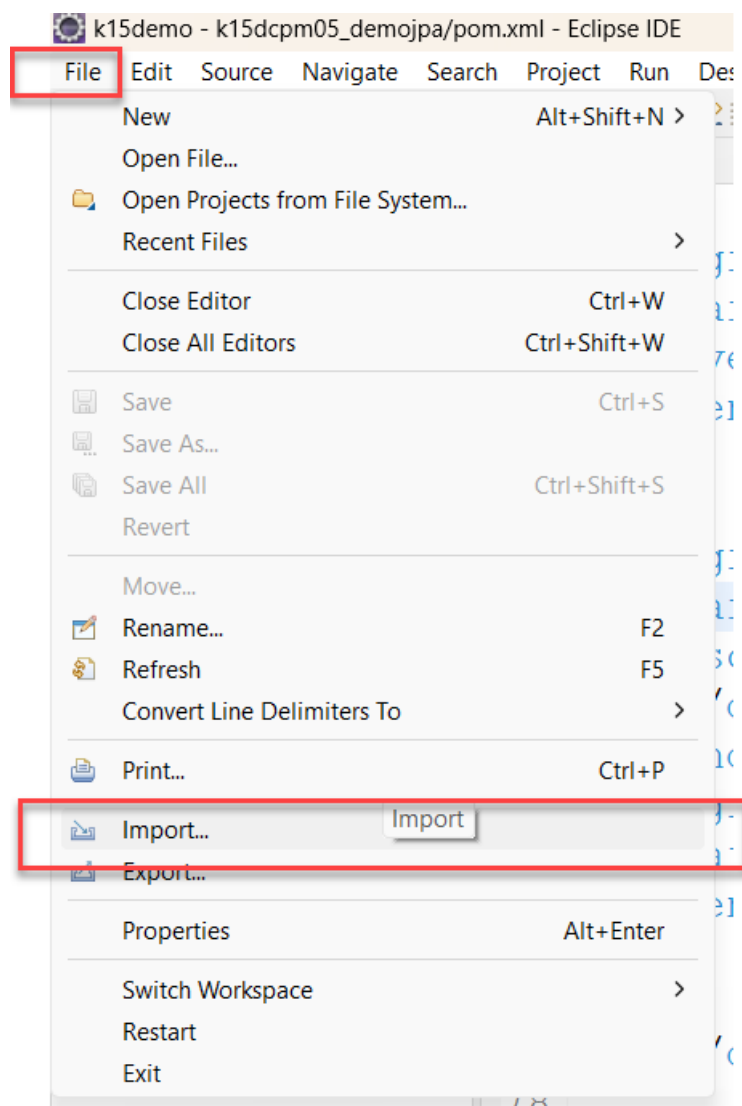
The screenshot shows the Spring Initializr web interface with dependencies selected. The 'Dependencies' section has 'Spring Web', 'H2 Database', 'Spring Data JPA', and 'MS SQL Server Driver' selected. Handwritten annotations in red include: 'Môi trường web' pointing to Spring Web; 'Để làm memory database' pointing to H2 Database; 'Sử dụng jpa (ảnh xạ object-entity trong CSDL)' pointing to Spring Data JPA; and 'Tùy chọn cái này với CSDL bạn muốn (trong trường hợp này là SQL Server. Bạn có thể chọn MySQL)' pointing to MS SQL Server Driver.

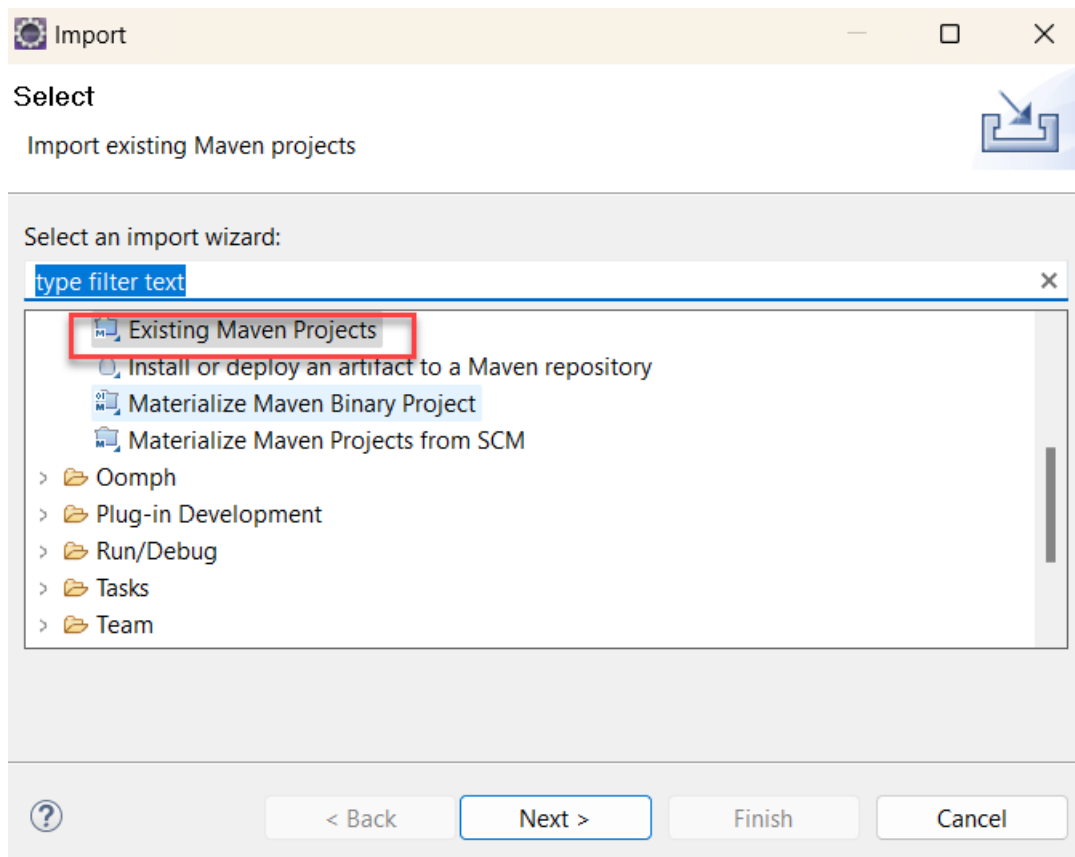
### Bước 4: Nhấn Generate để tạo



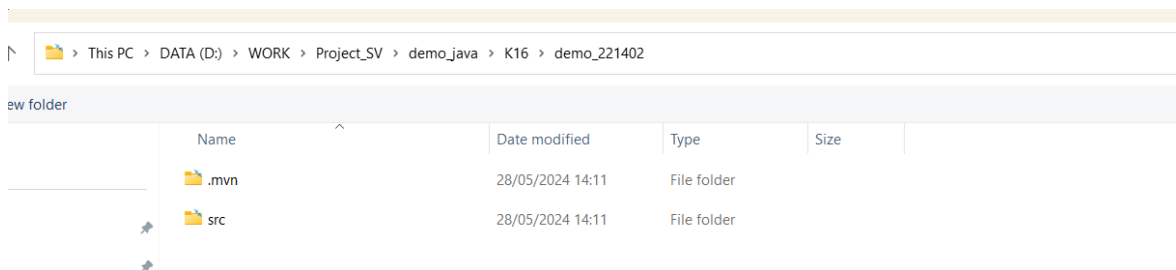
Bước 5: Download file project từ web về

Bước 6: Giải nén mở trong Eclipse như sau:

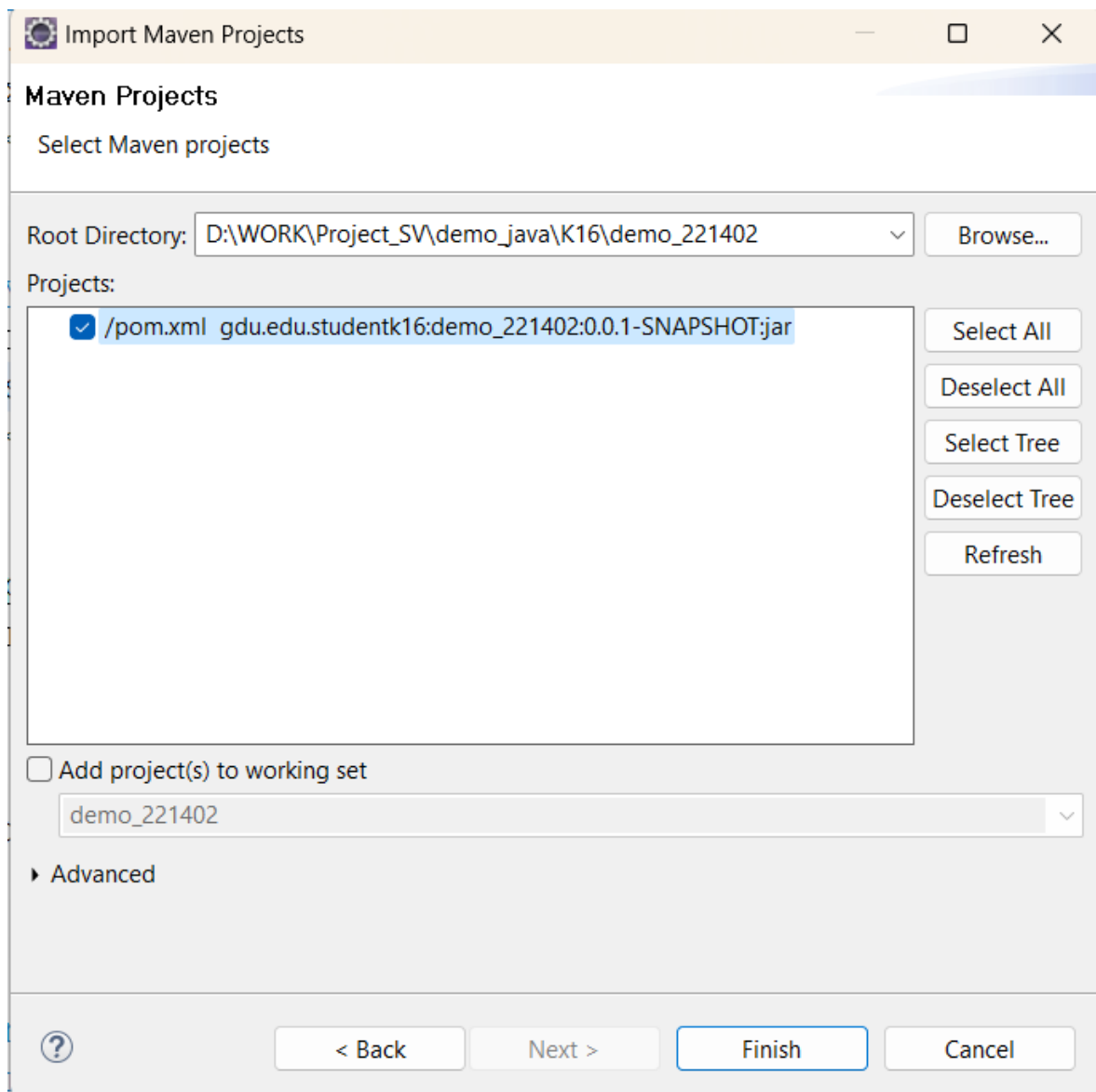




Browser vào thư mục, lúc nào thấy 2 thư mục như thế này là được



Nhấn button Select Folder



Sau đó ta mở project sẽ có dạng:

- ❏ pom.xml
- ▼ demo\_221402 [boot]
  - > ❏ src/main/resources
  - > ❏ JRE System Library [jdk-23]
  - > ❏ Maven Dependencies
  - > ❏ src/main/java
  - ▼ ❏ src/test/java
    - ▼ ❏ gdu.edu.studentk16.demo\_221402
      - > ❏ Demo221402ApplicationTests.java
  - > ❏ src
  - > ❏ target
  - ❏ demo\_221402.rar
  - ❏ HELP.md
  - ❏ mvnw
  - ❏ mvnw.cmd
  - ❏ pom.xml

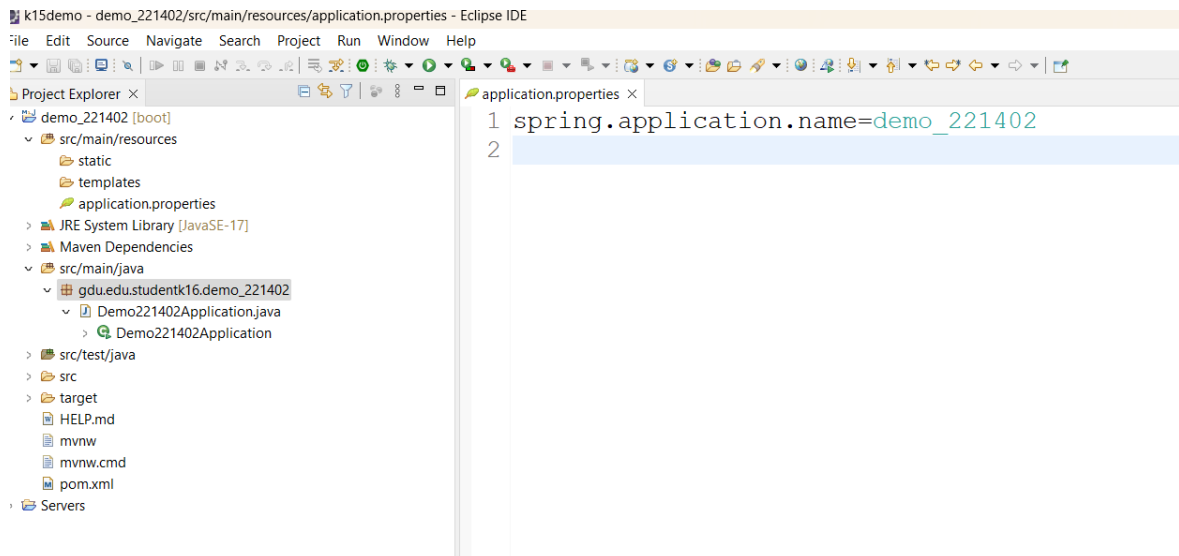


## PHẦN 2. LẬP TRÌNH SPRING BOOT TRÊN ECLIPSE

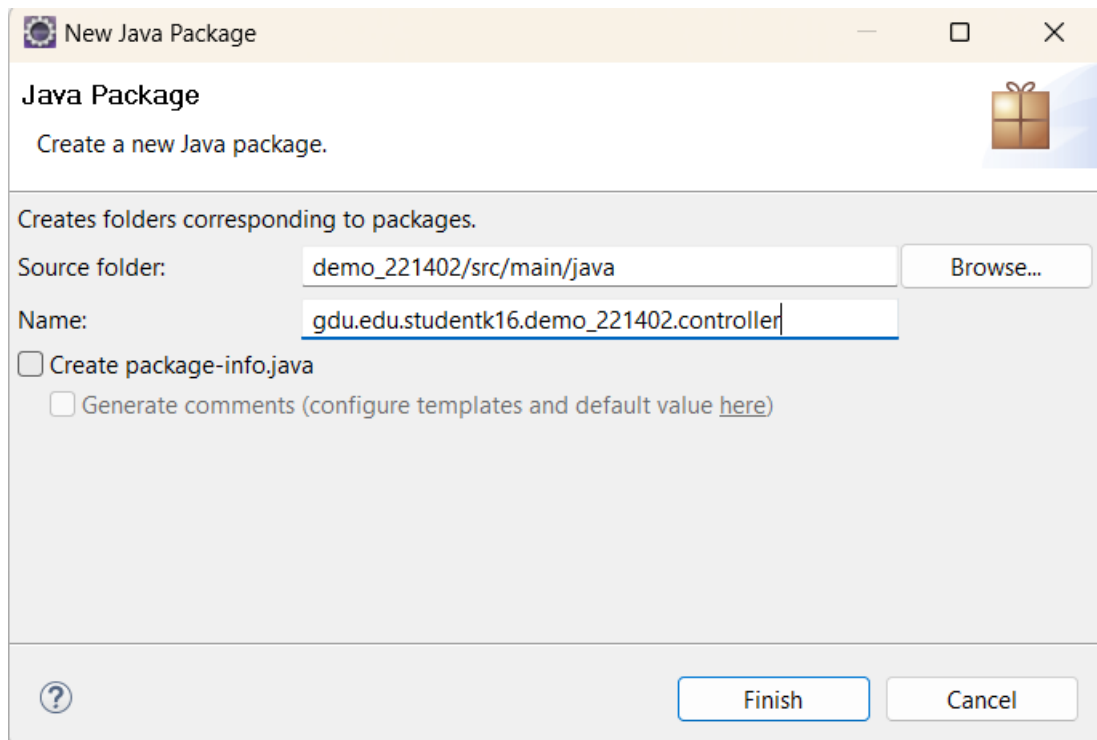
### 2.1. Tạo controller

Controller sẽ giúp cho ứng dụng hiểu đây là backend (ResfulAPI), là dạng dịch vụ dữ liệu, chứ không phải ứng dụng web

Ta tiến hành vào project



Vào gói package mặc định ta tạo gói controller (nằm trong gói mặc định nhé), riêng chữ “controller” các bạn phải ghi thật đúng nhé, ko sẽ ko chạy được web



Tạo tiếp 1 class tên là StudentController nằm trong thư mục controller vừa tạo

**New Java Class**

Create a new Java class.

Source folder: demo\_221402/src/main/java Browse...

Package: gdu.edu.studentk16.demo\_221402.controller Browse...

☐ Enclosing type: Browse...

Name: StudentController

Modifiers: ☒ public ☐ package ☐ private ☐ protected  
☐ abstract ☐ final ☐ static  
☒ none ☐ sealed ☐ non-sealed ☐ final

Superclass: java.lang.Object Browse...

Interfaces: Add...  
Remove

Which method stubs would you like to create?

☐ public static void main(String[] args)  
☐ Constructors from superclass  
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))  
☐ Generate comments

? Finish Cancel

Viết class như sau:

**package**

```
gdu.edu.studentk16.demo_221402.controller;
```

**import**

```
org.springframework.web.bind.annotation.GetMapping;
```

```
import
org.springframework.web.bind.annotation.RestController;

@RestController
public class StudentController {

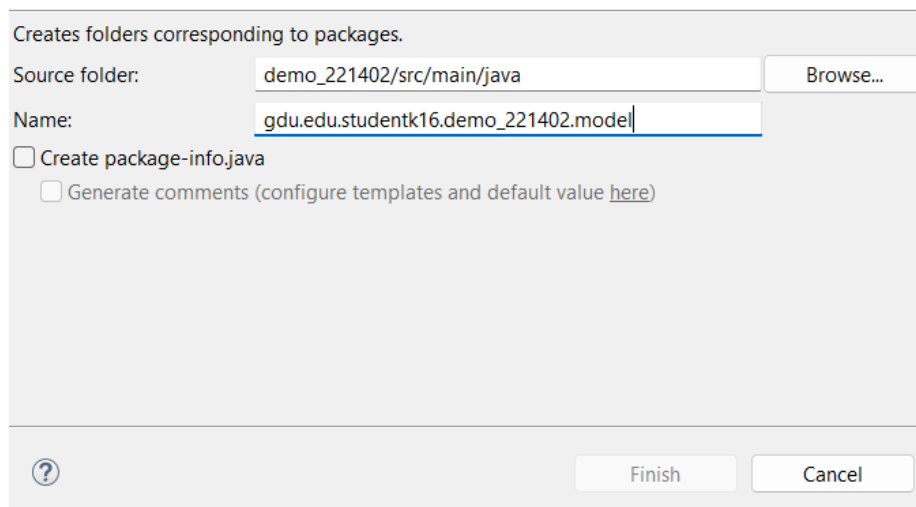
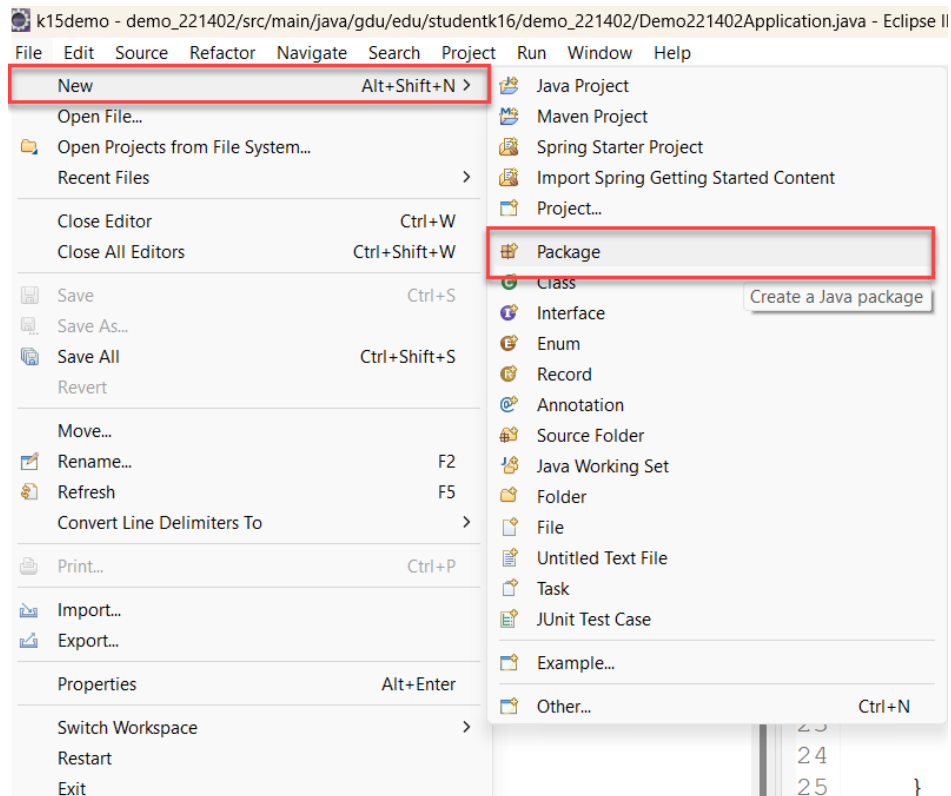
    @GetMapping("/liststudents")
    public String listStudents()
    {
        return "Xin chào các bạn 221402";
    }

}
```

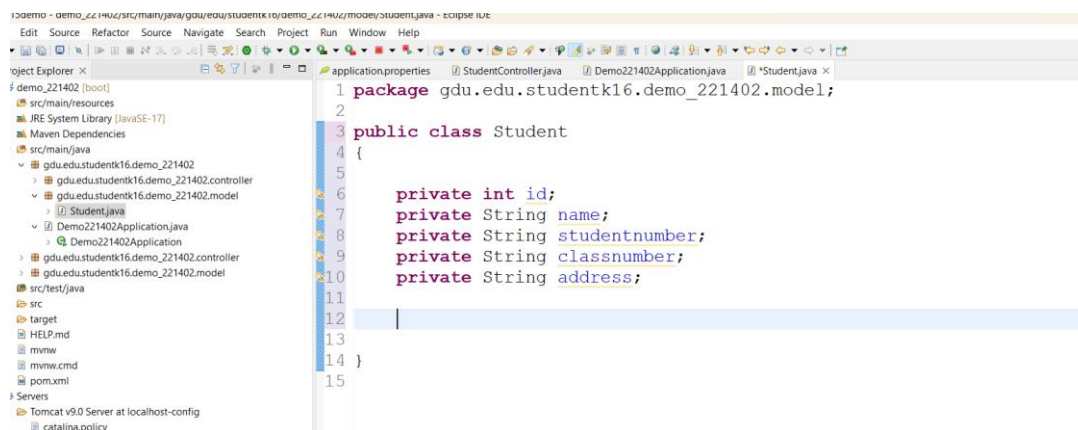
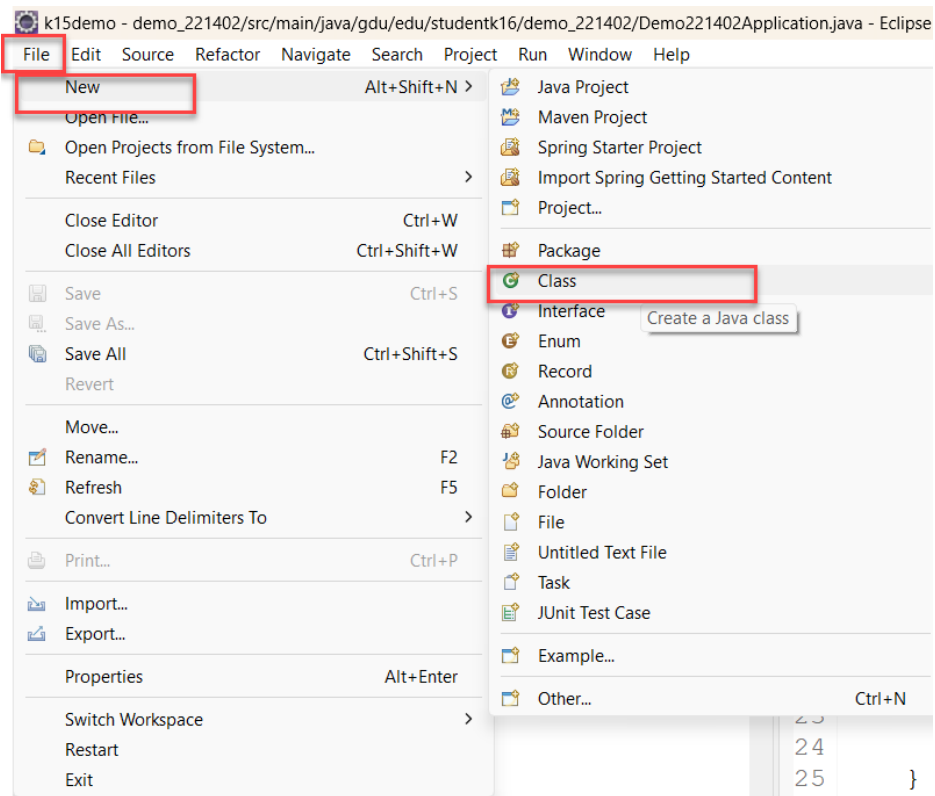
## 2.2. Tạo model

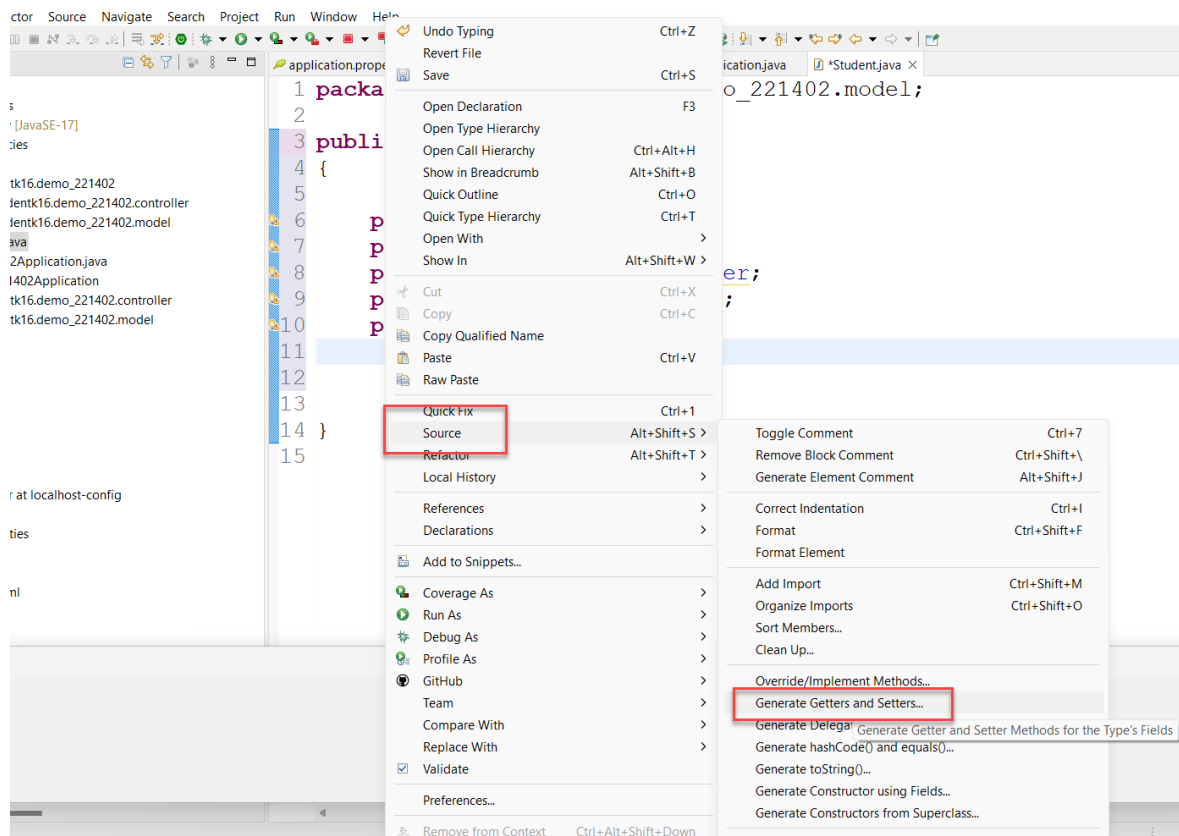
Model là khuôn mẫu dữ liệu để sử dụng cho ứng dụng, trong ví dụ này ta sử dụng mẫu dữ liệu là Student với các thông tin là id, name, studentnumber, address.

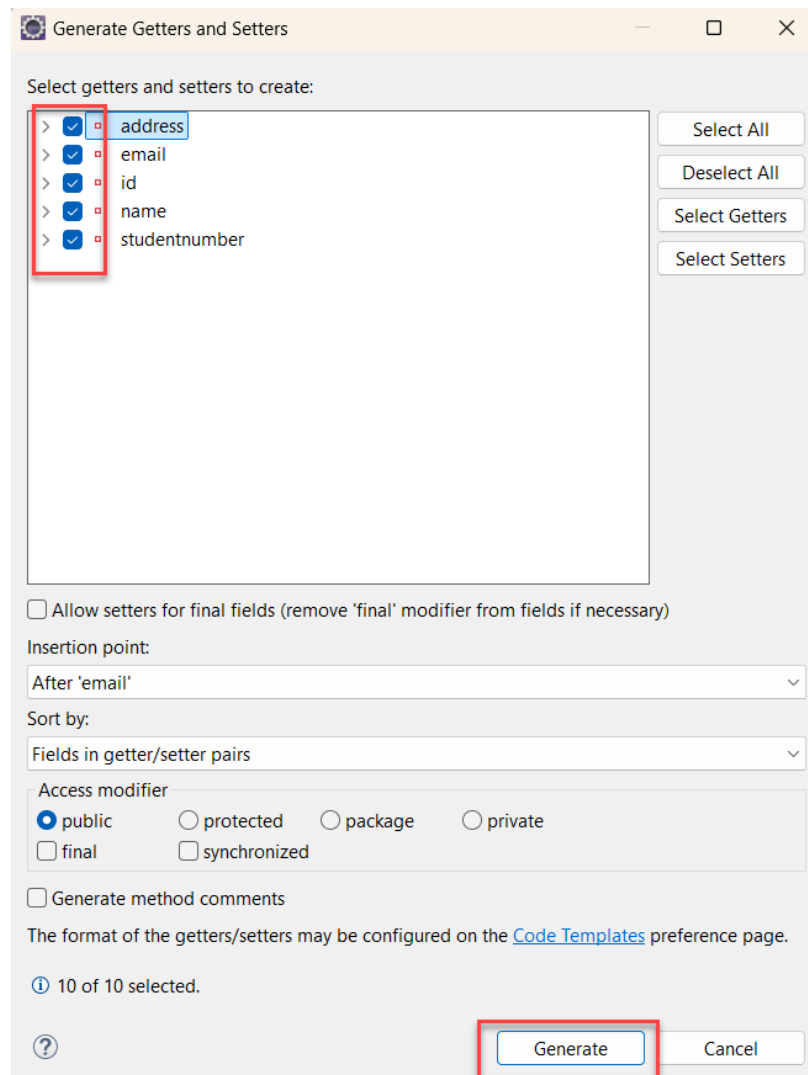
Trước hết ta tạo package model như sau:



Tiếp theo, ta tạo class Student trong package model vừa tạo:







**package**

```
gdu.edu.studentk16.demo_221402.controller;
```

```
import jakarta.persistence.Column;
```

```
import jakarta.persistence.Entity;
```

```
import jakarta.persistence.GeneratedValue;
```

```
import jakarta.persistence.GenerationType;
```

```
import jakarta.persistence.Id;
```

```
@Entity
```

```
public class Student
```



```

{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    @Column(columnDefinition = "nvarchar(255)")
    private String name;
    private String studentnumber;
        @Column(columnDefinition = "nvarchar(255)")
    private String address;
    private String email;
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getStudentnumber() {
        return studentnumber;
    }
    public void setStudentnumber(String studentnumber) {
        this.studentnumber = studentnumber;
    }
}

```

```

    }

    public String getAddress() {
        return address;
    }

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

    public String getEmail() {
        return email;
    }

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

    public Student(int id, String name, String studentnumber, String
address, String email) {
        super();
        this.id = id;
        this.name = name;
        this.studentnumber = studentnumber;
        this.address = address;
        this.email = email;
    }

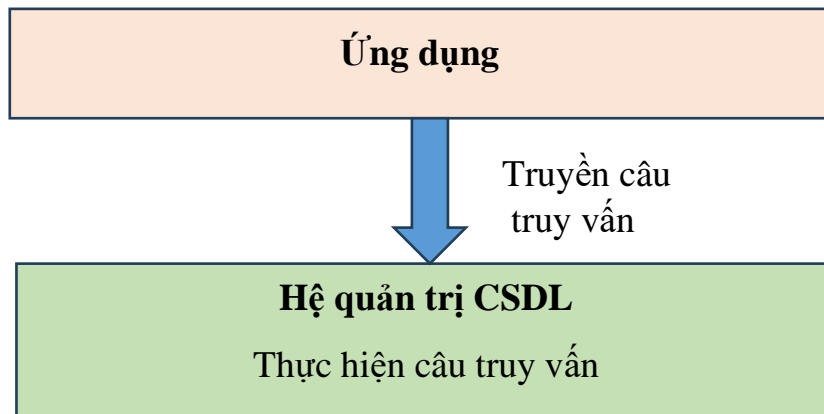
    public Student() {
        super();
        // TODO Auto-generated constructor stub
    }

}

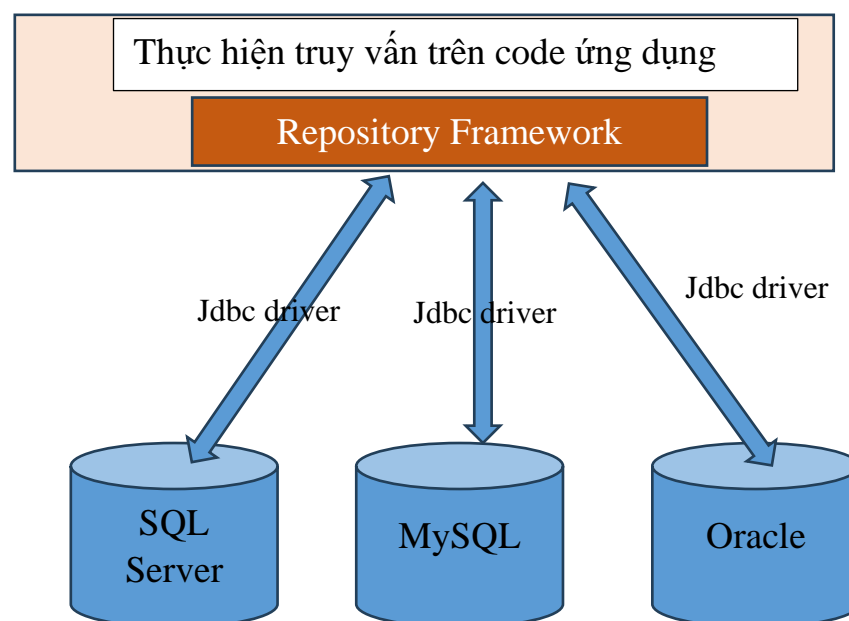
```

### 2.3. Tạo interface MyDB khai báo sử dụng JpaRepository framework để truy vấn

Theo thông thường chúng ta làm việc với cơ sở dữ liệu sẽ truyền câu truy vấn từ ứng dụng đến Hệ quản trị CSDL theo mô hình như sau:



Phương pháp này có 1 khó khăn là với mỗi hệ quản trị CSDL khác nhau thì sẽ có bộ truy vấn khác nhau, dẫn đến ta phải sửa đổi chương trình khi chuyển đổi CSDL. Vì vậy trong các mô hình lập trình cải tiến, người ta sử dụng Repository thực hiện câu truy vấn trên code



Ta sẽ tạo interface MyDB thừa kế từ JpaRepository

```

package gdu.__java.API_demo.model;

import org.springframework.data.jpa.repository.JpaRepository;

public interface MyDB extends JpaRepository<Student, Integer>
{
}

```

Tên class model
Kiểu dữ liệu trường khóa

```

package gdu.__java.API_demo.model;

import org.springframework.data.jpa.repository.JpaRepository;

public interface MyDB extends JpaRepository<Student, Integer>
{
}

```

## 2.4. Sử dụng câu truy vấn JpaRepository

Sau khi đã khai báo jpa MyDB ta sẽ sử dụng trong class StudentController.java như sau:

```

18 @RestController
19 public class StudentController {
20
21
22     @Autowired
23     MyDB mydb;
24
25     @GetMapping("/liststudents")
26     public List<Student> ListStudents()
27     {
28         return mydb.findAll();
29     }
30
31
32     @PostMapping("/insertStudent")
33     public String insertStudent(@RequestBody Student newstudent)
34     {
35         mydb.save(newstudent);
36         return "Thêm sinh viên thành công!";
37     }
38
39
40     @PutMapping("/updateStudent")
41     public String updateStudent(@RequestBody Student student)
42     {
43         mydb.save(student);
44         return "Sửa sinh viên thành công!";
45     }
46
47     @DeleteMapping("/deleteStudent/{id}")
48     public String deleteStudent(@PathVariable Integer id)

```

Khai báo đối tượng. Đối tượng này là bean sẽ được khởi tạo tự động

Câu truy vấn dành cho select

Câu truy vấn dành cho insert

Câu truy vấn dành cho update

```

package gdu.edu.studentk16.demo_221402.controller;

```

```

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import gdu.__java.API_demo.model.MyDB;
import gdu.__java.API_demo.model.Student;

@RestController
public class StudentController {

    @Autowired
    MyDB mydb;

    @GetMapping("/liststudents")
    public List<Student> ListStudents()
    {
        return mydb.findAll();
    }
}

```

```

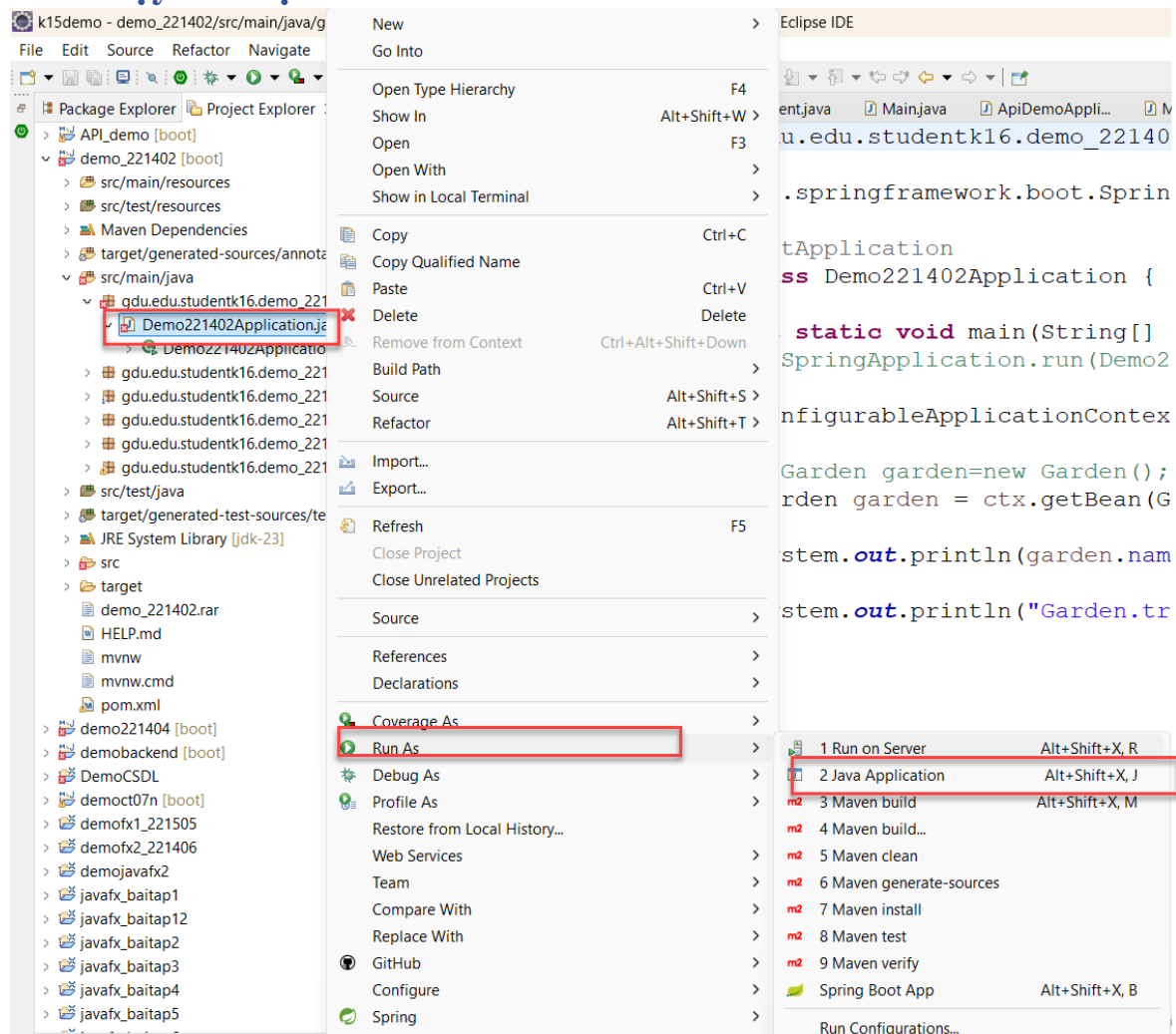
    @PostMapping("/insertStudent")
    public String insertStudent(@RequestBody Student newstudent)
    {
        mydb.save(newstudent);
        return "Thêm sinh viên thành công!";
    }

    @PutMapping("/updateStudent")
    public String updateStudent(@RequestBody Student student)
    {
        mydb.save(student);
        return "Sửa sinh viên thành công!";
    }

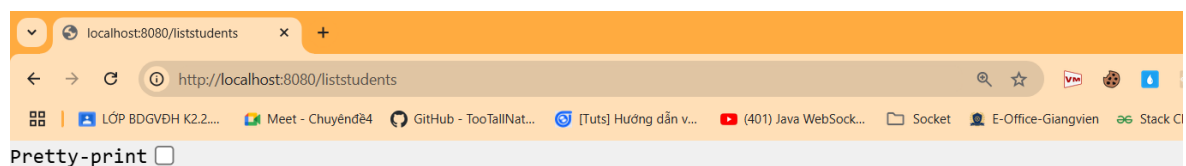
    @DeleteMapping("/deleteStudent/{id}")
    public String deleteStudent(@PathVariable Integer id)
    {
        mydb.deleteById(id);
        return "Xóa thành công!";
    }
}

```

## 2.5. Chạy thử dự án



Mở trình duyệt và gõ: <http://localhost:8080/liststudents>. Giao diện như sau là thành công vì hiện tại chưa có dữ liệu



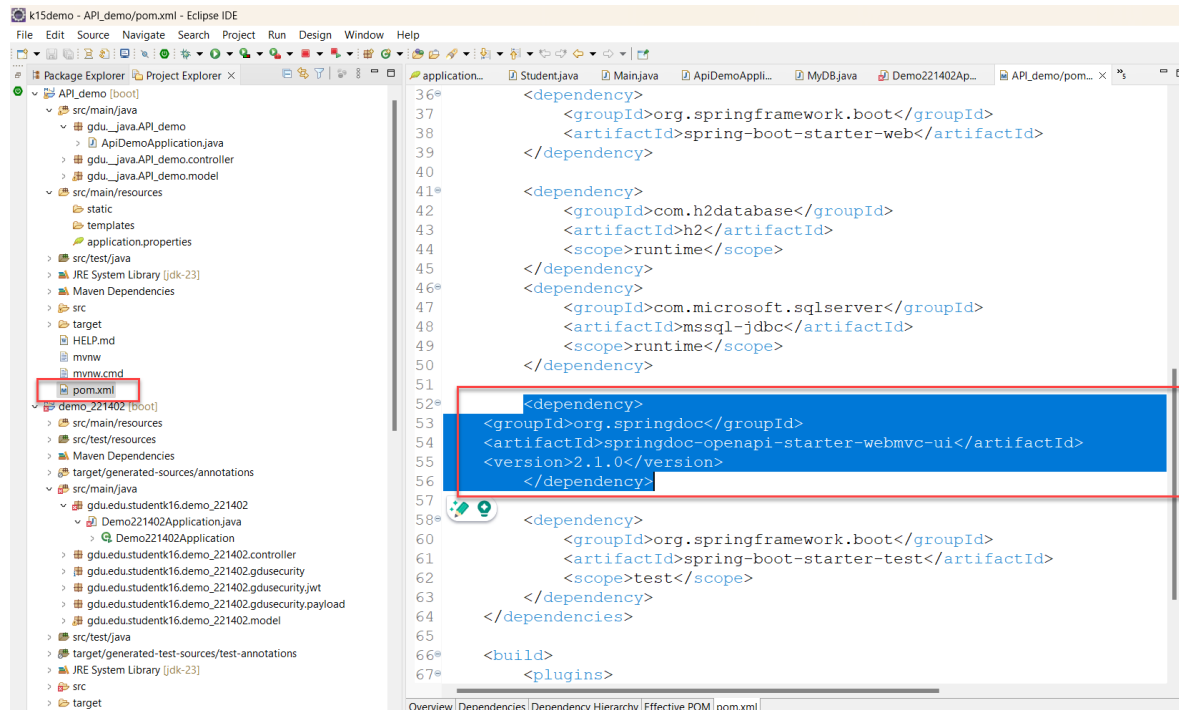
[ ]

## 2.6. Cấu hình swagger để testing backend

Ở phần trên dù ta mở trình duyệt và gõ <http://localhost:8080/liststudents> để hiển thị dữ liệu. Tuy nhiên, để thử những chức năng như thêm, sửa dữ liệu ta

cần phải có giao diện để gửi dữ liệu lên backend. Do đó ta cần phải cấu hình swagger.

Ta vào file pom.xml, thêm đoạn dependency như sau:



<dependency>

<groupId>org.springdoc</groupId>

<artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>

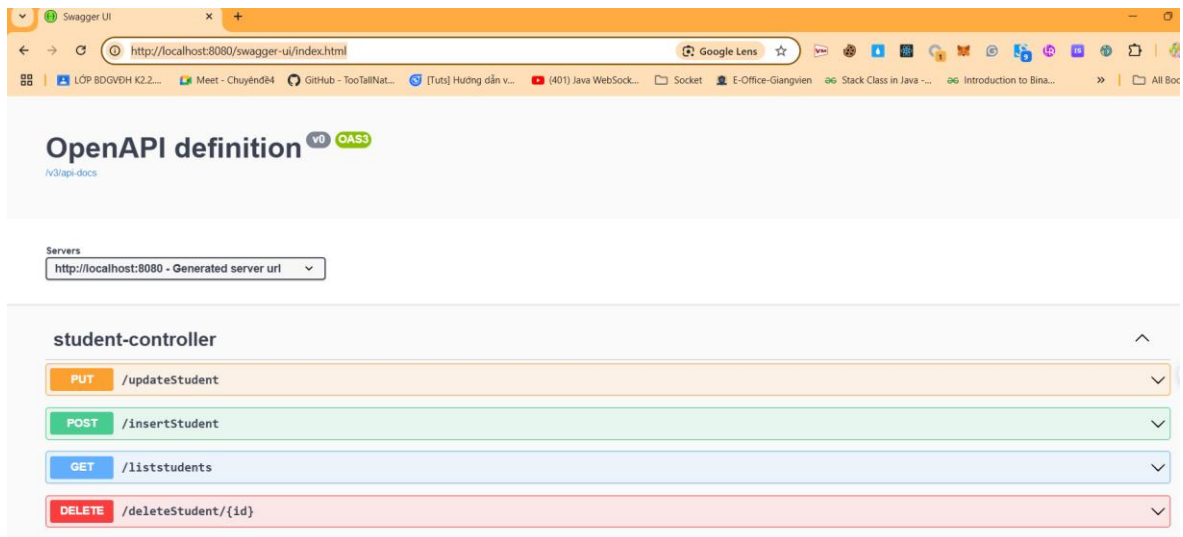
<version>2.1.0</version>

</dependency>

Bây giờ ta chạy lại ứng dụng.

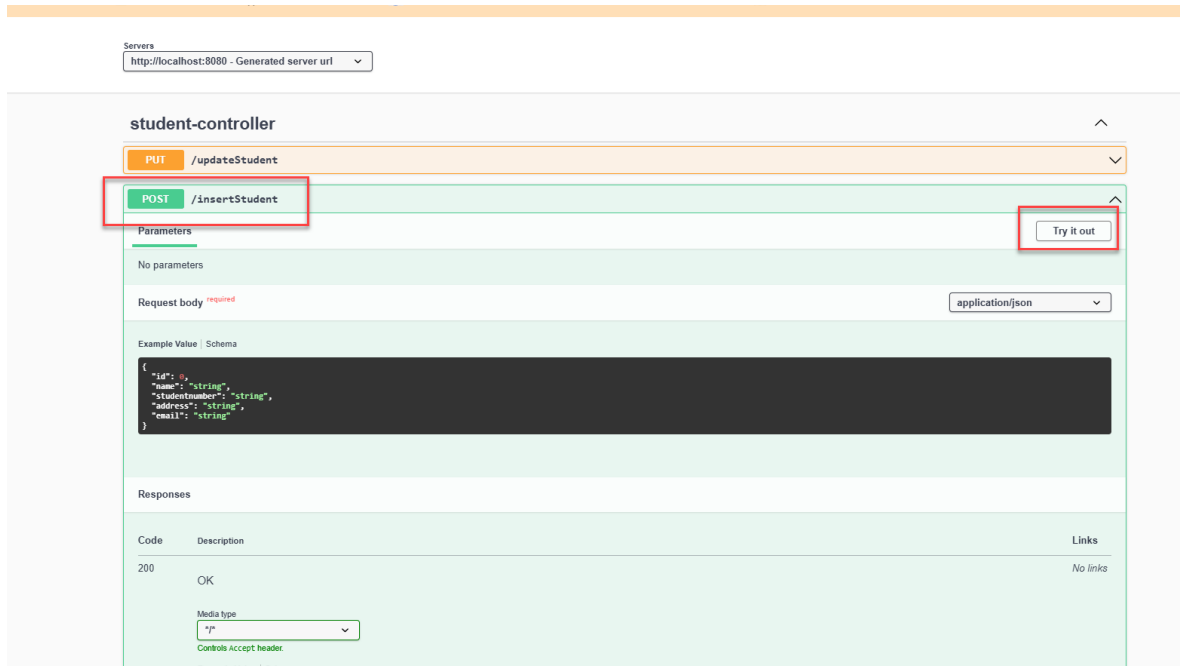
Vào trình duyệt gõ url: <http://localhost:8080/swagger-ui/index.html>, ta có giao diện sau:





Dựa vào giao diện swagger, ta sẽ test các chức năng như sau:

## + Thêm sản phẩm



**POST** /insertStudent

Parameters

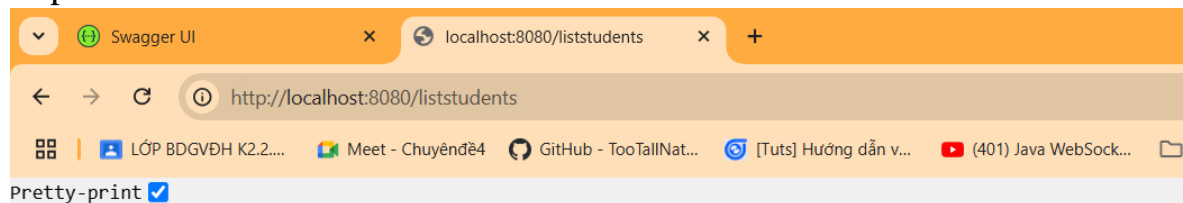
No parameters

Request body required application/json

```
{
  "id": 0,
  "name": "Nguyễn Văn A",
  "studentnumber": "123456",
  "address": "371 Nguyễn Kiệm, Gò Vấp, TP. Hồ Chí Minh",
  "email": "abc@gmail.com"
}
```

Execute Clear

Sau khi nhập dữ liệu, ta mở 1 tab trên trình duyệt và gõ url:  
http://localhost:8080/liststudents



```
[
  {
    "id": 1,
    "name": "Nguyễn Văn A",
    "studentnumber": "123456",
    "address": "371 Nguyễn Kiệm, Gò Vấp, TP. Hồ Chí Minh",
    "email": "abc@gmail.com"
  },
  {
    "id": 2,
    "name": "Nguyễn Văn B",
    "studentnumber": "123456",
    "address": "371 Nguyễn Kiệm, Gò Vấp, TP. Hồ Chí Minh",
    "email": "abc@gmail.com"
  }
]
```

## + Sửa sản phẩm

**student-controller**

**PUT** /updateStudent Try it out

Parameters

No parameters

Request body required application/json

Example Value | Schema

```
{
  "id": 0,
  "name": "string",
  "studentnumber": "string",
  "address": "string",
  "email": "string"
}
```

student-controller

PUT /updateStudent

Parameters

No parameters

Request body required

application/json

```
{
  "id": 2,
  "name": "Nguyễn Văn Cường",
  "studentnumber": "221504",
  "address": "371 Nguyễn Kiệm",
  "email": "nguyenvancuong@gmail.com"
}
```

Execute

Vào trình duyệt gõ url: <http://localhost:8080/liststudents>

← → ↻ ⓘ <http://localhost:8080/liststudents>

🗑️ | 👤 LỚP BDGVĐH K2.2.... 📺 Meet - Chuyên đề 4 🔄 GitHub - Too Tall Nat... 🔄 [Tuts] Hư

Pretty-print ☒

```
[
  {
    "id": 1,
    "name": "Nguyễn Văn A",
    "studentnumber": "123456",
    "address": "371 Nguyễn Kiệm, Gò Vấp, TP. Hồ Chí Minh",
    "email": "abc@gmail.com"
  },
  {
    "id": 2,
    "name": "Nguyễn Văn Cường",
    "studentnumber": "221504",
    "address": "371 Nguyễn Kiệm",
    "email": "nguyenvancuong@gmail.com"
  }
]
```

+ Xóa sản phẩm

DELETE /deleteStudent/{id}

Parameters

Try it out

Name	Description
id <small>required</small>	id
integer(\$int32)	(path)

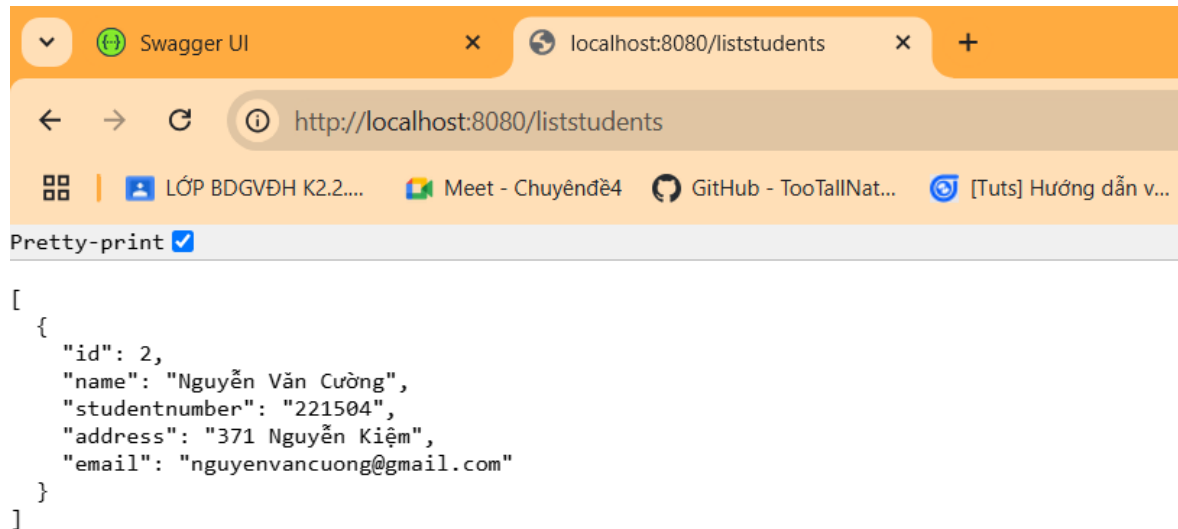
DELETE /deleteStudent/{id}

Parameters

Name	Description
id * required integer(\$int32) (path)	<input type="text" value="1"/>

Execute

Vào trình duyệt gõ địa chỉ: <http://localhost:8080/liststudents> xem kết quả, ta thấy đã mất bản ghi có id=1



## 2.7. Viết hàm khởi tạo dữ liệu nhiều bản ghi phục vụ cho việc kiểm thử

Với một chương trình, việc khởi tạo dữ liệu sử dụng để thử tải chương trình là yếu tố quan trọng, thay vì 1-2 bản ghi, người ta có thể tạo ra 1000, 10000 hay 1000.000 bản ghi để thử hiệu năng và độ chịu tải.

Ta sẽ viết class `KhoitaoCSDL` implements từ `ApplicationRunner` như sau:

```
package gdu.edu.studentk16.demo_221402.model;
```

```
import java.util.Random;
```

```
import org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.boot.ApplicationArguments;
```

```
import org.springframework.boot.ApplicationRunner;
import org.springframework.context.annotation.Configuration;
```

```
@Configuration
```

```
public class KhoitaoCSDL implements ApplicationRunner {
```

```
    String[] ho= {"Nguyễn", "Lê", "Trần", "Hồ", "Lê", "Đinh", "Mạc",
    "Văn", "Ninh", "Dương"};
```

```
    String[] ten= {"Anh", "Ngọc An", "Bình", "Quỳnh", "Hoài Ân", "Văn
    Toàn", "Thắng", "Thịnh", "Trường", "Ngân"};
```

```
    String[] diachi={"371 Nguyễn Kiệm, P3, Gò Vấp","Lý Thường Kiệt, P.7,
    Quận 10", "Ninh Thuận", "Bình Thuận","Bình Chánh", "Quận 8","Quận 1,
    TP. Hồ Chí Minh", "Quận 7, TP.Hồ Chí Minh", "Đức Hòa, Long An","Gò
    Công Đông, Tiền Giang"};
```

```
    private MyDB mydb;
```

```
@Autowired
```

```
    public KhoitaoCSDL(MyDB mydb) {
        this.mydb = mydb;
    }
```

```
    private static int random1()
    {
```

```
        Random rand = new Random();
```

```

    int randomNum = rand.nextInt((9) + 1);
    return randomNum;
}

@Override
public void run(ApplicationArguments args) throws Exception {
    // TODO Auto-generated method stub

    for (int i=0; i<1000; i++)
    {
        mydb.save(new Student(0, ho[randomI()]+
"+ten[randomI()], "221521"+(i+1), diachi[randomI()], "abc@gdu.edu.vn" ));
    }

}
}

```

Trong ví dụ này, annotation `@Configuration` để ứng dụng hiểu đây là class môi trường và sẽ được tự động chạy khi ứng dụng được khởi chạy.

Annotation `@Autowired` được ứng dụng sử dụng để khởi tạo bean (một loại đối tượng được tự động khởi tạo và sử dụng chung). Trong ví dụ này ứng dụng sẽ khởi tạo bean mydb.

Chạy lại ứng dụng và vào trình duyệt gõ url: [localhost:8080/liststudents](http://localhost:8080/liststudents)

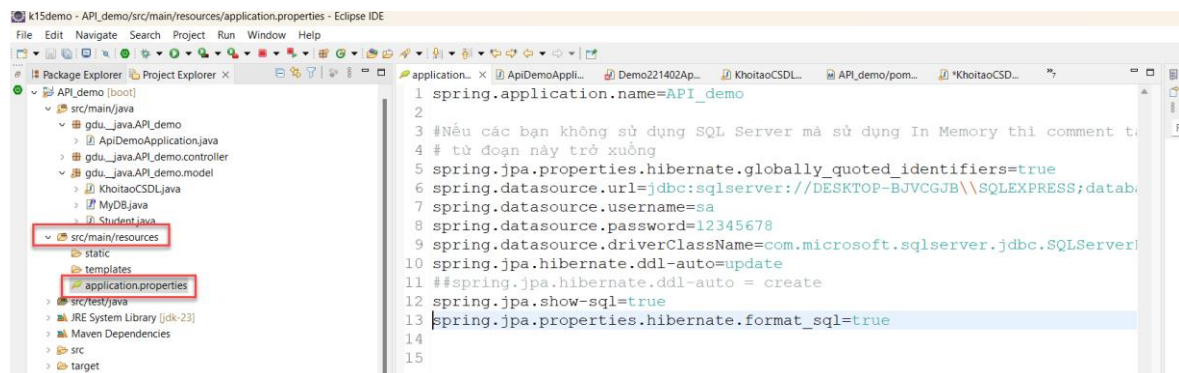
```
http://localhost:8080/liststudents

pretty-print
{
  "id": 997,
  "name": "Mạc Ngọc An",
  "studentnumber": "221521997",
  "address": "Đức Hòa, Long An",
  "email": "abc@gdu.edu.vn"
},
{
  "id": 998,
  "name": "Dương Bình",
  "studentnumber": "221521998",
  "address": "Quận 7, TP.Hồ Chí Minh",
  "email": "abc@gdu.edu.vn"
},
{
  "id": 999,
  "name": "Nguyễn Bình",
  "studentnumber": "221521999",
  "address": "Quận 7, TP.Hồ Chí Minh",
  "email": "abc@gdu.edu.vn"
},
{
  "id": 1000,
  "name": "Ninh Ngọc An",
  "studentnumber": "2215211000",
  "address": "Ninh Thuận",
  "email": "abc@gdu.edu.vn"
}
```

## 2.8. Kết nối cơ sở dữ liệu trong backend

Theo mặc định, CSDL H2 sẽ được ứng dụng sử dụng. Tuy nhiên, ta có thể cấu hình để chạy các CSDL khác như SQL Server, MySQL,... như sau (Trong ví dụ này, sử dụng cơ sở dữ liệu SQL Server).

Vào thư mục src/main/resources, file application.properties:



```
1 spring.application.name=API_demo
2
3 #Nếu các bạn không sử dụng SQL Server mà sử dụng In Memory thì comment t
4 # từ đoạn này trở xuống
5 spring.jpa.properties.hibernate.globally_quoted_identifiers=true
6 spring.datasource.url=jdbc:sqlserver://DESKTOP-BJVCGBJ\\SQLEXPRESS;datab
7 spring.datasource.username=sa
8 spring.datasource.password=12345678
9 spring.datasource.driverClassName=com.microsoft.sqlserver.jdbc.SQLServerI
10 spring.jpa.hibernate.ddl-auto=update
11 ##spring.jpa.hibernate.ddl-auto = create
12 spring.jpa.show-sql=true
13 spring.jpa.properties.hibernate.format_sql=true
14
15
```

Đoạn lệnh config CSDL:

spring.application.name=demo\_221402

#Nếu các bạn không sử dụng SQL Server mà sử dụng In Memory thì comment tất cả

# từ đoạn này trở xuống

```
spring.jpa.properties.hibernate.globally_quoted_identifiers=true
spring.datasource.url=jdbc:sqlserver://DESKTOP-
BJVCGJB\\SQLEXPRESS;databaseName=221521_demo;trustServerCertific
ate=true
spring.datasource.username=sa
spring.datasource.password=12345678
spring.datasource.driverClassName=com.microsoft.sqlserver.jdbc.SQLServer
Driver
spring.jpa.hibernate.ddl-auto=update
##spring.jpa.hibernate.ddl-auto = create
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
```

Vào Hệ quản trị CSDL, tạo CSDL **221521\_demo**, và chạy script tạo bảng như sau:

```
USE [221521_demo]
```

```
GO
```

```
/****** Object: Table [dbo].[student]   Script Date: 06/11/2024 16:30:30
*****/
```

```
IF EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[student]') AND type in (N'U'))
```

```
DROP TABLE [dbo].[student]
```

```
GO
```

```
/****** Object: Table [dbo].[student]   Script Date: 06/11/2024 16:30:30
*****/
```

```
SET ANSI_NULLS ON
```



GO

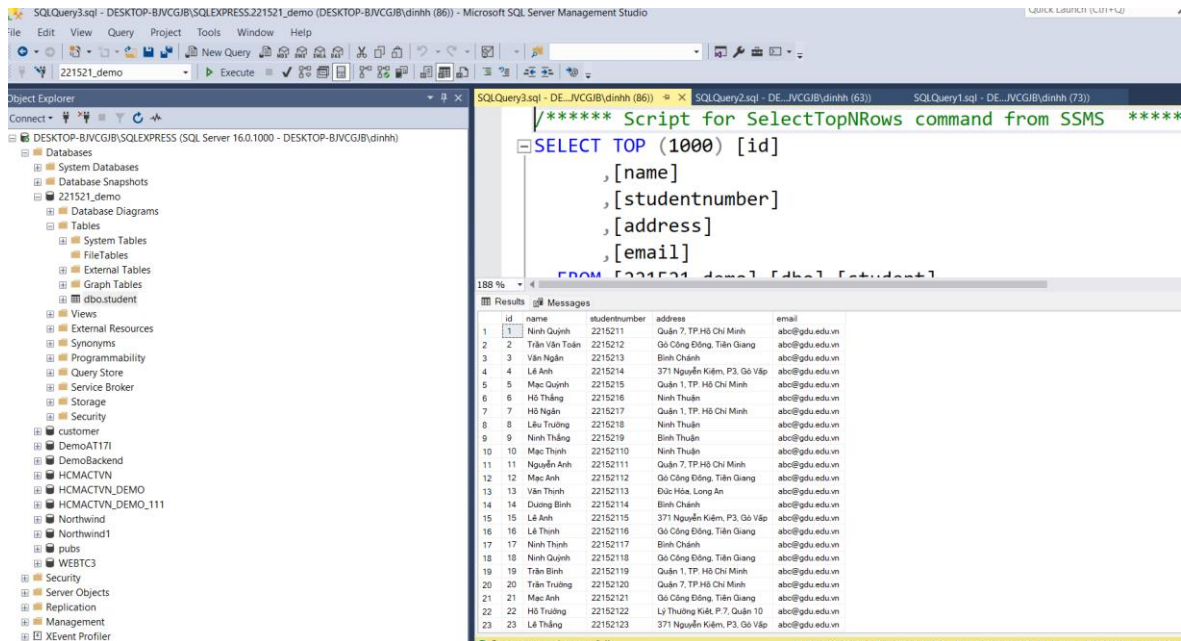
SET QUOTED\_IDENTIFIER ON

GO

```
CREATE TABLE [dbo].[student](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [name] [nvarchar](255) NULL,
    [studentnumber] [nvarchar](255) NULL,
    [address] [nvarchar](255) NULL,
    [email] [nvarchar](255) NULL,
    CONSTRAINT [PK_student] PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

-----

Chạy lại ứng dụng và kiểm tra bảng trong CSDL



Vào trình duyệt và gõ url: [localhost:8080/liststudents](http://localhost:8080/liststudents)



**Lưu ý:** Mỗi lần khởi chạy ứng dụng thì chương trình sẽ tự động sinh 1000 bản ghi với code này. Do đó nếu không muốn tạo thêm dữ liệu, ta vào class KhoitaoCSDL để comment annotation @Configuration

```
1 package gdu.__java.API_demo.model;
2
3 import java.util.Random;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.boot.ApplicationArguments;
7 import org.springframework.boot.ApplicationRunner;
8 import org.springframework.context.annotation.Configuration;
9
10
11
12 // @Configuration
13 public class KhoitaoCSDL implements ApplicationRunner {
14
15     String[] ho= {"Nguyễn", "Lê", "Trần", "Hồ", "Lê", "Đinh", "Mạc", "Vũ"};
16     String[] ten= {"Anh", "Ngọc An", "Bình", "Quỳnh", "Hoài Ân", "Văn Toà"};
17     String[] diachi= {"371 Nguyễn Kiệm, P3, Gò Vấp", "Lý Thường Kiệt, P.7,"};
18
19
20     private MyDB mydb;
21
22
23     @Autowired
24     public KhoitaoCSDL(MyDB mydb) {
25         this.mydb = mydb;
26     }
27
28     private static int random1()
29     {
30         Random rand = new Random();
31
32         int randomNum = rand.nextInt((9) + 1);
```

## PHẦN 3. KẾT NỐI VÀ SỬ DỤNG SQL SERVER TRONG SPRING BOOT

### Bước 1: Cấu hình dự án Spring Boot với SQL Server

Trước tiên, cần thiết lập kết nối đến cơ sở dữ liệu SQL Server trong tệp `application.properties` hoặc `application.yml`.

#### **application.properties**

properties

Sao chép mã

# Cấu hình cơ sở dữ liệu

```
spring.datasource.url=jdbc:sqlserver://localhost:1433;databaseName=Northwind
```

```
spring.datasource.username=your_username
```

```
spring.datasource.password=your_password
```

```
spring.datasource.driver-class-name=com.microsoft.sqlserver.jdbc.SQLServerDriver
```

# Sử dụng Hibernate với JPA

```
spring.jpa.hibernate.ddl-auto=none
```

```
spring.jpa.show-sql=true
```

```
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.SQLServerDialect
```

### Bước 2: Cấu hình mô hình Product Entity

Tạo một lớp Java để ánh xạ với bảng Product trong cơ sở dữ liệu:

java

Sao chép mã

```
import javax.persistence.Entity;
```

```
import javax.persistence.GeneratedValue;
```

```
import javax.persistence.GenerationType;
```

```

import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "Product")
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long productId;
    private String productName;
    private Integer supplierId;
    private Integer categoryId;
    private String quantityPerUnit;
    private Double unitPrice;
    private Integer unitsInStock;
    private Integer unitsOnOrder;
    private Integer reorderLevel;
    private Boolean discontinued;

    // Getters và Setters
    public Long getProductId() { return productId; }
    public void setProductId(Long productId) { this.productId = productId; }
    public String getProductName() { return productName; }
    public void setProductName(String productName) { this.productName =
productName; }
    public Integer getSupplierId() { return supplierId; }

```

```

    public void setSupplierId(Integer supplierId) { this.supplierId = supplierId;
}

    public Integer getCategoryId() { return categoryId; }

    public void setCategoryId(Integer categoryId) { this.categoryId =
categoryId; }

    public String getQuantityPerUnit() { return quantityPerUnit; }

    public void setQuantityPerUnit(String quantityPerUnit) {
this.quantityPerUnit = quantityPerUnit; }

    public Double getUnitPrice() { return unitPrice; }

    public void setUnitPrice(Double unitPrice) { this.unitPrice = unitPrice; }

    public Integer getUnitsInStock() { return unitsInStock; }

    public void setUnitsInStock(Integer unitsInStock) { this.unitsInStock =
unitsInStock; }

    public Integer getUnitsOnOrder() { return unitsOnOrder; }

    public void setUnitsOnOrder(Integer unitsOnOrder) { this.unitsOnOrder =
unitsOnOrder; }

    public Integer getReorderLevel() { return reorderLevel; }

    public void setReorderLevel(Integer reorderLevel) { this.reorderLevel =
reorderLevel; }

    public Boolean getDiscontinued() { return discontinued; }

    public void setDiscontinued(Boolean discontinued) { this.discontinued =
discontinued; }

}

```

### **Bước 3: Sử dụng JpaRepository để lấy danh sách sản phẩm**

Tạo một repository để truy xuất dữ liệu từ cơ sở dữ liệu sử dụng JPA.

java

Sao chép mã

```

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

```

@Repository

```
public interface ProductRepository extends JpaRepository<Product, Long> {  
}
```

Tạo một Service để xử lý việc lấy danh sách sản phẩm.

java

Sao chép mã

```
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;
```

```
import java.util.List;
```

@Service

```
public class ProductService {
```

```
    @Autowired
```

```
    private ProductRepository productRepository;
```

```
    public List<Product> getAllProducts() {  
        return productRepository.findAll();  
    }  
}
```

Tạo một Controller để cung cấp endpoint lấy danh sách sản phẩm qua HTTP.

java

Sao chép mã

```
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RequestMapping;
```

```
import org.springframework.web.bind.annotation.RestController;
```

```
import java.util.List;
```

```
@RestController
```

```
@RequestMapping("/api/products")
```

```
public class ProductController {
```

```
    @Autowired
```

```
    private ProductService productService;
```

```
    @GetMapping
```

```
    public List<Product> getProducts() {
```

```
        return productService.getAllProducts();
```

```
    }
```

```
}
```

#### **Bước 4: Sử dụng DriverManager để thực hiện truy vấn SQL thuần**

Nếu muốn sử dụng DriverManager thay vì JpaRepository, bạn có thể thiết lập một dịch vụ khác để thực hiện truy vấn SQL thuần bằng JDBC.

```
java
```

Sao chép mã

```
import org.springframework.stereotype.Service;
```

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
```

```
import java.sql.PreparedStatement;
```

```
import java.sql.ResultSet;
```

```
import java.util.ArrayList;
```



```

import java.util.List;

@Service

public class ProductJdbcService {

    private final String jdbcUrl =
"jdbc:sqlserver://localhost:1433;databaseName=Northwind";

    private final String username = "your_username";
    private final String password = "your_password";

    public List<Product> getAllProducts() {
        List<Product> products = new ArrayList<>();

        try (Connection conn = DriverManager.getConnection(jdbcUrl,
username, password)) {
            String query = "SELECT * FROM Product";
            PreparedStatement ps = conn.prepareStatement(query);
            ResultSet rs = ps.executeQuery();

            while (rs.next()) {
                Product product = new Product();
                product.setProductId(rs.getLong("productId"));
                product.setProductName(rs.getString("productName"));
                product.setSupplierId(rs.getInt("supplierId"));
                product.setCategoryId(rs.getInt("categoryId"));
                product.setQuantityPerUnit(rs.getString("quantityPerUnit"));
                product.setUnitPrice(rs.getDouble("unitPrice"));
                product.setUnitsInStock(rs.getInt("unitsInStock"));
            }
        }
    }
}

```

```

        product.setUnitsOnOrder(rs.getInt("unitsOnOrder"));
        product.setReorderLevel(rs.getInt("reorderLevel"));
        product.setDiscontinued(rs.getBoolean("discontinued"));

        products.add(product);
    }

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

    return products;
}
}

```

Để sử dụng dịch vụ này, tạo một Controller mới hoặc cập nhật controller hiện tại:

java

Sao chép mã

```

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController
@RequestMapping("/api/products-jdbc")
public class ProductJdbcController {

```

@Autowired

```
private ProductJdbcService productJdbcService;
```

@GetMapping

```
public List<Product> getProducts() {  
    return productJdbcService.getAllProducts();  
}  
}
```

### **Chạy ứng dụng**

Chạy ứng dụng Spring Boot và truy cập các endpoint:

- **/api/products**: Sử dụng JpaRepository
- **/api/products-jdbc**: Sử dụng DriverManager