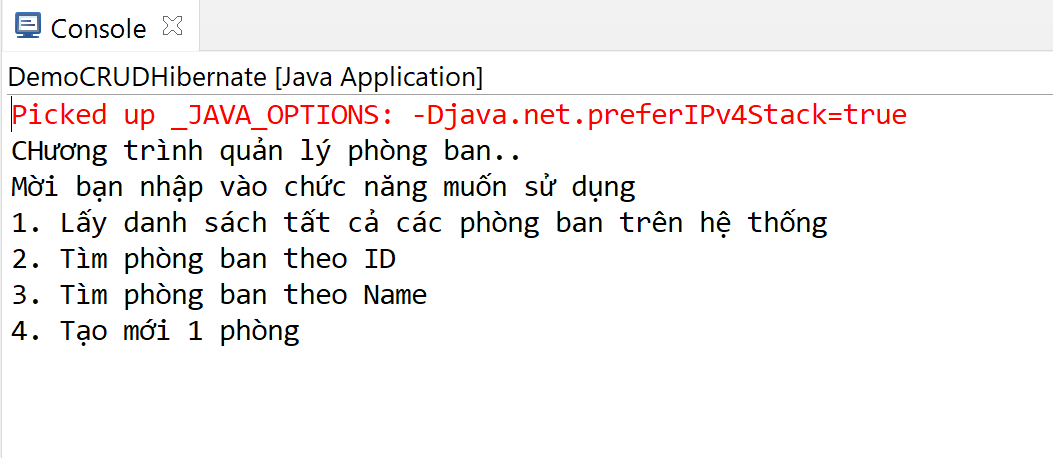
Tạo ứng dụng kết nối tới DB TestingSystem bằng Hibernate sử dụng bảng Department trong DB- Tích hợp Lombok để tạo Getter Setter, Contrustor, ToString tự động.



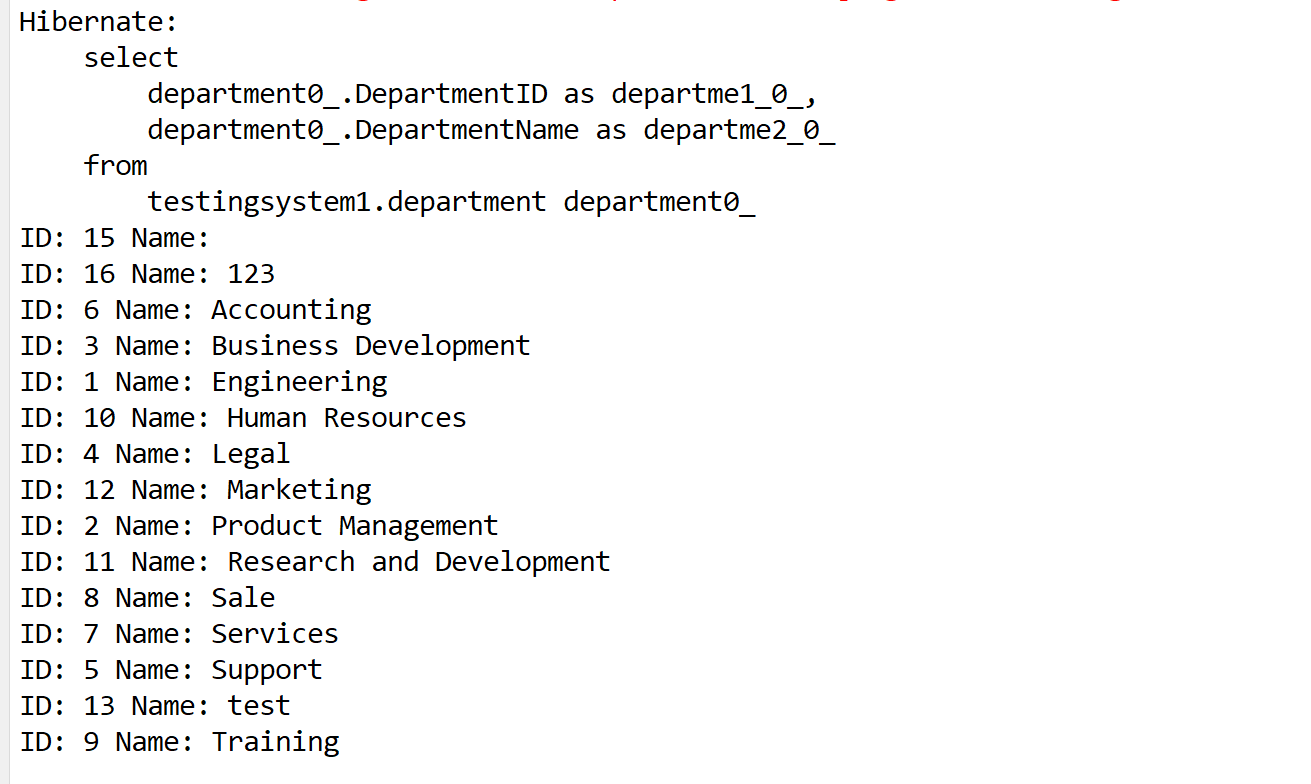
* Thêm bước 5 để tích hợp Lombok
* Viết lại Class Department theo cách viết Lombok.

# **Giao diện chính chương trình**

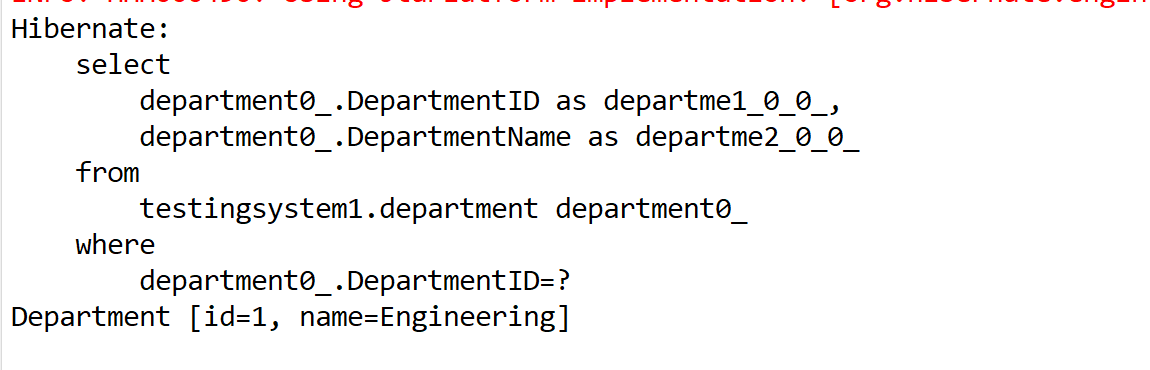
## Giao diện chính:



## Lấy danh sách tất cả phòng ban:



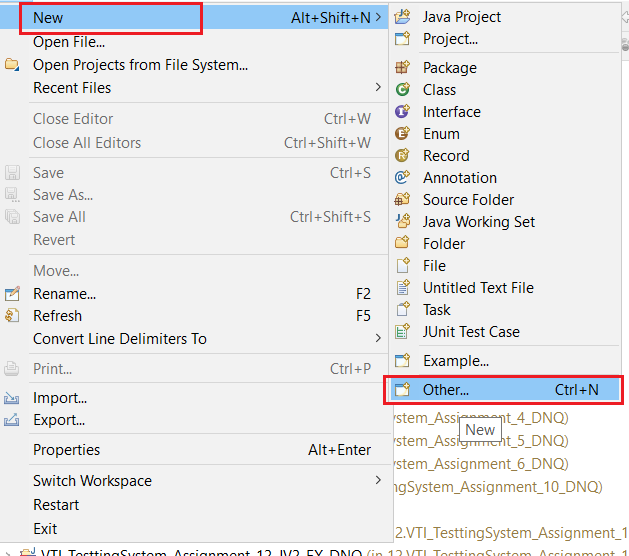
## Lấy phòng ban theo ID:

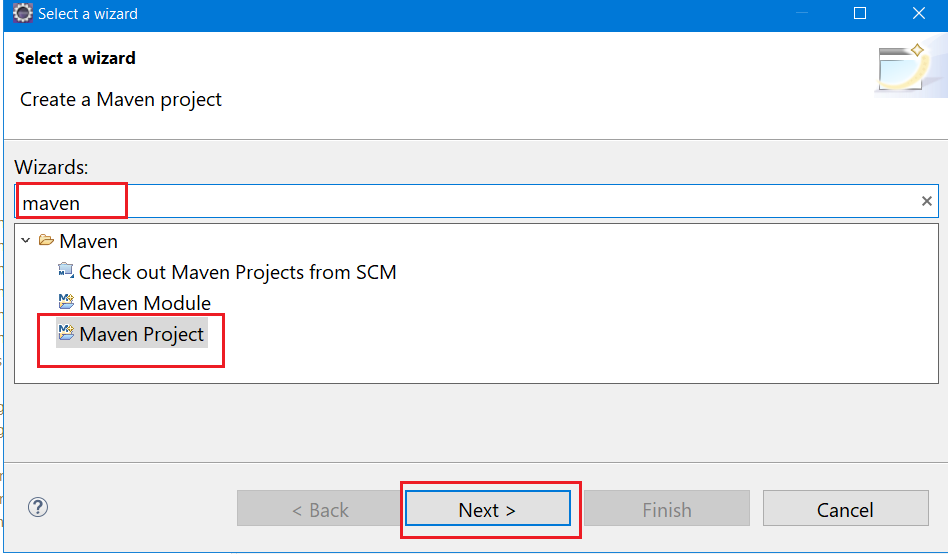


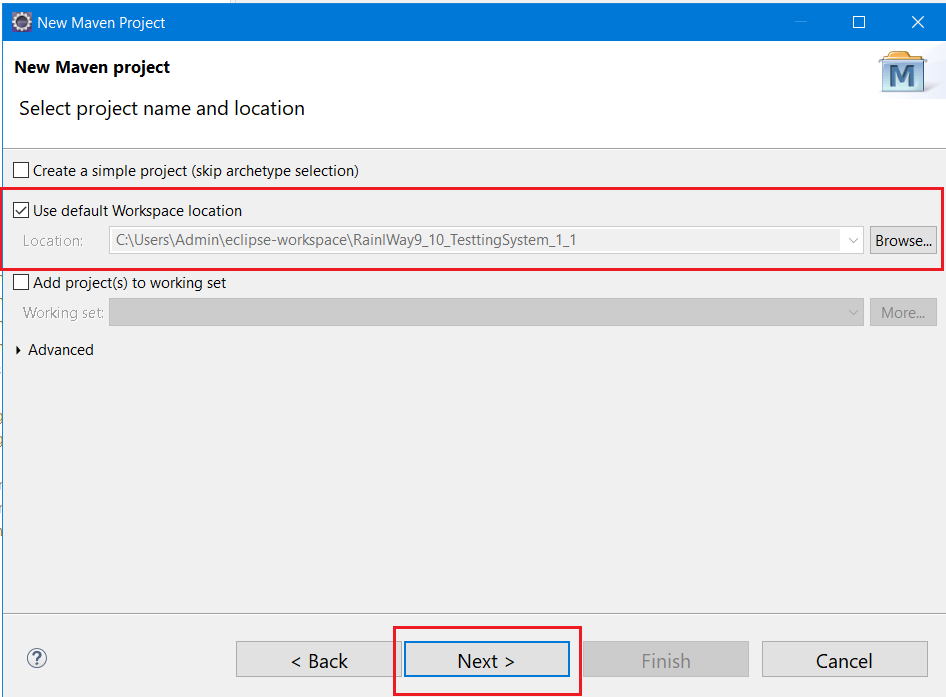
# **Tạo DB.**

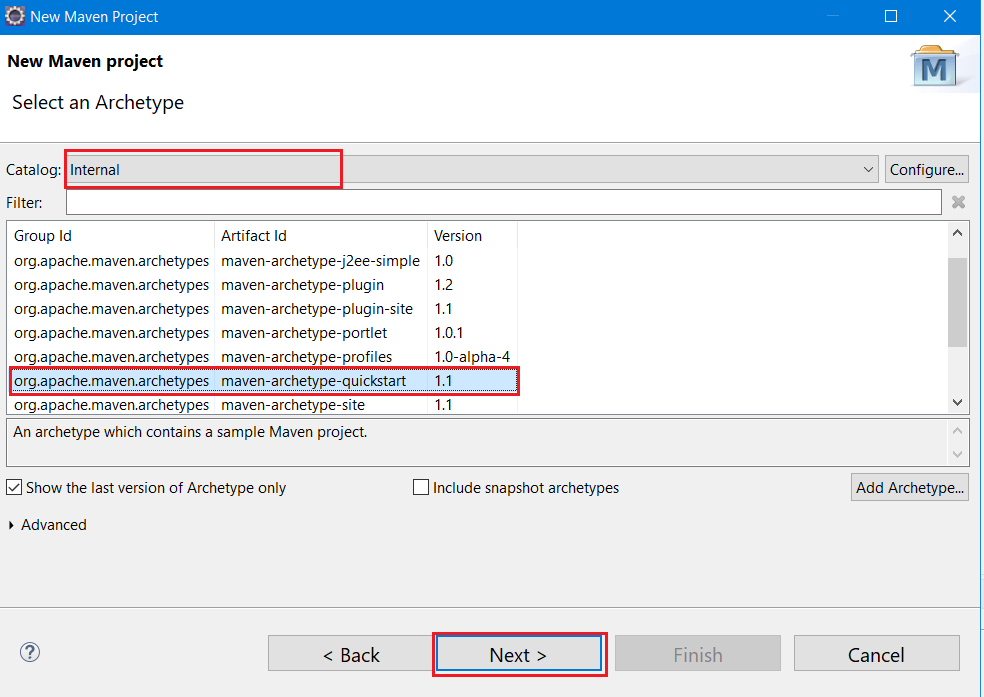
|  |
| --- |
| -- create database  DROP DATABASE IF EXISTS TestingSystem;  CREATE DATABASE TestingSystem;  USE TestingSystem;  -- create table 1: Department  DROP TABLE IF EXISTS Department;  CREATE TABLE Department(  DepartmentID TINYINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  DepartmentName NVARCHAR(30) NOT NULL UNIQUE KEY  );  -- Add data Department  INSERT INTO Department(DepartmentName)  VALUES  (N'Marketing' ),  (N'Sale' ),  (N'Bảo vệ' ),  (N'Nhân sự' ),  (N'Kỹ thuật' ),  (N'Tài chính' ),  (N'Phó giám đốc'),  (N'Giám đốc' ),  (N'Thư kí' ),  (N'Bán hàng' );` |

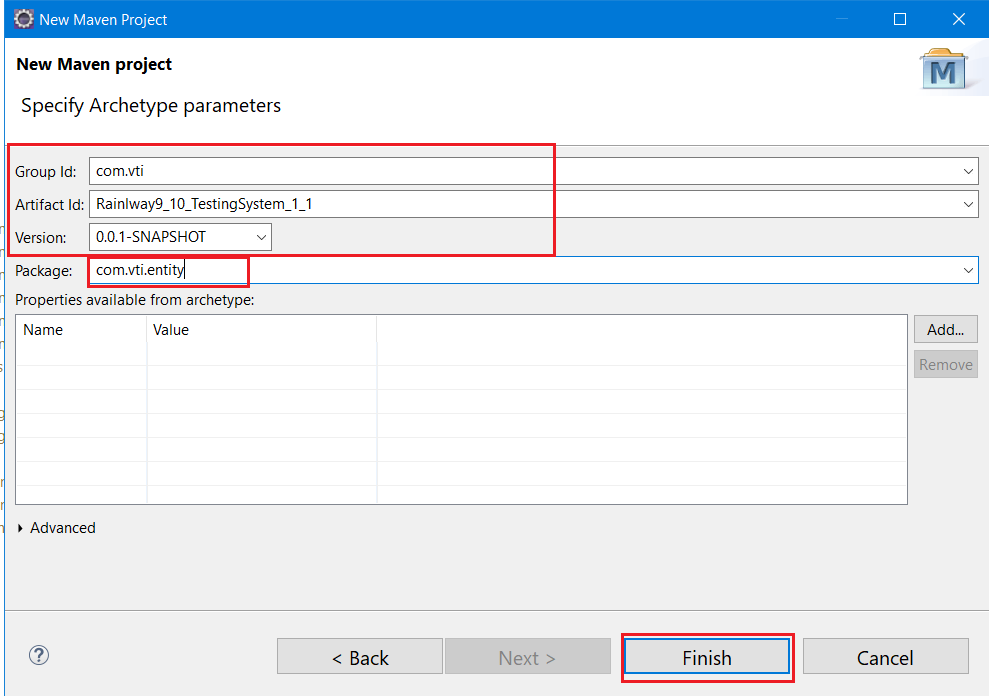
# **Tạo** Project maven**: hình minh họa.**

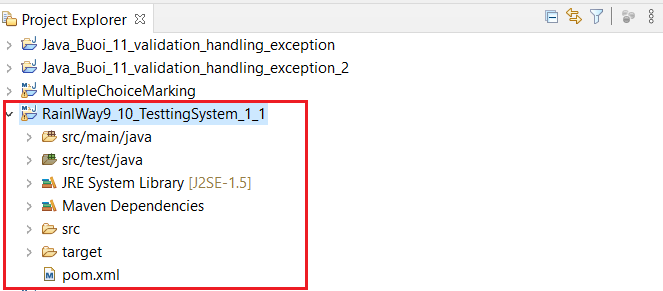










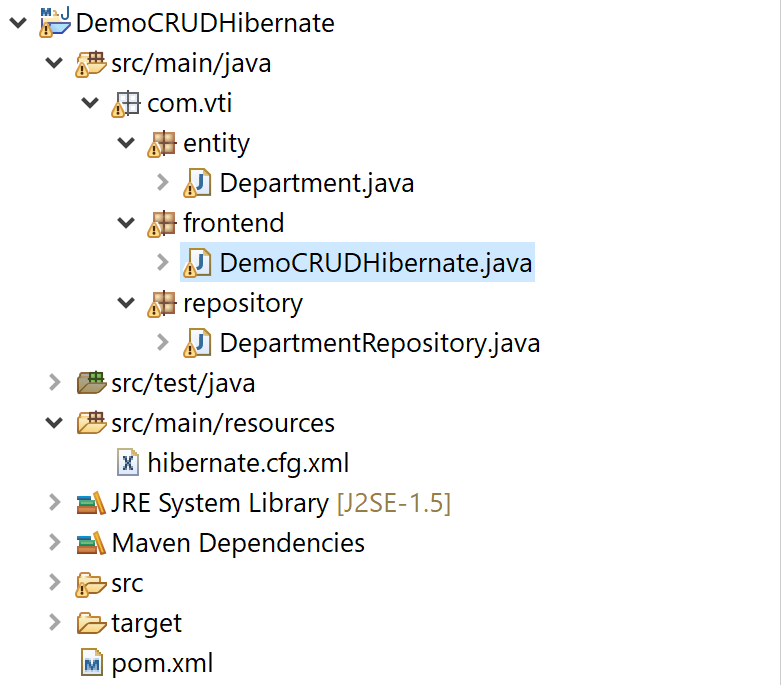


# **Tạo** khung **chương trình:**

Tạo mới 1 project: DemoCRUDHibernate, Tạo mới các package và class như dưới:

Package: com.vti.entity, com.vti.frontend, com.vti.repository

Class: DepartmentRepository, Department, DemoCRUDHibernate

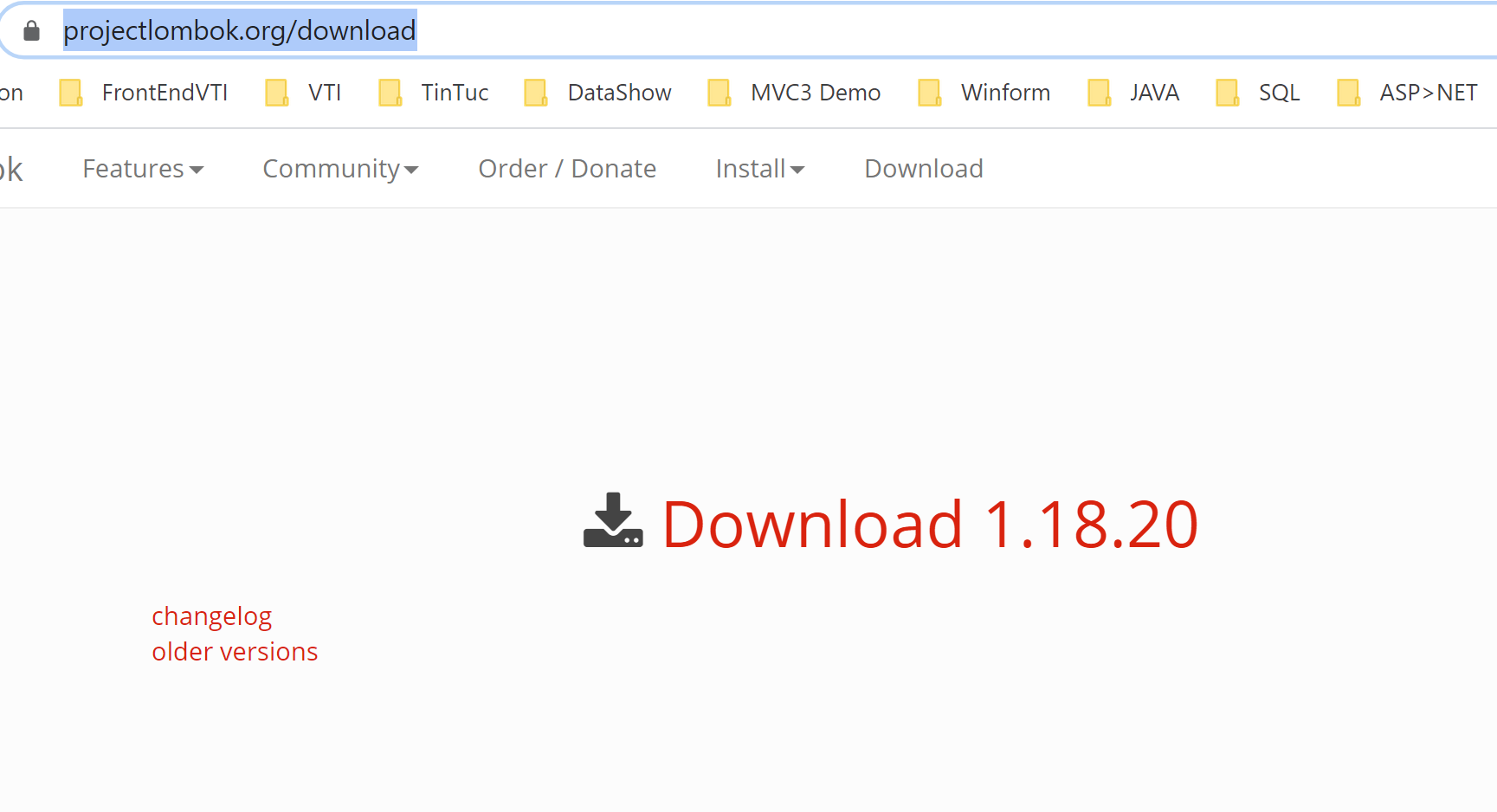


# **Tích hợp Lombok cho Project:**

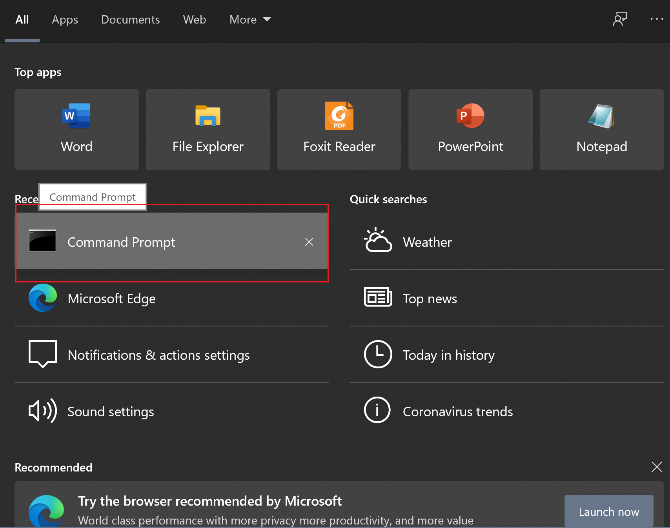
Bước 1: Thêm thư viện vào file Pom.xml trong Project:

|  |
| --- |
| <dependency>  <groupId>org.projectlombok</groupId>  <artifactId>lombok</artifactId>  <version>1.18.20</version>  </dependency> |

Bước 2. Tải Lombok từ link: <https://projectlombok.org/download> về máy:

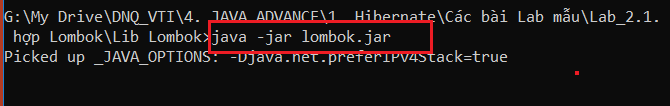


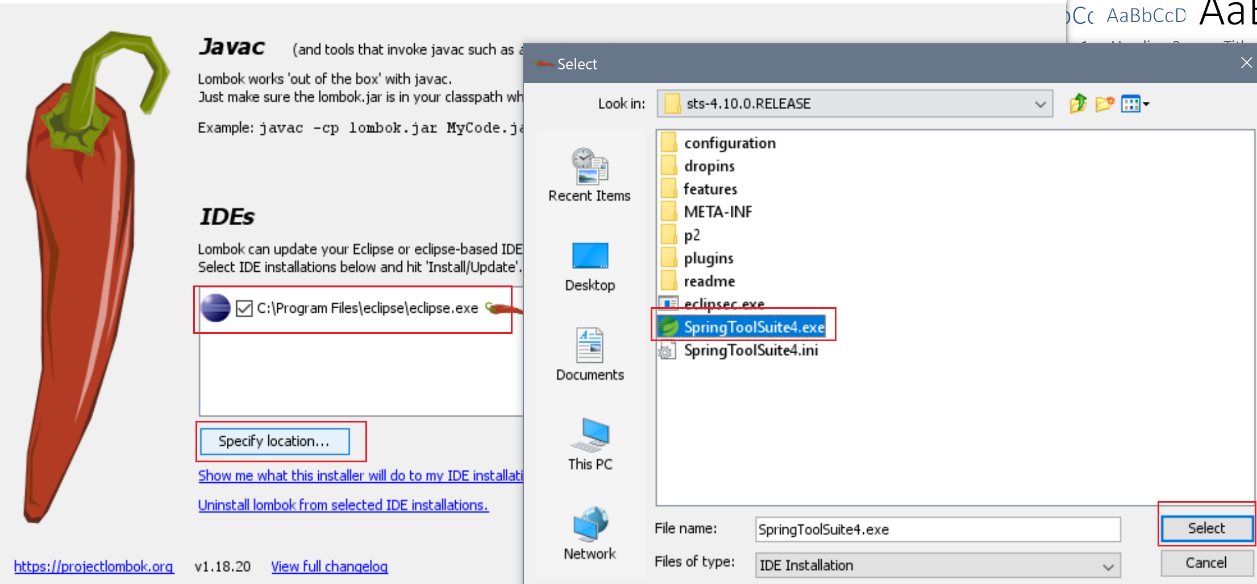
Bước 3. Mở cmd với quyền Admin(Nhấn ctrl+ Shif+ Enter) để cài đặt Lombok cho Eclipse và Spring Tool Suite 4:



cd: G:\My Drive\DNQ\_VTI\4. JAVA ADVANCE\2. Spring\LAB\_16\_Demo LomBok\Lib Lombok, vào thư mục chứa file vừa tải. và chạy file để cài đặt.

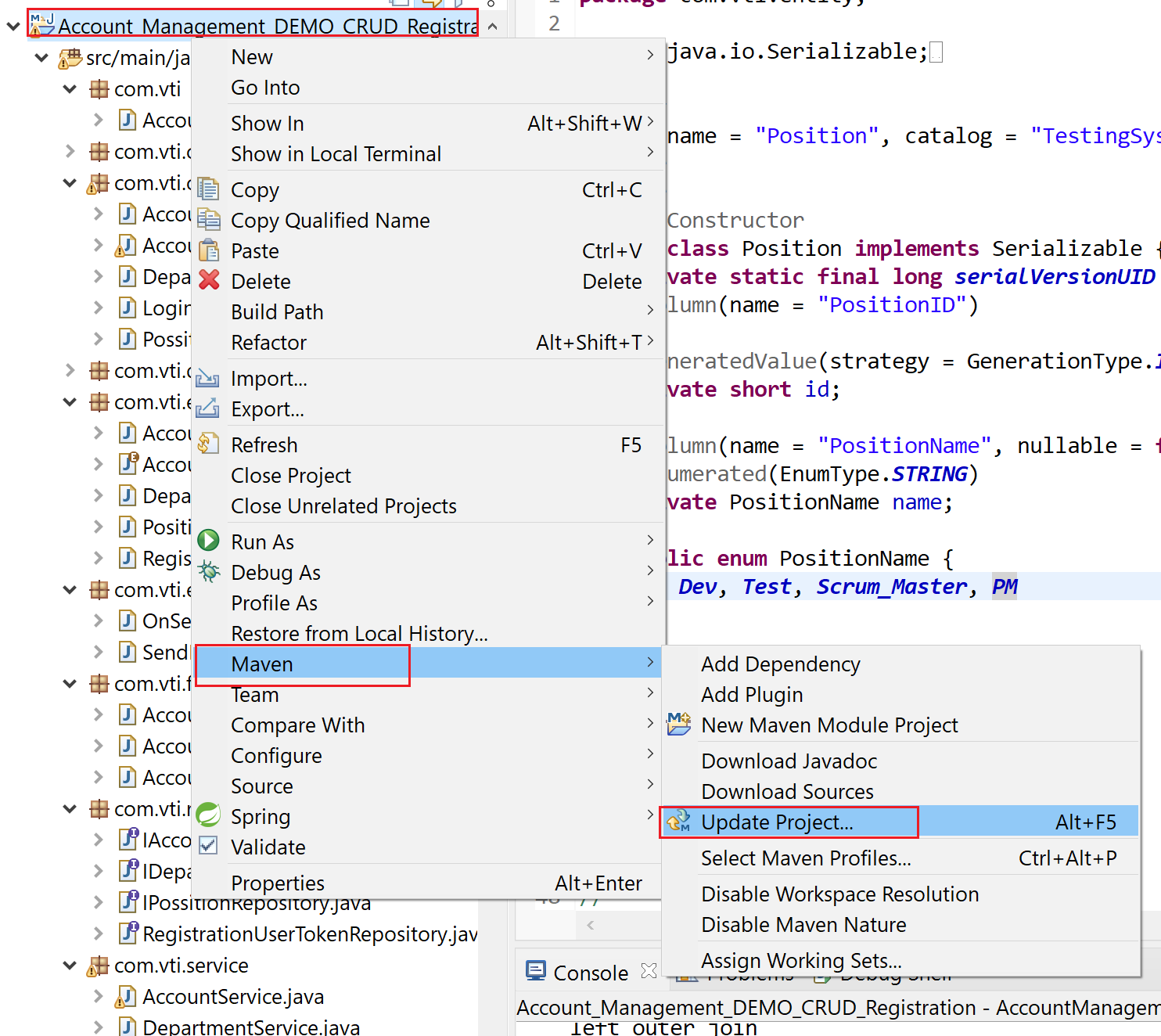
Dùng lệnh **java –jar lombok.jar** để khởi động chương trình cài đặt



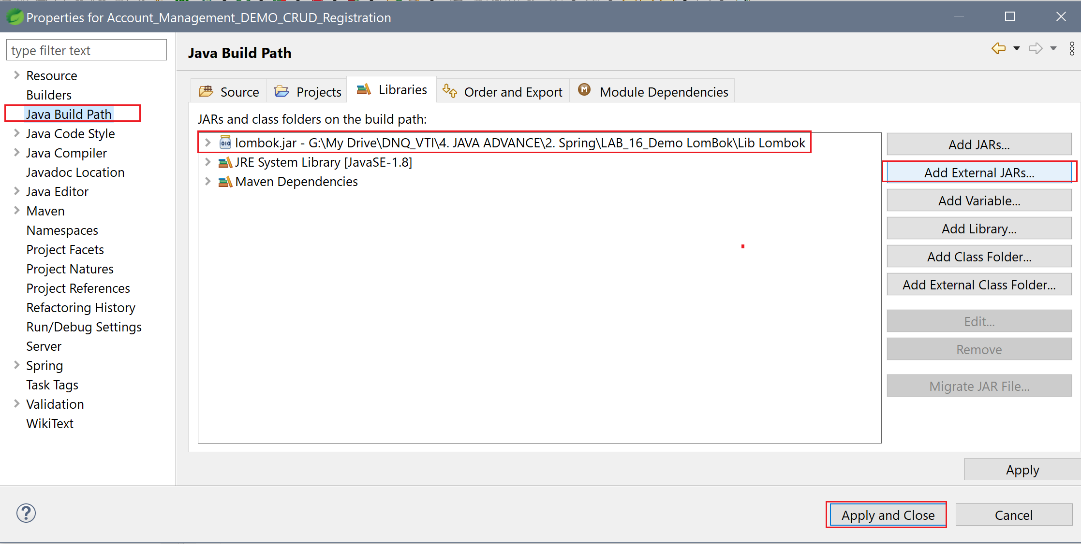




Bước 4. Thực hiện Update lại Project trên Spring Tool Suite để Build lại cấu hình(Chú ý tắt mở lại ứng dụng.):

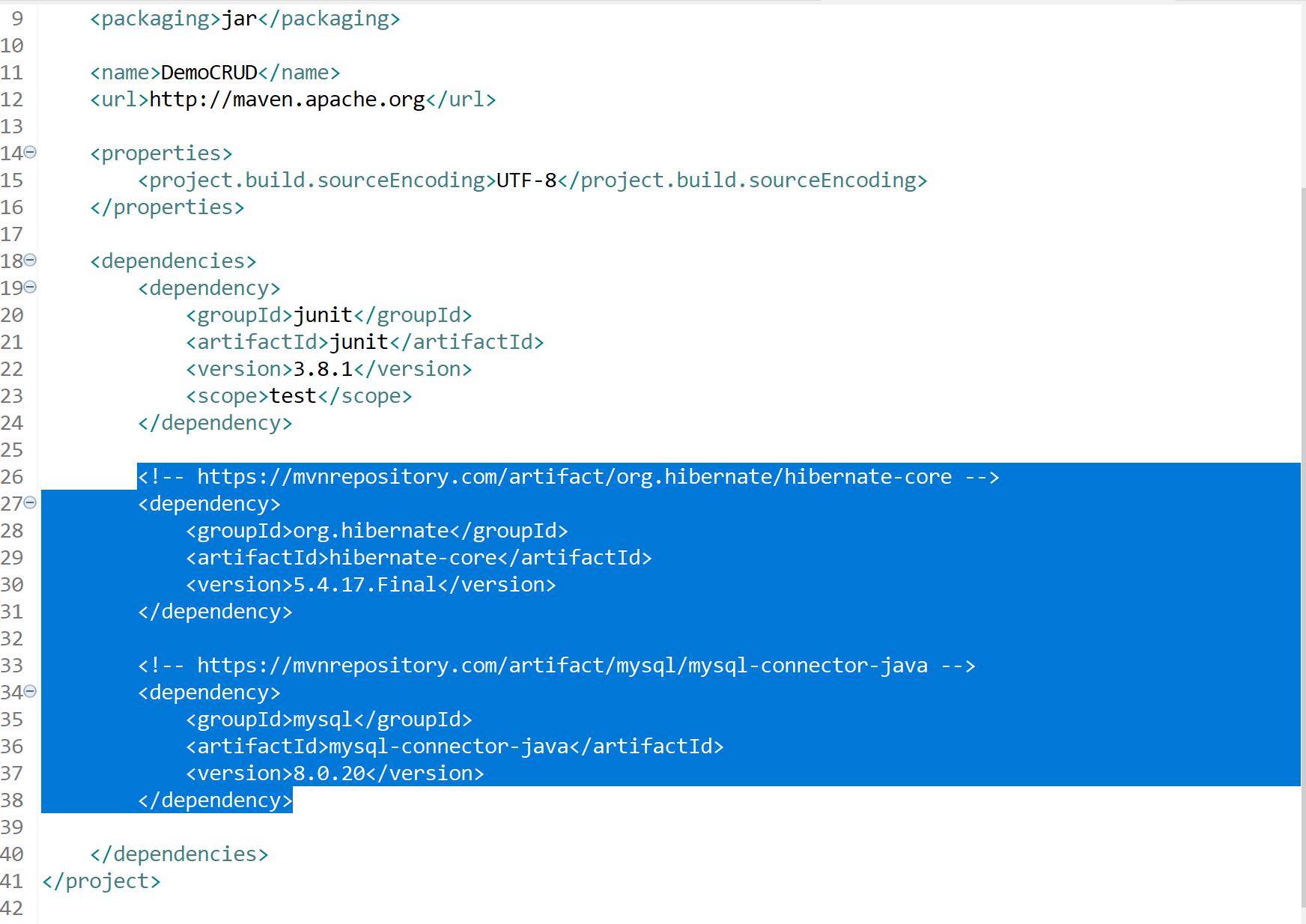


Bước 5. Nếu vẫn không được thì Add thêm thư viện Lombok:



# **Thêm các cấu hình cho file: pom.xml**

|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>com.vti</groupId>  <artifactId>RainlWay9\_10\_TesttingSystem\_1\_2</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>jar</packaging>  <name>RainlWay9\_10\_TesttingSystem\_1\_2</name>  <url>http://maven.apache.org</url>  <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  </properties>  <dependencies>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>3.8.1</version>  <scope>test</scope>  </dependency>  <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->  <dependency>  <groupId>org.hibernate</groupId>  <artifactId>hibernate-core</artifactId>  <version>5.4.17.Final</version>  </dependency>  <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>8.0.20</version>  </dependency>    <dependency>  <groupId>org.projectlombok</groupId>  <artifactId>lombok</artifactId>  <version>1.18.20</version>  </dependency>    </dependencies>  </project> |

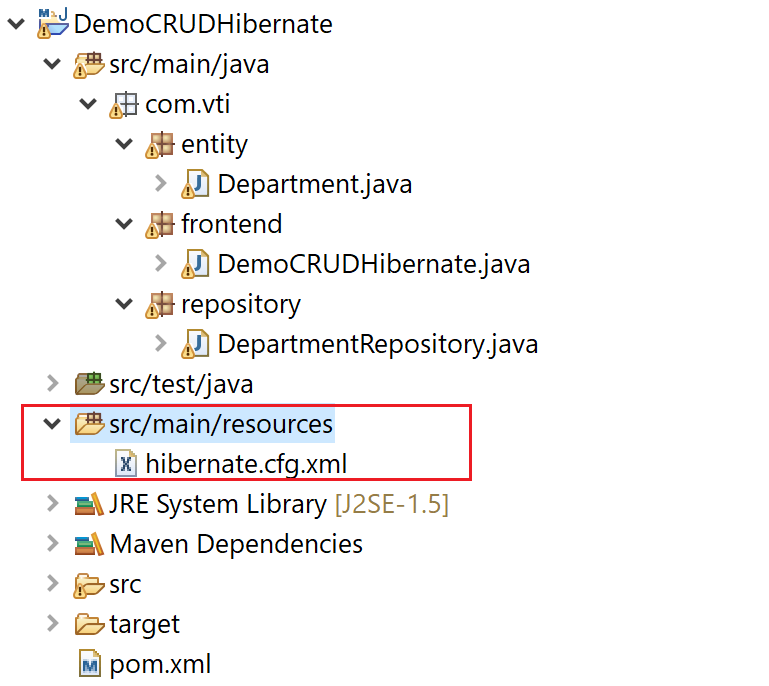




# **Thêm các cấu hình cho file: hibernate.cfg.xml trong resources**

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <!DOCTYPE hibernate-configuration PUBLIC  "-//Hibernate/Hibernate Configuration DTD 3.0//EN"  "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">  <hibernate-configuration>  <session-factory>  <!-- Database connection settings -->  <property name=*"connection.driver\_class"*>com.mysql.cj.jdbc.Driver</property>  <property name=*"connection.url"*>jdbc:mysql://localhost:3306/testingsystem1</property>  <property name=*"connection.username"*>root</property>  <property name=*"connection.password"*>root</property>  <!-- format code SQL -->  <property name=*"show\_sql"*>true</property>  <property name=*"hibernate.format\_sql"*>true</property>  <property name=*"connection.pool\_size"*>10</property>  <!-- other -->  <property name=*"hibernate.connection.characterEncoding"*>utf8</property>  </session-factory>  </hibernate-configuration> |





# **Tạo Class Department trong package com.vti.entity:**

|  |
| --- |
| **package** com.vti.entity;  **import** javax.persistence.Column;  **import** javax.persistence.Entity;  **import** javax.persistence.GeneratedValue;  **import** javax.persistence.GenerationType;  **import** javax.persistence.Id;  **import** javax.persistence.Table;  **import** lombok.Getter;  **import** lombok.NoArgsConstructor;  **import** lombok.Setter;  **import** lombok.ToString;  @Entity  @Table(name = "department", catalog = "testingsystem1")  @Getter  @Setter  @NoArgsConstructor  @ToString  **public** **class** Department {  @Column(name = "DepartmentID")  @Id  @GeneratedValue(strategy = GenerationType.***IDENTITY***)  **private** **short** id;  @Column(name = "DepartmentName", nullable = **false**, unique = **true**)  **private** String name;  } |

****

# **Tạo Class DepartmentRepository trong package com.vti.repository:**

|  |
| --- |
| **package** com.vti.repository;  **import** java.util.List;  **import** org.hibernate.Query;  **import** org.hibernate.Session;  **import** org.hibernate.boot.registry.StandardServiceRegistryBuilder;  **import** org.hibernate.cfg.Configuration;  **import** org.hibernate.service.ServiceRegistry;  **import** com.vti.entity.Department;  **public** **class** DepartmentRepository {  **private** org.hibernate.SessionFactory sessionFactory;  **public** DepartmentRepository() {  sessionFactory = buildSessionFactory();  }  **private** org.hibernate.SessionFactory buildSessionFactory() {  Configuration configuration = **new** Configuration();  configuration.configure("hibernate.cfg.xml");  configuration.addAnnotatedClass(Department.**class**);  ServiceRegistry serviceRegistry = **new** StandardServiceRegistryBuilder()  .applySettings(configuration.getProperties()).build();  **return** configuration.buildSessionFactory(serviceRegistry);  }  @SuppressWarnings("deprecation")  **public** List<Department> getAllDepartments() {  Session session = **null**;  **try** {  session = sessionFactory.openSession();  Query<Department> query = session.createQuery("FROM Department");  **return** query.list();  } **finally** {  **if** (session != **null**) {  session.close();  }  }  }  **public** Department getDepartmentByID(**short** id) {  Session session = **null**;  **try** {  session = sessionFactory.openSession();  Department department = session.get(Department.**class**, id);  **return** department;  } **finally** {  **if** (session != **null**) {  session.close();  }  }  }  **public** Department getDepartmentByName(String name) {  Session session = **null**;  **try** {  // get session  session = sessionFactory.openSession();  // create hql query  Query<Department> query = session.createQuery("FROM Department WHERE name = :nameParameter");  // set parameter  query.setParameter("nameParameter", name);  // get result  Department department = query.uniqueResult();  **return** department;  } **finally** {  **if** (session != **null**) {  session.close();  }  }  }  **public** **void** createDepartment(Department department) {  Session session = **null**;  **try** {  // get session  session = sessionFactory.openSession();  session.beginTransaction();  // create  session.save(department);  session.getTransaction().commit();  } **finally** {  **if** (session != **null**) {  session.close();  }  }  }  **public** **boolean** isDepartmentExistsByID(**short** id) {  // get department  Department department = getDepartmentByID(id);  // return result  **if** (department == **null**) {  **return** **false**;  }  **return** **true**;  }  **public** **boolean** isDepartmentExistsByName(String name) {  Department department = getDepartmentByName(name);  **if** (department == **null**) {  **return** **false**;  }  **return** **true**;  }  } |

****

# **Tạo Class DemoCRUDHibernate trong com.vti.frontend:**

|  |
| --- |
| **package** com.vti.frontend;  **import** java.util.List;  **import** java.util.Scanner;  **import** com.vti.entity.Department;  **import** com.vti.repository.DepartmentRepository;  **public** **class** DemoCRUDHibernate {  **public** **static** **void** main(String[] args) {  Scanner scanner = **new** Scanner(System.in);  **int** choose = menu();  **switch** (choose) {  **case** 1:  showAllDepartmet();  **break**;  **case** 2:  System.out.println("Nhập vào ID của phòng ban");  **short** id = scanner.nextShort();  showDepartmentByID(id);  **break**;  **case** 3:  System.out.println("Nhập vào tên phòng ban cần tìm kiếm");  String nameDep = scanner.nextLine();  showDepartmentByName(nameDep);  **break**;    **case** 4:  System.out.println("Tạo mới phòng ban");  System.out.println("Nhập tên phòng cần tạo:");  String nameDepCreate = scanner.nextLine();  Department dep = **new** Department();  dep.setName(nameDepCreate);  createDep(dep);  showAllDepartmet();  **break**;  **default**:  **break**;  }  }  **private** **static** **void** createDep(Department dep) {  DepartmentRepository depRepository = **new** DepartmentRepository();  depRepository.createDepartment(dep);  }  **private** **static** **void** showDepartmentByName(String nameDep) {  DepartmentRepository depRepository = **new** DepartmentRepository();  Department depByName = depRepository.getDepartmentByName(nameDep);  **if** (depByName == **null**) {  System.out.println("Không có phòng ban này trên hệ thống");  } **else** {  System.out.println(depByName.toString());  }  }  **private** **static** **void** showDepartmentByID(Short id) {  DepartmentRepository depRepository = **new** DepartmentRepository();  Department dep = depRepository.getDepartmentByID(id);  System.out.println(dep.toString());  }  **private** **static** **void** showAllDepartmet() {  System.out.println("Danh sách phòng ban trên hệ thống");  DepartmentRepository depRepository = **new** DepartmentRepository();  List<Department> listDep = depRepository.getAllDepartments();  **for** (Department department : listDep) {  System.out.println("ID: " + department.getId() + " Name: " + department.getName());  }  }  @SuppressWarnings("resource")  **private** **static** **int** menu() {  **while** (**true**) {  System.out.println("CHương trình quản lý phòng ban..");  System.out.println("Mời bạn nhập vào chức năng muốn sử dụng");  System.out.println("1. Lấy danh sách tất cả các phòng ban trên hệ thống");  System.out.println("2. Tìm phòng ban theo ID");  System.out.println("3. Tìm phòng ban theo Name");  System.out.println("4. Tạo mới 1 phòng");  Scanner sc = **new** Scanner(System.in);  **if** (sc.hasNextInt()) {  **int** i = sc.nextInt();  **if** ((i == 1) || (i == 2) || (i == 3)|| (i == 4)) {  **return** i;  } **else** {  System.out.println("Nhập lại");  }  } **else** {  System.out.println("Nhập lại");  }  }  }  } |