Be coder - <https://www.youtube.com/watch?v=4CJWeJG3a5w&list=PLQTYNpk8jwk2HlM9OeUT2J92mZZRD4vES&index=4&ab_channel=BeCoder>

# Lec – 4 Internal working of Spring Boot Application

A diagram of a run method

Description automatically generated

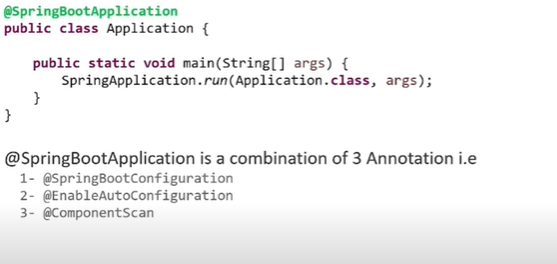
-> what run method do internally.

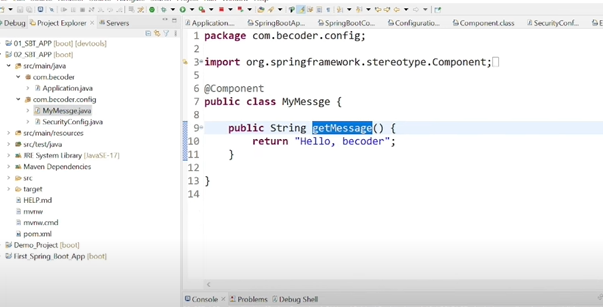
# 5 How work auto configuration in Spring Boot application

A screenshot of a computer

Description automatically generated

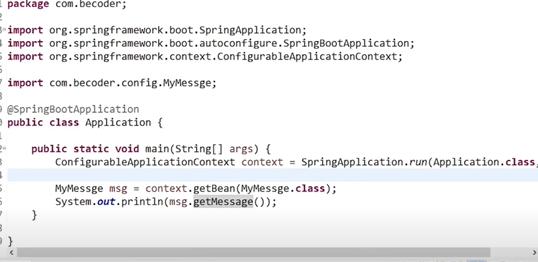
# 6 - Why use @SpringBootApplication annotation





-> Create one class called “MyMessage” define the method as shown in the picture above mark as “@component”.

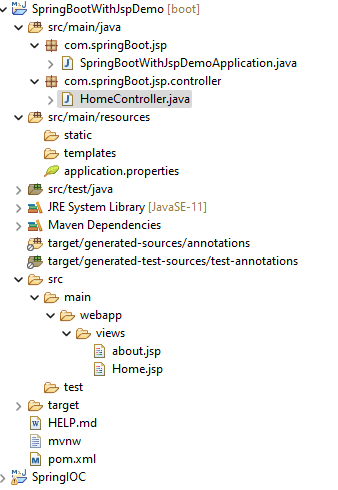
-> call that class from the @springBootApp class.



# 7 why use spring-boot-starter-parent | Spring Boot Tutorials In Hindi

## C:\Users\advishwa\OneDrive - Cisco\Desktop\Aditya Folder\Program-folder\Spring-Practice-folder

# 8 Configure JSP view in Spring boot web application



**@Controller**

**public class HomeController {**

**@RequestMapping("/home")**

**public String home() {**

**System.*out*.println("jai shree ram");**

**return "Home";**

**}**

**@RequestMapping("/about")**

**public String about() {**

**return "about";**

**}**

**}**

about.jsp

-----------

**<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"***

**pageEncoding=*"ISO-8859-1"*%>**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset=*"ISO-8859-1"*>**

**<title>Insert title here</title>**

**</head>**

**<body>**

**<h1>This is Aditya Tutorials </h1>**

**</body>**

**</html>**

Home.jsp

-------------

**<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"***

**pageEncoding=*"ISO-8859-1"*%>**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset=*"ISO-8859-1"*>**

**<title>Insert title here</title>**

**</head>**

**<body>**

**<h1>Home Page</h1>**

**</body>**

**</html>**

application.properties

-----------------------------

**spring.mvc.view.prefix=/views/**

**spring.mvc.view.suffix=.jsp**

pom.xml

**<dependencies>**

**<dependency>**

**<groupId>org.springframework.boot</groupId>**

**<artifactId>spring-boot-starter-web</artifactId>**

**</dependency>**

**<dependency>**

**<groupId>org.apache.tomcat.embed</groupId>**

**<artifactId>tomcat-embed-jasper</artifactId>**

**<scope>provided</scope>**

**</dependency>**

**<dependency>**

**<groupId>javax.servlet</groupId>**

**<artifactId>jstl</artifactId>**

**<scope>provided</scope>**

**</dependency>**

**<dependency>**

**<groupId>org.springframework.boot</groupId>**

**<artifactId>**

**spring-boot-starter-test</artifactId>**

**<scope>test</scope>**

**</dependency>**

**</dependencies>**

# 9 application.properties file why used in spring boot

A close-up of a computer screen

Description automatically generated

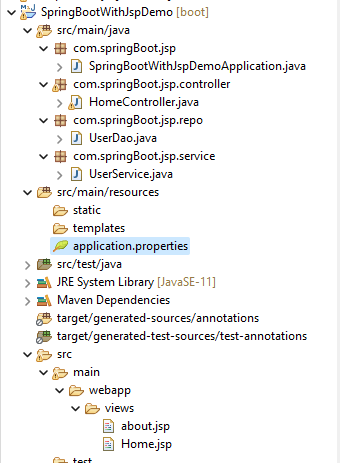
# 10 Dependency injection in Spring Boot Project based Example

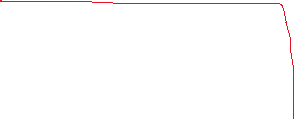
A screenshot of a computer

Description automatically generated

A computer screen shot of a computer code

Description automatically generated





-> Check this folder structure when you make the jsp pages.

application.properties

-----------------------------

**spring.mvc.view.prefix=/views/**

**spring.mvc.view.suffix=.jsp**

**HomeController.java**

**---------------------**

**@Controller**

**public class HomeController {**

**@Autowired**

**UserService service;**

**@RequestMapping("/")**

**public String home() {**

**System.*out*.println("jai shree ram");**

**return "Home";**

**}**

**@RequestMapping("/registerUser")**

**public String about() {**

**String saveUser = service.saveUser();**

**System.*out*.println(saveUser);**

**Model model = (Model) new ModelAndView();**

**return "about";**

**}**

**}**

**UserDao.java**

**---------------**

**@Repository**

**public class UserDao {**

**public boolean save() {**

**return true;**

**}**

**}**

**UserService.java**

**------------------**

**@Service**

**public class UserService {**

**@Autowired**

**UserDao dao;**

**public String saveUser() {**

**boolean f = dao.save();**

**if(f) {**

**return "Registration successfull";**

**}**

**else {**

**return "server error";**

**}**

**}**

**}**

**about.jsp**

**<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"***

**pageEncoding=*"ISO-8859-1"*%>**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset=*"ISO-8859-1"*>**

**<title>Insert title here</title>**

**</head>**

**<body>**

**<h1>This is Aditya Tutorials </h1>**

**</body>**

**</html>**

**Home.jsp**

**<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"***

**pageEncoding=*"ISO-8859-1"*%>**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset=*"ISO-8859-1"*>**

**<title>Insert title here</title>**

**</head>**

**<body>**

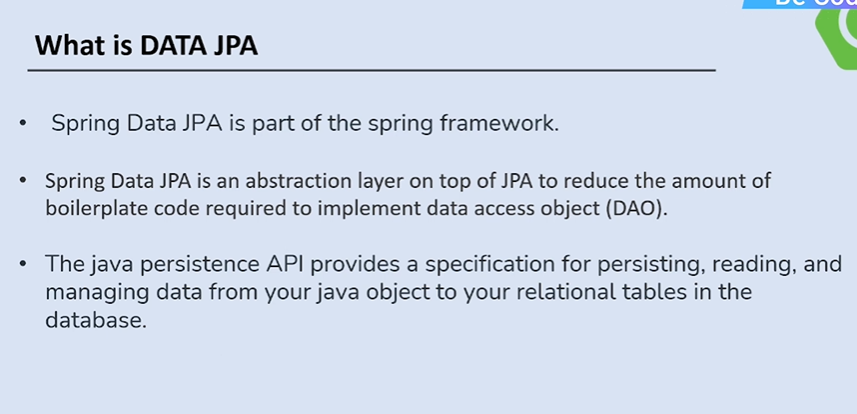
**<h1>Home Page</h1>**

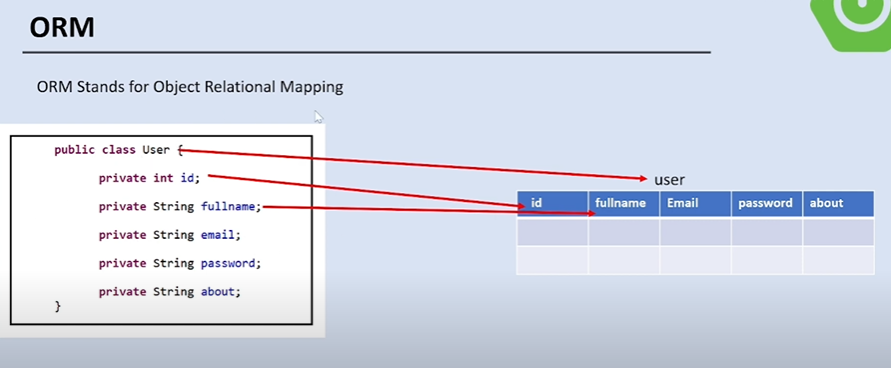
**<a href=*"registerUser"* > Register </a>**

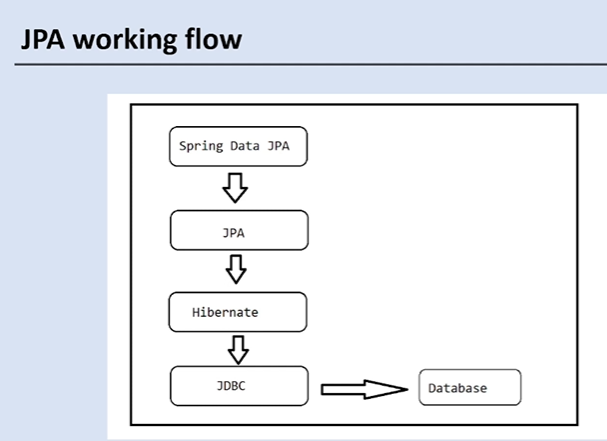
**</body>**

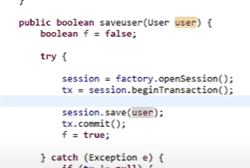
**</html>**

# 11,12,13 Introduction of Spring Data JPA

****

****





-> in hibernate we need to open the session , start the transaction ,execute the query and then commit the query so in this hibernate we need to do lot of code but in JPA just we need to call the method like save, update, findById

Very less code.

-> JPA internally using the hibernate only.

13 Crud Operation Using Spring DATA JPA | Spring Data JPA Tutorials

**pom.xml**

**<dependencies>**

**<dependency>**

**<groupId>org.springframework.boot</groupId>**

**<artifactId>spring-boot-starter-data-jpa</artifactId>**

**</dependency>**

**<dependency>**

**<groupId>org.springframework.boot</groupId>**

**<artifactId>spring-boot-starter-web</artifactId>**

**</dependency>**

**<dependency>**

**<groupId>com.mysql</groupId>**

**<artifactId>mysql-connector-j</artifactId>**

**<scope>runtime</scope>**

**</dependency>**

**<dependency>**

**<groupId>org.springframework.boot</groupId>**

**<artifactId>spring-boot-starter-test</artifactId>**

**<scope>test</scope>**

**</dependency>**

**</dependencies>**

**A screenshot of a computer program

Description automatically generated**

**SpringBootJpaApplication.java**

**package com.springBoot.jsp;**

**import java.util.List;**

**import java.util.Optional;**

**import org.springframework.boot.SpringApplication;**

**import org.springframework.boot.autoconfigure.SpringBootApplication;**

**import org.springframework.context.ConfigurableApplicationContext;**

**import com.springBoot.jsp.entity.Student;**

**import com.springBoot.jsp.repo.StudentRepo;**

**@SpringBootApplication**

**public class SpringBootJpaApplication {**

**public static void main(String[] args) {**

**ConfigurableApplicationContext context = SpringApplication.*run*(SpringBootJpaApplication.class, args);**

**StudentRepo studentRepo = context.getBean(StudentRepo.class);**

**Student student1 = new Student();**

**student1.setName("Naveen");**

**student1.setAddress("kahi nahi");**

**// Student student2 = new Student();**

**// student2.setName("laxmi");**

**// student2.setAddress("sab jagah");**

**//**

**// Student student3 = new Student();**

**// student3.setName("Abhishek");**

**// student3.setAddress("kachara ghar");**

**// Student save = studentRepo.save(student1);**

**// System.out.println("data is save::"+save);**

**// List<Student> saveAll = studentRepo.saveAll(List.of(student1,student2,student3));**

**// System.out.println("Printing this saveAll data");**

**// saveAll.forEach(Student -> System.out.println(Student));**

**// finds by id**

**// Optional<Student> findById = Optional.of(studentRepo.findById(8).get());**

**// System.out.println("Id is ::"+ findById);**

**// delete by ID**

**int id = 8;**

**String name = "Naveen";**

**System.*out*.println("Printing all the data");**

**List<Student> allData = studentRepo.findAll();**

**for (Student student : allData) {**

**if(student.getName().equals(name)) {**

**studentRepo.delete(student);**

**}**

**else {**

**System.*out*.println(student);**

**}**

**}**

**}**

**}**

Student.java

-=============

**@Entity**

**public class Student {**

**@Id**

**@GeneratedValue(strategy = GenerationType.*IDENTITY*)**

**private int id;**

**private String name;**

**private String address;**

**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 getAddress() {**

**return address;**

**}**

**public void setAddress(String address) {**

**this.address = address;**

**}**

**@Override**

**public String toString() {**

**return "Student [id=" + id + ", name=" + name + ", address=" + address + "]";**

**}**

**}**

**StudentRepo.java**

**public interface StudentRepo extends JpaRepository<Student, Integer>{**

**}**

application.properties

**spring.datasource.url=jdbc:mysql://localhost:3306/be\_code**

**spring.datasource.driver-class-name=com.mysql.jdbc.Driver**

**spring.datasource.username=root**

**spring.datasource.password=root**

**spring.jpa.hibernate.ddl-auto=update**

# 14 FindBy method and Custom Query in Spring DATA JPA

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

# 17 Pagination & Sorting in Spring Data JPA Example

**@SpringBootApplication**

**public class SpringBootJpaApplication {**

**public static void main(String[] args) {**

**ConfigurableApplicationContext context = SpringApplication.*run*(SpringBootJpaApplication.class, args);**

**StudentRepo studentRepo = context.getBean(StudentRepo.class);**

**// pagination**

**Sort sort = Sort.*by*("name").ascending();**

**Pageable pageable = PageRequest.*of*(0, 3,sort);**

**Page<Student> page = studentRepo.findAll(pageable);**

**page.stream().forEach(e->{**

**System.*out*.println(e);**

**});**

**System.*out*.println("Page number ::"+page.getNumber());**

**System.*out*.println("page size ::"+page.getSize());**

**System.*out*.println("Total Elements ::"+page.getTotalElements());**

**System.*out*.println("Total Pages ::"+page.getTotalPages());**

**}**

**}**

**public interface StudentRepo extends JpaRepository<Student, Integer>{**

**public Student findByName(String name);**

**public List<Student> findByAddress(String address);**

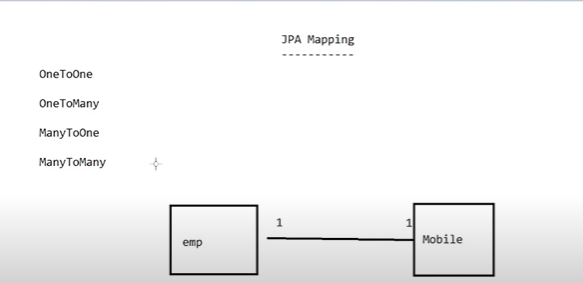
**public List<Student> findByNameAndAddress(String name, String address);**

**@Query("select u from Student u where u.name=?1 and u.address=?2")**

**public Student getStudentByNameAndAddress(String nm, String ad);**

**}**

# 18 Relationship Mapping in Data JPA in One Video

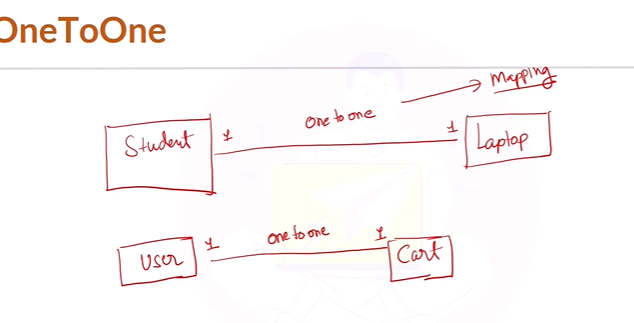


A diagram of a car model

Description automatically generated

<https://www.youtube.com/watch?v=H__ELR1y3FQ&ab_channel=LearnCodeWithDurgesh>

-> we will learn by this video



One to one mapping code

**import jakarta.persistence.CascadeType;**

**import jakarta.persistence.Entity;**

**import jakarta.persistence.GeneratedValue;**

**import jakarta.persistence.GenerationType;**

**import jakarta.persistence.Id;**

**import jakarta.persistence.OneToOne;**

**@Entity**

**public class Student {**

**@Id**

**@GeneratedValue(strategy = GenerationType.*IDENTITY*)**

**private int studentId;**

**private String studentName;**

**private String about;**

**@OneToOne(mappedBy = "student", cascade = CascadeType.*ALL*)**

**private Laptop laptop;**

**public int getStudentId() {**

**return studentId;**

**}**

**public void setStudentId(int studentId) {**

**this.studentId = studentId;**

**}**

**public String getStudentName() {**

**return studentName;**

**}**

**public void setStudentName(String studentName) {**

**this.studentName = studentName;**

**}**

**public String getAbout() {**

**return about;**

**}**

**public void setAbout(String about) {**

**this.about = about;**

**}**

**public Laptop getLaptop() {**

**return laptop;**

**}**

**public void setLaptop(Laptop laptop) {**

**this.laptop = laptop;**

**}**

**@Override**

**public String toString() {**

**return "Student [studentId=" + studentId + ", studentName=" + studentName + ", about=" + about + ", laptop="**

**+ laptop + "]";**

**}**

**}**

**@Entity**

**public class Laptop {**

**@Id**

**@GeneratedValue(strategy = GenerationType.*IDENTITY*)**

**private int laptopId;**

**private String modelNumber;**

**private String brand;**

**@OneToOne**

**private Student student;**

**public int getLaptopId() {**

**return laptopId;**

**}**

**public void setLaptopId(int laptopId) {**

**this.laptopId = laptopId;**

**}**

**public String getModelNumber() {**

**return modelNumber;**

**}**

**public void setModelNumber(String modelNumber) {**

**this.modelNumber = modelNumber;**

**}**

**public String getBrand() {**

**return brand;**

**}**

**public void setBrand(String brand) {**

**this.brand = brand;**

**}**

**public Student getStudent() {**

**return student;**

**}**

**public void setStudent(Student student) {**

**this.student = student;**

**}**

**@Override**

**public String toString() {**

**return "Laptop [laptopId=" + laptopId + ", modelNumber=" + modelNumber + ", brand=" + brand + ", student="**

**+ student + "]";**

**}**

**}**

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

**import org.springframework.boot.CommandLineRunner;**

**import org.springframework.boot.SpringApplication;**

**import org.springframework.boot.autoconfigure.SpringBootApplication;**

**import com.springBoot.jsp.StudentRepo.StudentRepo;**

**import com.springBoot.jsp.entity.Laptop;**

**import com.springBoot.jsp.entity.Student;**

**@SpringBootApplication**

**public class JpaMappingProjectApplication implements CommandLineRunner{**

**@Autowired**

**StudentRepo studentRepo;**

**public static void main(String[] args) {**

**SpringApplication.*run*(JpaMappingProjectApplication.class, args);**

**}**

**@Override**

**public void run(String... args) throws Exception {**

**// TODO Auto-generated method stub**

**// Student student = new Student();**

**// student.setStudentName("Amartya Vish");**

**// student.setAbout("Software engineer");**

**//**

**// // we are setting here laptop data**

**// Laptop laptop = new Laptop();**

**// laptop.setBrand("Dell G3");**

**// laptop.setModelNumber("84000");**

**//**

**// // This one is imp**

**// laptop.setStudent(student);**

**//**

**// // this also we need to save**

**// student.setLaptop(laptop);**

**//**

**// // saving the student data**

**// studentRepo.save(student);**

**// we are fetching the data**

**Student student = studentRepo.findById(2).get();**

**System.*out*.println("Student Name ::"+ student.getStudentName());**

**String modelNumber = student.getLaptop().getModelNumber();**

**System.*out*.println("Laptop Model Number ::"+ modelNumber);**

**}**

**}**

application.properties

**spring.datasource.url=jdbc:mysql://localhost:3306/jpa\_mapping**

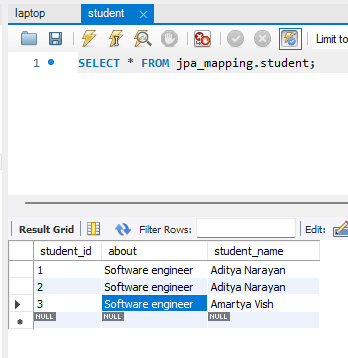
**spring.datasource.driver-class-name=com.mysql.jdbc.Driver**

**spring.datasource.username=root**

**spring.datasource.password=root**

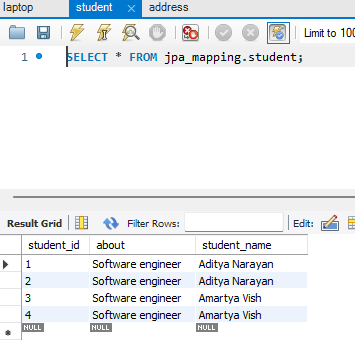
**spring.jpa.hibernate.ddl-auto=update**

A screenshot of a computer

Description automatically generated

A diagram of a business

Description automatically generated with medium confidence



A screenshot of a computer

Description automatically generated

**@Entity**

**public class Student {**

**@Id**

**@GeneratedValue(strategy = GenerationType.*IDENTITY*)**

**private int studentId;**

**private String studentName;**

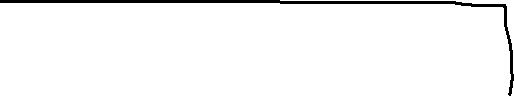
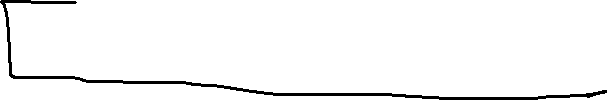
**private String about;**

**@OneToOne(mappedBy = "student", cascade = CascadeType.*ALL*)**

**private Laptop laptop;**

**// many address**

**@OneToMany(mappedBy = "student", cascade = CascadeType.*ALL*)**



**List<Address> addressList = new ArrayList<>();**

**public int getStudentId() {**

**return studentId;**

**}**

**public void setStudentId(int studentId) {**

**this.studentId = studentId;**

**}**

**public String getStudentName() {**

**return studentName;**

**}**

**public void setStudentName(String studentName) {**

**this.studentName = studentName;**

**}**

**public String getAbout() {**

**return about;**

**}**

**public void setAbout(String about) {**

**this.about = about;**

**}**

**public Laptop getLaptop() {**

**return laptop;**

**}**

**public void setLaptop(Laptop laptop) {**

**this.laptop = laptop;**

**}**

**public List<Address> getAddressList() {**

**return addressList;**

**}**

**public void setAddressList(List<Address> addressList) {**

**this.addressList = addressList;**

**}**

**@Override**

**public String toString() {**

**return "Student [studentId=" + studentId + ", studentName=" + studentName + ", about=" + about + ", laptop="**

**+ laptop + "]";**

**}**

**}**

**@Entity**

**public class Address {**

**@Id**

**@GeneratedValue(strategy = GenerationType.*IDENTITY*)**

**private int addressId;**

**private String street;**

**private String city;**

**private String country;**

**@ManyToOne**



**private Student student;**



**public int getAddressId() {**

**return addressId;**

**}**

**public void setAddressId(int addressId) {**

**this.addressId = addressId;**

**}**

**public String getStreet() {**

**return street;**

**}**

**public void setStreet(String street) {**

**this.street = street;**

**}**

**public String getCity() {**

**return city;**

**}**

**public void setCity(String city) {**

**this.city = city;**

**}**

**public String getCountry() {**

**return country;**

**}**

**public void setCountry(String country) {**

**this.country = country;**

**}**

**public Student getStudent() {**

**return student;**

**}**

**public void setStudent(Student student) {**

**this.student = student;**

**}**

**@Override**

**public String toString() {**

**return "Address [addressId=" + addressId + ", street=" + street + ", city=" + city + ", country=" + country**

**+ "]";**

**}**

**}**

