**SPRINGBOOT (84)**

Dependencies: Spring Web (Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.)

* File ->Import -> exiting maven projects

Ex: when we hit the url (/courses)..it will show the courses (Simple rest API using SpringBoot)

**Course.java**

package com.sup.springboot.learn\_spring\_boot;

public class Course {

private long id;

private String name;

private String author;

//getters,constructor,toString

}

**CourseController.java**

@RestController

**public** **class** CourseController {

@RequestMapping("/courses")

**public** List<Course> retrieveAllCourses() {

**return** Arrays.*asList*(

**new** Course(1, "Learn AWS", "in28minutes"),

**new** Course(2, "Learn DevOps", "in28minutes")

);}}

**LearnSpringBootApplication.java**

package com.sup.springboot.learn\_spring\_boot;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class LearnSpringBootApplication {

public static void main(String[] args) {

SpringApplication.run(LearnSpringBootApplication.class, args);

}

}

**NOTE: Spring Boot provides variety of starter projects:**

**Web Application & REST API** - Spring Boot Starter Web (spring-webmvc, spring-web, spring-boot-starter-tomcat, spring-boot-starter-json)

**Unit Tests** - Spring Boot Starter Test

**Talk to database using JPA** - Spring Boot Starter Data JPA

**Talk to database using JDBC** - Spring Boot Starter JDBC

**Secure your web application or REST API** - Spring Boot Starter Security

**NOTE: In application.properties**

**Logging.level.org.springframework =debug**

**NOTE: SPRINGBOOT Dev Tools**

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

</dependency>

By using this, server started automatically. We don’t have to start the server everytime.

But if we want to change the pom.xml, then we have to restart the server manually.

**Logging level:** trace < debug < info < warn < error < off

**NOTE:IF WE WANT TO CONFIGURE PROPERTIES**

**Application.properties:**

spring.application.name=learn-spring-boot

logging.level.org.springframework=debug

currency-service.url= http://default.sup.com

currency-service.username= defaultusername

currency-service.key= dedaultkey

**CurrencyServiceConfiguration.java**

package com.sup.springboot.learn\_spring\_boot;

import org.springframework.boot.context.properties.ConfigurationProperties;

import org.springframework.stereotype.Component;

@ConfigurationProperties(prefix = "currency-service")

@Component

public class CurrencyServiceConfiguration {

private String url;

private String username;

private String key;

public String getUrl() {

return url;

}

public void setUrl(String url) {

this.url = url;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getKey() {

return key;

}

public void setKey(String key) {

this.key = key;

}

}

**CurrencyConfigurationController.java**

package com.sup.springboot.learn\_spring\_boot;

import java.util.Arrays;

import java.util.List;

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

import org.springframework.web.bind.annotation.RequestMapping;

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

@RestController

public class CurrencyConfigurationController {

@Autowired

private CurrencyServiceConfiguration configuration;

@RequestMapping("/currency-configuration")

public CurrencyServiceConfiguration retrieveAllCourses() {

return configuration;

}

}

Note: we can see the following data when we hit the url(<http://localhost:8080/currency-configuration>)

currency-service.url= http://default.sup.com

currency-service.username= defaultusername

currency-service.key= dedaultkey

**Note: simple deployment in springboot**

There is an embedded server in springboot to make the deployment easy (spring-boot-starter-tomcat)

**Monitor Applications using Spring Boot Actuator**

* Monitor and manage your application in your production
* **Provides a number of endpoints:**

**beans** - Complete list of Spring beans in your app

**health** - Application health information

**metrics** - Application metrics

**mappings** - Details around Request Mappings

In pom.xml

<dependency>

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

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

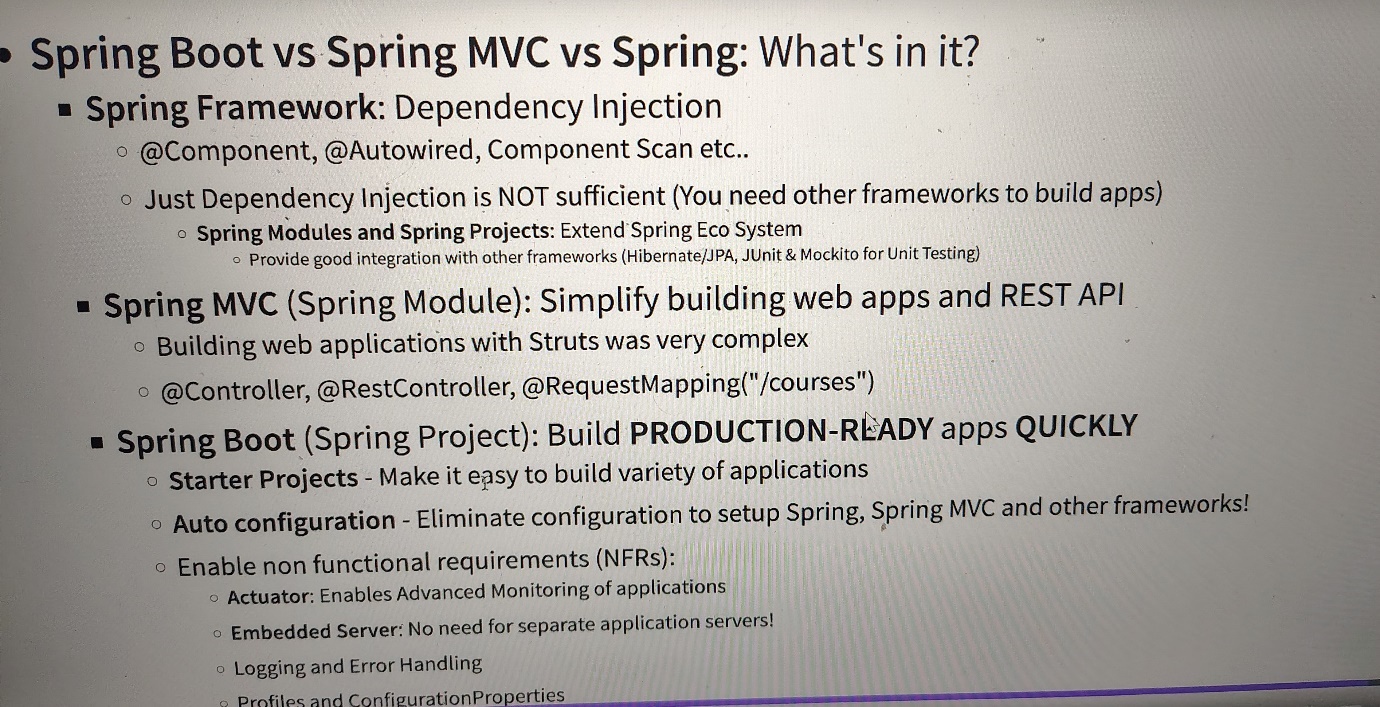
Now type the url in browser: <http://localhost:8080/actuator> to see the info

* To see the more no of links in the browser type the following in application.properties file

management.endpoints.web.exposure.include=\*

management.endpoints.web.exposure.include=health,metrics (if we want some specific endpoint)

**SPRINGBOOT VS SPRING MVC VS SPRING**

****

**JPA AND HIBERNATE**

01: Create a Spring Boot Project with H2

02: Create COURSE table

03: Use Spring JDBC to play with COURSE table

04: Use JPA and Hibernate to play with COURSE table

05: Use Spring Data JPA to play with COURSE table

Creating a project with dependencies (spring web,spring data jdbc, spring data JPA, H2 database)

Write the following line in application.properties file: spring.h2.console.enabled = true

Paste the url in browser to see the h2 database: <http://localhost:8080/h2-console>

Go the the console of eclipse and take (jdbc:h2:mem:b99383c3-08dc-453a-a99d-d4b7a7927215) and paste it in JDBC URL in the h2 interface in the browser. Click on connect

Write(spring.datasourse.url=jdbc:h2:mem:testdb) in application.properties

Now, go to the broser and refresh it and paste (jdbc:h2:mem:testdb) in JDBC URL and click on connect.

* Create schema.sql in src>main>resources

Write the following in schema.sql

create table course

(

id bigint not null,

name varchar(255) not null,

author varchar(255) not null,

primary key(id)

);

Start the server. Paste (<http://localhost:8080/h2-console>) in browser. We can see the course table got generated in the h2 database.

Q-> run insert into course (id, name, author) values(1, 'aws','sup');(insert the data into the database)

Ans: create the package (com.supr.springboot.learn\_jpa\_and\_hibernate.course.jdbc) with two classes (CourseJdbcRepository) and (CourseJdbcComamndLineRunner)

**CourseJdbcRepository.java**

**package** com.supr.springboot.learn\_jpa\_and\_hibernate.course.jdbc;

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

**import** org.springframework.jdbc.core.JdbcTemplate;

**import** org.springframework.stereotype.Repository;

@Repository

**public** **class** CourseJdbcRepository {

@Autowired

**private** JdbcTemplate springJdbcTemplate;

**private** **static** String *INSERT\_QUERY* =

"""

insert into course (id, name, author)

values(1, 'aws','sup');

""";

**public** **void** insert() {

springJdbcTemplate.update(*INSERT\_QUERY*);

}

}

**CourseJdbcComamndLineRunner.java**

**package** com.supr.springboot.learn\_jpa\_and\_hibernate.course.jdbc;

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

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

**import** org.springframework.stereotype.Component;

@Component

**public** **class** CourseJdbcComamndLineRunner **implements** CommandLineRunner {

@Autowired

**private** CourseJdbcRepository repository;

@Override

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

repository.insert();

}

}

**Schema.sql**

create table course

(

    id bigint not null,

    name varchar(255) not null,

    author varchar(255) not null,

    primary key (id)

);

Run the server and check select \* from course in h2 database.

**->INSERTING AND DELETING DATA USING SPRING JDBC**

**Course.java**

**package** com.supr.springboot.learn\_jpa\_and\_hibernate.course;

**public** **class** Course {

**private** **long** id;

**private** String name;

**private** String author;

//constructor,getter,setter,toString,default constructor

}

**CourseJdbcComamndLineRunner.java**

**package** com.supr.springboot.learn\_jpa\_and\_hibernate.course.jdbc;

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

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

**import** org.springframework.stereotype.Component;

**import** com.sup.springboot.learn\_spring\_boot.Course;

@Component

**public** **class** CourseJdbcComamndLineRunner **implements** CommandLineRunner {

@Autowired

**private** CourseJdbcRepository repository;

@Override

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

repository.insert(**new** Course(3,"react","aaaaa"));

repository.insert(**new** Course(4,"angular","bbbb"));

repository.deleteById(3);

System.***out***.println(repository.findById(4));

}

}

**CourseJdbcRepository.java**

**package** com.supr.springboot.learn\_jpa\_and\_hibernate.course.jdbc;

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

**import** org.springframework.jdbc.core.JdbcTemplate;

**import** org.springframework.stereotype.Repository;

**import** com.sup.springboot.learn\_spring\_boot.Course;

@Repository

**public** **class** CourseJdbcRepository {

@Autowired

**private** JdbcTemplate springJdbcTemplate;

**private** **static** String *INSERT\_QUERY* =

"""

insert into course (id, name, author)

values(?, ?,?);

""";

private static String *DELETE\_QUERY* =

"""

delete from course where id =?

""";

private static String SELECT\_QUERY =

"""

select \* from course where id = ?

""";

public void insert(Course course) {

springJdbcTemplate.update(*INSERT\_QUERY*, course.getId(), course.getName(),course.getAuthor());

}

public void deleteById(long id) {

springJdbcTemplate.update(*DELETE\_QUERY*,id);

}

**public** Course findById(**long** id) {

**return** springJdbcTemplate.queryForObject(*SELECT\_QUERY*, **new** BeanPropertyRowMapper<>(Course.**class**),id);

//ResultSet->Bean => Row Mapper

}

}

**Doing the above thing with JPA**

**Courses.java**

**package** com.supr.springboot.learn\_jpa\_and\_hibernate.course;

**import** jakarta.persistence.Column;

**import** jakarta.persistence.Entity;

**import** jakarta.persistence.Id;

@Entity(name="Courses")

**public** **class** Courses {

@Id

**private** **long** id;

@Column(name="name")

**private** String name;

@Column(name="author")

**private** String author;

//constructor,getters,toString,setters

**public** Courses() {

}

**public** Courses(**long** id, String name, String author) {

**super**();

**this**.id = id;

**this**.name = name;

**this**.author = author;

}

**public** **long** getId() {

**return** id;

}

**public** String getName() {

**return** name;

}

**public** String getAuthor() {

**return** author;

}

**public** **void** setId(**long** id) {

**this**.id = id;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **void** setAuthor(String author) {

**this**.author = author;

}

@Override

**public** String toString() {

**return** "Course [id=" + id + ", name=" + name + ", author=" + author + "]";

}

}

**CourseJpaCommandLineRunner.java**

**package** com.supr.springboot.learn\_jpa\_and\_hibernate.course.jpa;

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

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

**import** org.springframework.stereotype.Component

**import** com.supr.springboot.learn\_jpa\_and\_hibernate.course.Courses;

@Component

**public** **class** CourseJpaCommandLineRunner **implements** CommandLineRunner{

@Autowired

**private** CourseJpaRepository repository;

@Override

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

repository.insert(**new** Courses(3,"react","aaaaa"));

repository.insert(**new** Courses(4,"angular","bbbb"));

repository.deleteById(3);

System.***out***.println(repository.findById(4));

}

}

**CourseJpaRepository.java**

**package** com.supr.springboot.learn\_jpa\_and\_hibernate.course.jpa;

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

**import** org.springframework.stereotype.Repository;

**import** com.supr.springboot.learn\_jpa\_and\_hibernate.course.Courses;

**import** jakarta.persistence.EntityManager;

**import** jakarta.persistence.PersistenceContext;

**import** jakarta.transaction.Transactional;

@Repository

@Transactional

**public** **class** CourseJpaRepository {

//@Autowired (instaedb of autowire we can use PersistenceContext

@PersistenceContext

**private** EntityManager entityManager;

**public** **void** insert(Courses courses) {

entityManager.merge(courses);

}

**public** Courses findById(**long** id) {

**return** entityManager.find(Courses.**class**, id);

}

**public** **void** deleteById(**long** id) {

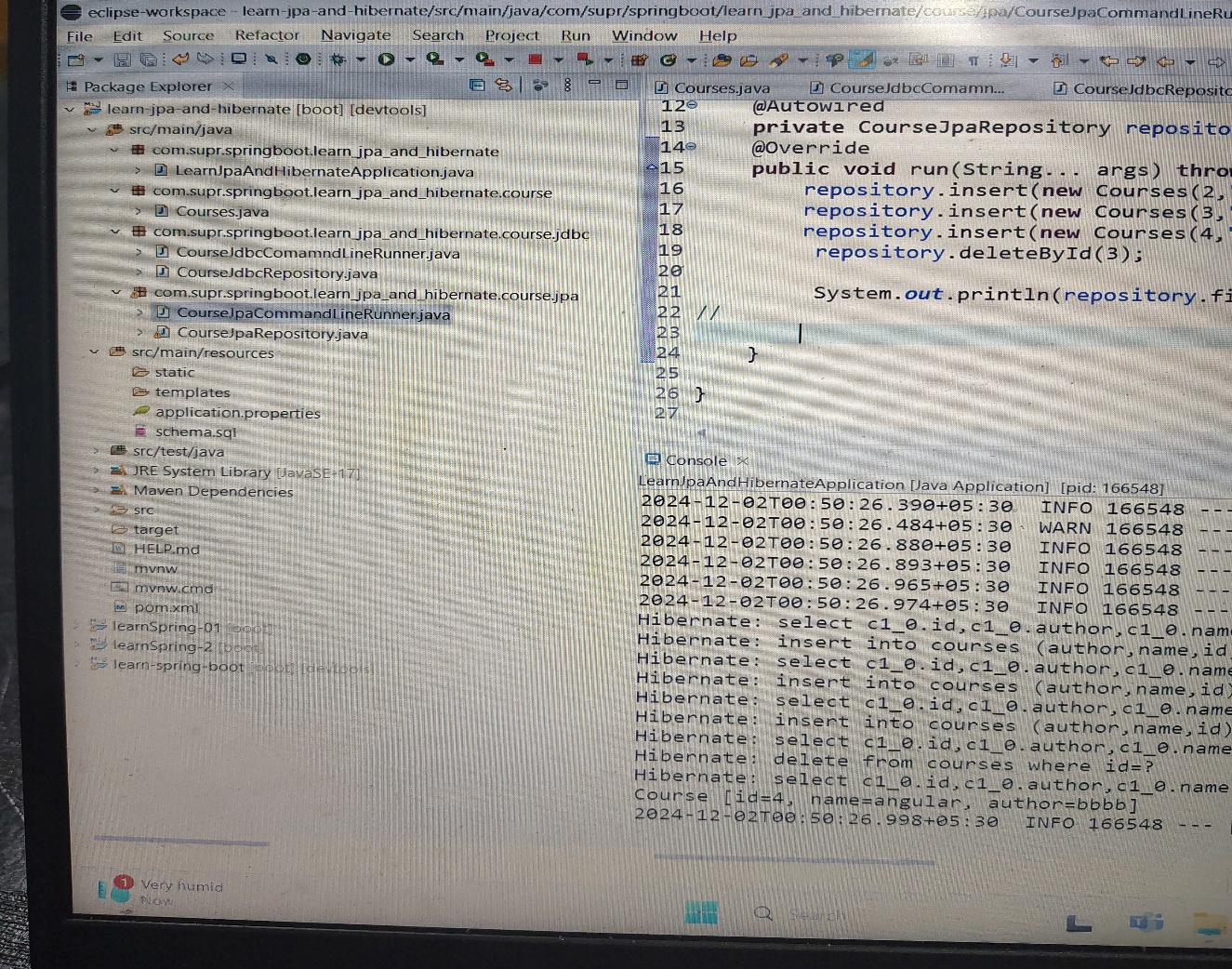
Courses courses=entityManager.find(Courses.**class**, id);

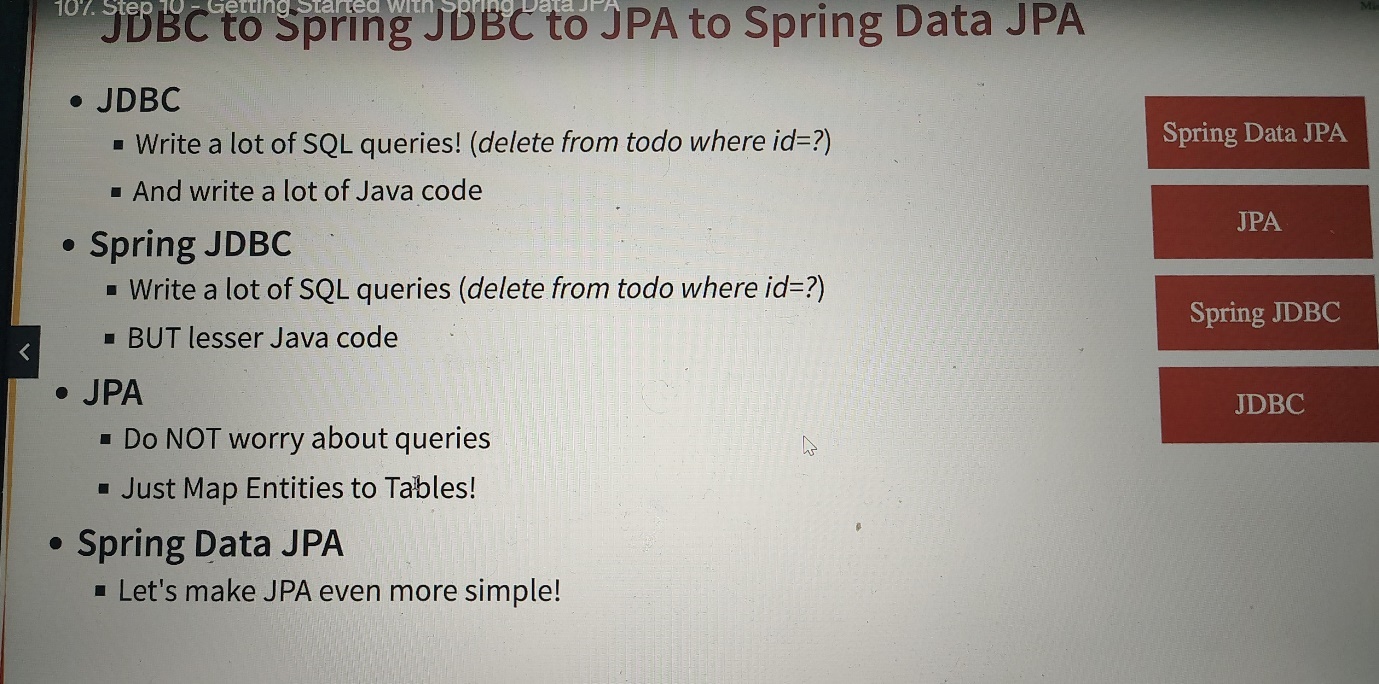
entityManager.remove(courses);

}

}

Note: to see the sql queries generated by the JPA , write the following code in application.properties: (spring.jpa.show-sql=true)





**SPRING DATA JPA**

**Courses.java**

**package** com.supr.springboot.learn\_jpa\_and\_hibernate.course;

**import** jakarta.persistence.Column;

**import** jakarta.persistence.Entity;

**import** jakarta.persistence.Id;

@Entity(name="Courses")

**public** **class** Courses {

@Id

**private** **long** id;

@Column(name="name")

**private** String name;

@Column(name="author")

**private** String author;

//constructor,getters,toString,setters

**public** Courses() {

}

**public** Courses(**long** id, String name, String author) {

**super**();

**this**.id = id;

**this**.name = name;

**this**.author = author;

}

**public** **long** getId() {

**return** id;

}

**public** String getName() {

**return** name;

}

**public** String getAuthor() {

**return** author;

}

**public** **void** setId(**long** id) {

**this**.id = id;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **void** setAuthor(String author) {

**this**.author = author;

}

@Override

**public** String toString() {

**return** "Course [id=" + id + ", name=" + name + ", author=" + author + "]";

}

}

**CourseSpringDataJpaRepository.java**

**package** com.supr.springboot.learn\_jpa\_and\_hibernate.course.springdatajpa;

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

**import** com.supr.springboot.learn\_jpa\_and\_hibernate.course.Courses;

**public** **interface** CourseSpringDataJpaRepository **extends** JpaRepository<Courses,Long> {

//<Entity class,Type of id>

}

**CourseSpringDataJpaCommandLineRunner.java**

**package** com.supr.springboot.learn\_jpa\_and\_hibernate.course.springdatajpa;

//

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

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

**import** org.springframework.stereotype.Component;

**import** com.supr.springboot.learn\_jpa\_and\_hibernate.course.Courses;

@Component

**public** **class** CourseSpringDataJpaCommandLineRunner **implements** CommandLineRunner{

@Autowired

**private** CourseSpringDataJpaRepository repository;

@Override

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

repository.save(**new** Courses(2,"react2","aaaaa2"));

repository.save(**new** Courses(3,"react","aaaaa"));

repository.save(**new** Courses(4,"angular","bbbb"));

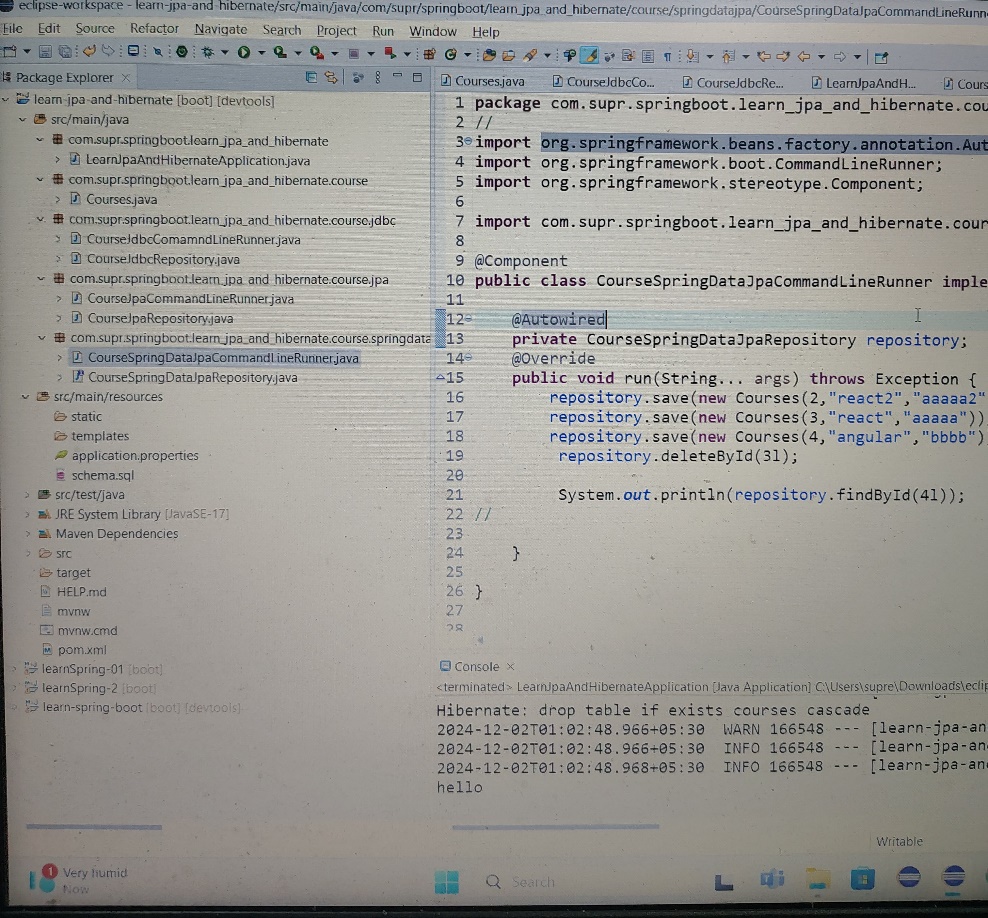
repository.deleteById(3l);

System.***out***.println(repository.findById(4l));

//

}

}



**Hibernate vs JPA**

* JPA defines the specification. It is an API.
* Hibernate is one of the popular implementations of JPA