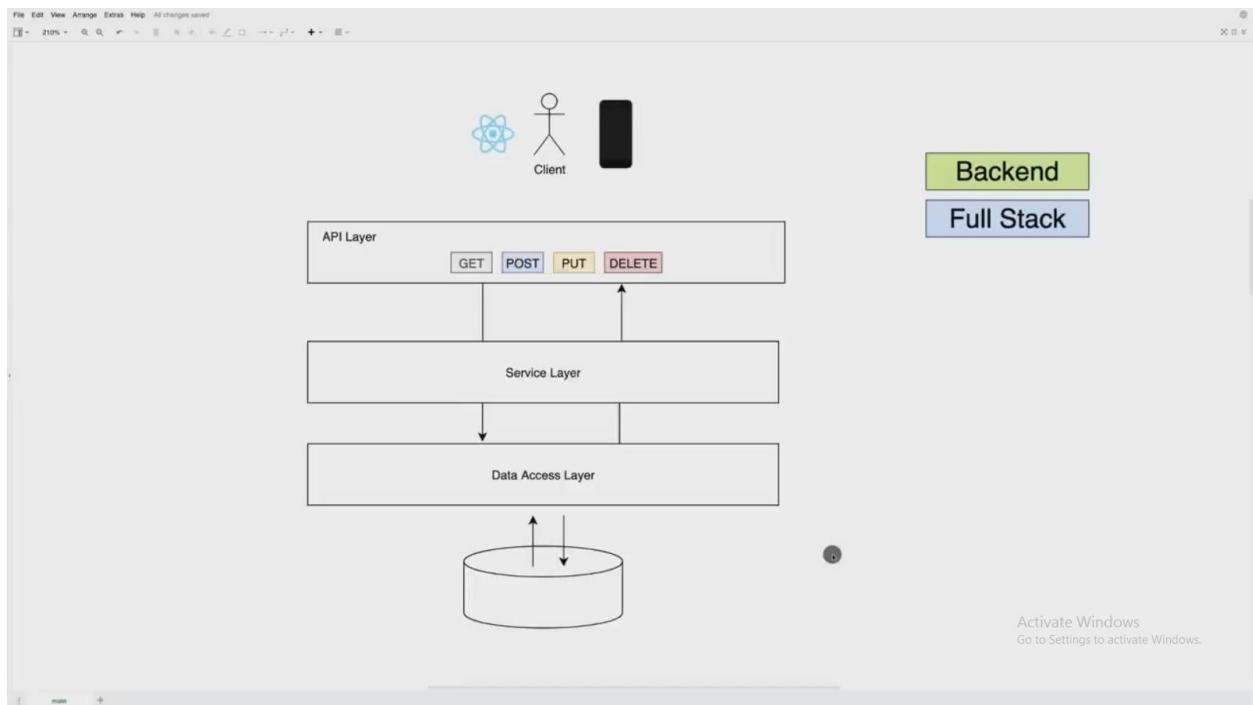
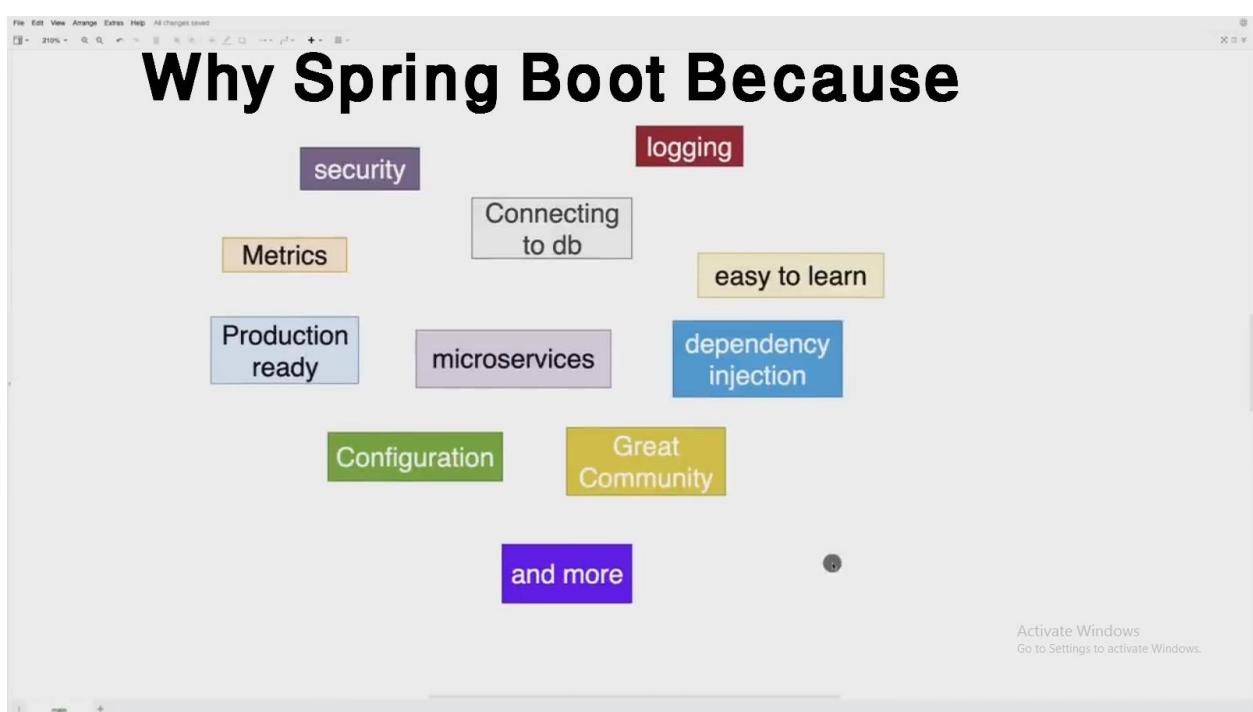


SPRING BOOT FULL COURSE

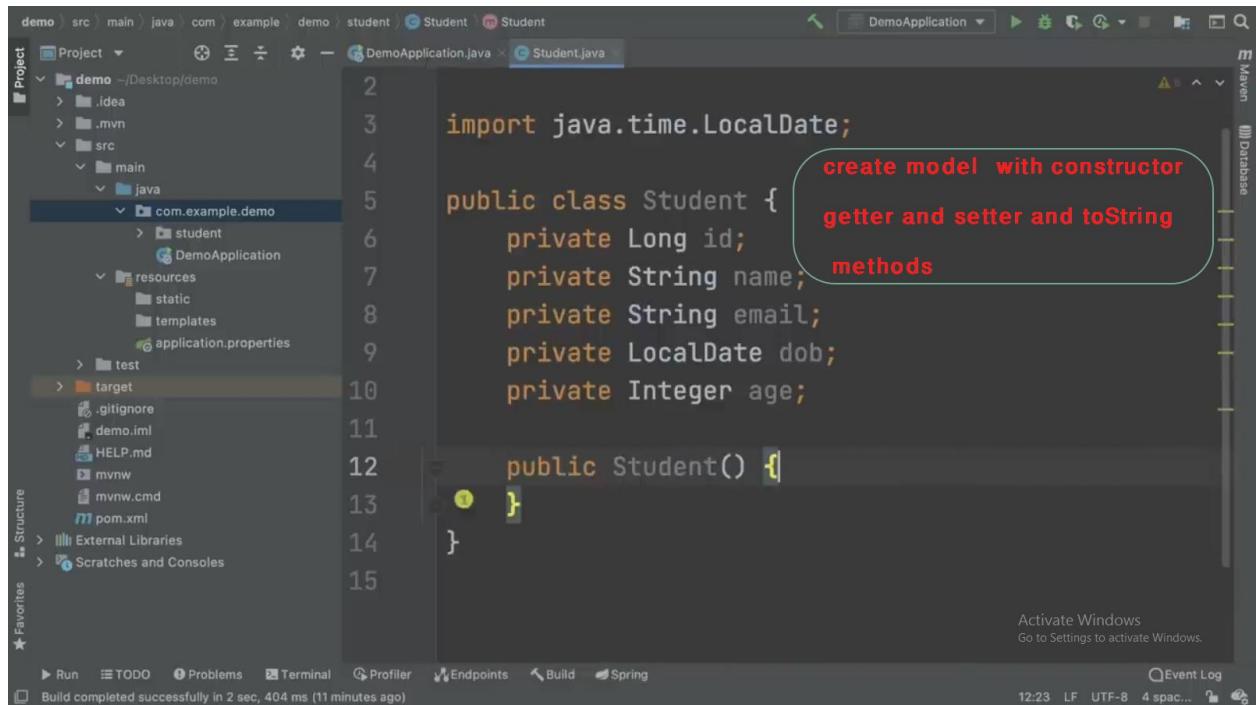


Activate Windows
Go to Settings to activate Windows.



The screenshot shows the Spring Initializr web interface. On the left, there's a sidebar with three horizontal lines and icons for GitHub and Twitter. The main area has a header with the Spring Initializr logo and a sun/moon icon. It includes sections for Project (Maven Project selected), Language (Java selected), Dependencies (empty), and Spring Boot (2.4.1 selected). Below these are Project Metadata fields for Group (com.example) and three buttons: GENERATE (⌘ + ↩), EXPLORE (CTRL + SPACE), and SHARE... A note says "No dependency selected". At the bottom right, it says "Activate Windows Go to Settings to activate Windows".

This screenshot is similar to the first one but features a large, bold, black overlay text in the center reading "In there select the dependency". The rest of the interface, including the sidebar, project configuration, and dependency section, remains visible.



demo > src > main > java > com > example > demo > student > Student.java

```
import java.time.LocalDate;

public class Student {
    private Long id;
    private String name;
    private String email;
    private LocalDate dob;
    private Integer age;

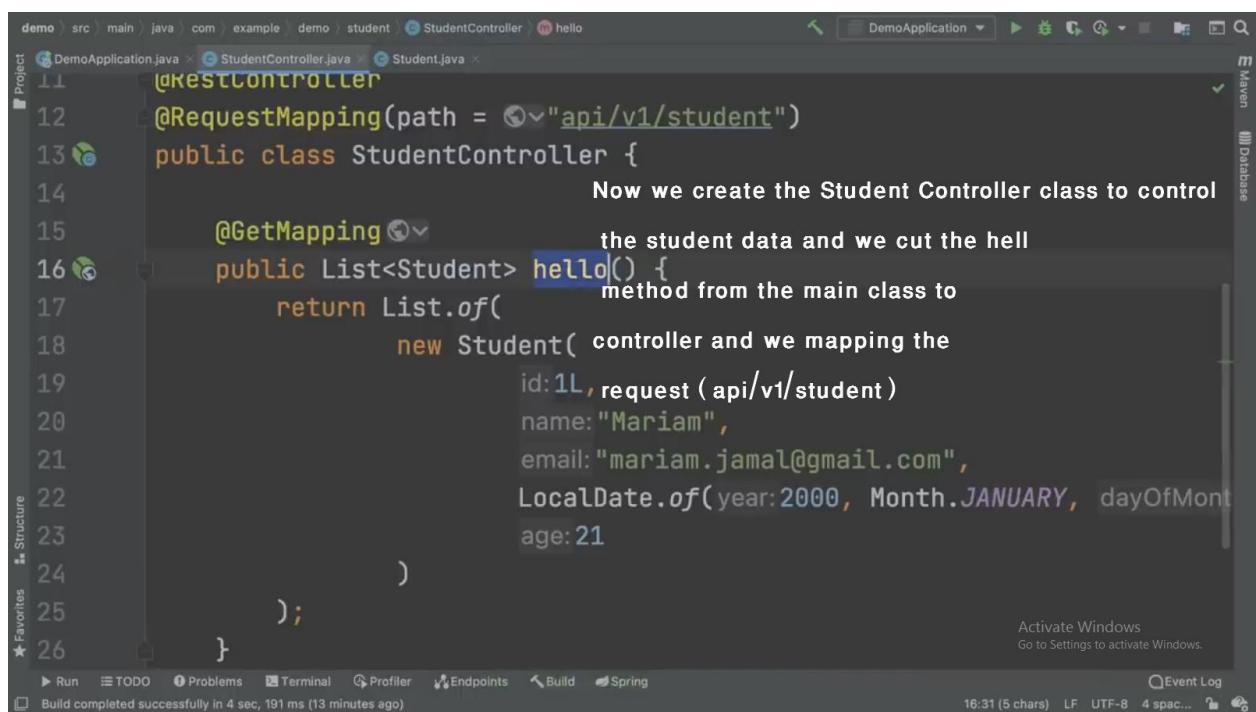
    public Student() {
    }

    }
```

create model with constructor
getter and setter and toString
methods

Activate Windows
Go to Settings to activate Windows.

12:23 LF UTF-8 4 spac... Event Log



demo > src > main > java > com > example > demo > student > StudentController.java

```
@RequestMapping(path = "/api/v1/student")
public class StudentController {

    @GetMapping
    public List<Student> hello() {
        return List.of(
            new Student(
                id: 1L,
                request(api/v1/student),
                name: "Mariam",
                email: "mariam.jamal@gmail.com",
                LocalDate.of(year: 2000, Month.JANUARY, dayOfMonth: 1),
                age: 21
            )
        );
    }
}
```

Now we create the Student Controller class to control
the student data and we cut the hell
method from the main class to
controller and we mapping the
id: 1L, request(api/v1/student)
name: "Mariam",
email: "mariam.jamal@gmail.com",
LocalDate.of(year: 2000, Month.JANUARY, dayOfMonth:
age: 21

Activate Windows
Go to Settings to activate Windows.

16:31 (5 chars) LF UTF-8 4 spac... Event Log

```

16
17     public static void main(String[] args) {
18         SpringApplication.run(DemoApplication.class, args);
19
20
21     @GetMapping
22     public List<Student> hello() {
23         return List.of(
24             new Student(      Give the Data inside the Student Constructor
25                 id:1L,
26                 name:"Mariam",
27                 email:"mariam.jamal@gmail.com",
28                 LocalDate.of(year:2000, Month.JANUARY, dayOfMonth:5),
29                 age:21
30             )
31         );

```

Activate Windows
Go to Settings to activate Windows.

28:33 (8 chars) LF UTF-8 Tab Event Log

Build completed successfully in 2 sec, 404 ... (15 minutes ago)

Project

- Maven Project
- Gradle Project

Language

- Java
- Kotlin
- Groovy

Spring Boot

- 2.5.0 (SNAPSHOT)
- 2.4.2 (SNAPSHOT) **2.4.1**
- 2.3.8 (SNAPSHOT) **2.3.7**

Project Metadata

Group	com.example
Artifact	demo
Name	demo
Description	Demo project for Spring Boot

Dependencies

Spring Web WEB **1 Spring Web**
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Data JPA SQL **2 Spring Data JPA**
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

PostgreSQL Driver SQL **3 SQL Driver**
A JDBC and R2DBC driver that allows Java programs to connect to a PostgreSQL database using standard, database independent Java code.

Activate Windows
Go to Settings to activate Windows.

GENERATE ⌘ + ↵ **EXPLORE** CTRL + SPACE **SHARE...**



```
demo > src > main > java > com > example > demo > student > StudentController.java > studentService.java
Project DemoApplication.java > StudentController.java > StudentService.java > Student.java
11 @RequestMapping(path = "/api/v1/student")
12 public class StudentController {
13
14     private final StudentService studentService; service class
15
16     @Autowired
17     public StudentController(StudentService studentService) { injected to here
18         this.studentService = studentService;
19     }
20
21     @GetMapping
22     public List<Student> getStudents() {
23         return studentService.getStudents(); get data from the service look it in next page
24     }
25
26 }
```

demo > src > main > java > com > example > demo > student > StudentService.java

```

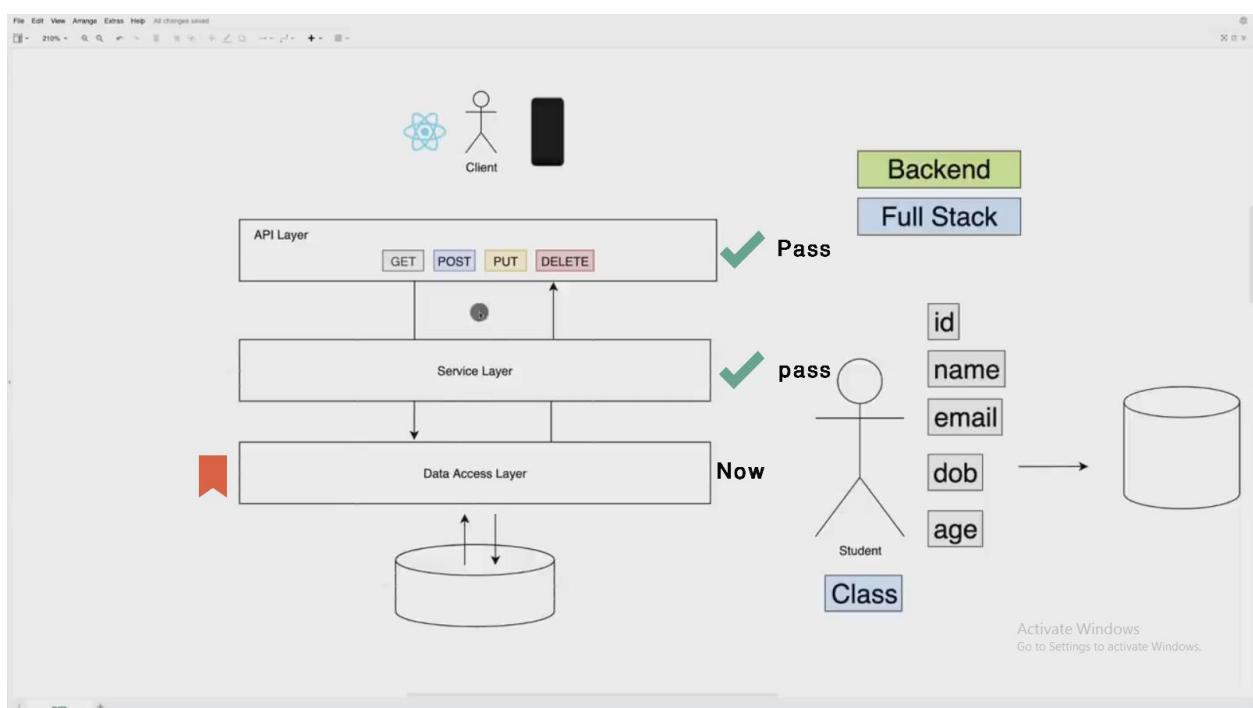
6 import java.time.Month;
7 import java.util.List;
8
9 @Component @Service It is very sensible Annotation
10 public class StudentService {
11
12     public List<Student> getStudents() {
13         return List.of(
14             new Student(
15                 id: 1L,
16                 name: "Mariam",
17                 email: "mariam.jamal@gmail.com",
18                 LocalDate.of(year: 2000, Month.JANUARY, dayOfMonth:
19                 age: 21
20             )
21         )
22     }
23 }

```

Activate Windows
Go to Settings to activate Windows.

Run TODO Problems Terminal Profiler Endpoints Build Spring Event Log

Build completed successfully in 2 sec, 558 ... (9 minutes ago) 9:11 LF UTF-8 4 spac...



The screenshot shows the IntelliJ IDEA interface with the pom.xml file open. A callout box highlights the following code snippet:

```
<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>org.postgresql</groupId>
        <artifactId>postgresql</artifactId>
        <scope>runtime</scope>
    </dependency>
```

A tooltip "add this dependency" points to the first dependency block. The code editor has syntax highlighting and line numbers from 21 to 34.



A screenshot of a Java IDE (IntelliJ IDEA) showing the contents of the `application.properties` file. The file contains the following configuration:

```
spring.datasource.url=jdbc:postgresql://localhost:5432/student
spring.datasource.username=
spring.datasource.password=
spring.jpa.hibernate.ddl-auto=create-drop This is actually means clean the application
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.properties.hibernate.format_sql=true
```

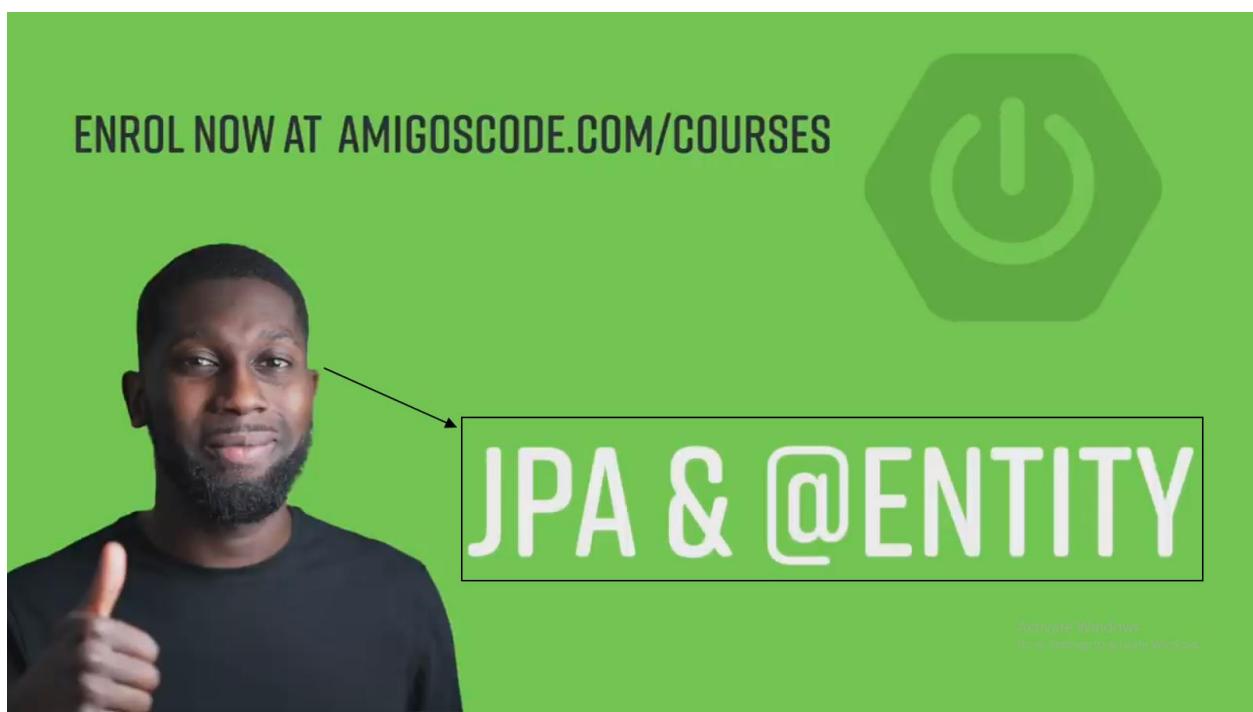
The line `spring.jpa.hibernate.ddl-auto=create-drop` is highlighted with a yellow box, and a tooltip says "This is actually means clean the application". The IDE interface includes tabs for `DemoApplication.java`, `StudentController.java`, `application.properties`, `StudentService.java`, and `Student.java`. The bottom status bar shows "Build completed successfully in 2 sec, 579 ... (9 minutes ago)".



The screenshot shows the IntelliJ IDEA interface with the pom.xml file open. A callout box highlights the following code snippet:

```
<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>org.postgresql</groupId>
        <artifactId>postgresql</artifactId>
        <scope>runtime</scope>
    </dependency>
```

A tooltip "add this dependency" points to the first dependency block. The status bar at the bottom shows "Undo Comment with Line Comment via %Z (Ctrl+Z for Win/Linux)".



The screenshot shows the IntelliJ IDEA interface with the code editor open to the `Student.java` file. The code defines a JPA entity named `Student` with annotations for `@Entity`, `@Table`, `@Id`, and `@SequenceGenerator`. A callout bubble points to the `@SequenceGenerator` annotation with the text "Now Map the Student class to DB". The code editor also shows several other files like `Application.java`, `StudentController.java`, `application.properties`, and `pom.xml`.

```
import java.time.LocalDate;

@Entity
@Table
public class Student {
    @Id
    @SequenceGenerator(
        name = "student_sequence",
        sequenceName = "student_sequence",
        allocationSize = 1
    )
    private Long id;
    private String name;
    private String email;
    private LocalDate dob;
    private Integer age;
}
```



```
demo > src > main > java > com > example > demo > student > StudentRepository.java
1 package com.example.demo.student;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4
5 @Repository
6 public interface StudentRepository
7     extends JpaRepository<Student, Long> { The ID type which is -Long-
8 }
9
10 Make interface of repository
```

Activate Windows
Go to Settings to activate Windows.

```
demo > src > main > java > com > example > demo > student > StudentRepository.java
1 package com.example.demo.student;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4 import org.springframework.stereotype.Repository;
5
6 @Repository Add this annotation
7 @Repository
8 public interface StudentRepository
9     extends JpaRepository<Student, Long> {
10 }
```

Activate Windows
Go to Settings to activate Windows.

A screenshot of a Java IDE (IntelliJ IDEA) showing the code for a `StudentService` class. The code uses Spring's `@Service` and `@Autowired` annotations. It injects a `StudentRepository` into the service and returns a list of students from the repository.

```
10  @Service
11  public class StudentService {
12
13      private final StudentRepository studentRepository;
14
15      @Autowired
16      public StudentService(StudentRepository studentRepository) {
17          this.studentRepository = studentRepository;
18      }
19
20      public List<Student> getStudents() {
21          return studentRepository.findAll(); // Now we take data from DB
22      }
23  }
```



The screenshot shows the IntelliJ IDEA interface with the code editor open. The code defines a `StudentConfig` class with a `@Bean` annotation for a `CommandLineRunner`. The runner creates two `Student` objects: `Mariam` (born in 2000) and `Alex` (born in 2004). Both students have an age of 21.

```
public class StudentConfig {
    @Bean
    CommandLineRunner commandLineRunner(
            StudentRepository repository) {
        return args -> {
            new Student(
                    name: "Mariam",
                    email: "mariam.jamal@gmail.com",
                    LocalDate.of(year: 2000, JANUARY, dayOfMonth: 5),
                    age: 21
            );
        };
    }
}
```

The screenshot shows the completed code for the `StudentConfig` class. It includes the creation of `Mariam` and `Alex` students and their saving into the database via the `repository.saveAll()` method. A callout box at the bottom right instructs the user to run the application to store data in the database.

```
LocalDate.of(year: 2000, JANUARY, dayOfMonth: 5),
age: 21
);

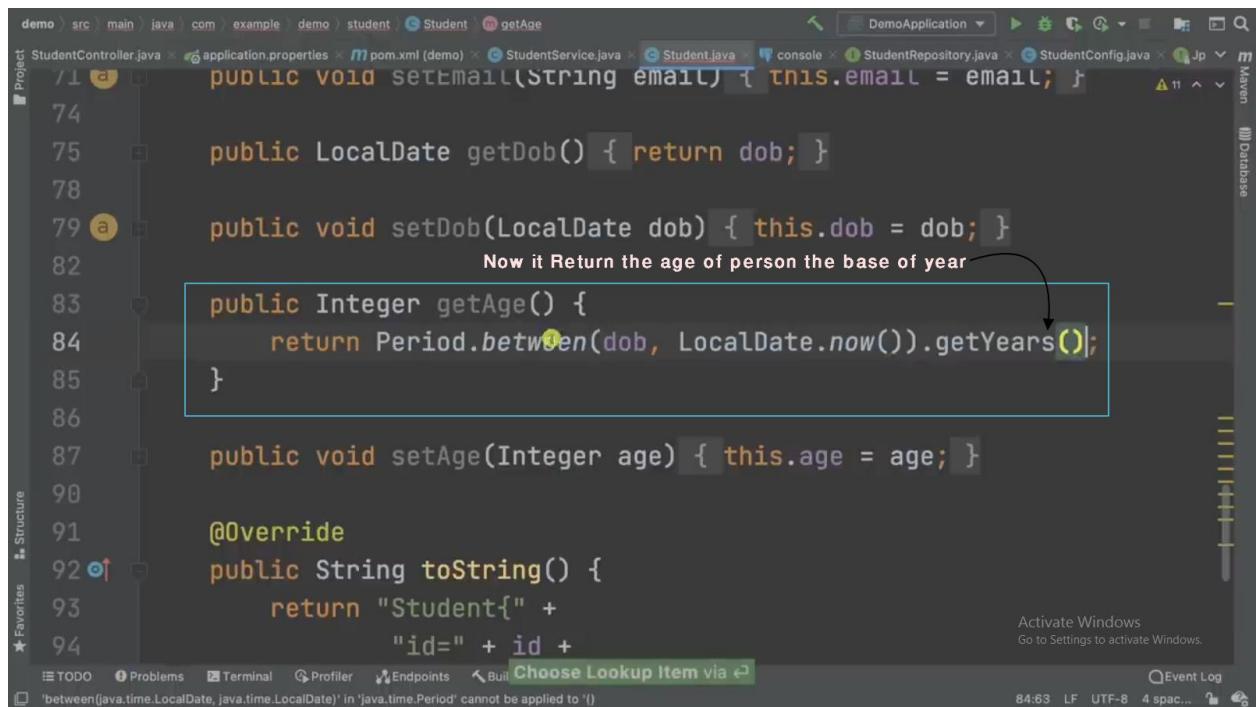
Student alex = new Student(
        name: "Alex",
        email: "alex@gmail.com",
        LocalDate.of(year: 2004, JANUARY, dayOfMonth: 5),
        age: 21
);
repository.saveAll(
        List.of(mariam, alex)
);
```

Now Run the Application to stor data inside the DB: Windows
Go to Settings to activate Windows.

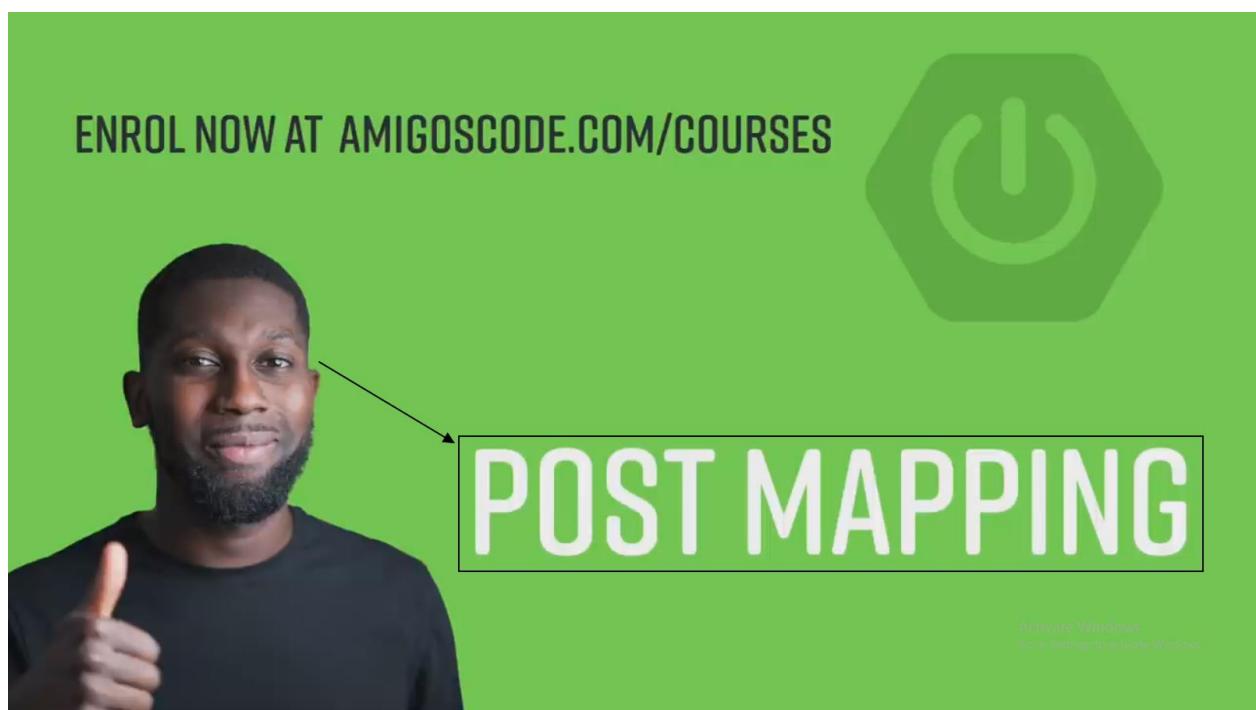
A screenshot of a Java IDE (IntelliJ IDEA) showing the code for a `Student` class. The code includes annotations for `id`, `name`, `email`, and `dob`. A tooltip is displayed over the `@Transient` annotation on the `age` field, stating: "remove the age from constructor it calculate by the dob(LocalDate) base". The IDE interface shows various tabs like `application.properties`, `pom.xml`, `StudentService.java`, `Student.java`, `console`, `StudentRepository.java`, `StudentConfig.java`, and `JpaRepository.class`.

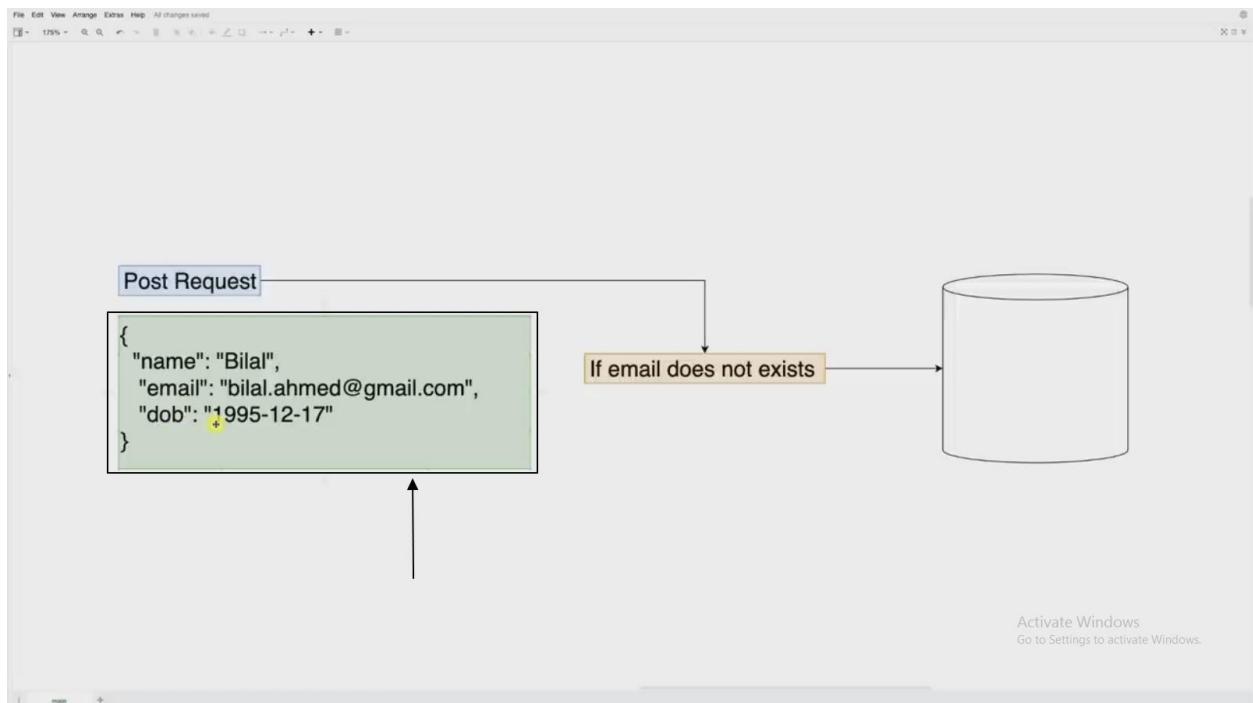
```
14     allocationSize = 1
15   )
16   @GeneratedValue(
17     strategy = GenerationType.SEQUENCE,
18     generator = "student_sequence"
19   )
20   private Long id;
21   private String name;
22   private String email;
23   private LocalDate dob;
24   @Transient
25   private Integer age;
26
27   public Student() {
28 }
```





```
demo > src > main > java > com > example > demo > student > Student > getAge  
StudentController.java x application.properties x pom.xml (demo) x StudentService.java x Student.java x console x StudentRepository.java x StudentConfig.java x Jp v m  
Project 71 PUBLIC VOID setEmail(String email) { this.email = email; }  
74  
75     public LocalDate getDob() { return dob; }  
76  
77     public void setDob(LocalDate dob) { this.dob = dob; }  
78             Now it Return the age of person the base of year  
79     a     public Integer getAge() {  
80         return Period.between(dob, LocalDate.now()).getYears();  
81     }  
82  
83     public void setAge(Integer age) { this.age = age; }  
84  
85     @Override  
86     public String toString() {  
87         return "Student{" +  
88             "id=" + id +  
89             "name=" + name +  
90             "email=" + email +  
91             "dob=" + dob +  
92             "age=" + age +  
93         }  
94  
Activate Windows  
Go to Settings to activate Windows.  
Choose Lookup Item via ↗  
'between[java.time.LocalDate, java.time.LocalDate]': in 'java.time.Period' cannot be applied to '()'
```





Screenshot of an IDE showing the `StudentController.java` file. The code defines two methods:

```

    @Autowired
    public StudentController(StudentService studentService) {
        this.studentService = studentService;
    }

    @GetMapping
    public List<Student> getStudents() {
        return studentService.getStudents();
    }

    @PostMapping
    PostMapping often use to inser data into db
    public void registerNewStudent(Student student) {
        studentService.addNewStudent(student);
    }

```

A callout box highlights the `@PostMapping` annotation with the note: "PostMapping often use to inser data into db". Another callout box highlights the `studentService.addNewStudent(student);` line with the note: "Create this method inside the StudentService Class".

A screenshot of a Java Spring Boot application in an IDE. The code in `StudentController.java` is as follows:

```
14 15 @Autowired
16     public StudentController(StudentService studentService) {
17         this.studentService = studentService;
18     }
19
20     @GetMapping
21     public List<Student> getStudents() {
22         return studentService.getStudents();
23     }
24
25     @PostMapping
26     public void registerNewStudent(@RequestBody Student student) {
27         studentService.addNewStudent(student);
28     }
29 }
```

A callout bubble points to the `@RequestBody` annotation on the `registerNewStudent` method, with the text "This means the request body mapping right here".

IDE interface elements include: Project, Favorites, Structure, TODO, Problems, Terminal, Profiler, Endpoints, Build, Spring, Event Log, and Build completed successfully in 5 sec, 161... (47 minutes ago).



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

import java.util.Optional;

@Repository
public interface StudentRepository
    extends JpaRepository<Student, Long> {

    //    SELECT * FROM student WHERE email = ?
    Optional<Student> findStudentByEmail(String email);
```

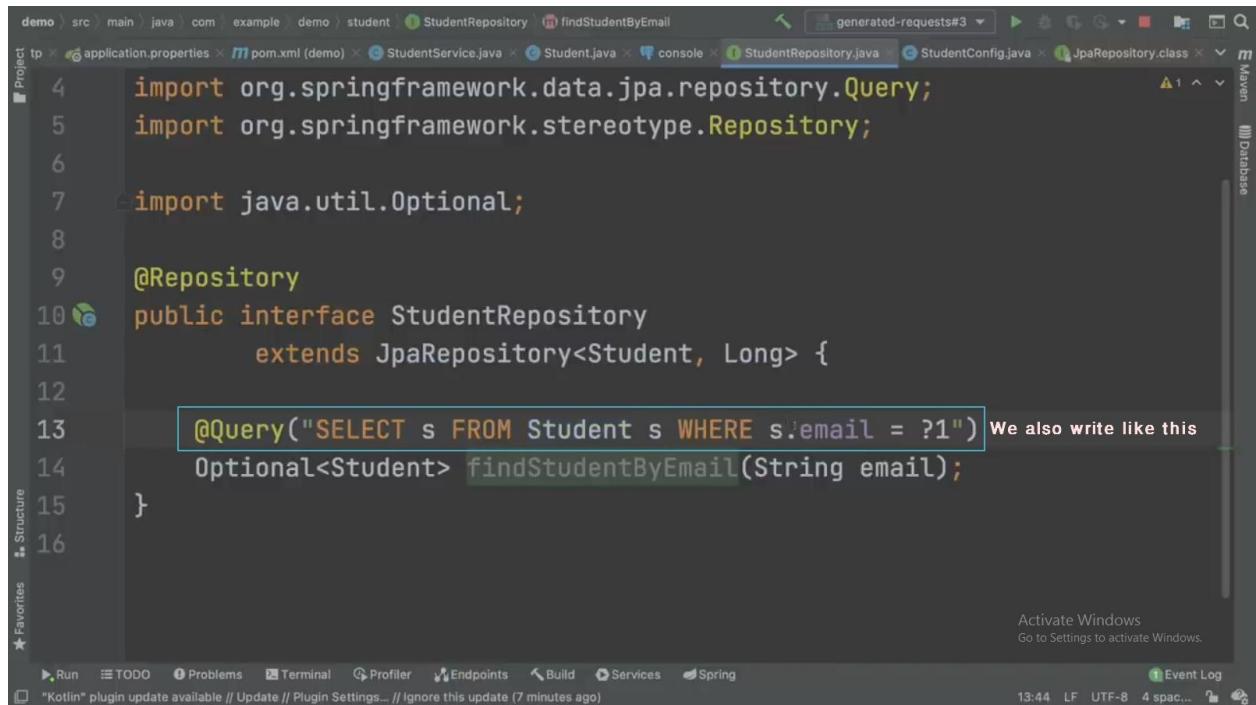
This mean is this
Custom method define in JpaRepository interface

```
@Autowired
public StudentService(StudentRepository studentRepository) { this.studentRepository = studentRepository; }

public List<Student> getStudents() { return studentRepository.findAll(); }

public void addNewStudent(Student student) {
    Optional<Student> studentOptional = studentRepository
        .findStudentByEmail(student.getEmail());
    if (studentOptional.isPresent()) {
        throw new IllegalStateException("email taken");
    }
}
```

get student by Email
We call the custom method in repository in here



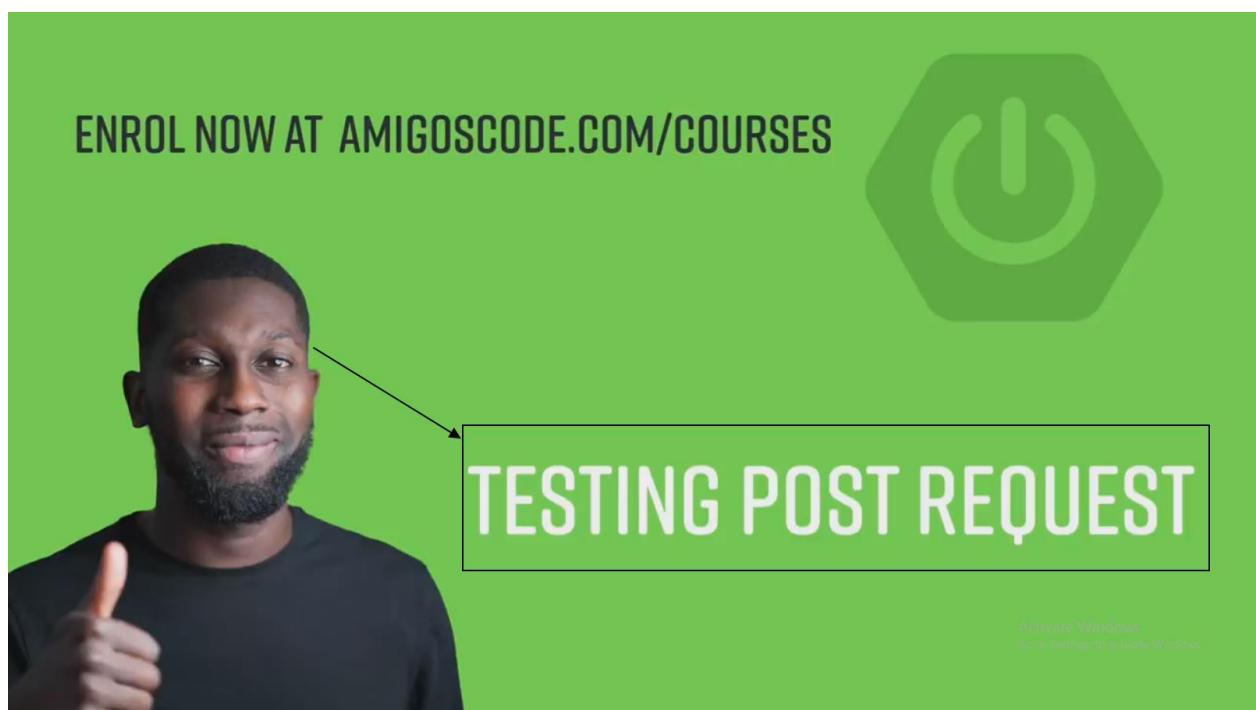
A screenshot of an IDE (IntelliJ IDEA) showing Java code for a Spring Data JPA repository. The code defines a `StudentRepository` interface extending `JpaRepository<Student, Long>`. It includes a method `findStudentByEmail` annotated with `@Query` and `Optional<Student>` return type. A tooltip suggests an alternative syntax: "We also write like this". The IDE interface shows various tabs like `application.properties`, `pom.xml`, `StudentService.java`, etc., and a bottom status bar with "Activate Windows" and "Event Log".

```
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;

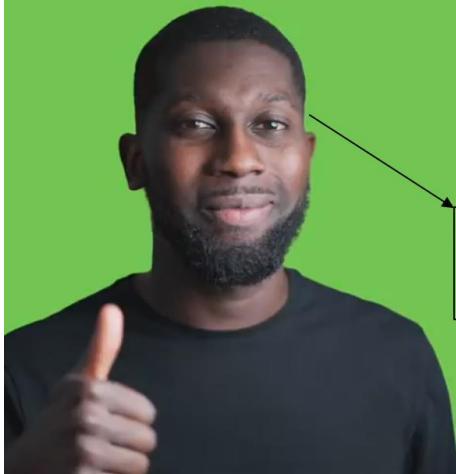
import java.util.Optional;

@Repository
public interface StudentRepository
    extends JpaRepository<Student, Long> {

    @Query("SELECT s FROM Student s WHERE s.email = ?1")
    Optional<Student> findStudentByEmail(String email);
}
```



ENROL NOW AT AMIGOSCODE.COM/COURSES



DELETING STUDENTS

Activate Windows
Go to Settings to activate Windows.

```
demo > src > main > java > com > example > demo > student > StudentController.java > deleteStudent
Project StudentController.java generated-requests.http application.properties pom.xml (demo) StudentService.java Student.java console StudentRepository.java maven Database
21     return studentService.getStudents();
22 }
23
24 @PostMapping
25 public void registerNewStudent(@RequestBody Student student) {
26     studentService.addNewStudent(student);
27 }
28
29 @DeleteMapping(path = "/{studentId}")
30 public void deleteStudent(@PathVariable("studentId") Long id) {
31     studentService.deleteStudent(id);
32 }
```

Match this student id to the below student id

Create deleteStudent Method inside the Service

Activate Windows
Go to Settings to activate Windows.

The screenshot shows the IntelliJ IDEA interface with the code editor open. The code is part of the `StudentService` class:

```
        .findStudentByEmail(student.getEmail());
    if (studentOptional.isPresent()) {
        throw new IllegalStateException("email taken");
    }
    studentRepository.save(student);
}

public void deleteStudent(Long studentId) {
    boolean exists = studentRepository.existsById(studentId);
    if (!exists) {
        throw new IllegalStateException(
            "student with id " + studentId + " does not exists"
        );
    }
    studentRepository.deleteById(studentId);
}
```

Annotations and arrows highlight specific parts of the code:

- An arrow points from the text "If student Exist" to the condition `exists`.
- An arrow points from the text "In Service class just call the deleteById method" to the line `studentRepository.deleteById(studentId);`.

The status bar at the bottom right shows "Activate Windows Go to Settings to activate Windows."

The screenshot shows the IntelliJ IDEA interface with the code editor open. The code is part of the `generated-requests.http` file:

```
<> 2021-01-07T001455.500.json
<> 2021-01-07T001331.500.json
###
DELETE http://localhost:8080/api/v1/student/1
```

Annotations and arrows highlight specific parts of the code:

- An annotation "This is the PathVariable ID" points to the path variable `1` in the URL.

The status bar at the bottom right shows "Activate Windows Go to Settings to activate Windows."

ENROL NOW AT AMIGOSCODE.COM/COURSES



EXERCISE

Activate Windows
Go to Settings to activate Windows

ENROL NOW AT AMIGOSCODE.COM/COURSES



SOLUTION

Activate Windows
Go to Settings to activate Windows

The screenshot shows a Java code editor with the following code:

```
44     @Transactional Use this annotation
45     public void updateStudent(Long studentId,
46                               String name,           If student not exist
47                               String email) {
48         Student student = studentRepository.findById(studentId)
49             .orElseThrow(() -> new IllegalStateException(
50                 "student with id " + studentId + " does not exist"
51             )
52             if (name != null &&
53                 name.length() > 0 &&
54                 !Objects.equals(student.getName(), name)) {
55                 student.setName(name);
56             }
57         )
```

Annotations and comments in the code:

- @Transactional**: Use this annotation
- If student not exist: A callout points to the `orElseThrow` method.
- same do for Email: A callout points to the `name.length()` check.

IDE interface elements:

- Project: StudentController.java, generated-requests.http, application.properties, pom.xml (demo), StudentService.java, Student.java, console, StudentRepository.java
- Toolbars: Run, Debug, TODO, Problems, Terminal, Profiler, Endpoints, Build, Services, Spring
- Status bar: DemoApplication: Failed to retrieve application JMX service U... (3 minutes ago)
- Bottom right: Activate Windows, Go to Settings to activate Windows, Event Log

The screenshot shows an IDE with an integrated terminal and services panel.

Terminal output:

```
4 <> 2020-12-29T224030.200.txt
5
6 #####
7 > PUT http://localhost:8080/api/v1/student/1?name=Maria
8 Content-Type: application/json
```

Services panel:

- HTTP Request: generated-requests#2, generated-requests#3, generated-requests#4
- student@localhost: console, console

Response details:

Date: Thu, 07 Jan 2021 01:04:15 GMT
Keep-Alive: timeout=60
Connection: keep-alive

<Response body is empty>

Response code: 200; Time: 207ms; Content length: 0

IDE interface elements:

- Project: StudentController.java, generated-requests.http, application.properties, pom.xml (demo), StudentService.java, Student.java, console, StudentRepository.java
- Toolbars: Run, Debug, TODO, Problems, Terminal, Profiler, Endpoints, Build, Services, Spring
- Status bar: DemoApplication: Failed to retrieve application JMX service U... (4 minutes ago)
- Bottom right: Activate Windows, Go to Settings to activate Windows, Event Log

ENROL NOW AT AMIGOSCODE.COM/COURSES



TESTING



Activate Windows
Go to Settings to activate Windows

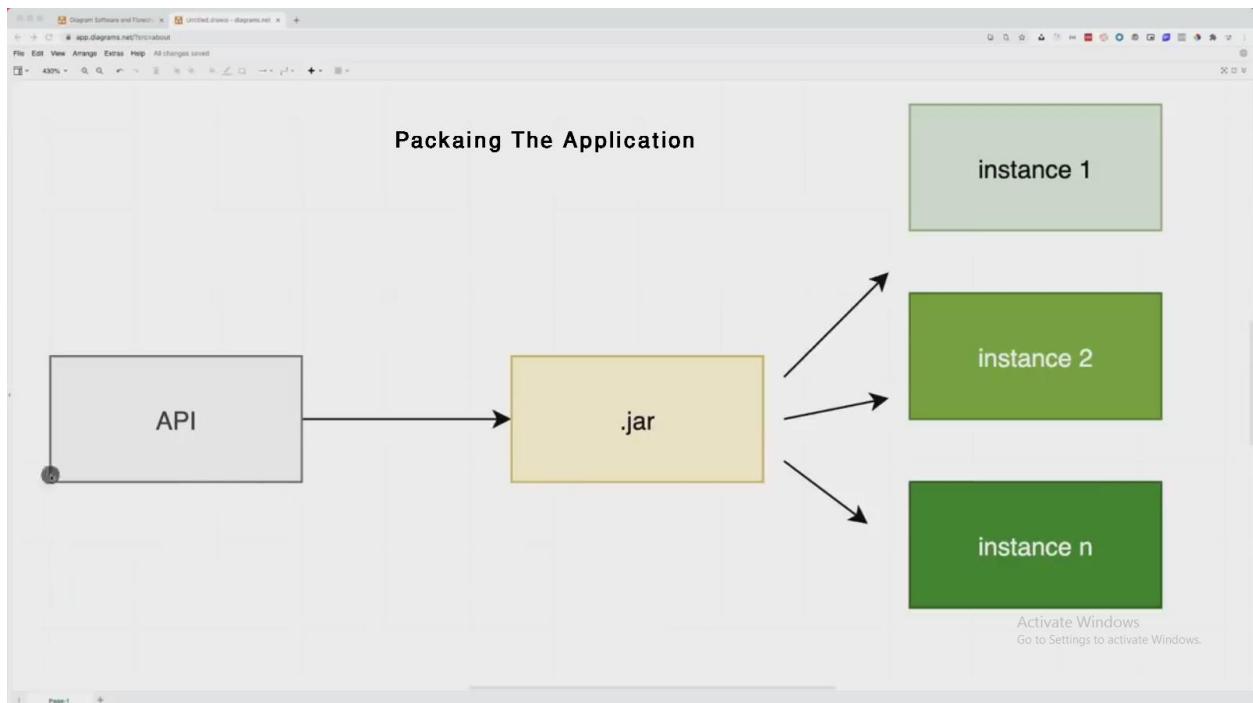
ENROL NOW AT AMIGOSCODE.COM/COURSES



**PACKAGING AND RUNNING
APPLICATION**



Activate Windows
Go to Settings to activate Windows



Open the Maven

Delete this folder

```

public String getSex() {
    return sex;
}

public void setSex(String sex) {
    this.sex = sex;
}

@Override
public String toString() {
    return "PassEmail{" +
        "id=" + id +
        ", email='" + email + '\'' +
        ", password='" + password + '\'' +
        ", sex='" + sex + '\'';
}
  
```

Activate Windows
Go to Settings to activate Windows.

The screenshot shows the IntelliJ IDEA interface. The left sidebar displays the project structure for 'demo2' with files like .idea, .mvn, .gitignore, pom.xml, and src. The main editor shows Java code for 'PassEmail.java' with annotations for getters and setters. The right panel shows the Maven tool window with the 'Lifecycle' section expanded, highlighting the 'clean' and 'install' steps. A tooltip says 'first clean the project' over 'clean' and 'click the install' over 'install'.

```
59     }
60
61     public String getSex() {
62         return sex;
63     }
64
65     public void setSex(String sex) {
66         this.sex = sex;
67     }
68
69     @Override
70     public String toString() {
71         return "PassEmail{" +
72             "id=" + id +
73             ", email='" + email + '\'' +
74             ", password='" + password + '\'' +
75             ", sex='" + sex + '\'' +
76             '}';
77     }
78 }
79
```

