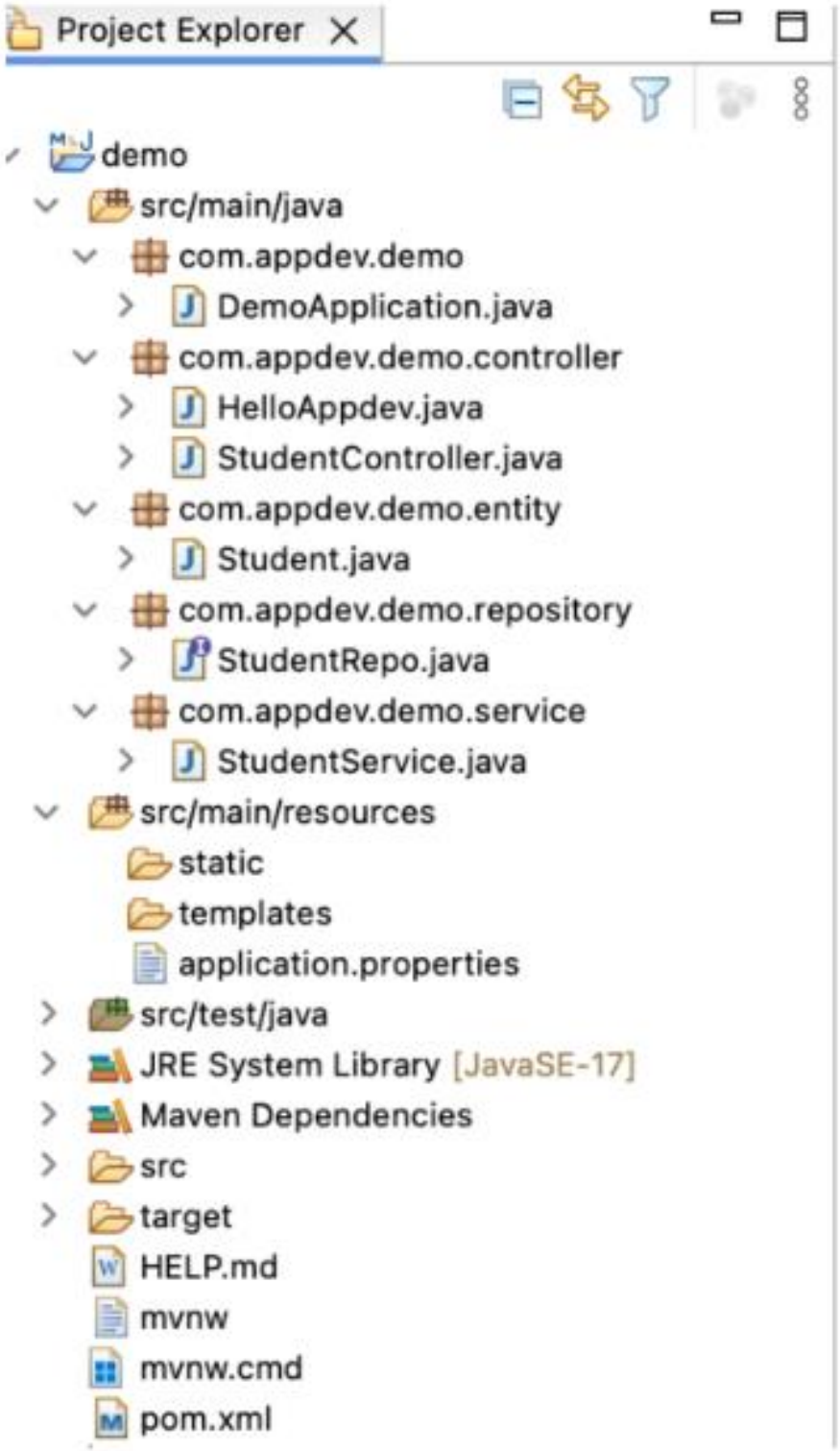
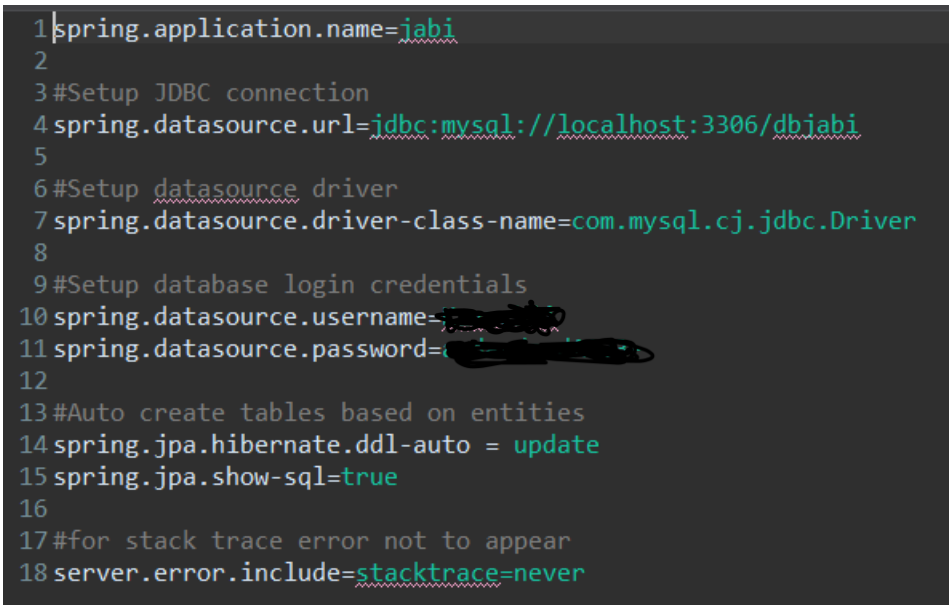

Project Explorer
demo
src/main/java
com.appdev.demo
DemoApplication.java
com.appdev.demo.controller
HelloAppdev.java
StudentController.java
com.appdev.demo.entity
Student.java
com.appdev.demo.repository
StudentRepo.java
com.appdev.demo.service
StudentService.java
src/main/resources
static
templates
application.properties
src/test/java
JRE System Library [JavaSE-17]
Maven Dependencies
src
target
HELP.md
mvnw
mvnw.cmd
pom.xml



```
spring.application.name=jabi

#Setup JDBC connection
spring.datasource.url=jdbc:mysql://localhost:3306/dbjabi

#Setup datasource driver
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

#Setup database login credentials
spring.datasource.username=[redacted]
spring.datasource.password=[redacted]

#Auto create tables based on entities
spring.jpa.hibernate.ddl-auto = update
spring.jpa.show-sql=true

#for stack trace error not to appear
server.error.include=stacktrace=never
```

Entity

```java
package com.g2appdev.jabi.entity;

import jakarta.persistence.CascadeType;

@Entity
public class StudentEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int studentID;

    private String firstName;
    private String lastName;
    private String email;
    private char gender;
    private int age;
    private int yearLevel;

    @ManyToOne(cascade = CascadeType.ALL)
    @JoinColumn (name = "groupID")
    private GroupEntity groups;

    public StudentEntity() {
        super();
    }

    public StudentEntity(int studentID, String firstName, String lastName, String email, char gender, int age, int yearLevel) {
        super();
        this.studentID = studentID;
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
        this.gender = gender;
        this.age = age;
        this.yearLevel = yearLevel;
    }

    public int getStudentID() {
        return studentID;
    }

    public String getFirstName() {
        return firstName;
    }
```

```java
package com.g2appdev.jabi.entity;

import java.util.List;

@Entity
public class GroupEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int groupID;

    private String groupName;

    @OneToMany(fetch = FetchType.LAZY,
            mappedBy = "groups",
            cascade = CascadeType.ALL)
    private List<StudentEntity> members;

    public GroupEntity() {
        super();
    }

    public GroupEntity(int groupID, String groupName) {
        super();
        this.groupID = groupID;
        this.groupName = groupName;
    }

    public int getGroupID() {
        return groupID;
    }

    public String getGroupName() {
        return groupName;
    }

    public void setGroupName(String groupName) {
        this.groupName = groupName;
    }
}
```

Repository

```java
package com.g2appdev.jabi.repository;

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

@Repository
public interface StudentRepository extends JpaRepository<StudentEntity, Integer>{

    //this is user-defined method to search a student record by last name
    public StudentEntity findByLastName(String lastName);

    //other searching methods
    public StudentEntity findByFirstName(String firstName);
    public StudentEntity findByEmail(String email);
    public StudentEntity findByGender(char gender);
    public StudentEntity findByAge(int age);
    public StudentEntity findByYearLevel(int yearLevel);
}
```

Service

```java
package com.g2appdev.jabi.service;

import java.util.List;

@Service
public class StudentService {

    @Autowired
    StudentRepository srepo;

    @Autowired
    GroupRepository grepo;

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

    //Create of CRUD
    public StudentEntity postStudentRecord(StudentEntity student) {
        if (student.getGroups() != null) {
            // Check if the group exists
            GroupEntity group = grepo.findById(student.getGroups().getGroupID())
                    .orElseThrow(() -> new NoSuchElementException("Group not found"));
            // Set the group in the student entity
            student.setGroups(group);
        }
        return srepo.save(student);
    }

    //Read of CRUD
    public List<StudentEntity>getAllStudents(){
        return srepo.findAll();
    }

}
```

```java
    //Update of CRUD
    @SuppressWarnings("finally")
    public StudentEntity putStudentDetails(int id, StudentEntity newStudentDetails) {
        StudentEntity student = new StudentEntity();

        try {
            //search the id number
            student = srepo.findById(id).get();

            //if id is found edit comes
            student.setFirstName(newStudentDetails.getFirstName());
            student.setLastName(newStudentDetails.getLastName());
            student.setEmail(newStudentDetails.getEmail());
            student.setGender(newStudentDetails.getGender());
            student.setAge(newStudentDetails.getAge());
            student.setYearLevel(newStudentDetails.getYearLevel());
        }
        catch(NoSuchElementException nex){
            throw new NameNotFoundException ("Student " + id + " not found");
        }
        finally {
            return srepo.save(student);
        }
    }

    //Delete of CRUD
    public String deleteStudent(int id) {
        String msg = "";

        if(srepo.findById(id) != null) {
            srepo.deleteById(id);
            msg = "Student Record successfully deleted!";
        } else {
            msg = "Student with id " + id + " NOT FOUND";
        }

        return msg;
    }
}
```

Controller

```java
package com.g2appdev.jabi.controller;

import java.util.List;

@RestController
@RequestMapping(method = RequestMethod.GET,path="/api/student")
public class StudentController {

    @Autowired
    StudentService sserv;

    @GetMapping("/print")
    public String print() {
        return "Hello, Firstname Lastname";
    }

    //Create of CRUD
    @PostMapping("/poststudentrecord")
    public StudentEntity postStudentRecord(@RequestBody StudentEntity student) {
        return sserv.postStudentRecord(student);
    }

    //Read of CRUD
    @GetMapping("/getAllStudents")
    public List<StudentEntity>getAllStudents(){
        return sserv.getAllStudents();
    }

    //Update of CRUD
    @PutMapping("/putStudentDetails")
    public StudentEntity putStudentDetails(@RequestParam int id, @RequestBody StudentEntity newStudentDetails) {
        return sserv.putStudentDetails(id, newStudentDetails);
    }

    //Delete of CRUD
    @DeleteMapping("/deleteStudentDetails/{id}")
    public String deleteStudent(@PathVariable int id) {
        return sserv.deleteStudent(id);
    }
}
```