

JPA Application Development & CRUD Operations

Objectives

- ◆ Data Management
 - Implement CRUD (Create, Read, Update, Delete) operations to manage data effectively.
 - Utilize JPA to map Java objects to database tables, simplifying data persistence and retrieval.
- ◆ User Interface Development
 - Design and develop a user-friendly interface using JavaFX, enabling seamless interaction with the application.
 - Implement features such as form input validation, error handling, and feedback mechanisms to enhance user experience.

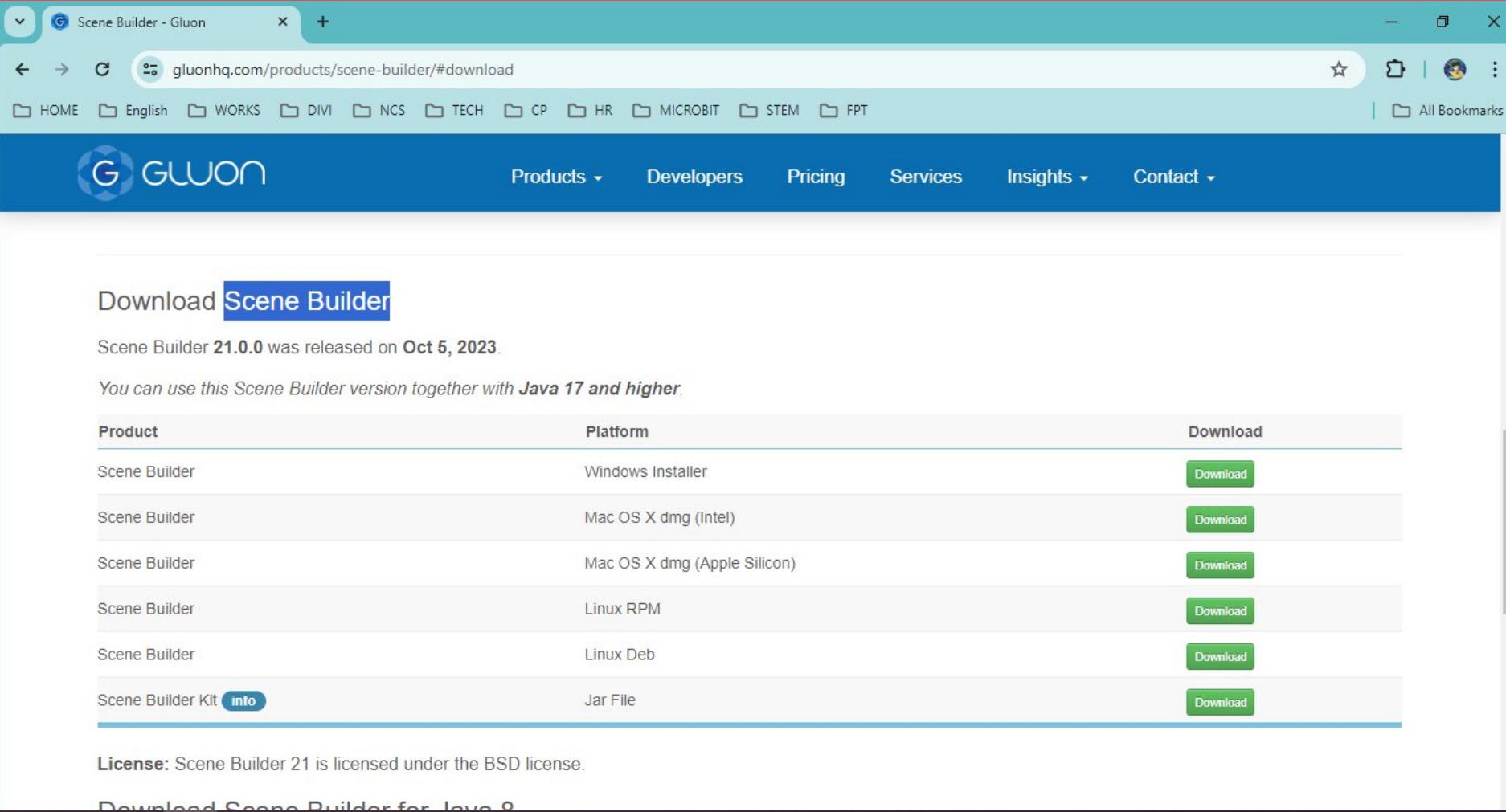
Student Management System

Build Simple Student Management System

- ◆ Step 1. Create Project in IntelliJ
- ◆ Step 2. Setup Scene Builder
- ◆ Step 3. Build CRUD Application with Repository Pattern
- ◆ Step 4. Design GUI Application with JavaFX

Setup Scene Builder

Go to website



The screenshot shows a web browser window with the title "Scene Builder - Gluon". The address bar contains the URL "gluonhq.com/products/scene-builder/#download". The page header includes the Gluon logo and navigation links for Products, Developers, Pricing, Services, Insights, and Contact.

Download Scene Builder

Scene Builder 21.0.0 was released on **Oct 5, 2023**.

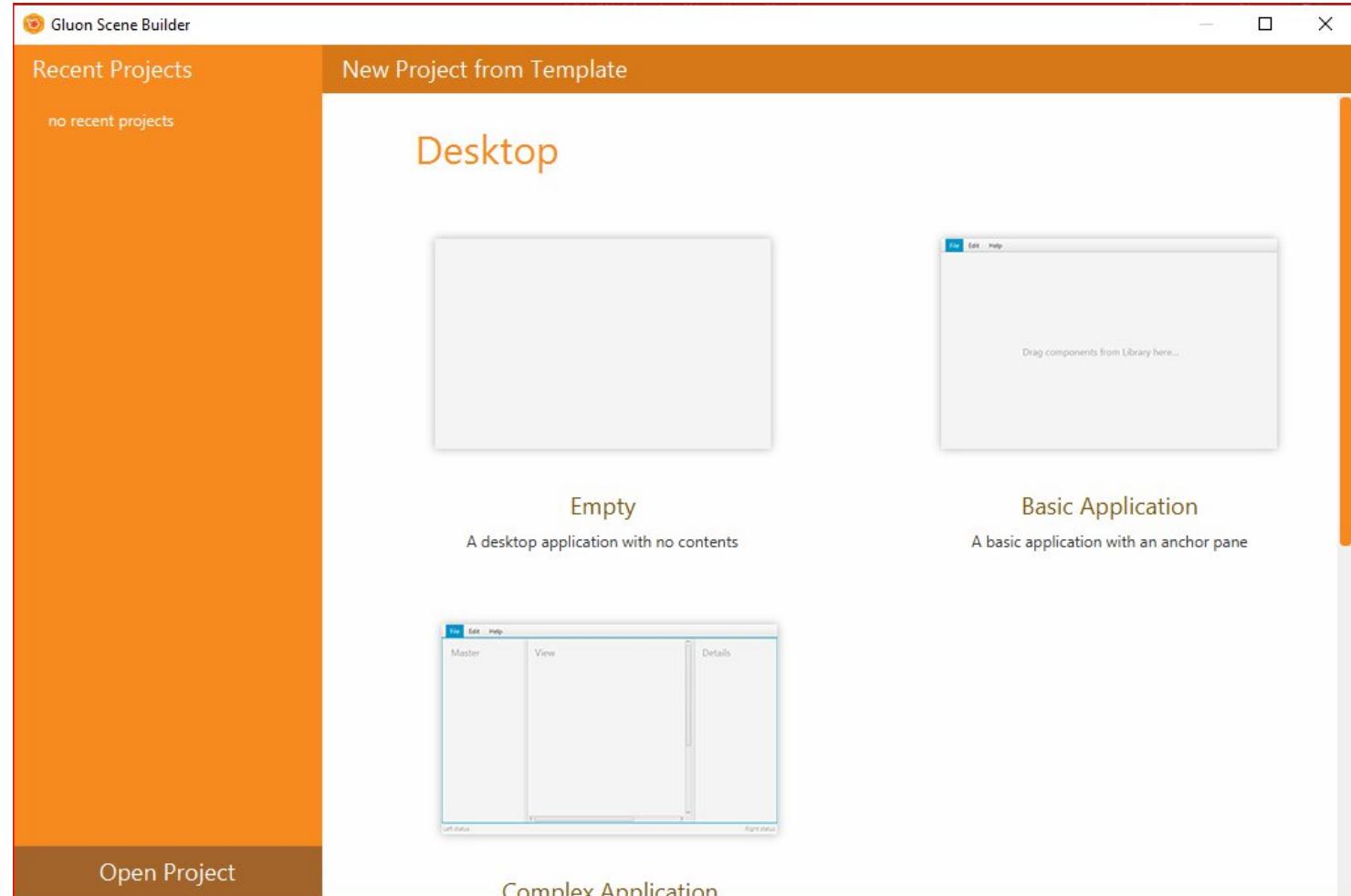
You can use this Scene Builder version together with **Java 17 and higher**.

Product	Platform	Download
Scene Builder	Windows Installer	Download
Scene Builder	Mac OS X dmg (Intel)	Download
Scene Builder	Mac OS X dmg (Apple Silicon)	Download
Scene Builder	Linux RPM	Download
Scene Builder	Linux Deb	Download
Scene Builder Kit <small>info</small>	Jar File	Download

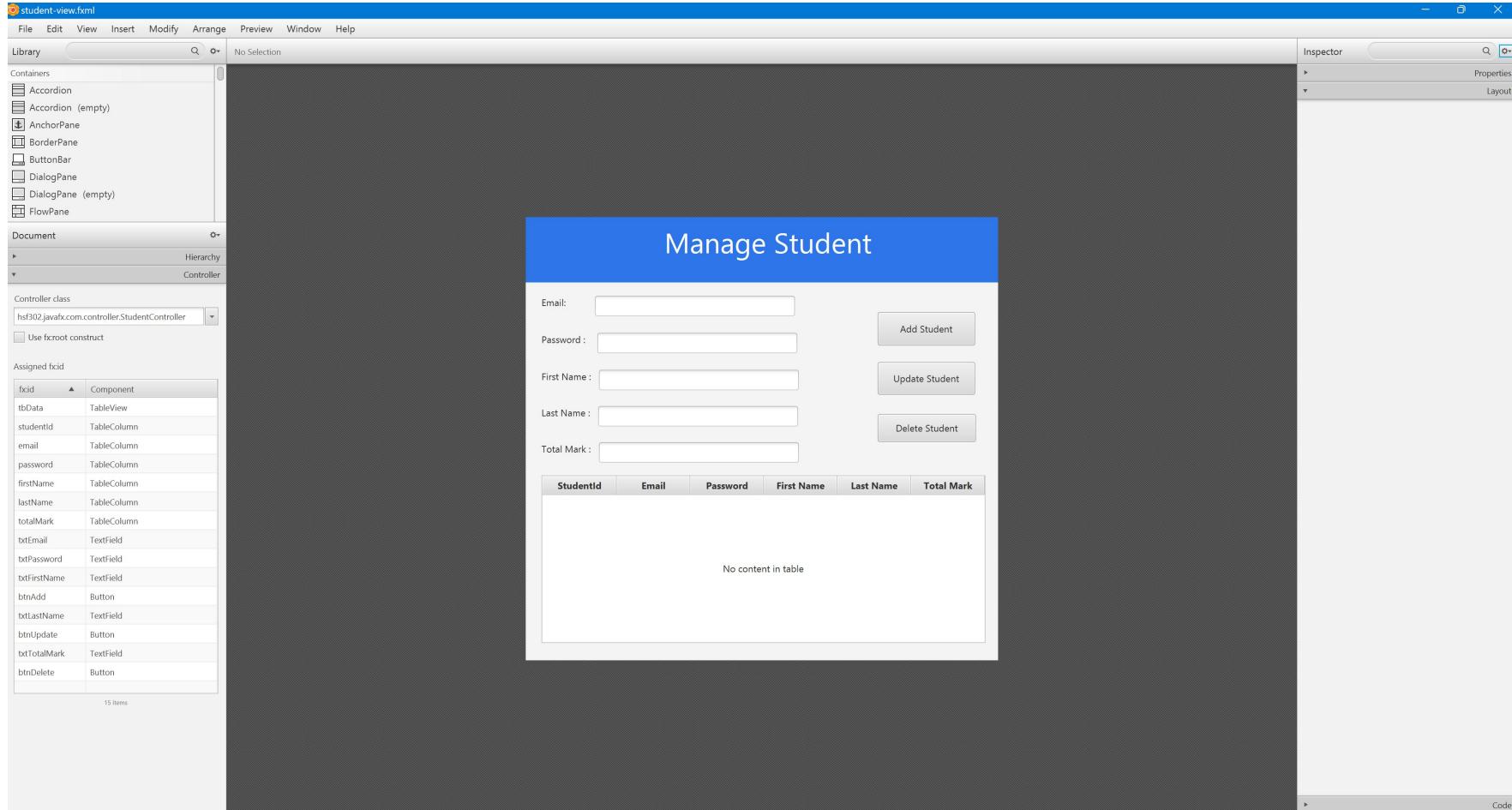
License: Scene Builder 21 is licensed under the BSD license.

[Download Scene Builder for Java 9](#)

2. Open Gluon Scene Builder

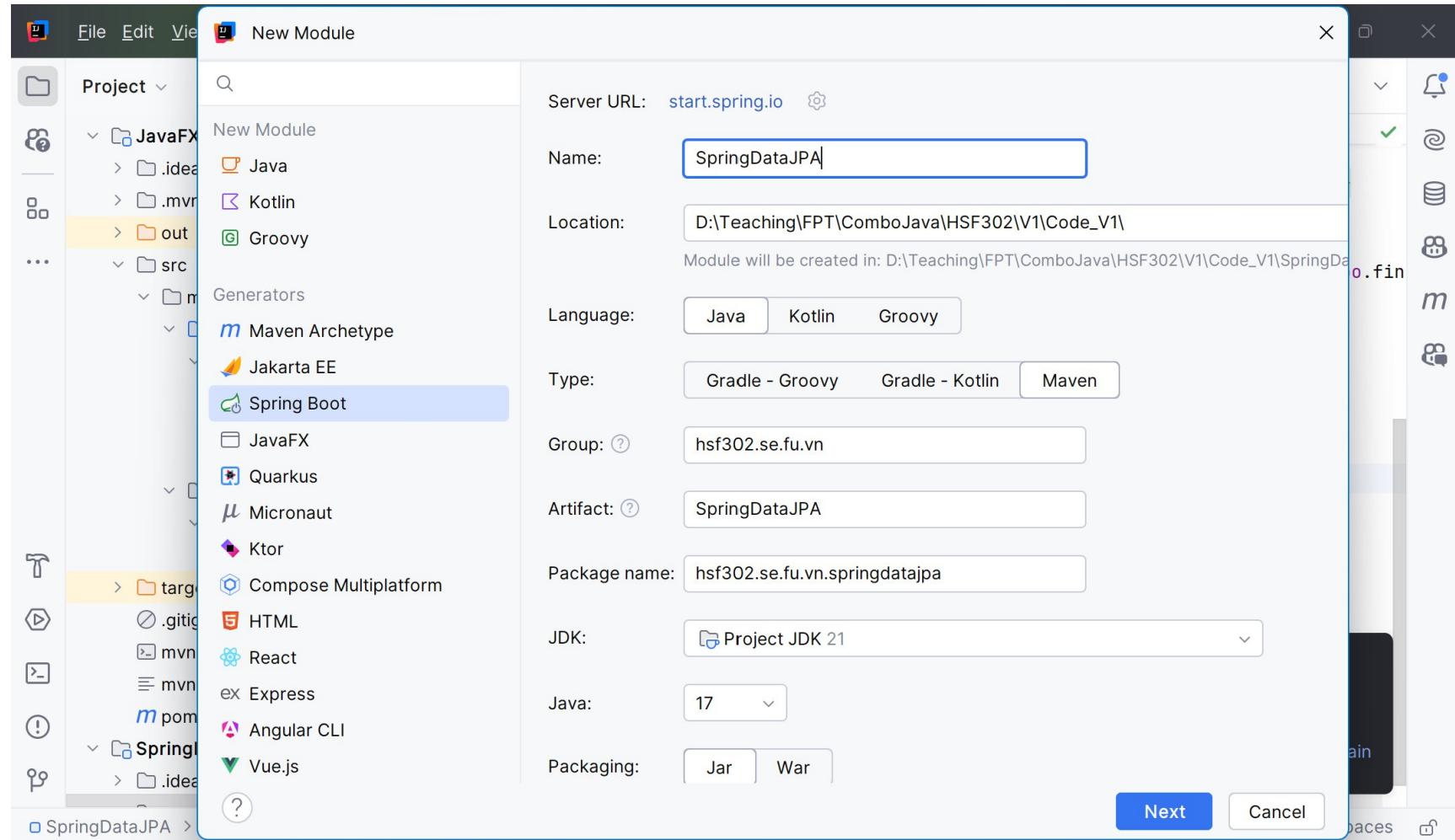


Open JavaFX File

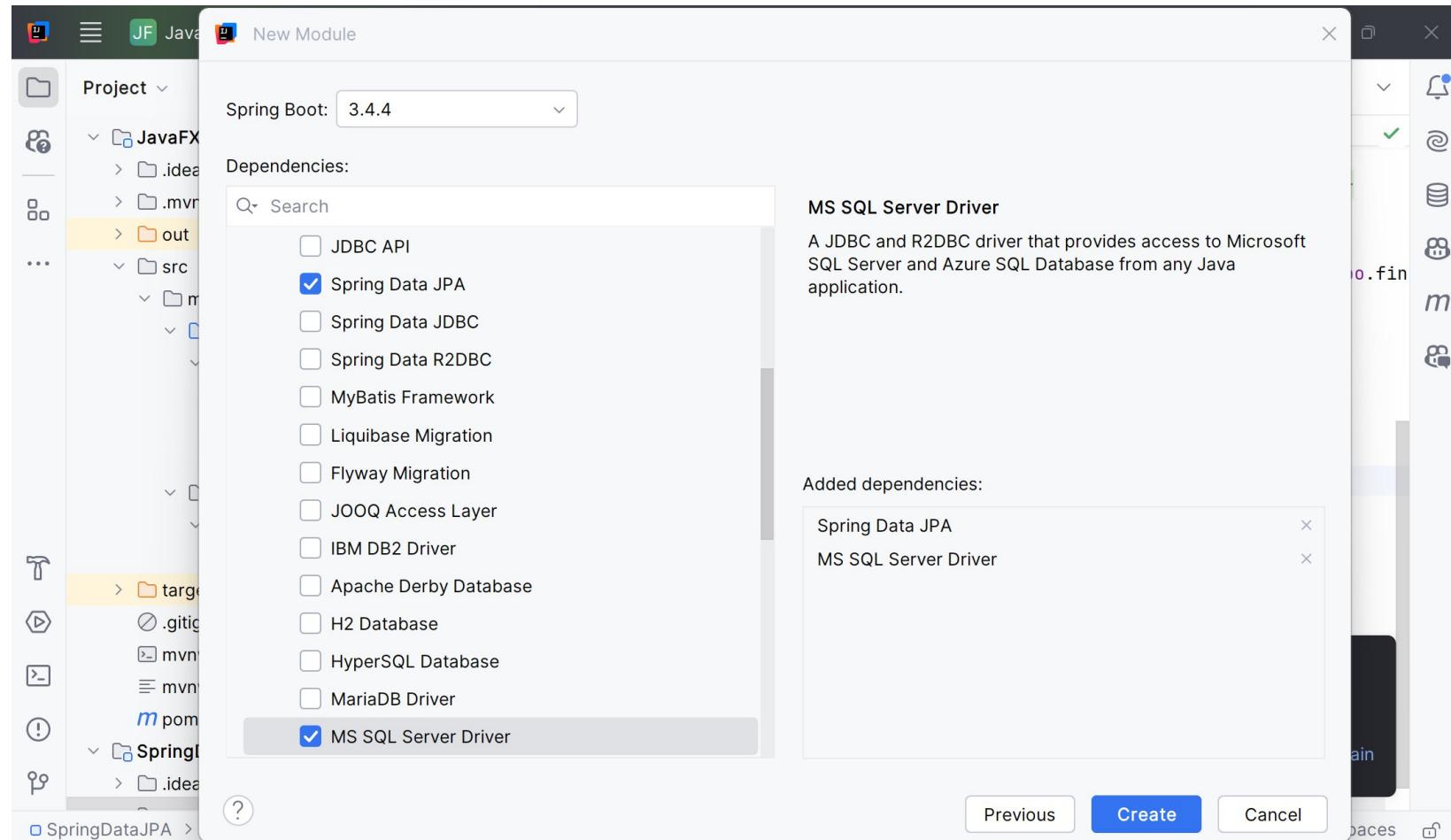


Build CRUD Application with Repository Pattern

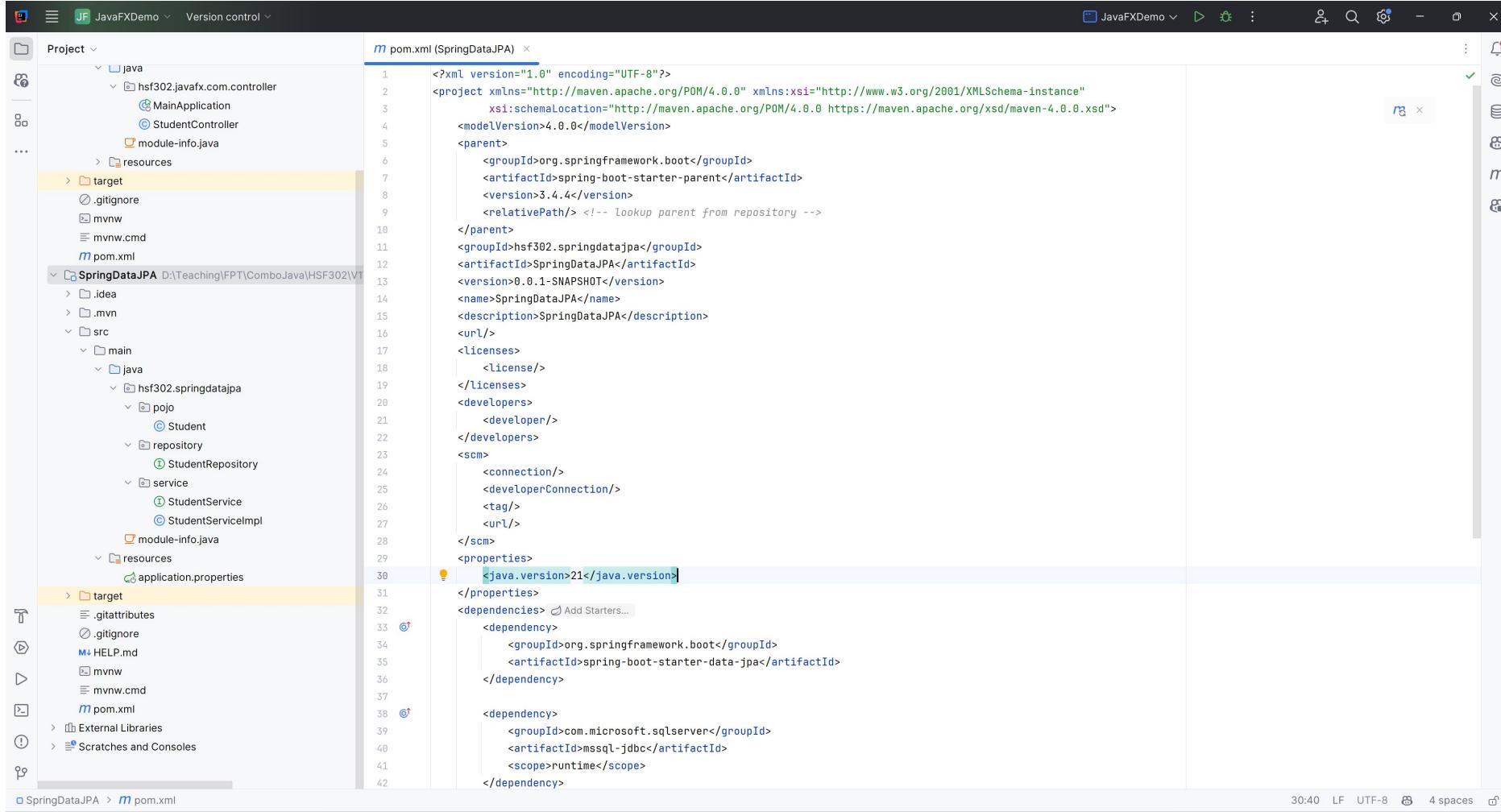
File -> New Module



File -> New Module



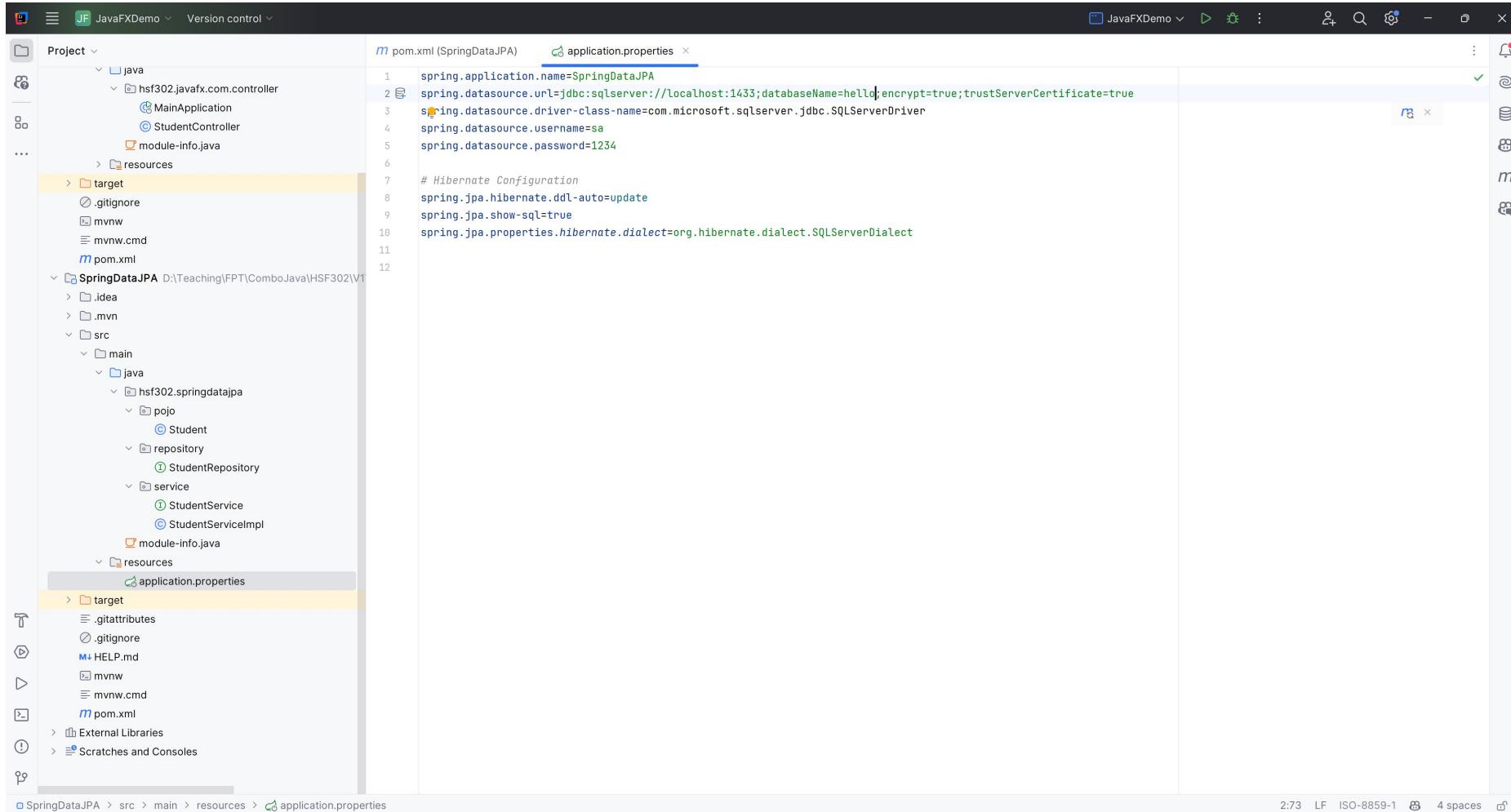
Create the Structure



The screenshot shows the IntelliJ IDEA interface with the project `JavaFXDemo` open. The left panel displays the project structure, which includes a `SpringDataJPA` module containing `java`, `resources`, and `target` directories, along with `.gitignore`, `mvnw`, and `pom.xml` files. The right panel shows the `pom.xml` file content:

```
<?xml version="1.0" encoding="UTF-8"?>
<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.4.4</version>
    <relativePath/> 
  </parent>
  <groupId>hsf302.springdatajpa</groupId>
  <artifactId>SpringDataJPA</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>SpringDataJPA</name>
  <description>SpringDataJPA</description>
  <url/>
  <licenses>
    <license/>
  </licenses>
  <developers>
    <developer/>
  </developers>
  <scm>
    <connection/>
    <developerConnection/>
    <tag/>
    <url/>
  </scm>
  <properties>
    <java.version>21</java.version>
  </properties>
  <dependencies> Add Starters...
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
      <groupId>com.microsoft.sqlserver</groupId>
      <artifactId>mssql-jdbc</artifactId>
      <scope>runtime</scope>
    </dependency>
  
```

Update application.properties



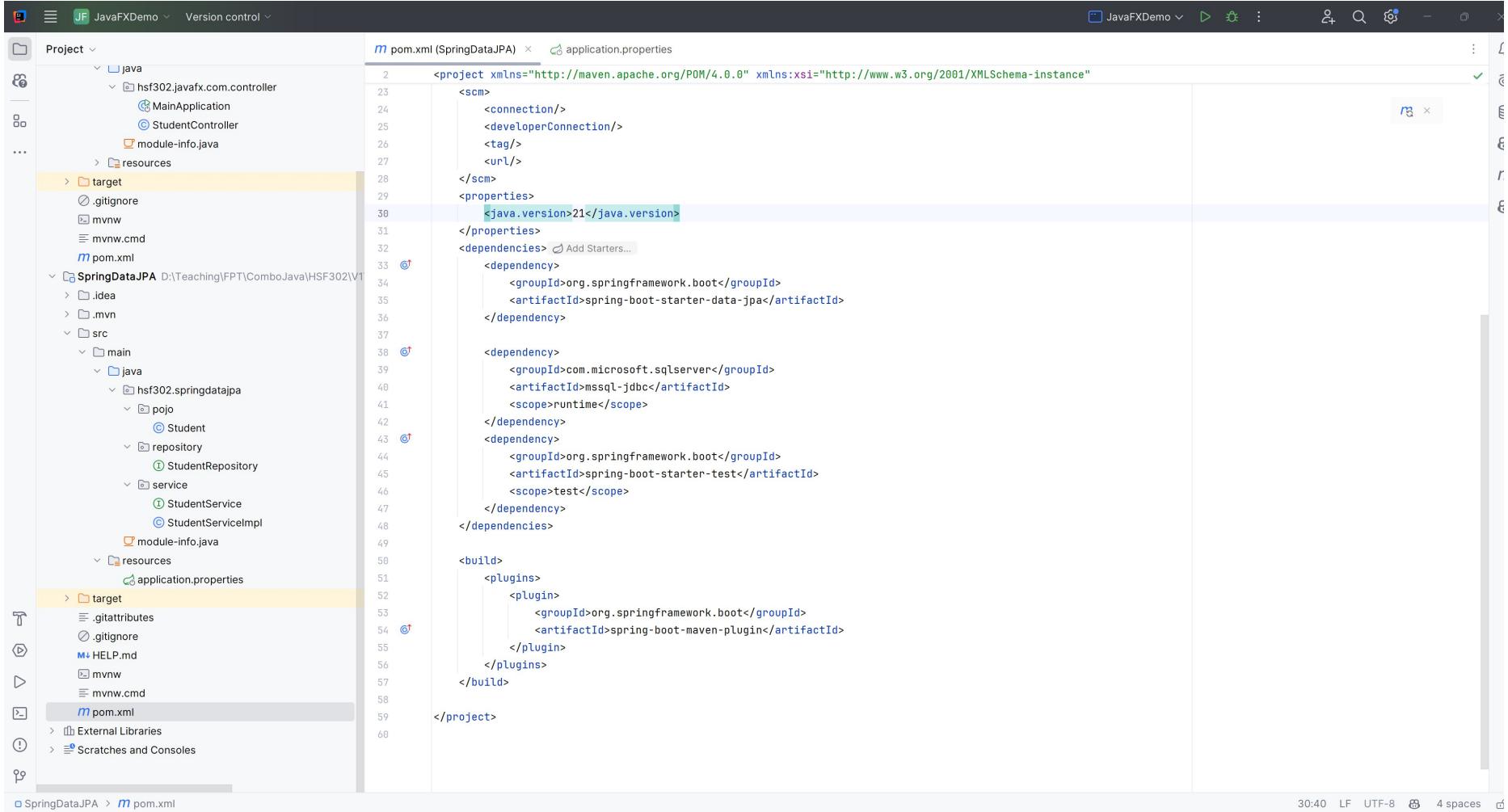
The screenshot shows the IntelliJ IDEA interface with the project structure on the left and the code editor on the right. The code editor displays the `application.properties` file for the `SpringDataJPA` module. The file contains the following configuration:

```
spring.application.name=SpringDataJPA
spring.datasource.url=jdbc:sqlserver://localhost:1433;databaseName=hello;encrypt=true;trustServerCertificate=true
spring.datasource.driver-class-name=com.microsoft.sqlserver.jdbc.SQLServerDriver
spring.datasource.username=sa
spring.datasource.password=1234

# Hibernate Configuration
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.SQLServerDialect
```

The IntelliJ IDEA interface includes toolbars, a status bar at the bottom, and various icons for navigating through the code and project.

Update pom.xml



The screenshot shows the IntelliJ IDEA interface with the project "JavaFDemo" open. The left sidebar displays the project structure, including the "src" directory which contains "main" and "SpringDataJPA". The "main" directory has "java", "resources", and "module-info.java" sub-directories. The "SpringDataJPA" directory contains ".idea", ".mvn", and "src" sub-directories, which further contain "main", "java", "resources", and "module-info.java". The "pom.xml" file is selected in both the project tree and the editor tab.

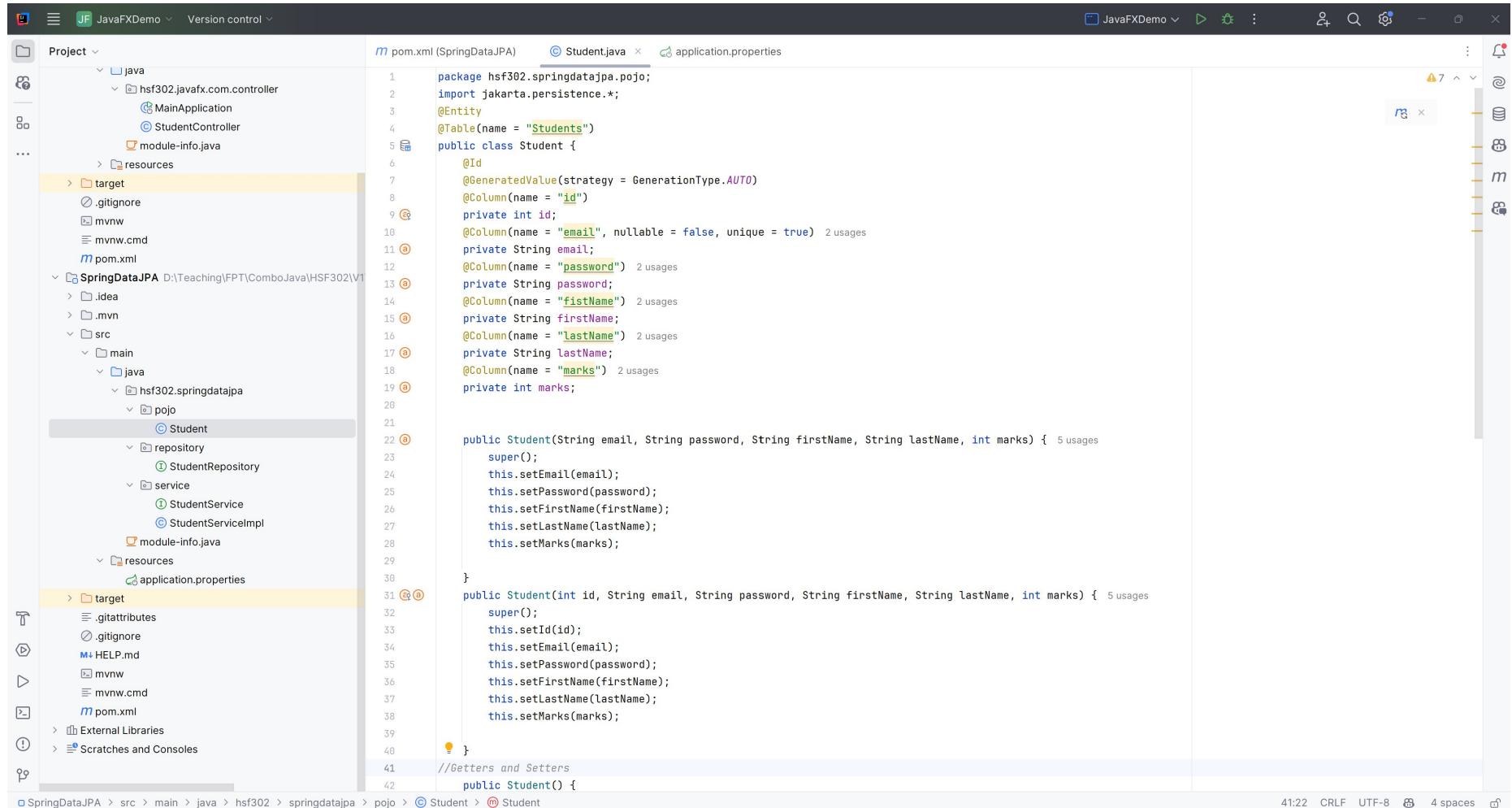
The editor tab shows the content of the pom.xml file:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <scm>
    <connection/>
    <developerConnection/>
    <tag/>
    <url/>
  </scm>
  <properties>
    <java.version>21</java.version>
  </properties>
  <dependencies> Add Starters...
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>

    <dependency>
      <groupId>com.microsoft.sqlserver</groupId>
      <artifactId>mssql-jdbc</artifactId>
      <scope>runtime</scope>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
    </plugins>
  </build>
</project>
```

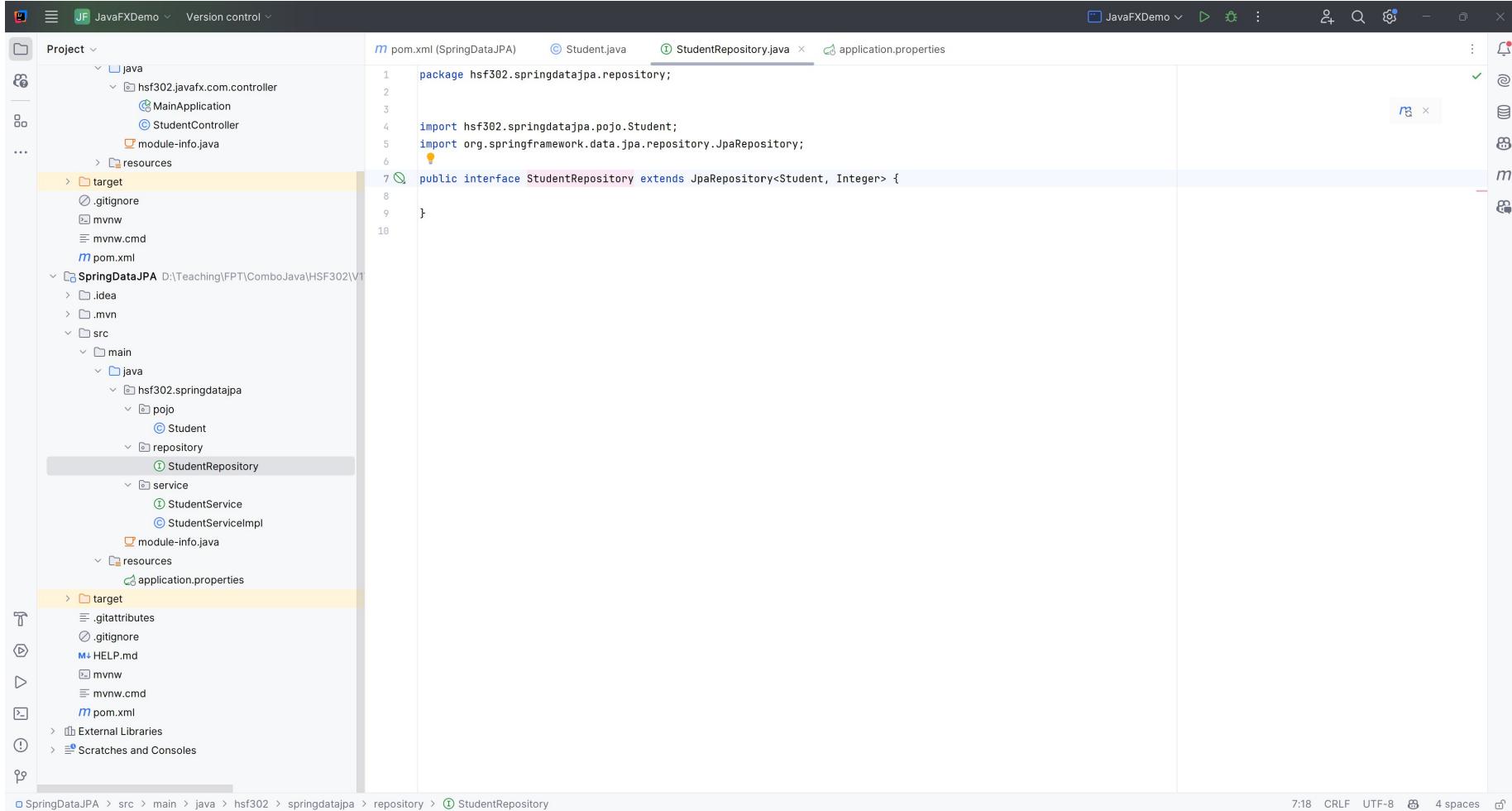
Create Student.java



The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The left sidebar displays the project structure under "Project". It includes the "java" directory containing "hsf302.javaFX.com.controller" (with "MainApplication" and "StudentController"), "module-info.java", and "resources". Below "java" are "target", ".gitignore", "mvnw", "mvnw.cmd", and "pom.xml". The "SpringDataJPA" module contains ".idea", ".mvn", "src" (with "main" and "hsf302.springdatajpa" sub-directories), "pom.xml", and "application.properties".
- Code Editor:** The right pane shows the code for "Student.java". The code defines a class "Student" with annotations for Entity and Table. It has fields for id, email, password, firstName, lastName, and marks, each with corresponding @Column annotations. The constructor and getters/setters are also defined.
- Status Bar:** The bottom status bar shows the file path: "SpringDataJPA > src > main > java > hsf302 > springdatajpa > pojo > Student.java". It also displays the current time (41:22), encoding (CRLF), character set (UTF-8), and code style settings (4 spaces).

Create StudentRepository.java



The screenshot shows the IntelliJ IDEA interface with the following details:

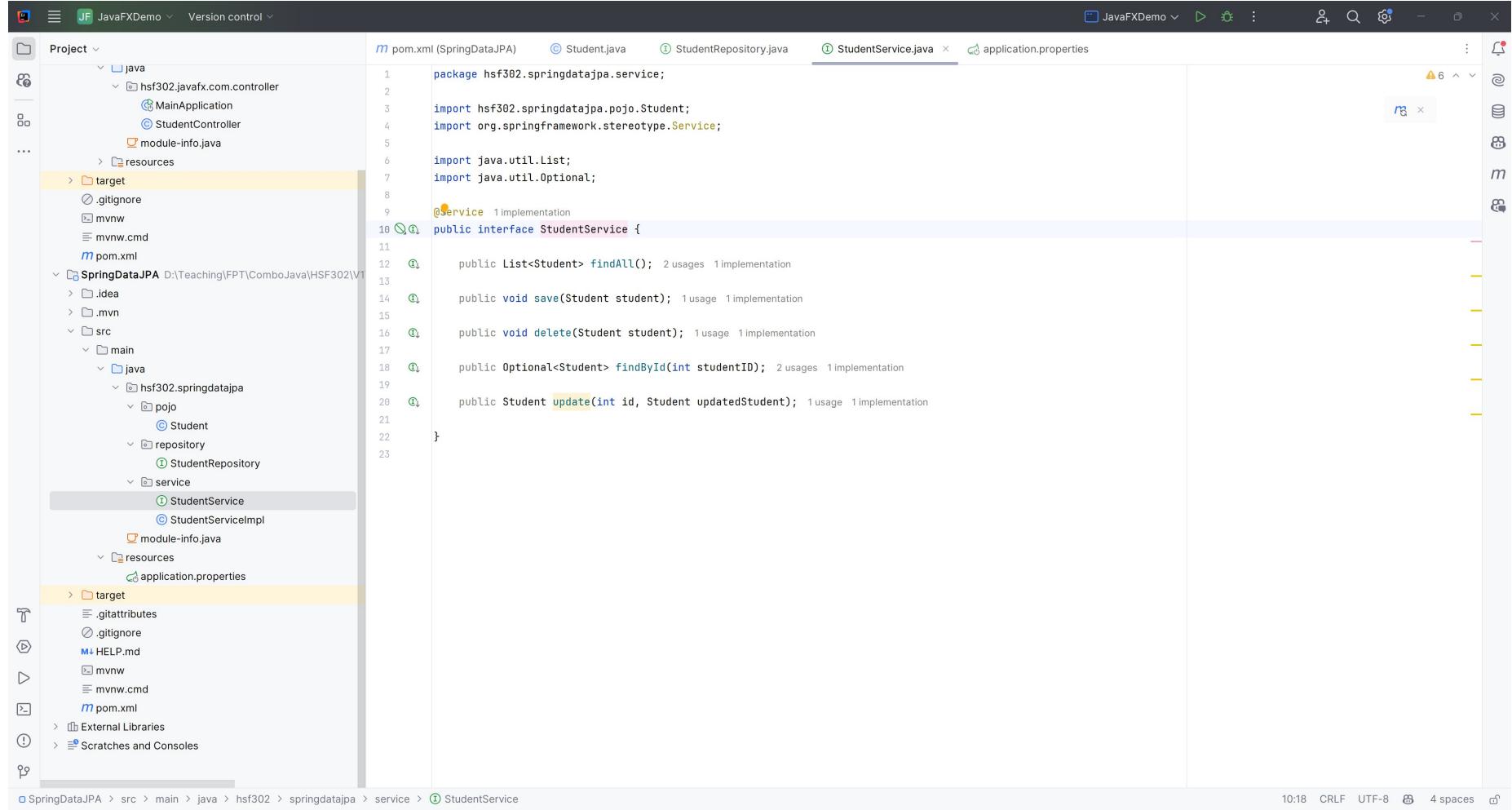
- Project Structure:** The left sidebar shows the project structure under "Project". It includes a "java" directory containing "hsf302.javafx.com.controller" (with "MainApplication" and "StudentController"), "resources", "target", ".gitignore", "mvnw", "mvnw.cmd", and "pom.xml". Inside "src/main/java", there is a "hsf302.springdatajpa" package with "pojo" (containing "Student") and "repository" (containing "StudentRepository").
 - "StudentRepository" is currently selected.
- Code Editor:** The right pane displays the code for "StudentRepository.java".

```
package hsf302.springdatajpa.repository;

import hsf302.springdatajpa.pojo.Student;
import org.springframework.data.jpa.repository.JpaRepository;

public interface StudentRepository extends JpaRepository<Student, Integer> {
```
- Status Bar:** At the bottom, the status bar shows the file path "SpringDataJPA > src > main > java > hsf302 > springdatajpa > repository > StudentRepository", and the status "7:18 CRLF UTF-8 4 spaces".

Create StudentService.java



The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The left sidebar shows the project structure under "JavaFXDemo". It includes a "java" folder containing "hsf302.javafx.com.controller" (with "MainApplication" and "StudentController"), "resources", "target", ".gitignore", "mvnw", "mvnw.cmd", and "pom.xml". A "SpringDataJPA" module is also listed, which contains ".idea", "mvn", "src" (with "main" and "java" subfolders), and "pom.xml".
- Code Editor:** The right pane displays the code for "StudentService.java". The code defines a public interface named "StudentService" with the following methods:

```
package hsf302.springdatajpa.service;

import hsf302.springdatajpa.pojo.Student;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;

@Service 1 implementation
public interface StudentService {

    public List<Student> findAll(); 2 usages 1 implementation

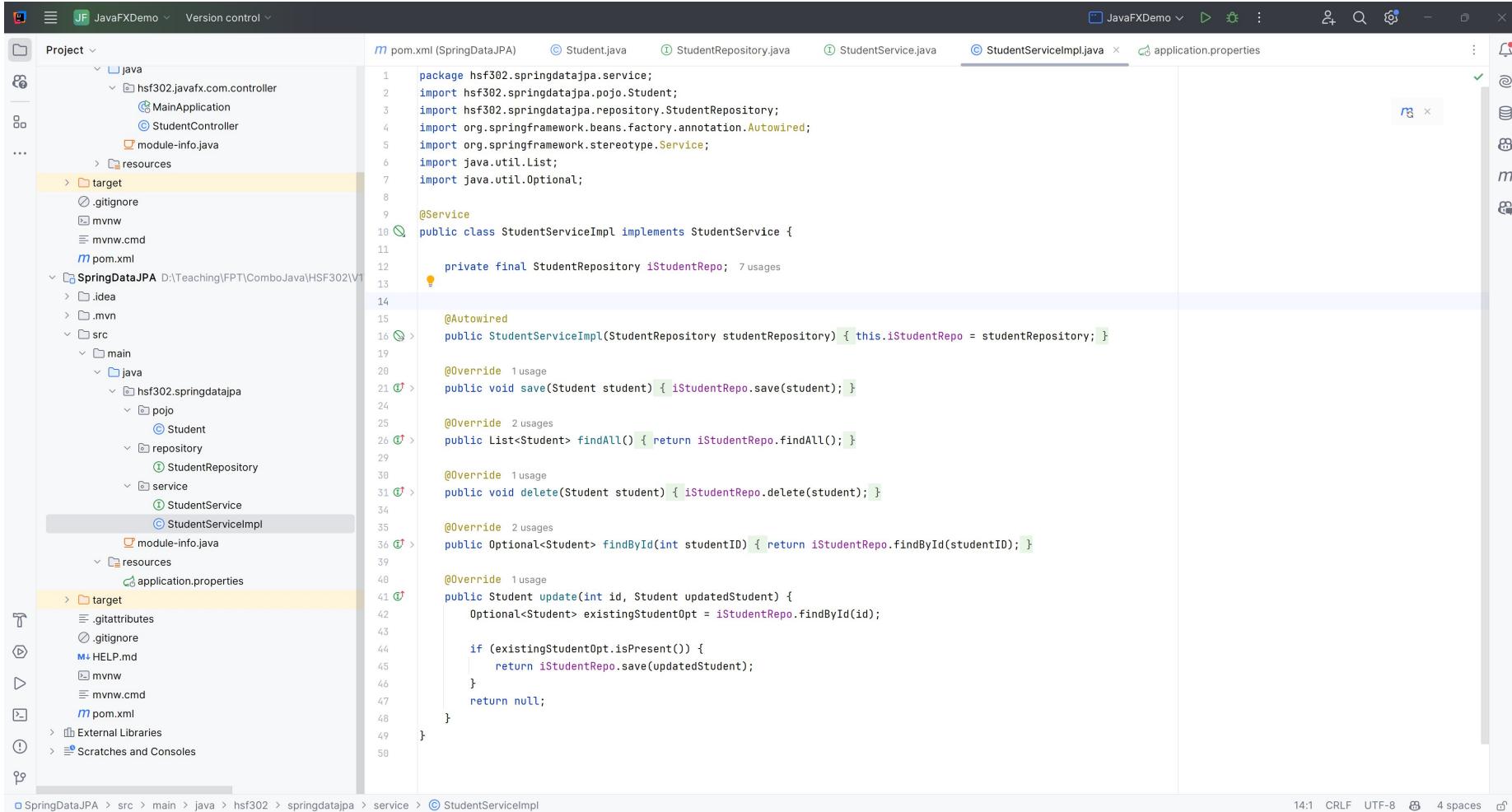
    public void save(Student student); 1 usage 1 implementation

    public void delete(Student student); 1 usage 1 implementation

    public Optional<Student> findById(int studentID); 2 usages 1 implementation

    public Student update(int id, Student updatedStudent); 1 usage 1 implementation
}
```
- Status Bar:** The bottom status bar shows the file path as "SpringDataJPA > src > main > java > hsf302 > springdatajpa > service > StudentService.java", and the status "10:18 CRLF UTF-8 4 spaces".

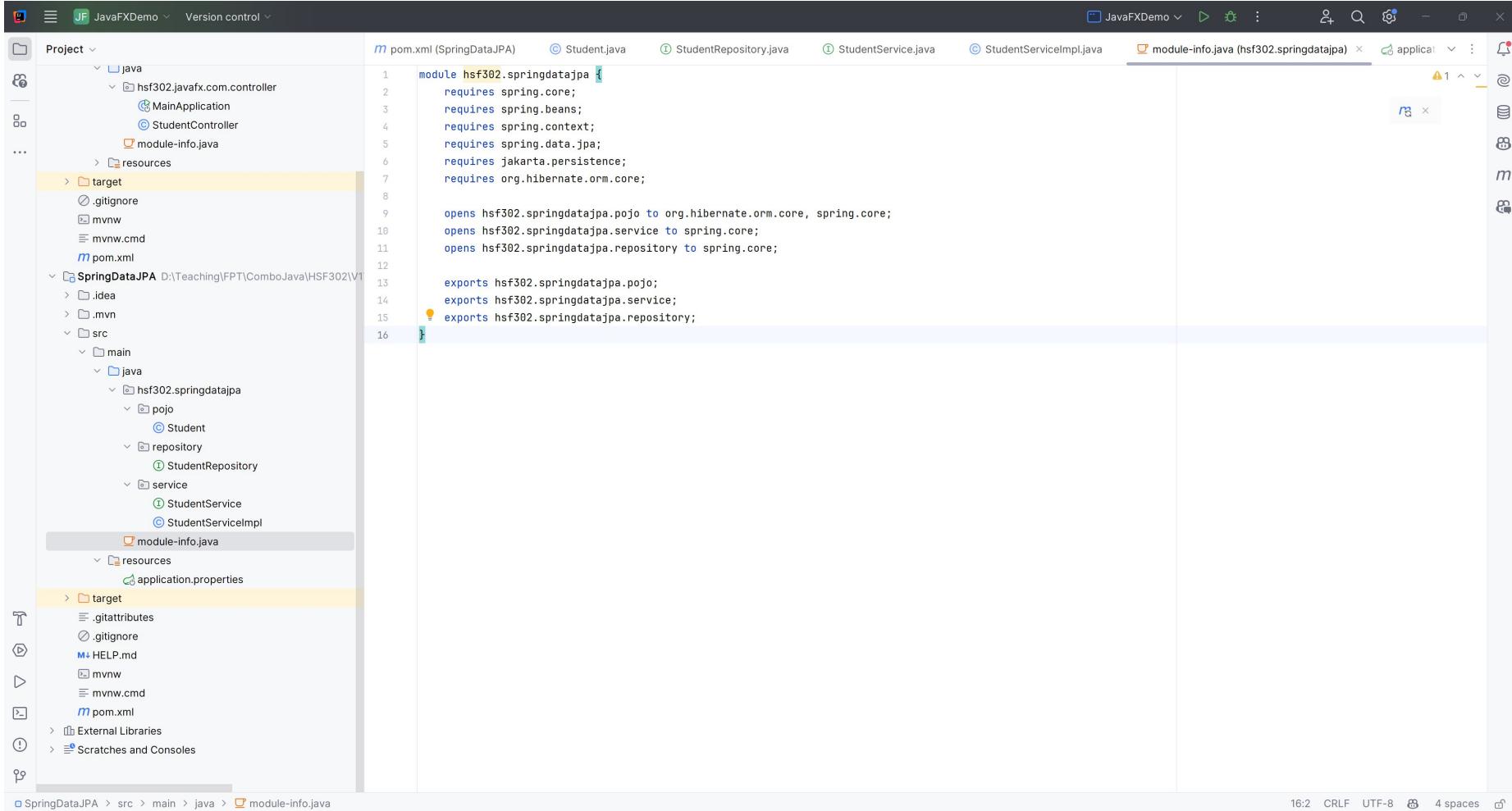
Create StudentServiceImpl.java



The screenshot shows the IntelliJ IDEA IDE interface with the following details:

- Project Structure:** The left sidebar shows the project structure under "JavaFXDemo". It includes a "SpringDataJPA" module with "src/main/java" containing "hsf302.springdatajpa" packages for "pojo", "repository", and "service". The "service" package contains the "StudentService" interface and the "StudentServiceImpl" implementation class, which is currently selected.
- Code Editor:** The main editor pane displays the code for "StudentServiceImpl.java". The code implements the "StudentService" interface and uses Spring annotations like @Service and @Autowired. It interacts with an "iStudentRepo" repository to perform operations like saving, finding all, deleting, and updating students.
- Toolbars and Status Bar:** The top bar shows tabs for pom.xml, Student.java, StudentRepository.java, StudentService.java, StudentServiceImpl.java (which is active), and application.properties. The bottom status bar shows file paths, encoding (CRLF, UTF-8), and code style settings (4 spaces).

Create module-info.java



The screenshot shows the IntelliJ IDEA interface with a JavaFXDemo project open. The left sidebar displays the project structure, including a `java` directory containing `hsf302.javafx.com.controller`, `resources`, and `target`. Inside `hsf302.javafx.com.controller` are `MainApplication` and `StudentController`, along with a `module-info.java` file. The `src` directory contains `main` and `test` packages, with `main` further divided into `java`, `pojo`, `repository`, `service`, and `resources`. The `java` package under `main` contains `hsf302.springdatajpa`, which includes `pojo`, `repository`, `service`, and `module-info.java`. The `resources` package contains `application.properties`. The right panel shows the `module-info.java` code:

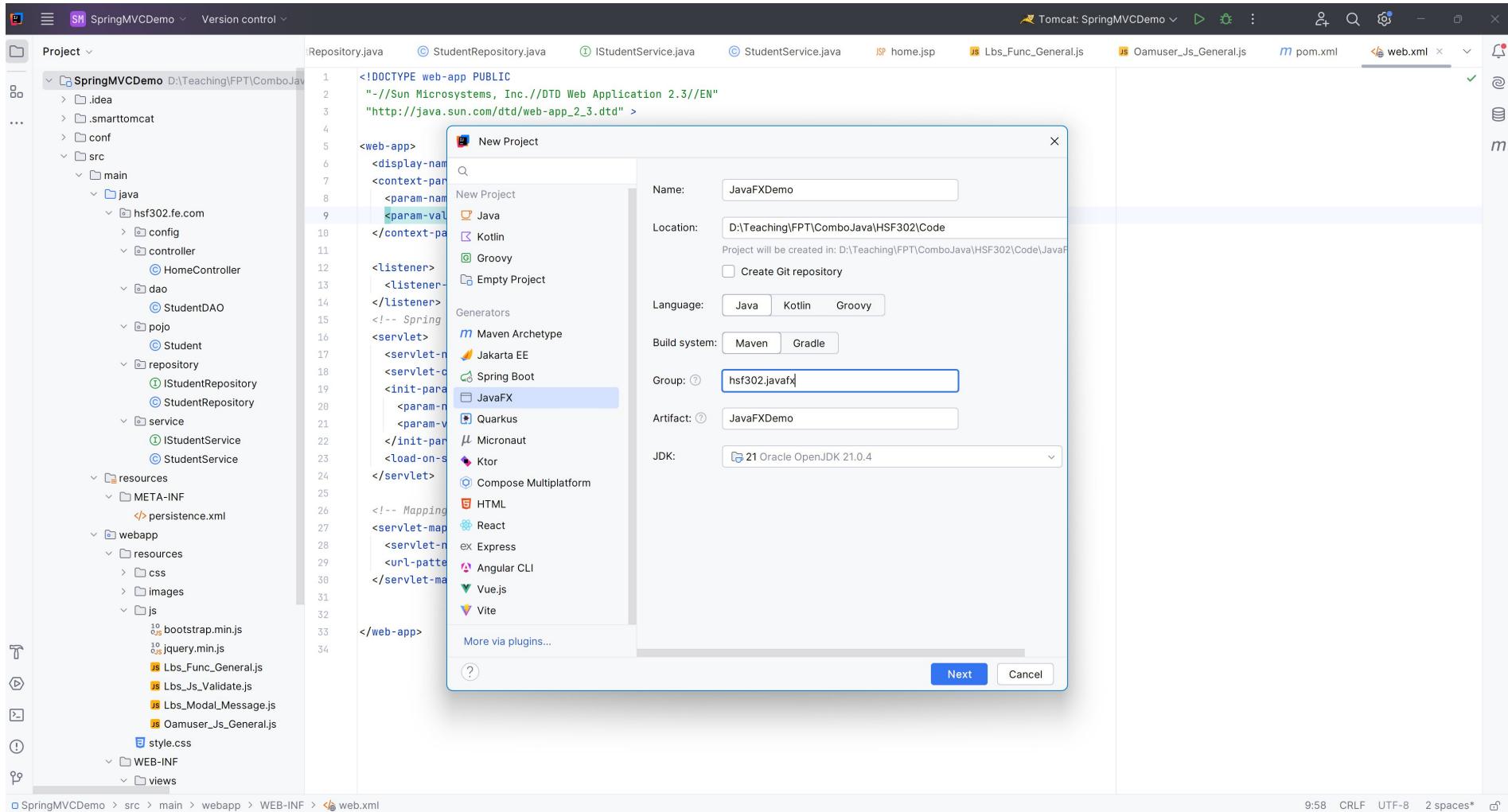
```
module hsf302.springdatajpa;
requires spring.core;
requires spring.beans;
requires spring.context;
requires spring.data.jpa;
requires jakarta.persistence;
requires org.hibernate.orm.core;

opens hsf302.springdatajpa.pojo to org.hibernate.orm.core, spring.core;
opens hsf302.springdatajpa.service to spring.core;
opens hsf302.springdatajpa.repository to spring.core;

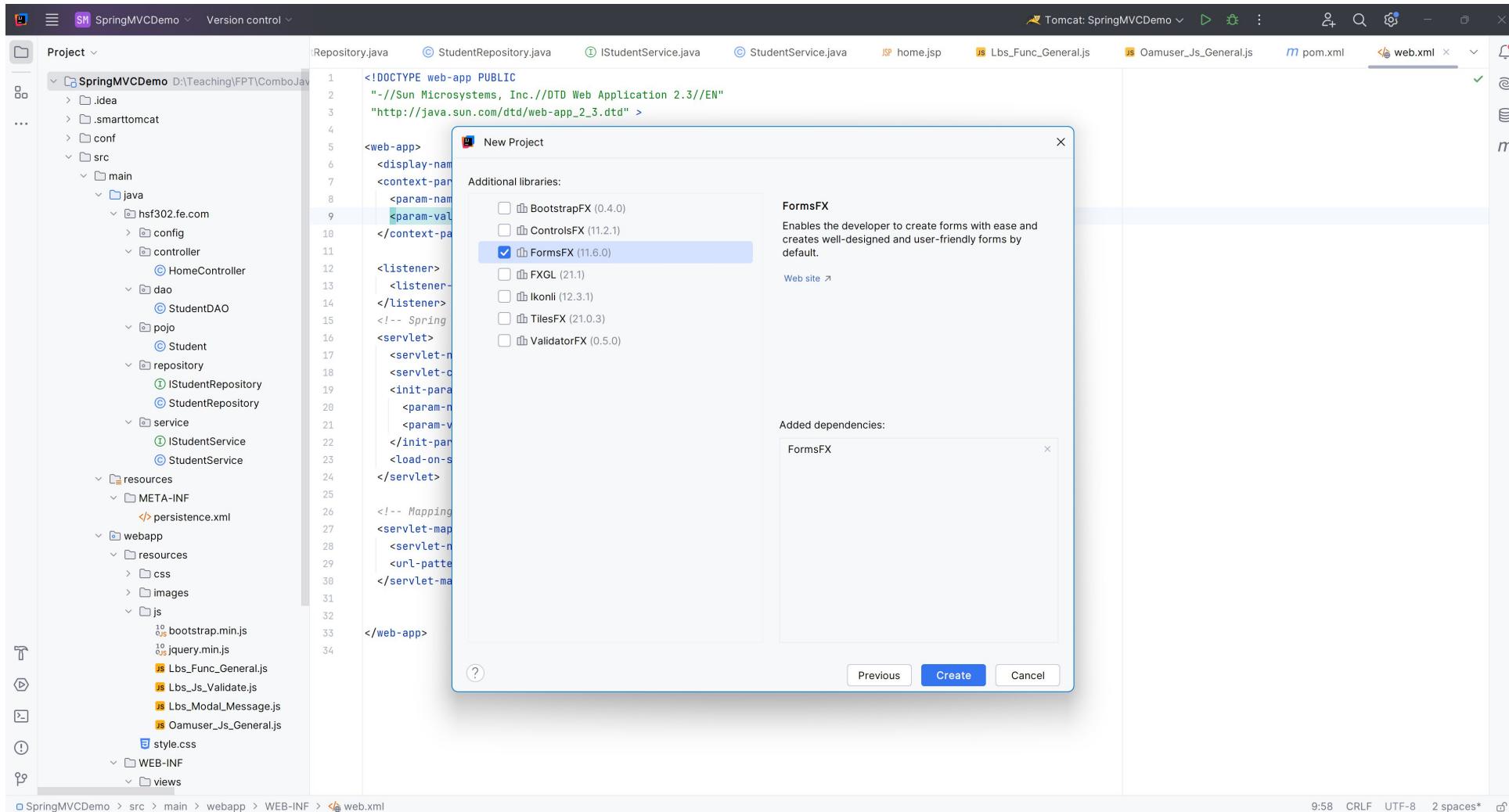
exports hsf302.springdatajpa.pojo;
exports hsf302.springdatajpa.service;
exports hsf302.springdatajpa.repository;
```

Design GUI Application with JavaFX

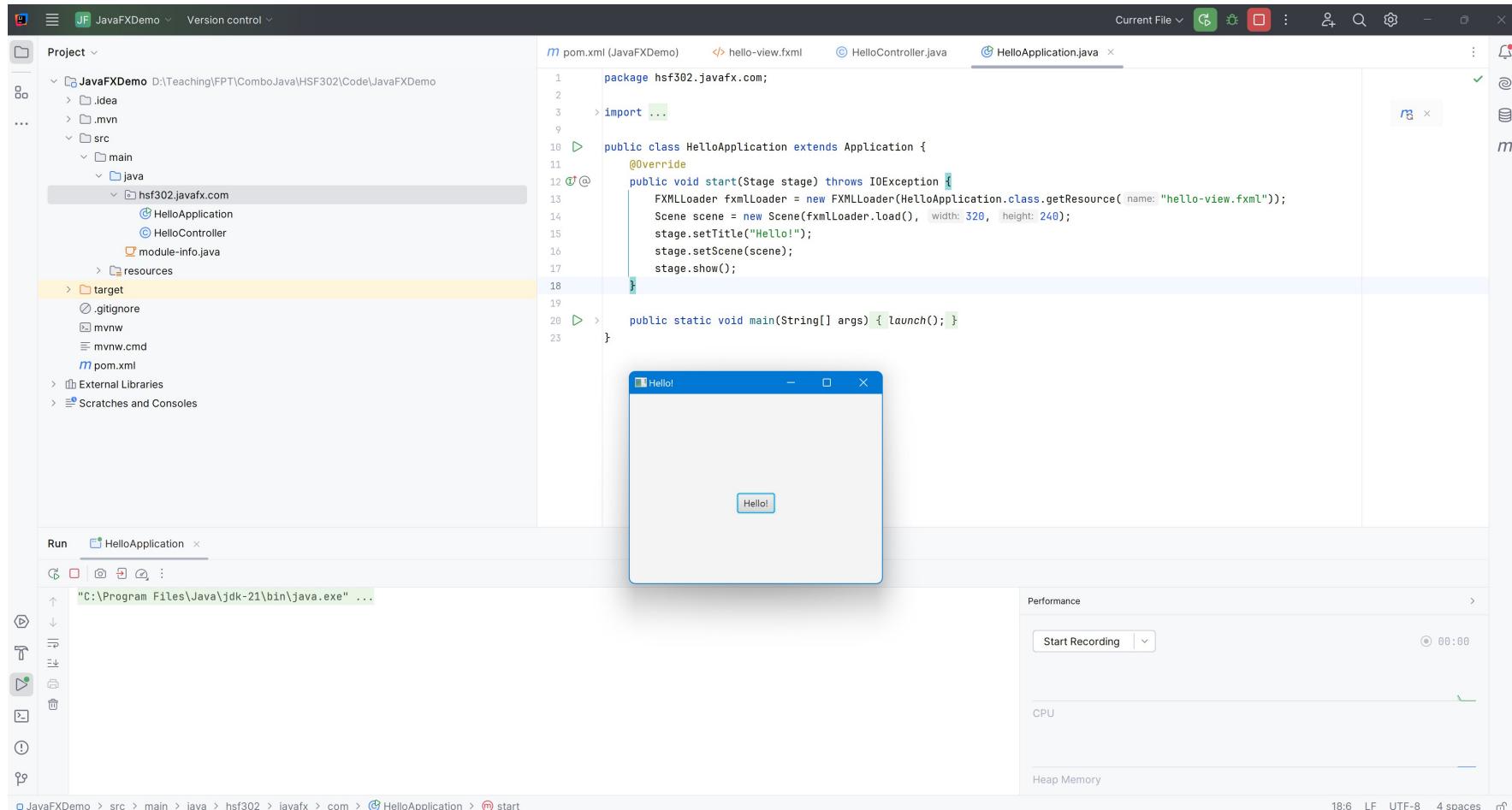
Open IntelliJ, File | New | Maven Project



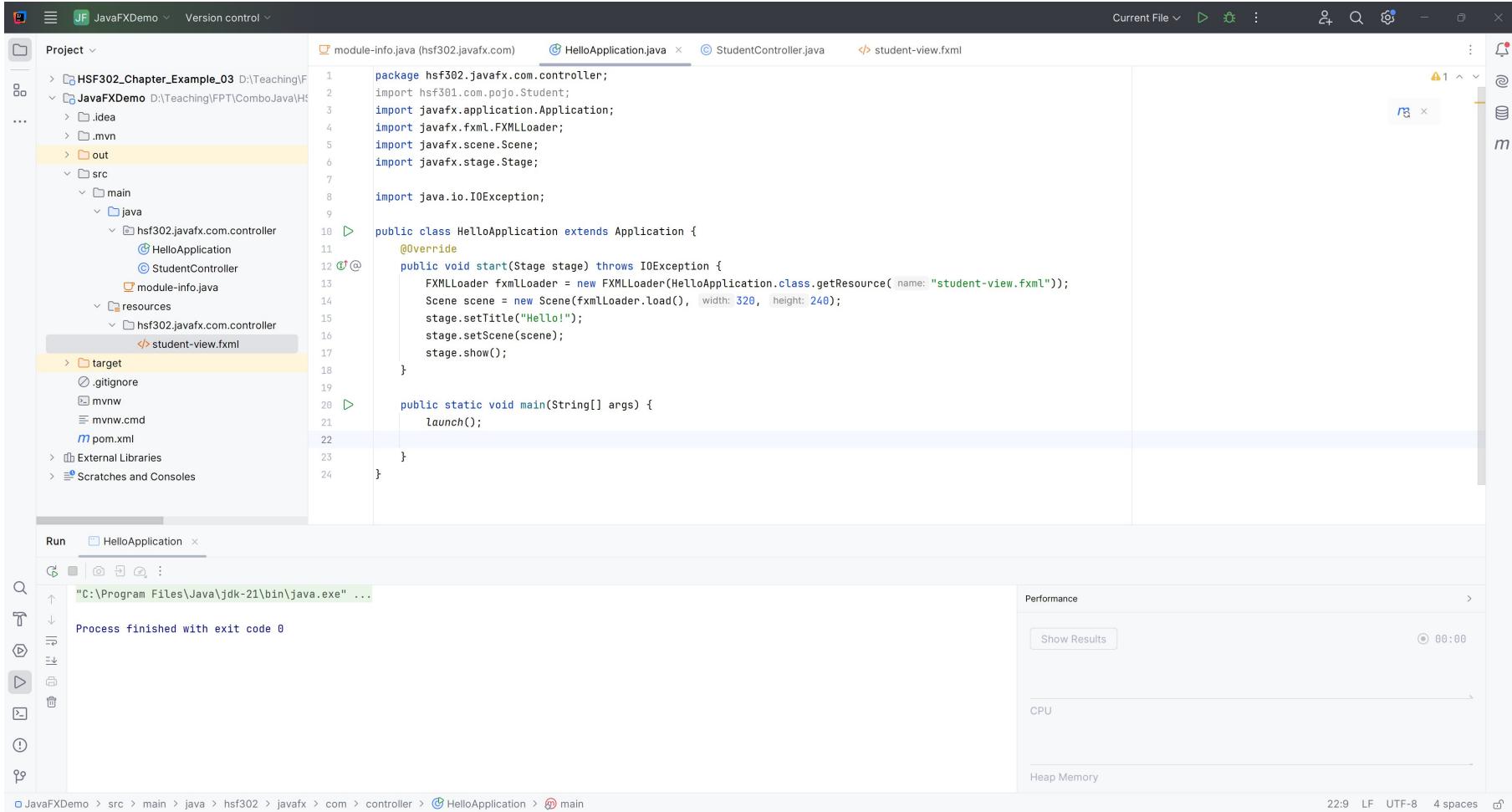
Check FormFX



Run Project



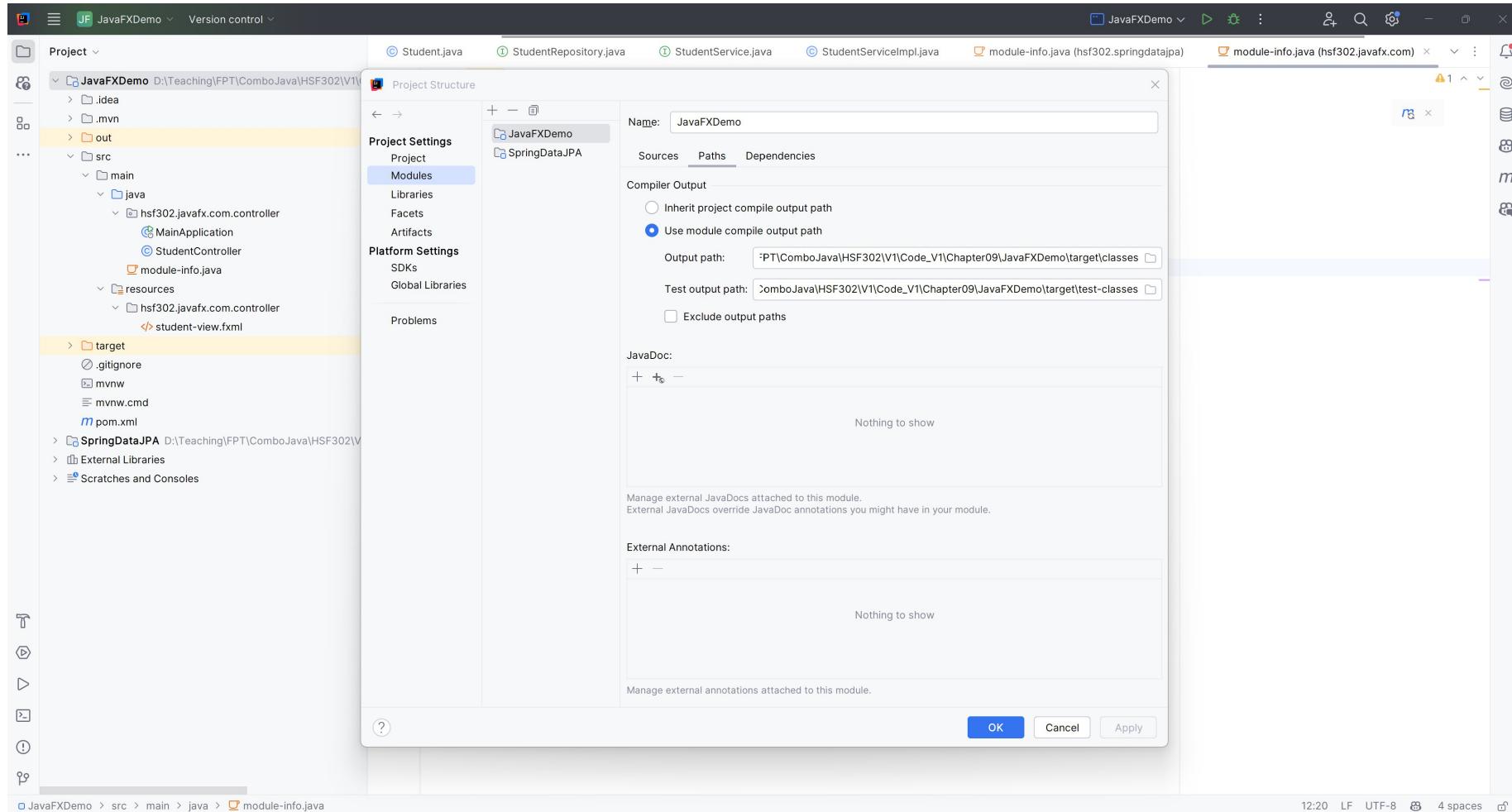
Build the Architecture



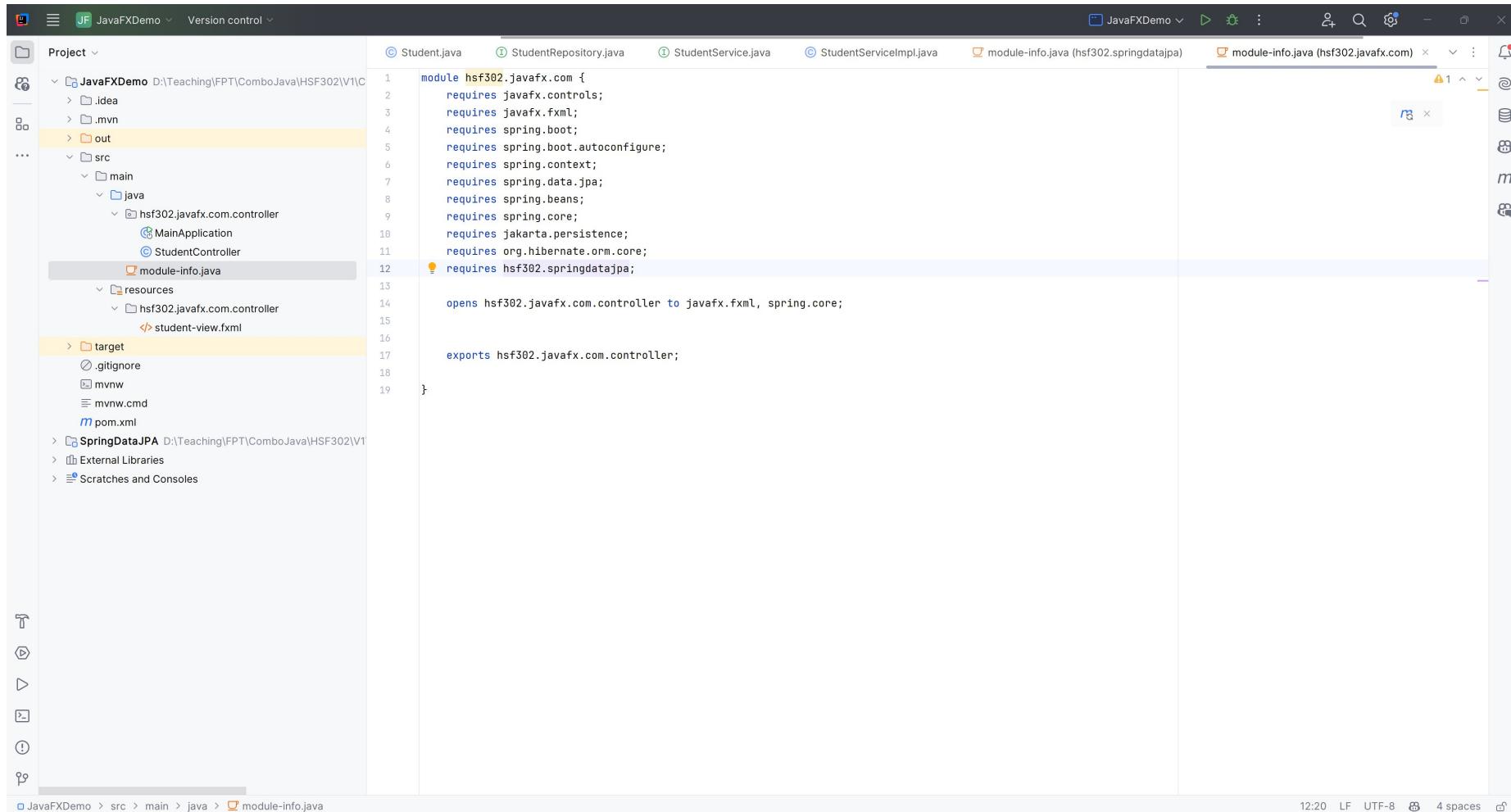
The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The left sidebar shows the project structure for "JavaFDemo". It includes a "src" folder containing "main" and "java" subfolders. The "java" folder contains "hsf302.java" and "hsf302.java.controller" packages. "hsf302.java.controller" contains "HelloApplication" and "StudentController" classes, along with "module-info.java" and "student-view.fxml".
- Code Editor:** The main editor window displays the "HelloApplication.java" file. The code defines a JavaFX application that loads a FXML file named "student-view.fxml" and shows it on a stage.
- Run Tab:** The bottom-left tab bar shows the "Run" tab is selected, with "HelloApplication" listed as the current configuration.
- Output Tab:** The bottom-left tab bar shows the "Output" tab is selected, displaying the message "Process finished with exit code 0".
- Performance Monitor:** The bottom-right panel shows performance monitoring for CPU and Heap Memory.
- Status Bar:** The bottom status bar shows the file path "JavaFDemo > src > main > java > hsf302 > javafx > com > controller > HelloApplication > main", and the file statistics "22:9 LF UTF-8 4 spaces ⌂".

Add Reference Another Project



Edit module-info.java in JavaFXDemo



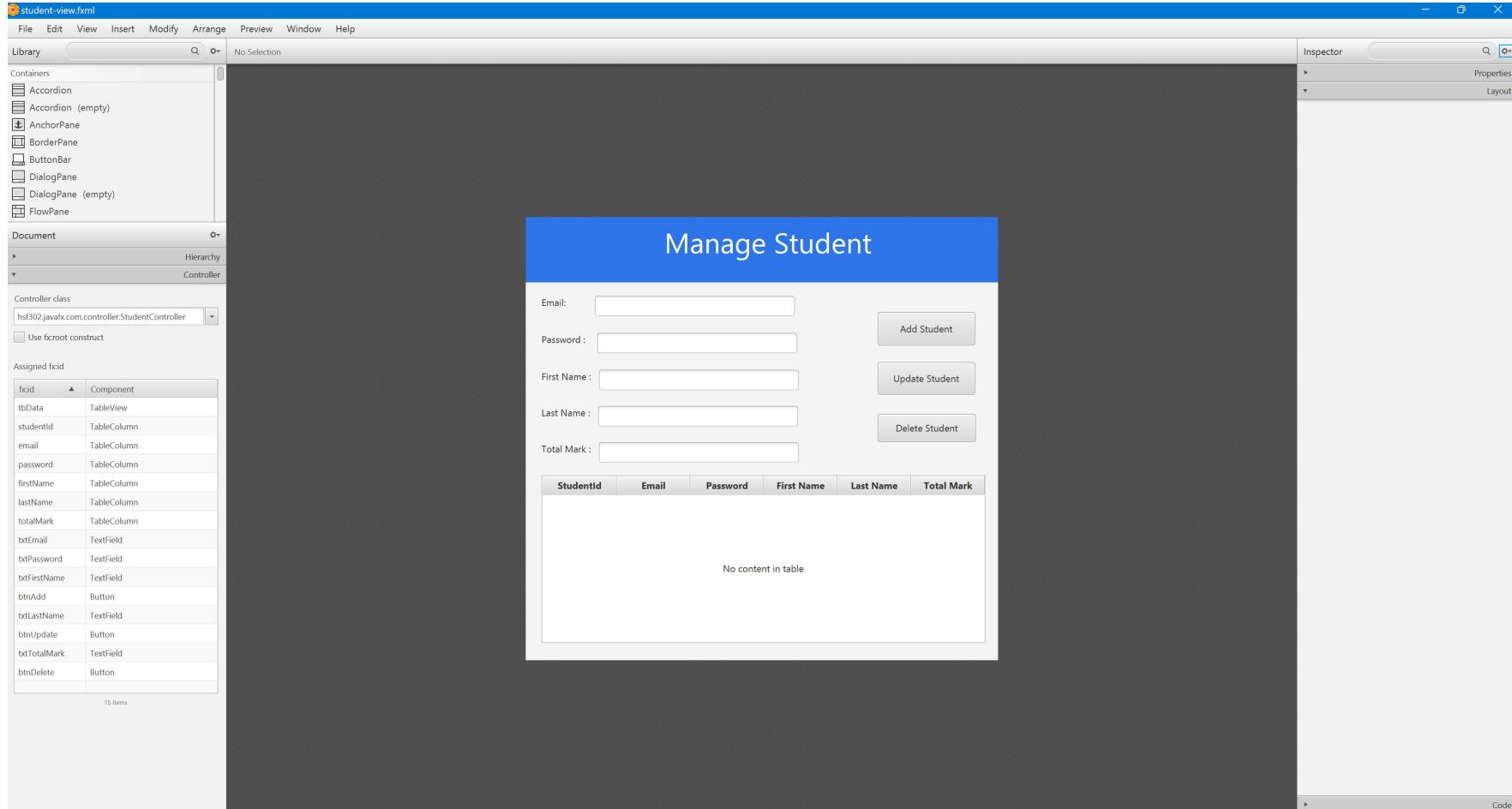
The screenshot shows the IntelliJ IDEA interface with the project 'JavaFXDemo' open. The 'module-info.java' file is selected in the editor. The code defines a module named 'hsf302.javafx.com' with various dependencies and exports.

```
1 module hsf302.javafx.com {  
2     requires javafx.controls;  
3     requires javafx.fxml;  
4     requires spring.boot;  
5     requires spring.boot.autoconfigure;  
6     requires spring.context;  
7     requires spring.data.jpa;  
8     requires spring.beans;  
9     requires spring.core;  
10    requires jakarta.persistence;  
11    requires org.hibernate.orm.core;  
12    requires hsf302.springdatajpa;  
13  
14    opens hsf302.javafx.com.controller to javafx.fxml, spring.core;  
15  
16  
17    exports hsf302.javafx.com.controller;  
18  
19 }
```

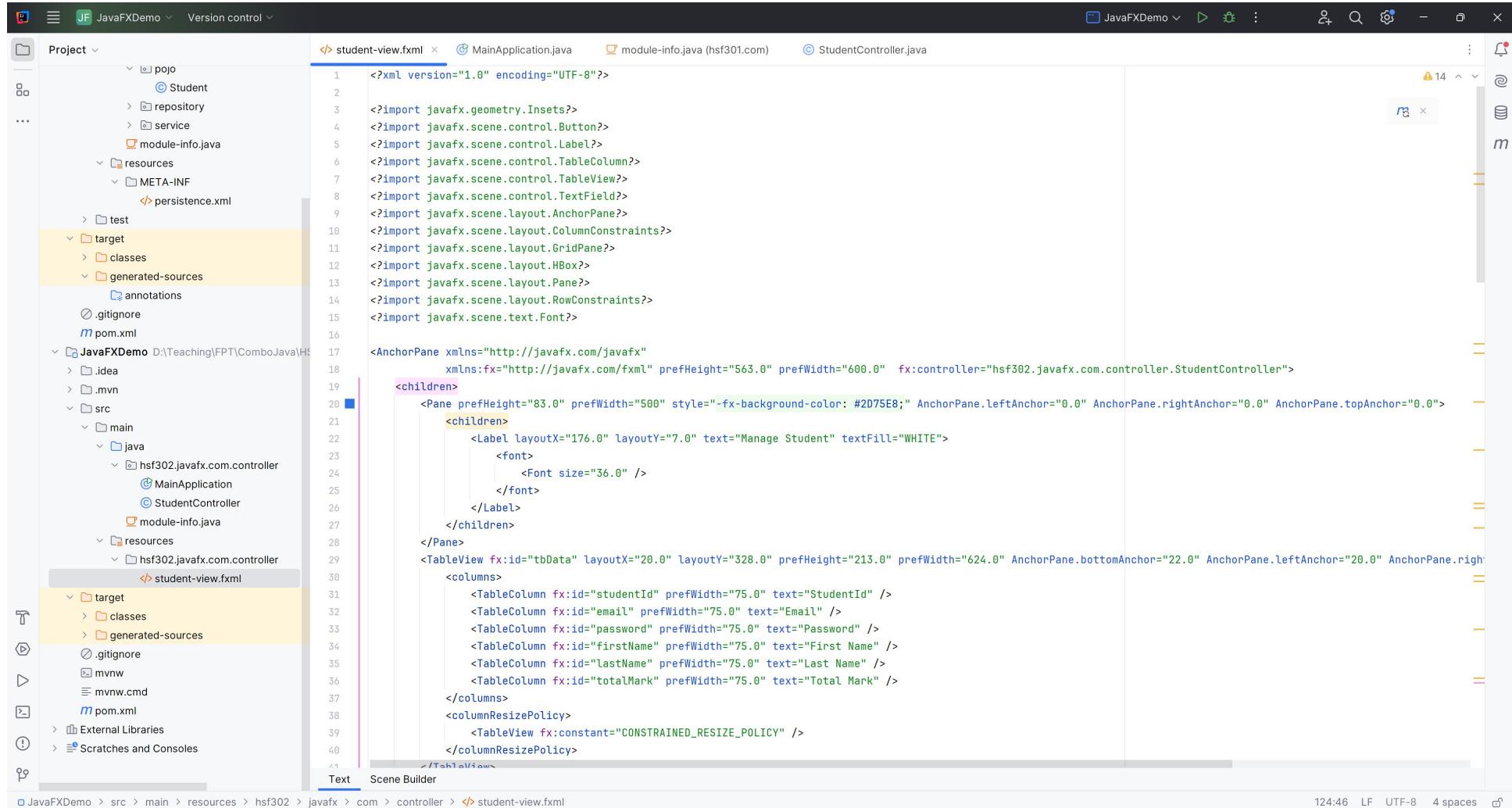
The project structure on the left shows the following directory tree:

- JavaFXDemo (Project)
- .idea
- .mvn
- out
- src
 - main
 - java
 - hsf302.javafx.com.controller
 - MainApplication
 - StudentController
 - module-info.java
 - student-view.fxml
 - resources
 - hsf302.javafx.com.controller
 - target
 - .gitignore
 - mvnw
 - mvnw.cmd
 - pom.xml- SpringDataJPA
- External Libraries
- Scratches and Consoles

Open Student-view.fxml in SceneBuilder



Edit student-view.fxml



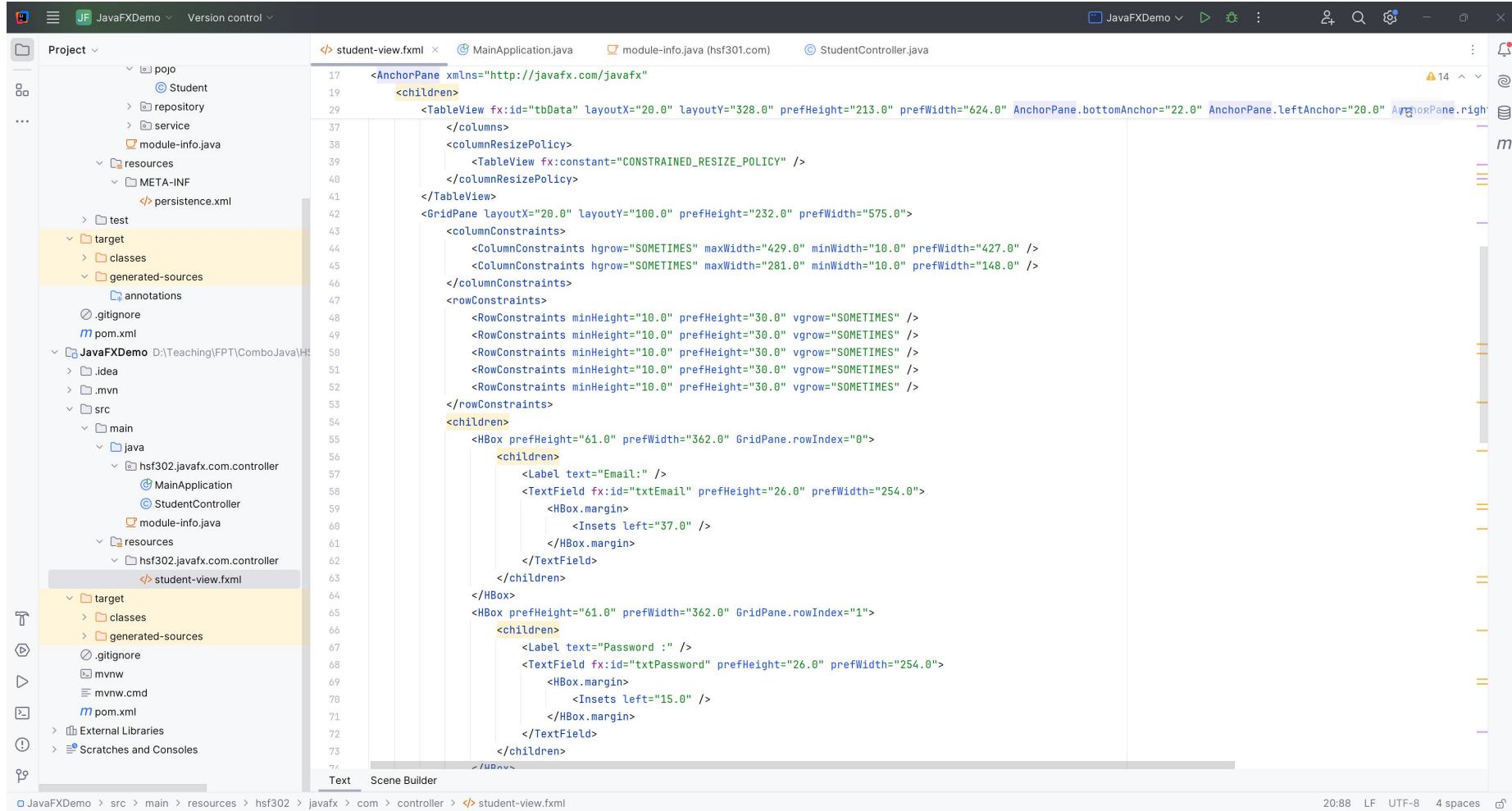
The screenshot shows the IntelliJ IDEA interface with the JavaFXDemo project open. The left sidebar displays the project structure, including the generated-sources folder under resources. The main editor window shows the student-view.fxml file, which defines an AnchorPane containing a Label and a TableView.

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.layout.Pane?>
<?import javafx.scene.layout.RowConstraints?>
<?import javafx.scene.text.Font?>

<AnchorPane xmlns="http://javafx.com/javafx"
             xmlns:fx="http://javafx.com/fxml" prefHeight="563.0" prefWidth="600.0" fx:controller="hsf302.java.com.controller.StudentController">
    <children>
        <Pane prefHeight="83.0" prefWidth="500" style="-fx-background-color: #2D75E8;" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">
            <children>
                <Label layoutX="176.0" layoutY="7.0" text="Manage Student" textFill="WHITE">
                    <font>
                        <Font size="36.0" />
                    </font>
                </Label>
            </children>
        </Pane>
        <TableView fx:id="tbData" layoutX="20.0" layoutY="328.0" prefHeight="213.0" prefWidth="624.0" AnchorPane.bottomAnchor="22.0" AnchorPane.leftAnchor="20.0" AnchorPane.rightAnchor="500.0" columnResizePolicy="CONSTRAINED_RESIZE_POLICY">
            <columns>
                <TableColumn fx:id="studentId" prefWidth="75.0" text="StudentId" />
                <TableColumn fx:id="email" prefWidth="75.0" text="Email" />
                <TableColumn fx:id="password" prefWidth="75.0" text="Password" />
                <TableColumn fx:id="firstName" prefWidth="75.0" text="First Name" />
                <TableColumn fx:id="lastName" prefWidth="75.0" text="Last Name" />
                <TableColumn fx:id="totalMark" prefWidth="75.0" text="Total Mark" />
            </columns>
            <columnResizePolicy>
                <TableView fx:constant="CONSTRAINED_RESIZE_POLICY" />
            </columnResizePolicy>
        </TableView>
    </children>

```

Edit student-view.fxml



The screenshot shows the IntelliJ IDEA interface with the FXML editor open. The left panel displays the project structure for 'JavaFDemo' under 'src/main/resources'. The 'student-view.fxml' file is selected in the tree, and its content is shown in the main editor area. The code defines an AnchorPane with children: a TableView, a GridPane, and two HBox containers for email and password fields.

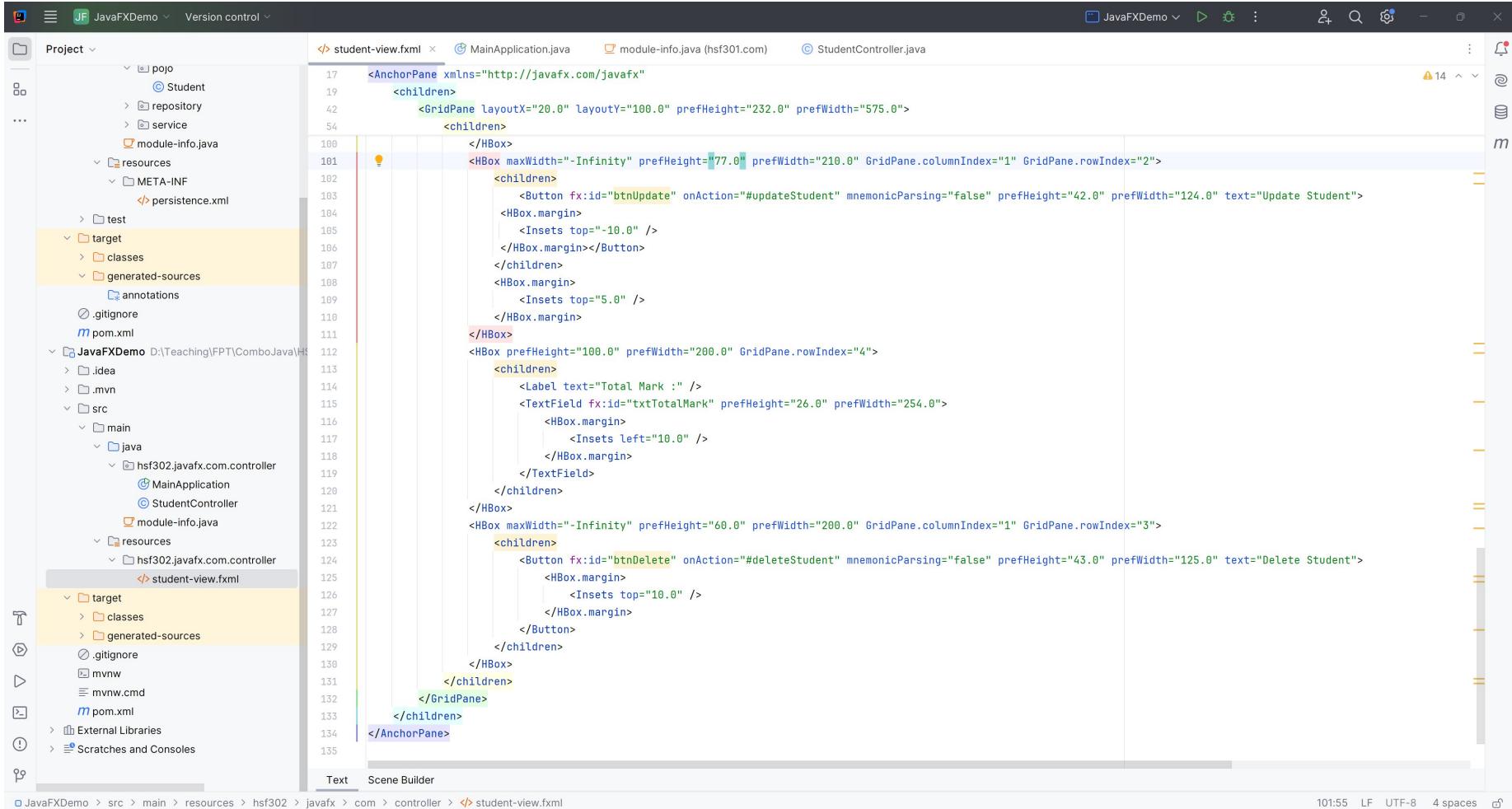
```
<AnchorPane xmlns="http://javafx.com/javafx"
    <children>
        <TableView fx:id="tbData" layoutX="20.0" layoutY="328.0" prefHeight="213.0" prefWidth="624.0" AnchorPane.bottomAnchor="22.0" AnchorPane.leftAnchor="20.0" AnchorPane.rightAnchor="14.0">
            <columns>
                <columnResizePolicy>
                    <TableColumnConstraints fx:constant="CONSTRAINED_RESIZE_POLICY" />
                </columnResizePolicy>
            </columns>
        </TableView>
        <GridPane layoutX="20.0" layoutY="100.0" prefHeight="232.0" prefWidth="575.0">
            <columnConstraints>
                <ColumnConstraints hgrow="SOMETIMES" maxWidth="429.0" minWidth="10.0" prefWidth="427.0" />
                <ColumnConstraints hgrow="SOMETIMES" maxWidth="281.0" minWidth="10.0" prefWidth="148.0" />
            </columnConstraints>
            <rowConstraints>
                <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
            </rowConstraints>
            <children>
                <HBox prefHeight="61.0" prefWidth="362.0" GridPane.rowIndex="0">
                    <children>
                        <Label text="Email:" />
                        <TextField fx:id="txtEmail" prefHeight="26.0" prefWidth="254.0">
                            <HBox.margin>
                                <Insets left="37.0" />
                            </HBox.margin>
                        </TextField>
                    </children>
                </HBox>
                <HBox prefHeight="61.0" prefWidth="362.0" GridPane.rowIndex="1">
                    <children>
                        <Label text="Password :" />
                        <PasswordField fx:id="txtPassword" prefHeight="26.0" prefWidth="254.0">
                            <HBox.margin>
                                <Insets left="15.0" />
                            </HBox.margin>
                        </PasswordField>
                    </children>
                </HBox>
            </children>
        </GridPane>
    </children>
</AnchorPane>
```

Edit student-view.fxml

The screenshot shows the JavaFX Scene Builder interface with the file `student-view.fxml` open. The left pane displays the project structure of `JavaFDemo`, which includes packages for `pojo`, `repository`, `service`, and `resources`, along with files like `module-info.java`, `persistence.xml`, and `StudentController.java`. The right pane shows the XML code for the FXML file, which defines an `AnchorPane` with a `GridPane` layout containing several `HBox` and `TextField` elements. The code is color-coded for syntax highlighting.

```
<AnchorPane xmlns="http://javafx.com/javafx"
    <children>
        <GridPane layout="20.0" layoutY="100.0" prefHeight="232.0" prefWidth="575.0">
            <children>
                <HBox prefHeight="61.0" prefWidth="362.0" GridPane.rowIndex="1">
                    <children>
                        <Label text="Password : " />
                        <TextField fx:id="txtPassword" prefHeight="26.0" prefWidth="254.0">
                            <HBox.margin>
                                <Insets left="15.0" />
                            </HBox.margin>
                        </TextField>
                    </children>
                </HBox>
                <HBox prefHeight="61.0" prefWidth="362.0" GridPane.rowIndex="2">
                    <children>
                        <Label text="First Name : " />
                        <TextField fx:id="txtFirstName" prefHeight="26.0" prefWidth="254.0">
                            <HBox.margin>
                                <Insets left="10.0" />
                            </HBox.margin>
                        </TextField>
                    </children>
                </HBox>
                <HBox maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="100.0" prefWidth="200.0" GridPane.getColumnIndex="1" GridPane.rowIndex="1">
                    <children>
                        <Button fx:id="btnAdd" onAction="#addStudent" mnemonicParsing="false" prefHeight="43.0" prefWidth="124.0" text="Add Student" />
                    </children>
                </HBox>
                <HBox prefHeight="100.0" prefWidth="200.0" GridPane.rowIndex="3">
                    <children>
                        <Label text="Last Name : " />
                        <TextField fx:id="txtLastName" prefHeight="26.0" prefWidth="254.0">
                            <HBox.margin>
                                <Insets left="10.0" />
                            </HBox.margin>
                        </TextField>
                    </children>
                </HBox>
            </children>
        </GridPane>
    </children>
</AnchorPane>
```

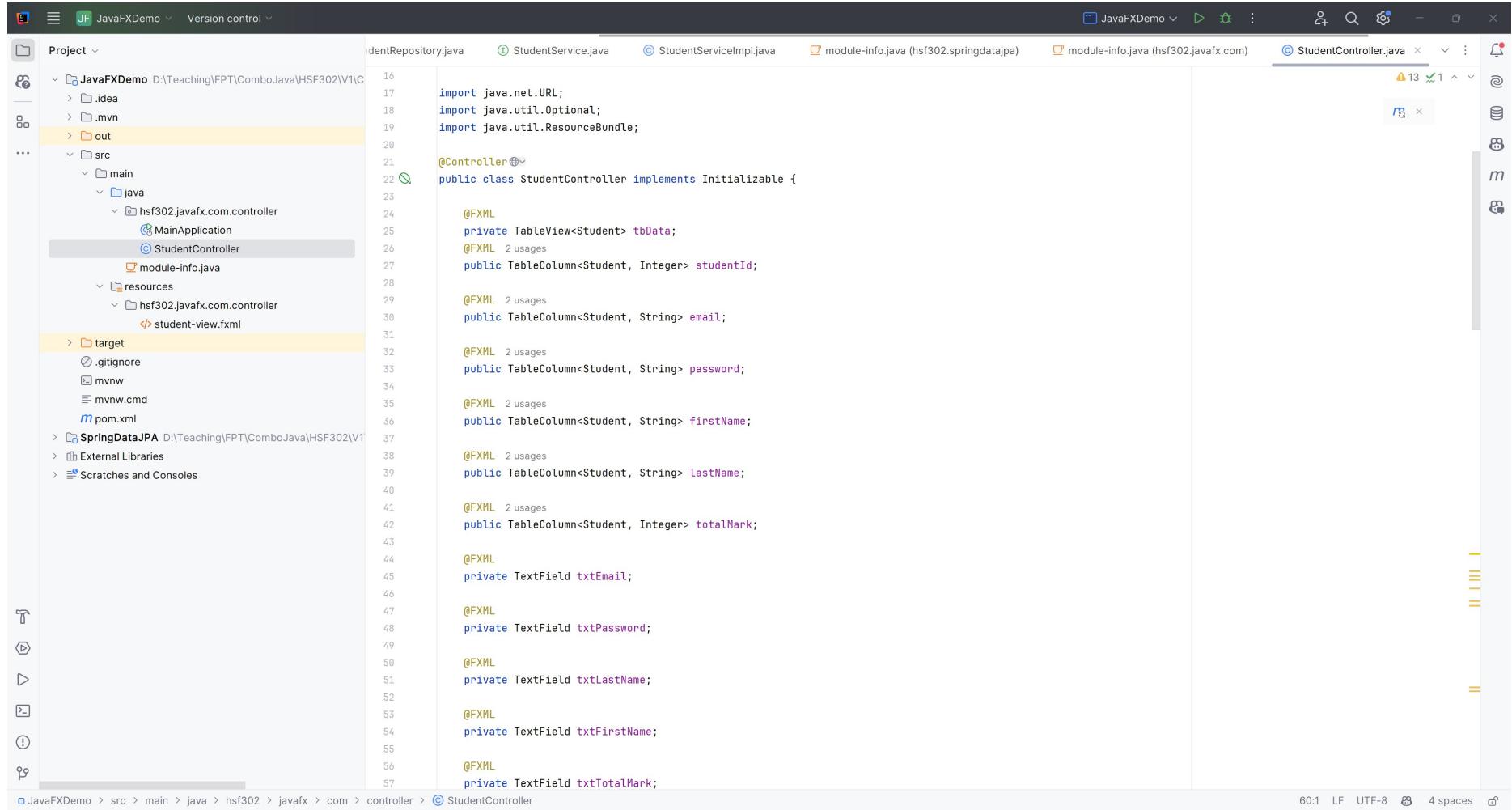
Edit student-view.fxml



The screenshot shows the IntelliJ IDEA interface with the JavaFXDemo project open. The left sidebar displays the project structure, including the generated-sources directory which contains the student-view.fxml file. The main editor window shows the FXML code for this file. The code defines an AnchorPane with various children, including a GridPane and several HBox components containing buttons and text fields. The code uses FXML syntax with fx:id and onAction attributes to define the UI elements and their behavior.

```
<AnchorPane xmlns="http://javafx.com/javafx"
    <children>
        <GridPane layoutX="20.0" layoutY="100.0" prefHeight="232.0" prefWidth="575.0">
            <children>
                <HBox>
                    <HBox maxWidth="-Infinity" prefHeight="77.0" prefWidth="210.0" GridPane.columnIndex="1" GridPane.rowIndex="2">
                        <children>
                            <Button fx:id="btnUpdate" onAction="#updateStudent" mnemonicParsing="false" prefHeight="42.0" prefWidth="124.0" text="Update Student">
                                <HBox.margin>
                                    <Insets top="-10.0" />
                                </HBox.margin>
                            </Button>
                        </children>
                        <HBox.margin>
                            <Insets top="5.0" />
                        </HBox.margin>
                    </HBox>
                    <HBox prefHeight="100.0" prefWidth="200.0" GridPane.rowIndex="4">
                        <children>
                            <Label text="Total Mark :" />
                            <TextField fx:id="txtTotalMark" prefHeight="26.0" prefWidth="254.0">
                                <HBox.margin>
                                    <Insets left="10.0" />
                                </HBox.margin>
                            </TextField>
                        </children>
                    </HBox>
                    <HBox maxWidth="-Infinity" prefHeight="60.0" prefWidth="200.0" GridPane.columnIndex="1" GridPane.rowIndex="3">
                        <children>
                            <Button fx:id="btnDelete" onAction="#deleteStudent" mnemonicParsing="false" prefHeight="43.0" prefWidth="125.0" text="Delete Student">
                                <HBox.margin>
                                    <Insets top="10.0" />
                                </HBox.margin>
                            </Button>
                        </children>
                    </HBox>
                </GridPane>
            </children>
        </AnchorPane>
```

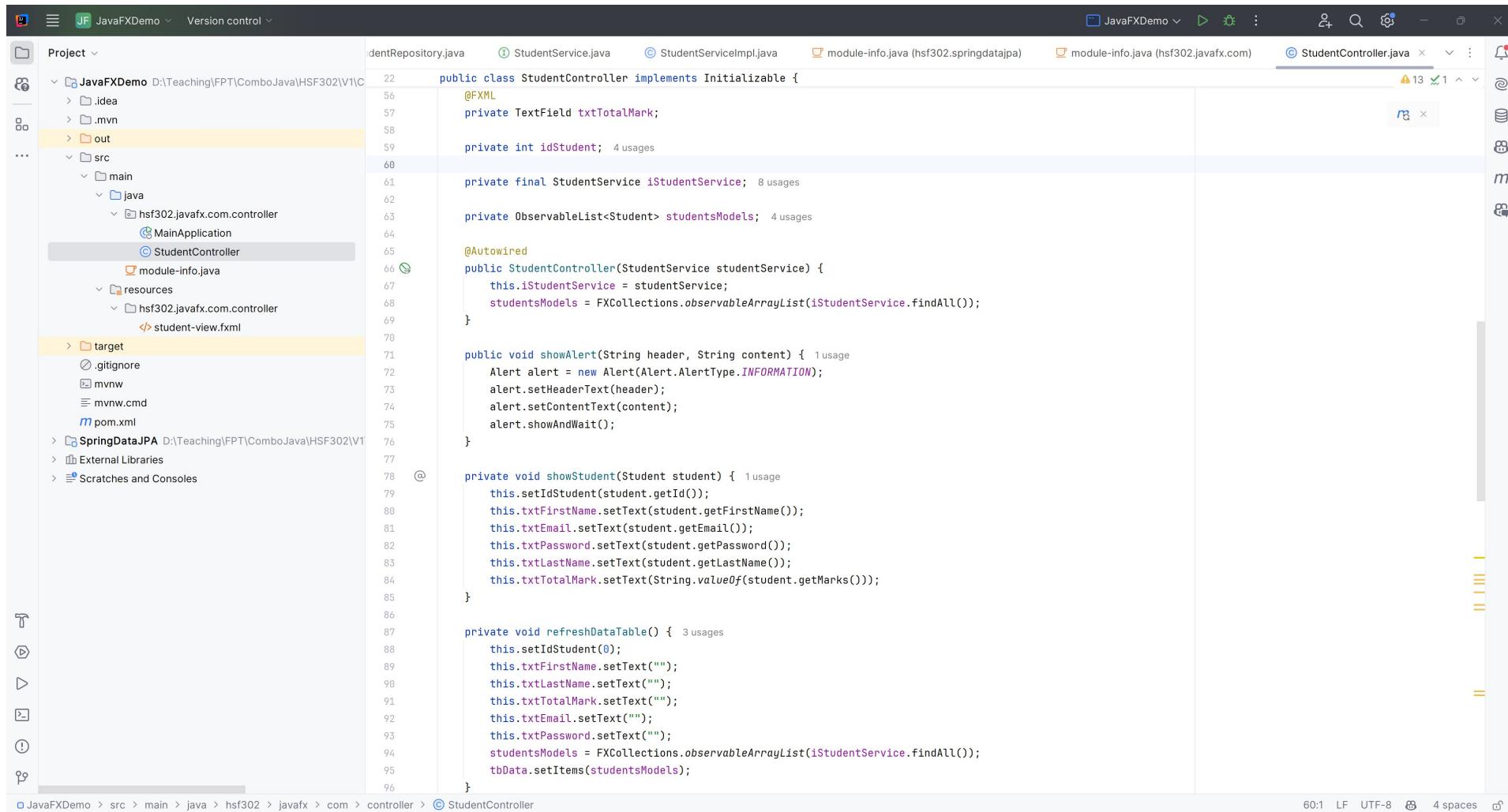
Edit the StudentController.java



The screenshot shows the IntelliJ IDEA interface with the project 'JavaFDemo' open. The left sidebar displays the project structure, including the 'src' directory which contains 'main', 'java', and 'hsf302.javaFX.com.controller'. Inside 'controller', the 'StudentController.java' file is selected and shown in the code editor. The code implements the `Initializable` interface and uses FXML annotations (`@FXML`) to map fields to UI components like `TableView`, `TableColumn`, and `TextField`. The code editor includes line numbers, code completion suggestions, and a status bar at the bottom.

```
16 import java.net.URL;
17 import java.util.Optional;
18 import java.util.ResourceBundle;
19
20 @Controller
21 public class StudentController implements Initializable {
22
23     @FXML
24     private TableView<Student> tbData;
25     @FXML 2 usages
26     public TableColumn<Student, Integer> studentId;
27
28     @FXML 2 usages
29     public TableColumn<Student, String> email;
30
31     @FXML 2 usages
32     public TableColumn<Student, String> password;
33
34     @FXML 2 usages
35     public TableColumn<Student, String> firstName;
36
37     @FXML 2 usages
38     public TableColumn<Student, String> lastName;
39
40     @FXML 2 usages
41     public TableColumn<Student, Integer> totalMark;
42
43     @FXML
44     private TextField txtEmail;
45
46     @FXML
47     private TextField txtPassword;
48
49     @FXML
50     private TextField txtLastName;
51
52     @FXML
53     private TextField txtFirstName;
54
55     @FXML
56     private TextField txtTotalMark;
```

Edit the StudentController.java



The screenshot shows the IntelliJ IDEA interface with the project structure on the left and the code editor on the right. The code editor displays the `StudentController.java` file.

```
public class StudentController implements Initializable {
    @FXML
    private TextField txtTotalMark;

    private int idStudent; 4 usages

    private final StudentService iStudentService; 8 usages

    private ObservableList<Student> studentsModels; 4 usages

    @Autowired
    public StudentController(StudentService studentService) {
        this.iStudentService = studentService;
        studentsModels = FXCollections.observableArrayList(iStudentService.findAll());
    }

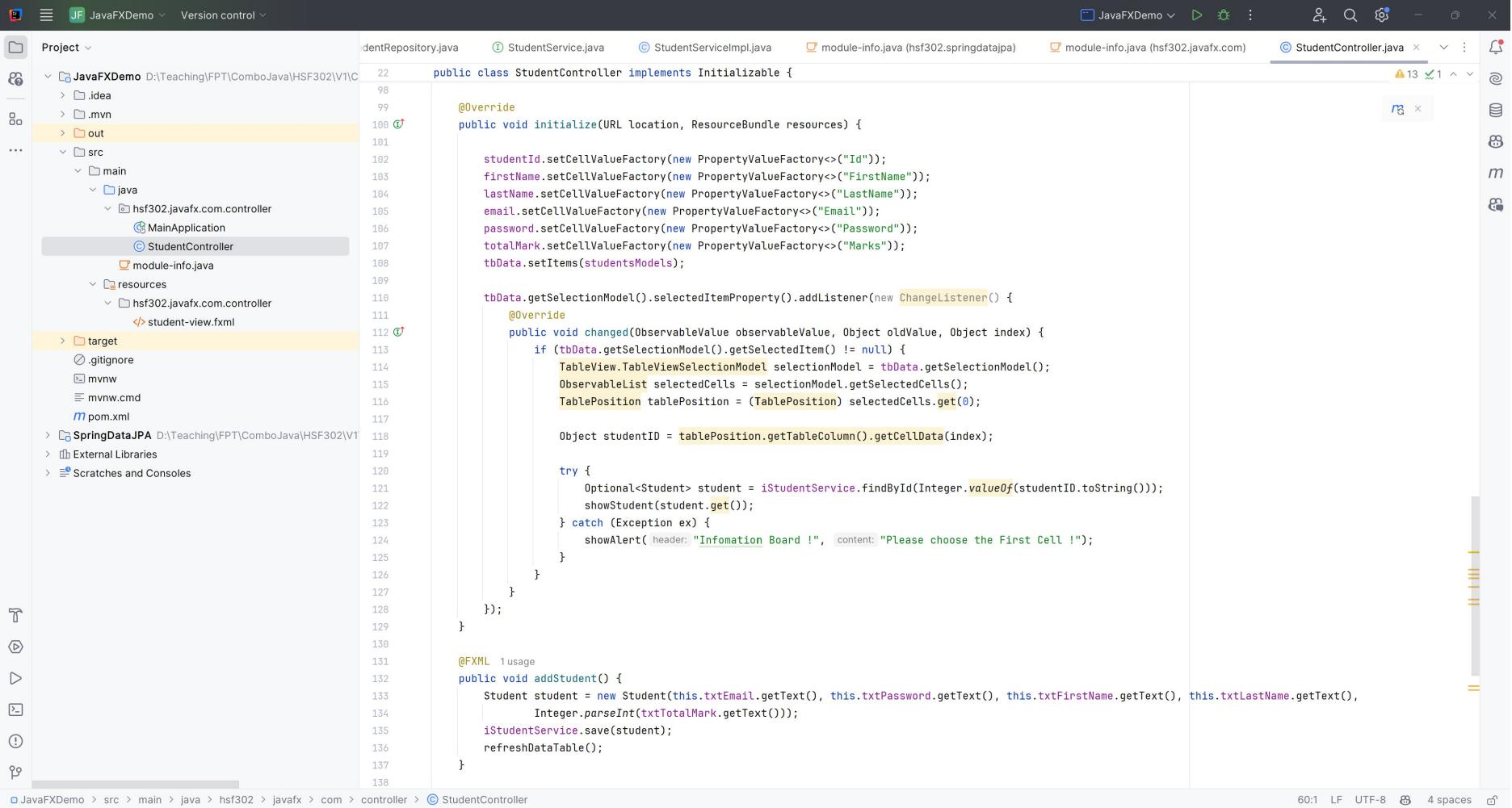
    public void showAlert(String header, String content) { 1 usage
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setHeaderText(header);
        alert.setContentText(content);
        alert.showAndWait();
    }

    private void showStudent(Student student) { 1 usage
        this.setIdStudent(student.getId());
        this.txtFirstName.setText(student.getFirstName());
        this.txtEmail.setText(student.getEmail());
        this.txtPassword.setText(student.getPassword());
        this.txtLastName.setText(student.getLastName());
        this.txtTotalMark.setText(String.valueOf(student.getMarks()));
    }

    private void refreshDataTable() { 3 usages
        this.setIdStudent(0);
        this.txtFirstName.setText("");
        this.txtLastName.setText("");
        this.txtTotalMark.setText("");
        this.txtEmail.setText("");
        this.txtPassword.setText("");
        studentsModels = FXCollections.observableArrayList(iStudentService.findAll());
        tbData.setItems(studentsModels);
    }
}
```

The code implements the `Initializable` interface and uses `@FXML` annotations to bind JavaFX controls to fields. It also uses `@Autowired` to inject the `iStudentService`. The `showAlert` method creates an `Alert` dialog. The `showStudent` method sets the values of several text fields. The `refreshDataTable` method clears the text fields and updates the observable list of students.

Edit the StudentController.java



The screenshot shows the IntelliJ IDEA IDE interface with the following details:

- Project Structure:** The left sidebar shows the project structure for "JavaFDemo". It includes a .idea folder, a .mvn folder, an out folder (highlighted in yellow), a src folder containing main, java, and resources. The resources folder contains hsf302.fxml and student-view.fxml. A module-info.java file is also present. The "StudentController" class is selected and highlighted in the list.
- Code Editor:** The right pane displays the "StudentController.java" code. The code implements the Initializable interface and overrides the initialize method. It initializes various PropertyValueFactories for studentId, firstName, lastName, email, password, totalMark, and tbData. It adds a ChangeListener to tbData's selection model to handle item changes. It also handles the addStudent button's click event by creating a new Student object, saving it to the database, and refreshing the data table.
- Status Bar:** The bottom status bar shows the file path as "JavaFDemo > src > main > java > hsf302 > javafx > com > controller > StudentController.java", and the bottom right corner shows "60:1 LF UTF-8 4 spaces".

```
public class StudentController implements Initializable {
    @Override
    public void initialize(URL location, ResourceBundle resources) {
        studentId.setCellValueFactory(new PropertyValueFactory<>("Id"));
        firstName.setCellValueFactory(new PropertyValueFactory<>("FirstName"));
        lastName.setCellValueFactory(new PropertyValueFactory<>("LastName"));
        email.setCellValueFactory(new PropertyValueFactory<>("Email"));
        password.setCellValueFactory(new PropertyValueFactory<>("Password"));
        totalMark.setCellValueFactory(new PropertyValueFactory<>("Marks"));
        tbData.setItems(studentsModels);
    }

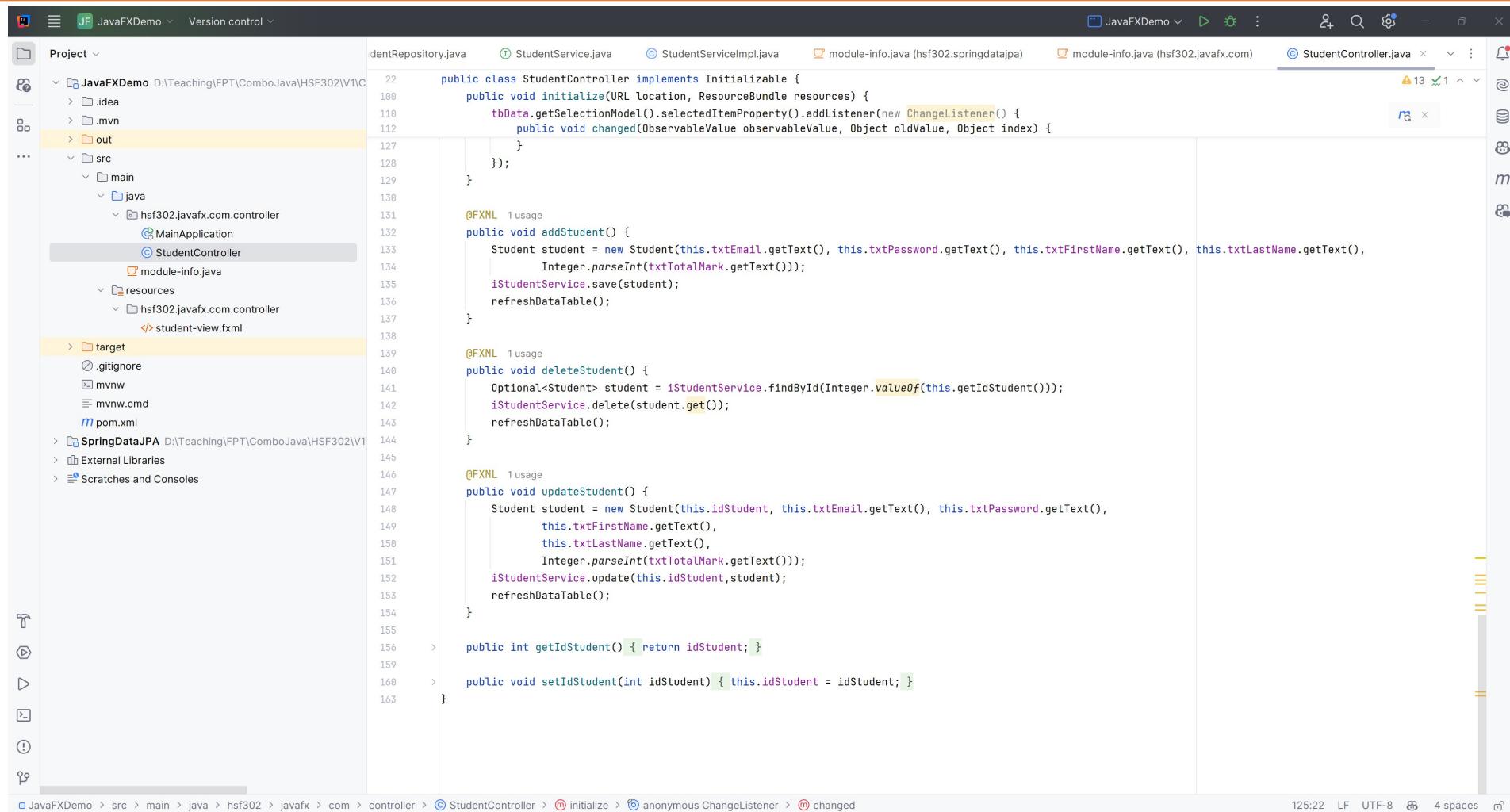
    tbData.getSelectionModel().selectedItemProperty().addListener(new ChangeListener<Object>() {
        @Override
        public void changed(ObservableValue observableValue, Object oldValue, Object index) {
            if (tbData.getSelectionModel().getSelectedItem() != null) {
                TableView.TableViewSelectionModel selectionModel = tbData.getSelectionModel();
                ObservableList selectedCells = selectionModel.getSelectedCells();
                TablePosition tablePosition = (TablePosition) selectedCells.get(0);

                Object studentID = tablePosition.getTableColumn().getCellData(index);

                try {
                    Optional<Student> student = iStudentService.findById(Integer.valueOf(studentID.toString()));
                    showStudent(student.get());
                } catch (Exception ex) {
                    showAlert(header: "Infomation Board !", content: "Please choose the First Cell !");
                }
            }
        }
    });
}

@FXML 1 usage
public void addStudent() {
    Student student = new Student(this.txtEmail.getText(), this.txtPassword.getText(), this.txtFirstName.getText(), this.txtLastName.getText(),
        Integer.parseInt(txtTotalMark.getText()));
    iStudentService.save(student);
    refreshDataTable();
}
```

Edit the StudentController.java



The screenshot shows the IntelliJ IDEA IDE interface with the following details:

- Project Structure:** The left sidebar shows the project structure for "JavaFXDemo". The "src" folder is expanded, showing "main" and "java". Inside "java", there is a package "hsf302.javafx.com.controller" containing files "MainApplication.java", "StudentController.java", and "module-info.java". Below "java" is "resources" and "hsf302.javafx.com.controller" which contains "student-view.fxml".
- Code Editor:** The main window displays the content of "StudentController.java". The code implements the `Initializable` interface and contains methods for adding, deleting, and updating students. It uses `@FXML` annotations to map Java code to FXML controls.
- Status Bar:** The bottom status bar shows the file path: "JavaFXDemo > src > main > java > hsf302 > javafx > com > controller > StudentController.java", and various status indicators like "13 changes", "1 error", and "1 warning".
- Toolbars and Icons:** Standard IntelliJ IDEA toolbars and icons are visible along the top and right edges of the interface.

```
public class StudentController implements Initializable {
    public void initialize(URL location, ResourceBundle resources) {
        tbData.getSelectionModel().selectedItemProperty().addListener(new ChangeListener<ObservableValue<Object>>() {
            public void changed(ObservableValue<Object> observableValue, Object oldValue, Object index) {
                ...
            }
        });
    }

    @FXML 1 usage
    public void addStudent() {
        Student student = new Student(this.txtEmail.getText(), this.txtPassword.getText(), this.txtFirstName.getText(), this.txtLastName.getText(),
            Integer.parseInt(txtTotalMark.getText()));
        iStudentService.save(student);
        refreshDataTable();
    }

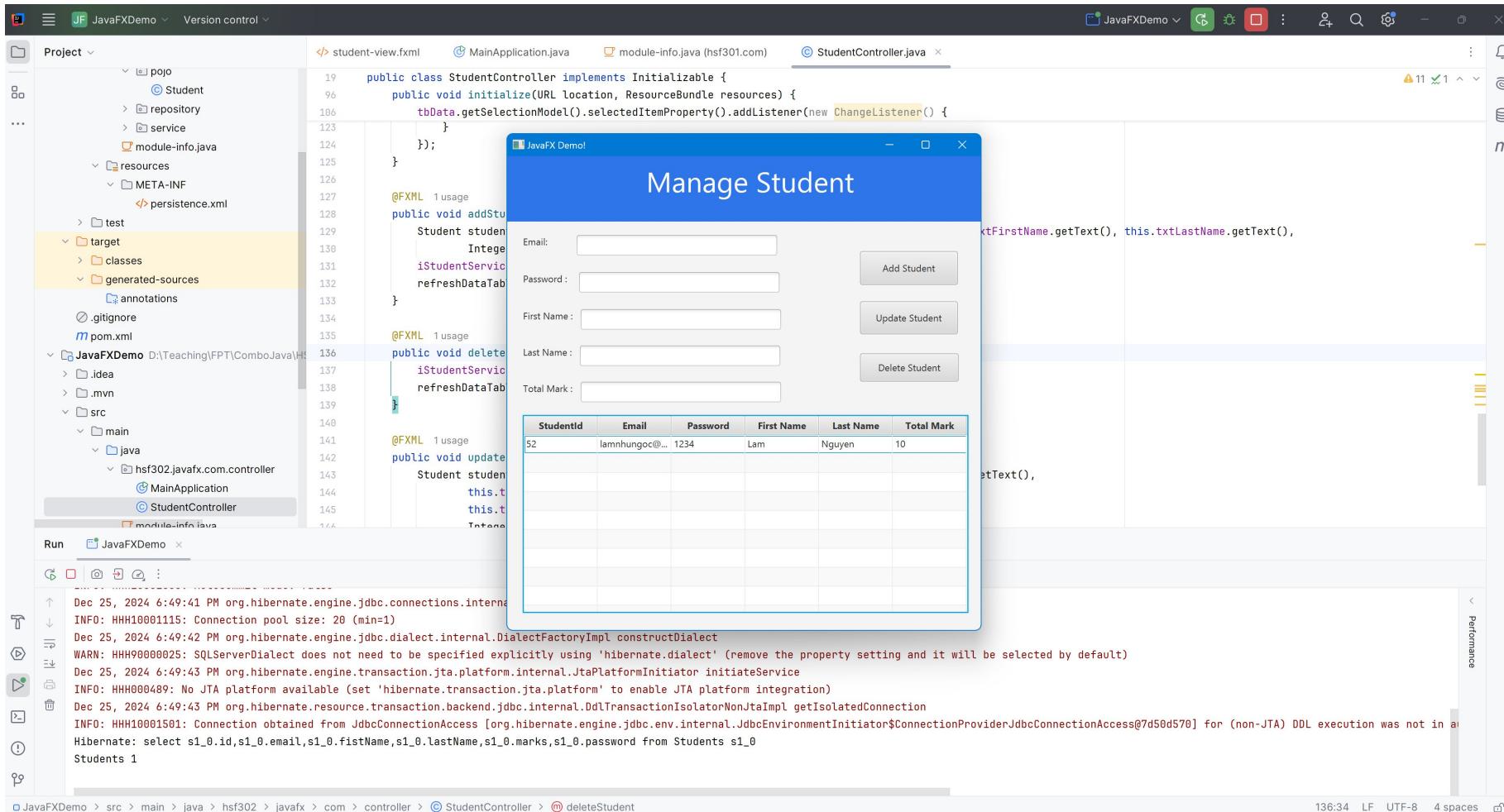
    @FXML 1 usage
    public void deleteStudent() {
        Optional<Student> student = iStudentService.findById(Integer.valueOf(this.getIdStudent()));
        iStudentService.delete(student.get());
        refreshDataTable();
    }

    @FXML 1 usage
    public void updateStudent() {
        Student student = new Student(this.idStudent, this.txtEmail.getText(), this.txtPassword.getText(),
            this.txtFirstName.getText(),
            this.txtLastName.getText(),
            Integer.parseInt(txtTotalMark.getText()));
        iStudentService.update(this.idStudent,student);
        refreshDataTable();
    }

    public int getIdStudent() { return idStudent; }

    public void setIdStudent(int idStudent) { this.idStudent = idStudent; }
}
```

Run Project



Summary

Concepts were introduced:

- Implement CRUD (Create, Read, Update, Delete) operations to manage data effectively.
- Utilize Hibernate to map Java objects to database tables, simplifying data persistence and retrieval.
- Design and develop a user-friendly interface using JavaFX, enabling seamless interaction with the application.
- Implement features such as form input validation, error handling, and feedback mechanisms to enhance user experience.