

Spring Boot Concepts

Objectives

- ◆ What is Spring Boot?
- ◆ Spring Version History
- ◆ Key features of Spring Boot
- ◆ Developing Your First Spring Boot Application

Spring Boot

What is Spring Boot ?

- Spring Boot is a powerful framework built on top of the Spring Framework that simplifies the development of Spring applications. It aims to reduce boilerplate code, configuration, and setup time, allowing developers to focus on building the core functionality of their applications.



Spring Version History

Spring Boot Version	Release Date	Spring framework version
1.0.0.RELEASE	Apr 2014	4.0.3
1.5.x	Jan 2017 - Aug 2019	4.3.6 - 4.3.25
2.0.0.RELEASE	Mar 2018	5.0.4
2.7.x	May 2022 - Aug 2023	5.3.20 - 5.3.29
3.0.0	Nov 2022	6.0.2
3.0.x		6.0.2 - 6.0.11
3.1.x	May 2023 - Now	
3.2.0	Nov 23, 2023	6.1
3.3.5	Oct 24, 2024	6.1.14
3.4.0	Nov 21, 2024	

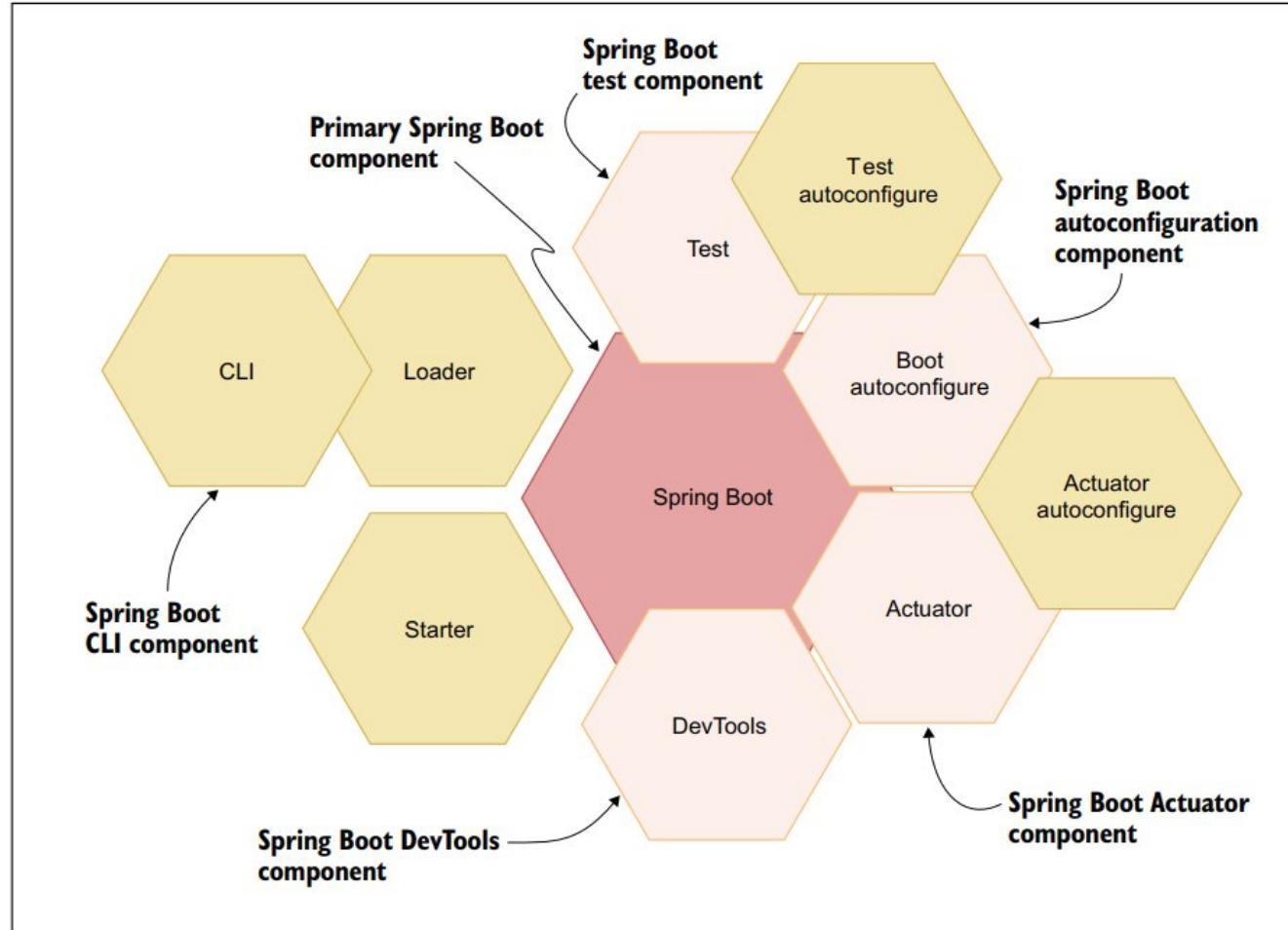
Spring Features

- ❖ **Autoconfiguration:** Spring Boot automatically configures your application based on the dependencies you include in your project.
- ❖ **Data Access:** Seamless integration with various data sources like relational databases, NoSQL databases, and JPA.
- ❖ **Stand-alone Applications:** Spring Boot allows you to create standalone applications that can be run directly without the need for external servers.
- ❖ **Embedded Servers:** Spring Boot includes embedded servers like Tomcat or Netty, so you can run your application without deploying it to a separate server.
- ❖ **Starter POMs:** Spring Boot provides starter POMs that simplify dependency management. You can easily include the necessary dependencies for specific functionalities, such as web development, database access, or security.

Benefits of Using Spring Boot

- ❖ **Rapid Application Development:** Quickly create and deploy applications.
- ❖ **Simplified Configuration:** Reduce boilerplate configuration.
- ❖ **Embedded Servers:** No need for external servers.
- ❖ **Production-Ready Features:** Built-in features for monitoring, metrics, and security.
- ❖ **Strong Community and Ecosystem:** Large community and extensive documentation.
- ❖ **Simplified deployment:** Package your applications as standalone JAR files or WAR files for easy deployment.
- ❖ **Cloud-native support:** Seamless integration with cloud platforms like AWS, Azure, and GCP.

Spring Boot Components



Spring Boot Components

- ◆ **Spring Boot:** This is the core of the framework. It provides the SpringApplication class (for creating standalone applications), support for embedded web servers (like Tomcat), and externalized configurations (like database connection details). Think of it as the foundation upon which the other components build.
- ◆ **Spring Boot CLI Component:** This is a developer-friendly command-line tool (CLI) that compiles and runs Groovy code. It can also watch for file changes and restart the application automatically, which makes prototyping Spring applications quicker and easier.
- ◆ **Spring Boot Test Component:** This enables the use of annotations and methods to write test cases for the Spring Boot application.

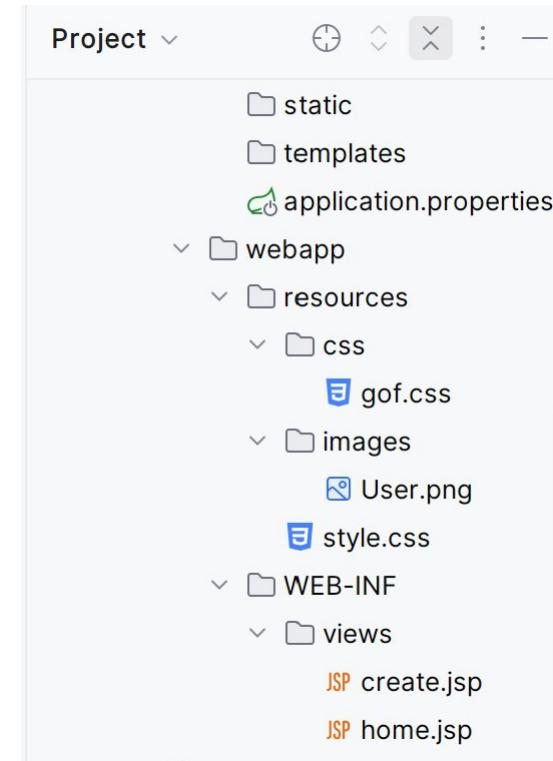
Spring Boot Components

- **Spring Boot DevTools Component:** This provides an additional toolkit aimed at improving the development experience. Features include automatic restart upon code changes and LiveReload to refresh the browser when HTML changes.
- **Spring Boot Autoconfiguration Component:** Autoconfiguration automatically configures the bare minimum components needed for a Spring application based on classpath JAR files and properties. It makes educated guesses to configure Spring beans based on the presence of dependencies.
- **Spring Boot Actuator Component:** This provides actuator endpoints to interact with, monitor, and audit a Spring Boot application. These endpoints offer health checks, thread dumps, and other useful metrics to manage the application in production.

Spring Boot Programming Demo

Key Points

- ◆ **Step 01 :** Create webapp in the src/main/webapp directory and its importance for JSP files.



Key Points

- ◆ Step 02: Config application.properties for JSP

```
spring.application.name=DemoLab01
spring.mvc.view.prefix=/WEB-INF/views/
spring.mvc.view.suffix=.jsp|
```

- ◆ Step 03 : Config the pom.xml

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    <scope>provided</scope>
</dependency>

<dependency>
    <groupId>org.apache.tomcat.embed</groupId>
    <artifactId>tomcat-embed-jasper</artifactId>
    <scope>provided</scope>
</dependency>
|
<dependency>
    <groupId>org.glassfish.web</groupId>
    <artifactId>jakarta.servlet.jsp.jstl</artifactId>
    <version>3.0.1</version>
</dependency>
```

Key Points

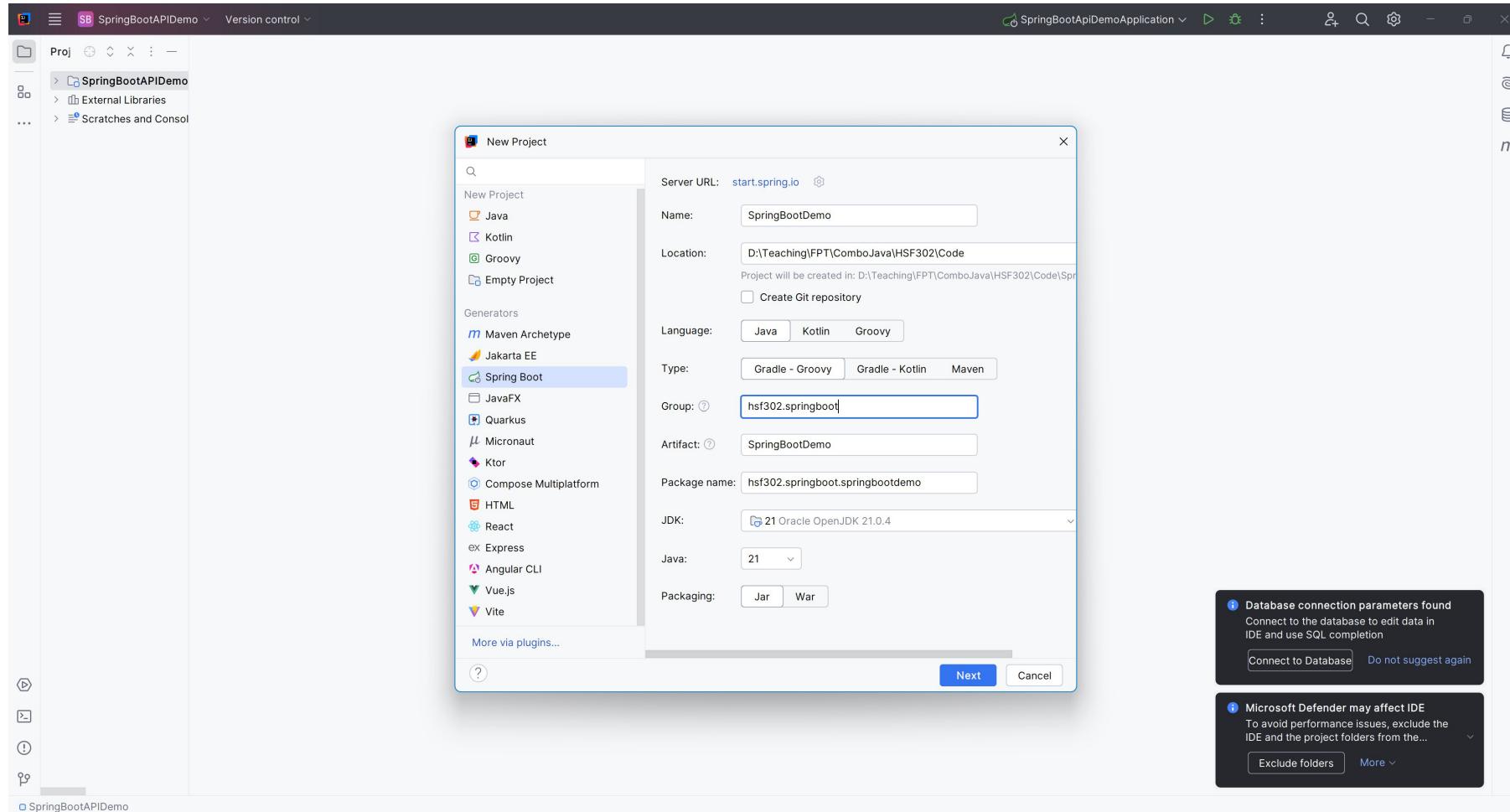


◆ Step 03 : Config the pom.xml

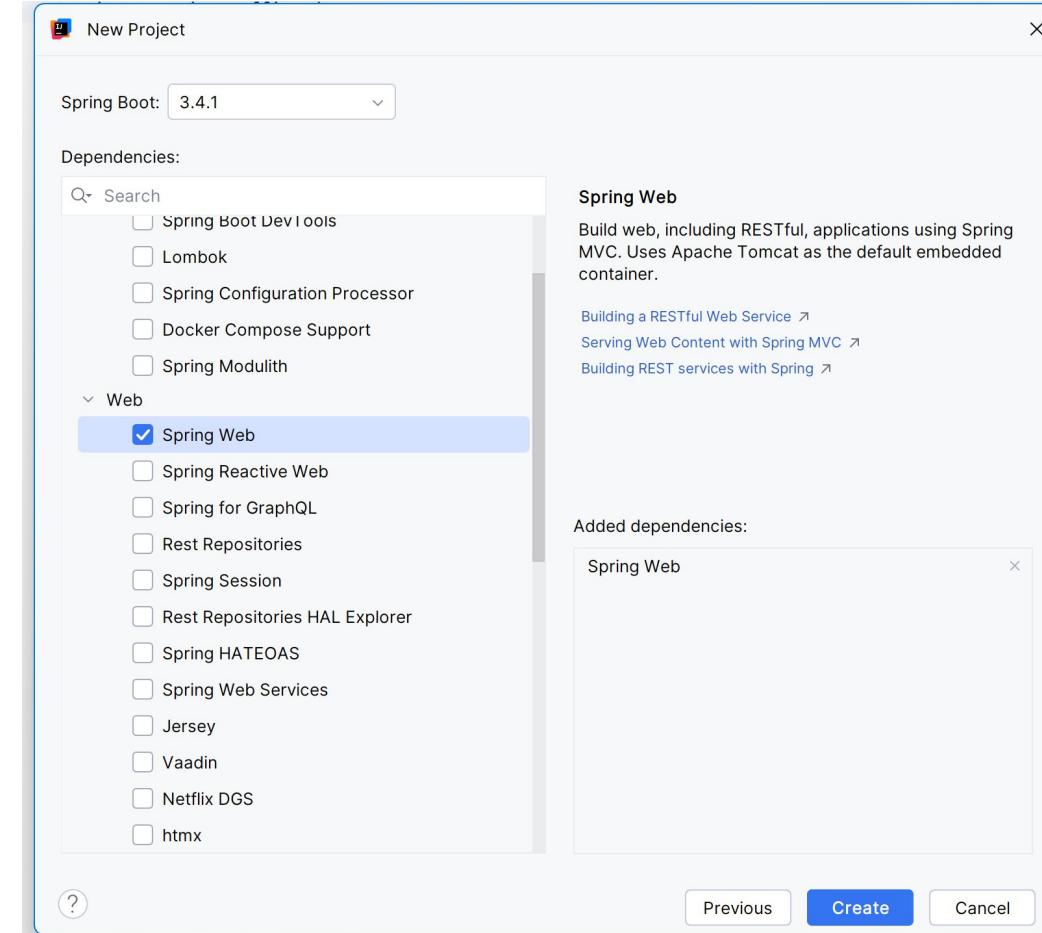
```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    <scope>provided</scope>
</dependency>

<dependency>
    <groupId>org.apache.tomcat.embed</groupId>
    <artifactId>tomcat-embed-jasper</artifactId>
    <scope>provided</scope>
</dependency>
|
<dependency>
    <groupId>org.glassfish.web</groupId>
    <artifactId>jakarta.servlet.jsp.jstl</artifactId>
    <version>3.0.1</version>
</dependency>
```

Open IntelliJ, File | New | Maven Project



Add Dependencies



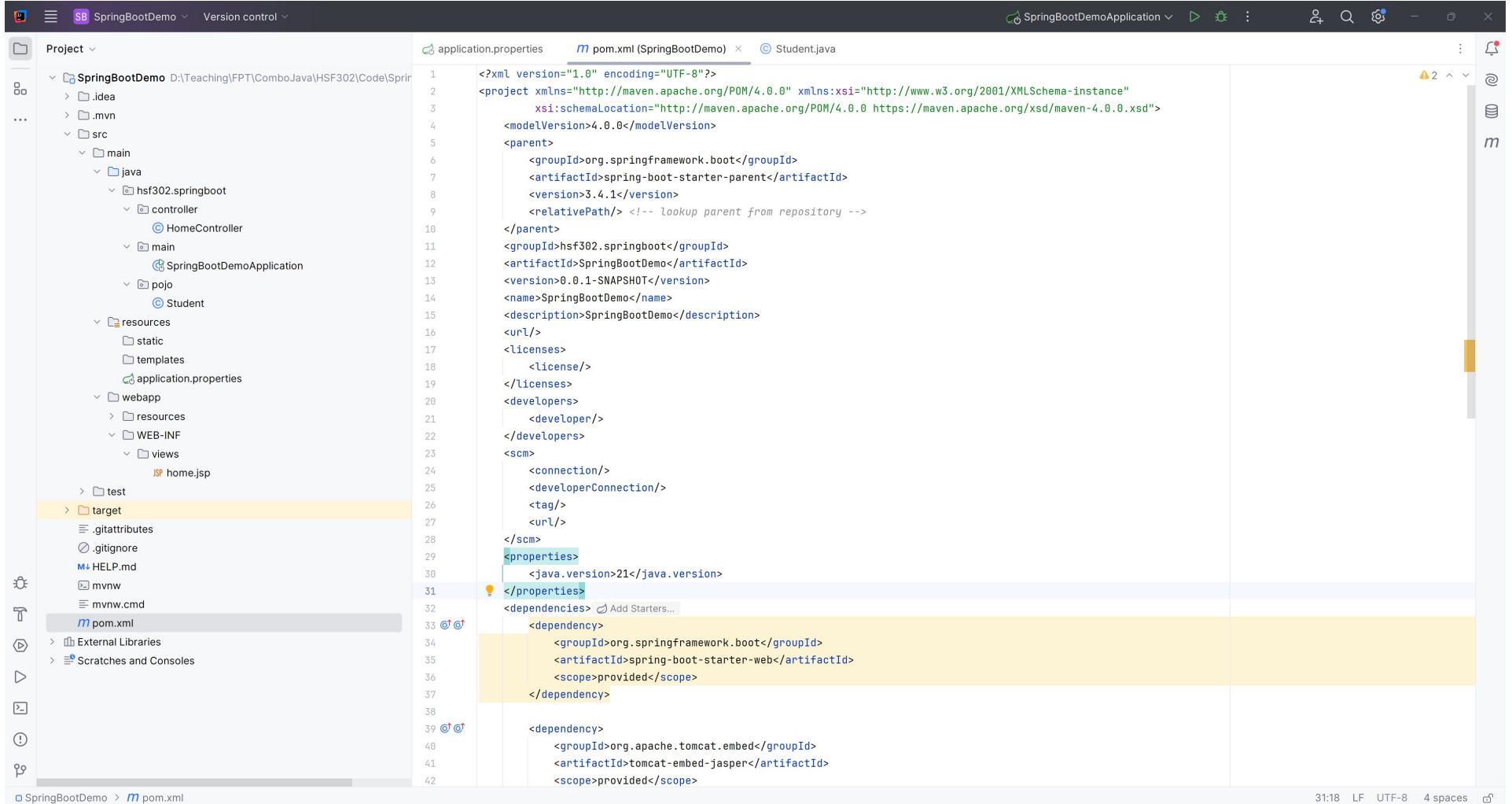
Create the Structure Project

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

- Project View:** On the left, the project structure is displayed under "SpringBootDemo". It includes the following directories:
 - .idea
 - .mvn
 - src
 - main
 - java
 - hsf302.springboot
 - controller
 - HomeController
 - main
 - SpringBootDemoApplication
 - pojo
 - Student
 - resources
 - static
 - templates
 - application.properties
 - webapp
 - resources
 - WEB-INF
 - views
 - home.jsp
 - test
 - target
 - .gitattributes
 - .gitignore
 - HELP.md
 - mvnw
 - mvnw.cmd
 - pom.xml
 - External Libraries
 - Scratches and Consoles
- Editor:** The main editor area displays the contents of the "application.properties" file.

```
spring.application.name=SpringBootDemo
#Enable JSP support
spring.mvc.view.prefix=/WEB-INF/views/
spring.mvc.view.suffix=.jsp
```
- Status Bar:** At the bottom, the status bar shows the path "SpringBootDemo > src > main > resources > application.properties", the time "4:28", and encoding information "LF ISO-8859-1 4 spaces".

Edit pom.xml



The screenshot shows an IDE interface with the following details:

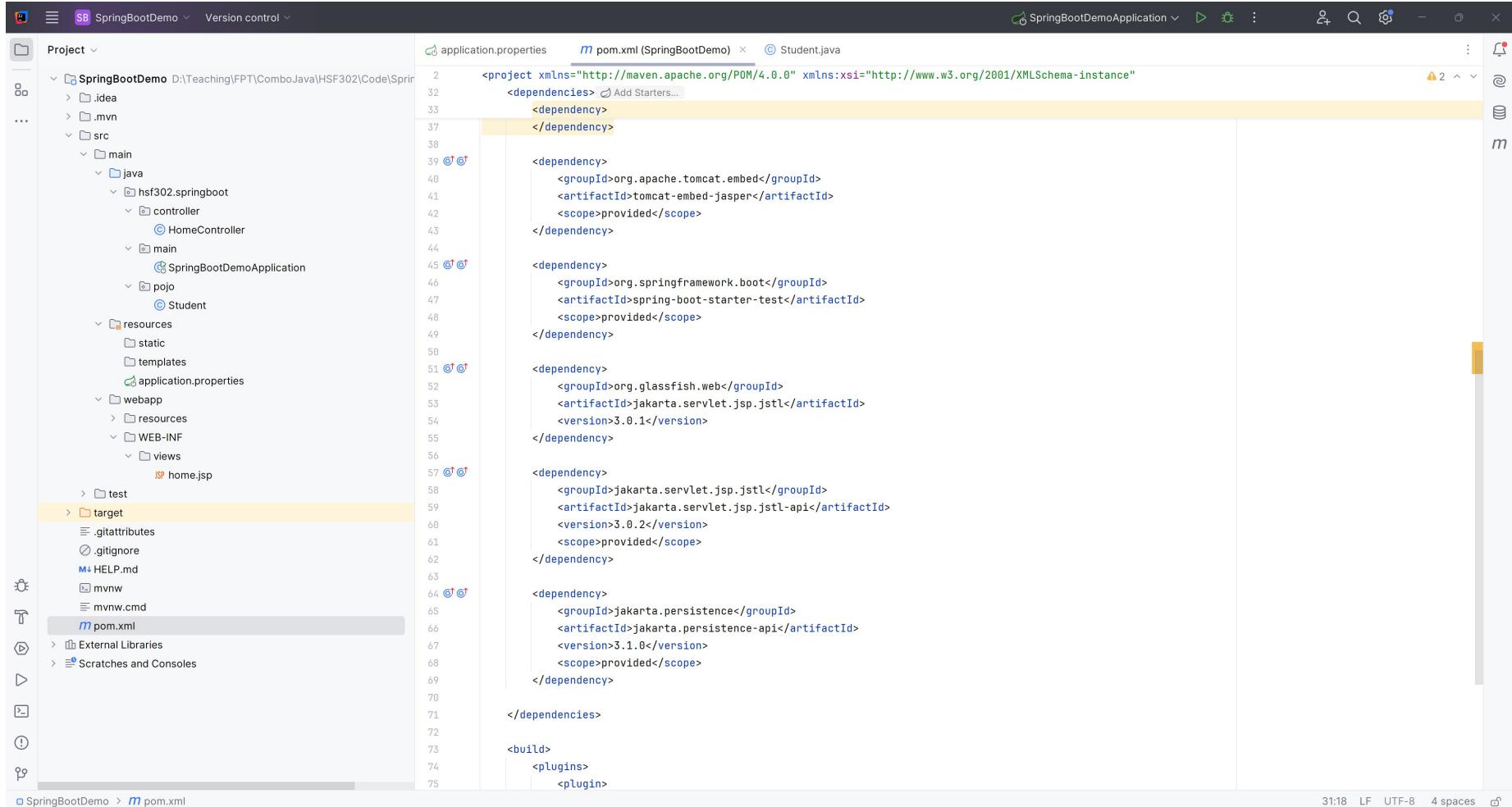
- Project Structure:** The left sidebar displays the project structure for "SpringBootDemo". It includes the following directories:
 - .idea
 - .mvn
 - src
 - main
 - java
 - hsf302.springboot
 - controller
 - HomeController
 - main
 - SpringBootDemoApplication
 - pojo
 - Student
 - resources
 - static
 - templates
 - application.properties
 - webapp
 - resources
 - WEB-INF
 - views
 - home.jsp
 - test
 - target
 - .gitattributes
 - .gitignore
 - HELP.md
 - mvnw
 - mvnw.cmd
 - pom.xml:** The right pane shows the content of the pom.xml file. The code is as follows:

```
<?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>3.4.1</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>hsf302.springboot</groupId>
  <artifactId>SpringBootDemo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>SpringBootDemo</name>
  <description>SpringBootDemo</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-web</artifactId>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>org.apache.tomcat.embed</groupId>
      <artifactId>tomcat-embed-jasper</artifactId>
      <scope>provided</scope>
    </dependency>
  </dependencies>

```

The code editor shows syntax highlighting and several error markers (red circles with 'E' and 'W') pointing to specific lines in the XML structure.

Edit pom.xml



The screenshot shows an IDE interface with the following details:

- Project Structure:** The left sidebar displays the project structure for "SpringBootDemo". It includes a .idea folder, a .mvn folder, a src folder containing main (with java, resources, static, templates, and application.properties), webapp (with resources, WEB-INF, and views), test, target, .gitattributes, .gitignore, HELP.md, mvnw, and mvnw.cmd.
- Pom.xml Content:** The main editor window shows the contents of the pom.xml file. The code is as follows:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <dependencies>
    <dependency>
      <groupId>org.apache.tomcat.embed</groupId>
      <artifactId>tomcat-embed-jasper</artifactId>
      <scope>provided</scope>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>provided</scope>
    </dependency>

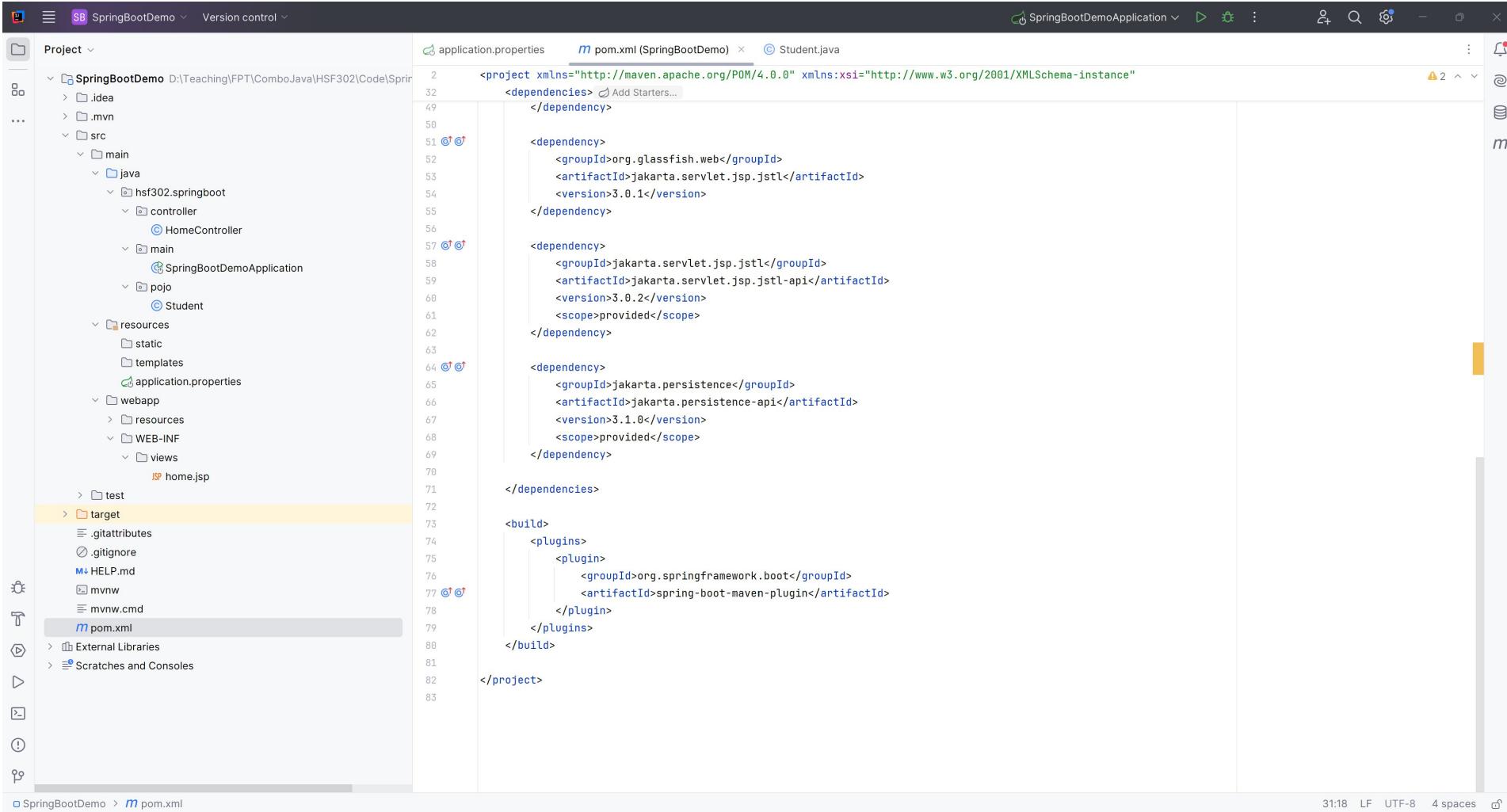
    <dependency>
      <groupId>org.glassfish.web</groupId>
      <artifactId>jakarta.servlet.jsp.jstl</artifactId>
      <version>3.0.1</version>
    </dependency>

    <dependency>
      <groupId>jakarta.servlet.jsp.jstl</groupId>
      <artifactId>jakarta.servlet.jsp.jstl-api</artifactId>
      <version>3.0.2</version>
      <scope>provided</scope>
    </dependency>

    <dependency>
      <groupId>jakarta.persistence</groupId>
      <artifactId>jakarta.persistence-api</artifactId>
      <version>3.1.0</version>
      <scope>provided</scope>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
```

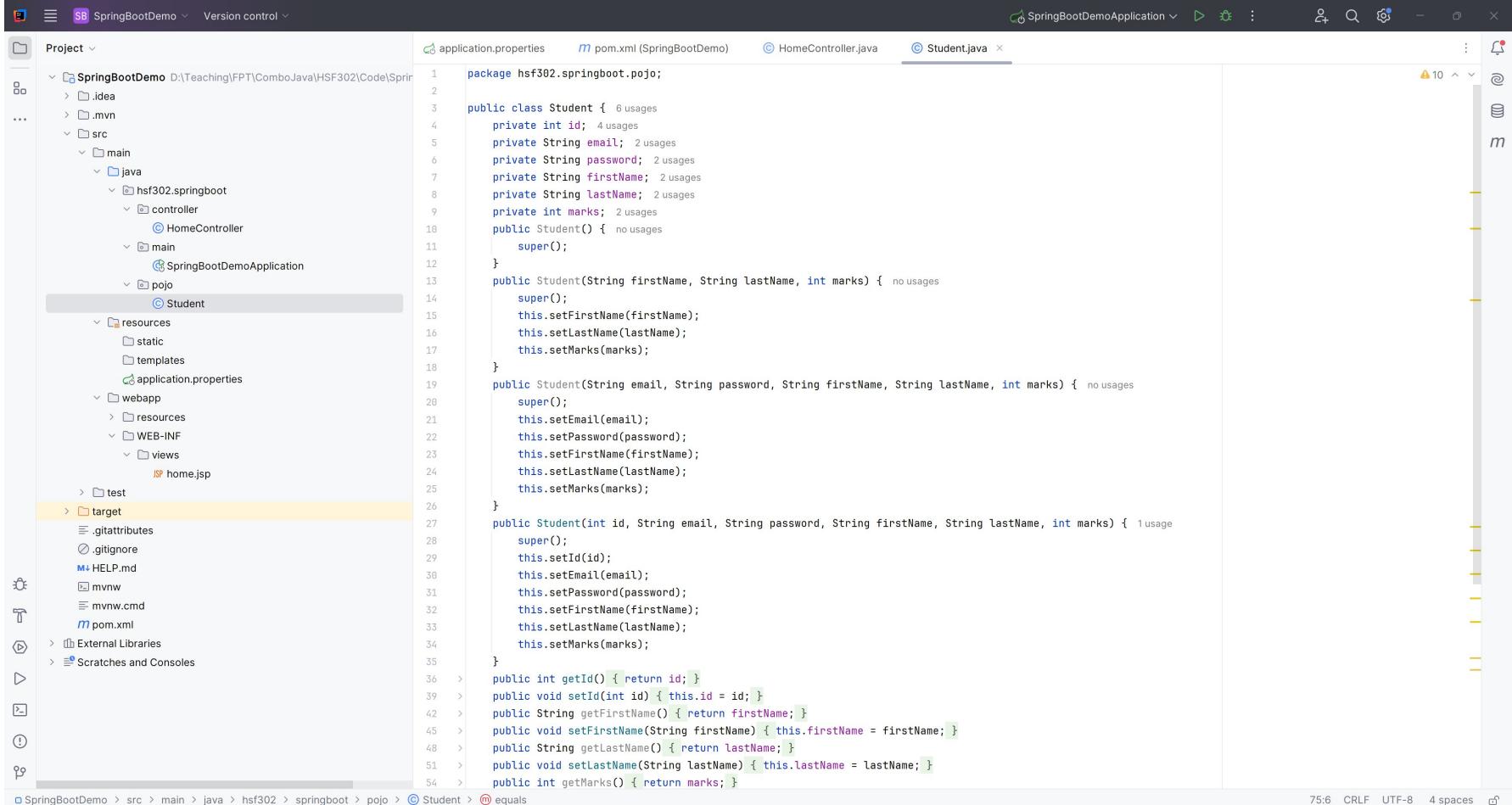
Edit pom.xml



The screenshot shows an IDE interface with the following details:

- Project Structure:** The left sidebar shows a project named "SpringBootDemo" with the following structure:
 - .idea
 - .mvn
 - src
 - main
 - java
 - hsf302.springboot
 - controller
 - HomeController
 - main
 - SpringBootDemoApplication
 - pojo
 - Student
 - resources
 - static
 - templates
 - application.properties
 - webapp
 - resources
 - WEB-INF
 - views
 - home.jsp
 - test
 - target
 - .gitattributes
 - .gitignore
 - HELP.md
 - mvnw
 - mvnw.cmd
 - pom.xml
 - External Libraries
 - Scratches and Consoles
 - POM File Content:** The main editor area displays the contents of the pom.xml file. The file defines a Maven project with dependencies on Jakarta Servlet API and Persistence API, and uses the Spring Boot Maven plugin.
 - Status Bar:** The bottom status bar indicates the file is 31:18, LF, UTF-8, 4 spaces, and has a save icon.

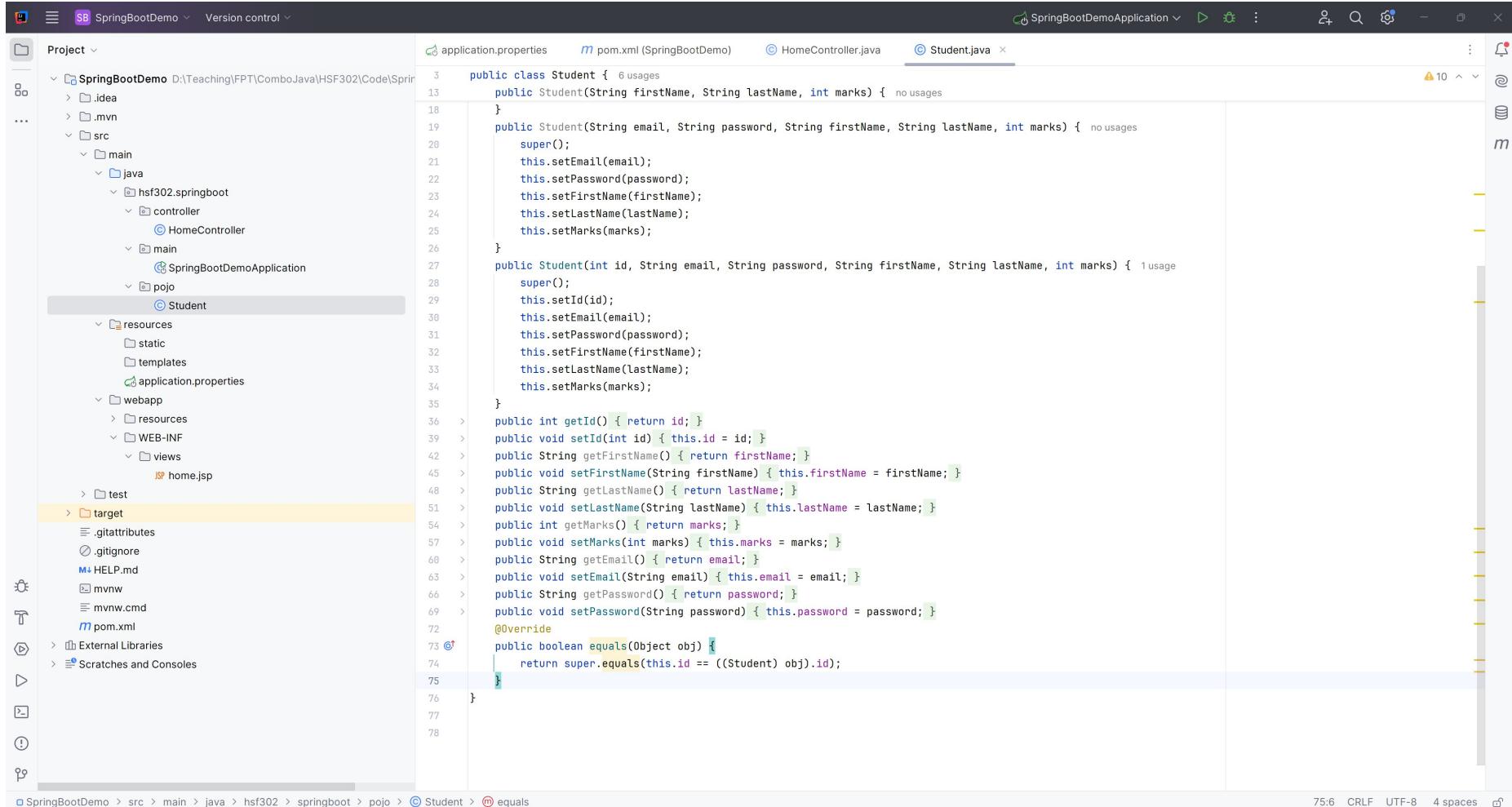
Create the Student.java in pojo



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

- Project Structure:** The project is named "SpringBootDemo". It contains a "src" directory with "main" and "test" packages. "main" contains "java", "resources", "static", "templates", and "application.properties". "java" contains "hsf302.springboot", "controller", "HomeController", "main", "SpringBootDemoApplication", and "pojo". "pojo" contains "Student".
- Code Editor:** The "Student.java" file is open. The code defines a class "Student" with fields: id (private int), email (private String), password (private String), firstName (private String), lastName (private String), and marks (private int). It includes constructors for different parameter sets and getters/setters for all fields.
- Toolbars and Status Bar:** The top bar shows tabs for "application.properties", "pom.xml (SpringBootDemo)", "HomeController.java", and "Student.java". The status bar at the bottom right shows "75:6 CRLF UTF-8 4 spaces".

Create the Student.java in pojo

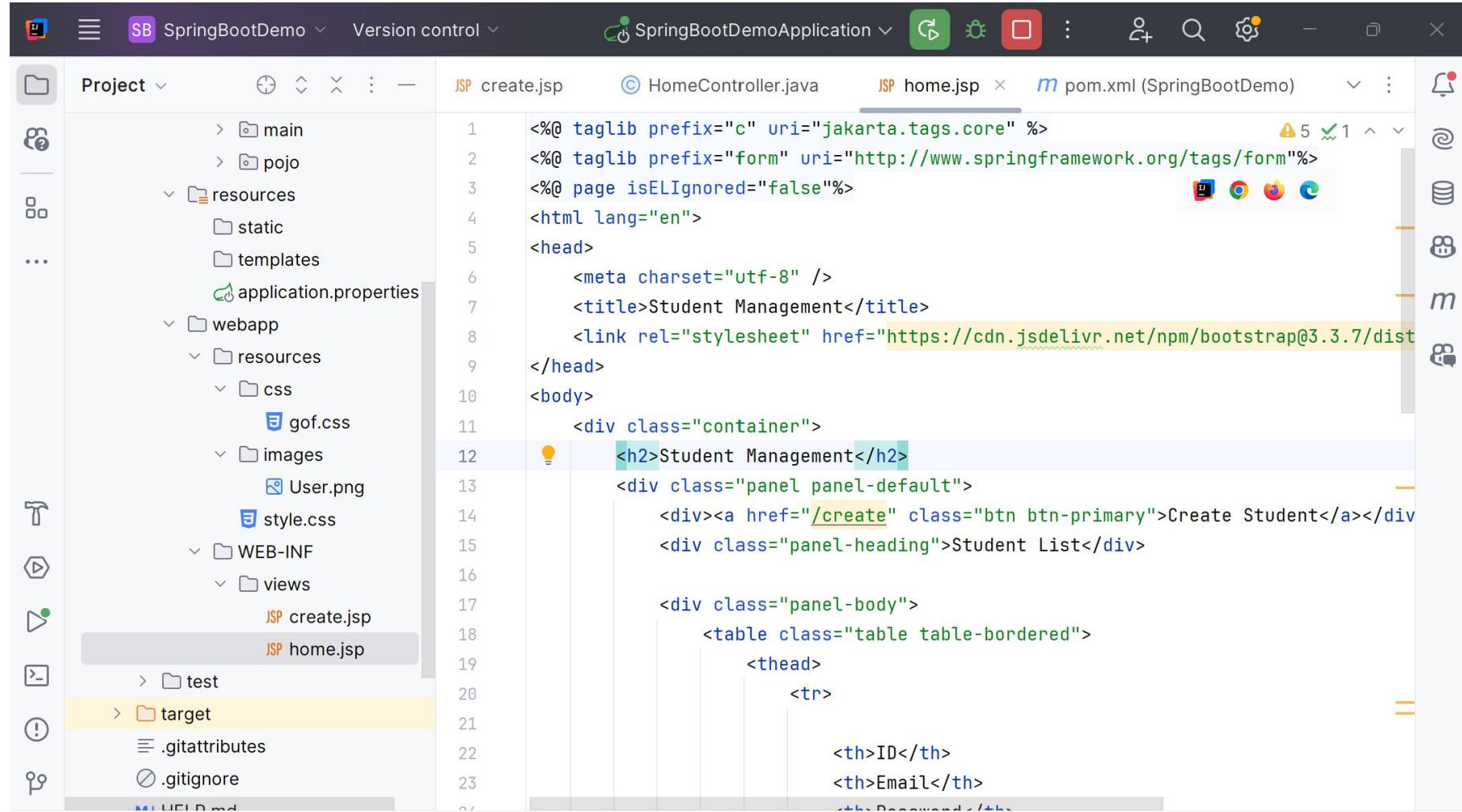


The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "SpringBootDemo" located at D:\Teaching\FPT\ComboJava\HSF302\Code\SpringBootDemo. It contains a .idea folder, .mvn folder, and a src folder. The src folder has main, resources, and webapp subfolders. The main folder contains java, hsf302.springboot, controller, main, and pojo subfolders. The pojo folder contains a Student.java file.
- Code Editor:** The Student.java file is open in the code editor. The code defines a Student class with various constructors, getters, and setters for id, firstName, lastName, and marks. It also includes an equals method.
- Status Bar:** The status bar at the bottom shows the file path as "SpringBootDemo > src > main > java > hsf302 > springboot > pojo > Student > equals", and the bottom right corner shows "75:6 CRLF UTF-8 4 spaces".

```
public class Student {    public Student(String firstName, String lastName, int marks) {        super();        this.setEmail(email);        this.setPassword(password);        this.setFirstName(firstName);        this.setLastName(lastName);        this.setMarks(marks);    }    public Student(int id, String email, String password, String firstName, String lastName, int marks) {        super();        this.setId(id);        this.setEmail(email);        this.setPassword(password);        this.setFirstName(firstName);        this.setLastName(lastName);        this.setMarks(marks);    }    public int getId() {        return id;    }    public void setId(int id) {        this.id = id;    }    public String getFirstName() {        return firstName;    }    public void setFirstName(String firstName) {        this.firstName = firstName;    }    public String getLastName() {        return lastName;    }    public void setLastName(String lastName) {        this.lastName = lastName;    }    public int getMarks() {        return marks;    }    public void setMarks(int marks) {        this.marks = marks;    }    public String getEmail() {        return email;    }    public void setEmail(String email) {        this.email = email;    }    public String getPassword() {        return password;    }    public void setPassword(String password) {        this.password = password;    }    @Override    public boolean equals(Object obj) {        if (obj == null || !(obj instanceof Student)) {            return false;        }        return super.equals(this.id == ((Student) obj).id);    }}
```

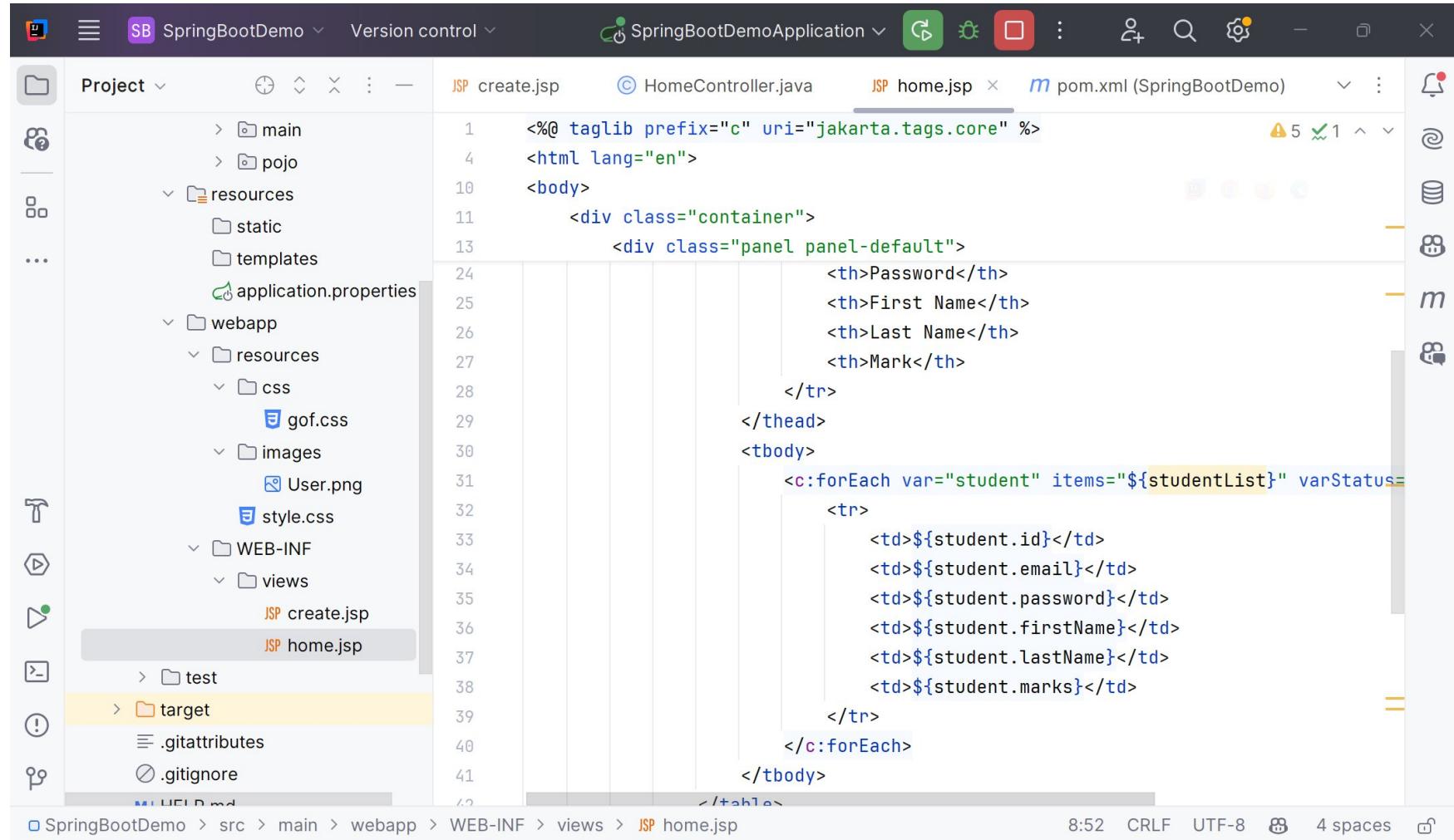
Create views : home.jsp



The screenshot shows a Java IDE interface with the following details:

- Project Structure:** On the left, the project tree shows a 'SpringBootDemo' project with a 'main' directory, 'pojo', 'resources' (containing 'static' and 'templates'), 'application.properties', 'webapp' (containing 'resources' with 'css' and 'images' subfolders, and 'style.css'), and 'WEB-INF' (containing 'views'). Inside 'views', 'create.jsp' and 'home.jsp' are listed.
- Code Editor:** The right pane displays two files:
 - HomeController.java:** A Java class with methods for handling student creation and listing.
 - home.jsp:** An JSP file containing HTML and JSTL code for a student management application. It includes a header with meta charset and title, a body with a container div, an h2 heading, and a panel-default div containing a link to '/create' and a panel-heading 'Student List'. It also includes a table with columns for ID and Email.
- IDE Tools:** Various icons for version control, build, and navigation are visible along the top and right edges of the interface.

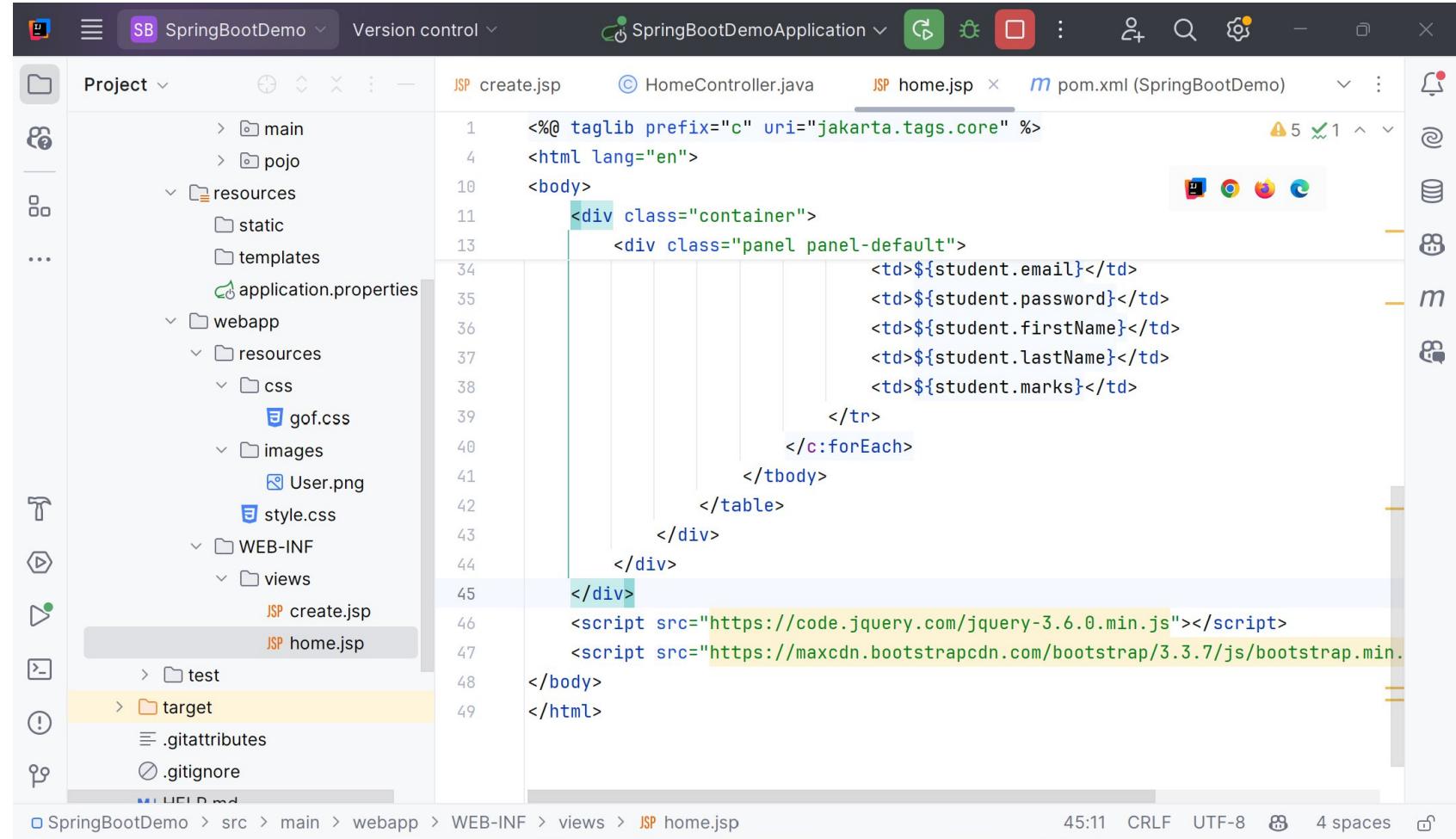
Create views : home.jsp



The screenshot shows a Java IDE interface with the following details:

- Project Structure:** The left sidebar shows the project structure under "Project". It includes main, pojo, resources (static, templates), application.properties, webapp (resources/css/gof.css, resources/images/User.png, style.css), WEB-INF (views), test, target, .gitattributes, and .gitignore.
- Editor:** The main editor area displays the content of the file "JSP home.jsp". The code is a JSP page with JSTL tags. It starts with a taglib declaration, followed by an HTML form with a container and panel-default divs. It contains a table with columns for Password, First Name, Last Name, and Mark. The body of the table is populated with a c:forEach loop over a student list, displaying student.id, email, password, first name, last name, and marks.
- Toolbar:** The top bar includes icons for file operations, version control, and search.
- Status Bar:** The bottom bar shows the file path (SpringBootDemo > src > main > webapp > WEB-INF > views > JSP home.jsp), the current time (8:52), encoding (CRLF), character set (UTF-8), and other settings.

Create views : home.jsp



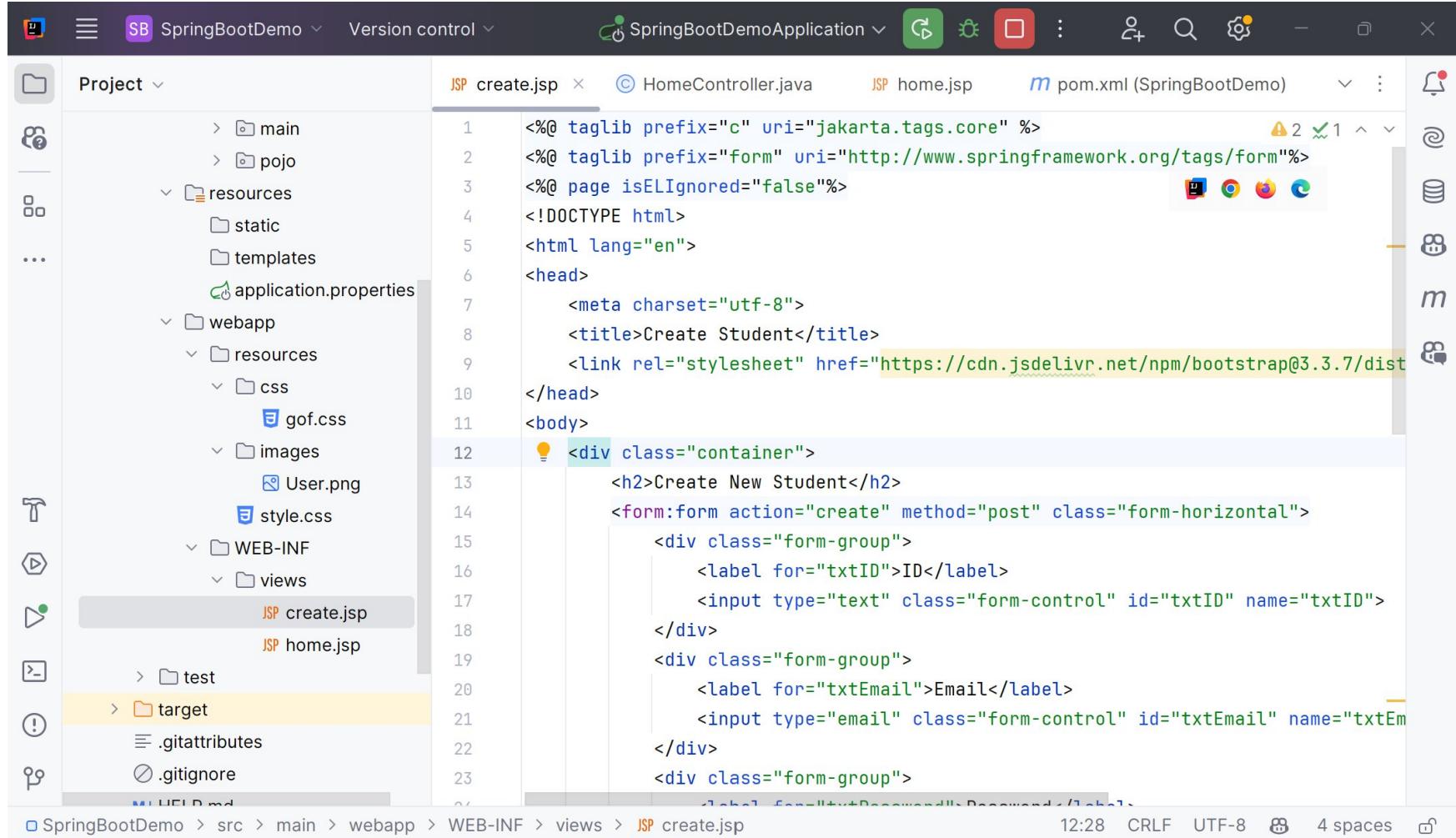
The screenshot shows a code editor interface with the following details:

- Project Structure:** The left sidebar displays the project structure under "Project". It includes main, pojo, resources (static, templates), application.properties, webapp (resources, css, images, WEB-INF), and test. The target folder is highlighted in yellow.
- Editors:** There are four tabs open in the top bar: "create.jsp", "HomeController.java", "home.jsp" (which is the active tab), and "pom.xml (SpringBootDemo)".
- Code Content:** The "home.jsp" tab contains JSP code. The code uses Jakarta Taglibs and includes a table to display student data. It also includes script tags for jQuery and Bootstrap.

```
<%@ taglib prefix="c" uri="jakarta.tags.core" %>
<html lang="en">
<body>
    <div class="container">
        <div class="panel panel-default">
            <table border="1">
                <tr>
                    <td>${student.email}</td>
                    <td>${student.password}</td>
                    <td>${student.firstName}</td>
                    <td>${student.lastName}</td>
                    <td>${student.marks}</td>
                </tr>
            </table>
        </div>
    </div>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</body>
</html>
```

- Bottom Status Bar:** The status bar at the bottom shows the file path "SpringBootDemo > src > main > webapp > WEB-INF > views > JSP home.jsp", and various file statistics like "45:11 CRLF UTF-8 4 spaces".

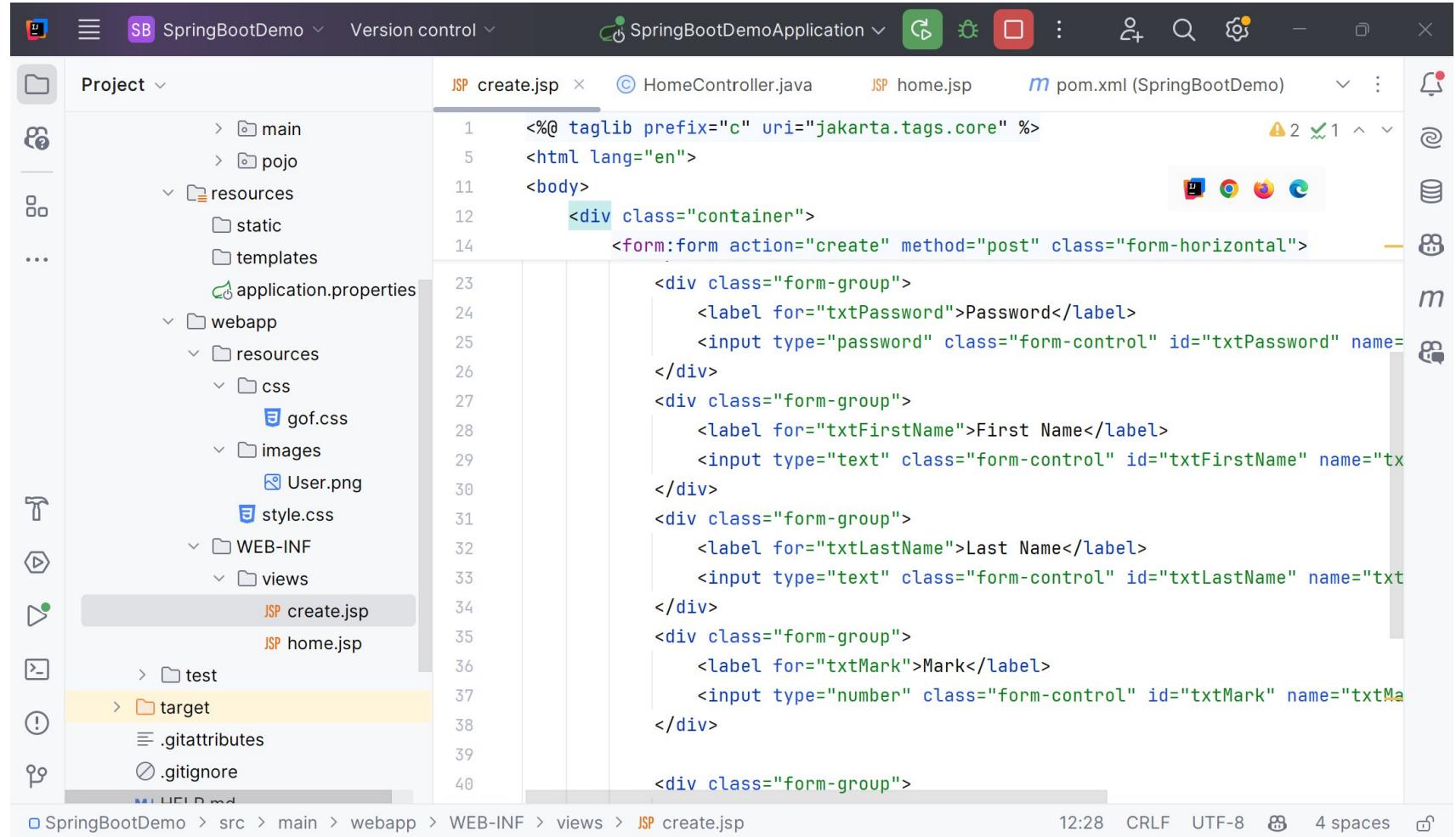
Create views : create.jsp



The screenshot shows a Visual Studio Code (VS Code) interface with the following details:

- Project Explorer:** Shows the project structure under "Project". Key folders include "main", "pojo", "resources" (containing "static" and "templates"), "application.properties", "webapp" (containing "resources" with "css" and "images" subfolders), and "WEB-INF" (containing "views"). A file named "JSP create.jsp" is currently selected.
- Editor:** Displays the content of "JSP create.jsp". The code is a JSP page with JSTL tags and Bootstrap CSS imports. It includes sections for the head and body, with an H2 title and a form for creating a new student.
- Status Bar:** Shows the file path as "SpringBootDemo > src > main > webapp > WEB-INF > views > JSP create.jsp", the time "12:28", and encoding "UTF-8".

Create views : create.jsp



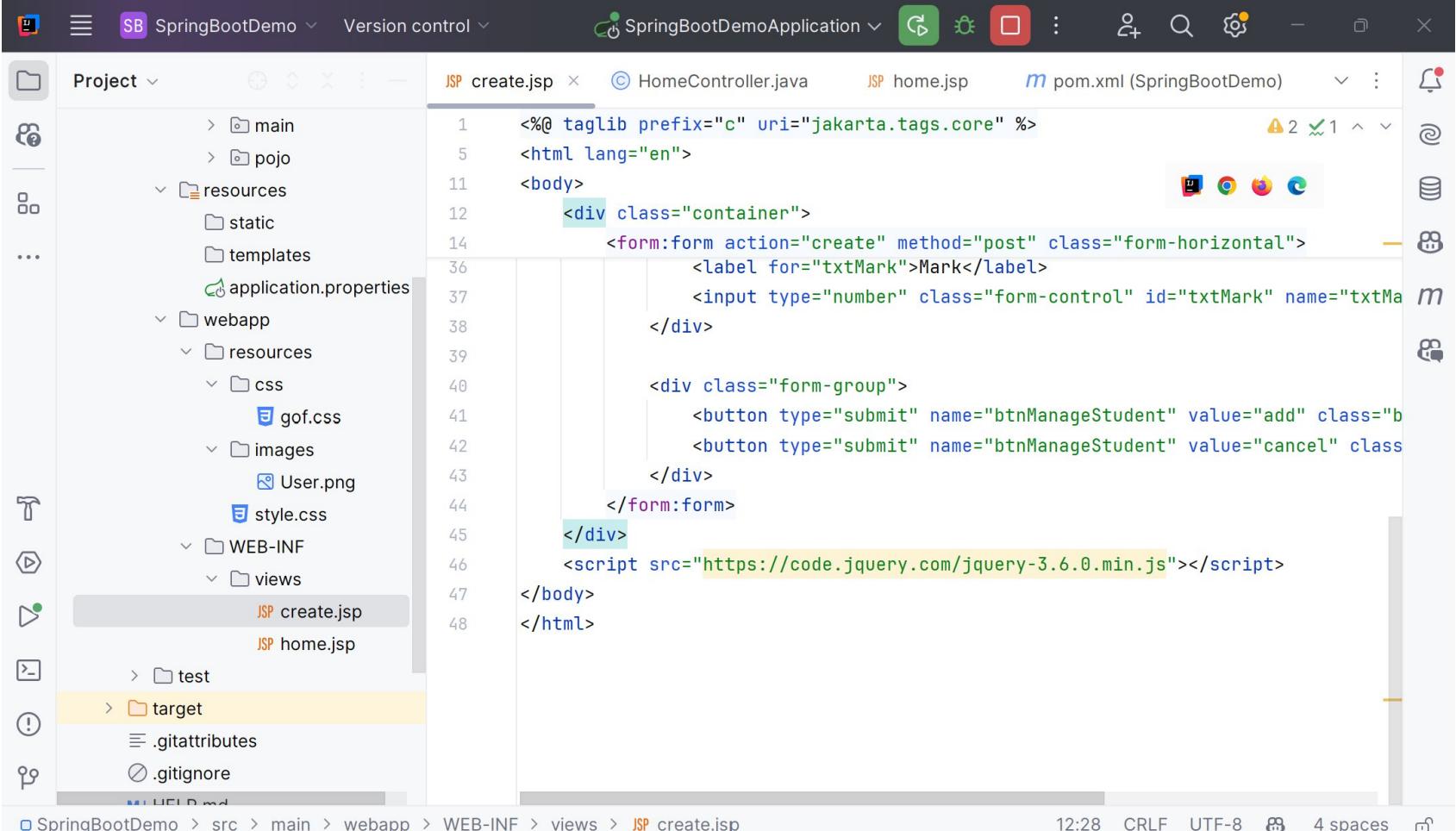
The screenshot shows a Java IDE interface with the following details:

- Project Structure:** The left sidebar shows a project named "SpringBootDemo". It contains a "main" folder, a "pojo" folder, a "resources" folder (containing "static" and "templates" subfolders), an "application.properties" file, a "webapp" folder (containing "resources" (with "css" and "gof.css"), "images" (with "User.png" and "style.css"), and "WEB-INF" (with "views" containing "create.jsp" and "home.jsp")), a "test" folder, and a "target" folder.
- Code Editor:** The right pane displays the content of the "create.jsp" file. The code is written in JSP and uses Jakarta Taglib syntax. It includes form fields for password, first name, last name, and mark.

```
<%@ taglib prefix="c" uri="jakarta.tags.core" %>
<html lang="en">
<body>
    <div class="container">
        <form:form action="create" method="post" class="form-horizontal">
            <div class="form-group">
                <label for="txtPassword">Password</label>
                <input type="password" class="form-control" id="txtPassword" name="txtPassword" />
            </div>
            <div class="form-group">
                <label for="txtFirstName">First Name</label>
                <input type="text" class="form-control" id="txtFirstName" name="txtFirstName" />
            </div>
            <div class="form-group">
                <label for="txtLastName">Last Name</label>
                <input type="text" class="form-control" id="txtLastName" name="txtLastName" />
            </div>
            <div class="form-group">
                <label for="txtMark">Mark</label>
                <input type="number" class="form-control" id="txtMark" name="txtMark" />
            </div>
    </div>
</body>
</html>
```

- Status Bar:** The bottom status bar shows the file path as "SpringBootDemo > src > main > webapp > WEB-INF > views > JSP create.jsp", the current time as "12:28", and encoding as "CRLF UTF-8 4 spaces".

Create views : create.jsp



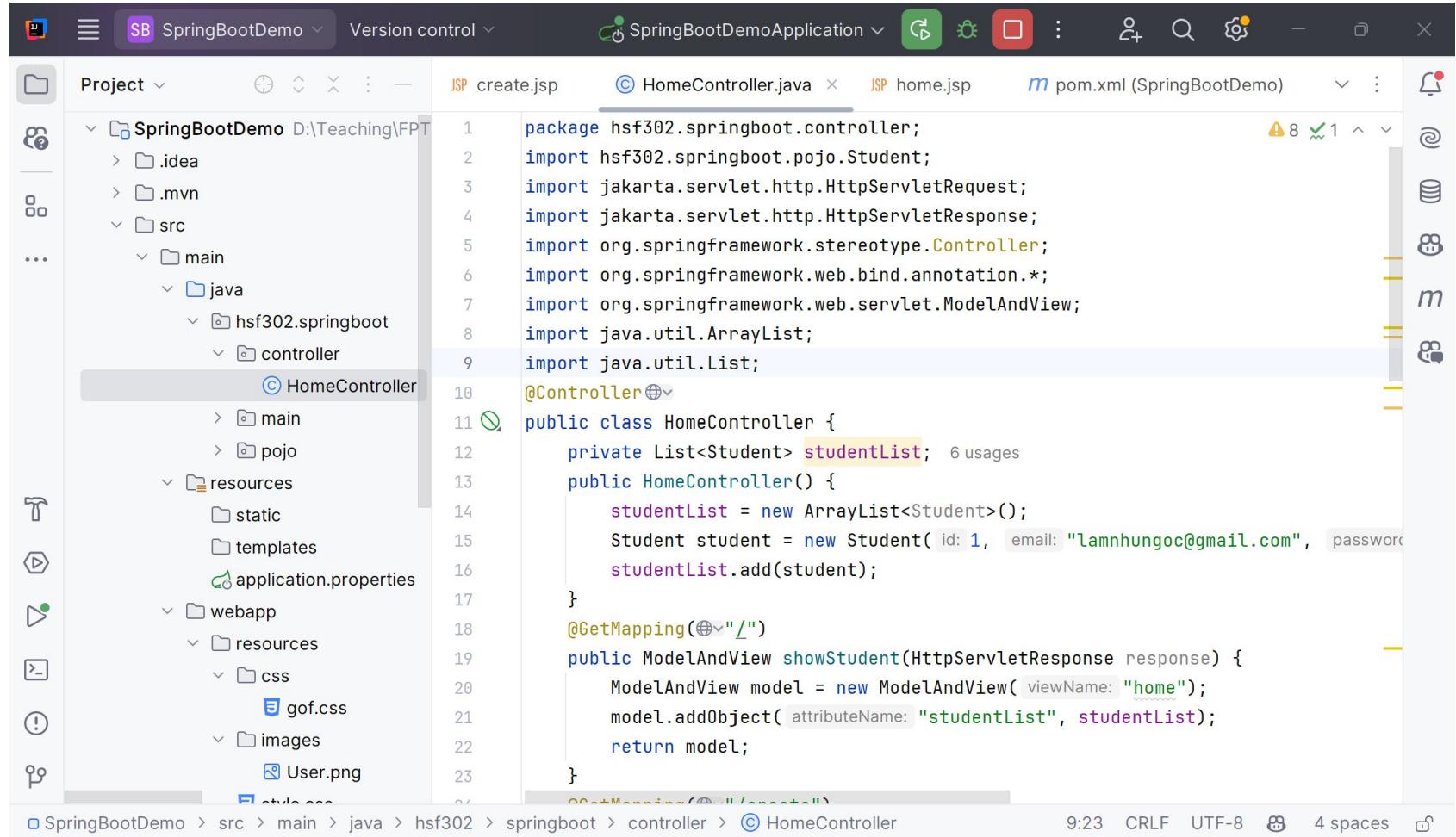
The screenshot shows a code editor interface with the following details:

- Project Structure:** The left sidebar shows a project named "SpringBootDemo" with the following structure:
 - main
 - pojo
 - resources
 - static
 - templates
 - application.properties
 - webapp
 - resources
 - css (containing gof.css)
 - images (containing User.png)
 - style.css
 - WEB-INF
 - views (containing create.jsp, home.jsp)
 - test
 - target
 - .gitattributes
 - .gitignore
- Code Editor:** The main area displays the content of the "create.jsp" file:

```
<%@ taglib prefix="c" uri="jakarta.tags.core" %>
<html lang="en">
<body>
    <div class="container">
        <form:form action="create" method="post" class="form-horizontal">
            <label for="txtMark">Mark</label>
            <input type="number" class="form-control" id="txtMark" name="txtMa
            </div>

            <div class="form-group">
                <button type="submit" name="btnManageStudent" value="add" class="b
                <button type="submit" name="btnManageStudent" value="cancel" class
            </div>
        </form:form>
    </div>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
</body>
</html>
```
- Toolbar:** The top bar includes icons for file operations, version control, and search.
- Status Bar:** The bottom bar shows the file path "SpringBootDemo > src > main > webapp > WEB-INF > views > JSP create.jsp", the timestamp "12:28", and encoding information "CRLF UTF-8 4 spaces".

Create the HomeController.java



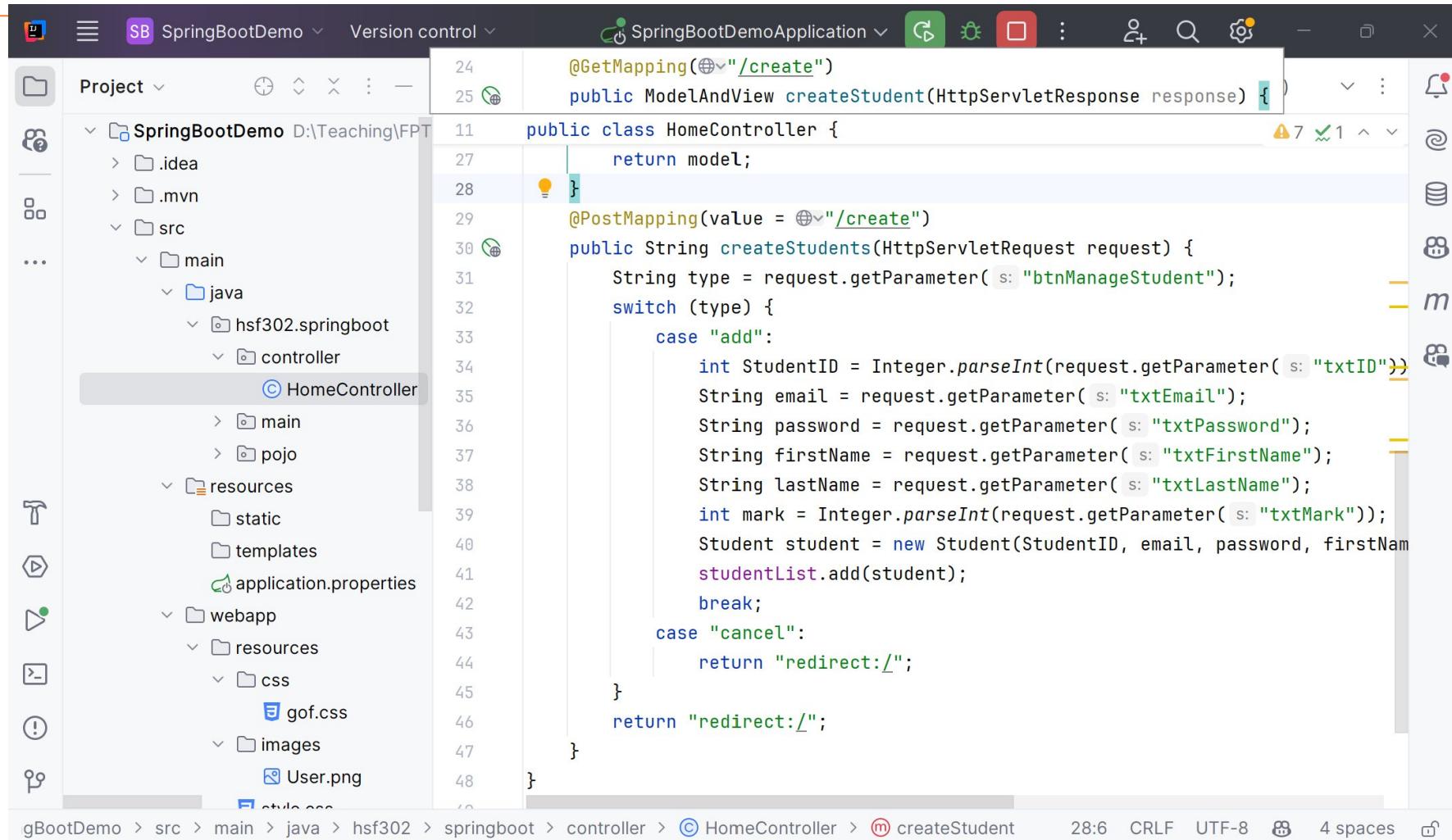
The screenshot shows the IntelliJ IDEA interface with the project 'SpringBootDemo' open. The code editor displays the file 'HomeController.java' under the package 'hsf302.springboot.controller'. The code implements a Spring MVC controller for displaying student information.

```
package hsf302.springboot.controller;
import hsf302.springboot.pojo.Student;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.ModelAndView;
import java.util.ArrayList;
import java.util.List;

@Controller
public class HomeController {
    private List<Student> studentList; 6 usages
    public HomeController() {
        studentList = new ArrayList<Student>();
        Student student = new Student(id: 1, email: "lamnhungoc@gmail.com", password: "123456");
        studentList.add(student);
    }
    @GetMapping("/")
    public ModelAndView showStudent(HttpServletRequest response) {
        ModelAndView model = new ModelAndView(viewName: "home");
        model.addObject(attributeName: "studentList", studentList);
        return model;
    }
}
```

The project structure on the left shows the directory tree for 'SpringBootDemo' including '.idea', '.mvn', 'src' (with 'main' and 'java' subfolders), 'resources' (with 'static', 'templates', and 'application.properties'), and 'webapp' (with 'resources' containing 'css' and 'images'). A file 'create.jsp' is also visible in the list.

Create the HomeController.java

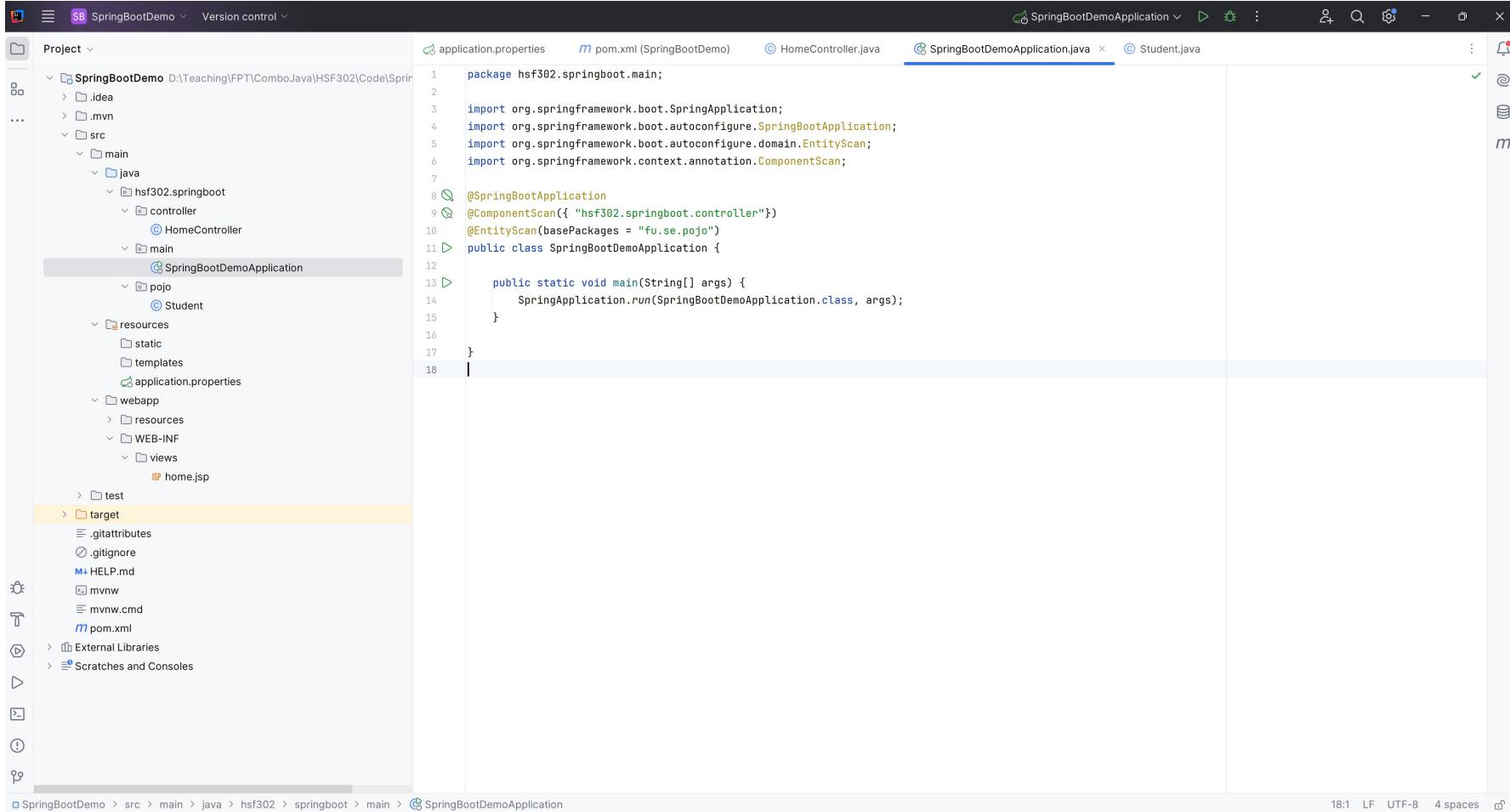


The screenshot shows the IntelliJ IDEA IDE interface. The left sidebar displays the project structure for 'SpringBootDemo' located at 'D:\Teaching\FPT'. The 'src' folder contains 'main' and 'resources' subfolders. 'main/java' contains 'hsf302.springboot/controller/HomeController.java'. The code editor shows the following Java code:

```
24 @GetMapping(@RequestMapping("/create"))
25 public ModelAndView createStudent(HttpServletRequest response) {
26     ModelAndView model = new ModelAndView("create");
27     return model;
28 }
29 @PostMapping(value = "/create")
30 public String createStudents(HttpServletRequest request) {
31     String type = request.getParameter("btnManageStudent");
32     switch (type) {
33         case "add":
34             int StudentID = Integer.parseInt(request.getParameter("txtID"));
35             String email = request.getParameter("txtEmail");
36             String password = request.getParameter("txtPassword");
37             String firstName = request.getParameter("txtFirstName");
38             String lastName = request.getParameter("txtLastName");
39             int mark = Integer.parseInt(request.getParameter("txtMark"));
40             Student student = new Student(StudentID, email, password, firstName, lastName);
41             studentList.add(student);
42             break;
43         case "cancel":
44             return "redirect:/";
45     }
46     return "redirect:/";
47 }
48 }
```

The code implements a Spring MVC controller for handling student creation. It uses annotations like @GetMapping and @PostMapping to map URLs to methods. It retrieves parameters from the request, such as 'btnManageStudent', 'txtID', 'txtEmail', etc., and creates a new 'Student' object. Finally, it adds the student to a list and returns a redirect response.

Edit SpringBoot Main

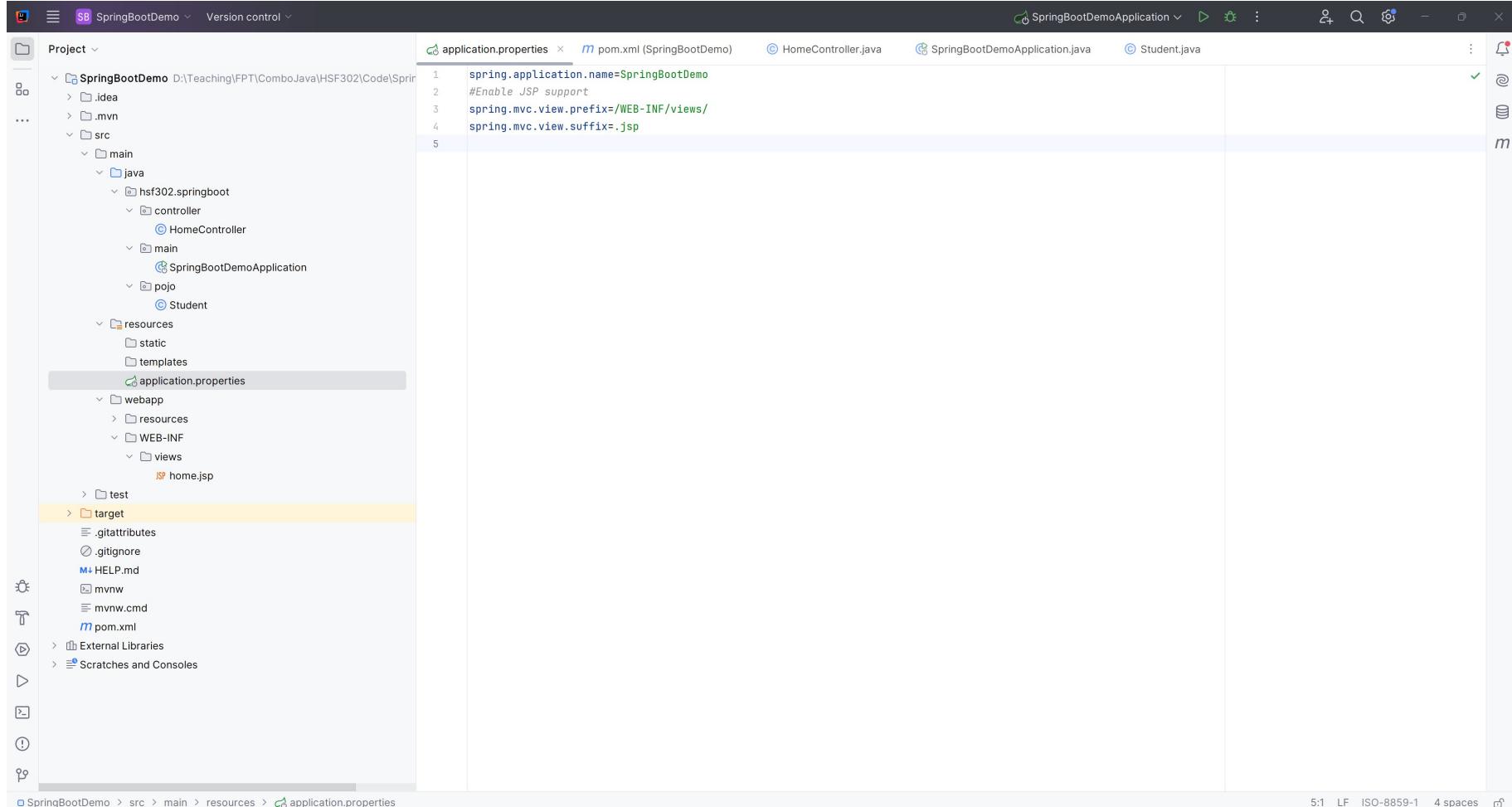


The screenshot shows a Java IDE interface with the following details:

- Project Explorer:** Shows the project structure under "SpringBootDemo". Key components include:
 - src/main/java:** Contains packages like hsf302.springboot.controller (HomeController), hsf302.springboot.main (SpringBootDemoApplication), pojo (Student), and main (application.properties).
 - resources:** Contains static, templates, and application.properties files.
 - webapp:** Contains resources, WEB-INF, and views (home.jsp).
- Panels:** The top navigation bar includes tabs for application.properties, pom.xml, HomeController.java, SpringBootDemoApplication.java (which is currently selected), and Student.java.
- Code Editor:** The main code editor displays the `SpringBootDemoApplication.java` file with the following content:

```
1 package hsf302.springboot.main;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.boot.autoconfigure.domain.EntityScan;
6 import org.springframework.context.annotation.ComponentScan;
7
8 @SpringBootApplication
9 @ComponentScan({ "hsf302.springboot.controller"})
10 @EntityScan(basePackages = "fu.se.pojo")
11 public class SpringBootDemoApplication {
12
13     public static void main(String[] args) {
14         SpringApplication.run(SpringBootDemoApplication.class, args);
15     }
16 }
17
18 }
```
- Status Bar:** At the bottom, it shows the file path as "SpringBootDemo > src > main > java > hsf302 > springboot > main > SpringBootDemoApplication", and the status "18:1 LF UTF-8 4 spaces".

Edit the application.properties



The screenshot shows the IntelliJ IDEA interface with the project `SpringBootDemo` open. The left sidebar displays the project structure, including the `src` directory which contains `main`, `resources`, and `webapp`. The `application.properties` file is selected in the editor tab bar. The code in the editor is:

```
1 spring.application.name=SpringBootDemo
2 #Enable JSP support
3 spring.mvc.view.prefix=/WEB-INF/views/
4 spring.mvc.view.suffix=.jsp
```

The `pom.xml` file is also visible in the tab bar. The status bar at the bottom right shows the file is 5:1 LF, ISO-8859-1, 4 spaces.

Run Program

The screenshot shows a web browser window titled "Student Management" with the URL "localhost:8080". The browser interface includes standard controls like back, forward, and search. Below the address bar, there's a navigation bar with links to "FPT", "DesignPattern", "Unity", "Mining", and "STEM". On the right side of the browser, there's a "All Bookmarks" link.

The main content area is titled "Student Management" and features a "Create Student" button. Below it is a table titled "Student List" with the following data:

ID	Email	Password	First Name	Last Name	Mark
1	lamnhungoc@gmail.com	passd	nguyen	lam	10

Result

Create Student

localhost:8080/create

FPT DesignPattern Unity Mining STEM All Bookmarks

Create New Student

ID

Email

Password

First Name

Last Name

Mark

Add Cancel

Summary

Concepts were introduced:

- ◆ Spring Boot
- ◆ Advantages of using Spring Boot
- ◆ Key features of Spring Boot
 - Demo Spring Boot with Collections