Assignment on Spring Boot: Understanding the @Autowired Annotation

---

Objective: To understand and implement the **@Autowired** annotation in a Spring Boot application, leveraging both field-based and constructor-based autowiring. This assignment will not involve the use of web modules or databases.

Prerequisites:

- Basic understanding of Java and Spring Framework.
- Java Development Kit (JDK) installed.
- Integrated Development Environment (IDE) of your choice (preferably Spring Tool Suite or IntelliJ IDEA with Spring Boot plugin).

Assignment Requirements:

1. Setting up the Spring Boot Application:

- Create a new Spring Boot application using Spring Initializr or your IDE. Only include the core modules; exclude web and database modules.
- Setup the basic structure with the standard Maven or Gradle project structure.

2. Field-based Autowiring:

a. Create three Java classes:

i. **Book** (A simple POJO with a **title** and **author** fields and their respective getters and setters).

ii. **Library** (Should have a field of type **Book**).

iii. **ApplicationRunnerImpl** (An implementation of the Spring's **ApplicationRunner** interface).

b. In the **Library** class:

- Use the **@Autowired** annotation to autowire the **Book** class at the field level.
- Add a method named **displayBookDetails** that prints the details of the book.

c. In the **ApplicationRunnerImpl** class:

- Use the **@Autowired** annotation to autowire the **Library** class at the field level.
- Implement the **run** method to call **displayBookDetails** method from the **Library** class.

3. Constructor-based Autowiring:

a. Create two more classes:

i. **Student** (A simple POJO with **name** and **age** fields, and their respective getters and setters).

ii. **Classroom** (Should have a constructor that accepts an argument of type **Student**).

b. In the **Classroom** class:

- Use the **@Autowired** annotation to autowire the **Student** class at the constructor level.
- Add a method named **displayStudentDetails** that prints the details of the student.

c. Update the **ApplicationRunnerImpl** class:

- Use the **@Autowired** annotation to autowire the **Classroom** class at the field level.
- Update the **run** method to also call **displayStudentDetails** method from the **Classroom** class.

4. Configuration:

- In your main Spring Boot application class or in a separate **@Configuration** class, use the **@Bean** annotation to create beans for **Book**, **Student**, **Library**, and **Classroom**. Ensure you provide necessary values for the **Book** and **Student** beans.
- Mark the **ApplicationRunnerImpl** class as a **@Component** so it gets executed when the application runs.

5. Execution and Validation:

- Run the Spring Boot application.
- Verify that both book details and student details are printed on the console.

Deliverables:

1. Source code for all the classes.
2. Screenshots of the console output.
3. A brief report discussing the differences between field-based and constructor-based autowiring, and which one you would recommend in what scenarios.

Notes:

- Remember that while field-based autowiring is convenient and reduces boilerplate code, it might make the dependencies less transparent. Constructor-based autowiring, on the other hand, ensures that the class always gets its required dependencies before it is used.
- Ensure you are following good coding practices and maintaining a clean code structure.