Spring Boot XML Context Setup Project

Overview

In this project, you will integrate Spring Boot with XML-based configuration. The goal is to learn how to load Spring beans from XML configuration files, using both classpath and filesystem-based XML files. You will implement the logic to load these configurations, access bean properties, and ensure the correct setup.

This guide will walk you through the setup and the parts that you need to implement in the provided template project.

Project Structure

Your project should have the following structure:
- **Main Java Class**: The starting point of the application where Spring contexts are loaded manually from XML files.**[This should be defined by you]**
- **TestBean Class**: A simple Java bean with a `name` property, which will be injected with values from the XML files. **[Already defined]**
- **XML Configuration Files**: These files will define Spring beans and their properties. **[This should be defined by you]**
- **Test Cases**: These will verify if your Spring configuration is correct. **[Already defined]**

Steps to Complete the Project

1. **Implement the `TestBean` Class**: This class is already defined and it is a simple Java class with a `name` property.

2. **In the main application class, your task is to load the Spring context manually from XML files. You will need to load:
- One context from the classpath-based XML configuration file (`applicationContext.xml`).
- Another context from the filesystem-based XML configuration file (`fileSystemContext.xml`).

Once the contexts are loaded, you should retrieve the `TestBean` from each context and print its `name` property to verify that the beans are correctly initialized for each classPath and fileSystem respectively as:

```
System.out.println("ClassPath Bean Name: " + testBean.getName());
```

System.out.println("FileSystem Bean Name: " +
testBeanFromFileSystem.getName());

3. **Create the XML Configuration Files**:
- Complete the `applicationContext.xml` file in the `resources` directory. This file should define the `TestBean` with the ID `testBean` and set the `name` property to `"Spring Boot Bean Example"`.

- Complete the `fileSystemContext.xml` file in the `external-config` directory. This file should define another `TestBean` with the ID `testBeanFromFileSystem` and set the `name` property to `"File System Bean"`.

Execution Steps to Follow

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**
2. **To open the command terminal the test takers need to go to the Application menu (Three horizontal lines at left top) -> Terminal -> New Terminal.**
3. **cd into your backend project folder.**
4. **To build your project use command:**

   sudo JAVA_HOME=$JAVA_HOME /usr/share/maven/bin/mvn clean package -Dmaven.test.skip

   ***If it asks for the password, provide password : pass@word1**
5. **To launch your application, move into the target folder (cd target). Run the following command to run the application:**

   java -jar <your application jar file name>
6. **To test any Restful application, the last option on the left panel of IDE, you can find ThunderClient, which is the lightweight equivalent of POSTMAN. Please use 127.0.0.1 instead of localhost to test rest endpoints.**
7. **Default credentials for MySQL:**
   a. **Username: root**
   b. **Password: pass@word1**
8. **To login to mysql instance: Open new terminal and use following command:**

a. **sudo systemctl enable mysql**

b. **sudo systemctl start mysql**

NOTE: After typing any of the above commands you might encounter any warnings.

>> Please note that this warning is expected and can be disregarded. Proceed to the next step.

c. **mysql -u root -p**

The last command will ask for password which is 'pass@word1'

9. **Mandatory: Before final submission run the following command:**

sudo JAVA_HOME=$JAVA_HOME /usr/share/maven/bin/mvn test

***If it asks for the password, provide password : pass@word1**