Enterprise Java with Spring
Spring Core
Lab 3
Exercises

1 Lab setup

Make sure you have the following items installed

- Latest LTS JDK version (at this point: JDK 21)
- A suitable IDE (Eclipse Enterprise Edition for Java) or IntelliJ IDEA
- Latest version of Maven (at this point: Maven 3.9.9)
- A suitable text editor (Notepad ++)
- A utility to extract zip files (7-zip)

In each of the main lab folders, there are two subfolders: `changes` and `final`. The `changes` subfolder holds the source code and other related files for the lab, while the `final` subfolder holds the complete Eclipse project starting from its project root folder. We will use the code from the `changes` subfolder to build up our applications from scratch and you can always fall back on the complete Eclipse project if you encounter any errors while building up the application.

2 Java-based configuration

2.1 Using @Bean, @Component, @Autowired and @Primary

Create a new project: `JavaConfigDIExercise`

Create a package: `com.exercise.javaconfig`
Place the main application class in it with the name of JavaConfigExerciseMainApp

Create a MainConfig to hold the @Configuration as well as @ComponentScan annotation to target this package.

Create an interface `Payment` with a single method signature `makePayment()`

Create these 3 normal classes which implement `Payment` and which each provide their own unique implementation of `makePayment()`

- class `OnlinePayment`
- class `DirectPayment`
- class `MobilePayment`

Create these 2 classes:

- normal class Supermarket that has a Payment object as a member field/property. This property will be initialized via constructor injection
- class WetMarket annotated with @Component and Payment object as a member field/property initialized via @Autowired

Each of these classes should have a method `doBusiness`, which in turn invokes `makePayment` on their respective `Payment` member properties

In MainConfig, create 4 @Bean methods:

- getOnlinePayment has a unique name (of your choice) associated with @Bean and returns a OnlinePayment object
- getDirectPayment returns a new DirectPayment object
- getMobilePayment returns a new MobilePayment object
- getSupermarket returns a new Supermarket object whose constructor is initialized via a call to any of the other 3 previous @Bean methods

In JavaConfigExerciseMainApp

- Create a OnlinePayment bean using getOnlinePayment's unique bean name, call `makePayment` on it.
- Create bean of type Payment and call `makePayment` on it. Use @Primary to distinguish which of the 3 possible Bean classes to use in MainConfig
- Create a Supermarket bean and call doBusiness on it
- Create a WetMarket bean and call doBusiness on it and verify that it was initialized with the correct Payment member object (make sure you can explain why you can the response you see).