**Assessment: Vending Machine with Spring DI**

**Elza Contiero**

Contents

[Introduction 2](#_Toc99665162)

[Aim 2](#_Toc99665163)

[Classes and Exceptions from DVD Vending machine 2](#_Toc99665164)

[Item 2](#_Toc99665165)

[Change 3](#_Toc99665166)

[NoItemInventoryException 3](#_Toc99665167)

[InventoryDao (DATA ACCESS OBJECT) 3](#_Toc99665168)

[InventoryDaoFileImp 3](#_Toc99665169)

[VendingMachine 3](#_Toc99665170)

[App class 3](#_Toc99665171)

[References 4](#_Toc99665172)

# Introduction

## Aim

To convert a normal Java project to support Dependency Injection. For this assignment, I am going to use java annotations. First, I am going to create an empty Maven project.

The idea is to have both the Vending Machine open in one screen and a new empty maven project open on another screen and go through the classes I have in the old project and one by one put them in the new project. So, the maven project will have a spring structure.

And to create an empty maven project, I found the recipe below in **maven.apache.org [1]**which I have adapted to my case.

mvn archetype:generate -DgroupId=org.elzacontiero.m3assessments.vendingmachinespring -DartifactId=Assessment5VendingMachineSpring -DarchetypeArtifactId=maven-archetype-quickstart -DarchetypeVersion=1.4 -DinteractiveMode=false

Maven creates the following file system structure:

.

├── pom.xml

├── src

   ├── main

   │   └── java

   │   └── org

   │   └── elzacontiero

   │   └── m3assessments

   │   └── vendingmachinespring

   │   └── App.java

   └── test

    └── java

   └── org

   └── elzacontiero

   └── m3assessments

   └── vendingmachinespring

   └── AppTest.java

Also, Maven configuration **pom.xml** file needs the following dependencies, under <dependencies>.

<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.17</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter</artifactId>

<version>2.6.5</version>

</dependency>

# Classes and Exceptions from DVD Vending machine

The classes in the program are:

## Item

This class models the items available in the vending machine. It has properties such as name, price, quantityAvailable.

This class should go into package **org.elzacontiero.m3assessments.vendingmachinespring.dto.**

## Change

Class Change just calculates the coins, so it is more of a utility class, therefore, there is no need for any annotation.

## NoItemInventoryException

As it is an exception, there shouldn’t be any annotations. It goes into package **org.elzacontiero.m3assessments.vendingmachinespring.**

## InventoryDao (DATA ACCESS OBJECT)

The inventory DAO is an interface, so it goes into **org.elzacontiero.m3assessments.vendingmachinespring.dao**  package.

As it is an interface it doesn’t need any annotations.

## InventoryDaoFileImp

This is an implementation of interface InventoryDao , so it goes alongside it in the package: **org.elzacontiero.m3assessments.vendingmachinespring.dao**

As said in InventoryDao, the annotation in this class is: **@Component** because it is a class that implements inventoryDAO and will be injected into VendingMachine.

## VendingMachine

This is the main class and it will be annotated with **@SpringBootApplication** and put it in package **org.elzacontiero.m3assessments.vendingmachinespring**

**On dependency Injection:** change VendingMachine class to include a constructor that accepts InventoryDao object. It is expected that Spring creates the object that implements inventory DAO and vending machine will accept it as a parameter to the constructor.

@SpringBootApplication

public class VendingMachine implements CommandLineRunner {

InventoryDao inventory;

VendingMachine(InventoryDao inventory) {

this.inventory = inventory;

}

@Override

public void run(String... args) {

…

The annotation @SpringBootApplication above makes the VendingMachine a Spring Application, so it knows that it needs to create the InventoryDao as part of the process of creating a new instance of VendingMachine.

Moreover, as now VendingMachine implements interface CommandLineRunner, there’s a need to change the original run() method to the one listed above.

This has been taken from [2]

## App class

The entry point of the application goes straight to App.java in the maven project. There won’t be any annotations on that. However, the main method will change to use Spring way of creating apps.

import org.springframework.boot.SpringApplication;

public class App {

public static void main(String[] args) {

SpringApplication.run(VendingMachine.class, args);

}

}

# References

1. <https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html>
2. <https://www.baeldung.com/spring-boot-console-app>