**Assignment 5: Vending Machine with Spring DI**

Maven Explanation

Maven is a project management framework, and one of the primary uses of Maven in Java projects is managing library dependencies. Using a POM (Project Object Model) XML configuration file, Maven connects external libraries to the project, which can then be used to provide additional functionality.

The Spring framework is an external library that can be managed by Maven. The developer can add the Spring framework to the list of dependencies in the pom file, and the functionality required for Spring DI is made available to the developer.

Maven Lifecycle

When a project is brought in by Maven, there are several steps that are completed behind the scenes, to ensure the project brought in as a dependency is working correctly. These include compiling the source code for the project and various tests that come with the project and then running the tests (to see if the imported project still behaves the way it should). If the tests all pass, the project is packaged up and placed in a local repository, where it can be used in the developer’s own projects.

Spring Explanation

The Spring framework provides functionality that aids the development of large-scale projects, using “Plain Old Java Objects” (POJOs). These objects provided by Spring are almost entirely unspecific to a framework and are therefore able to be used across a variety of projects.

Spring provides objects and functionality to handle dependency injection (DI), which aims to separate the instantiation of client object dependencies from the client object itself, and have the instantiation occur in a configuration file separate from the main source code of the project (the client object is the object that requires instances of other “service objects”. i.e., a controller with instances of the model and view classes). DI helps to achieve inversion of control.

\*I wanted to try and explain inversion of control, but I’m not 100% sure of my understanding. As far as my understanding goes, inversion of control is where instead of the developer’s code (i.e. my code) being the point of control and making calls to a framework’s functionality, the framework is the point of control and it makes calls to the developer’s code, which is designed to implement and provide functionality for the interfaces provided by the framework.\*

Spring DI in Vending Machine

I used the XML configuration file method of integrating Spring DI into my vending machine application.

If I were to use the annotation method, I would’ve gone through the classes that were components of my Vending Controller and labelled them with the @Component annotation. Then, I would’ve labelled the Vending Controller constructor (which takes the component classes as arguments) with the @Autowired annotation. Finally, the main class would use the annotation configuration method to get the application context and generate the beans that way.

XML file

Text

Description automatically generated

Create the bean for the controller class, with the component beans being passed in with the constructor-arg tag

itemStorage and auditStorage take in file paths as constructor arguments, pass them into the constructors here

Setting up the beans for the component classes (io, itemIO, and auditIO)

Main Method

Text

Description automatically generated with low confidence

Line 9 creates the application context by referring to the XML configuration file.

Line 11 creates a Vending Controller instance by getting the associated bean from the application context. This bean will have instances of all component classes passed in when the construction occurs.

Line 13 runs the vending machine program.