Student's Name

Professor's Name

Course

Date

Model View Controller (MVC) with Spring Boot

Model view controller (MVC) is a software architectural pattern for building web applications. Spring is a powerful, feature-rich, and enterprise-grade Java web and application development framework (Kaplan). MVC architecture separates an application into three parts: model, view, and controller (Patel). It also separates the functionality, logic, and interface of a web application into an organized structure. This paper focuses on implementing a simple MVC in Spring Boot, and explains the model and controller parts, only.

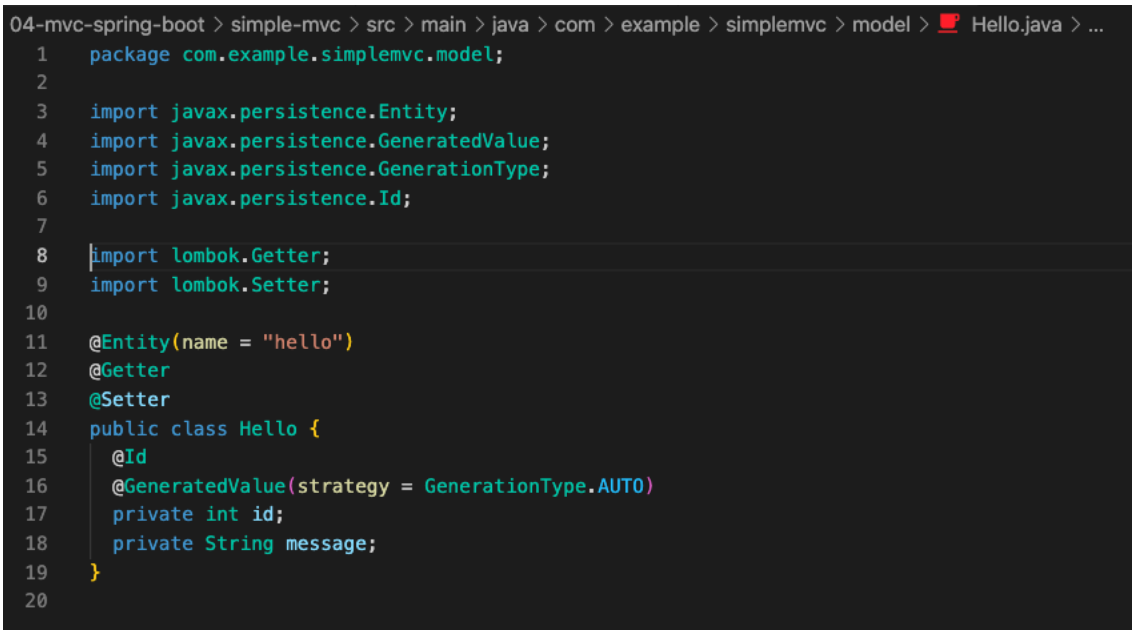
Spring Boot, being an evolution of the Spring Framework, creates stand-alone, production-grade Spring-based applications easily without a lot of boilerplate code. In MVC, the model represents a POJO (plain old Java object) carrying data. The view visualizes the data the model contains. The controller manages the data flow into the model and updates the view whenever the data changes.

To create a simple MVC in Spring Boot, a minimum of five classes are required: *Hello, HelloRepository, HelloService, HelloController, and SimpleMvcApplication,* as shown by figures 1 to 5, respectively. In this case, Figure 1 shows a model and Figure 4 shows a controller in MVC architecture. Figure 6 shows the output and sample queries.

MVC architecture organizes code in a simplified and understandable way. It also separates business logic, app functionality and the user interface (UI). Therefore, when building web applications, it is highly preferable to use MVC architecture.

Figure 1

Hello Model class



Note. The class saves the message and the id to the database named *hello.*

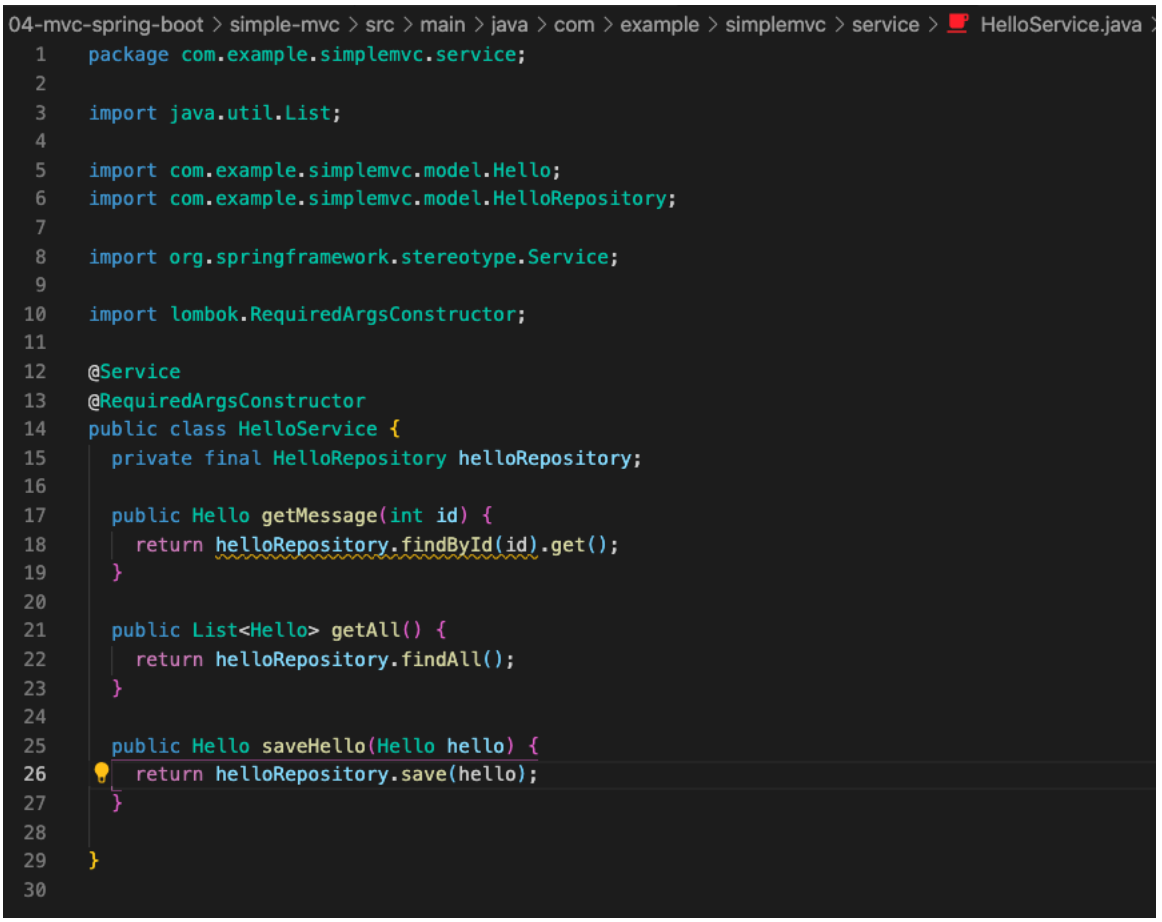
Figure 2

HelloRepository class acting between the database and the model

Note. The JpaRepository is an interface that provides useful abstractions to interact with the database.

Figure 3

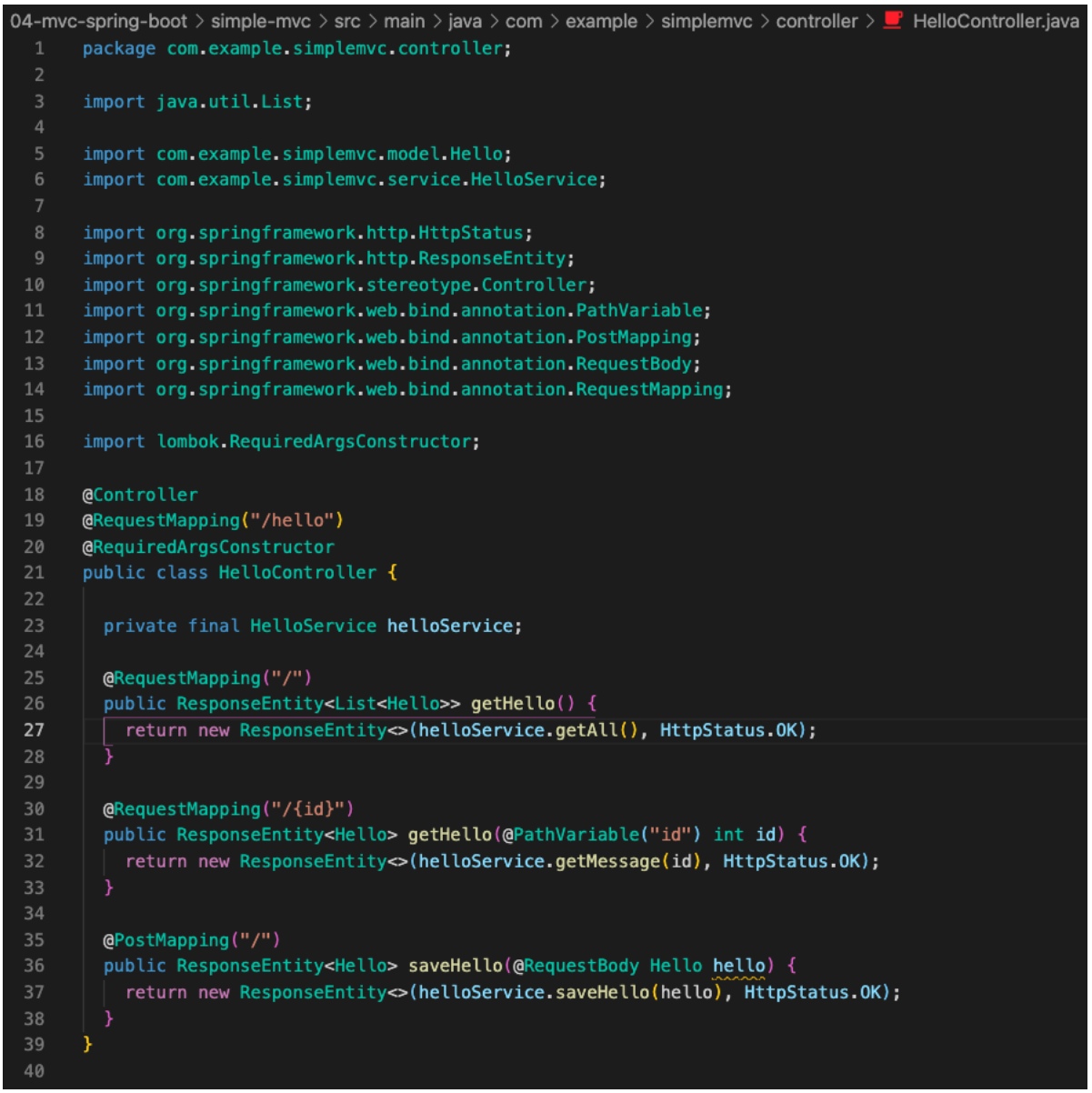
HelloService class containing business logic



Note. This class contains the business logic of the application and acts as an interface between the model and the database due to the annotation, @Service.

Figure 4

HelloController containing interfaces to web endpoints



Note. This is the class that serves HTTP requests due to the annotation, @RestController.

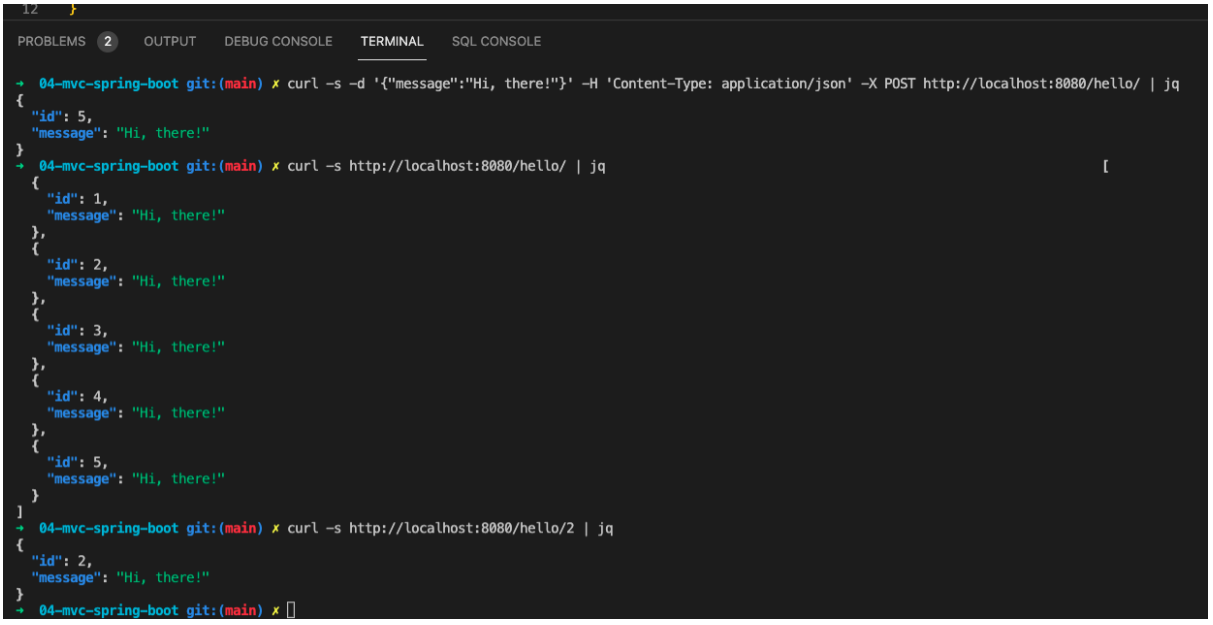
Figure 5

SImpleMvcApplication, the main entrypoint of the application

Note. This class is the main entrypoint of the spring boot application, the annotation @SpringBootApplication declares it the main entrypoint.

Figure 6

Output of HTTP Requests Using curl Command



Note. The Spring Boot application creates a web server and exposes it on port 8080. Using the curl command, HTTP requests are made to the server; the first is a POST request to create a new message, the second retrieves all messages, and the last one gets the message with id 2.

Works Cited

Kaplan, I. “A Simple Spring Boot Model View Controller (MVC) Example.” Topstone Software Consulting, Mar. 2018, topstonesoftware.com/publications/simple_spring_boot_mvc_example.html. Accessed 18 Nov. 2021.

Patel, D. “An Introduction to MVC Architecture: A Web Developer's Point of View.” DZone, 10 July 2019, dzone.com/articles/introduction-to-mvc-architecture-web-developer-poi. Accessed 18 Nov. 2021.