18-CommandLineRunner

Author: Vincent Lau

Learning Objectives

> Understand the usage of Data Transfer Object (DTO) and its benefits
>
> How to use it in Spring Boot Project.
>
> What is ModelMapper and how convenient it is.

Introduction

`CommandLineRunner` is an interface in the Spring Framework that provides a way to run custom code when a Spring Boot application starts. It is part of the Spring Boot application lifecycle and allows you to **perform initialization tasks, data loading, or other custom actions at application startup**.

To use `CommandLineRunner`, you need to create a bean that implements this interface and override its `run` method. The `run` method is executed by the Spring Boot application context

just before the application fully starts up. It provides access to the command-line arguments passed to the application, which can be useful for customizing startup behavior.

Implements CommandLineRunner

Implement the `CommandLineRunner` interface by creating a class that overrides the `run` method:

```
import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;

@Component
public class MyCommandLineRunner implements CommandLineRunner {

    @Override
    public void run(String... args) throws Exception {
        // Your custom code to run at application startup
        System.out.println("Application started. Custom initialization code goes here.");
    }
}

```

In the above example, the `MyCommandLineRunner` class implements the `CommandLineRunner` interface, and the `run` method contains the custom code to run when the application starts.

Multiple CommandLineRunners

You can create multiple `CommandLineRunner` beans, and they will be executed in the order defined by their `@Order` annotation or by their order in the Spring context.

```
@Component
@Order(1)
public class MyFirstRunner implements CommandLineRunner {
    // ...
}

@Component
@Order(2)
public class MySecondRunner implements CommandLineRunner {
    // ...
```

```
11 }
```

Start Your Spring Boot Application

When you start your Spring Boot application, the `run` method of each `CommandLineRunner` bean will be executed, allowing you to perform your desired initialization or setup tasks, before the server starts.

```java
public static void main(String[] args) {
    SpringApplication.run(MyApplication.class, args);
}
```

Common use cases for `CommandLineRunner` include database migrations, loading initial data, creating required directories or files, performing health checks, and any other tasks that need to be executed at application startup.

Bean Creating & Executing Stages

The `CommandLineRunner` beans are created as well during the application context initialization, but they are executed after the core beans, controllers, services, and repositories. The `CommandLineRunner` beans are specifically designed to perform additional tasks, such as initialization, data loading, or other custom actions, just before the application becomes fully operational.

Here's the typical order of execution during application startup:

1. Bean Definition and Initialization

The core beans, including Controllers, Services, and Repositories, are defined and initialized as part of the application context setup.

2. CommandLineRunner Bean Identification

Spring Boot identifies and instantiates all beans that implement the `CommandLineRunner` interface. These beans are created during the same application context initialization.

3. Execution of CommandLineRunner Beans

The `run` method of each `CommandLineRunner` bean is executed one by one, in the order specified by their `@Order` annotation or their order in the Spring context. These tasks are executed sequentially after the core beans are initialized.

4. Application Start

After the execution of `CommandLineRunner` beans is complete, the application context is fully loaded, and the Spring Boot application is ready to handle incoming requests.

Summary

By using `CommandLineRunner` , you can ensure that your custom code runs automatically when your Spring Boot application starts, without the need for additional configuration or manual intervention.