Learning Consolidation

Build the Skeleton of Spring Boot Application

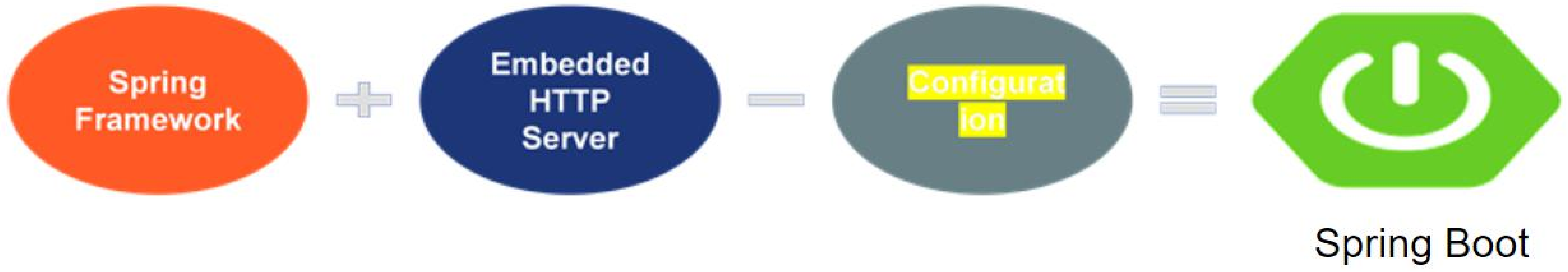
Learning Objectives

- Explore Spring Boot
- Differentiate Between Spring and Spring Boot
- Add a Service to Spring Boot

Slide Note

What is Spring Boot?

- Spring Boot makes it easy to create standalone, production-grade, Spring-based applications that you can run. (A standalone application is a software program that can be executed by itself without the need for other programs or files to be present.)
- You can develop Spring applications without Spring Boot. But this involves lots of configurations, which are time-consuming.
- Spring Boot makes it easier to configure the Spring framework by giving it a set of rules. If, as a developer, you follow those rules, Spring Boot will do all the configuration for you.
- So, as a programmer, you are required to only focus on and write your application code instead of using the configuration code of the framework.



Menu

Slide Note

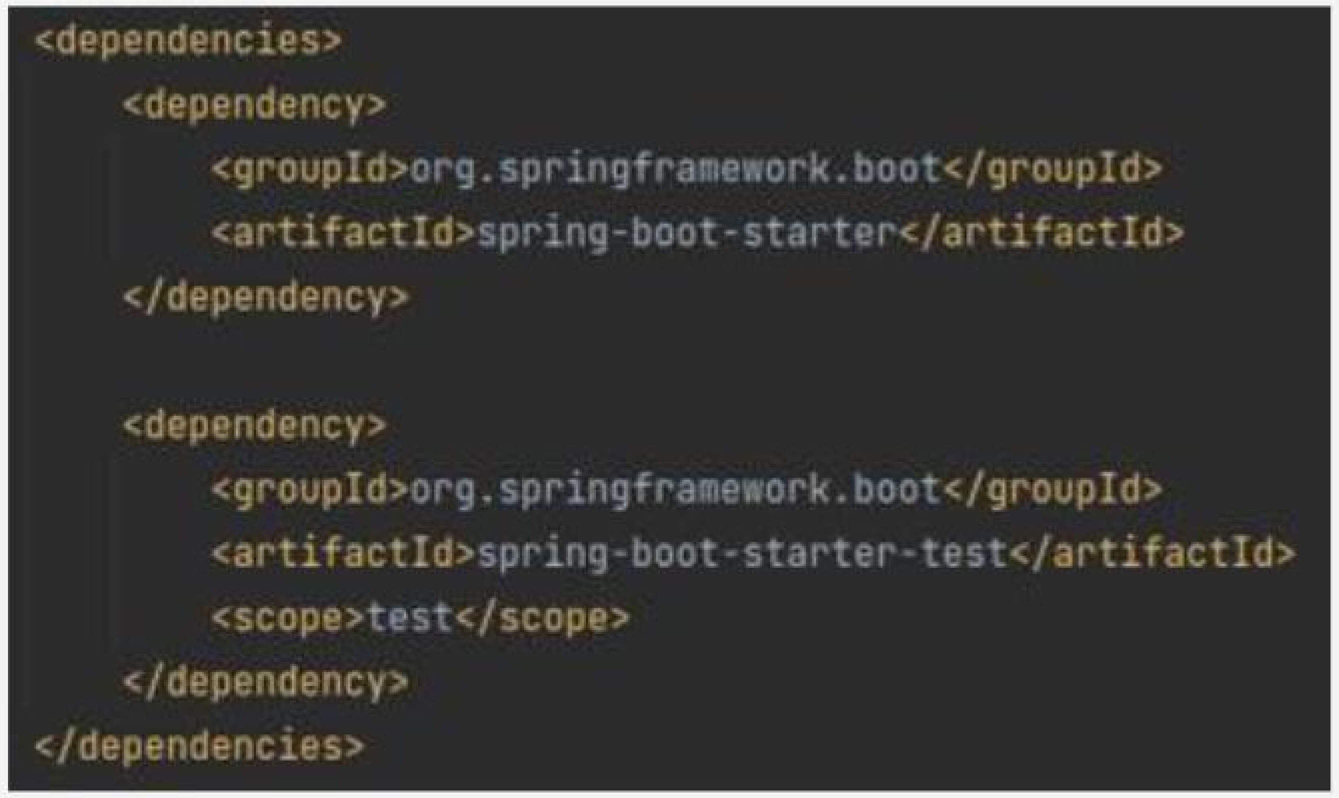
Spring vs. Spring Boot

Spring Framework	Spring Boot
It offers high versatility but comes with lots of configuration management.	It simplifies configuration management and maintains the versatility of Spring.
It needs the support of other tools to develop and run an application. For example, it requires separate installation and configuration of the Tomcat server to run or deploy a web application.	It offers an embedded server, like the Tomcat server. These services are already packaged and configured for use.
It is not a standalone application. It requires other services/packages to run an application.	It is a standalone application. No other services/packages are needed, as everything is pre-packaged.
It does not offer auto-configuration. A developer must maintain an XML configuration file for the application.	It offers auto-configuration, managing dependencies and configuration behind the scenes automatically.
It does not offer packaged dependencies, and each dependency must be manually added as needed.	It offers prepackaged dependencies as starter-dependency packages. These are prepackaged with all the required dependencies to develop an application.

Slide Note

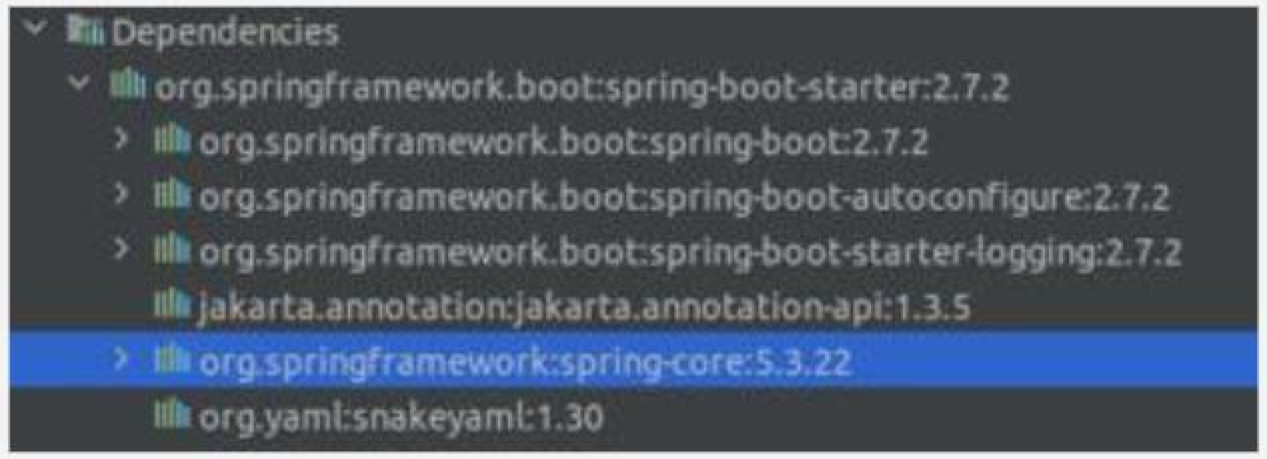
POM.xml

- In the pom.xml, there are two dependencies, `spring-boot-starter` and `spring-boot-starter-test`.
- They are the basic dependencies required to write a simple Spring Boot application.
- Starters are a one-stop shop for all Spring and related technology dependencies.
- `spring-boot-starter` brings in dependencies like Spring-core, auto configuration, etc.
- `spring-boot-starter-test` brings in dependencies required for testing, like JUnit, Mockito, and Hamcrest. We do not have to add these dependencies individually.
- So, we shall not add any individual dependencies.



```xml
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
```



```
v Dependencies
  v org.springframework.boot:spring-boot-starter:2.7.2
    > org.springframework.boot:spring-boot:2.7.2
    > org.springframework.boot:spring-boot-autoconfigure:2.7.2
    > org.springframework.boot:spring-boot-starter-logging:2.7.2
      jakarta.annotation:jakarta.annotation-api:1.3.5
    > org.springframework:spring-core:5.3.22
      org.yaml:snakeyaml:1.30
```

Menu

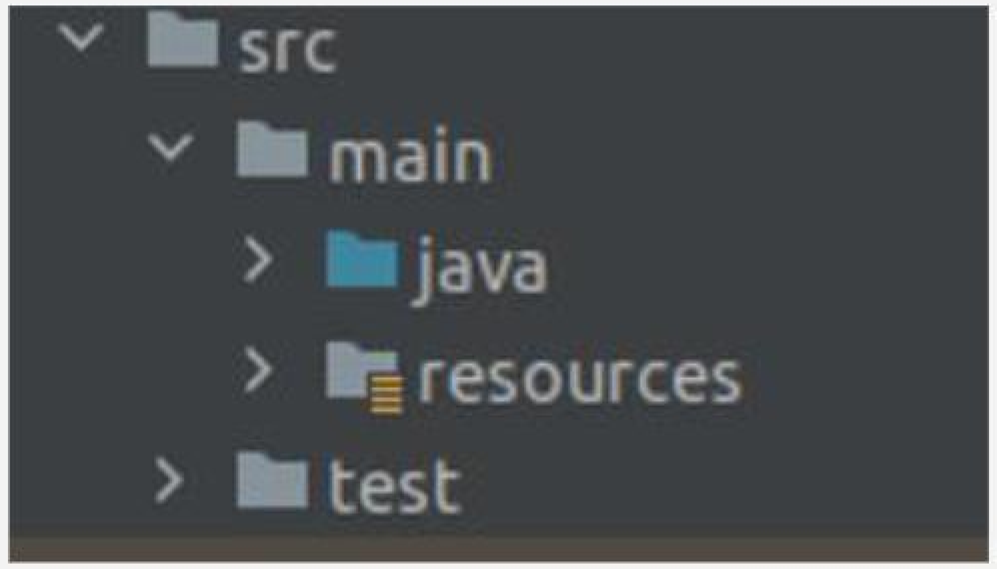
00:05 00:00:10 05/ 10

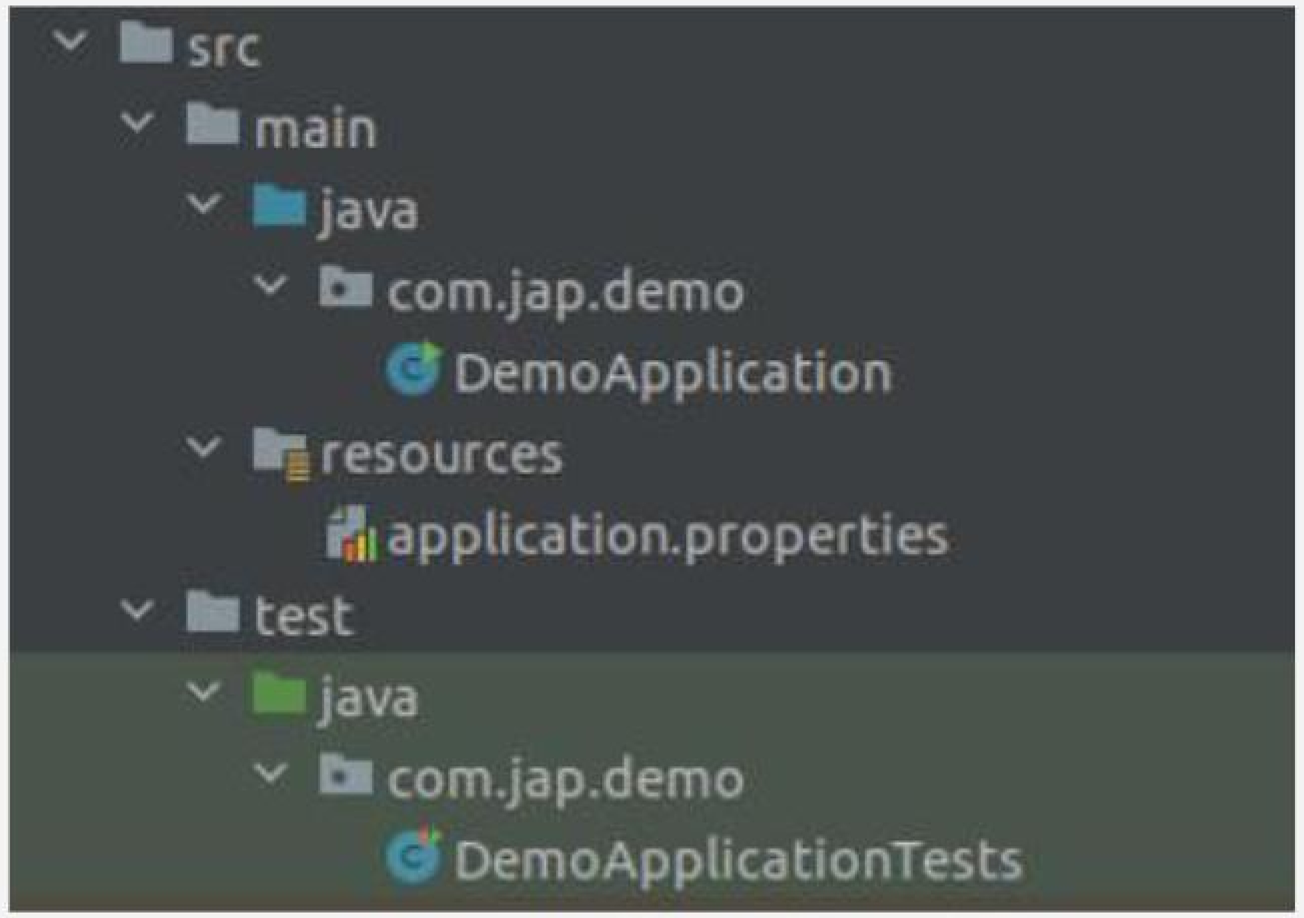
Slide Note

Another important thing to note in the project structure is the application. properties file present in the resources folder. In this file, we add project configurations, such as database connection URL, username, and password. We can also add configuration, such as the application name and the port that the Tomcat server serving this web application should listen to.

Spring Boot Structure

- Within the `src` folder, there are `main` and test folders.
- Within the main there are `java` and `resources` folders.
  - The java folder contains all the java classes.
  - Within the resources folder, there is the application.properties file which contains all project configurations.



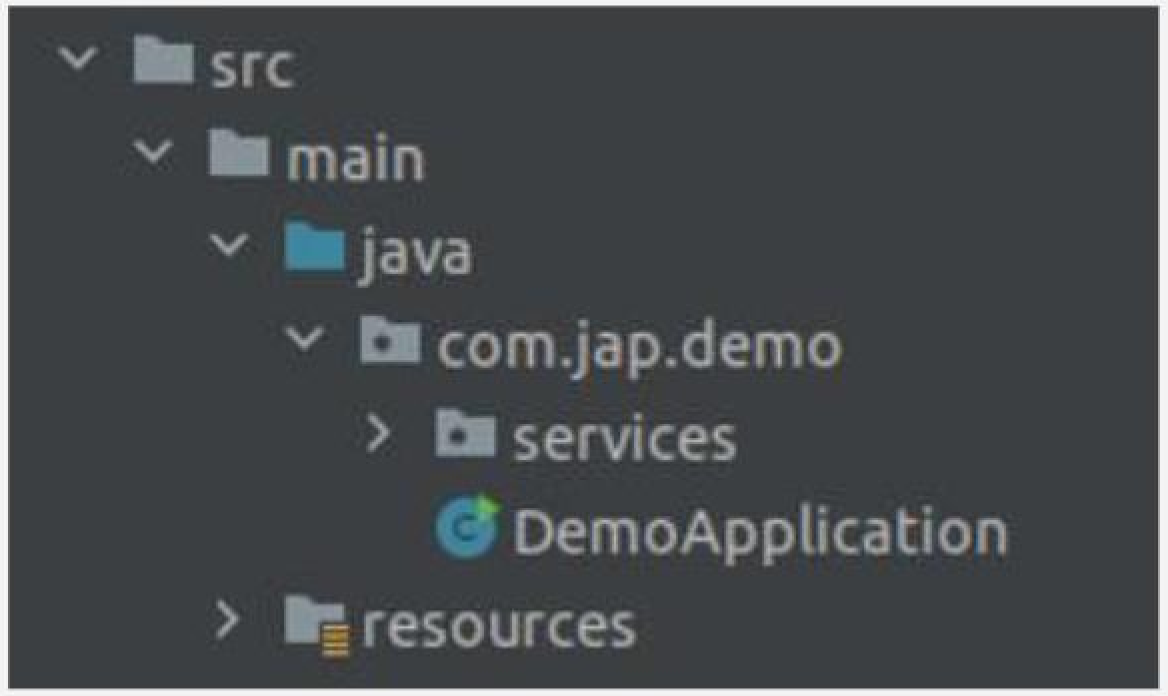


Menu

00:06 00:00:10 06/ 10

Slide Note

Adding a Service Layer



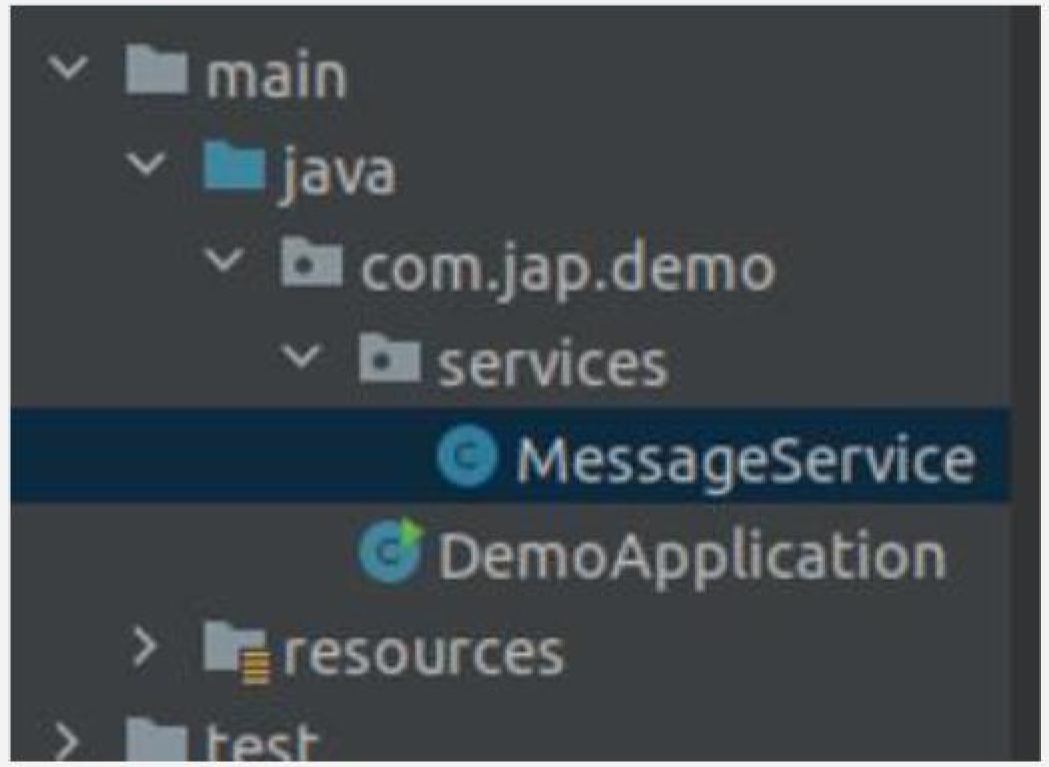
```
src
  main
    java
      com.jap.demo
        services
        DemoApplication
    resources
```

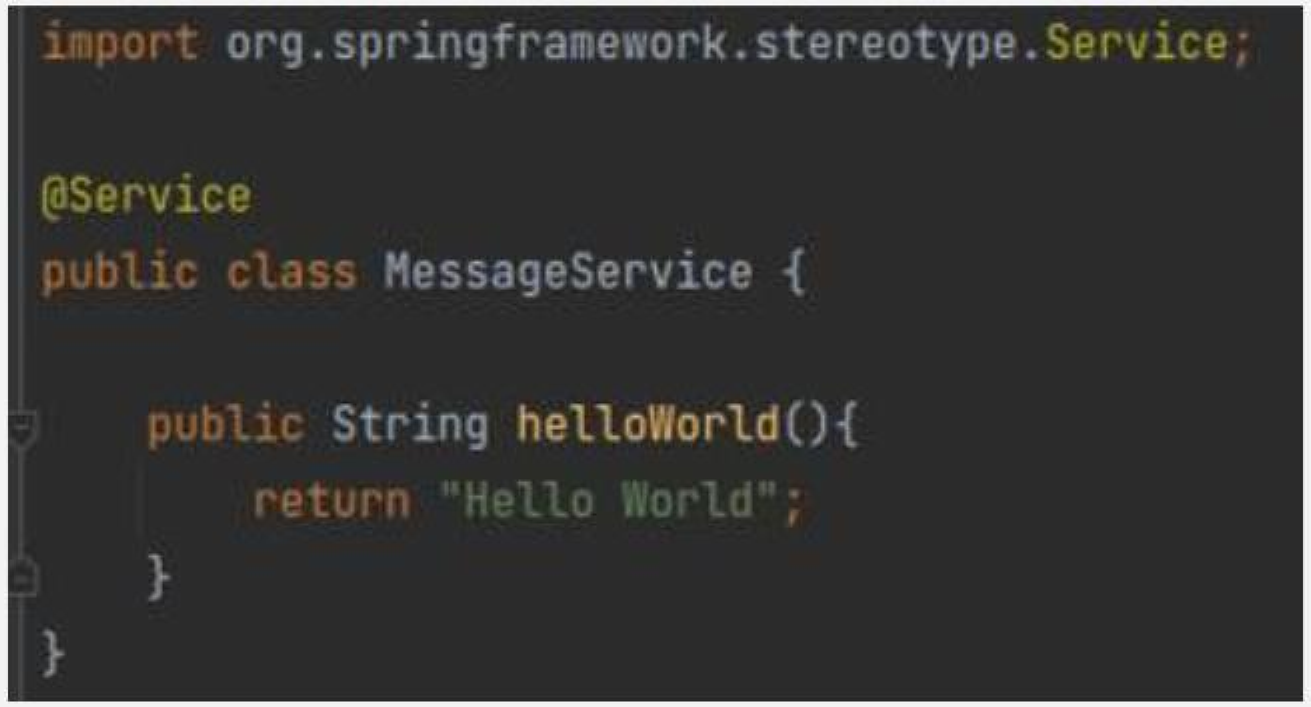
- Creating a package as a service.
- As a convention, Spring Boot expects all user-defined packages to be present inside the package containing the main method.
- So, a service package is created inside the com.jap.demo.

Menu

Slide Note

This package is having a
class MessageService that
will basically publish the message
"Hello World"





```
import org.springframework.stereotype.Service;

@Service
public class MessageService {

    public String helloWorld(){
        return "Hello World";
    }
}
```

Creating a Service Class

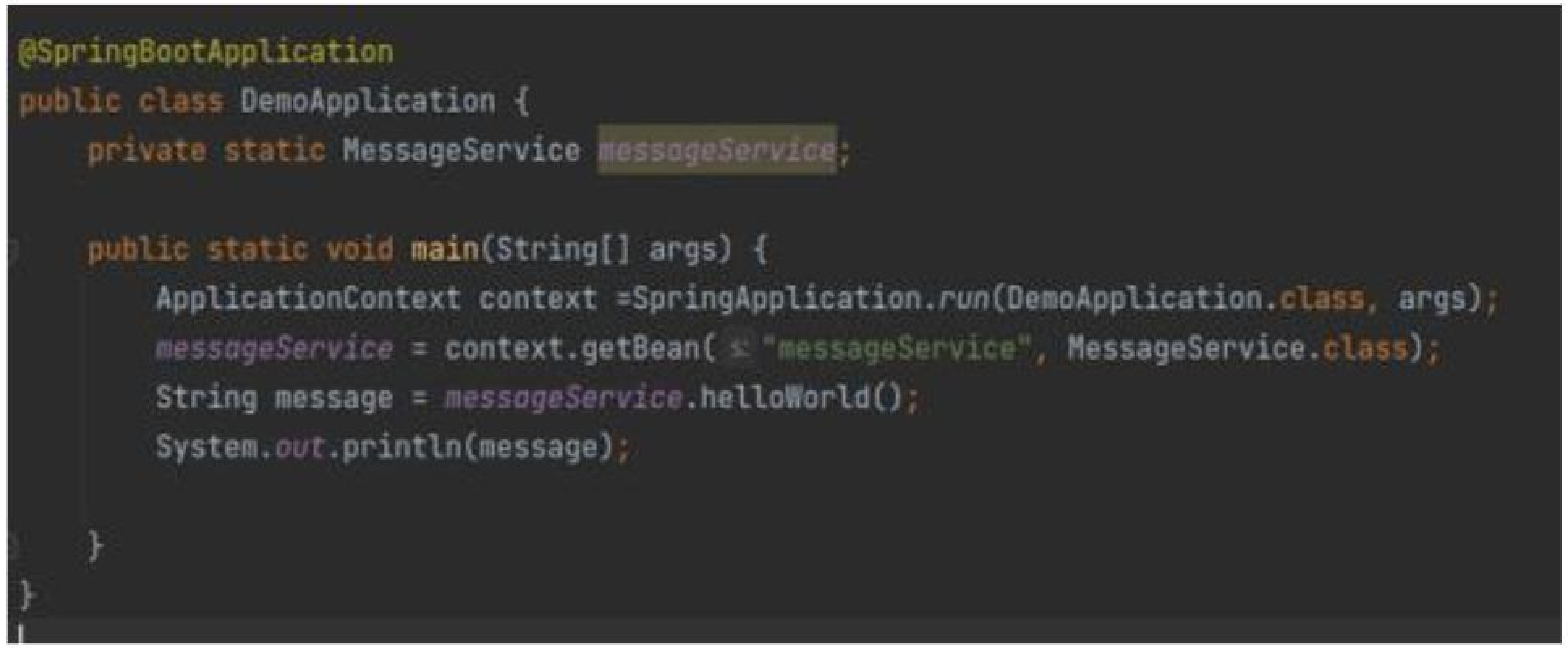
- This package has a class, MessageService, which will publish the message, "Hello World".
- This class has a string method.
- The class is annotated with @Service annotation.
- Services are the layer in which we write our business logic, and these classes are annotated with the @Service annotation.

Note: The @Service annotation will be discussed in more detail in the next Sprint.

Menu

Slide Note

The Main Method


```
@SpringBootApplication
public class DemoApplication {
    private static MessageService messageService;

    public static void main(String[] args) {
        ApplicationContext context =SpringApplication.run(DemoApplication.class, args);
        messageService = context.getBean( "messageService", MessageService.class);
        String message = messageService.helloWorld();
        System.out.println(message);

    }
}
```

- The Spring Boot application creates the ApplicationContext object, through which the other objects of the application can be accessed.
- By using context `getBean()` method, we are accessing the `MessageService` class.
- We are calling the method of `MessageService` class.

Menu

00:09 00:00:10 09/ 10