How a Spring Boot Application Works

1. Starting the application

- When a Spring Boot application is started it looks for the main class
- The main class is found by having the @SpringBootApplication annotation
- This annotation is a combination of the annotations @Configuration, @EnableAutoConfiguration and @ComponentScan

1.1 @Configuration

- It’s a stereotype annotation - indicating the purpose of the class
- Indicates that the class is a Configuration class
- Configuration classes typically contains definitions of Spring Beans, using the the @Bean annotation for defining a bean
- A Spring bean is a Java object that is managed by the Spring Container

1.2 @EnableAutoConfiguration

- Used to enable the auto-configuration feature of Spring Boot
- Auto-configuration is a feature that automatically configures the Spring application based on the dependencies that are available on the classpath
- It scans the classpath to find and configure components, such as a web server, a database, and other components

1.3 @ComponentScan

- Used to specify the base package to scan for Spring beans
- Classes outside of this base package will not be scanned and therefore not discovered by Spring Boot
- Often used in conjunction with the @Configuration (like in the situation with the @SpringBootApplication annotation)

2. Configuration

- Spring Boot uses the application.properties file or the application.yml file to configure the application

- These files provide a way to configure properties such as database setting, logging levels and application-specific settings

3. Auto-configuration

- The auto-configuration kicks in and configures the application
- The configuration is based on dependencies in the classpath
- This includes configuring the web server, the database, and other components

4. Dependency Injection

- Spring Boot injects dependencies into components, such as controllers and services

- This allow the components to be loosely coupled and easily tested

5. Application Context

- The Spring application context is created
- This is a central container for all the Spring beans of the application
- Typically beans are defined in @Configuration classes or are defined by having a Spring annotation on the class and found by component scan
- The application context manages the lifecycle of the beans and provides the necessary dependencies when they are required

6. Request Processing

- When a request is received, Spring Boot routes it to the appropriate controller method
- The Controller method processes the request, interacts with services and repositories and returns the response to the client

Demo 1 - How it works

- The @SpringBootApplication annotation
- The @ComponentScan base package

Exercise 1 - How it works

- Experiment a little with a Spring Boot project
- What happens if you remove the @SpringBootApplication annotation?
- What happens if you put the Controller outside of the package of the Application class with the @SpringBootApplication annotation (will it work?)
- Could you add your own code into the main method?