**Spring Boot**

Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run".

We take an opinionated view of the Spring platform and third-party libraries so you can get started with minimum fuss. Most Spring Boot applications need very little Spring configuration.

**Features** :

* Create stand-alone Spring applications
* Embed Tomcat, Jetty or Undertow directly (no need to deploy WAR files)
* Provide opinionated 'starter' dependencies to simplify your build configuration
* Automatically configure Spring and 3rd party libraries whenever possible
* Provide production-ready features such as metrics, health checks and externalized configuration
* Absolutely no code generation and no requirement for XML configuration

**How does it work?**

1. Spring Boot automatically configures your application based on the dependencies you have added to the project by using **@EnableAutoConfiguration** annotation. For example, if MySQL database is on your classpath, but you have not configured any database connection, then Spring Boot auto-configures an in-memory database.
2. The entry point of the spring boot application is the class contains **@SpringBootApplication** annotation and the main method.
3. Spring Boot automatically scans all the components included in the project by using **@ComponentScan** annotation.

**@AutoConfiguration?**

* Spring Boot Auto Configuration automatically configures your Spring application based on the JAR dependencies you added in the project. For example, if MySQL database is on your class path, but you have not configured any database connection, then Spring Boot auto configures an in-memory database.
* For this purpose, you need to add **@EnableAutoConfiguration** annotation or **@SpringBootApplication** annotation to your main class file. Then, your Spring Boot application will be automatically configured.

import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;  
  
@EnableAutoConfiguration  
public class DemoApplication {  
 public static void main(String[] args) {  
 SpringApplication.run(DemoApplication.class, args);  
 }  
}

**@SpringBootApplication?**

* The entry point of the Spring Boot Application is the class contains **@SpringBootApplication** annotation. This class should have the main method to run the Spring Boot application. **@SpringBootApplication** annotation includes Auto- Configuration, Component Scan, and Spring Boot Configuration.
* If you added **@SpringBootApplication** annotation to the class, you do not need to add the **@EnableAutoConfiguration, @ComponentScan** and **@SpringBootConfiguration** annotation. The **@SpringBootApplication** annotation includes all other annotations.

import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class DemoApplication {  
 public static void main(String[] args) {  
 SpringApplication.run(DemoApplication.class, args);  
 }  
}

**@ComponentScan?**

* Spring Boot application scans all the beans and package declarations when the application initializes. You need to add the @ComponentScan annotation for your class file to scan your components added in your project.

import org.springframework.boot.SpringApplication;  
import org.springframework.context.annotation.ComponentScan;  
  
@ComponentScan  
public class DemoApplication {  
 public static void main(String[] args) {  
 SpringApplication.run(DemoApplication.class, args);  
 }  
}

References :

<https://start.spring.io/>

<https://spring.io/projects/spring-boot#overview>

<https://docs.spring.io/spring-boot/docs/2.0.5.RELEASE/reference/htmlsingle/>

<https://docs.spring.io/spring-boot/docs/2.0.5.RELEASE/api/>