What is the spring boot?

First of all Spring Boot is not a framework, it is a way to ease to create stand-alone application with minimal or zero configurations. It is approach to develop spring based application with very less configuration. It provides defaults for code and annotation configuration to quick start new spring projects within no time. It leverages existing spring projects as well as Third party projects to develop production ready applications. It provides a set of Starter Pom’s or gradle build files which one can use to add required dependencies and also facilitate auto configuration.

Spring Boot automatically configures required classes depending on the libraries on its classpath. Suppose your application want to interact with DB, if there are Spring Data libraries on class path then it automatically sets up connection to DB along with the Data Source class.

**Spring Boot Reference Guide :- https://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle/**

What are the advantages of spring boot ?

* It is very easy to develop Spring Based applications with Java or Groovy.
* It reduces lots of development time and increases productivity.
* It avoids writing lots of boilerplate Code, Annotations and XML Configuration.
* It is very easy to integrate Spring Boot Application with its Spring Ecosystem like Spring JDBC, Spring ORM, Spring Data, Spring Security etc.
* It follows “Opinionated Defaults Configuration” Approach to reduce Developer effort
* It provides Embedded HTTP servers like Tomcat, Jetty etc. to develop and test our web applications very easily.
* It provides CLI (Command Line Interface) tool to develop and test Spring Boot (Java or Groovy) Applications from command prompt very easily and quickly.
* It provides lots of plugins to develop and test Spring Boot Applications very easily using Build Tools like Maven and Gradle
* It provides lots of plugins to work with embedded and in-memory Databases very easily.

What is the disadvantages of spring boot application?

It is very tough and time consuming process to convert existing or legacy Spring Framework projects into Spring Boot Applications. It is applicable only for brand new/Greenfield Spring Projects.

Why it is opinionated?

It follows “Opinionated Defaults Configuration” Approach to reduce Developer effort. Due to opinionated view of spring boot, what is required to get started but also we can get out if not suitable for application.  
• Spring Boot uses sensible defaults, “opinions”, mostly based on the classpath contents.  
• For example  
– Sets up a JPA Entity Manager Factory if a JPA implementation is on the classpath.  
– Creates a default Spring MVC setup, if Spring MVC is on the classpath.  
• Everything can be overridden easily  
– But most of the time not needed

How does it work ? how does it know it configure :-

Auto-configuration works by analyzing the classpath  
– If you forget a dependency, Spring Boot can’t configure it  
– A dependency management tool is recommended  
– Spring Boot parent and starters make it much easier  
• Spring Boot works with Maven, Gradle, Ant/Ivy  
– Our content here will show Maven

**How are properties defined? Where?**

in spring boot, we have to define properties in the application.properties

or

application.yml

file exists in classpath of application as follow.  
Example: configure default DataSource bean

database.host=localhost

database.user=admin

**What is the embedded container and a WAR?**

There is no force to go container less  
– Embedded container is just one feature of Spring Boot  
• Traditional WAR also benefits a lot from Spring Boot  
– Automatic Spring MVC setup, including DispatcherServlet  
– Sensible defaults based on the classpath content  
– Embedded container can be used during development

**. What embedded containers does Spring Boot support?**

Spring Boot includes support for embedded Tomcat, Jetty, and Undertow servers.By default the embedded server will listen for HTTP requests on port 8080.

**9. What does @EnableAutoConfiguration do? What about @SpringBootApplication?**

**@EnableAutoConfiguration annotation** on a Spring Java configuration class  
– Causes Spring Boot to automatically create beans it thinks you need  
– Usually based on classpath contents, can easily override

**What is a Spring Boot starter POM? Why is it useful?**

Starters are a set of convenient dependency descriptors that you can include in your application. The starters contain a lot of the dependencies that you need to get a project up and running quickly and with a consistent, supported set of managed transitive dependencies.

The starter POMs are convenient dependency descriptors that can be added to your application’s Maven. In simple words, if you are developing a project that uses Spring Batch for batch processing, you just have to include spring-boot-starter-batch that will import all the required dependencies for the Spring Batch application. This reduces the burden of searching and configuring all the dependencies required for a framework.

**Spring Boot supports both Java properties and YML files. Would you recognize and understand them if you saw them?**

One of the advantage Spring Boot provides us is "lesser configuration" compared to standard spring framework. Spring Boot applies it’s typical convention over configuration approach to property files. Spring boot introduced its default application properties named as "application.properties" file and it is auto detected without any spring based configurations. We need to place application.properties file inside "src/main/resources" directory.

So, by using this default file, we don’t have to explicitly register a PropertySource, or even provide a path to a property file.

Spring boot specified various common default properties inside application.properties to support Logging, AOP, Identity, Hibernat, JPA, JMX, Email, etc. etc. We dont need to specify all the default properties in all the cases. We can specify them only on-demand.

This is how spring reducing XML based configurations and changing them to simple properties.

You can find spring boot properties here:-

https://docs.spring.io/spring-boot/docs/current/reference/html/common-application-properties.html

**Customizing default Configuration for Logging:**

By adding logback.xml file to the application we can override the default logging configuration providing by the Spring Boot. This file place in the classpath (src/main/resources) of the application for Spring Boot to pick the custom configuration.

**Spring Boot can control the logging level**

Just set it in application.properties  
• Works with most logging frameworks  
– Java Util Logging, Logback, Log4J, Log4J2

logging.level.org.springframework=DEBUG

logging.level.com.acme.your.code=INFO

**13. How to reload my changes on Spring Boot without having to restart server?**

include following maven dependency in the application.

<dependency>

<groupId>org.springframework</groupId>

<artifactId>springloaded</artifactId>

<version>1.2.6.RELEASE</version>

</dependency>

**Automatic restart**  
Applications that use spring-boot-devtools will automatically restart whenever files on the classpath change. This can be a useful feature when working in an IDE as it gives a very fast feedback loop for code changes. By default, any entry on the classpath that points to a folder will be monitored for changes.

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<optional>true</optional>

</dependency>

This can be achieved using DEV Tools. With this dependency any changes you save, the embedded tomcat will restart. Spring Boot has a Developer tools (DevTools) module which helps to improve the productivity of developers. One of the key challenge for the Java developers is to auto deploy the file changes to server and auto restart the server. Developers can reload changes on Spring Boot without having to restart my server. This will eliminates the need for manually deploying the changes every time. Spring Boot doesn’t have this feature when it has released it’s first version. This was a most requested features for the developers. The module DevTools does exactly what is needed for the developers. This module will be disabled in the production environment.

**What is Actuator in Spring Boot :-**

spring Boot Actuator is a sub-project of Spring Boot. It adds several production grade services to your application with little effort on your part. There are also has many features added to your application out-of-the-box for managing the service in a production (or other) environment. They’re mainly used to expose different types of information about the running application – health, metrics, info, dump, env etc.

or

Production ready features to help you monitor and manage your application

**How to run Spring boot application to custom port?**

In application.properties, add following property.

server.port = 8080

spring boot support custom port

like server.port=8090 or any

**How to implement security for Spring boot application**

In order to add security to our Spring Boot application, we need to add the security starter dependency

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

This will include the SecurityAutoConfiguration class – containing the initial/default security configuration.

Notice how we didn’t specify the version here, with the assumption that the project is already using Boot as the parent.

**Simply put, by default, the application will get Basic Authentication enabled.**

There are some predefined properties, such as:

security.user.name=user

security.basic.enabled=true

If we start the application, we’ll notice that the default password is randomly generated and printed in the console log:

Using default security password: c8be15de-4488-4490-9dc6-fab3f91435c6

override the default password by adding our own:-

If we’ve chosen the path of disabling security auto-configuration, we naturally need to provide our own configuration.

As we’ve discussed before, this is the default security configuration; we can customize it by modifying the property file.

We can, for example, override the default password by adding our own:

**Configuration class:-**

@Autowired

public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {

auth

.inMemoryAuthentication()

.withUser("user").password("password").roles("USER");

## **Spring Boot OAuth2 Auto-Configuration**

Spring Boot has a dedicated auto-configuration support for OAuth2.

Before we get to that, let’s add the Maven dependency to start setting up our application:

<dependency>

   <groupId>org.springframework.security.oauth</groupId>

   <artifactId>spring-security-oauth2</artifactId>

</dependency>

This dependency includes a set of classes that are capable of triggering the auto-configuration mechanism defined in OAuth2AutoConfiguration class.

Now, we have multiple choices to continue, depending on the scope of our application.

### **OAuth2 Authorization Server Auto-Configuration**

If we want our application to be an OAuth2 provider, we can use @EnableAuthorizationServer.

On startup, we’ll notice in the logs that the auto-configuration classes will generate a client id and a client-secret for our authorization server and of course a random password for basic authentication.

Using default security password: a81cb256-f243-40c0-a585-81ce1b952a98

security.oauth2.client.client-id = 39d2835b-1f87-4a77-9798-e2975f36972e

security.oauth2.client.client-secret = f1463f8b-0791-46fe-9269-521b86c55b71

These credentials can be used to obtain an access token:

curl -X POST -u 39d2835b-1f87-4a77-9798-e2975f36972e:f1463f8b-0791-46fe-9269-521b86c55b71 \

 -d grant\_type=client\_credentials -d username=user -d password=a81cb256-f243-40c0-a585-81ce1b952a98 \

 -d scope=write  <http://localhost:8080/oauth/token>

**Spring Boot ExceptionHandiling?**

Spring provides a very useful way to handle exceptions using ControllerAdvice.   
We will be implementing a ControlerAdvice class which will handle all exceptions thrown by the controller class

Exceptions thrown by a Controller method is mapped to the ControllerAdvice method using @ExceptionHandler annotations.



**What is the configuration file name used by Spring Boot?**

The configuration file used in spring boot projects is application.properties. This file is very important where we would over write all the default configurations. Normally we have to keep this file under the resources folder of the project.

**How to implement Spring web using Spring boot?**  
**Web Application Convenience**

Boot automatically configures  
– A DispatcherServlet & ContextLoaderListener  
– Spring MVC using same defaults as @EnableWebMvc  
• Plus many useful extra features:  
– Static resources served from classpath  
• /static, /public, /resources or /META-INF/resources  
– Templates served from /templates  
• If Velocity, Freemarker, Thymeleaf, or Groovy on classpath  
– Provides default /error mapping  
• Easily overridden  
– Default MessageSource for I18N

**What is YAML**

• Spring Boot support YAML for Properties  
– An alternative to properties files  
application.properties

database.host = localhost

database.user = admin

database:

host: localhost

user: admin

YAML is convenient for hierarchical configuration data  
– Spring Boot properties are organized in groups  
– Examples: server, database, etc

## **Thymeleaf**

It is a server side Java template engine for web application. It's main goal is to bring elegant natural templates to your web application.

It can be integrate with Spring Framework and ideal for HTML5 Java web applications.

In the following example, we are using Thymeleaf as our HTML template and rendering it from controller.In order to use Thymeleaf we must add it into our pom.xml file like:

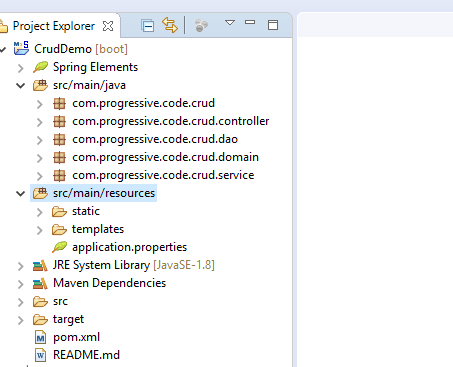
**<dependency>**

**<groupId>**org.springframework.boot**</groupId>**

**<artifactId>**spring-boot-starter-thymeleaf**</artifactId>**

**</dependency>**

project directly structure



**Spring Initializr**[**http://start.spring.io/**](http://start.spring.io/)**is great tool to bootstrap your Spring Boot projects.**

