==========
Spring Boot
==========

=> Spring Boot is one approach to develop Spring Based Applications with less configurations.

Spring Boot = Spring Framework - Xml Configurations

=> Spring Boot is not replacement for Spring Framework. Boot developed on top of Spring Framework

Note: All Spring Framework concepts can be used in Spring Boot also.

======================
Spring Boot - Advantages
======================

1) Less Configuration & No Xmls Configurations

2) Pom Starters to simplify dependencies configuration

Ex: web-starter, jpa-starter, security-starter, mail-starter

3) Auto Configuration

4) Embedded Servers (Ex: Tomcat, Jetty, Netty)

5) Actuators (Production Ready Features)

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

Spring Boot 1.0 released in 2014

Current version of Spring Boot is 3.x ===> Nov-2022

Note: Java 17 is mandatory to work with Spring Boot 3.x version

============================
Spring Boot Application Creation
============================

=> We can create boot application in 2 ways

1) Initializr website (start.spring.io)

2) Spring Starter Project in IDE

Note: If we try to create boot application using IDE then internally IDE will communicate with Intializr website to create the project.

Note: Internet Connection is mandatory for system to create Spring Boot Application.

==========
Options to choose While creating boot application
==========

Build Tool : Maven / Gradle

Language : Java / Groovy / Kotlin

groupId : It represents company name / Project Name

Ex: in.ashokit

artifactId : It represents project name / module name

Ex: MyFirstApp / UserManagementApp / TicketBookingApp

version : 0.0.1 - SNAPSHOT

SNAPSHOT - Project under development

RELEASE / FINAL - Project development completed

packageName : It represents base package in the project

Ex: in.ashokit

==========
Spring Boot Application Folder Structure
==========

src/main/java : To keep our project source code

- Application.java : It is start class of the spring boot (main class)

src/main/resources : To keep project configuration files

- application.properties / yml

src/test/java : To keep junit code (unit testing)

- ApplicationTest.java

src/test/resources : Unit testing related config files goes here

Maven dependencies : Downloaded jars will be available here

pom.xml : Maven configuration file (dependencies)

==========
Spring Boot Start Class
==========

```
@SpringBootApplication
public class Application {

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

=> It is entrypoint for boot application execution

=> @SpringBootApplication annotation is equal to below 3 annotations

@SpringBootConfiguration
@EnableAutoConfiguration
@ComponentScan

=> Spring Boot start class will act as Configuration class because of @SpringBootConfiguration

annotation

=> In Spring boot application auto configuration feature will be available because of @EnableAutoConfiguration annotation

=> Component Scan will be performed in spring boot because of @ComponentScan annotation

Note: Package Naming convention will play important role in component scanning

basePackage : in.ashokit

in.ashokit.service -- will be scanned
in.ashokit.dato -- will be scanned
com.ashokit.util ---- will not get scanned

==========
What is SpringApplication.run ( ) method
==========

=> Spring Boot start class main ( ) method will call SpringApplication.run ( ) method.

=> SpringApplication.run ( ) is entry point for boot application execution. This run ( ) method will return reference of IOC.

Note: SpringApplication is a predefined class and it will identify what type of application we have created based on dependencies added in pom.xml file.

=> If we add "spring-boot-starter" ===> standalone app
=> for standalone app, It will use "AnnotationConfigApplicationContext" class to start IoC container

=> If we add "spring-boot-starter-web" ===> Web application
=> For web apps, "AnnotationConfigServletWebServerApplicationContext" class will used to start IoC container

=> If we add "spring-boot-starter-webflux" ===> Reactive Application
=> For reactive app, "AnnotationConfigReactiveWebServerApplicationContext" class will be used to start IoC container

Note: Based on Type of our application, It will start IOC container.

=> run ( ) method will print banner on the console (i.e spring logo).

=> run ( ) method will start IoC container

=> run ( ) method will call runners

=> run ( ) method will return context of IoC container

######## ConfigurableApplicationContext context = SpringApplication.run(Application.class, args); #######

==========
What is Banner in Spring Boot ?
==========

=> When we run boot application, it will print spring logo on the console that is called as Banner in Spring Boot.

=> Spring Boot banner is having 3 modes

1) Off
2) Log
3) Console (default)

=> We can set banner mode using below property in "application.properties" file

spring.main.banner-mode=off

=> We can customize banner text in spring boot application by creating "banner.txt" file in src/main/resources folder

URL To Generate ASCII Text :
https://patorjk.com/software/taag/#p=display&f=Graffiti&t=Ashok%20IT

==========
Runners in Spring Boot
==========

=> Runners are used to execute the logic only once when boot application got started

=> SpringApplication.run ( ) method will call runners

=> In Spring Boot we have 2 types Runners

1) ApplicationRunner (I)

2) CommandLineRunner (I)

```
@Component
public class MyAppRunner implements ApplicationRunner {

	@Override
	public void run(ApplicationArguments args) throws Exception {
		System.out.println("AppRunner :: run ( ) method executed.....");
	}
}

@Component
public class MyCmdRunner implements CommandLineRunner{

	@Override
	public void run(String... args) throws Exception {
		System.out.println("MyCmdRunner :: run ( ) method called...");
	}
}
```

==========
Runners Usecases
==========

1) Loading data from db to cache memory

==========
application.properties file & application.yml
==========

=> application.properties & application.yml files are used to manage configuration properties in the application

Ex: data source properties, smtp properties & application messages etc...

=> properties file is used to configure properties in linear structure

Ex:

```
spring.datasource.username=ashokit
spring.datasource.password=ashokit@123
spring.datasource.url=jdbc://mysql:localhost:3306/db
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
```

=> YML / YAML file is used to configure properties in hierarichal structure

Ex:

```
spring:
  datasource:
    url: null
    username: null
    password: null
    driver-class-name: null
  main:
    banner-mode: false
  application:
    name: app
```

=> Properties files are used only in java language

=> YML files used by several programming languages

Note: Indent spaces are very important in YML files.

=> In Properties, for every profile seperate properties file should be created.

```
application_dev.properties
application_qa.properties
application_prod.properties
```

=> In YML, multiple profiles can be configured in single YML file (three --- will be used to seperate profile)

Note: Spring Boot supports both .properties and .yml.

Note: Recommended to use .yml files

==========
Spring Boot - Summary
==========

1) What is Spring Boot

2) Spring Boot Version History

3) Advantages of Spring Boot

4) Spring Boot Application Creation

5) Spring Boot Application Folder Structure

6) What is Start Class

7) What is @SpringBootApplication

8) What is Component Scanning

9) What is base package naming convention ?

10) What is SpringApplication.run ( ) method in Spring Boot ?

11) How IoC container will be created in Spring Boot ?

12) Which class will be used to start IoC in spring boot ?

13) What is the return type of SpringApplication.run ( ) method ?

14) What is Banner in Spring Boot ? Can we customize that ?

15) What is Runner in Spring Boot ?

16) What is application.properties file in Spring Boot ?

17) What is Dependency Injection ?

18) What type of DI we can perform in Spring Boot ?

19) What is Bean Scope ?

20) What is Autowiring ?

21) What are the modes available for Autowiring ?

22) What is Bean Lifecycle ?

23) How to represent java class as Spring Bean ?

24) What is the difference between @Component & @Bean annotations ?

25) Which dependency injection is better (SI or CI or FI) ?

26) @Comonent vs @Service vs @Repository annotations ?

27) What is @Value annotation ?

28) application.properties vs application.yml

29) What is @Primary annotation ?