www.youtube.com/c/javaexpress
+91 7801007910
Java Express

Spring Framework

Spring Introduction:

- It is very popular framework *for* building java applications
- It was initially written by Rod Johnson and was first released under the Apache 2.0 license in June 2003.
- Spring is alternative framework for EJB Technology (For simplifying the problems of EJB they introduced Spring).
- Spring is light weight and opensource framework for building java applications.
    - Reason: In EJB components are heavy weight because it is forcibly to extend a class or interface for predefined API.
- It provides support to various frameworks such as Struts, Hibernate, EJB, JSF, etc.
- The main feature of Spring is Dependency Injection & IOC Container.
- To achieve loose coupling between different components by implementing dependency injection.
- Spring can be used for the development of particular layer of a real time application unlike struts [only for front end related] and hibernate [only for database related], but with spring we can develop all layers.
- Testing is very simple.
- Provides a large number of helper classes. makes things easier
- **If we use SF to develop project, we need to manage all the configurations required for that project**.
- Spring is developed by modular Fasion
    - Spring core
    - Spring context
    - Spring jdbc
    - Spring orm
    - Spring transaction
- Official Website is www.spring.io

Goals of Spring:

- Lightweight development with Java POJOS
- Dependency Injection to promote loose coupling
- Declarative programming with AOP (Aspect Oriented Programming)
    - Add our application services to given objects
- Minimize boilerplate Java Code

Spring Projects:

- Additional Spring modules built-on top of the Spring Framework
- Only use what you need
    - Spring Cloud, Spring Data
    - Spring Batch, Spring Security
    - Spring for Android, Spring Web Flow
    - Spring Web Services, Spring LDAP

- Spring | Projects

Spring Jars Download link: repo.spring.io

Types of Applications:

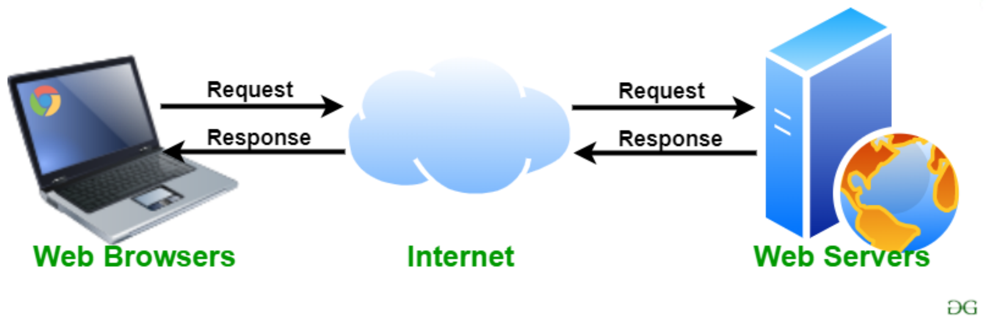
1. Standalone Application
2. Web Application
3. Distributed Application or Enterprise application

Standalone Application:

- The application which are installed in one computer and performs actions in the same computer are knows as standalone applications or Desktop Based Applications or Windows Applications.
- Note:
- According to programmer, Standalone applications means
  - No HTML input
  - No Server Environment
  - No Database Storage

Web Application:

- A web application (or web app) is application software that runs on a web server
- Web applications are accessed by the user through a web browser with an active network connection.
- Web applications are meant for C 2 B (Customer to Business)
- End users will interact with web applications directly
- Web applications include online forms, shopping carts, word processors, spreadsheets, video and photo editing, file conversion, file scanning, and email programs
  - Shopping Websites
  - Facebook
  - Gmail

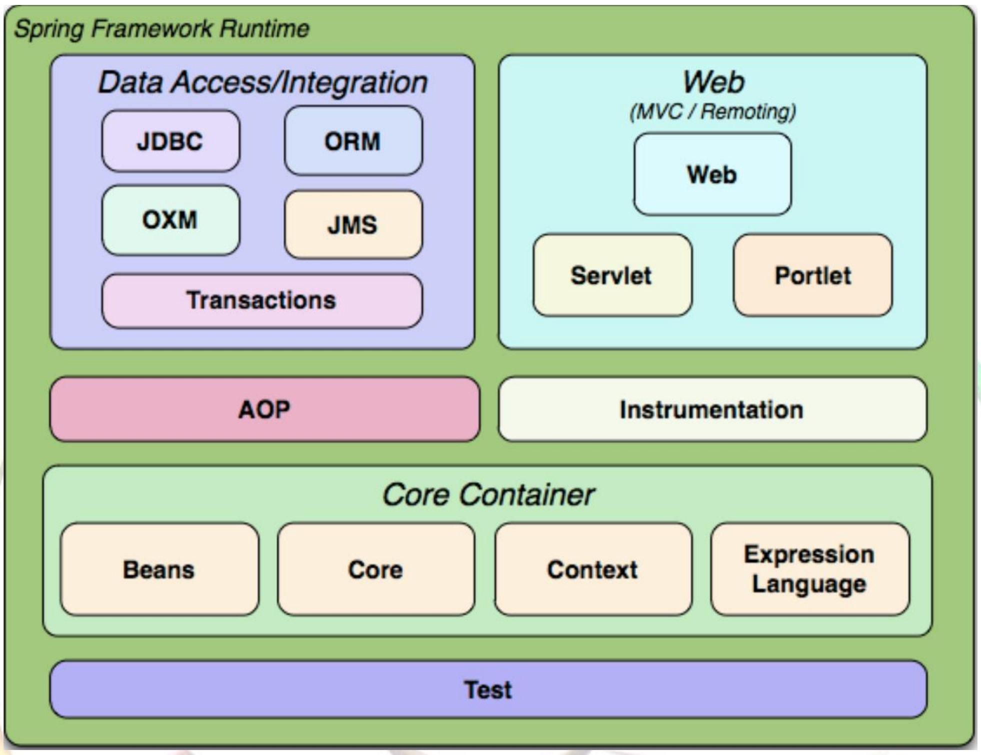


Distributed Application:

- The application which is running in distributed environment and depending on the features like Security, Load Balancing & Clustering is known as Enterprise application or
- A distributed application is **a program that runs on more than one computer and communicates through a network**
- The application which is interacting with another application is called as Distributed application

- To develop distributed applications, we will use Webservices (spring web mvc module)
- Distributed applications are meant for B 2 B
  - Flipkart ----> Payment Integration
  - Zomato -----> Banking apps
  - MakeMyTrip ---> Airlines apps

Spring Modules:



Spring Modules:

- The **Core Container** consists of the Core, Beans, Context and Expression modules.
- **The Core and Beans modules** provide the most fundamental parts of the framework and provides the IoC and Dependency Injection features.
- The **Context** module build on the solid base provided by the Core and Beans modules, This module supports internationalization (I18N), EJB, JMS, Basic Remoting.
- Spring Context module will take care of Configurations required in our applications
- **Spring DAO/Spring JDBC module** provides abstraction layer on plain JDBC to develop persistence logic.
- Spring AOP Module is used to separate business logic and secondary logic (helping code)
  - transaction
  - Security
  - Logging
  - Auditing
- Spring also provides support to **ORM solutio**ns, and it provides integration with ORM tools for easy persistence of POJO objects in relational databases. such as Hibernate, JPA, OpenJPA, TopLink, iBATIS, and so on.
- Spring ORM Module is used to develop persistence layer

  - It is used to represent data in the form of objects
- **Spring AOP** module is given to apply aspects on Spring Applications
- This group comprises of JDBC, ORM, OXM, JMS and Transaction modules. These modules basically provide support to interact with the database.
- Web group comprises of **Web-MVC, Web-Sockets**. These modules provide support to create web application.

Environment Setup

- JDK Software download from Oracle Website
  - Spring 6 requires Java 17 or higher
- Java Web Server
- Java Integrated Development Environment (IDE)
  - Eclipse
  - IntelliJ
  - STS (Spring Tool Suite)

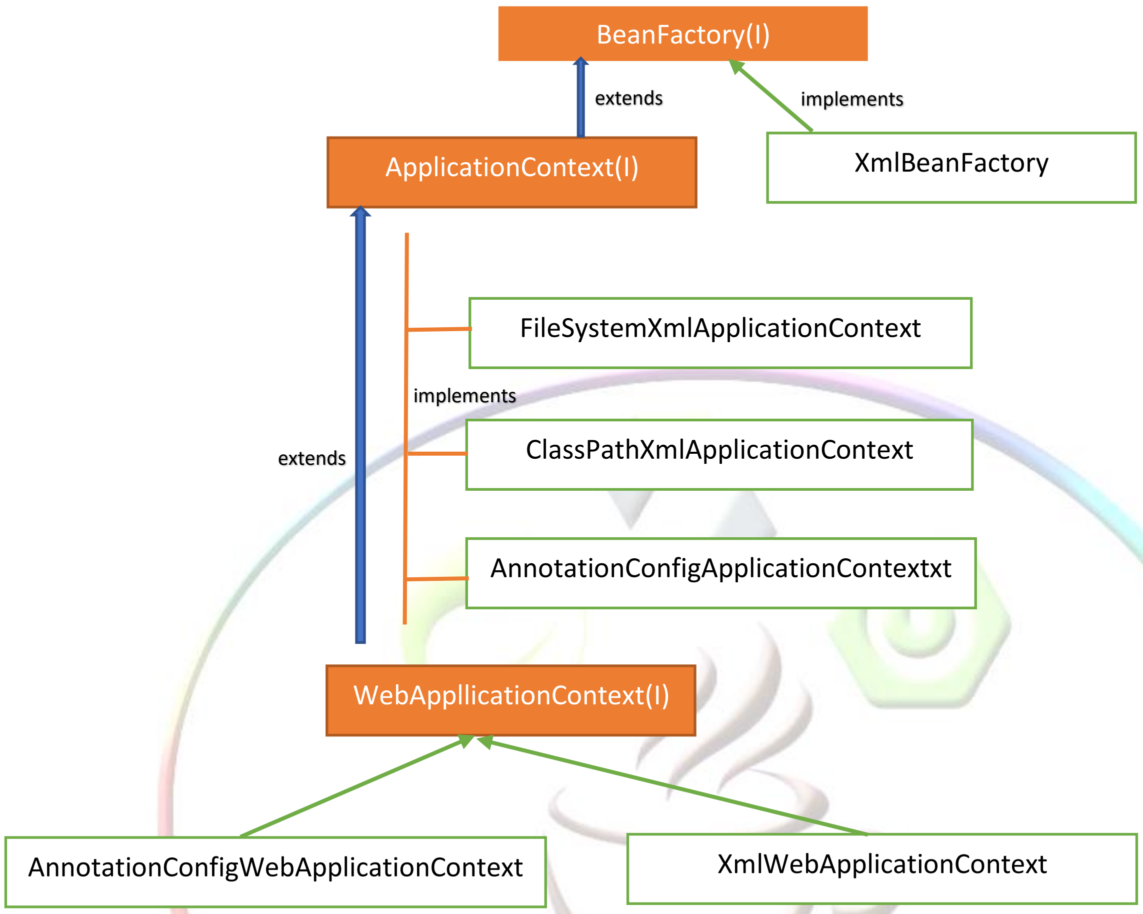
Spring IOC Container:

- Inversion of Control
- Giving control to the container to get instance of object is called IOC.
- IOC container generating objects and giving to the programmer.
- IOC Container is a mechanism to achieve loose coupling between object dependencies.
- By default, container maintains Singleton object.
- Primary Functions
  - Create and Manage Objects (IOC)
  - Inject Object's dependencies (Dependency Injection)
- It will Perform some tasks like
  - To Create Objects
  - To Configure the objects
  - To Assemble the dependencies between objects

What is Spring Bean?

- Any Normal Java class is initialized by Spring IOC Container is called Spring Bean.
- Spring IOC Manages the life cycle of spring bean cycle, bean scopes & injecting any required dependencies in the bean

How to start IOC Container?


BeanFactory(I)
extends
implements
ApplicationContext(I)
XmlBeanFactory
FileSystemXmlApplicationContext
implements
extends
ClassPathXmlApplicationContext
AnnotationConfigApplicationContextxt
WebApplicationContext(I)
AnnotationConfigWebApplicationContext
XmlWebApplicationContext

Dependency Injection:

- Dependency Inversion Principle
- The client delegates to calls to another object the responsibility of providing its dependencies.
- Dependency Means Helper Objects
- Dependency Injections is a design pattern in order to remove dependency from the programming code.
- Way of injecting properties to an object is called Dependency Injection.

Injection Types:

- There are 3 types of injection with Spring
  - Constructor Injection
  - Setter Injection
  - Field Injection - Autowired

Setter Injection:

- Inject dependencies by calling setter method on your class

Constructor Injection:

- Inject dependencies by calling parameterized constructor on your class

@Autowired:

- The @Autowired annotation is applied on fields, instance variables, setter methods, and constructors. It provides auto wiring to bean properties.
- When we apply the @Autowired to a field, Spring contain will automatically assign the fields with assigned values.
- We can also use this annotation on private fields. Consider the below code:

@Qualifier:

- The @Qualifier annotation is used along with @Autowired annotation.
- It provides more control over the dependency injection process.
- It can be specified on individual method parameters or constructors' arguments.
- It is useful for reducing the duplicity while creating more bean files of the same type and we want to use only one of them with a property.

Spring Boot

Problem Statement:

- Potential for compatibility issues:
    - We need to hunt for all the compatible libraries for the specific spring version and add them.
- Most of the times we have to configure the Data Source, JdbcTemplate, Transaction Manager, DispatcherServlet, HandlerMapping, ViewResolver etc beans in the same way.
- We should always deploy in external server.

Spring Boot:

- Spring Boot is a powerful framework for building spring applications.
- Spring Boot is an open-source java-based framework maintained by a company called Pivotal.
- Spring Boot is built on top of the Spring framework, with minimal or less configurations.
- Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run".
- Spring Boot can be used to develop Spring based applications either by using JAVA or Kotlin or Groovy
- Spring Boot applications can run embedded servers like Tomcat or Jetty directly within the application, eliminating the need for a separate server installation and configuration.
- Spring Boot provides production-ready features (Actuators) such as metrics, health checks and externalized configuration.
- Spring Boot reduces application development time.
- Spring Boot will improve Productivity.
- SpringBoot provided auto-configuration mechanism to simplify programmers' life.

Sample Project Creation using STS: Live Demo

GroupId: A unique base name of the company or group that created the project (reverse of domain Name)
ArtifactId: A unique name of the project

Features of Spring Boot:

Spring Boot Starters:

- Spring Boot Starters are pre-configured dependency packages designed to simplify the process of building Spring Boot applications.
- **Convenience:**
    - They act as single-dependency libraries that pull in all the necessary libraries for a specific purpose or technology within the spring ecosystem.
- **Focused:**
    - Each starter targets a specific functionality, like web development, data access, security, or testing.
- **Version management:**

  - They manage dependencies and ensure compatibility between different spring versions.
- **Faster Development:**
  - No need to spend time searching for and adding individual dependencies.
- **Reduced Boilerplate:**
  - Starters eliminate the need for manual configuration and bean creation for common tasks.
- **Consistency:**
  - Starters ensure projects use compatible versions of Spring libraries, avoiding potential conflicts.
- Without Spring Boot, we **need** to configure all the dependencies required in our pom.xml file
- The advantage of using the spring-boot-starter-parent POM file is that developers need not worry about find the right compatible version of different libraries such as spring, Jersey, Junit, Hibernate, Jackson and so on.

Examples of Spring Boot Starters:

- spring-boot-starter-web: Provides dependencies for building web applications with Spring MVC and Tomcat.
- spring-boot-starter-data-jpa: Includes dependencies for data access with JPA and relational databases.
- spring-boot-starter-security: Offers libraries for implementing security features like authentication and authorization.
- spring-boot-starter-test: Provides testing frameworks like JUnit and Mockito.

Overall, Spring Boot Starters are a valuable tool for developers, offering convenience, speed, and consistency in building Spring Boot applications.

Auto Configuration:

- Spring Boot auto configuration is a powerful feature that automatically configures your spring application based on the jar dependencies you've included. Imagine it as a magic helper that sets up your application's basic needs without you writing any additional configuration code. - @EnableAutoConfiguration
- **Scan the Classpath:**
  - Spring Boot scans your project's classpath for specific libraries and dependencies.
- **Identify Starters:**
  - It then identifies Spring Boot starter dependencies.
- **Configure Beans:**
  - Based on the identified starters, Spring Boot automatically creates and configures the necessary Spring beans.

Benefits

- **Faster Development:**
  - No need to write boilerplate configuration code for common tasks.
- **Reduced Errors:**
  - Auto configuration minimizes the risk of typos or inconsistencies in manual configuration.
- **Clearer Code:**

- Your code focuses on your application logic rather than configuration details.

- **Easier Maintenance:**
    - Updates to Spring Boot starters automatically update your application's configuration.

Spring Boot Annotations:

- The @SpringBootApplication is the combination of three annotations
    - @EnableAutoConfiguration,
    - @ComponentScan
    - @Configuration.
- The @SpringBootApplication annotation is used on the application class while setting up a new Spring Boot project.
- The class that is annotated with this annotation must be placed in the base package.
- The main task of this annotation is to scan the projects; it scans its sub-packages only.
    - For example, if we annotate a class that is available in a sub-package then it will not scan the main package.

@SpringBootConfiguration

- It represents our class as Configuration class
- is an alternative to the @Configuration annotation and main difference is that this annotation allows configuration to be automatically located (Especially useful for unit or integration tests)
- To customize bean creation, we can write @Bean method in this class

@ComponentScan

- It is the built-in functionality in Spring Boot Application.
- The is the process of identifying spring beans available in our application.
- It will scan from base package. After base package it will scan sub-packages of base package.
    - com.javaexpress - it will scan this package (Base Package)
    - com.javaexpress.beans - it will scan this package (Sub-package)
    - com.javaexpress.services – it will scan this package
    - in.javaexpress.dao - it will not participate scanning(different)

@EnableAutoConfiguration

- The @EnableAutoConfiguration annotation is used to implicitly defines a base "search package". Usually, it is placed on the main application class. It will automatically configure the projects by adding beans based on classpath settings, beans, and other property settings.

@Component

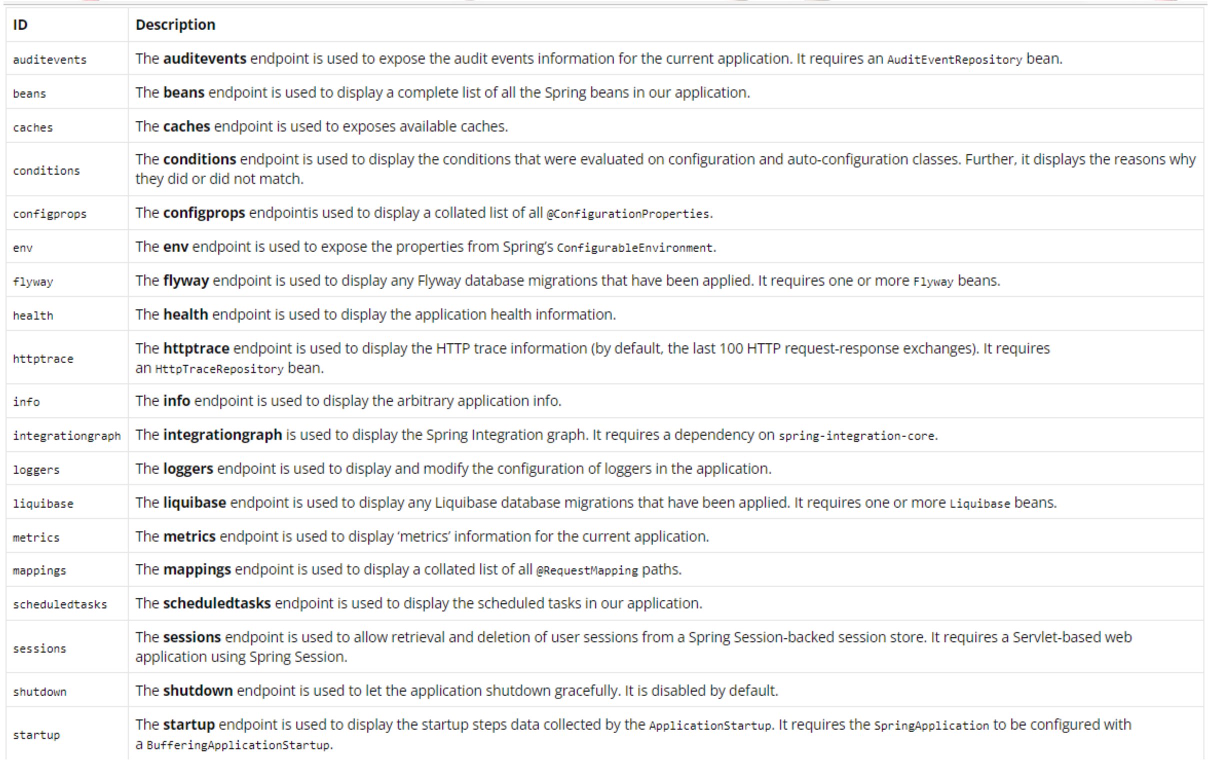
- This act as a more generic stereotype annotation to manage any component by spring whereas the other annotations are specialized for the specific use case.
- The @Component annotation is used at the class level to mark a Java class as a bean. When we mark a class with @Component annotation, it will be found during the classpath.
- The Spring framework will configure the annotated class in the application context as a Spring Bean.

- **@Bean**

- The @Bean annotation is an alternative of XML <bean> tag. If you ever have gone through the <bean> tag, you must be familiar with it's working.
- It is a method level annotation, which tells the method to produce a bean to be managed by the Spring Container.
- It also works with @Configuration annotation to create Spring beans. As the @Configuration holds methods to instantiate and configure dependencies.
- The methods, which are annotated with @Bean annotation acts as bean ID and it creates and returns the actual bean.

Spring Boot Actuators:

- Actuators are used to provide production ready features of our application
- Using Actuators, we can monitor and manage our applications
- Actuator endpoints are available to get information about application
- The actuator starter dependency should be added in pom.xml
- Spring Boot Actuators providing following Details:
    - What beans have been configured in the Spring Application Context
    - Information about the application
    - Health of the application
    - What are the configuration properties, Environment variables, System Variables, Command Line Arguments are available in our application?
    - A trace of recent HTTP Requests handled by our application.
    - Various metrics pertaining to memory usage, Garbage collection, web requests & Data source usage.
- To get complete endpoints list use below URL
    - http://localhost:8080/actuator



ID	Description
`auditevents`	The **auditevents** endpoint is used to expose the audit events information for the current application. It requires an `AuditEventRepository` bean.
`beans`	The **beans** endpoint is used to display a complete list of all the Spring beans in our application.
`caches`	The **caches** endpoint is used to exposes available caches.
`conditions`	The **conditions** endpoint is used to display the conditions that were evaluated on configuration and auto-configuration classes. Further, it displays the reasons why they did or did not match.
`configprops`	The **configprops** endpointis used to display a collated list of all `@ConfigurationProperties`.
`env`	The **env** endpoint is used to expose the properties from Spring's `ConfigurableEnvironment`.
`flyway`	The **flyway** endpoint is used to display any Flyway database migrations that have been applied. It requires one or more `Flyway` beans.
`health`	The **health** endpoint is used to display the application health information.
`httptrace`	The **httptrace** endpoint is used to display the HTTP trace information (by default, the last 100 HTTP request-response exchanges). It requires an `HttpTraceRepository` bean.
`info`	The **info** endpoint is used to display the arbitrary application info.
`integrationgraph`	The **integrationgraph** is used to display the Spring Integration graph. It requires a dependency on `spring-integration-core`.
`loggers`	The **loggers** endpoint is used to display and modify the configuration of loggers in the application.
`liquibase`	The **liquibase** endpoint is used to display any Liquibase database migrations that have been applied. It requires one or more `Liquibase` beans.
`metrics`	The **metrics** endpoint is used to display 'metrics' information for the current application.
`mappings`	The **mappings** endpoint is used to display a collated list of all `@RequestMapping` paths.
`scheduledtasks`	The **scheduledtasks** endpoint is used to display the scheduled tasks in our application.
`sessions`	The **sessions** endpoint is used to allow retrieval and deletion of user sessions from a Spring Session-backed session store. It requires a Servlet-based web application using Spring Session.
`shutdown`	The **shutdown** endpoint is used to let the application shutdown gracefully. It is disabled by default.
`startup`	The **startup** endpoint is used to display the startup steps data collected by the `ApplicationStartup`. It requires the `SpringApplication` to be configured with a `BufferingApplicationStartup`.

Logging Levels in Springboot

- In spring boot, it is very easy to log at a different level; also, spring boot provides us default logging. While using it, we do not require an external dependency because it is already included.
- Logging is used to keep track of all the activity of our application; by the use of this, we can identify the errors in our application at runtime if they occur. Also, we can check and track if the application working fine.
- These are very easy to use and handle because we do not require lots of configuration for this as the spring boot framework manages it.

Syntax:

logging. level. root = your logging level

Example: logging. level. root=DEBUG

Change Default Banner in SpringBoot:

- To change the default banner text that we usually see during the startup time, we need to put our custom text content in file or custom banner either in the classpath or in a sperate location
- When we run spring boot applications, SpringBoot logo is printing on the console that is called as Spring Boot Banner
- Banner Printing is part of SpringApplication.run(..) method
- Spring Boot banner is having below 3 modes
    - OFF
    - LOG
    - CONSOLE
- We can customize banner text in SpringBoot by creating banner.txt file in src/main/resources folder.
- First preference will be given to banner.txt file if banne.txt not available then default banner text will be printed in the console