SPRING BOOT

This is an extension of the Spring Framework designed to simplify the process of building, deploying, and running production-ready applications. It aims to provide a convention-over-configuration approach, reducing boilerplate code and configuration, making it easier for developers to create stand-alone, production-grade Spring-based Applications. Here are the key features and concepts associated with Spring Boot:

1. Opinionated Defaults:

- Spring Boot follows the convention-over-configuration principle, which means it comes with predefined defaults and sensible conventions.

- Developers can get started quickly without needing to configure many aspects manually.

2. Embedded Web Server:

- One of the notable features of Spring Boot is the inclusion of an embedded web server (Tomcat, Jetty, or Undertow by default).

- This allows developers to package their applications as standalone JARs or WARs without requiring an external web server.

3. Auto-Configuration:

- Spring Boot provides a feature called auto-configuration, where it automatically configures application components based on the dependencies present in the classpath.

- Developers can override or customize these configurations based on their specific needs.

4. Standalone Applications:

- Spring Boot applications are self-contained and do not require external servlet containers or application servers.

- They can be run with a simple `java -jar` command, making deployment and scaling straightforward.

5. Spring Boot Starters:

- Starters are pre-configured Spring Boot projects that offer a set of dependencies to simplify the inclusion of common features. For example, there are starters for web applications, data access, messaging, etc.

- Developers can include starters in their project's dependencies, reducing the need to manually configure a wide range of components.

6. Spring Boot CLI (Command Line Interface):

- The CLI is a command-line tool that allows developers to write, build, and run Spring Boot applications using Groovy scripts. It provides a quick way to prototype and develop applications.

7. Spring Boot Actuator:

- Actuator is a set of production-ready features that help monitor and manage Spring Boot applications. It provides endpoints for application health, metrics, environment properties, and more.

- These features are invaluable for operational aspects of an application in a production environment.

8. Micro services Support:

- Spring Boot is well-suited for developing microservices architectures. It supports the creation of standalone, independently deployable services, and it integrates seamlessly with Spring Cloud for building distributed systems.

9. Spring Boot DevTools:

- DevTools is a set of tools aimed at improving developer productivity. It includes features like automatic

application restarts, remote application debugging, and live reloading of changes during development.

10. Externalized Configuration:

- Spring Boot encourages externalizing configuration, allowing developers to configure their applications using properties files, YAML files, environment variables, or command-line arguments.

- The configuration is hierarchical and can be easily overridden or extended.

11. Spring Boot Testing:

- Spring Boot provides support for testing applications with a range of testing tools. It includes testing annotations, support for embedded databases, and the ability to test different layers of the application in isolation.

12. Spring Boot and Spring Cloud Integration:

- Spring Boot seamlessly integrates with Spring Cloud, a set of tools for building and deploying cloud-native applications. Spring Cloud provides solutions for distributed configuration, service discovery, and load balancing.

Spring Boot is widely adopted in the Java community due to its ease of use, productivity enhancements, and the ability to quickly create robust, production-ready applications. It has become a standard choice for developing modern Java applications and microservices.