Spring Boot is an open-source project that is part of the **larger Spring Framework ecosystem.** The Spring Framework is developed and maintained by the company **Pivotal Software, which is a division of VMware.** The development of Spring Boot is a collaborative effort involving contributions from the open-source community and is governed by the principles of the **Apache Software Foundation**

1. Why Spring Boot:

- **Simplified Development:** Spring Boot simplifies the process of building production ready applications with the Spring framework. It provides default configurations and eliminates boilerplate code, allowing developers to focus on business logic.

  Spring Boot provides a simplified and opinionated approach to configuration, **reducing** the need for **extensive XML configuration or complex setup.**

- **Rapid Development:** With its convention over configuration approach, **Spring Boot** reduces the need for **manual setup**, enabling developers to **quickly** prototype and **develop** applications.

It comes with sensible defaults and **auto-configuration**, which means you can get a basic application up and running quickly without a lot of **manual configuration**.

- **Microservices Architecture:** Spring Boot is well suited for building microservices, allowing developers to create independently deployable and scalable services.
- **Embedded Servers**: Spring Boot comes with embedded servers (like **Tomcat, Jetty, or Undertow**), **eliminating** the need for **external server** setup and configuration.
- **Opinionated Defaults: Spring Boot** provides sensible default configurations, reducing the need for developers to make decisions on certain aspects, while still allowing customization when necessary.

2. Spring Boot Overview:

- **Convention over Configuration: Spring Boot** follows the convention over configuration paradigm, reducing the need for explicit configurations by providing defaults based on conventions.
- **Auto Configuration: Spring Boot** automatically configures beans based on dependencies present in the classpath, simplifying the configuration process.
- **Spring Boot Starters: Starters** are **preconfigured templates** that help set up various types of **projects** quickly, such as web applications, data access, messaging, etc.
- **Spring Boot Actuator: Actuator** provides production ready features like **health checks**, **metrics**, and **monitoring** out of the box, facilitating better manageability of applications.
- **Externalized Configuration: Spring Boot** allows configuration properties to be externalized, making it easier to manage configurations in different environments.

3. Basic Introduction of Maven:

- **Dependency Management: Maven** is a build and project management tool that simplifies the process of managing dependencies in a project.
- **Project Object Model (POM): Maven** uses POM, an XML file that describes the project, its dependencies, build process, and configuration details.
- **Convention over Configuration: Maven** follows the convention over configuration principle, providing defaults for project structures and build lifecycle phases.
- **Centralized** Repository: **Maven** central repository is a centralized location for storing and retrieving project dependencies, ensuring easy access to a vast array of libraries.
- **Plugin Based Architecture: Maven** functionality is extended through plugins, allowing developers to customize and extend the build process.

4. Building Spring Web Application with Boot:

- **Project Setup:** Use Spring **Initializr** or **start.spring.io** to generate a **Spring Boot project** with the necessary dependencies.
- **Project Structure: Spring Boot projects** typically follow a standard structure with main application class in the root package, controllers in a separate package, and resources like templates and static files in designated folders.
- **AnnotationBased Configuration:** Leverage annotations like **@SpringBootApplication, @Controller, @RestController**, etc., for easy configuration and development.
- **Dependency Injection:** Utilize Spring's dependency injection for managing and wiring components.

- **RESTful Web Services:** Create RESTful endpoints using annotations like **@RequestMapping, @GetMapping, @PostMapping,** etc.
- **Run and Test:** Run the application using the embedded server and test the endpoints using tools like Postman or a web browser.
- **Spring Boot** includes **starters**, such as **spring-boot-starter-web**, that include the **necessary dependencies** for building web applications with **Spring MVC.**

Spring Boot and Spring MVC are **complementary**. **Spring Boot simplifies** the setup and configuration of **Spring applications**, and **Spring MVC** provides a **robust framework** for building **web applications** within the broader **Spring ecosystem**. **You can leverage both to build modern, scalable, and maintainable web applications**.