**Introduction to Spring & Spring Boot**

* Spring boot is built up on Spring Framework
* Spring Framework was first introduced in 2003 by Rod Johnson
* Before 2003 developers were creating Java Application using Java Enterprise Edition
* The problem with JEE was, they had to configure a lot before they could write a single piece of business logic
* Configurations like Server, DB and logs
* Spring Framework helped developers configure all that
* The part of Spring Framework doing all those configurations is the IoC container. It manages the lifecycle and dependencies of the objects in a Spring application
* IoC uses XML files to configure all that
* IoC containers were still taking a lot of time in configurations
* In 2009 Rod Johnson sold spring to VMware
* In 2014 through VMware, came the Spring boot, which was already configured with starter dependencies and ready for business logic.

**Spring Framework**

* Spring is a Dependency Injection framework to make Java applications loosely coupled
* Spring framework makes the development process easy for JavaEE applications
* Spring enables you to build applications from “plain old Java objects” (POJO) and to apply enterprise services non-invasively to POJOS.
* Important Components
  + Core Container
  + AOP
  + JDBC
  + Web
  + Testing

**IoC Container**

* In the Spring framework, IoC container is responsible for managing the components of an application and injecting dependencies into them. The container creates the objects (beans), wires them together, configure them, and manages their complete lifecycle.
* IoC takes care of the event driven programming (EDP), Dependency Injection (DI), Aspect Oriented Programming (AOP)
* **IoC Container**:
  + The **Inversion of Control (IoC) container** is **responsible for instantiating, configuring, and assembling beans**.
  + It starts working **as soon as the application context is loaded**, either via XML configuration, annotations, or Java-based config.
* **Dependency Injection (DI)**:
  + The process where the IoC container **injects dependencies into beans** based on the configuration (e.g., via XML).
  + DI **happens during the initialization phase** of the container, **not before**.
* **POJOs (Business Objects)**:
  + These are the actual beans managed by the IoC container.
  + They are **instantiated and injected by the container** as it processes the configuration.

**Gradle and Maven**

In the Spring framework, Gradle and Maven are build automation tools sued to manage dependencies, built the project, and automate tasks like testing, packaging and deployment.

**What Are Gradle and Maven?**

| **Feature** | **Maven** | **Gradle** |
| --- | --- | --- |
| Language | Uses **XML** (pom.xml) | Uses **Groovy/Kotlin DSL** (build.gradle or build.gradle.kts) |
| Performance | Slower builds due to XML parsing | Faster with **incremental builds** and **daemon** process |
| Popularity | Very mature and widely used | Newer, but increasingly popular |
| Configuration | More rigid and convention-based | More flexible and scriptable |
| Dependency Management | Based on a centralized repository model | Also uses Maven Central, but with more customization |