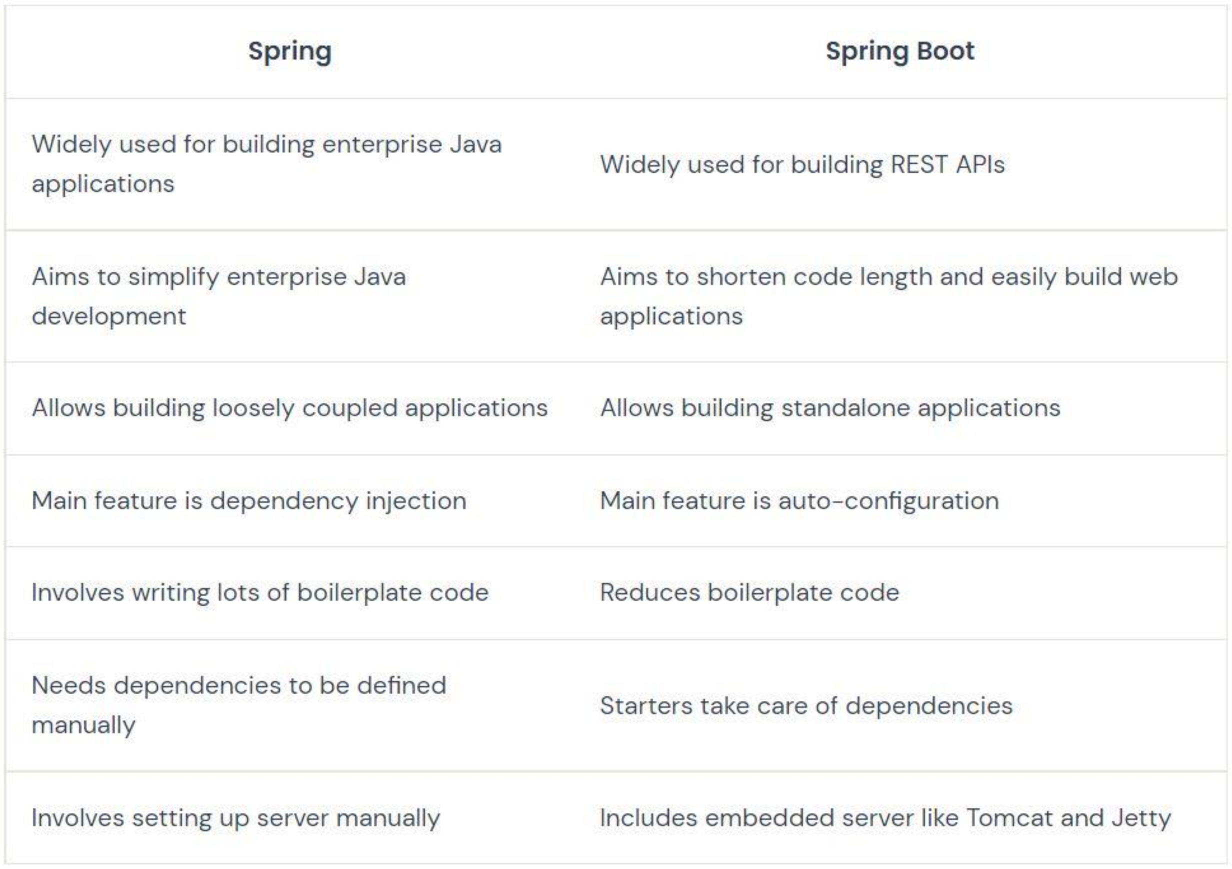
Spring Boot

Spring Boot is an extension of Spring, which eliminates the boilerplate configurations required for setting up a Spring application. Featuring default codes and annotation based configuration, Spring Boot enables a faster and more efficient development ecosystem. Since Spring Boot is built on top of Spring, it offers all the features and benefits of Spring. Spring Boot aims to reduce code length and provides developers with the easiest way to build an application.



Spring	Spring Boot
Widely used for building enterprise Java applications	Widely used for building REST APIs
Aims to simplify enterprise Java development	Aims to shorten code length and easily build web applications
Allows building loosely coupled applications	Allows building standalone applications
Main feature is dependency injection	Main feature is auto-configuration
Involves writing lots of boilerplate code	Reduces boilerplate code
Needs dependencies to be defined manually	Starters take care of dependencies
Involves setting up server manually	Includes embedded server like Tomcat and Jetty

Benefits of Spring Boot

- Can be used to build standalone applications
- No need to deploy WAR files
- Doesn't require XML configuration
- Embeds Tomcat, Jetty and Undertow directly
- Offers production ready features
- Easier to launch
- Easier management and customization

auto-configuration

Spring Boot auto-configuration attempts to automatically configure your Spring application based on the jar dependencies that you have added. For example, If *HSQLDB* is on your class path, and you have not manually configured any database connection beans, then it will auto-configure an in-memory database.

bean

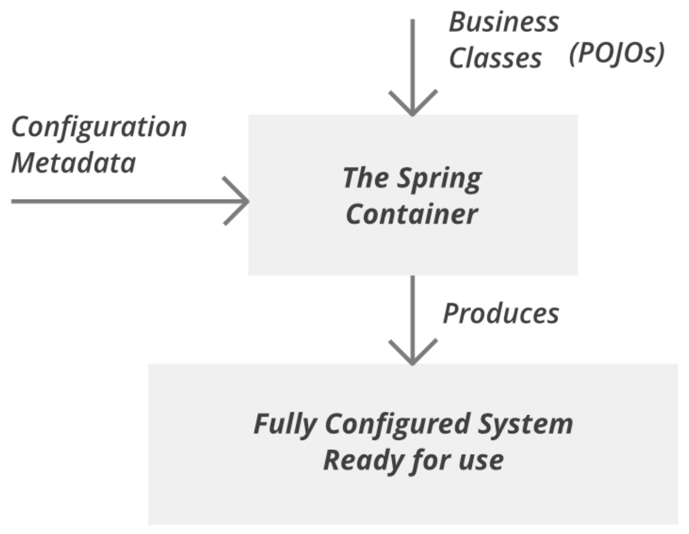
By definition, a Spring bean is an object that form the backbone of your application and that is managed by the Spring IoC container. A bean is an object that is instantiated, assembled, and otherwise managed by a Spring IoC container. Otherwise, a bean is simply one of many objects in your application.

Dependency Injection

Dependency Injection is a fundamental aspect of the Spring framework, through which the Spring container "injects" objects into other objects or "dependencies". Simply put, this allows for loose coupling of components and moves the responsibility of managing components onto the container.

IOC

Spring IoC Container is the core of Spring Framework. It creates the objects, configures and assembles their dependencies, manages their entire life cycle. The Container uses Dependency Injection(DI) to manage the components that make up the application. It gets the information about the objects from a configuration file(XML) or Java Code or Java Annotations and Java POJO class. These objects are called Beans. Since the Controlling of Java objects and their lifecycle is not done by the developers, hence the name Inversion of Control**.**


Business Classes (POJOs)
Configuration Metadata
The Spring Container
Produces
Fully Configured System Ready for use