Spring Boot + Spring Core Notes

Arjun Jadhav (He/Him)

DevOps Engineer @WebMinds | AWS Certified | CI/CD Expert | Software Devloper | Docker | AWS Resources management, Cost Optimization & Troubleshooting

Pune, Maharashtra, India · Contact info

Web Minds IT Solution

Sinhgad Institutes

About

This file contains notes on Spring core and Spring Boot.

Next, I will be posting similar notes on other important spring modules like spring MVC, Spring Cloud, Spring Security etc.

'Follow' my LinkedIn Profile '**Amol Limaye**' **to get notified** when I post the same.

Spring Core

- Spring Core technologies are absolutely integral to the spring framework
- The core technologies include
  - IoC : Inversion of Control or DI: Dependency Injection
  - Spring Beans
  - Spring Container
  - AOP: Aspect Oriented Programming

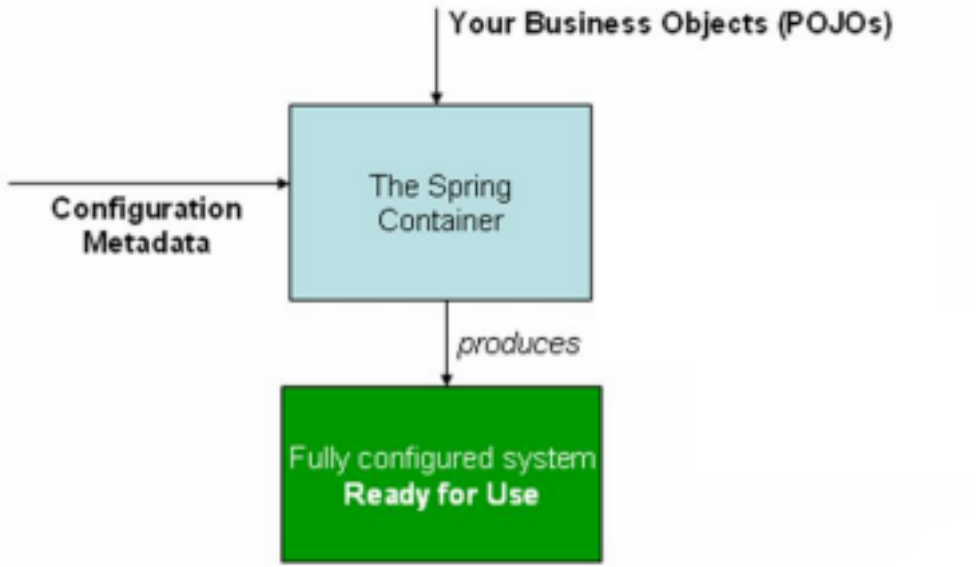
Important terms

- **Dependency Injection or Inversion of Control** – It is a process in which the objects define their dependencies. The Spring container then creates and injects those dependencies when it creates beans.
- **Beans** – A bean is an object that is instantiated and managed by Spring IoC container

Important terms

- **Spring Container** – A spring container creates, configures and assembles beans. It gets instructions on how to do that using configuration metadata. Configuration metadata could be in form of XML, annotations or Java code.

Important terms



Spring Container

Important terms

- **Spring AOP (Aspect Oriented Programming)** – It enables modularization of concerns that cut across multiple types and objects such as transaction management or logging.
  - AOP is proxy-based framework
  - Aspect is the unit of modularity in AOP just like a class is unit of modularity in OOP.

Important classes and interfaces

- **BeanFactory** – Provides the configuration framework and basic spring functionality
- **ApplicationContext** – Child interface of BeanFactory. Provides more enterprise specific functionality
- Implementations of ApplicationContext
  - **ClassPathXmlApplicationContext**
  - **FileSystemApplicationContext**
  - **WebApplicationContext**

Important annotations

Annotations to inject

- **@Autowired** - Inject an object
- **@Value** – Inject a property value

Stereotype annotations

- **@Component** – Generic stereotype
- **@Service** – For bean in service layer
- **@Repository** – For bean in database access layer
- **@Controller** – For bean in Controller layer

Important annotations

@Scope

Defines scope of bean from one of following:

- singleton(default)
- Prototype
- Request
- Session
- Application
- websocket

Important annotations

Configuration related

- **@Configuration** – Indicates a class declaring one or more beans
- **@Import** – Import a configuration class •

@ImportResource – Import spring configuration from a resource file e.g. XML

- **@PropertySource** – Load properties from a file and map onto a POJO class

Important annotations

Others

- **@Bean** – Method level annotation to create a spring bean
- **@Primary** – Mark a bean as primary in case of multiple autowire candidates
- **@Lazy** – Create bean on first access instead of startup
- **@DependsOn** – Make sure other beans are created before current bean
- **@Qualifier** – Give the name of the bean during autowiring used when there are multiple autowiring candidates
- **@Profile** – This is a way to segregate parts of your application configuration and make it available only in certain environment

Spring Boot

- Spring Boot provides a quick way to create standalone, production ready, spring based application, that you can 'just run'
- Provides embedded server (tomcat/jetty/undertow) •

Provides starter dependencies to simplify build configuration

- Provides prod-ready features like metrics, health checks and externalized configuration

Important terms

- **Starters** – These are ‘one-stop-shop’ dependencies to include in your application. They will pull all the needed spring and non-spring dependencies without you having to search for them separately. They follow pattern like ‘spring-boot-starter-*’ .

  Example-

  - spring-boot-starter-batch
  - spring-boot-starter-data-mongodb
  - Spring-boot-starter-tomcat

Important terms

- **Configuration properties** – These are config data files packaged inside or outside your jar. Spring Boot will automatically find and load these files when application starts. Example:- application.properties or application-{profile}.properties or YAML equivalent .

Important annotations

- **@SpringBootApplication** – 3 in 1 annotation for below annotations:
  1. @ComponentScan – Scan current package for spring bean candidates
  2. @EnableAutoConfiguration – Automatically configure beans based on added dependencies
  3. @SpringBootConfiguration – Enable registration of extra beans in the context or import other configuration

Important class

- **SpringApplication** – This class provides a convenient way to start your spring boot application from the main method.

Spring Boot features

- Provides **logging** library as per selected starter
- Provides **internationalization** if 'messages' resource bundle is provided at root of classpath
- Provides integration with **JSON mapping** libraries (default is Jackson)
- 'spring-boot-starter-test' provides **testing** dependencies like junit, spring test, mockito etc.

Starters for various uses

- **spring-boot-starter-web** :- Spring Boot for web application development including RESTful services
- **spring-boot-starter-batch** :- Spring Boot for Batch application
- **spring-boot-starter-websocket** :- Starter for building websocket applications

…and many more

Save this PDF for quick reference

- Source: Official Spring documentation
- Follow Arjun Jadhav to more see such content in your feed

  https://www.linkedin.com/in/kunajun77/