Java Backend Development Live-85

lecture-8

Agenda

- Spring Framework
- Spring vs Spring Boot
- Spring Core (DI and IoC)
- Spring Initializr, Spring Boot and Tomcat/Jetty
- Spring Bean Life Cycle
- Spring Bean Scope
- Application Properties

Spring Framework

The framework in a broader sense can be defined as a structure using which you can solve many technical problems.

Spring is a powerful lightweight application development framework used for Java Enterprise Edition (JEE).

In a way, it is a **framework of frameworks** because it provides support to various frameworks such as Struts, Hibernate, EJB, JSF, etc.

Spring vs Spring Boot

- **Spring**: A comprehensive framework for enterprise Java development that provides a wide range of features for building robust applications. It focuses on IoC, AOP, and modularity.
- **Spring Boot:** An extension of the Spring Framework that simplifies development with auto-configuration, embedded servers, and production-ready features. It aims to reduce boilerplate code and configuration, making it easier to get started with Spring-based applications.
- By using Spring Boot, you can leverage the power of the Spring Framework with greater ease and productivity, making it a popular choice for modern Java applications.

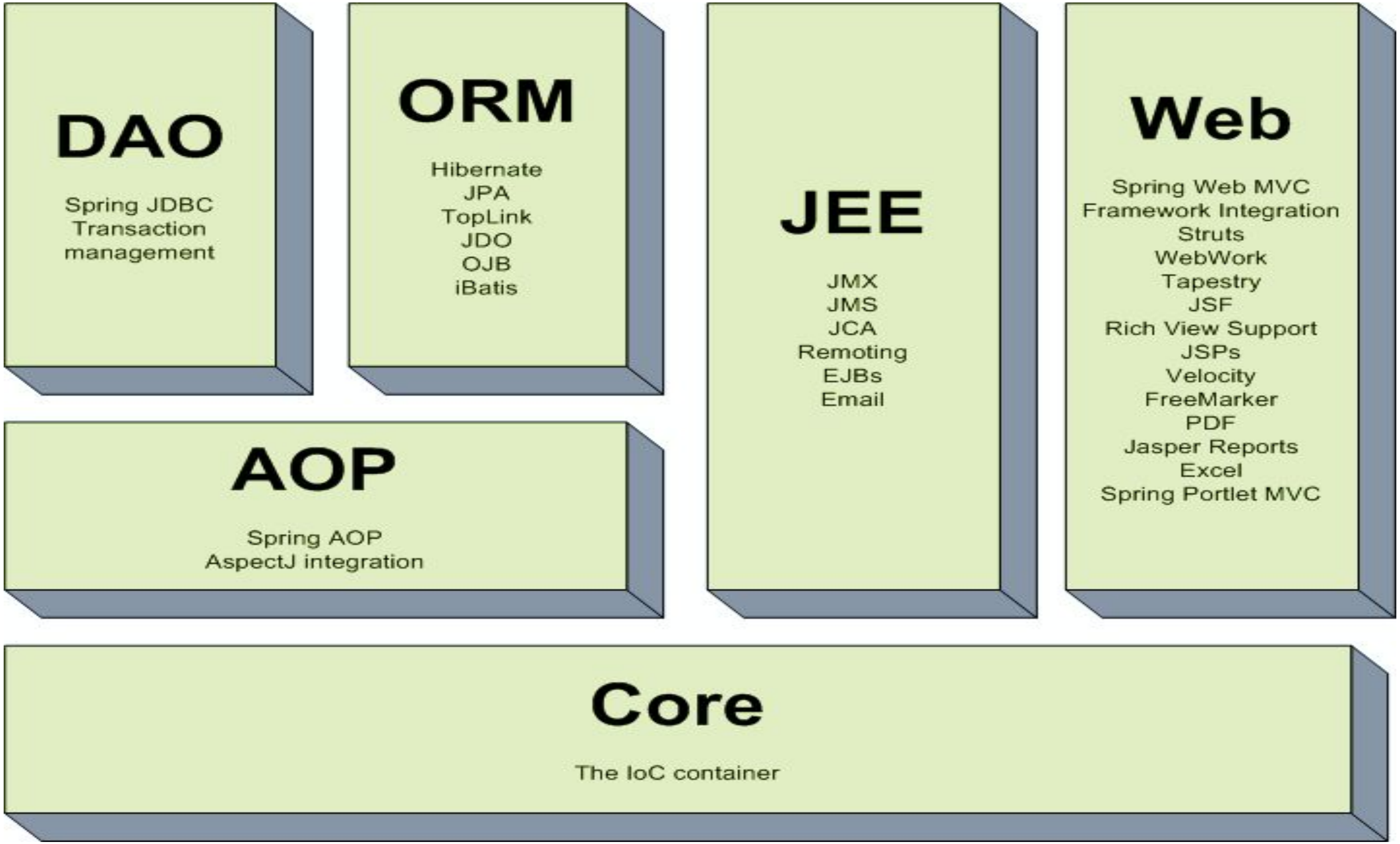
Modules Of Spring Framework

Core Container

Spring Data Access/ Integration

Spring Web

Aspect-Oriented Programming (AOP)


DAO
Spring JDBC
Transaction
management
ORM
Hibernate
JPA
TopLink
JDO
OJB
iBatis
JEE
JMX
JMS
JCA
Remoting
EJBs
Email
Web
Spring Web MVC
Framework Integration
Struts
WebWork
Tapestry
JSF
Rich View Support
JSPs
Velocity
FreeMarker
PDF
Jasper Reports
Excel
Spring Portlet MVC
AOP
Spring AOP
AspectJ integration
Core
The IoC container

Core Container

Spring Core: This module is the core of the Spring Framework. It provides an implementation for features like **IoC (Inversion of Control)** and **Dependency Injection** with a singleton design pattern.

Spring Bean: This module provides an implementation for the factory design pattern through **BeanFactory**.

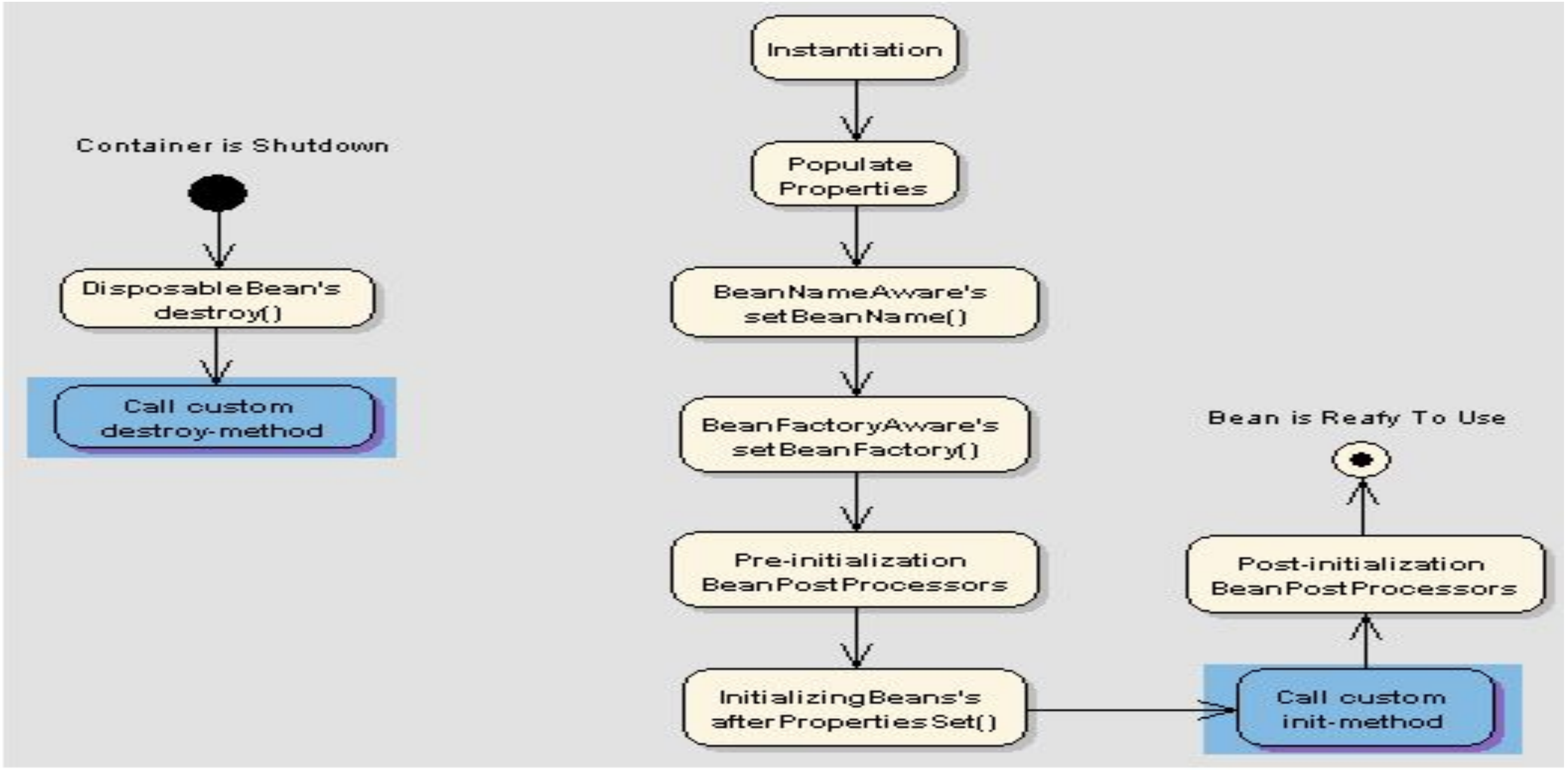
Spring Context: This module is built on the solid base provided by the Core and the Beans modules and is a medium to access any object defined and configured.

Spring Expression Languages (SpEL): This module is an extension to expression language supported by Java server pages. It provides a powerful expression language for querying and manipulating an object graph, at runtime.

Spring Core

- Spring IoC (Inversion of Control) Container is the core of Spring Framework. It **creates** the objects, **configures** and **assembles** their dependencies, **manages** their entire life cycle.
- Inversion of Control (IoC) and Dependency Injection (DI) are used interchangeably. IoC is achieved through DI.
- By DI, the responsibility of creating objects is shifted from our application code to the Spring container; this phenomenon is called **IoC**.

Spring Bean Life Cycle


Instantiation
Populate Properties
BeanNameAware's setBeanName()
BeanFactoryAware's setBeanFactory()
Pre-initialization BeanPostProcessors
InitializingBeans's afterPropertiesSet()
Call custom init-method
Post-initialization BeanPostProcessors
Bean is Reafy To Use
Container is Shutdown
DisposableBean's destroy()
Call custom destroy-method

Spring Bean Scope

singleton: the container creates a single instance of that bean; all requests for that bean name will return the **same object.**

prototype: return a different instance every time it is requested from the container.

request: The request scope creates a bean instance for a single HTTP request

session: The session scope creates a bean instance for an HTTP Session.

application: The application scope creates the bean instance for the lifecycle of a ServletContext.

websocket: the websocket scope creates it for a particular WebSocket session.

Spring Initializr

Spring Initializr is a Web-based tool that generates the Spring Boot project structure.

The Spring Initializr tool takes care of the following configuration for any Spring-based project.

- Build tool(Maven or Gradle) to build the application.
- Spring Boot version(Dependencies are added based on the version).
- Dependencies required for the project.
- Language and its version.
- Project Metadata like name, packaging (Jar or War), package name etc.

Spring Boot

Spring Boot is basically an extension of the Spring framework, which eliminates the boilerplate configurations required for setting up a Spring application.

- Opinionated ‘starter' dependencies to simplify the build and application configuration
- Embedded server to avoid complexity in application deployment
- Metrics, Health check, and externalized configuration
- Automatic config for Spring functionality – whenever possible

Replace Tomcat with Jetty or Undertow

Exclude default added spring-boot-starter-tomcat dependency and add dependency for new server (spring-boot-starter-jetty)

Jetty Dependency:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-jetty</artifactId>
</dependency>
```

Undertow Dependency:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-undertow</artifactId>
</dependency>
```

Application Properties

Spring Boot provides various properties that can be configured in the **application.properties** file. The properties have default values

The application.properties file allows us to run an application in a **different environment.**

application-dev.properties

application-test.properties

application-live.properties

```
java -jar -Dspring.profiles.active=live  project-0.0.1-SNAPSHOT.jar

java -jar -Dspring.profiles.active=dev -Dindigo.url=https:/indigo-test.com/search  project-0.0.1-SNAPSHOT.jar
```