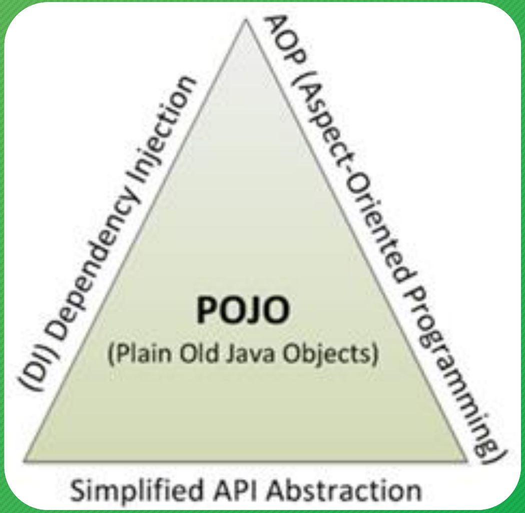
Spring

Peter Alagna Jr.

What is Spring?

Framework build to improve **EJB** (Enterprise Java Beans) development issues.

- **Reduces** the complexity of JEE development.
- POJO **Based** - Interface **Driven**.
- **Light** weight.
- **Unobtrusive** (shouldn't make development harder).
- Uses **Annotations**.
- It supports and uses **AOP** (Aspect Oriented Programming).
  - It eliminates **cross-cutting** concerns.
- Makes you use **best** practices.
  - Implicit usage of Singletons and Factories, for example.


(DI) Dependency Injection
AOP (Aspect-Oriented Programming)
POJO
(Plain Old Java Objects)
Simplified API Abstraction

Spring Solutions

Since Spring is very decoupled (because is **POJO** based), Spring increases:

- Testability.
- Maintainability.
- Scalability.
- Reduces Complexity.

Spring Modules

Core Container	Data Access	MVC	Aspects	Testing
Core	JDBC	Web	AOP	Test
Context	ORM	Servlet		
Beans	Transactions			

REVATURE

How does Spring work?

Similar to a **HashMap** of **POJOs**:

The Spring Container

Bean Name	Bean Instance
sessionFactory	…
userDao	new UserDaoHibernate()
userService	new UserService()
…	…

Bean Scopes

Spring has four (4) different bean **scopes** within the Spring Container:

- **Singleton**
  - All Spring beans are singletons by **default**.
- **Prototype**
  - Each time a bean is referenced, an instance of it is created.
- **Request**
  - The bean lifecycle joins the HTTP **request** lifecycle.
- **Session**
  - The bean lifecycle joins the HTTP **session** lifecycle.
- **Global Session (old ~ portlets)**
  - The bean lifecycle joins a HTTP **global session** lifecycle.
  - If you have a standard Servlet web application and have this scope, the regular Session scope will be used.

Creational

HTTP

Spring Configuration

Spring can be configured via **XML** or **Java** itself.

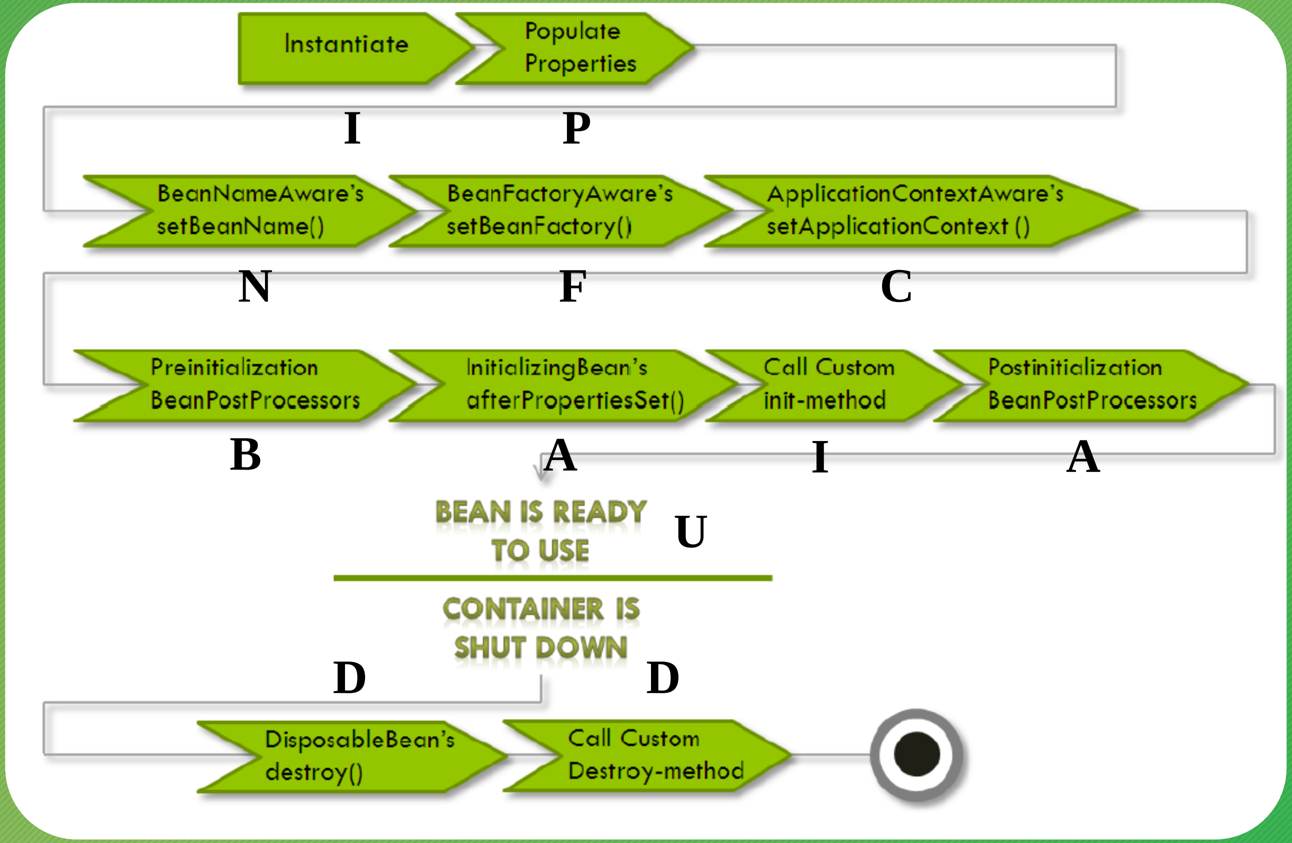
- We are going to use **XML** because:
  - XML **separates** configuration from the actual code.
  - **Simpler**.
  - No need to **recompile** our code.
- Spring configuration is usually inside **applicationContext.xml**
  - This file can be named differently, this is the standard.
- Beans are stored in the **XML**.
  - **<bean name="beanName" class="package.BeanName" scope="scopeType"></bean>**

ApplicationContext vs. BeanFactory

ApplicationContext (i) **extends** BeanFactory (i).

- **BeanFactory** holds bean definitions and instantiates them when requested (Factory).
- **ApplicationContext** is a BeanFactory with enhanced features:
  - Text messaging.
  - Generic resource loaders.
  - Bean event listeners.

Bean Lifecycle


Instantiate
Populate Properties
I
P
BeanNameAware's setBeanName()
BeanFactoryAware's setBeanFactory()
ApplicationContextAware's setApplicationContext ()
N
F
C
Preinitialization BeanPostProcessors
InitializingBean's afterPropertiesSet()
Call Custom init-method
Postinitialization BeanPostProcessors
B
A
I
A
BEAN IS READY TO USE
U
CONTAINER IS SHUT DOWN
D
D
DisposableBean's destroy()
Call Custom Destroy-method

How to remember this?

Remembering: Bean Lifecycle

- I - Instantiate.
- P - Populate properties.
- N - setName()
- F - setFactory() **
- C - setApplicationContext() **
- B - Before post processing.
- A - After populating properties.
- I - Custom Init.
- A - After post processing.
- U - Using.
- D - Destroy.
- D - Custom Destroy.

REVATURE

Materials

- Spring: https://spring.io
- Spring documentation: https://spring.io/docs