

Introduction to Spring. The Best Java framework.



IT Learning &
Outsourcing Center

www.pragmatic.bg

Lector: Stefan Vadev

Skype: stefanvadev77

E-mail: stefan.vadev@gmail.com

Facebook:

<https://www.facebook.com/stefan.vadev.7>

Copyright © Pragmatic LLC

2013–2018



Summary

- Spring overview
- Spring architecture
- Inversion of control
- Dependency injection
- Spring configuration
- Spring beans
- More

Spring

www.pragmatic.bg



IT Learning &
Outsourcing Center



spring

by Pivotal™



Spring overview

Spring framework is an open source Java platform that provides comprehensive infrastructure support for developing robust Java applications very easily and very rapidly. Spring framework was initially written by Rod Johnson and was first released under the Apache 2.0 license in June 2003.

Spring is the most popular application development framework for enterprise Java. Millions of developers around the world use Spring Framework to create high performing, easily testable, and reusable code.

Spring framework is an open source Java platform.



Spring overview

Spring is lightweight when it comes to size and transparency. The basic version of Spring framework is around 2MB.

The core features of the Spring Framework can be used in developing any Java application, but there are extensions for building web applications on top of the Java EE platform. Spring framework targets to make J2EE development easier to use and promotes good programming practices by enabling a POJO-based programming model.



Benefits of Spring

Spring enables developers to develop enterprise-class applications using POJOs. The benefit of using only POJOs is that you do not need an EJB container product such as an application server but you have the option of using only a robust servlet container such as Tomcat or some commercial product.

Spring is organized in a modular fashion. Even though the number of packages and classes are substantial, you have to worry only about the ones you need and ignore the rest.



Benefits of Spring

Spring does not reinvent the wheel, instead it truly makes use of some of the existing technologies like several ORM frameworks, logging frameworks, JEE, Quartz and JDK timers, and other view technologies.

Testing an application written with Spring is simple because environment-dependent code is moved into this framework. Furthermore, by using Java Bean style POJOs, it becomes easier to use dependency injection for injecting test data.



Benefits of Spring

Spring's web framework is a well-designed web MVC framework, which provides a great alternative to web frameworks such as Struts or other over-engineered or less popular web frameworks.

Spring provides a convenient API to translate technology-specific exceptions (thrown by JDBC, Hibernate, or JDO, for example) into consistent, unchecked exceptions.



Benefits of Spring

Lightweight IoC containers tend to be lightweight, especially when compared to EJB containers, for example. This is beneficial for developing and deploying applications on computers with limited memory and CPU resources.

Spring provides a consistent transaction management interface that can scale down to a local transaction (using a single database, for example) and scale up to global transactions (using JTA, for example).



Spring architecture

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



Dependency injection

The technology that Spring is most identified with is the Dependency Injection (DI) flavor of Inversion of Control. The Inversion of Control (IoC) is a general concept, and it can be expressed in many different ways. Dependency Injection is merely one concrete example of Inversion of Control.

When writing a complex Java application, application classes should be as independent as possible of other Java classes to increase the possibility to reuse these classes and to test them independently of other classes while unit testing. Dependency Injection helps in gluing these classes together and at the same time keeping them independent.

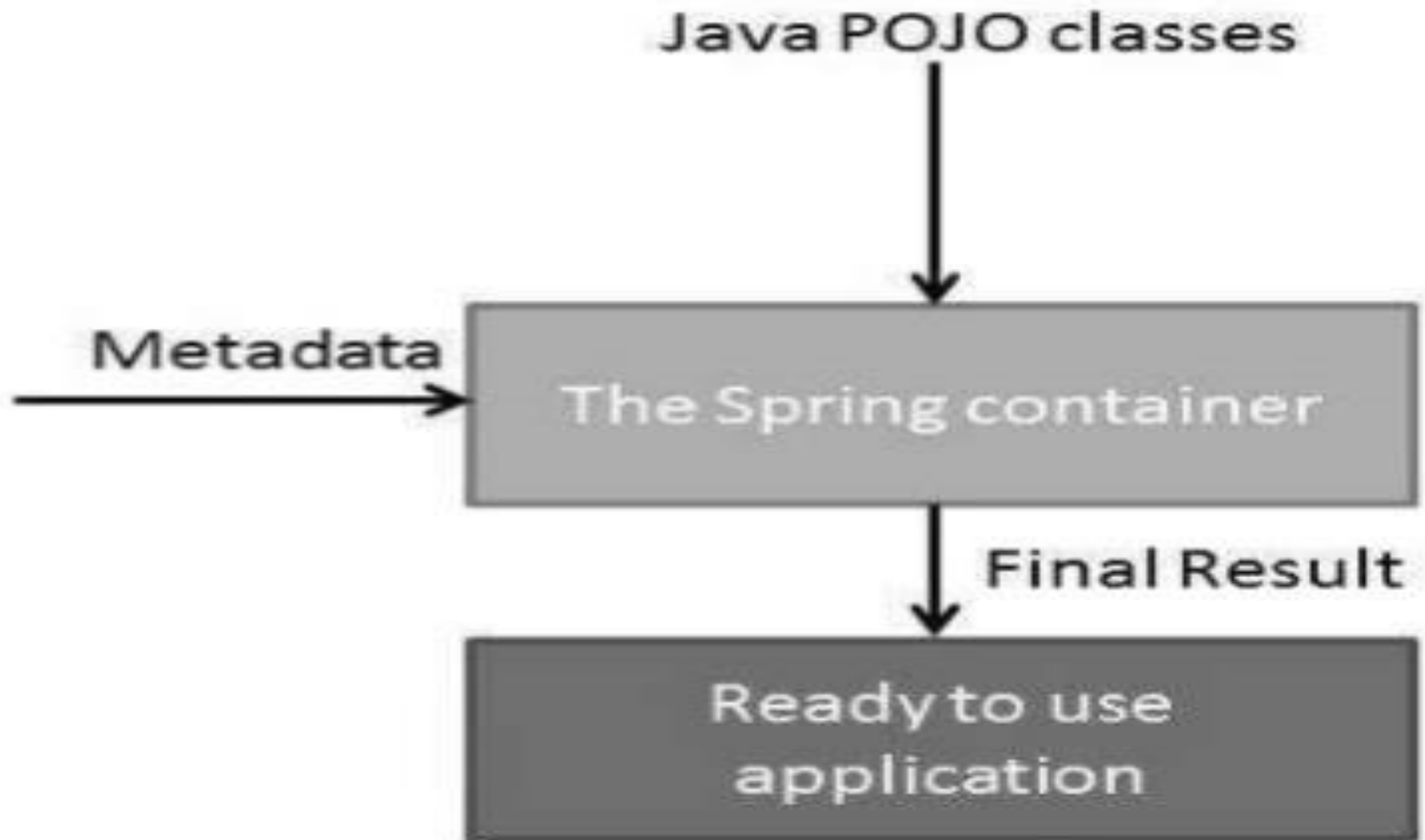


Dependency injection

Dependency injection can happen in the way of passing parameters to the constructor or by post-construction using setter methods. As Dependency Injection is the heart of Spring Framework.



Inversion of control



IOC – Inversion of control



The Spring container is at the core of the Spring Framework. The container will create the objects, wire them together, configure them, and manage their complete life cycle from creation till destruction. The Spring container uses DI to manage the components that make up an application. These objects are called Spring Beans.

IOC – Inversion of control



The container gets its instructions on what objects to instantiate, configure, and assemble by reading the configuration metadata provided. The configuration metadata can be represented either by XML, Java annotations, or Java code. The following diagram represents a high-level view of how Spring works. The Spring IoC container makes use of Java POJO classes and configuration metadata to produce a fully configured and executable system or application.



Application context

This container adds more enterprise-specific functionality such as the ability to resolve textual messages from a properties file and the ability to publish application events to interested event listeners. This container is defined by the ***org.springframework.context.ApplicationContext*** interface.

Aspect Oriented programming



One of the key components of Spring is the Aspect Oriented Programming (AOP) framework. The functions that span multiple points of an application are called cross-cutting concerns and these cross-cutting concerns are conceptually separate from the application's business logic. There are various common good examples of aspects including logging, declarative transactions, security, caching, etc.

The key unit of modularity in OOP is the class, whereas in AOP the unit of modularity is the aspect. DI helps you decouple your application objects from each other, while AOP helps you decouple cross-cutting concerns from the objects that they affect.



Spring Documentation



DOCUMENTATION

Spring configuration and more



<https://docs.spring.io/spring-framework/docs/current/Javadoc-api/org/springframework/context/annotation/Configuration.html>

Use documentation whenever needed, it is very good and very useful:

<https://spring.io/docs>

<https://spring.io/docs/reference>

For example: Spring boot docs: <https://docs.spring.io/spring-boot/docs/2.0.1.BUILD-SNAPSHOT/reference/htmlsingle/>



Spring beans

The objects that form the backbone of your application and that are managed by the **Spring IoC** container are called **beans**. A bean is an object that is instantiated, assembled, and otherwise managed by a Spring IoC container. These beans are created with the configuration metadata that you supply to the container.




Spring beans

<https://docs.spring.io/spring-javaconfig/docs/1.0.0.M4/reference/html/ch02s02.html>



Scope of beans

Bean Scopes

Scope	Description
Singleton	(Default) Only one single instance will be created
Prototype	Creates any number of instances from a single bean configuration
 Request	Scope of the bean instance will be limited to the Request life cycle
Session	Limited to session
Global session	Limited to global session-Portlet context.

```
<bean name ="student" class ="Student" scope ="prototype"/>
```



Summary

- Spring overview
- Spring architecture
- Inversion of control
- Dependency injection
- Spring configuration
- Spring beans
- More