# Spring Framework Overview

(Version 5.0.20.RELEASE)

Spring ایجاد برنامه های سازمانی جاوا را آسان می کند. با پشتیبانی از Groovy و Kotlin به‌عنوان زبان‌های جایگزین در JVM، و با انعطاف‌پذیری برای ایجاد انواع معماری‌ها بسته به نیازهای یک برنامه، هر آنچه را که برای پذیرش زبان جاوا در یک محیط سازمانی نیاز دارید، فراهم می‌کند. از Spring Framework 5.0، Spring به JDK 8+ (Java SE 8+) نیاز دارد و از قبل از JDK 9 پشتیبانی خارج از جعبه را فراهم می کند.

Spring طیف گسترده ای از سناریوهای کاربردی را پشتیبانی می کند. در یک شرکت بزرگ، برنامه‌ها اغلب برای مدت طولانی وجود دارند و باید روی یک JDK و سرور برنامه‌ای اجرا شوند که چرخه ارتقا آن خارج از کنترل توسعه‌دهنده است. برخی دیگر ممکن است به صورت یک jar با سرور تعبیه شده اجرا شوند، احتمالاً در یک محیط ابری. با این حال، برخی دیگر ممکن است برنامه‌های مستقل باشند (مانند بارهای کاری دسته‌ای یا یکپارچه) که نیازی به سرور ندارند.

بهار منبع باز است. دارای یک جامعه بزرگ و فعال است که بازخورد مستمری را بر اساس طیف متنوعی از موارد استفاده در دنیای واقعی ارائه می دهد. این به بهار کمک کرده است تا در مدت زمان طولانی با موفقیت تکامل یابد.

## 1. What We Mean by "Spring"

اصطلاح "بهار" به معنای چیزهای متفاوت در زمینه های مختلف است. می توان از آن برای اشاره به خود پروژه Spring Framework استفاده کرد، جایی که همه چیز از آنجا شروع شد. با گذشت زمان، پروژه‌های بهار دیگری در بالای چارچوب Spring ساخته شده‌اند. اغلب، وقتی مردم می گویند "بهار"، منظور آنها کل خانواده پروژه ها است. این مستندات مرجع بر اساس تمرکز دارد: خود چارچوب بهار.

Spring Framework به ماژول ها تقسیم می شود. برنامه ها می توانند ماژول های مورد نیاز خود را انتخاب کنند. در قلب ماژول های core container، از جمله مدل پیکربندی و مکانیزم تزریق وابستگی قرار دارند. فراتر از آن، Spring Framework پشتیبانی اساسی برای معماری‌های کاربردی مختلف، از جمله پیام‌رسانی، داده‌های تراکنش و پایداری، و وب فراهم می‌کند. همچنین شامل چارچوب وب Spring MVC مبتنی بر Servlet و به طور موازی، چارچوب وب واکنشی Spring WebFlux است.

نکته ای در مورد ماژول ها: جارهای فریمورک Spring اجازه استقرار در مسیر ماژول JDK 9 ("Jigsaw") را می دهند. برای استفاده در برنامه‌های دارای قابلیت Jigsaw، شیشه‌های Spring Framework 5 دارای ورودی‌های مانیفست «Automatic-Module-Name» هستند که نام‌های ماژول در سطح زبان پایدار («spring.core»، «spring.context» و غیره) را مستقل از آرتیفکت jar تعریف می‌کنند. نام‌ها (شیشه‌ها از الگوی نام‌گذاری یکسانی با "-" به جای "." پیروی می‌کنند، به عنوان مثال، "spring-core" و "spring-context"). البته، جارهای فریمورک Spring به خوبی در مسیر کلاس در JDK 8 و 9 کار می کنند.

## 2. History of Spring and the Spring Framework

بهار در سال 2003 به عنوان پاسخی به پیچیدگی مشخصات اولیه J2EE به وجود آمد. در حالی که برخی جاوا EE و Spring را با هم رقابت می کنند، Spring در واقع مکمل جاوا EE است. مدل برنامه نویسی Spring مشخصات پلت فرم Java EE را در بر نمی گیرد. بلکه با مشخصات فردی با دقت انتخاب شده از چتر EE ادغام می شود

Servlet API (JSR 340)

WebSocket API (JSR 356)

Concurrency Utilities (JSR 236)

JSON Binding API (JSR 367)

Bean Validation (JSR 303)

JPA (JSR 338)

JMS (JSR 914)

as well as JTA/JCA setups for transaction coordination, if necessary.

چارچوب Spring همچنین از مشخصات Dependency Injection (JSR 330) و Common Annotations (JSR 250) پشتیبانی می کند، که توسعه دهندگان برنامه ممکن است به جای مکانیسم های خاص Spring ارائه شده توسط Spring Framework استفاده کنند.

از Spring Framework 5.0، Spring حداقل به سطح Java EE 7 (به عنوان مثال Servlet 3.1+، JPA 2.1+) نیاز دارد - در حالی که در عین حال یکپارچه سازی خارج از جعبه با API های جدیدتر در سطح Java EE 8 ارائه می کند. (به عنوان مثال Servlet 4.0، JSON Binding API) هنگامی که در زمان اجرا با آن مواجه می شوید. این امر اسپرینگ را کاملاً با به عنوان مثال سازگار نگه می دارد. Tomcat 8 و 9، WebSphere 9 و JBoss EAP 7.

با گذشت زمان، نقش Java EE در توسعه اپلیکیشن تکامل یافته است. در روزهای اولیه Java EE و Spring، برنامه‌ها برای استقرار در سرور برنامه ایجاد شدند. امروزه، با کمک Spring Boot، برنامه‌ها به روشی توسعه‌دهنده و سازگار با فضای ابری ایجاد می‌شوند، با ظرف Servlet تعبیه‌شده و تغییر آن بی‌اهمیت است. از Spring Framework 5، یک برنامه WebFlux حتی از Servlet API مستقیماً استفاده نمی کند و می تواند روی سرورهایی (مانند Netty) که کانتینرهای Servlet نیستند اجرا شود.

بهار به نوآوری و تکامل خود ادامه می دهد. فراتر از چارچوب Spring، پروژه های دیگری مانند Spring Boot، Spring Security، Spring Data، Spring Cloud، Spring Batch و غیره وجود دارد. مهم است که به یاد داشته باشید که هر پروژه مخزن کد منبع، ردیاب مسئله و آهنگ انتشار خود را دارد. برای لیست کامل پروژه های Spring.io/projects را ببینید.

## 3. Design Philosophy

وقتی در مورد یک چارچوب یاد می گیرید ، مهم است که نه تنها چه کاری انجام می دهد بلکه چه اصولی را دنبال می کند. در اینجا اصول راهنما چارچوب بهار آورده شده است:

انتخاب را در هر سطح ارائه دهید. بهار به شما امکان می دهد تا هر چه زودتر تصمیمات طراحی را به تعویق بیندازید. به عنوان مثال ، شما می توانید ارائه دهندگان پایداری را از طریق پیکربندی بدون تغییر کد خود تغییر دهید. در مورد بسیاری از نگرانی های زیرساخت های دیگر و ادغام با API های شخص ثالث نیز همین مسئله صادق است.

دیدگاههای متنوعی را در خود جای دهید. بهار انعطاف پذیری را در بر می گیرد و در مورد نحوه انجام کارها نظر نمی شود. این برنامه از طیف گسترده ای از نیازهای برنامه با دیدگاه های مختلف پشتیبانی می کند.

سازگاری قوی به عقب را حفظ کنید. تکامل بهار با دقت موفق شده است تا تغییرات کمی بین نسخه ها را مجبور کند. بهار از طیف وسیعی از نسخه های JDK و کتابخانه های شخص ثالث برای تسهیل نگهداری برنامه ها و کتابخانه هایی که به بهار بستگی دارند ، پشتیبانی می کند.

در مورد طراحی API مراقبت کنید. تیم بهار فکر و زمان زیادی را برای ساخت API هایی که شهودی هستند و در بسیاری از نسخه ها و سالهای زیادی حفظ می کنند ، می گذارد.

استانداردهای بالایی را برای کیفیت کد تنظیم کنید. چارچوب بهار تأکید جدی بر Javadoc معنی دار ، فعلی و دقیق دارد. این یکی از پروژه های بسیار معدودی است که می تواند ساختار کد تمیز و بدون وابستگی دایره ای بین بسته ها را ادعا کند.

## 4. Feedback and Contributions

برای سؤالات چگونه یا مشکلات تشخیص یا اشکال زدایی ، پیشنهاد می کنیم از StackOverflow استفاده کنید ، و یک صفحه سؤالی داریم که برچسب های پیشنهادی را برای استفاده در لیست قرار می دهد. اگر کاملاً مطمئن هستید که در چارچوب بهار مشکلی وجود دارد یا می خواهید یک ویژگی را پیشنهاد دهید ، لطفاً از ردیاب شماره JIRA استفاده کنید.

اگر راه حل در ذهن دارید یا یک مشکل پیشنهادی دارید ، می توانید درخواست کشش را در GitHub ارسال کنید. با این حال ، لطفاً بخاطر داشته باشید که ، برای همه اما بی اهمیت ترین مسائل ، ما انتظار داریم بلیط در ردیاب موضوع ثبت شود ، جایی که بحث ها در آن صورت می گیرد و سابقه ای را برای مرجع بعدی باقی می گذارد.

برای اطلاعات بیشتر به دستورالعمل های صفحه پروژه کمک کننده و سطح بالا مراجعه کنید.

## 5. Getting Started

اگر به تازگی با Spring شروع کرده اید، ممکن است بخواهید با ایجاد یک برنامه مبتنی بر Spring Boot استفاده از Spring Framework را شروع کنید. Spring Boot یک راه سریع (و با نظر) برای ایجاد یک برنامه کاربردی مبتنی بر Spring آماده تولید ارائه می دهد. این بر اساس چارچوب Spring است، قرارداد را به پیکربندی ترجیح می دهد، و طراحی شده است تا شما را در سریع ترین زمان ممکن راه اندازی کند.

می‌توانید از start.spring.io برای تولید یک پروژه اساسی استفاده کنید یا یکی از راهنماهای «شروع به کار» را دنبال کنید، مانند شروع به ساخت یک وب سرویس RESTful. این راهنماها علاوه بر اینکه هضم آسان‌تری دارند، بر روی کار متمرکز هستند و اکثر آنها بر اساس Spring Boot هستند. آنها همچنین پروژه های دیگری از مجموعه بهار را پوشش می دهند که ممکن است بخواهید هنگام حل یک مشکل خاص در نظر بگیرید.

# Core Technologies

این بخش از مستندات مرجع، تمام آن فناوری‌هایی را پوشش می‌دهد که کاملاً در چارچوب Spring Framework هستند.

مهمترین آنها کانتینر Spring Framework's Inversion of Control (IoC) است. بررسی کامل ظرف IoC Spring Framework با پوشش جامع فناوری های برنامه نویسی جنبه گرا (AOP) Spring دنبال می شود. Spring Framework چارچوب AOP خود را دارد که از نظر مفهومی قابل درک است و با موفقیت 80% نیازهای AOP را در برنامه نویسی سازمانی جاوا برطرف می کند.

پوشش ادغام Spring با AspectJ (در حال حاضر غنی ترین - از نظر ویژگی ها - و مطمئناً بالغ ترین پیاده سازی AOP در فضای سازمانی جاوا) نیز ارائه شده است.

## 1. The IoC container

### 1.1. Introduction to the Spring IoC container and beans

این فصل اجرای چارچوب Spring از اصل وارونگی کنترل (IoC) [1] را پوشش می دهد. IoC همچنین به عنوان تزریق وابستگی (DI) شناخته می شود. این فرآیندی است که طی آن اشیاء وابستگی های خود را تعریف می کنند، یعنی اشیایی که با آنها کار می کنند، فقط از طریق آرگومان های سازنده، آرگومان های یک متد کارخانه، یا ویژگی هایی که روی نمونه شی پس از ساخته شدن یا بازگرداندن آن از یک متد کارخانه ای تنظیم می شوند. . سپس ظرف زمانی که لوبیا را ایجاد می کند آن وابستگی ها را تزریق می کند. این فرآیند اساساً معکوس است، از این رو Inversion of Control (IoC) نامیده می شود، که خود bean با استفاده از ساخت مستقیم کلاس ها یا مکانیزمی مانند الگوی Service Locator، نمونه یا مکان وابستگی های خود را کنترل می کند.

بسته‌های org.springframework.beans و org.springframework.context مبنایی برای کانتینر IoC Spring Framework هستند. رابط BeanFactory مکانیزم پیکربندی پیشرفته ای را ارائه می دهد که قادر به مدیریت هر نوع شی است. ApplicationContext یک رابط فرعی BeanFactory است. ادغام آسان تر با ویژگی های AOP Spring را اضافه می کند. مدیریت منابع پیام (برای استفاده در بین المللی)، انتشار رویداد. و زمینه های خاص لایه برنامه مانند WebApplicationContext برای استفاده در برنامه های کاربردی وب.

به طور خلاصه، BeanFactory چارچوب پیکربندی و عملکردهای اساسی را فراهم می کند، و ApplicationContext عملکردهای ویژه سازمانی بیشتری را اضافه می کند. ApplicationContext یک ابر مجموعه کامل از BeanFactory است و به طور انحصاری در این فصل در توضیحات کانتینر IoC Spring استفاده می شود. برای اطلاعات بیشتر در مورد استفاده از BeanFactory به جای ApplicationContext، به The BeanFactory مراجعه کنید.

در Spring، اشیایی که ستون فقرات برنامه شما را تشکیل می دهند و توسط کانتینر Spring IoC مدیریت می شوند، beans نامیده می شوند. یک لوبیا شیئی است که توسط یک کانتینر Spring IoC نمونه سازی، مونتاژ و مدیریت می شود. در غیر این صورت، یک لوبیا به سادگی یکی از بسیاری از اشیاء در برنامه شما است. لوبیاها و وابستگی‌های موجود در میان آن‌ها، در فراداده‌های پیکربندی استفاده‌شده توسط یک ظرف منعکس می‌شوند.

### 1.2. Container overview

رابط org.springframework.context.ApplicationContext نشان دهنده ظرف Spring IoC است و مسئول نمونه سازی، پیکربندی و مونتاژ دانه های فوق الذکر است. ظرف با خواندن فراداده های پیکربندی دستورالعمل های خود را در مورد مواردی که باید نمونه سازی، پیکربندی و مونتاژ شود، دریافت می کند. فراداده پیکربندی در XML، حاشیه نویسی جاوا یا کد جاوا نشان داده می شود. این به شما امکان می دهد اشیایی که برنامه شما را تشکیل می دهند و وابستگی های متقابل غنی بین چنین اشیایی را بیان کنید.

چندین پیاده سازی از رابط ApplicationContext به صورت خارج از جعبه با Spring عرضه می شود. در برنامه های مستقل ایجاد یک نمونه از ClassPathXmlApplicationContext یا FileSystemXmlApplicationContext معمول است. در حالی که XML فرمت سنتی برای تعریف فراداده پیکربندی بوده است، می‌توانید با ارائه مقدار کمی از پیکربندی XML برای فعال کردن پشتیبانی از این فرمت‌های فراداده اضافی، به ظرف دستور دهید که از حاشیه‌نویسی یا کد جاوا به عنوان قالب فراداده استفاده کند.

در اکثر سناریوهای برنامه کاربردی، برای نمونه سازی یک یا چند نمونه از یک ظرف Spring IoC، به کد کاربر صریح نیازی نیست. به عنوان مثال، در یک سناریوی برنامه وب، یک هشت خط (یا بیشتر) ساده از XML توصیفگر وب boilerplate در فایل web.xml برنامه معمولاً کافی است (نمونه سازی مناسب ApplicationContext برای برنامه های کاربردی وب را ببینید). اگر از محیط توسعه مبتنی بر Spring Tool Suite Eclipse استفاده می کنید، این پیکربندی دیگ بخار را می توان به راحتی با چند کلیک ماوس یا فشار دادن کلید ایجاد کرد.

نمودار زیر نمای سطح بالایی از نحوه عملکرد Spring است. کلاس های برنامه شما با ابرداده های پیکربندی ترکیب می شوند تا پس از ایجاد و تنظیم اولیه ApplicationContext، یک سیستم یا برنامه کاملاً پیکربندی شده و قابل اجرا داشته باشید.



#### 1.2.1. Configuration metadata

همانطور که نمودار قبل نشان می دهد، محفظه Spring IoC شکلی از فراداده پیکربندی را مصرف می کند. این فراداده پیکربندی نشان می‌دهد که چگونه شما به‌عنوان یک توسعه‌دهنده برنامه به کانتینر Spring می‌گویید که اشیاء را در برنامه شما نمونه‌سازی، پیکربندی و مونتاژ کند.

فراداده پیکربندی به طور سنتی در قالب XML ساده و شهودی ارائه می‌شود، که بیشتر این فصل از آن برای انتقال مفاهیم و ویژگی‌های کلیدی کانتینر Spring IoC استفاده می‌کند.

فراداده مبتنی بر XML تنها فرم مجاز فراداده پیکربندی نیست. خود محفظه Spring IoC کاملاً از قالبی که این فراداده پیکربندی واقعاً در آن نوشته شده است جدا شده است. این روزها بسیاری از توسعه دهندگان پیکربندی مبتنی بر جاوا را برای برنامه های Spring خود انتخاب می کنند.

فراداده مبتنی بر XML تنها فرم مجاز فراداده پیکربندی نیست. خود محفظه Spring IoC کاملاً از قالبی که این فراداده پیکربندی واقعاً در آن نوشته شده است جدا شده است. این روزها بسیاری از توسعه دهندگان پیکربندی مبتنی بر جاوا را برای برنامه های Spring خود انتخاب می کنند.

برای کسب اطلاعات در مورد استفاده از سایر اشکال فراداده با ظرف Spring، نگاه کنید به:

پیکربندی مبتنی بر حاشیه نویسی: Spring 2.5 پشتیبانی از ابرداده پیکربندی مبتنی بر حاشیه نویسی را معرفی کرد.

Java bean config: با شروع Spring 3.0، بسیاری از ویژگی های ارائه شده توسط پروژه Spring JavaConfig بخشی از چارچوب اصلی Spring Framework شدند. بنابراین شما می توانید با استفاده از جاوا به جای فایل های XML، beans را برای کلاس های برنامه خود تعریف کنید. برای استفاده از این ویژگی‌های جدید، به حاشیه‌نویسی‌های @Configuration، @Bean، @Import و @DependsOn مراجعه کنید.

پیکربندی فنری شامل حداقل یک تعریف و معمولاً بیش از یک تعریف است که ظرف باید آن را مدیریت کند. فراداده پیکربندی مبتنی بر XML این دانه‌ها را به‌عنوان عناصر <bean/> در یک عنصر سطح بالای <beans/> پیکربندی می‌کند. پیکربندی جاوا معمولاً از روش‌های حاشیه‌نویسی @Bean در یک کلاس @Configuration استفاده می‌کند.

این تعاریف bean با اشیاء واقعی که برنامه شما را تشکیل می دهند مطابقت دارد. معمولاً اشیاء لایه سرویس، اشیاء دسترسی به داده (DAO)، اشیاء ارائه مانند نمونه‌های Struts Action، اشیاء زیرساخت مانند Hibernate SessionFactories، صف‌های JMS و غیره را تعریف می‌کنید. به طور معمول، یکی اشیاء دامنه ریز را در کانتینر پیکربندی نمی کند، زیرا معمولاً مسئولیت DAO ها و منطق تجاری ایجاد و بارگذاری اشیاء دامنه است. با این حال، می توانید از ادغام Spring با AspectJ برای پیکربندی اشیایی که خارج از کنترل یک ظرف IoC ایجاد شده اند استفاده کنید. استفاده از AspectJ برای وابستگی تزریق اشیاء دامنه با Spring را ببینید.

مثال زیر ساختار اصلی فراداده های پیکربندی مبتنی بر XML را نشان می دهد:

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans

https://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="..." class="...">

*<!-- collaborators and configuration for this bean go here -->*

</bean>

<bean id="..." class="...">

*<!-- collaborators and configuration for this bean go here -->*

</bean>

*<!-- more bean definitions go here -->*

</beans>

خصیصه id رشته ای است که از آن برای شناسایی تعریف منحصر به فرد bean استفاده می کنید. ویژگی class نوع bean را تعریف می کند و از نام کلاس کاملاً واجد شرایط استفاده می کند. مقدار مشخصه id به اشیاء همکار اشاره دارد. XML برای ارجاع به اشیاء همکار در این مثال نشان داده نشده است. برای اطلاعات بیشتر به Dependencies مراجعه کنید.

#### 1.2.2. Instantiating a container

نمونه سازی یک کانتینر Spring IoC ساده است. مسیر یا مسیرهای ارائه‌شده به سازنده ApplicationContext در واقع رشته‌های منبعی هستند که به ظرف اجازه می‌دهند ابرداده‌های پیکربندی را از انواع منابع خارجی مانند سیستم فایل محلی، از Java CLASSPATH و غیره بارگیری کند.

ApplicationContext context = **new** ClassPathXmlApplicationContext("services.xml", "daos.xml");