|  |  |  |
| --- | --- | --- |
| **1. Introduction to Spring Framework** | | |
|  |  |  |

**Part I. Overview of Spring Framework**

# Introduction to Spring Framework

Spring Framework یک پلتفرم جاوا است که پشتیبانی زیرساختی جامعی را برای توسعه برنامه های جاوا فراهم می کند. Spring زیرساخت را کنترل می کند تا بتوانید روی برنامه خود تمرکز کنید.

Spring شما را قادر می سازد تا برنامه هایی را از "اشیاء ساده قدیمی جاوا" (POJOs) بسازید و خدمات سازمانی را به صورت غیر تهاجمی برای POJO ها اعمال کنید. این قابلیت برای مدل برنامه نویسی Java SE و برای Java EE کامل و جزئی اعمال می شود.

نمونه هایی از نحوه استفاده شما به عنوان یک توسعه دهنده اپلیکیشن از مزیت پلتفرم Spring:

یک روش جاوا را در یک تراکنش پایگاه داده بدون نیاز به پرداختن به API های تراکنش اجرا کنید.

یک روش جاوا محلی را بدون نیاز به پرداختن به APIهای راه دور، به یک روش راه دور تبدیل کنید.

یک روش محلی جاوا را بدون نیاز به مقابله با APIهای JMX به یک عملیات مدیریت تبدیل کنید.

یک روش جاوا محلی را بدون نیاز به سر و کار داشتن با JMS API به عنوان یک کنترل کننده پیام تبدیل کنید.

## 1.1 Dependency Injection and Inversion of Control

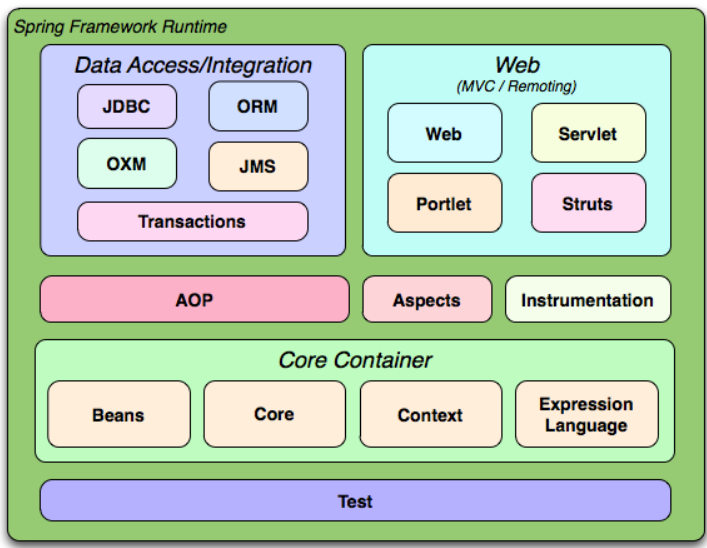
برنامه‌های کاربردی جاوا - یک اصطلاح آزاد که وسعت را از اپلت‌های محدود به برنامه‌های کاربردی سازمانی سمت سرور n-سطح اجرا می‌کند - معمولاً از اشیایی تشکیل می‌شوند که برای تشکیل برنامه مناسب با یکدیگر همکاری می‌کنند. بنابراین اشیا در یک برنامه به یکدیگر وابستگی دارند.

اگرچه پلتفرم جاوا تعداد زیادی از قابلیت‌های توسعه برنامه‌ها را فراهم می‌کند، اما فاقد ابزاری برای سازماندهی بلوک‌های ساختمانی اساسی در یک کل منسجم است و این وظیفه را به معماران و توسعه‌دهندگان واگذار می‌کند. درست است، می‌توانید از الگوهای طراحی مانند Factory، Abstract Factory، Builder، Decorator و Service Locator برای ترکیب کلاس‌ها و نمونه‌های شی مختلف که یک برنامه کاربردی را تشکیل می‌دهند، استفاده کنید. با این حال، این الگوها به سادگی عبارتند از: بهترین شیوه‌هایی که یک نام داده می‌شود، با توضیحی در مورد اینکه الگو چه کاری انجام می‌دهد، کجا باید آن را اعمال کرد، مشکلاتی که به آن پرداخته می‌شود و غیره. الگوها بهترین شیوه های رسمی هستند که باید خودتان در برنامه خود پیاده سازی کنید.

جزء Spring Framework Inversion of Control (IoC) این نگرانی را با ارائه یک ابزار رسمی برای ترکیب اجزای متفاوت در یک برنامه کاربردی کاملاً کارآمد و آماده برای استفاده برطرف می کند. Spring Framework الگوهای طراحی رسمی شده را به عنوان اشیاء درجه یک کدگذاری می کند که می توانید آنها را در برنامه(های) خود ادغام کنید. سازمان‌ها و مؤسسات متعددی از Spring Framework به این روش برای مهندسی برنامه‌های کاربردی قوی و قابل نگهداری استفاده می‌کنند.

## Modules

Spring Framework از ویژگی هایی تشکیل شده است که در حدود 20 ماژول سازماندهی شده اند. این ماژول ها همانطور که در نمودار زیر نشان داده شده است در Core Container، Data Access/Integration، Web، AOP (Aspect Oriented Programming)، Instrumentation و Test گروه بندی می شوند.



## Core Container

Core Container از ماژول های Core، Beans، Context و Expression Language تشکیل شده است.

ماژول‌های Core و Beans بخش‌های اساسی چارچوب، از جمله ویژگی‌های IoC و Dependency Injection را ارائه می‌کنند. BeanFactory یک پیاده سازی پیچیده از الگوی کارخانه است. نیاز به singelton برنامه ای را از بین می برد و به شما امکان می دهد پیکربندی و مشخصات وابستگی ها را از منطق برنامه واقعی خود جدا کنید.

ماژول Context بر پایه پایه محکم ارائه شده توسط ماژول های Core و Beans ساخته شده است: این ابزاری است برای دسترسی به اشیاء به شیوه ای چارچوبی که شبیه به یک رجیستری JNDI است. ماژول Context ویژگی‌های خود را از ماژول Beans به ارث می‌برد و پشتیبانی برای بین‌المللی‌سازی (برای مثال با استفاده از بسته‌های منابع)، انتشار رویداد، بارگذاری منابع، و ایجاد شفاف زمینه‌ها توسط، برای مثال، یک کانتینر servlet اضافه می‌کند. ماژول Context همچنین از ویژگی های Java EE مانند EJB، JMX و کنترل از راه دور اولیه پشتیبانی می کند. رابط ApplicationContext نقطه کانونی ماژول Context است.

ماژول Expression Language یک زبان عبارت قدرتمند برای پرس و جو و دستکاری یک نمودار شی در زمان اجرا فراهم می کند. این یک فرمت از زبان بیان یکپارچه (یکپارچه EL) است که در مشخصات JSP 2.1 مشخص شده است. این زبان از تنظیم و دریافت مقادیر ویژگی، تخصیص ویژگی، فراخوانی روش، دسترسی به زمینه آرایه ها، مجموعه ها و نمایه سازها، عملگرهای منطقی و حسابی، متغیرهای نامگذاری شده و بازیابی اشیاء با نام از ظرف IoC Spring پشتیبانی می کند. همچنین از طرح ریزی و انتخاب لیست و همچنین تجمیع لیست رایج پشتیبانی می کند.

#### Data Access/Integration

لایه دسترسی/ادغام داده ها از ماژول های JDBC، ORM، OXM، JMS و Transaction تشکیل شده است.

ماژول JDBC یک لایه انتزاعی JDBC را فراهم می کند که نیاز به کدگذاری خسته کننده JDBC و تجزیه کدهای خطای خاص فروشنده پایگاه داده را برطرف می کند.

ماژول ORM لایه‌های یکپارچه‌سازی را برای APIهای نگاشت شی رابطه‌ای محبوب از جمله JPA، JDO، Hibernate و iBatis فراهم می‌کند. با استفاده از بسته ORM می‌توانید از همه این چارچوب‌های O/R-mapping در ترکیب با سایر ویژگی‌هایی که Spring ارائه می‌دهد استفاده کنید، مانند ویژگی ساده مدیریت تراکنش اعلامی که قبلاً ذکر شد.

ماژول OXM یک لایه انتزاعی ارائه می‌کند که از پیاده‌سازی نگاشت Object/XML برای JAXB، Castor، XMLBeans، JiBX و XStream پشتیبانی می‌کند.

ماژول سرویس پیام رسانی جاوا (JMS) دارای ویژگی هایی برای تولید و مصرف پیام است.

ماژول Transaction از مدیریت تراکنش های برنامه ای و اعلامی برای کلاس هایی که رابط های ویژه را پیاده سازی می کنند و برای همه POJO های شما (اشیاء ساده قدیمی جاوا) پشتیبانی می کند.

#### Web

لایه وب از ماژول های Web، Web-Servlet، Web-Struts و Web-Portlet تشکیل شده است.

ماژول وب اسپرینگ ویژگی‌های یکپارچه‌سازی مبتنی بر وب را فراهم می‌کند، مانند قابلیت آپلود فایل چند قسمتی و مقداردهی اولیه ظرف IoC با استفاده از شنونده‌های سرولت و زمینه برنامه کاربردی وب‌گرا. همچنین شامل بخش های مربوط به وب پشتیبانی از راه دور Spring است.

ماژول Web-Servlet شامل پیاده سازی Spring's model-view-controller (MVC) برای برنامه های کاربردی وب است. چارچوب MVC Spring یک جداسازی تمیز بین کد مدل دامنه و فرم‌های وب فراهم می‌کند و با سایر ویژگی‌های Spring Framework یکپارچه می‌شود.

ماژول Web-Struts شامل کلاس های پشتیبانی برای ادغام یک لایه وب کلاسیک Struts در یک برنامه Spring است. توجه داشته باشید که این پشتیبانی اکنون از Spring 3.0 منسوخ شده است. مهاجرت برنامه خود را به Struts 2.0 و ادغام Spring آن یا به یک راه حل Spring MVC در نظر بگیرید.

ماژول Web-Portlet پیاده سازی MVC را برای استفاده در محیط پورتلت فراهم می کند و عملکرد ماژول Web-Servlet را منعکس می کند.

#### AOP and Instrumentation

ماژول AOP Spring یک پیاده‌سازی برنامه‌نویسی جنبه‌محور منطبق با AOP Alliance را ارائه می‌کند که به شما امکان می‌دهد، برای مثال، متد-رهگیرها و نقاط برش را برای جدا کردن کدهایی که عملکردهایی را که باید از هم جدا شوند، تعریف کنید. با استفاده از عملکرد metaData سطح منبع، می‌توانید اطلاعات رفتاری را به روشی مشابه ویژگی‌های .NET در کد خود بگنجانید.

ماژول جداگانه Aspects یکپارچه سازی با AspectJ را فراهم می کند.

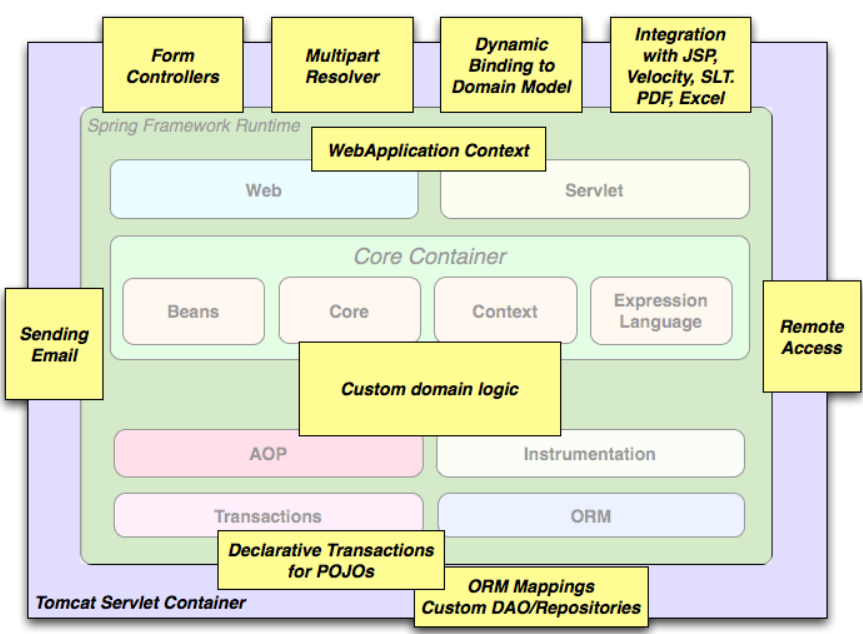
ماژول Instrumentation پشتیبانی از ابزار دقیق کلاس و پیاده سازی کلاس لودر را برای استفاده در سرورهای برنامه خاص فراهم می کند

#### Test

ماژول Test از تست اجزای Spring با JUnit یا TestNG پشتیبانی می کند. بارگیری مداوم Spring ApplicationContexts و ذخیره سازی آن conext ها را فراهم می کند. همچنین اشیاء ساختگی را ارائه می دهد که می توانید از آنها برای آزمایش کد خود به صورت مجزا استفاده کنید.

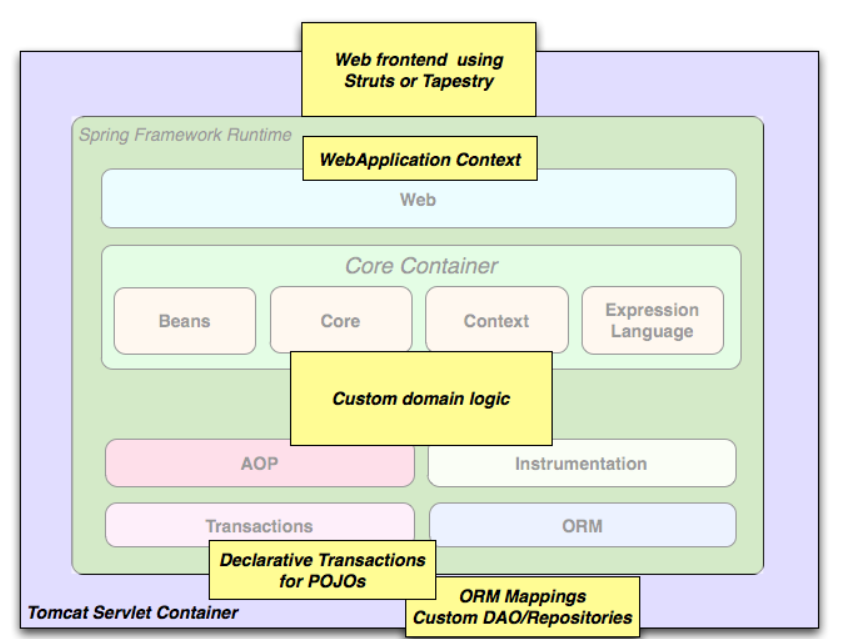
## Usage scenarios

بلوک‌های ساختمانی که قبلاً توضیح داده شد، Spring را در بسیاری از سناریوها، از اپلت‌ها گرفته تا برنامه‌های کاربردی تمام عیار سازمانی که از عملکرد مدیریت تراکنش Spring و یکپارچه‌سازی چارچوب وب استفاده می‌کنند، به یک انتخاب منطقی تبدیل می‌کند.



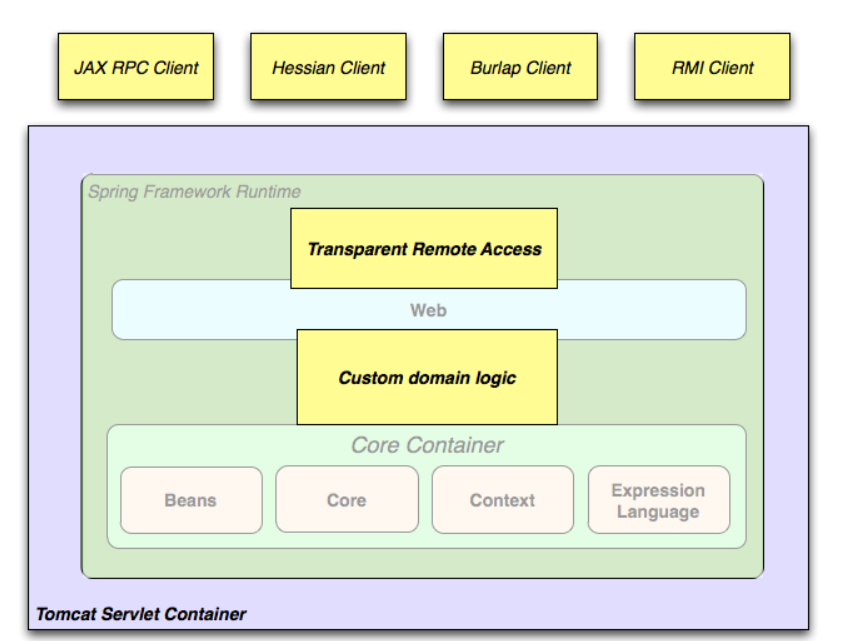
Typical full-fledged Spring web application

ویژگی‌های مدیریت تراکنش‌های declarative Spring، برنامه وب را کاملاً تراکنشی می‌کند، درست همانطور که اگر از تراکنش‌های مدیریت شده توسط EJB استفاده می‌کردید. تمام منطق کسب و کار سفارشی شما را می توان با POJO های ساده پیاده سازی کرد و توسط کانتینر Spring's IoC مدیریت شد. خدمات اضافی شامل پشتیبانی از ارسال ایمیل و تأیید اعتبار مستقل از لایه وب است که به شما امکان می دهد مکان اجرای قوانین اعتبار سنجی را انتخاب کنید. پشتیبانی ORM Spring با JPA، Hibernate، JDO و iBatis یکپارچه شده است. به عنوان مثال، هنگام استفاده از Hibernate، می توانید به استفاده از فایل های نقشه برداری موجود و پیکربندی استاندارد Hibernate SessionFactory ادامه دهید. کنترل‌کننده‌های فرم به‌طور یکپارچه لایه وب را با مدل دامنه ادغام می‌کنند و نیاز به ActionForms یا کلاس‌های دیگر را که پارامترهای HTTP را به مقادیر مدل دامنه شما تبدیل می‌کنند، از بین می‌برند.



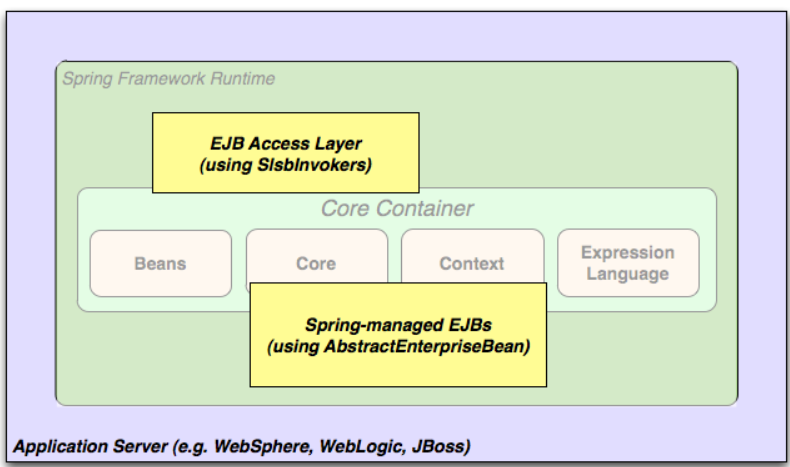
Spring middle-tier using a third-party web framework

گاهی اوقات شرایط به شما اجازه نمی دهد که به طور کامل به یک چارچوب متفاوت تغییر دهید. چارچوب Spring شما را مجبور به استفاده از همه چیز در آن نمی کند. این یک راه حل همه یا هیچ نیست. فریم‌اندهای موجود که با WebWork، Struts، Tapestry یا سایر چارچوب‌های رابط کاربری ساخته شده‌اند، می‌توانند با یک لایه میانی مبتنی بر Spring ادغام شوند، که به شما امکان می‌دهد از ویژگی‌های تراکنش Spring استفاده کنید. شما به سادگی باید منطق کسب و کار خود را با استفاده از ApplicationContext متصل کنید و از WebApplicationContext برای ادغام لایه وب خود استفاده کنید.



Remoting usage scenario

هنگامی که نیاز به دسترسی به کدهای موجود از طریق سرویس های وب دارید، می توانید از کلاس های Hessian-، Burlap-، Rmi- یا JaxRpcProxyFactory Spring's استفاده کنید. فعال کردن دسترسی از راه دور به برنامه های موجود دشوار نیست.



EJBs - Wrapping existing POJOs

Spring Framework همچنین یک لایه دسترسی و انتزاع برای Enterprise JavaBeans فراهم می کند که به شما امکان می دهد از POJO های موجود خود استفاده مجدد کنید و آنها را در Session Beans برای استفاده در برنامه های کاربردی وب مقیاس پذیر و ایمن که ممکن است به امنیت declarative نیاز داشته باشند، بپیچید.

### Dependency Management and Naming Conventions

مدیریت وابستگی و تزریق وابستگی چیزهای متفاوتی هستند. برای دریافت آن ویژگی های خوب Spring در برنامه خود (مانند تزریق وابستگی) باید تمام کتابخانه های مورد نیاز (فایل های jar) را جمع آوری کنید و آنها را در زمان اجرا و احتمالاً در زمان کامپایل در مسیر کلاس خود قرار دهید. این وابستگی ها اجزای مجازی نیستند که تزریق می شوند، بلکه منابع فیزیکی در یک سیستم فایل (معمولا) هستند. فرآیند مدیریت وابستگی شامل مکان یابی آن منابع، ذخیره آنها و افزودن آنها به مسیرهای کلاس است. وابستگی ها می توانند مستقیم باشند (مثلاً برنامه من در زمان اجرا به Spring بستگی دارد) یا غیر مستقیم (مثلاً برنامه من به commons-dbcp بستگی دارد که به commons-pool بستگی دارد). وابستگی‌های غیرمستقیم به عنوان «گذرا» نیز شناخته می‌شوند و این وابستگی‌ها هستند که شناسایی و مدیریت آن‌ها سخت‌تر است.

اگر می‌خواهید از Spring استفاده کنید، باید یک کپی از کتابخانه‌های jar که شامل تکه‌های Spring مورد نیاز شما هستند، تهیه کنید. برای آسان‌تر کردن این کار، Spring به‌عنوان مجموعه‌ای از ماژول‌ها بسته‌بندی می‌شود که وابستگی‌ها را تا حد امکان از هم جدا می‌کنند، بنابراین برای مثال اگر نمی‌خواهید یک برنامه وب بنویسید، به ماژول‌های Spring-Web نیازی ندارید. برای رجوع به ماژول های کتابخانه Spring در این راهنما، از یک قرارداد نامگذاری مختصر Spring-\* یا Spring-\*.jar استفاده می کنیم، که در آن "\*" نشان دهنده نام کوتاه ماژول است (به عنوان مثال Spring-core، Spring-webmvc، Spring-jms، و غیره.). نام واقعی فایل jar که استفاده می‌کنید ممکن است به این شکل باشد (به زیر مراجعه کنید) یا ممکن است نباشد، و معمولاً یک شماره نسخه در نام فایل نیز دارد (به عنوان مثال Spring-core-3.0.0.RELEASE.jar).

به طور کلی، بهار مصنوعات خود را در چهار مکان مختلف منتشر می کند:

در سایت دانلود انجمن http://www.springsource.org/download/community. در اینجا تمام jar‌های Spring را در یک فایل فشرده برای دانلود آسان پیدا می‌کنید. نام jars در اینجا از نسخه 3.0 به شکل org.springframework.\*-<version>.jar است.

Maven Central، که مخزن پیش فرضی است که Maven پرس و جو می کند و برای استفاده نیازی به پیکربندی خاصی ندارد. بسیاری از کتابخانه های رایج که Spring به آنها وابسته است نیز از Maven Central در دسترس هستند و بخش بزرگی از جامعه Spring از Maven برای مدیریت وابستگی استفاده می کنند، بنابراین این برای آنها راحت است. نام jar ها در اینجا به صورت Spring-\*-<version>.jar و Maven groupId org.springframework است.

Enterprise Bundle Repository (EBR) که توسط SpringSource اجرا می شود و همچنین میزبان تمام کتابخانه هایی است که با Spring یکپارچه می شوند. مخازن Maven و Ivy هر دو در اینجا برای تمام JAR‌های Spring و وابستگی‌های آن‌ها در دسترس هستند، به علاوه تعداد زیادی از کتابخانه‌های رایج دیگر که مردم در برنامه‌های کاربردی با Spring از آنها استفاده می‌کنند. هر دو نسخه کامل و همچنین نقاط عطف و تصاویر فوری توسعه در اینجا مستقر شده اند. نام فایل‌های jar به همان شکل دانلود انجمن است (org.springframework.\*-<version>.jar)، و وابستگی‌ها نیز به این شکل طولانی هستند، با کتابخانه‌های خارجی (نه از SpringSource) پیشوند com.springsource. برای اطلاعات بیشتر به سوالات متداول مراجعه کنید. در یک مخزن عمومی Maven که در آمازون S3 میزبانی می‌شود، برای عکس‌های فوری توسعه و انتشارات نقطه عطف (یک کپی از نسخه‌های نهایی نیز در اینجا نگهداری می‌شود). نام فایل‌های jar به شکل Maven Central هستند، بنابراین این مکان مفیدی برای دریافت نسخه‌های توسعه‌دهنده Spring برای استفاده با سایر کتابخانه‌های مستقر در Maven Central است.

بنابراین اولین چیزی که باید تصمیم بگیرید این است که چگونه وابستگی های خود را مدیریت کنید: اکثر مردم از یک سیستم خودکار مانند Maven یا Ivy استفاده می کنند، اما شما می توانید این کار را به صورت دستی با دانلود همه JAR ها انجام دهید. هنگام به دست آوردن اسپرینگ با Maven یا Ivy باید تصمیم بگیرید که از کدام مکان آن را تهیه کنید. به طور کلی، اگر به OSGi اهمیت می دهید، از EBR استفاده کنید، زیرا مصنوعات سازگار با OSGi را برای همه وابستگی های Spring مانند Hibernate و Freemarker در خود جای داده است. اگر OSGi برای شما اهمیتی ندارد، هر کدام از این مکان‌ها کار می‌کنند، هرچند مزایا و معایبی بین آنها وجود دارد. به طور کلی، یک مکان یا مکان دیگر را برای پروژه خود انتخاب کنید. آنها را مخلوط نکنید این امر به ویژه مهم است زیرا مصنوعات EBR لزوماً از نامگذاری متفاوتی نسبت به مصنوعات Maven Central استفاده می کنند.

**Table 1.1. Comparison of Maven Central and SpringSource EBR Repositories**

| **Feature** | **Maven Central** | **EBR** |
| --- | --- | --- |
| OSGi Compatible | Not explicit | Yes |
| Number of Artifacts | Tens of thousands; all kinds | Hundreds; those that Spring integrates with |
| Consistent Naming Conventions | No | Yes |
| Naming Convention: GroupId | Varies. Newer artifacts often use domain name, e.g. org.slf4j. Older ones often just use the artifact name, e.g. log4j. | Domain name of origin or main package root,  e.g. org.springframework |
| Naming Convention: ArtifactId | Varies. Generally the project or module name, using a hyphen "-" separator, e.g. spring-core, logj4. | Bundle Symbolic Name, derived from the main package root,  e.g. org.springframework.beans. If the jar had to be patched to  ensure OSGi compliance then com.springsource is appended,  e.g. com.springsource.org.apache.log4j |
| Naming Convention: Version | Varies. Many new artifacts use m.m.m or m.m.m.X (with m=digit, X=text). Older ones use m.m. Some neither. Ordering is defined but not often relied on, so not strictly reliable. | OSGi version number m.m.m.X, e.g. 3.0.0.RC3. The  text qualifier imposes alphabetic ordering on versions with  the same numeric values. |
| Publishing | Usually automatic via rsync or source control updates. Project authors can upload individual jars to JIRA. | Manual (JIRA processed by SpringSource) |
| Quality Assurance | By policy. Accuracy is responsibility of authors. | Extensive for OSGi manifest, Maven POM and Ivy metadata.  QA performed by Spring team. |
| Hosting | Contegix. Funded by Sonatype with several mirrors. | S3 funded by SpringSource. |
| Search Utilities | Various | <http://www.springsource.com/repository> |
| Integration with SpringSource Tools | Integration through STS with Maven dependency management | Extensive integration through STS with Maven, Roo,  CloudFoundry |

#### Spring Dependencies and Depending on Spring

اگرچه Spring یکپارچه سازی و پشتیبانی را برای طیف عظیمی از سازمانی و سایر ابزارهای خارجی فراهم می کند، اما عمداً وابستگی های اجباری خود را به حداقل می رساند: برای اینکه بتوانید تعداد زیادی کتابخانه jar را پیدا و دانلود کنید (حتی به طور خودکار) لازم نیست. از Spring برای موارد استفاده ساده استفاده کنید. برای تزریق وابستگی اولیه تنها یک وابستگی خارجی اجباری وجود دارد، و آن برای logging است (برای توضیحات بیشتر در مورد گزینه‌های logging به زیر مراجعه کنید).

در ادامه مراحل اولیه مورد نیاز برای پیکربندی برنامه‌ای که به Spring بستگی دارد، ابتدا با Maven و سپس با Ivy شرح می‌دهیم. در همه موارد، اگر چیزی نامشخص است، به مستندات سیستم مدیریت وابستگی خود مراجعه کنید، یا به برخی از کدهای نمونه نگاه کنید - اسپرینگ خودش از Ivy برای مدیریت وابستگی ها هنگام ساخت استفاده می کند و نمونه های ما بیشتر از Maven استفاده می کنند.

#### Maven Dependency Management

اگر از Maven برای مدیریت وابستگی استفاده می کنید، حتی نیازی به ارائه وابستگی logging ندارید. به عنوان مثال، برای ایجاد یک زمینه برنامه و استفاده از تزریق وابستگی برای پیکربندی یک برنامه، وابستگی های Maven شما به شکل زیر خواهد بود:

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>3.0.0.RELEASE</version>

<scope>runtime</scope>

</dependency>

</dependencies>

خودشه. توجه داشته باشید که اگر نیازی به کامپایل در برابر Spring API ندارید، می‌توان دامنه را به‌عنوان زمان اجرا اعلام کرد، که معمولاً برای موارد استفاده از تزریق وابستگی اولیه صادق است.

ما از قراردادهای نامگذاری Maven Central در مثال بالا استفاده کردیم، به طوری که با Maven Central یا مخزن SpringSource S3 Maven کار می کند. برای استفاده از مخزن S3 Maven (به عنوان مثال برای نقاط عطف یا عکس‌های فوری برنامه‌نویس)، باید مکان مخزن را در پیکربندی Maven خود مشخص کنید. برای نسخه های کامل:

<repositories>

<repository>

<id>com.springsource.repository.maven.release</id>

<url>http://repo.springsource.org/release/</url>

<snapshots><enabled>false</enabled></snapshots>

</repository>

</repositories>

برای milestone:

<repositories>

<repository>

<id>com.springsource.repository.maven.milestone</id>

<url>http://repo.springsource.org/milestone/</url>

<snapshots><enabled>false</enabled></snapshots>

</repository>

</repositories>

برای snapshots:

<repositories>

<repository>

<id>com.springsource.repository.maven.snapshot</id>

<url>http://repo.springsource.org/snapshot/</url>

<snapshots><enabled>true</enabled></snapshots>

</repository>

</repositories>

برای استفاده از SpringSource EBR باید از یک قرارداد نامگذاری متفاوت برای وابستگی ها استفاده کنید. حدس زدن نام ها معمولا آسان است، به عنوان مثال. در این مورد این است:

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>org.springframework.context</artifactId>

<version>3.0.0.RELEASE</version>

<scope>runtime</scope>

</dependency>

</dependencies>

همچنین باید مکان مخزن را به صراحت اعلام کنید (فقط URL مهم است):

<repositories>

<repository>

<id>com.springsource.repository.bundles.release</id>

<url>http://repository.springsource.com/maven/bundles/release/</url>

</repository>

</repositories>

اگر وابستگی‌های خود را به صورت دستی مدیریت می‌کنید، URL موجود در اعلامیه مخزن بالا قابل مرور نیست، اما یک رابط کاربری در http://www.springsource.com/repository وجود دارد که می‌توان از آن برای جستجو و دانلود وابستگی‌ها استفاده کرد. همچنین دارای قطعات مفیدی از پیکربندی Maven و Ivy است که می توانید در صورت استفاده از آن ابزارها را کپی و جایگذاری کنید.

#### Ivy Dependency Management

اگر ترجیح می دهید از Ivy برای مدیریت وابستگی ها استفاده کنید، نام ها و گزینه های پیکربندی مشابهی وجود دارد.

برای پیکربندی Ivy برای اشاره به SpringSource EBR حل کننده های زیر را به ivysettings.xml خود اضافه کنید:

<resolvers>

<url name="com.springsource.repository.bundles.release">

<ivy pattern="http://repository.springsource.com/ivy/bundles/release/

[organisation]/[module]/[revision]/[artifact]-[revision].[ext]" />

<artifact pattern="http://repository.springsource.com/ivy/bundles/release/

[organisation]/[module]/[revision]/[artifact]-[revision].[ext]" />

</url>

<url name="com.springsource.repository.bundles.external">

<ivy pattern="http://repository.springsource.com/ivy/bundles/external/

[organisation]/[module]/[revision]/[artifact]-[revision].[ext]" />

<artifact pattern="http://repository.springsource.com/ivy/bundles/external/

[organisation]/[module]/[revision]/[artifact]-[revision].[ext]" />

</url>

</resolvers>

XML بالا معتبر نیست زیرا خطوط بیش از حد طولانی هستند - اگر کپی-پیست کنید، انتهای خطوط اضافی را در وسط الگوهای آدرس اینترنتی حذف کنید.

هنگامی که Ivy پیکربندی شد تا در EBR جستجو کند، افزودن یک وابستگی آسان است. به سادگی صفحه جزئیات بسته مورد نظر را در مرورگر مخزن بالا بکشید و یک قطعه پیچک آماده برای اضافه کردن در بخش وابستگی‌ها خواهید یافت. به عنوان مثال (در ivy.xml):

<dependency org="org.springframework"

name="org.springframework.core" rev="3.0.0.RELEASE" conf="compile->runtime"/>

### Logging

Logging یک وابستگی بسیار مهم برای Spring است زیرا الف) تنها وابستگی خارجی اجباری است، ب) همه دوست دارند برخی از خروجی ها را از ابزارهایی که استفاده می کنند ببینند، و ج) Spring با بسیاری از ابزارهای دیگر ادغام می شود که همه آنها نیز ساخته شده اند. انتخاب وابستگی logging. یکی از اهداف یک توسعه‌دهنده برنامه اغلب این است که گزارش‌گیری یکپارچه را در یک مکان مرکزی برای کل برنامه، از جمله تمام اجزای خارجی، پیکربندی کند. این دشوارتر از آن چیزی است که ممکن است باشد، زیرا انتخاب های زیادی برای logging framework وجود دارد.

وابستگی اجباری logging در بهار، Jakarta Commons Logging API (JCL) است. ما در برابر JCL کامپایل می کنیم و همچنین اشیاء JCL Log را برای کلاس هایی که چارچوب Spring را گسترش می دهند قابل مشاهده می کنیم. برای کاربران مهم است که همه نسخه‌های Spring از یک کتابخانه گزارش‌گیری استفاده کنند: مهاجرت آسان است زیرا سازگاری به عقب حتی با برنامه‌هایی که Spring را گسترش می‌دهند حفظ می‌شود. روشی که ما این کار را انجام می‌دهیم این است که یکی از ماژول‌ها را در Spring به طور صریح به Commons-logging (پیاده‌سازی متعارف JCL) وابسته کنیم و سپس همه ماژول‌های دیگر را در زمان کامپایل به آن وابسته کنیم. اگر برای مثال از Maven استفاده می‌کنید، و نمی‌دانید که وابستگی به logging مشترک را از کجا دریافت کرده‌اید، پس این وابستگی از Spring و به طور خاص از ماژول مرکزی به نام Spring-core است.

نکته خوب در مورد Commons-logging این است که برای کارکرد برنامه خود به هیچ چیز دیگری نیاز ندارید. این یک الگوریتم کشف زمان اجرا دارد که به دنبال سایر چارچوب‌های گزارش‌گیری در مکان‌های شناخته شده در مسیر کلاس می‌گردد و از یکی که فکر می‌کند مناسب است استفاده می‌کند (یا می‌توانید در صورت نیاز به آن بگویید کدام یک). اگر هیچ چیز دیگری در دسترس نباشد، فقط از JDK (java.util.logging یا به اختصار JUL) لاگ‌های زیبا به‌نظر می‌رسد. باید متوجه شوید که برنامه Spring شما در اکثر مواقع به خوبی کار می کند و به راحتی به کنسول وارد می شود و این مهم است.

**Not Using Commons Logging**

متأسفانه، الگوریتم کشف زمان اجرا commons-logging، در حالی که برای کاربر نهایی مناسب است، مشکل ساز است. اگر می‌توانستیم ساعت را به عقب برگردانیم و Spring را اکنون به عنوان یک پروژه جدید شروع کنیم، از وابستگی گزارش متفاوتی استفاده می‌کردیم. اولین انتخاب احتمالاً Simple Logging Facade برای جاوا (SLF4J) است که توسط بسیاری از ابزارهای دیگر که افراد با Spring در داخل برنامه های خود استفاده می کنند نیز استفاده می شود.

خاموش کردن Commons-logging آسان است: فقط مطمئن شوید که در زمان اجرا در مسیر کلاس نباشد. در اصطلاح Maven شما وابستگی را حذف می کنید و به دلیل نحوه اعلان وابستگی های Spring، فقط یک بار باید این کار را انجام دهید.

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>3.0.0.RELEASE</version>

<scope>runtime</scope>

<exclusions>

<exclusion>

<groupId>commons-logging</groupId>

<artifactId>commons-logging</artifactId>

</exclusion>

</exclusions>

</dependency>

</dependencies>

اکنون این برنامه احتمالاً خراب است زیرا هیچ پیاده سازی API JCL در مسیر کلاس وجود ندارد، بنابراین برای رفع آن باید یک برنامه جدید ارائه شود. در بخش بعدی به شما نشان می دهیم که چگونه می توانید یک پیاده سازی جایگزین از JCL با استفاده از SLF4J به عنوان مثال ارائه دهید.

#### Using SLF4J

SLF4J یک وابستگی تمیزتر و در زمان اجرا کارآمدتر از Commons-logging است زیرا از اتصالات زمان کامپایل به جای کشف زمان اجرا سایر چارچوب‌های گزارش‌گیری که ادغام می‌کند استفاده می‌کند. این همچنین به این معنی است که شما باید در مورد آنچه می خواهید در زمان اجرا اتفاق بیفتد واضح تر باشید و آن را اعلام کنید یا بر اساس آن پیکربندی کنید. SLF4J اتصال به بسیاری از چارچوب‌های گزارش‌گیری رایج را فراهم می‌کند، بنابراین معمولاً می‌توانید یکی را که قبلاً استفاده می‌کنید انتخاب کنید و برای پیکربندی و مدیریت به آن متصل شوید.

SLF4J اتصال به بسیاری از چارچوب‌های گزارش‌گیری رایج، از جمله JCL را فراهم می‌کند، و همچنین برعکس را انجام می‌دهد: پل‌هایی بین سایر چارچوب‌های گزارش‌گیری و خودش. بنابراین برای استفاده از SLF4J با Spring، باید وابستگی Commons-logging را با پل SLF4J-JCL جایگزین کنید. هنگامی که این کار را انجام دادید، تماس‌های ثبت‌نام از داخل Spring به تماس‌های ورود به سیستم API SLF4J ترجمه می‌شوند، بنابراین اگر کتابخانه‌های دیگر در برنامه شما از آن API استفاده می‌کنند، شما یک مکان واحد برای پیکربندی و مدیریت logging خواهید داشت.

یک انتخاب رایج ممکن است این باشد که Spring را به SLF4J متصل کنید، و سپس اتصال صریح از SLF4J به Log4J را ارائه دهید. شما باید 4 وابستگی را تامین کنید (و از اشتراک‌گذاری‌های موجود حذف کنید): پل، API SLF4J، اتصال به Log4J و خود پیاده‌سازی Log4J. در Maven شما این کار را به این صورت انجام می دهید

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>3.0.0.RELEASE</version>

<scope>runtime</scope>

<exclusions>

<exclusion>

<groupId>commons-logging</groupId>

<artifactId>commons-logging</artifactId>

</exclusion>

</exclusions>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>jcl-over-slf4j</artifactId>

<version>1.5.8</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.5.8</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>1.5.8</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.14</version>

<scope>runtime</scope>

</dependency>

</dependencies>

این ممکن است وابستگی های زیادی به نظر برسد فقط برای گرفتن گزارش. خوب است، اما اختیاری است، و باید بهتر از ثبت نام وانیلیcommons-logging با توجه به مشکلات کلاس لودر رفتار کند، به ویژه اگر در یک ظرف سخت مانند یک پلت فرم OSGi هستید. ظاهراً یک مزیت عملکرد نیز وجود دارد زیرا اتصالات در زمان کامپایل هستند و نه در زمان اجرا.

انتخاب رایج‌تر در میان کاربران SLF4J، که از مراحل کمتری استفاده می‌کند و وابستگی‌های کمتری ایجاد می‌کند، اتصال مستقیم به Logback است. این مرحله اتصال اضافی را حذف می‌کند زیرا Logback مستقیماً SLF4J را پیاده‌سازی می‌کند، بنابراین شما فقط باید به دو کتابخانه وابسته باشید نه چهار (jcl-over-slf4j و logback). اگر این کار را انجام دهید، ممکن است لازم باشد وابستگی slf4j-api را از سایر وابستگی‌های خارجی (نه Spring) حذف کنید، زیرا فقط یک نسخه از آن API را در مسیر کلاس می‌خواهید.

**Using Log4J**

بسیاری از افراد از Log4j به عنوان یک logging framework برای اهداف پیکربندی و مدیریت استفاده می کنند. این کارآمد و به خوبی تثبیت شده است، و در واقع همان چیزی است که در زمان اجرا هنگام ساخت و آزمایش Spring از آن استفاده می کنیم. Spring همچنین برخی از ابزارهای کاربردی را برای پیکربندی و مقداردهی اولیه Log4j فراهم می کند، بنابراین در برخی ماژول ها وابستگی زمان کامپایل اختیاری به Log4j دارد.

برای اینکه Log4j با وابستگی پیش فرض JCL (commons-logging) کار کند، تنها کاری که باید انجام دهید این است که Log4j را در مسیر کلاس قرار دهید و یک فایل پیکربندی (log4j.properties یا log4j.xml در ریشه classpath) برای آن ارائه دهید. بنابراین برای کاربران Maven این اعلامیه وابستگی شما است

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>3.0.0.RELEASE</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.14</version>

<scope>runtime</scope>

</dependency>

</dependencies>

و در اینجا یک نمونه log4j.properties برای ورود به کنسول وجود دارد:

log4j.rootCategory=INFO, stdout

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%d{ABSOLUTE} %5p %t %c{2}:%L - %m%n

log4j.category.org.springframework.beans.factory=DEBUG

##### Runtime Containers with Native JCL3

بسیاری از مردم برنامه‌های Spring خود را در ظرفی اجرا می‌کنند که خود اجرای JCL را ارائه می‌کند. IBM Websphere Application Server (WAS) کهن الگو است. این اغلب باعث مشکلاتی می شود و متأسفانه هیچ راه حل گلوله نقره ای وجود ندارد. صرفاً کنار گذاشتن commons-loggingاز برنامه شما در اکثر مواقع کافی نیست.

برای روشن شدن این موضوع: مشکلات گزارش شده معمولاً به خودی خود مربوط به JCL یا حتی با ثبت نام مشترک نیستند: بلکه مربوط به اتصال ثبت مشترکات به چارچوب دیگری (اغلب Log4J) است. این ممکن است شکست بخورد زیرا ثبت نام مشترک روشی را که در زمان اجرا بین نسخه‌های قدیمی (1.0) موجود در برخی از کانتینرها و نسخه‌های مدرنی که اکثر مردم اکنون استفاده می‌کنند (1.1) را تغییر داده است. Spring از هیچ بخش غیرمعمولی از JCL API استفاده نمی کند، بنابراین هیچ چیز در آنجا خراب نمی شود، اما به محض اینکه Spring یا برنامه شما سعی می کند هر گونه logging را انجام دهد، می توانید متوجه شوید که اتصالات به Log4J کار نمی کنند.

در چنین مواردی با WAS ساده‌ترین کار این است که سلسله مراتب کلاس بارگذار را وارونه کنید (IBM آن را "آخرین والد" می‌نامد) تا برنامه وابستگی JCL را کنترل کند، نه کانتینر. این گزینه همیشه باز نیست، اما بسیاری از پیشنهادات دیگر در حوزه عمومی برای رویکردهای جایگزین وجود دارد، و مسافت پیموده شده شما ممکن است بسته به نسخه دقیق و مجموعه ویژگی‌های ظرف متفاوت باشد.

# 2. New Features and Enhancements in Spring Framework 3.0

اگر مدتی است که از Spring Framework استفاده می‌کنید، می‌دانید که Spring تحت دو بازنگری اصلی قرار گرفته است: Spring 2.0، که در اکتبر 2006 منتشر شد، و Spring 2.5، که در نوامبر 2007 منتشر شد. اکنون زمان برای سومین بازنگری است. در Spring Framework 3.0.

Java SE و Java EE پشتیبانی می کند

Spring Framework اکنون بر اساس جاوا 5 است و جاوا 6 به طور کامل پشتیبانی می شود.

علاوه بر این، Spring با J2EE 1.4 و Java EE 5 سازگار است، در حالی که در همان زمان برخی از پشتیبانی اولیه را برای Java EE 6 ارائه می کند.

## 2.1 Java 5

کل کد چارچوب برای استفاده از ویژگی‌های جاوا 5 مانند ژنریک، varargs و سایر پیشرفت‌های زبانی اصلاح شده است. ما تمام تلاش خود را انجام داده‌ایم تا همچنان کد را با هم سازگار نگه داریم. ما اکنون استفاده مداوم از مجموعه‌ها و نقشه‌های عمومی، استفاده مداوم از FactoryBeans عمومی، و همچنین وضوح ثابت روش‌های bridge در Spring AOP API داریم. Generic ApplicationListeners به ​​طور خودکار فقط انواع خاصی از رویداد را دریافت می کند. همه واسط های تماس برگشتی مانند TransactionCallback و HibernateCallback اکنون یک مقدار نتیجه عمومی را اعلام می کنند. به طور کلی، پایگاه کد هسته Spring اکنون به تازگی برای جاوا 5 اصلاح و بهینه شده است.

چکیده Spring's TaskExecutor برای ادغام نزدیک با امکانات java.util.concurrent جاوا 5 به روز شده است. ما اکنون پشتیبانی درجه یک برای Calables و Futures و همچنین آداپتورهای ExecutorService، ادغام ThreadFactory و غیره ارائه می‌کنیم. علاوه بر این، ما از طریق استفاده از حاشیه نویسی @Async جدید (یا حاشیه نویسی Asynchronous@ EJB 3.1) از فراخوانی روش ناهمزمان پشتیبانی می کنیم.

## 2.2 Improved documentation

مستندات مرجع Spring همچنین بطور قابل توجهی به روز شده است تا همه تغییرات و ویژگی های جدید Spring Framework 3.0 را منعکس کند. در حالی که تمام تلاش‌ها برای اطمینان از عدم وجود خطایی در این مستندات انجام شده است، اما ممکن است برخی از اشتباهات وارد شده باشد. اگر اشتباهات تایپی یا حتی خطاهای جدی‌تری را مشاهده کردید، و می‌توانید در طول ناهار از چند چرخه صرف نظر کنید، لطفاً آن را به همراه داشته باشید. خطایی که با طرح موضوع مورد توجه تیم بهار قرار گرفت.

## 2.3 New articles and tutorials

مقالات و آموزش های بسیار خوبی وجود دارد که نحوه شروع کار با ویژگی های Spring Framework 3 را نشان می دهد. آنها را در صفحه مستندات بهار بخوانید.

نمونه ها برای بهره مندی از ویژگی های جدید در Spring Framework 3 بهبود یافته و به روز شده اند. علاوه بر این، نمونه ها به خارج از درخت منبع به یک مخزن اختصاصی SVN منتقل شده اند که در آدرس زیر موجود است:

https://anonsvn.springframework.org/svn/spring-samples/

به این ترتیب، نمونه ها دیگر در کنار Spring Framework 3 توزیع نمی شوند و باید جداگانه از مخزن ذکر شده در بالا دانلود شوند. با این حال، این مستندات همچنان به برخی از نمونه ها (به ویژه Petclinic) برای نشان دادن ویژگی های مختلف اشاره می کند.

## 2.4 New module organization and build system

ماژول های فریم ورک بازنگری شده اند و اکنون به طور جداگانه با یک منبع درخت در هر jar مدول مدیریت می شوند:

* org.springframework.aop
* org.springframework.beans
* org.springframework.context
* org.springframework.context.support
* org.springframework.expression
* org.springframework.instrument
* org.springframework.jdbc
* org.springframework.jms
* org.springframework.orm
* org.springframework.oxm
* org.springframework.test
* org.springframework.transaction
* org.springframework.web
* org.springframework.web.portlet
* org.springframework.web.servlet
* org.springframework.web.struts

مصنوع Spring.jar که تقریباً کل چارچوب را در بر می گرفت، دیگر ارائه نشده است.

ما اکنون از یک سیستم ساخت Spring جدید به نام Spring Web Flow 2.0 استفاده می کنیم. این به ما می دهد:

* Ivy-based "Spring Build" system
* consistent deployment procedure
* consistent dependency management
* consistent generation of OSGi manifests

## 2.5 Overview of new features

این لیستی از ویژگی های جدید Spring Framework 3.0 است. در ادامه این بخش به جزئیات بیشتر این ویژگی ها خواهیم پرداخت.

* Spring Expression Language
* IoC enhancements/Java based bean metadata
* General-purpose type conversion system and field formatting system
* Object to XML mapping functionality (OXM) moved from Spring Web Services project
* Comprehensive REST support
* @MVC additions
* Declarative model validation
* Early support for Java EE 6
* Embedded database support

### 2.5.1 Core APIs updated for Java 5

رابط BeanFactory نمونه‌هایtype bean را تا آنجا که ممکن است برمی‌گرداند:

* T getBean(Class<T> requiredType)
* T getBean(String name, Class<T> requiredType)
* Map<String, T> getBeansOfType(Class<T> type)

رابط TaskExecutor Spring اکنون java.util.concurrent.Executor را گسترش می دهد:

توسعه یافته AsyncTaskExecutor از Calables استاندارد با Futures پشتیبانی می کند

API و SPI مبدل جدید مبتنی بر جاوا 5:

ConversionService و Converters بدون تابعیت

جایگزین JDK PropertyEditors استاندارد

تایپ شده ApplicationListener<E>

### 2.5.2 Spring Expression Language

Spring یک زبان عبارتی را معرفی می کند که از نظر نحو شبیه به Unified EL است اما ویژگی های قابل توجهی بیشتری را ارائه می دهد. زبان عبارت را می توان در هنگام تعریف XML و تعاریف Bean مبتنی بر Annotation استفاده کرد و همچنین به عنوان پایه ای برای پشتیبانی از زبان بیان در سراسر مجموعه Spring عمل می کند. جزئیات این قابلیت جدید را می توان در فصل Spring Expression Language (SpEL) یافت.

زبان بیان بهار برای ارائه یک زبان بیانی واحد با پشتیبانی خوب برای جامعه Spring ایجاد شده است که می تواند در همه محصولات موجود در مجموعه Spring استفاده شود. ویژگی‌های زبان آن بر اساس الزامات پروژه‌های مجموعه Spring، از جمله الزامات ابزار برای پشتیبانی از تکمیل کد در مجموعه ابزار SpringSource مبتنی بر Eclipse، هدایت می‌شود.

در زیر مثالی از نحوه استفاده از Expression Language برای پیکربندی برخی از ویژگی های تنظیمات پایگاه داده آورده شده است

<bean class="mycompany.RewardsTestDatabase">

<property name="databaseName"

value="#{systemProperties.databaseName}"/>

<property name="keyGenerator"

value="#{strategyBean.databaseKeyGenerator}"/>

</bean>

اگر ترجیح می دهید اجزای خود را با استفاده از حاشیه نویسی پیکربندی کنید، این عملکرد نیز در دسترس است:

*@Repository*

**public** **class** RewardsTestDatabase {

*@Value("#{systemProperties.databaseName}")*

**public** **void** setDatabaseName(String dbName) { … }

*@Value("#{strategyBean.databaseKeyGenerator}")*

**public** **void** setKeyGenerator(KeyGenerator kg) { … }

}

### 2.5.3 The Inversion of Control (IoC) container

#### Java based bean metadata

برخی از ویژگی های اصلی پروژه JavaConfig اکنون به Spring Framework اضافه شده است. این بدان معنی است که حاشیه نویسی های زیر اکنون مستقیماً پشتیبانی می شوند:

* @Configuration
* @Bean
* @DependsOn
* @Primary
* @Lazy
* @Import
* @ImportResource
* @Value

در اینجا نمونه ای از کلاس جاوا است که با استفاده از ویژگی های JavaConfig جدید، پیکربندی اولیه را ارائه می دهد:

**package** org.example.config;

*@Configuration*

**public** **class** AppConfig {

**private** *@Value("#{jdbcProperties.url}")* String jdbcUrl;

**private** *@Value("#{jdbcProperties.username}")* String username;

**private** *@Value("#{jdbcProperties.password}")* String password;

*@Bean*

**public** FooService fooService() {

**return** **new** FooServiceImpl(fooRepository());

}

*@Bean*

**public** FooRepository fooRepository() {

**return** **new** HibernateFooRepository(sessionFactory());

}

*@Bean*

**public** SessionFactory sessionFactory() {

*// wire up a session factory*

AnnotationSessionFactoryBean asFactoryBean =

**new** AnnotationSessionFactoryBean();

asFactoryBean.setDataSource(dataSource());

*// additional config*

**return** asFactoryBean.getObject();

}

*@Bean*

**public** DataSource dataSource() {

**return** **new** DriverManagerDataSource(jdbcUrl, username, password);

}

}

برای انجام این کار، باید ورودی اسکن مؤلفه زیر را در فایل XML application contextحداقل برنامه خود اضافه کنید.

<context:component-scan base-package="org.example.config"/>

<util:properties id="jdbcProperties" location="classpath:org/example/config/jdbc.properties"/>

یا می توانید یک کلاس Configuration@ را مستقیماً با استفاده از AnnotationConfigApplicationContext بوت استرپ کنید:

**public** **static** **void** main(String[] args) {

ApplicationContext ctx = **new** AnnotationConfigApplicationContext(AppConfig.**class**);

FooService fooService = ctx.getBean(FooService.**class**);

fooService.doStuff();

}

برای اطلاعات کامل در مورد AnnotationConfigApplicationContext به بخش 5.12.2، "نمونه سازی ظرف Spring با استفاده از AnnotationConfigApplicationContext" مراجعه کنید.

#### Defining bean metadata within components

روش‌های حاشیه‌نویسی @Bean نیز در اجزای Spring پشتیبانی می‌شوند. آنها تعریف factory beanرا به ظرف ارائه می دهند. برای اطلاعات بیشتر به تعریف فراداده bean در اجزاء مراجعه کنید

### 2.5.4 General purpose type conversion system and field formatting system