第二部分 Spring Framework 4.x的新增功能

Part II. What's New in Spring Framework 4.x

This chapter provides an overview of the new features and improvements that have been introduced with Spring Framework 4.3. If you are interested in more details, please see the link: Issue Tracker tickets that were resolved as part of the 4.3 development process.

翻译：本章概述了Spring Framework 4.3引入的新功能和改进。 如果您对更多详细信息感兴趣，请参阅链接：在4.3开发过程中已解决的Issue Tracker票证。

3. New Features and Enhancements in Spring Framework 4.0

The Spring Framework was first released in 2004; since then there have been significant major revisions: Spring 2.0 provided XML namespaces and AspectJ support; Spring 2.5 embraced annotation-driven configuration; Spring 3.0 introduced a strong Java 5+ foundation across the framework codebase, and features such as the Java-based `@Configuration` model.
Version 4.0 is the latest major release of the Spring Framework and the first to fully support Java 8 features. You can still use Spring with older versions of Java, however, the minimum requirement has now been raised to Java SE 6. We have also taken the opportunity of a major release to remove many deprecated classes and methods.
A migration guide for upgrading to Spring 4.0 is available on the Spring Framework GitHub Wiki.

翻译：3. Spring Framework 4.0中的新功能和增强功能
Spring框架于2004年首次发布。 从那时起，已经进行了重大的重大修订：Spring 2.0提供了XML名称空间和AspectJ支持。 Spring 2.5包含了注释驱动的配置； Spring 3.0在整个框架代码库中引入了强大的Java 5+基础，并提供了基于Java的@Configuration模型等功能。

版本4.0是Spring Framework的最新主要版本，也是第一个完全支持Java 8功能的版本。 您仍然可以将Spring与Java的较早版本一起使用，但是，最低要求现已提高到Java SE6。我们还利用主要版本的机会删除了许多不赞成使用的类和方法。

Spring Framework GitHub Wiki上提供了用于升级到Spring 4.0的迁移指南。

3.1 Improved Getting Started Experience

The new spring.io website provides a whole series of "Getting Started" guides to help you learn Spring. You can read more about the guides in the Chapter 1, *Getting Started with Spring* section in this document. The new website also provides a comprehensive overview of the many additional projects that are released under the Spring umbrella.
If you are a Maven user you may also be interested in the helpful bill of materials POM file that is now published with each Spring Framework release.

翻译：3.1改进的入门经验
新的spring.io网站提供了一系列“入门”指南，以帮助您学习Spring。 您可以在本文档的第1章“ Spring入门”部分中阅读有关指南的更多信息。 新网站还提供了在Spring框架下发布的许多其他项目的全面概述。

如果您是Maven用户，则可能对现在与每个Spring Framework版本一起发布的有用的物料清单POM文件感兴趣。

3.2 Removed Deprecated Packages and Methods

All deprecated packages, and many deprecated classes and methods have been removed with version 4.0. If you are upgrading from a previous release of Spring, you should ensure that you have fixed any deprecated calls that you were making to outdated APIs.
For a complete set of changes, check out the API Differences Report.
Note that optional third-party dependencies have been raised to a 2010/2011 minimum (i.e. Spring 4 generally only supports versions released in late 2010 or later now): notably, Hibernate 3.6+, EhCache 2.1+, Quartz 1.8+, Groovy 1.8+, and Joda-Time 2.0+. As an exception to the rule, Spring 4 requires the recent Hibernate Validator 4.3+, and support for Jackson has been focused on 2.0+ now (with Jackson 1.8/1.9 support retained for the time being where Spring 3.2 had it; now just in deprecated

form).

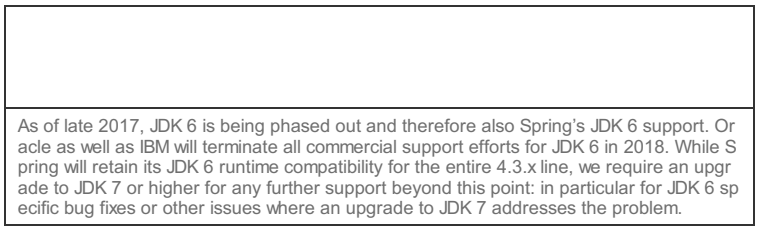
翻译：3.2移除过时的软件包和方法
所有不推荐使用的程序包以及许多不推荐使用的类和方法已在4.0版中删除。 如果要从Spring的早期版本进行升级，则应确保已修复了对过时的API进行的所有不推荐使用的调用。

要获取完整的更改集，请查看API差异报告。

请注意，可选的第三方依赖项已提高到2010/2011的最低要求（即，Spring 4通常仅支持2010年末或现在发布的版本）：尤其是Hibernate 3.6 +，EhCache 2.1 +，Quartz 1.8 +，Groovy 1.8+ 以及Joda-Time 2.0+。 作为规则的例外，Spring 4需要最新的Hibernate Validator 4.3+，并且对Jackson的支持现在集中于2.0+（暂时保留了Spring 3.2所拥有的Jackson 1.8 / 1.9支持；现在已弃用） 形成）。

3.3 Java 8 (as well as 6 and 7)

Spring Framework 4.0 provides support for several Java 8 features. You can make use of *lambda expressions* and *method references* with Spring's callback interfaces. There is first-class support for `java.time` (JSR-310), and several existing annotations have been retrofitted as `@Repeatable`. You can also use Java 8's parameter name discovery (based on the `-parameters` compiler flag) as an alternative to compiling your code with debug information enabled.
Spring remains compatible with older versions of Java and the JDK: concretely, Java SE 6 (specifically, a minimum level equivalent to JDK 6 update 18, as released in January 2010) and above are still fully supported. However, for newly started development projects based on Spring 4, we recommend the use of Java 7 or 8.



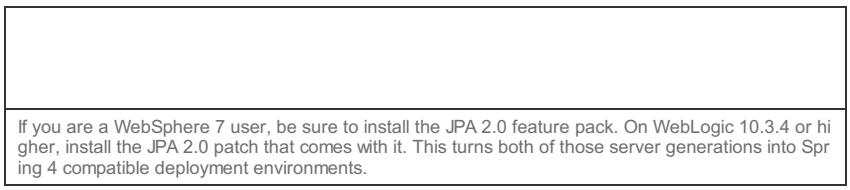
翻译：3.3 Java 8（以及6和7）
Spring Framework 4.0支持多种Java 8功能。您可以在Spring的回调接口中使用lambda表达式和方法引用。对java.time（JSR-310）有一流的支持，并且将几个现有的注释改型为@Repeatable。您也可以使用Java 8的参数名称发现（基于-parameters编译器标志）来替代启用调试信息的代码。

Spring仍与Java和JDK的较早版本兼容：具体而言，仍完全支持Java SE 6（特别是最低级别，等同于JDK 6 update 18（于2010年1月发布））。但是，对于基于Spring 4的新开始的开发项目，我们建议使用Java 7或8。

[注意]
截至2017年底，JDK 6已被淘汰，因此Spring的JDK 6支持也已淘汰。 Oracle和IBM将在2018年终止对JDK 6的所有商业支持。 Spring将保留其对整个4.3.x系列的JDK 6运行时兼容性，但我们需要升级到JDK 7或更高版本，以获取更多支持。 ：特别是针对JDK 6的特定错误修复或其他问题，其中升级到JDK 7可以解决该问题。

3.4 Java EE 6 and 7

Java EE version 6 or above is now considered the baseline for Spring Framework 4, with the JPA 2.0 and Servlet 3.0 specifications being of particular relevance. In order to remain compatible with Google App Engine and older application servers, it is possible to deploy a Spring 4 application into a Servlet 2.5 environment. However, Servlet 3.0+ is strongly recommended and a prerequisite in Spring's test and mock packages for test setups in development environments.



On a more forward-looking note, Spring Framework 4.0 supports the Java EE 7 level of applicable specifications now: in particular, JMS 2.0, JTA 1.2, JPA 2.1, Bean Validation 1.1, and JSR-236 Concurrency Utilities. As usual, this support focuses on individual use of those specifications, e.g. on Tomcat or in standalone environments. However, it works equally well when a Spring application is deployed to a Java EE 7 server.
Note that Hibernate 4.3 is a JPA 2.1 provider and therefore only supported as of Spring Framework 4.0. The same applies to Hibernate Validator 5.0 as a Bean Validation 1.1 provider. Neither of the two are officially supported with Spring Framework 3.2.

翻译：3.4 Java EE 6和7
现在，Java EE 6或更高版本被视为Spring Framework 4的基线，其中特别重要的是JPA 2.0和Servlet 3.0规范。为了与Google App Engine和较旧的应用程序服务器保持兼容，可以将Spring 4应用程序部署到Servlet 2.5环境中。但是，强烈建议使用Servlet 3.0+，这是Spring的测试和模拟包中用于开发环境中测试设置的先决条件。

[注意]
如果您是WebSphere 7用户，请确保安装JPA 2.0功能包。在WebLogic 10.3.4或更高版本上，安装其随附的JPA 2.0补丁。这将这两代服务器转变为与Spring 4兼容的部署环境。

更具前瞻性的是，Spring Framework 4.0现在支持Java EE 7级别的适用规范：特别是JMS 2.0，JTA 1.2，JPA 2.1，Bean Validation 1.1和JSR-236并发实用程序。与往常一样，此支持侧重于这些规范的单独使用，例如在Tomcat或独立环境中。但是，当将Spring应用程序部署到Java EE 7服务器时，它同样可以很好地工作。

请注意，Hibernate 4.3是JPA 2.1提供程序，因此从Spring Framework 4.0开始仅受支持。同样适用于Hibernate Validator 5.0作为Bean Validation 1.1提供程序。 Spring Framework 3.2均未正式支持这两者。

3.5 Groovy Bean Definition DSL

Beginning with Spring Framework 4.0, it is possible to define external bean configuration using a Groovy DSL. This is similar in concept to using XML bean definitions but allows for a more concise syntax. Using Groovy also allows you to easily embed bean definitions directly in your bootstrap code. For example:

```
def reader = new GroovyBeanDefinitionReader(myApplicationContext)
reader.beans {
    dataSource(BasicDataSource) {
        driverClassName = "org.hsqldb.jdbcDriver"
        url = "jdbc:hsqldb:mem:grailsDB"
        username = "sa"
        password = ""
        settings = [mynew:"setting"]
    }
    sessionFactory(SessionFactory) {
        dataSource = dataSource
    }
    myService(MyService) {
        nestedBean = { AnotherBean bean ->
            dataSource = dataSource
        }
    }
}
```

For more information consult the `GroovyBeanDefinitionReader` javadocs.

翻译：3.5 Groovy Bean定义DSL
从Spring Framework 4.0开始，可以使用Groovy DSL定义外部bean配置。 这在概念上与使用XML bean定义相似，但是允许使用更简洁的语法。 使用Groovy还可以使您轻松地将Bean定义直接嵌入到引导代码中。 例如：

```
def reader = new GroovyBeanDefinitionReader(myApplicationContext)
reader.beans {
    dataSource(BasicDataSource) {
        driverClassName = "org.hsqldb.jdbcDriver"
        url = "jdbc:hsqldb:mem:grailsDB"
        username = "sa"
        password = ""
        settings = [mynew:"setting"]
    }
    sessionFactory(SessionFactory) {
        dataSource = dataSource
    }
    myService(MyService) {
        nestedBean = { AnotherBean bean ->
            dataSource = dataSource
        }
    }
}
```

有关更多信息，请参阅GroovyBeanDefinitionReader javadocs。

3.6 Core Container Improvements

There have been several general improvements to the core container:

- Spring now treats *generic types as a form of qualifier* when injecting Beans. For example, if you are using a Spring Data `Repository` you can now easily inject a specific implementation: `@Autowired Repository<Customer> customerRepository`.
- If you use Spring's meta-annotation support, you can now develop custom annotations that expose specific attributes from the source annotation.
- Beans can now be *ordered* when they are autowired into lists and arrays. Both the `@Order` annotation and `Ordered` interface are supported.
- The `@Lazy` annotation can now be used on injection points, as well as on `@Bean` definitions.
- The `@Description` annotation has been introduced for developers using Java-based configuration.
- A generalized model for conditionally filtering beans has been added via the `@Conditional` annotation. This is similar to `@Profile` support but allows for user-defined strategies to be developed programmatically.
- CGLIB-based proxy classes no longer require a default constructor. Support is provided via the objenesis library which is repackaged *inline* and distributed as part of the Spring Framework. With this strategy, no constructor at all is being invoked for proxy instances anymore.
- There is managed time zone support across the framework now, e.g. on `LocaleContext`.

翻译：3.6核心容器的改进
对核心容器进行了一些常规改进：

现在，在注入Bean时，Spring现在将泛型类型视为限定符的一种形式。例如，如果您使用的是Spring Data Repository，则现在可以轻松注入特定的实现：@Autowired Repository <Customer> customerRepository。
如果您使用Spring的元注释支持，则现在可以开发自定义注释，以显示源注释中的特定属性。
现在，可以在将Bean自动装配到列表和数组中时对其进行订购。同时支持@Order批注和Ordered接口。
@Lazy注释现在可以在注入点以及@Bean定义上使用。
@Description注释已为使用基于Java的配置的开发人员引入。
通过@Conditional批注添加了用于条件过滤bean的通用模型。这类似于@Profile支持，但是允许以编程方式开发用户定义的策略。
基于CGLIB的代理类不再需要默认的构造函数。通过objenesis库提供支持，该库以内联方式重新打包并作为Spring框架的一部分分发。通过这种策略，不再为代理实例调用任何构造函数。
现在，整个框架都提供托管时区支持，例如在LocaleContext上。

3.7 General Web Improvements

Deployment to Servlet 2.5 servers remains an option, but Spring Framework 4.0 is now focused primarily on Servlet 3.0+ environments. If you are using the Spring MVC Test Framework you will need to ensure that a Servlet 3.0 compatible JAR is in your *test classpath*.
In addition to the WebSocket support mentioned later, the following general improvements have been made to Spring's Web modules:

- You can use the new `@RestController` annotation with Spring MVC applications, removing the need to add `@ResponseBody` to each of your `@RequestMapping` methods.
- The `AsyncRestTemplate` class has been added, allowing non-blocking asynchronous support when developing REST clients.
- Spring now offers comprehensive timezone support when developing Spring MVC applications.

翻译：3.7常规Web改进
仍然可以选择部署到Servlet 2.5服务器，但是Spring Framework 4.0现在主要集中在Servlet 3.0+环境。 如果您使用的是Spring MVC测试框架，则需要确保测试类路径中包含与Servlet 3.0兼容的JAR。

除了后面提到的WebSocket支持之外，Spring的Web模块还进行了以下常规改进：

您可以在Spring MVC应用程序中使用新的@RestController批注，而无需将@ResponseBody添加到每个@RequestMapping方法中。
添加了AsyncRestTemplate类，在开发REST客户端时允许非阻塞异步支持。
在开发Spring MVC应用程序时，Spring现在提供全面的时区支持。

3.8 WebSocket, SockJS, and STOMP Messaging

A new `spring-websocket` module provides comprehensive support for WebSocket-based, two-way communication between client and server in web applications. It is compatible with JSR-356, the Java WebSocket API, and in addition provides SockJS-based fallback options (i.e. WebSocket emulation) for use in browsers that don't yet support the WebSocket protocol (e.g. Internet Explorer < 10).
A new `spring-messaging` module adds support for STOMP as the WebSocket sub-protocol to use in applications along with an annotation programming model for routing and processing STOMP messages from WebSocket clients. As a result an `@Controller` can now contain both `@RequestMapping` and `@MessageMapping` methods for handling HTTP requests and messages from WebSocket-connected clients. The new `spring-messaging` module also contains key abstractions formerly from the Spring Integration project such as `Message`, `MessageChannel`, `MessageHandler`, and others to serve as a foundation for messaging-based applications.
For further details, including a more thorough introduction, see the Chapter 26, *WebSocket Support* section.

翻译：3.8 WebSocket，SockJS和STOMP消息传递
一个新的spring-websocket模块为Web应用程序中客户端和服务器之间基于WebSocket的双向通信提供了全面的支持。它与Java WebSocket API JSR-356兼容，此外还提供了基于SockJS的后备选项（即WebSocket仿真），可在尚不支持WebSocket协议（例如Internet Explorer <10）的浏览器中使用。

一个新的spring-messaging模块增加了对STOMP作为WebSocket子协议的支持，以在应用程序中使用，以及一个注释编程模型，用于路由和处理来自WebSocket客户端的STOMP消息。结果，@ Controller现在可以同时包含@RequestMapping和@MessageMapping方法，用于处理来自WebSocket连接的客户端的HTTP请求和消息。新的spring-messaging模块还包含以前来自Spring Integration项目的关键抽象，例如Message，MessageChannel，MessageHandler等，这些抽象作为基于消息的应用程序的基础。

有关更多详细信息，包括更全面的介绍，请参见第26章，WebSocket支持。

3.9 Testing Improvements

In addition to pruning of deprecated code within the `spring-test` module, Spring Framework 4.0 introduces several new features for use in unit and integration testing.

- Almost all annotations in the `spring-test` module (e.g., `@ContextConfiguration`, `@WebAppConfiguration`,

`@ContextHierarchy`, `@ActiveProfiles`, etc.) can now be used as meta-annotations to create custom *composed annotations* and reduce configuration duplication across a test suite.

- Active bean definition profiles can now be resolved programmatically, simply by implementing a custom `ActiveProfilesResolver` and registering it via the `resolver` attribute of `@ActiveProfiles`.
- A new `SocketUtils` class has been introduced in the `spring-core` module which enables you to scan for free TCP and UDP server ports on localhost. This functionality is not specific to testing but can prove very useful when writing integration tests that require the use of sockets, for example tests that start an in-memory SMTP server, FTP server, Servlet container, etc.
- As of Spring 4.0, the set of mocks in the `org.springframework.mock.web` package is now based on the Servlet 3.0 API. Furthermore, several of the Servlet API mocks (e.g., `MockHttpServletRequest`, `MockServletContext`, etc.) have been updated with minor enhancements and improved configurability.

翻译：3.9测试改进
除了在spring-test模块中修剪不赞成使用的代码外，Spring Framework 4.0还引入了一些新功能，用于单元测试和集成测试。

弹簧测试模块中的几乎所有注释（例如@ ContextConfiguration，@ WebAppConfiguration，@ ContextHierarchy，@ ActiveProfiles等）现在都可以用作元注释，以创建自定义的组合注释并减少整个测试套件中的配置重复项。
现在，只需实现自定义ActiveProfilesResolver并通过@ActiveProfiles的resolver属性对其进行注册，就可以通过编程方式解析活动bean定义概要文件。
在spring-core模块中引入了新的SocketUtils类，使您可以扫描本地主机上的免费TCP和UDP服务器端口。此功能不是特定于测试的，但是在编写要求使用套接字的集成测试（例如，启动内存中的SMTP服务器，FTP服务器，Servlet容器等的测试）时，可以证明非常有用。
从Spring 4.0开始，org.springframework.mock.web包中的模拟集现在基于Servlet 3.0 API。此外，已更新了一些Servlet API模拟（例如MockHttpServletRequest，MockServletContext等），并进行了较小的增强和改进了可配置性。

4. New Features and Enhancements in Spring Framework 4.1

Version 4.1 included a number of improvements, as described in the following sections:

- Section 4.1, “JMS Improvements”
- Section 4.2, “Caching Improvements”
- Section 4.3, “Web Improvements”
- Section 4.4, “WebSocket Messaging Improvements”
- Section 4.5, “Testing Improvements”

翻译：4. Spring Framework 4.1中的新功能和增强功能
4.1版包括许多改进，如以下各节所述：

第4.1节“ JMS改进”
第4.2节“缓存改进”
第4.3节“ Web改进”
第4.4节“ WebSocket消息传递改进”
第4.5节“测试改进”

4.1 JMS Improvements

Spring 4.1 introduces a much simpler infrastructure to register JMS listener endpoints by annotating bean methods with `@JmsListener`. The XML namespace has been enhanced to support this new style (`jms:annotation-driven`), and it is also possible to fully configure the infrastructure using Java config (`@EnableJms`, `JmsListenerContainerFactory`). It is also possible to register listener endpoints programmatically using `JmsListenerConfigurer`.
Spring 4.1 also aligns its JMS support to allow you to benefit from the `spring-messaging` abstraction introduced in 4.0, that is:

- Message listener endpoints can have a more flexible signature and benefit from standard messaging annotations such as `@Payload`, `@Header`, `@Headers`, and `@SendTo`. It is also possible to use a standard `Message` in lieu of `javax.jms.Message` as method argument.

- A new `JmsMessageOperations` interface is available and permits `JmsTemplate` like operations using the `Message` abstraction.

Finally, Spring 4.1 provides additional miscellaneous improvements:

- Synchronous request-reply operations support in `JmsTemplate`
- Listener priority can be specified per `<jms:listener/>` element
- Recovery options for the message listener container are configurable using a `BackOff` implementation
- JMS 2.0 shared consumers are supported

翻译：4.1 JMS的改进
Spring 4.1通过使用@JmsListener注释bean方法，引入了一种更为简单的基础结构来注册JMS侦听器端点。 XML名称空间已得到增强，以支持此新样式（jms：由注释驱动），并且还可以使用Java config（@ EnableJms，JmsListenerContainerFactory）完全配置基础结构。也可以使用JmsListenerConfigurer以编程方式注册侦听器端点。

Spring 4.1还调整了其对JMS的支持，使您可以从4.0中引入的spring-messaging抽象中受益，即：

消息侦听器终结点可以具有更灵活的签名，并可以从标准消息注释中受益，例如@ Payload，@ Header，@ Headers和@SendTo。也可以使用标准Message代替javax.jms.Message作为方法参数。
新的JmsMessageOperations接口可用，并允许使用Message抽象的JmsTemplate之类的操作。
最后，Spring 4.1提供了其他杂项改进：

JmsTemplate中的同步请求-答复操作支持
可以根据每个<jms：listener />元素指定侦听器优先级
可以使用BackOff实现配置消息侦听器容器的恢复选项
支持JMS 2.0共享使用者

4.2 Caching Improvements

Spring 4.1 supports JCache (JSR-107) annotations using Spring's existing cache configuration and infrastructure abstraction; no changes are required to use the standard annotations.
Spring 4.1 also improves its own caching abstraction significantly:

- Caches can be resolved at runtime using a `CacheResolver`. As a result the `value` argument defining the cache name(s) to use is no longer mandatory.
- More operation-level customizations: cache resolver, cache manager, key generator
- A new `@CacheConfig` class-level annotation allows common settings to be shared at the class level **without** enabling any cache operation.
- Better exception handling of cached methods using `CacheErrorHandler`

Spring 4.1 also has a breaking change in the `Cache` interface as a new `putIfAbsent` method has been added.

翻译：4.2缓存改进
Spring 4.1使用Spring现有的缓存配置和基础架构抽象来支持JCache（JSR-107）批注； 使用标准注释无需进行任何更改。

Spring 4.1还大大改善了自己的缓存抽象：

可以在运行时使用CacheResolver解析缓存。 结果，定义要使用的缓存名称的value参数不再是必需的。
更多操作级别的自定义：缓存解析器，缓存管理器，密钥生成器
新的@CacheConfig类级别注释允许在类级别共享通用设置，而无需启用任何缓存操作。
使用CacheErrorHandler更好地处理缓存方法的异常
由于添加了新的putIfAbsent方法，Spring 4.1在Cache接口中也发生了重大变化。

4.3 Web Improvements

- The existing support for resource handling based on the `ResourceHttpRequestHandler` has been expanded with new abstractions `ResourceResolver`, `ResourceTransformer`, and `ResourceUrlProvider`. A number of built-in implementations provide support for versioned resource URLs (for effective HTTP caching), locating gzipped resources, generating an HTML 5 AppCache manifests, and more. See Section 22.16.9, "Serving of Resources".
- JDK 1.8's `java.util.Optional` is now supported for `@RequestParam`, `@RequestHeader`, and `@MatrixVariable` controller method arguments.
- `ListenableFuture` is supported as a return value alternative to `DeferredResult` where an underlying service (or perhaps a call to `AsyncRestTemplate`) already returns `ListenableFuture`.
- `@ModelAttribute` methods are now invoked in an order that respects inter-dependencies. See SPR-6299.
- Jackson's `@JsonView` is supported directly on `@ResponseBody` and `ResponseEntity` controller methods for serializing different amounts of detail for the same POJO (e.g. summary vs. detail page). This is also supported with View-based rendering by adding the serialization view type as a model attribute under a special key. See the section called "Jackson Serialization View Support" for details.
- JSONP is now supported with Jackson. See the section called "Jackson JSONP Support".
- A new lifecycle option is available for intercepting `@ResponseBody` and `ResponseEntity` methods just after the controller method returns and before the response is written. To take advantage declare an `@ControllerAdvice` bean that implements `ResponseBodyAdvice`. The built-in support for `@JsonView` and JSONP take advantage of this. See Section 22.4.1, "Intercepting requests with a HandlerInterceptor".
- There are three new `HttpMessageConverter` options:
    - Gson — lighter footprint than Jackson; has already been in use in Spring Android.
    - Google Protocol Buffers — efficient and effective as an inter-service communication data protocol within an enterprise but can also be exposed as JSON and XML for browsers.
    - Jackson based XML serialization is now supported through the jackson-dataformat-xml extension. When using `@EnableWebMvc` or `<mvc:annotation-driven/>`, this is used by default instead of JAXB2 if `jackson-dataformat-xml` is in the classpath.
- Views such as JSPs can now build links to controllers by referring to controller mappings by name. A default name is assigned to every `@RequestMapping`. For example `FooController` with method `handleFoo` is named "FC#handleFoo". The naming strategy is pluggable. It is also possible to name an `@RequestMapping` explicitly through its name attribute. A new `mvcUrl` function in the Spring JSP tag library makes this easy to use in JSP pages. See Section 22.7.3, "Building URIs to Controllers and methods from views".
- `ResponseEntity` provides a builder-style API to guide controller methods towards the preparation of server-side responses, e.g. `ResponseEntity.ok()`.
- `RequestEntity` is a new type that provides a builder-style API to guide client-side REST code towards the preparation of HTTP requests.
- MVC Java config and XML namespace:
    - View resolvers can now be configured including support for content negotiation, see Section 22.16.8, "View Resolvers".
    - View controllers now have built-in support for redirects and for setting the response status. An application can use this to configure redirect URLs, render 404 responses with a view, send "no content" responses, etc. Some use cases are listed here.
    - Path matching customizations are frequently used and now built-in. See Section 22.16.11, "Path Matching".
- Groovy markup template support (based on Groovy 2.3). See the `GroovyMarkupConfigurer` and respecitve `ViewResolver` and `View' implementations.

翻译：4.3网站改进

基于ResourceHttpRequestHandler的对资源处理的现有支持已通过新的抽象ResourceResolver，ResourceTransformer和ResourceUrlProvider进行了扩展。许多内置实现提供对版本化资源URL（用于有效的HTTP缓存），定位压缩后的资源，生成HTML 5 AppCache清单等的支持。请参见第22.16.9节“资源服务”。
@ RequestParam，@ RequestHeader和@MatrixVariable控制器方法参数现在支持JDK 1.8的java.util.Optional。
支持将ListenableFuture作为DeferredResult的替代返回值，在该方法中，基础服务（或对AsyncRestTemplate的调用）已返回ListenableFuture。
现在按尊重相互依赖关系的顺序调用@ModelAttribute方法。请参阅SPR-6299。
@ResponseBody和ResponseEntity控制器方法直接支持Jackson的@JsonView，用于序列化同一POJO的不同数量的细节（例如，摘要与细节页面）。通过将序列化视图类型添加为特殊键下的模型属性，基于视图的渲染也支持此功能。有关详细信息，请参见“Jackson序列化视图支持”部分。
杰克逊现在支持JSONP。请参阅“杰克逊JSONP支持”部分。
一个新的生命周期选项可用于在控制器方法返回之后和写入响应之前拦截@ResponseBody和ResponseEntity方法。为了利用这一点，请声明一个实现ResponseBodyAdvice的@ControllerAdvice bean。对@JsonView和JSONP的内置支持利用了这一点。请参见第22.4.1节“使用HandlerInterceptor拦截请求”。
新增了三个HttpMessageConverter选项：

格森（Gson）-比杰克逊（Jackson）足迹更轻；已在Spring Android中使用。
Google Protocol Buffers –高效，有效地用作企业内部的服务间通信数据协议，但对于浏览器也可以公开为JSON和XML。
现在，通过jackson-dataformat-xml扩展支持基于Jackson的XML序列化。当使用@EnableWebMvc或<mvc：annotation-driven />时，如果在类路径中包含jackson-dataformat-xml，则默认使用它而不是JAXB2。
现在，诸如JSP之类的视图可以通过按名称引用控制器映射来建立到控制器的链接。默认名称分配给每个@RequestMapping。例如，将带有handleFoo方法的FooController命名为“ FC＃handleFoo”。命名策略是可插入的。也可以通过名称属性@RequestMapping明确命名。 Spring JSP标记库中的新mvcUrl函数使此功能易于在JSP页面中使用。请参见第22.7.3节“从视图为控制器和方法构建URI”。
ResponseEntity提供了一个构建器样式的API，可指导控制器方法准备服务器端响应，例如ResponseEntity.ok（）。
RequestEntity是一种新类型，它提供了一个生成器样式的API，以指导客户端REST代码编写HTTP请求。
MVC Java配置和XML名称空间：

现在可以配置视图解析器，包括对内容协商的支持，请参见第22.16.8节“视图解析器”。
View Controller现在具有对重定向和设置响应状态的内置支持。应用程序可以使用它来配置重定向URL，使用视图呈现404响应，发送“无内容”响应等。此处列出了一些用例。
路径匹配自定义是常用的，现在已内置。请参见第22.16.11节“路径匹配”。
Groovy标记模板支持（基于Groovy 2.3）。请参阅GroovyMarkupConfigurer并指定ViewResolver和`View'实现。

4.4 WebSocket Messaging Improvements

- SockJS (Java) client-side support. See `SockJsClient` and classes in same package.
- New application context events `SessionSubscribeEvent` and `SessionUnsubscribeEvent` published when STOMP clients subscribe and unsubscribe.
- New "websocket" scope. See Section 26.4.16, “WebSocket Scope”.
- `@SendToUser` can target only a single session and does not require an authenticated user.
- `@MessageMapping` methods can use dot "." instead of slash "/" as path separator. See SPR-11660.
- STOMP/WebSocket monitoring info collected and logged. See Section 26.4.18, “Monitoring”.
- Significantly optimized and improved logging that should remain very readable and compact even at DEBUG level.
- Optimized message creation including support for temporary message mutability and avoiding automatic message id and timestamp creation. See Javadoc of `MessageHeaderAccessor`.
- Close STOMP/WebSocket connections that have no activity within 60 seconds after the WebSocket session is established. See SPR-11884.

翻译：4.4 WebSocket消息传递改进
SockJS（Java）客户端支持。请参阅同一软件包中的SockJsClient和类。

当STOMP客户端订阅和取消订阅时，将发布新的应用程序上下文事件SessionSubscribeEvent和SessionUnsubscribeEvent。
新的“ websocket”作用域。请参见第26.4.16节“ WebSocket范围”。
@SendToUser只能针对单个会话，并且不需要经过身份验证的用户。
@MessageMapping方法可以使用点“。”而不是使用斜杠“ /”作为路径分隔符。请参阅SPR-11660。
收集并记录STOMP / WebSocket监视信息。请参见第26.4.18节“监视”。
显着优化和改进的日志记录，即使在DEBUG级别也应保持可读性和紧凑性。
优化的消息创建，包括对临时消息可变性的支持，并避免自动创建消息ID和时间戳。请参阅MessageHeaderAccessor的Javadoc。
建立WebSocket会话后60秒内，关闭没有活动的STOMP / WebSocket连接。参见SPR-11884。

4.5 Testing Improvements

- Groovy scripts can now be used to configure the `ApplicationContext` loaded for integration tests in the TestContext framework.
    - See the section called “Context configuration with Groovy scripts” for details.
- Test-managed transactions can now be programmatically started and ended within transactional test methods via the new `TestTransaction` API.
    - See the section called “Programmatic transaction management” for details.
- SQL script execution can now be configured declaratively via the new `@Sql` and `@SqlConfig` annotations on a per-class or per-method basis.
    - See Section 15.5.8, “Executing SQL scripts” for details.
- Test property sources which automatically override system and application property sources can be configured via the new `@TestPropertySource` annotation.
    - See the section called “Context configuration with test property sources” for details.
- Default `TestExecutionListener`s can now be automatically discovered.
    - See the section called “Automatic discovery of default TestExecutionListeners” for details.
- Custom `TestExecutionListener`s can now be automatically merged with the default listeners.
    - See the section called “Merging TestExecutionListeners” for details.
- The documentation for transactional testing support in the TestContext framework has been improved with more thorough explanations and additional examples.
    - See Section 15.5.7, “Transaction management” for details.
- Various improvements to `MockServletContext`, `MockHttpServletRequest`, and other Servlet API mocks.
- `AssertThrows` has been refactored to support `Throwable` instead of `Exception`.
- In Spring MVC Test, JSON responses can be asserted with JSON Assert as an extra option to using JSONPath much like it has been possible to do for XML with XMLUnit.
- `MockMvcBuilder` *recipes* can now be created with the help of `MockMvcConfigurer`. This was added to make it easy to apply Spring Security setup but can be used to encapsulate common setup for any 3rd party framework or within a project.
- `MockRestServiceServer` now supports the `AsyncRestTemplate` for client-side testing.

翻译：4.5测试改进
Groovy脚本现在可以用于配置为TestContext框架中的集成测试加载的ApplicationContext。

有关详细信息，请参见“使用Groovy脚本进行上下文配置”一节。

现在，可以通过新的TestTransaction API以编程方式在事务测试方法中启动和结束测试管理的事务。

有关详细信息，请参见“程序化事务管理”部分。
现在，可以在每个类或每个方法的基础上，通过新的@Sql和@SqlConfig注释性地配置SQL脚本执行。

有关详细信息，请参见第15.5.8节“执行SQL脚本”。
可以通过新的@TestPropertySource批注来配置自动覆盖系统和应用程序属性源的测试属性源。

有关详细信息，请参见“具有测试属性源的上下文配置”一节。
现在可以自动发现默认的TestExecutionListeners。

有关详细信息，请参见“自动发现默认的TestExecutionListeners”一节。
现在可以将自定义TestExecutionListeners与默认侦听器自动合并。

有关详细信息，请参见“合并TestExecutionListeners”一节。
通过更详尽的解释和更多示例，对TestContext框架中的事务测试支持文档进行了改进。

有关详细信息，请参见第15.5.7节“事务管理”。
对MockServletContext，MockHttpServletRequest和其他Servlet API模拟进行了各种改进。
AssertThrows已重构为支持Throwable而不是Exception。
在Spring MVC Test中，可以使用JSON Assert声明JSON响应，这是使用JSONPath的额外选项，就像使用XMLUnit对XML进行处理一样。
现在可以借助MockMvcConfigurer创建MockMvcBuilder配方。添加它是为了使应用Spring Security设置变得容易，但可以用于封装任何第三方框架或项目内的通用设置。
MockRestServiceServer现在支持AsyncRestTemplate进行客户端测试。

5. New Features and Enhancements in Spring Framework 4.2

Version 4.2 included a number of improvements, as described in the following sections:

- Section 5.1, “Core Container Improvements”
- Section 5.2, “Data Access Improvements”
- Section 5.3, “JMS Improvements”
- Section 5.4, “Web Improvements”
- Section 5.5, “WebSocket Messaging Improvements”
- Section 5.6, “Testing Improvements”

翻译：5. Spring Framework 4.2中的新功能和增强功能
4.2版包括许多改进，如以下各节所述：

第5.1节“核心容器的改进”
第5.2节“数据访问改进”
第5.3节“ JMS改进”
第5.4节“ Web改进”
第5.5节“ WebSocket消息传递改进”
第5.6节“测试改进”

5.1 Core Container Improvements

- Annotations such as `@Bean` get detected and processed on Java 8 default methods as well, allowing for composing a configuration class from interfaces with default `@Bean` methods.
- Configuration classes may declare `@Import` with regular component classes now, allowing for a mix of imported configuration classes and component classes.

- Configuration classes may declare an `@Order` value, getting processed in a corresponding order (e.g. for overriding beans by name) even when detected through classpath scanning.
- `@Resource` injection points support an `@Lazy` declaration, analogous to `@Autowired`, receiving a lazy-initializing proxy for the requested target bean.
- The application event infrastructure now offers an annotation-based model as well as the ability to publish any arbitrary event.
    - Any public method in a managed bean can be annotated with `@EventListener` to consume events.
    - `@TransactionalEventListener` provides transaction-bound event support.
- Spring Framework 4.2 introduces first-class support for declaring and looking up aliases for annotation attributes. The new `@AliasFor` annotation can be used to declare a pair of aliased attributes within a single annotation or to declare an alias from one attribute in a custom composed annotation to an attribute in a meta-annotation.
    - The following annotations have been retrofitted with `@AliasFor` support in order to provide meaningful aliases for their `value` attributes: `@Cacheable`, `@CacheEvict`, `@CachePut`, `@ComponentScan`, `@ComponentScan.Filter`, `@ImportResource`, `@Scope`, `@ManagedResource`, `@Header`, `@Payload`, `@SendToUser`, `@ActiveProfiles`, `@ContextConfiguration`, `@Sql`, `@TestExecutionListeners`, `@TestPropertySource`, `@Transactional`, `@ControllerAdvice`, `@CookieValue`, `@CrossOrigin`, `@MatrixVariable`, `@RequestHeader`, `@RequestMapping`, `@RequestParam`, `@RequestPart`, `@ResponseStatus`, `@SessionAttributes`, `@ActionMapping`, `@RenderMapping`, `@EventListener`, `@TransactionalEventListener`.
    - For example, `@ContextConfiguration` from the `spring-test` module is now declared as follows:

```
public @interface ContextConfiguration {

    @AliasFor("locations")
    String[] value() default {};

    @AliasFor("value")
    String[] locations() default {};

    // ...
}
```

- Similarly, *composed annotations* that override attributes from meta-annotations can now use `@AliasFor` for fine-grained control over exactly which attributes are overridden within an annotation hierarchy. In fact, it is now possible to declare an alias for the `value` attribute of a meta-annotation.
- For example, one can now develop a composed annotation with a custom attribute override as follows.

```
@ContextConfiguration
public @interface MyTestConfig {

    @AliasFor(annotation = ContextConfiguration.class, attribute = "value")
    String[] xmlFiles();

    // ...
}
```

- See Spring Annotation Programming Model.

- Numerous improvements to Spring's search algorithms used for finding meta-annotations. For example, locally declared *composed annotations* are now favored over inherited annotations.
- *Composed annotations* that override attributes from meta-annotations can now be discovered on interfaces and on abstract, bridge, & interface methods as well as on classes, standard methods, constructors, and fields.
- Maps representing annotation attributes (and `AnnotationAttributes` instances) can be *synthesized* (i.e., converted) into an annotation.
- The features of field-based data binding (`DirectFieldAccessor`) have been aligned with the current property-based data binding (`BeanWrapper`). In particular, field-based binding now supports navigation for Collections, Arrays, and Maps.
- `DefaultConversionService` now provides out-of-the-box converters for `Stream`, `Charset`, `Currency`, and `TimeZone`. Such converters can be added individually to any arbitrary `ConversionService` as well.
- `DefaultFormattingConversionService` comes with out-of-the-box support for the value types in JSR-354 Money & Currency (if the 'javax.money' API is present on the classpath): namely, `MonetaryAmount` and `CurrencyUnit`. This includes support for applying `@NumberFormat`.
- `@NumberFormat` can now be used as a meta-annotation.
- `JavaMailSenderImpl` has a new `testConnection()` method for checking connectivity to the server.
- `ScheduledTaskRegistrar` exposes scheduled tasks.
- Apache `commons-pool2` is now supported for a pooling AOP `CommonsPool2TargetSource`.
- Introduced `StandardScriptFactory` as a JSR-223 based mechanism for scripted beans, exposed through the `lang:std` element in XML. Supports e.g. JavaScript and JRuby. (Note: JRubyScriptFactory and `lang:jruby` are deprecated now, in favor of using JSR-223.)

翻译：还可以在Java 8默认方法上检测并处理诸如@Bean之类的注释，从而允许使用具有默认@Bean方法的接口组成配置类。
配置类现在可以使用常规组件类声明@Import，从而允许混合使用导入的配置类和组件类。
配置类可以声明一个@Order值，即使通过类路径扫描检测到，也将以相应的顺序进行处理（例如用于按名称覆盖bean）。
@Resource注入点支持@Lazy声明，类似于@Autowired，它接收请求的目标bean的延迟初始化代理。
现在，应用程序事件基础结构提供了基于注释的模型以及发布任何任意事件的能力。

可以使用@EventListener注释托管bean中的任何公共方法以使用事件。
@TransactionalEventListener提供事务绑定事件支持。
Spring Framework 4.2引入了对声明和查找注释属性别名的一流支持。新的@AliasFor批注可用于在单个批注内声明一对别名属性，或声明从自定义组合批注中的一个属性到元批注中的属性的别名。

以下注释已通过@AliasFor支持进行了改进，以便为其值属性提供有意义的别名：@ Cacheable，@ CacheEvict，@ CachePut，@ ComponentScan，@ ComponentScan.Filter，@ ImportResource，@ Scope，@ ManagedResource，@ Header， @ Payload，@ SendToUser，@ ActiveProfiles，@ ContextConfiguration，@ Sql，@ TestExecutionListeners，@ TestPropertySource，@ Transactional，@ ControllerAdvice，@ CookieValue，@ CrossOrigin，@ MatrixVariable，@ RequestHeader，@ RequestMapping，@ RequestParam，@ RequestPart，@ ResponseStatus ，@ SessionAttributes，@ ActionMapping，@ RenderMapping，@ EventListener，@ TransactionalEventListener。
例如，来自spring-test模块的@ContextConfiguration现在声明如下：

```
public @interface ContextConfiguration {

    @AliasFor("locations")
    String[] value() default {};

    @AliasFor("value")
    String[] locations() default {};

    // ...
}
```

同样，覆盖元注释中属性的组合注释现在可以使用@AliasFor进行细粒度控制，以精确控制注释层次结构中哪些属性被覆盖。 实际上，现在可以为元注释的value属性声明别名。
例如，现在可以开发具有自定义属性覆盖的组合注释，如下所示。

```
@ContextConfiguration
public @interface MyTestConfig {

    @AliasFor(annotation = ContextConfiguration.class, attribute = "value")
    String[] xmlFiles();

    // ...
}
```

请参阅Spring注释编程模型。
改进了Spring用于查找元注释的搜索算法。例如，本地声明的组合注释现在比继承的注释更受青睐。
现在，可以在接口，抽象，桥接和接口方法以及类，标准方法，构造函数和字段上发现覆盖元注释属性的组合注释。
可以将表示注释属性（和AnnotationAttributes实例）的地图合成（即转换）为注释。
基于字段的数据绑定（DirectFieldAccessor）的功能已与当前基于属性的数据绑定（BeanWrapper）保持一致。特别是，基于字段的绑定现在支持对Collections，Arrays和Maps的导航。
DefaultConversionService现在为Stream，Charset，Currency和TimeZone提供了开箱即用的转换器。此类转换器也可以单独添加到任何任意ConversionService中。
DefaultFormattingConversionService为JSR-354货币和货币（如果类路径中存在“ javax.money” API）中的值类型提供了现成的支持：即MonetaryAmount和CurrencyUnit。这包括对应用@NumberFormat的支持。
@NumberFormat现在可以用作元注释。
JavaMailSenderImpl具有一个新的testConnection（）方法，用于检查与服务器的连接。
ScheduledTaskRegistrar公开计划的任务。
现在，池AOP CommonsPool2TargetSource支持Apache commons-pool2。
引入StandardScriptFactory作为基于JSR-223的脚本化bean的机制，通过XML中的lang：std元素公开。支持例如JavaScript和JRuby。 （注意：现在不建议使用JRubyScriptFactory和lang：jruby，而建议使用JSR-223。）

5.2 Data Access Improvements

- `javax.transaction.Transactional` is now supported via AspectJ.
- `SimpleJdbcCallOperations` now supports named binding.
- Full support for Hibernate ORM 5.0: as a JPA provider (automatically adapted) as well as through its native API (covered by the new `org.springframework.orm.hibernate5` package).
- Embedded databases can now be automatically assigned unique names, and `<jdbc:embedded-database>` supports a new `database-name` attribute. See "Testing Improvements" below for further details.

翻译：5.2数据访问改进
现在，通过AspectJ支持javax.transaction.Transactional。
现在，SimpleJdbcCallOperations支持命名绑定。
完全支持Hibernate ORM 5.0：作为JPA提供程序（自动改编）以及通过其本机API（由新的org.springframework.orm.hibernate5包覆盖）。
现在可以自动为嵌入式数据库分配唯一的名称，并且<jdbc：embedded-database>支持新的数据库名称属性。 有关更多详细信息，请参见下面的“测试改进”。

5.3 JMS Improvements

- The `autoStartup` attribute can be controlled via `JmsListenerContainerFactory`.
- The type of the reply `Destination` can now be configured per listener container.
- The value of the `@SendTo` annotation can now use a SpEL expression.

- The response destination can be computed at runtime using `JmsResponse`
- `@JmsListener` is now a repeatable annotation to declare several JMS containers on the same method (use the newly introduced `@JmsListeners` if you're not using Java8 yet).

翻译：5.3 JMS的改进
可以通过JmsListenerContainerFactory来控制autoStartup属性。
现在可以在每个侦听器容器中配置回复目标的类型。
@SendTo批注的值现在可以使用SpEL表达式。
可以在运行时使用JmsResponse计算响应目标
@JmsListener现在是一个可重复的批注，用于以相同的方法声明多个JMS容器（如果尚未使用Java8，请使用新引入的@JmsListeners）。

5.4 Web Improvements

- HTTP Streaming and Server-Sent Events support, see the section called "HTTP Streaming".
- Built-in support for CORS including global (MVC Java config and XML namespace) and local (e.g. `@CrossOrigin`) configuration. See Chapter 27, *CORS Support* for details.
- HTTP caching updates:
    - new `CacheControl` builder; plugged into `ResponseEntity`, `WebContentGenerator`, `ResourceHttpRequestHandler`.
    - improved ETag/Last-Modified support in `WebRequest`.
- Custom mapping annotations, using `@RequestMapping` as a meta-annotation.
- Public methods in `AbstractHandlerMethodMapping` to register and unregister request mappings at runtime.
- Protected `createDispatcherServlet` method in `AbstractDispatcherServletInitializer` to further customize the `DispatcherServlet` instance to use.
- `HandlerMethod` as a method argument on `@ExceptionHandler` methods, especially handy in `@ControllerAdvice` components.
- `java.util.concurrent.CompletableFuture` as an `@Controller` method return value type.
- Byte-range request support in `HttpHeaders` and for serving static resources.
- `@ResponseStatus` detected on nested exceptions.
- `UriTemplateHandler` extension point in the `RestTemplate`.
    - `DefaultUriTemplateHandler` exposes `baseUrl` property and path segment encoding options.
    - the extension point can also be used to plug in any URI template library.
- OkHTTP integration with the `RestTemplate`.
- Custom `baseUrl` alternative for methods in `MvcUriComponentsBuilder`.
- Serialization/deserialization exception messages are now logged at WARN level.
- Default JSON prefix has been changed from "{} && " to the safer ")]}', " one.
- New `RequestBodyAdvice` extension point and built-in implementation to support Jackson's `@JsonView` on `@RequestBody` method arguments.
- When using GSON or Jackson 2.6+, the handler method return type is used to improve serialization of parameterized types like `List<Foo>`.
- Introduced `ScriptTemplateView` as a JSR-223 based mechanism for scripted web views, with a focus on JavaScript view templating on Nashorn (JDK 8).

翻译：5.4网站改进
HTTP流和服务器发送事件支持，请参阅“ HTTP流”一节。

对CORS的内置支持，包括全局（MVC Java配置和XML名称空间）和本地（例如@CrossOrigin）配置。有关详细信息，请参见第27章，CORS支持。
HTTP缓存更新：

新的CacheControl构建器；插入ResponseEntity，WebContentGenerator，ResourceHttpRequestHandler。
改进了WebRequest中的ETag / Last-Modified支持。
使用@RequestMapping作为元注释的自定义映射注释。
AbstractHandlerMethodMapping中的公共方法在运行时注册和注销请求映射。
受保护的AbstractDispatcherServletInitializer中的createDispatcherServlet方法可进一步自定义要使用的DispatcherServlet实例。
HandlerMethod作为@ExceptionHandler方法的方法参数，在@ControllerAdvice组件中尤其方便。
java.util.concurrent.CompletableFuture作为@Controller方法的返回值类型。
HttpHeaders中的字节范围请求支持以及用于服务静态资源。
在嵌套异常上检测到@ResponseStatus。
RestTemplate中的UriTemplateHandler扩展点。

DefaultUriTemplateHandler公开baseUrl属性和路径段编码选项。
扩展点还可以用于插入任何URI模板库。
OkHTTP与RestTemplate集成。
MvcUriComponentsBuilder中方法的自定义baseUrl替代方法。
现在，序列化/反序列化异常消息记录在WARN级别。
默认JSON前缀已从“ {} &&“更改为更安全的“ ）]}'”。
新的RequestBodyAdvice扩展点和内置实现，以支持@RequestBody方法参数上的Jackson的@JsonView。
当使用GSON或Jackson 2.6+时，处理程序方法返回类型用于改进诸如List <Foo>之类的参数化类型的序列化。
引入ScriptTemplateView作为用于脚本化Web视图的基于JSR-223的机制，重点关注基于Nashorn的JavaScript视图模板（JDK 8）。

5.5 WebSocket Messaging Improvements

- Expose presence information about connected users and subscriptions:
    - new `SimpUserRegistry` exposed as a bean named "userRegistry".
    - sharing of presence information across cluster of servers (see broker relay config options).
- Resolve user destinations across cluster of servers (see broker relay config options).
- `StompSubProtocolErrorHandler` extension point to customize and control STOMP ERROR frames to clients.
- Global `@MessageExceptionHandler` methods via `@ControllerAdvice` components.
- Heart-beats and a SpEL expression 'selector' header for subscriptions with `SimpleBrokerMessageHandler`.
- STOMP client for use over TCP and WebSocket; see Section 26.4.15, “STOMP Client”.
- `@SendTo` and `@SendToUser` can contain destination variable placeholders.
- Jackson’s `@JsonView` supported for return values on `@MessageMapping` and `@SubscribeMapping` methods.
- `ListenableFuture` and `CompletableFuture` as return value types from `@MessageMapping` and `@SubscribeMapping` methods.
- `MarshallingMessageConverter` for XML payloads.

翻译：5.5 WebSocket消息传递改进
公开有关已连接用户和订阅的状态信息：

作为名为“ userRegistry”的bean公开的新SimpUserRegistry。
在服务器集群之间共享状态信息（请参阅代理中继配置选项）。
在服务器集群之间解析用户目标（请参阅代理中继配置选项）。
StompSubProtocolErrorHandler扩展点可自定义和控制客户端的STOMP ERROR帧。
通过@ControllerAdvice组件的全局@MessageExceptionHandler方法。

心跳和SpEL表达式“ selector”标头，用于使用SimpleBrokerMessageHandler进行订阅。
用于TCP和WebSocket的STOMP客户端；请参见第26.4.15节“ STOMP客户端”。
@SendTo和@SendToUser可以包含目标变量占位符。
支持Jackson的@JsonView，以获取@MessageMapping和@SubscribeMapping方法的返回值。
ListenableFuture和CompletableFuture作为@MessageMapping和@SubscribeMapping方法的返回值类型。
用于XML有效负载的MarshallingMessageConverter。

5.6 Testing Improvements

- JUnit-based integration tests can now be executed with JUnit rules instead of the `SpringJUnit4ClassRunner`. This allows Spring-based integration tests to be run with alternative runners like JUnit's `Parameterized` or third-party runners such as the `MockitoJUnitRunner`.
    - See the section called “Spring JUnit 4 Rules” for details.
- The Spring MVC Test framework now provides first-class support for HtmlUnit, including integration with Selenium's WebDriver, allowing for page-based web application testing without the need to deploy to a Servlet container.
    - See Section 15.6.2, “HtmlUnit Integration” for details.
- `AopTestUtils` is a new testing utility that allows developers to obtain a reference to the underlying target object hidden behind one or more Spring proxies.
    - See Section 14.2.1, “General testing utilities” for details.
- `ReflectionTestUtils` now supports setting and getting `static` fields, including constants.
- The original ordering of bean definition profiles declared via `@ActiveProfiles` is now retained in order to support use cases such as Spring Boot's `ConfigFileApplicationListener` which loads configuration files based on the names of active profiles.
- `@DirtiesContext` supports new `BEFORE_METHOD`, `BEFORE_CLASS`, and `BEFORE_EACH_TEST_METHOD` modes for closing the `ApplicationContext` *before* a test — for example, if some rogue (i.e., yet to be determined) test within a large test suite has corrupted the original configuration for the `ApplicationContext`.
- `@Commit` is a new annotation that may be used as a direct replacement for `@Rollback(false)`.
- `@Rollback` may now be used to configure class-level *default rollback* semantics.
    - Consequently, `@TransactionConfiguration` is now deprecated and will be removed in a subsequent release.
- `@Sql` now supports execution of *inlined SQL statements* via a new `statements` attribute.
- The `ContextCache` that is used for caching `ApplicationContext`s between tests is now a public API with a default implementation that can be replaced for custom caching needs.
- `DefaultTestContext`, `DefaultBootstrapContext`, and `DefaultCacheAwareContextLoaderDelegate` are now public classes in the `support` subpackage, allowing for custom extensions.
- `TestContextBootstrapper`s are now responsible for building the `TestContext`.
- In the Spring MVC Test framework, `MvcResult` details can now be logged at `DEBUG` level or written to a custom `OutputStream` or `Writer`. See the new `log()`, `print(OutputStream)`, and `print(Writer)` methods in `MockMvcResultHandlers` for details.
- The JDBC XML namespace supports a new `database-name` attribute in `<jdbc:embedded-database>`, allowing developers to set unique names for embedded databases — for example, via a SpEL expression or a property placeholder that is influenced by the current active bean definition profiles.
- Embedded databases can now be automatically assigned a unique name, allowing common test database configuration to be reused in different `ApplicationContext`s within a test suite.
    - See Section 19.8.6, “Generating unique names for embedded databases” for details.

- `MockHttpServletRequest` and `MockHttpServletResponse` now provide better support for date header formatting via the `getDateHeader` and `setDateHeader` methods.

翻译：5.6测试改进
现在，可以使用JUnit规则而不是SpringJUnit4ClassRunner执行基于JUnit的集成测试。这样，就可以与其他运行程序（例如JUnit的参数化运行程序）或第三方运行程序（例如MockitoJUnitRunner）一起运行基于Spring的集成测试。

有关详细信息，请参见“ Spring JUnit 4规则”一节。
Spring MVC Test框架现在提供对HtmlUnit的一流支持，包括与Selenium的WebDriver集成，从而允许进行基于页面的Web应用程序测试，而无需部署到Servlet容器。

有关详细信息，请参见第15.6.2节“ HtmlUnit集成”。
AopTestUtils是一种新的测试实用程序，允许开发人员获取对隐藏在一个或多个Spring代理之后的基础目标对象的引用。

有关详细信息，请参见第14.2.1节“常规测试实用程序”。
ReflectionTestUtils现在支持设置和获取静态字段，包括常量。
现在保留通过@ActiveProfiles声明的bean定义配置文件的原始顺序，以支持诸如Spring Boot的ConfigFileApplicationListener之类的用例，该用例基于活动配置文件的名称加载配置文件。
@DirtiesContext支持新的BEFORE_METHOD，BEFORE_CLASS和BEFORE_EACH_TEST_METHOD模式，用于在测试之前关闭ApplicationContext-例如，如果大型测试套件中的某些流氓（即，尚未确定）测试破坏了ApplicationContext的原始配置。
@Commit是一个新注释，可以用作@Rollback（false）的直接替换。
@Rollback现在可以用于配置类级别的默认回滚语义。

因此，@ TransactionConfiguration现在已弃用，并将在后续版本中删除。
@Sql现在支持通过新的statement属性执行内联SQL语句。
现在，用于在测试之间缓存ApplicationContext的ContextCache现在是具有默认实现的公共API，可以将其替换为自定义缓存需求。
现在，DefaultTestContext，DefaultBootstrapContext和DefaultCacheAwareContextLoaderDelegate是support子包中的公共类，允许自定义扩展。
现在，TestContextBootstrappers负责构建TestContext。
在Spring MVC测试框架中，现在可以在DEBUG级别记录MvcResult详细信息，或将其写入自定义OutputStream或Writer。有关详细信息，请参见MockMvcResultHandlers中的新log（），print（OutputStream）和print（Writer）方法。
JDBC XML名称空间在<jdbc：embedded-database>中支持新的数据库名称属性，从而使开发人员可以设置嵌入式数据库的唯一名称，例如，通过受当前活动bean影响的SpEL表达式或属性占位符定义配置文件。
现在可以自动为嵌入式数据库分配一个唯一的名称，从而允许在测试套件中的不同ApplicationContext中重用常见的测试数据库配置。

有关详细信息，请参见第19.8.6节“为嵌入式数据库生成唯一名称”。
现在，MockHttpServletRequest和MockHttpServletResponse通过getDateHeader和setDateHeader方法提供了对日期标头格式的更好支持。

6. New Features and Enhancements in Spring Framework 4.3

Version 4.3 included a number of improvements, as described in the following sections:

翻译：6. Spring Framework 4.3中的新功能和增强功能
4.3版包括许多改进，如以下各节所述：

第6.1节“核心容器改进”
第6.2节“数据访问改进”
第6.3节“缓存改进”
第6.4节“ JMS改进”
第6.5节“ Web改进”
第6.6节“ WebSocket消息传递改进”
第6.7节“测试改进”
第6.8节“对新一代库和服务器的支持”

6.1 Core Container Improvements

- Core container exceptions provide richer metadata to evaluate programmatically.
- Java 8 default methods get detected as bean property getters/setters.
- Lazy candidate beans are not being created in case of injecting a primary bean.
- It is no longer necessary to specify the `@Autowired` annotation if the target bean only defines one constructor.
- `@Configuration` classes support constructor injection.
- Any SpEL expression used to specify the `condition` of an `@EventListener` can now refer to beans (e.g. `@beanName.method()`).
- *Composed annotations* can now override array attributes in meta-annotations with a single element of the component type of the array. For example, the `String[] path` attribute of `@RequestMapping` can be overridden with `String path` in a composed annotation.
- `@PersistenceContext`/`@PersistenceUnit` selects a primary `EntityManagerFactory` bean if declared as such.
- `@Scheduled` and `@Schedules` may now be used as *meta-annotations* to create custom *composed annotations* with attribute overrides.
- `@Scheduled` is properly supported on beans of any scope.

翻译：6.1核心容器的改进
核心容器异常提供了更丰富的元数据，可以通过编程进行评估。
Java 8默认方法被检测为bean属性的获取器/设置器。
注入主bean时不会创建惰性候选bean。
如果目标bean仅定义一个构造函数，则不再需要指定@Autowired注释。
@Configuration类支持构造函数注入。
现在，用于指定@EventListener条件的任何SpEL表达式都可以引用bean（例如@ beanName.method（））。
现在，组合的批注可以使用数组的组件类型的单个元素覆盖元批注中的数组属性。例如，@ RequestMapping的String []路径属性可以用组合注释中的String路径覆盖。
如果声明为这样，@ PersistenceContext / @ PersistenceUnit选择一个主EntityManagerFactory bean。
@Scheduled和@Schedules现在可以用作元注释，以创建具有属性覆盖的自定义组合注释。
任何范围的bean都正确支持@Scheduled。

6.2 Data Access Improvements

- `jdbc:initialize-database` and `jdbc:embedded-database` support a configurable separator to be applied to each script.

翻译：6.2数据访问改进
jdbc：initialize-database和jdbc：embedded-database支持将可配置的分隔符应用于每个脚本。

6.3 Caching Improvements

Spring 4.3 allows concurrent calls on a given key to be synchronized so that the value is only computed once. This is an opt-in feature that should be enabled via the new `sync` attribute on `@Cacheable`. This features introduces a breaking change in the `Cache` interface as a `get(Object key, Callable<T> valueLoader)` method has been added.
Spring 4.3 also improves the caching abstraction as follows:

- SpEL expressions in caches-related annotations can now refer to beans (i.e. `@beanName.method()`).
- `ConcurrentMapCacheManager` and `ConcurrentMapCache` now support the serialization of cache entries via a new `storeByValue` attribute.
- `@Cacheable`, `@CacheEvict`, `@CachePut`, and `@Caching` may now be used as *meta-annotations* to create custom *composed annotations* with attribute overrides.

翻译：6.3缓存改进
Spring 4.3允许同步对给定键的并发调用，因此该值仅计算一次。 这是一项选择启用功能，应通过@Cacheable上的新同步属性启用。 由于已添加get（Object key，Callable <T> valueLoader）方法，此功能在Cache接口中引入了重大更改。

Spring 4.3还改进了缓存抽象，如下所示：

与缓存相关的注释中的SpEL表达式现在可以引用bean（即@ beanName.method（））。
ConcurrentMapCacheManager和ConcurrentMapCache现在通过新的storeByValue属性支持缓存条目的序列化。
@ Cacheable，@ CacheEvict，@ CachePut和@Caching现在可以用作元注释，以创建具有属性覆盖的自定义组合注释。

6.4 JMS Improvements

- `@SendTo` can now be specified at the class level to share a common reply destination.
- `@JmsListener` and `@JmsListeners` may now be used as *meta-annotations* to create custom *composed annotations* with attribute overrides.

翻译：6.4 JMS的改进
现在可以在类级别指定@SendTo，以共享公共回复目标。
@JmsListener和@JmsListeners现在可以用作元注释，以创建具有属性覆盖的自定义组合注释。

6.5 Web Improvements

- Built-in support for HTTP HEAD and HTTP OPTIONS.
- New `@GetMapping`, `@PostMapping`, `@PutMapping`, `@DeleteMapping`, and `@PatchMapping` *composed annotations* for `@RequestMapping`.
    - See Composed @RequestMapping Variants for details.
- New `@RequestScope`, `@SessionScope`, and `@ApplicationScope` *composed annotations* for web scopes.
    - See Request scope, Session scope, and Application scope for details.
- New `@RestControllerAdvice` annotation with combined `@ControllerAdvice` with `@ResponseBody` semantics.
- `@ResponseStatus` is now supported at the class level and inherited by all methods.
- New `@SessionAttribute` annotation for access to session attributes (see example).
- New `@RequestAttribute` annotation for access to request attributes (see example).
- `@ModelAttribute` allows preventing data binding via `binding=false` attribute (see reference).
- `@PathVariable` may be declared as optional (for use on `@ModelAttribute` methods).
- Consistent exposure of Errors and custom Throwables to MVC exception handlers.
- Consistent charset handling in HTTP message converters, including a UTF-8 default for multipart text content.

- Static resource handling uses the configured `ContentNegotiationManager` for media type determination.
- `RestTemplate` and `AsyncRestTemplate` support strict URI variable encoding via `DefaultUriTemplateHandler`.
- `AsyncRestTemplate` supports request interception.

翻译：6.5网站改进
对HTTP HEAD和HTTP OPTIONS的内置支持。
新的@ GetMapping，@ PostMapping，@ PutMapping，@ DeleteMapping和@PatchMapping组成了@RequestMapping的注释。

有关详细信息，请参见Composed @RequestMapping Variants。
新的@ RequestScope，@ SessionScope和@ApplicationScope组成了Web范围的批注。

有关详细信息，请参见请求范围，会话范围和应用程序范围。
新的@RestControllerAdvice批注结合了@ControllerAdvice和@ResponseBody语义。
现在，@ ResponseStatus在类级别受支持，并且被所有方法继承。
新的@SessionAttribute批注用于访问会话属性（请参见示例）。
新的@RequestAttribute批注用于访问请求属性（请参见示例）。
@ModelAttribute允许通过binding = false属性阻止数据绑定（请参阅参考资料）。
@PathVariable可以声明为可选的（用于@ModelAttribute方法）。
MVC异常处理程序对错误和自定义Throwables的一致暴露。
HTTP消息转换器中一致的字符集处理，包括多部分文本内容的UTF-8默认值。
静态资源处理使用配置的ContentNegotiationManager来确定媒体类型。
RestTemplate和AsyncRestTemplate通过DefaultUriTemplateHandler支持严格的URI变量编码。
AsyncRestTemplate支持请求拦截。

6.6 WebSocket Messaging Improvements

- `@SendTo` and `@SendToUser` can now be specified at class-level to share a common destination.

翻译：6.6 WebSocket消息传递改进
现在可以在类级别指定@SendTo和@SendToUser以共享公共目标。

6.7 Testing Improvements

- The JUnit support in the *Spring TestContext Framework* now requires JUnit 4.12 or higher.
- New `SpringRunner` *alias* for the `SpringJUnit4ClassRunner`.
- Test related annotations may now be declared on interfaces — for example, for use with *test interfaces* that make use of Java 8 based interface default methods.
- An empty declaration of `@ContextConfiguration` can now be completely omitted if default XML files, Groovy scripts, or `@Configuration` classes are detected.
- `@Transactional` test methods are no longer required to be `public` (e.g., in TestNG and JUnit 5).
- `@BeforeTransaction` and `@AfterTransaction` methods are no longer required to be `public` and may now be declared on Java 8 based interface default methods.
- The `ApplicationContext` cache in the *Spring TestContext Framework* is now bounded with a default maximum size of 32 and a *least recently used* eviction policy. The maximum size can be configured by setting a JVM system property or Spring property called `spring.test.context.cache.maxSize`.
- New `ContextCustomizer` API for customizing a test `ApplicationContext` *after* bean definitions have been loaded into the context but *before* the context has been refreshed. Customizers can be registered globally by third parties, foregoing the need to implement a custom `ContextLoader`.
- `@Sql` and `@SqlGroup` may now be used as *meta-annotations* to create custom *composed annotations* with attribute overrides.
- `ReflectionTestUtils` now automatically unwraps proxies when setting or getting a field.
- Server-side Spring MVC Test supports expectations on response headers with multiple values.

- Server-side Spring MVC Test parses form data request content and populates request parameters.
- Server-side Spring MVC Test supports mock-like assertions for invoked handler methods.
- Client-side REST test support allows indicating how many times a request is expected and whether the order of declaration for expectations should be ignored (see Section 15.6.3, “Client-Side REST Tests”).
- Client-side REST Test supports expectations for form data in the request body.

翻译：6.7测试改进
Spring TestContext Framework中的JUnit支持现在需要JUnit 4.12或更高版本。
SpringJUnit4ClassRunner的新SpringRunner别名。
现在可以在接口上声明与测试相关的注释，例如，供与使用基于Java 8的接口默认方法的测试接口一起使用。
如果检测到默认XML文件，Groovy脚本或@Configuration类，则可以完全省略@ContextConfiguration的空声明。
@Transactional测试方法不再需要公开（例如，在TestNG和JUnit 5中）。
@BeforeTransaction和@AfterTransaction方法不再需要公开，现在可以在基于Java 8的接口默认方法中声明。
现在，Spring TestContext Framework中的ApplicationContext缓存受默认最大大小32和最近最少使用的逐出策略限制。可以通过设置JVM系统属性或称为spring.test.context.cache.maxSize的Spring属性来配置最大大小。
新的ContextCustomizer API，用于在将bean定义加载到上下文中之后但在刷新上下文之前自定义测试ApplicationContext。定制程序可以由第三方进行全局注册，而无需实现自定义ContextLoader。
@Sql和@SqlGroup现在可以用作元注释，以创建具有属性覆盖的自定义组合注释。
现在，ReflectionTestUtils会在设置或获取字段时自动解开代理。
服务器端Spring MVC测试支持对具有多个值的响应头的期望。
服务器端Spring MVC Test解析表单数据请求内容并填充请求参数。
服务器端Spring MVC测试为调用的处理程序方法支持类似模拟的断言。
客户端REST测试支持可指示期望请求的次数以及是否应忽略期望的声明顺序（请参见第15.6.3节“客户端REST测试”）。
客户端REST测试支持对请求正文中的表单数据的期望。

6.8 Support for new library and server generations

- Hibernate ORM 5.2 (still supporting 4.2/4.3 and 5.0/5.1 as well, with 3.6 deprecated now)
- Hibernate Validator 5.3 (minimum remains at 4.3)
- Jackson 2.8 (minimum raised to Jackson 2.6+ as of Spring 4.3)
- OkHttp 3.x (still supporting OkHttp 2.x side by side)
- Tomcat 8.5 as well as 9.0 milestones
- Netty 4.1
- Undertow 1.4
- WildFly 10.1

Furthermore, Spring Framework 4.3 embeds the updated ASM 5.1, CGLIB 3.2.4, and Objenesis 2.4 in `spring-core.jar`.

翻译：

6.8 Support for new library and server generations

- Hibernate ORM 5.2 (still supporting 4.2/4.3 and 5.0/5.1 as well, with 3.6 deprecated now)
- Hibernate Validator 5.3 (minimum remains at 4.3)
- Jackson 2.8 (minimum raised to Jackson 2.6+ as of Spring 4.3)
- OkHttp 3.x (still supporting OkHttp 2.x side by side)
- Tomcat 8.5 as well as 9.0 milestones
- Netty 4.1
- Undertow 1.4
- WildFly 10.1

Furthermore, Spring Framework 4.3 embeds the updated ASM 5.1, CGLIB 3.2.4, and Objenesis 2.4 in `spring-core.jar`.