This document provides a summary of features and changes in Spring Framework 5.0, 5.1, 5.2, and 5.3. Also see the [[Spring Framework 5 FAQ]] for answers to common questions. Or back to [[Spring Framework Versions]].

What's New in Version 5.3

General Core Revision

- Upgrade to ASM 9.0 and Kotlin 1.4.
- Support for RxJava 3 in `ReactiveAdapterRegistry` while support for RxJava 1.x is deprecated.
- Improve GraalVM native support by removing unsupported features from native images.
- A `spring.spel.ignore` property to remove SpEL support for applications not using it.

Core Container

- Binding support for Java 14/15 record classes and similarly styled constructors/accessors.
- `ObjectProvider.ifAvailable/ifUnique` explicitly ignores beans from currently inactive scopes.
- `ApplicationListener.forPayload(Consumer)` method for convenient programmatic `PayloadApplicationEvent` handling.
- Support for Quartz extensions in `CronExpression`:
  - the day-of-month field can use `L` to express the last day of the month, `nL` to express the nth-to-last day of the month, or `nW` to express the nearest weekday to day-of-month n.
  - the day-of-week field can use `DDDL` to express the last day-of-week DDD in the month, or `DDD#n` to express the nth day-of-week DDD.

Data Access and Transactions

- New `spring-r2dbc` support module, moving core R2DBC support and the reactive `R2dbcTransactionManager` into the Spring Framework umbrella.
- New `JdbcTransactionManager` subclass of `DataSourceTransactionManager`, adding data access exception translation on commit.
- New `DataClassRowMapper` for constructor-based binding support, including Kotlin/Lombok data classes and Java 14/15 record classes.
- Support for `queryForStream` on `JdbcTemplate`, allowing for lazy iteration over a closeable `java.util.stream.Stream`.
- Configurable EntityManager/Session initializers on `Jpa/HibernateTransactionManager` and `Local(Container)EntityManagerFactoryBean`.
- `HibernateJpaVendorAdapter` exposes Hibernate ORM 5.2+ conventions by default (e.g. `SessionFactory` as EMF vendor interface).
- Transaction definitions may declare custom labels now (for use in custom transaction managers).

- Support for timeout values with `${...}` placeholders in transaction definitions.
- `TransactionalApplicationListener` interface with `forPayload` factory methods, callback support, and adapter classes for programmatic registration (as an alternative to `@TransactionalEventListener` annotated methods).
- Support for `@Transactional` suspending functions (Kotlin Coroutines)

Spring Messaging

- `RSocketRequester` support for the new `RSocketClient` as a result of which an `RSocketRequester` can be obtained as an instance, i.e. without a `Mono` wrapper or the need to connect first. A connection is transparently obtained as requests are made including support for reconnecting.
- `RSocketRequester` support for the new `LoadbalanceRSocketClient`.
- `RSocketRequester` support for metadataPush interactions.
- The `preservePublishOrder` option for STOMP/WebSocket applications now works in combination with send buffer size and time limits.
- Support for Kotlin multiplatform serialization (JSON only for now)

General Web Revision

- CORS configuration exposes a new `allowedOriginPatterns` property for declaring a dynamic range of domains via wildcard patterns.
- `RequestEntity` supports URI templates with variables.
- `Jackson2ObjectMapperBuilder` exposes `Consumer<ObjectMapper>` option for advanced customizations.
- `DataBinder` allows switching between direct field and bean property access during initialization. An example scenario is an `@ControllerAdvice` configuring direct field access by default globally with some controllers overriding that locally, via `@InitBinder` method, to bean property access.
- A `spring.xml.ignore` property to remove XML support for applications not using it, including related converters and codecs.

Spring MVC

- Efficient URL matching with parsed `PathPattern`'s in Spring MVC; see "URI Patterns" in the "Web Servlet" section of the documentation and blog post "URL Matching with PathPattern in Spring MVC".
- `UrlPathHelper` checks the `HttpServletMapping` (Servlet 4.0) for a more efficient determination of the application path, see #25100.
- `@ControllerAdvice` can handle exceptions from any handler type (i.e. not just `@Controller` but others like `HttpRequestHandler`, `HandlerFunction`, etc) as long as it matches the handler mappings set on `ExceptionHandlerExceptionResolver`.
- `@ExceptionHandler` can target exception causes at any level of nesting.

- `ForwardedHeaderFilter` updates the remote address/port from "Forwarded For" headers.
- Add missing beans to `WebMvcConfigurationSupport` in order to make `DispatcherServlet.properties` (now lazily parsed) not needed for most use cases.
- Support for Kotlin multiplatform serialization (JSON only for now)

Spring WebFlux

- New `DefaultPartHttpMessageReader` provides a fully reactive message reader that converts a buffer stream into a `Flux<Part>`
- New `PartHttpMessageWriter` to write the `Flux<Part>` received from a client to a remote service.
- New `WebClient` connector for Apache Http Components.
- `WebClient` and `ClientRequest` provide access to the `ClientHttpRequest` and the native request. This is useful for customizing per-request options specific to the HTTP library.
- `Encoder` and `Decoder` implementations for Netty `ByteBuf`.
- `ForwardedHeaderTransformer` updates the remote address/port from "Forwarded For" headers.
- `@EnableWebFlux` enables support for handlers of type `WebSocketHandler`.
- `WebSocketSession` provides access to the `CloseStatus`.
- `WebHttpHandlerBuilder` option to decorate the entire `WebFilter` chain at the level of the `HttpHandler`.
- More efficient direct path lookups for `@RequestMapping` methods that don't have any patterns or URI variables.
- `ClientResponse` performance optimizations and `mutate()` method for efficient changes through a client filter or `onStatus` handler, see #24680.
- Support for Kotlin multiplatform serialization (JSON only for now)

Testing

- The *Spring TestContext Framework* is now built and tested against JUnit Jupiter 5.7, JUnit 4.13.1, and TestNG 7.3.0.
- Test-related annotations on enclosing classes are now *inherited* by default for JUnit Jupiter `@Nested` test classes.
  - This is a potentially breaking change, but the behavior can be reverted to *override* configuration from enclosing classes via the `@NestedTestConfiguration` annotation, a JVM system property, or an entry in a `spring.properties` file in the root of the classpath.
  - Consult the Javadoc for `@NestedTestConfiguration` and the reference manual for details.
- The `spring.test.constructor.autowire.mode` property can now be set via a JUnit Platform configuration parameter to change the default `@TestConstructor` autowiring mode — for example, via the `junit-platform.properties` file.

- A `PlatformTransactionManager` configured via the `TransactionManagementConfigurer` API now takes precedence over any transaction manager configured as a bean in the `ApplicationContext` unless `@Transactional` is configured with a qualifier for the explicit transaction manager to use in tests.
- Test-managed transactions may now be disabled via `@Transactional(propagation = NEVER)` in addition to the existing support for `propagation = NOT_SUPPORTED` — for example, to override a `@Transactional` declaration from a composed annotation, on a superclass, etc.
- `WebTestClient` support for performing requests against `MockMvc`. This enables the possibility to use the same API for `MockMvc` tests and for full HTTP tests. See the updated section on testing in the reference documentation.
- `WebTestClient` has improved support for asserting all values of a header.
- Multipart data matchers in the client-side REST test support for the `RestTemplate`.
- HtmlUnit integration for Spring MVC Test supports file upload parameters.
- Minor enhancements to `MockHttpServletResponse` regarding character encoding and multiple `Content-Language` header values.
- Major revision of MockMVC Kotlin DSL to support multiple matchers

What's New in Version 5.2

General Core Revision

- Upgrade to ASM 7.1 and Kotlin 1.3.
- Annotation retrieval optimizations:
  - New `MergedAnnotations` API for efficient sophisticated annotation retrieval checks.
  - Candidate class mechanism for indications about the potential presence of certain annotation types.
- Commons Logging conveniences:
  - `LogMessage` for first-class message supplier and argument-based formatting support.
  - `LogAccessor` as a convenient `Log` alternative with out-of-the-box support for message suppliers.

Core Container

- `@Configuration` model improvements:
  - Optimized annotation introspection on configuration candidate classes.
  - `proxyBeanMethods` attribute for `@Configuration`-demarcated classes in lite mode, i.e. without CGLIB subclasses.
  - Support for annotation detection on factory methods with common `ListableBeanFactory` retrieval methods: `getBeanNamesForAnnotation`, `getBeansWithAnnotation`, `findAnnotationOnBean`.

- Bean registration with Kotlin DSL using callable reference with autowired parameters.

Transaction Management

- Support for reactive transaction management on Reactive Streams Publishers
  - `ReactiveTransactionManager` SPI as alternative to `PlatformTransactionManager`.
  - Programmatic `TransactionalOperator` as well as `@Transactional` integration.
- Support for transaction control via Vavr `Try` return type on `@Transactional` methods.

General Web Revision

- Complete set of `java.time` based setters on `HttpHeaders`, `CacheControl`, `CorsConfiguration`.
- `@RequestMapping` has enhanced produces condition support such that if a media type is declared with a specific parameter, and the requested media types (e.g. from "Accept" header) also has that parameter, the parameter values must match. This can be used for example to differentiate methods producing ATOM feeds `"application/atom+xml;type=feed"` vs ATOM entries `"application/atom+xml;type=entry"`.
- CORS revision that adds `Vary` header for non CORS requests on CORS enabled endpoints and avoid considering same-origin requests with an `Origin` header as a CORS request.
- Upgrade to Jackson 2.10

Spring Web MVC

- New "WebMvc.fn" programming model, analogous to the existing "WebFlux.fn":
  - A functional alternative to annotated controllers built on the Servlet API.
  - WebMvc.fn Kotlin DSL.
- Request mapping performance optimizations through caching of the lookup path per `HandlerMapping`, and pre-computing frequently used data in `RequestCondition` implementations.
- Improved, compact logging of request mappings on startup.

Spring WebFlux

- Refinements to `WebClient` API to make the `retrieve()` method useful for most common cases, specifically adding the ability to retrieve status and headers and addition to the body. The `exchange()` method is only for genuinely advanced cases, and when using it, applications can now rely

on `ClientResponse#createException` to simplify selective handling of exceptions.
- Configurable limits on input stream processing in all `Decoder` and `HttpMessageReader` implementations, with `maxInMemorySize` set to 256K by default. See WebFlux reference for details.
- Support for Kotlin Coroutines.
- Server and client now use Reactor checkpoints to insert information about the request URL being processed,sce or the handler used, that is then inserted into exceptions and logged below the exception stacktrace.
- Request mapping performance optimizations through pre-computing frequently used data in `RequestCondition` implementations.
- Header management performance optimizations by wrapping rather than copying server headers, and caching parsed representations of media types. Available from 5.1.1, see issue #21783 and commits under "Issue Links".
- Improved, compact logging of request mappings on startup.
- Add `ServerWebExchangeContextFilter` to expose the Reactor Context as an exchange attribute.
- Add FreeMarker macros support.
- `MultipartBodyBuilder` improvements to allow `Publisher` and `Part` as input along with option to specify the filename to use for a part.

Spring Messaging

- RSocket support:
  - Response handling via annotated `@MessageMapping` methods.
  - Performing requests via `RSocketRequester` with encoding and decoding to and from higher-level objects.
  - Support for Kotlin Coroutines.

Testing

- JUnit Jupiter 5.5.2 support.
- New `@TestConstructor` annotation and `spring.test.constructor.autowire.mode` JVM system property for configuring the autowiring mode for test constructors when using JUnit Jupiter.
- Support for built-in test execution events.
- `@TestPropertySource` can now be used as a repeatable annotation.
- Class-level and method-level `@Sql` declarations can now be merged.
- `@SqlConfig` now supports multiple comment prefixes for scripts configured via `@Sql`.
- Enhancements to the `TestContext` API:
  - New `hasApplicationContext()` method to determine if the application context for the current test is known to be available.
  - New `publishEvent()` method for simplified `ApplicationEvent` publication.
- Improved support for setting cookie headers in `MockHttpServletResponse`.

- `MockMvcResultMatchers.jsonPath()` now supports a target type.
- MockMvc Kotlin DSL
- New `headerDoesNotExist()` method in `MockRestServiceServer` to verify that a header does not exist.
- `ReflectionTestUtils` supports the invocation of `static` methods via new `invokeMethod()` variants.

Documentation

- Code samples in the reference documentation are now provided in Kotlin in addition to Java

To see all changes, please check the release notes for individual milestones: * 5.2 GA * 5.2 RC2 * 5.2 RC1 * 5.2 M3 * 5.2 M2 * 5.2 M1

What's New in Version 5.1

General Core Revision

- Infrastructure:
  - Warning-free support for JDK 11 on the classpath and the module path.
  - Support for Graal native image constraints (reflection, parameter names).
  - Upgrade to Reactor Core 3.2 and Reactor Netty 0.8 ("Reactor Californium").
  - Upgrade to ASM 7.0 and CGLIB 3.2.8.
- Core facilities:
  - NIO.2 Path support in FileSystemResource (superseding PathResource).
  - Performance improvements for core type and annotation resolution.
  - Consistent detection of method annotations on generic interfaces.
- Logging revision:
  - Spring's JCL bridge can be detected by standard Commons Logging.
  - Less noise on info, readable debug logs, details at trace level.

Core Container

- Bean definitions:
  - Support for logical and/or expressions in @Profile conditions.
  - Consistent (non-)detection of nested configuration classes.
  - Refined Kotlin beans DSL.
    - Unique implicit bean names for multiple beans of same type.
- Bean retrieval:
  - Consistent non-exposure of null beans in the BeanFactory API.
  - Programmatic ObjectProvider retrieval through the BeanFactory API.
  - ObjectProvider iterable/stream access for beans-of-type resolution.

    - Empty collection/map/array injection in single constructor scenarios.

General Web Revision

- Controller parameter annotations get detected on interfaces as well:
    - Allowing for complete mapping contracts in controller interfaces.
- Support for stricter encoding of URI variables in `UriComponentsBuilder`:
    - See updated "URI Encoding" in the reference.
- Servlet requests params with HTTP PUT, PATCH, and DELETE:
    - See "Form Data".

Spring Web MVC

- Logging
    - Improved, human-friendly, compact, DEBUG and TRACE logging.
    - Control over DEBUG logging of potentially sensitive data.
        * via `DispatcherServlet#enableLoggingRequestDetails`
- Updated web locale representation:
    - Language tag compliant by default.
    - CookieLocaleResolver sends RFC6265-compliant timezone cookies.
- Specific MVC exceptions for missing header, cookie, path variable:
    - Allowing for differentiated exception handling and status codes.
- Externally configured base path for sets of annotated controllers.
- Centralized handling of "forwarded" type headers via `ForwardedHeaderFilter`:
    - Please see important upgrade note.
- Support for serving Brotli, in addition to GZip, pre-encoded static resources.

Spring WebFlux

- HTTP/2 server support when running with Reactor Netty 0.8.
- Logging
    - Improved, human-friendly, compact, DEBUG and TRACE logging.
    - Correlated log messages for HTTP requests and WebSocket sessions.
    - Control over DEBUG logging of potentially sensitive data.
        * via `CodecConfigurer#defaultCodecs`
- Configurable limits on input stream processing in all `Decoder` and `HttpMessageReader` implementations, which by default are not set in 5.1 with the exception of `FormHttpMessageReader` which does limit input to 256K. Note that in 5.2 `maxInMemorySize` property for all codecs is set to 256K.
- Session cookies now have `SameSite=Lax` to protect against CSRF attacks:
    - See OWASP page and this article.
- Cookies are no longer adapted to cookie objects from the underlying server API, and are instead written to the `Set-Cookie` header directly because most servers don't support `sameSite`. This change includes validations to cookie attribute values that may differ slightly from similar validations

previously applied by the server. The validations however do conform to the syntax from RFC 6265, section 4.1. See #23693.
- DSL enhancements:
  - DSL-style builder for `RouterFunction` without static imports (sample).
  - Refined Kotlin router DSL.
- Externally configured base path for sets of annotated controllers.
- Third-party integration:
  - Support for Protobuf serialization, including message streaming.
  - `WebClient` connector for the Jetty reactive HTTP Client.
- WebSocket:
  - Support for `WebSocketSession` attributes.
  - Improve docs on reactive WebSocket API handling.
- Support for serving Brotli, in addition to GZip, pre-encoded static resources.

Spring Messaging

- Support for reactive clients in @MessageMapping methods:
  - Out-of-the-box support for Reactor and RxJava return values.
- Option to preserve publication order of messages by STOMP broker.
- `@SendTo` and `@SendToUser` can both be used on controller method.
- Improved docs on handling of messages and subscriptions.

Spring ORM

- Support for Hibernate ORM 5.3:
  - Bean container integration with Hibernate's new SPI.
- LocalSessionFactoryBean and HibernateTransactionManager support JPA interaction:
  - Allowing for native Hibernate as well as JPA access within the same transaction.
- Read-only transactions do not retain Hibernate entity snapshots in memory anymore:
  - Session.setDefaultReadOnly(true) by default.
- SAP HANA as a common JpaVendorAdapter database platform.

Testing

- Hamcrest and XML assertions in `WebTestClient`.
- `MockServerWebExchange` can be configured with fixed `WebSession`.

What's New in Version 5.0

JDK 8+ and Java EE 7+ Baseline

- Entire framework codebase based on Java 8 source code level now.

  - Improved readability through inferred generics, lambdas, etc.
  - Conditional support for Java 8 features now in straight code.
- Full compatibility with JDK 9 for development and deployment.
  - On classpath as well as module path (with stable automatic module names).
  - Framework build and test suite passes on JDK 9 (runs on JDK 8 by default).
- Java EE 7 API level required in Spring's corresponding features now.
  - Servlet 3.1, Bean Validation 1.1, JPA 2.1, JMS 2.0
  - Recent servers: e.g. Tomcat 8.5+, Jetty 9.4+, WildFly 10+
- Compatibility with Java EE 8 API level at runtime.
  - Servlet 4.0, Bean Validation 2.0, JPA 2.2, JSON Binding API 1.0
  - Tested against Tomcat 9.0, Hibernate Validator 6.0, Apache Johnzon 1.1

Removed Packages, Classes and Methods

- Package `beans.factory.access` (`BeanFactoryLocator` mechanism).
- Package `jdbc.support.nativejdbc` (`NativeJdbcExtractor` mechanism).
- Package `mock.staticmock` removed from `spring-aspects` module.
  - No support for `AnnotationDrivenStaticEntityMockingControl` anymore.
- Packages `web.view.tiles2` and `orm.hibernate3/hibernate4` dropped.
  - Minimum requirement: Tiles 3 and Hibernate 5 now.
- Dropped support: Portlet, Velocity, JasperReports, XMLBeans, JDO, Guava.
  - Recommendation: Stay on Spring Framework 4.3.x for those if needed.
- Many deprecated classes and methods removed across the codebase.
  - A few compromises made for commonly used methods in the ecosystem.

General Core Revision

- JDK 8+ enhancements:
  - Efficient method parameter access based on Java 8 reflection enhancements.
  - Selective declarations of Java 8 default methods in core Spring interfaces.
  - Consistent use of JDK 7 `Charset` and `StandardCharsets` enhancements.
- JDK 9 compatibility:
  - Avoiding JDK APIs which are deprecated in JDK 9 wherever possible.
  - Consistent instantiation via constructors (with revised exception handling).
  - Defensive use of reflection against core JDK classes.
- Non-null API declaration at the package level:

- Nullable arguments, fields and return values explicitly annotated with `@Nullable`.
  - Primarily for use with IntelliJ IDEA and Kotlin, but also Eclipse and FindBugs.
  - Some Spring APIs are not tolerating null values anymore (e.g. in `StringUtils`).
- `Resource` abstraction provides `isFile` indicator for defensive `getFile` access.
  - Also features NIO-based `readableChannel` accessor in the `Resource` interface.
  - File system access via NIO.2 streams (no `FileInput/OutputStream` used anymore).
- Spring Framework 5.0 comes with its own Commons Logging bridge out of the box:
  - `spring-jcl` instead of standard Commons Logging; still excludable/overridable.
  - Autodetecting Log4j 2.x, SLF4J, JUL (java.util.logging) without any extra bridges.
- `spring-core` comes with ASM 6.0 (next to CGLIB 3.2.5 and Objenesis 2.6).

Core Container

- Support for any `@Nullable` annotations as indicators for optional injection points.
- Functional style on `GenericApplicationContext/AnnotationConfigApplicationContext`
  - `Supplier`-based bean registration API with bean definition customizer callbacks.
- Consistent detection of transaction, caching, async annotations on interface methods.
  - In case of CGLIB proxies.
- XML configuration namespaces streamlined towards unversioned schemas.
  - Always resolved against latest `xsd` files; no support for deprecated features.
  - Version-specific declarations still supported but validated against latest schema.
- Support for candidate component index (as alternative to classpath scanning).

Spring Web MVC

- Full Servlet 3.1 signature support in Spring-provided `Filter` implementations.
- Support for Servlet 4.0 `PushBuilder` argument in Spring MVC controller methods.
- `MaxUploadSizeExceededException` for Servlet 3.0 multipart parsing on

common servers.
- Unified support for common media types through `MediaTypeFactory` delegate.
  - Superseding use of the Java Activation Framework.
- Data binding with immutable objects (Kotlin / Lombok / `@ConstructorProperties`)
- Support for the JSON Binding API (with Eclipse Yasson or Apache Johnzon as an alternative to Jackson and GSON).
- Support for Jackson 2.9.
- Support for Protobuf 3.
- Support for Reactor 3.1 `Flux` and `Mono` as well as RxJava 1.3 and 2.1 as return values from Spring MVC controller methods targeting use of the new reactive `WebClient` (see below) or Spring Data Reactive repositories in Spring MVC controllers.
- New `ParsingPathMatcher` alternative to `AntPathMatcher` with more efficient parsing and extended syntax.
- `@ExceptionHandler` methods allow `RedirectAttributes` arguments (and therefore flash attributes).
- Support for `ResponseStatusException` as a programmatic alternative to `@ResponseStatus`.
- Support script engines that do not implement `Invocable` via direct rendering of the script provided using `ScriptEngine#eval(String, Bindings)`, and also i18n and nested templates in `ScriptTemplateView` via the new `RenderingContext` parameter.
- Spring's FreeMarker macros (`spring.ftl`) use HTML output formatting now (requiring FreeMarker 2.3.24+).

Spring WebFlux

- New spring-webflux module, an alternative to `spring-webmvc` built on a reactive foundation – fully asynchronous and non-blocking, intended for use in an event-loop execution model vs traditional large thread pool with thread-per-request execution model.
- Reactive infrastructure in `spring-core` such as `Encoder` and `Decoder` for encoding and decoding streams of Objects; `DataBuffer` abstraction, e.g. for using Java `ByteBuffer` or Netty `ByteBuf`; `ReactiveAdapterRegistry` for transparent support of reactive libraries in controller method signatures.
- Reactive infrastructure in `spring-web` including `HttpMessageReader` and `HttpMessageWriter` that build on and delegate to `Encoder` and `Decoder`; server `HttpHandler` with adapters to (non-blocking) runtimes such as Servlet 3.1+ containers, Netty, and Undertow; `WebFilter`, `WebHandler` and other non-blocking contract alternatives to Servlet API equivalents.
- `@Controller` style, annotation-based, programming model, similar to Spring MVC, but supported in WebFlux, running on a reactive stack, e.g. capable of supporting reactive types as controller method arguments, never blocking on I/O, respecting backpressure all the way to the HTTP socket, and running on extra, non-Servlet containers such as Netty and

Undertow.
- New functional programming model ("WebFlux.fn") as an alternative to the `@Controller`, annotation-based, programming model – minimal and transparent with an endpoint routing API, running on the same reactive stack and WebFlux infrastructure.
- New `WebClient` with a functional and reactive API for HTTP calls, comparable to the `RestTemplate` but through a fluent API and also excelling in non-blocking and streaming scenarios based on WebFlux infrastructure; in 5.0 the `AsyncRestTemplate` is deprecated in favor of the `WebClient`.

Kotlin support

- Null-safe API when using Kotlin 1.1.50 or higher.
- Support for Kotlin immutable classes with optional parameters and default values.
- Functional bean definition Kotlin DSL.
- Functional routing Kotlin DSL for WebFlux.
- Leveraging Kotlin reified type parameters to avoid specifying explicitly the `Class` to use for serialization/deserialization in various APIs like `RestTemplate` or WebFlux APIs.
- Kotlin null-safety support for `@Autowired`/`@Inject` and `@RequestParam`/`@RequestHeader`/etc annotations in order to determine if an injection point or handler method parameter is required or not.
- Kotlin script support in `ScriptTemplateView` for both Spring MVC and Spring WebFlux.
- Array-like setters added to `Model`, `ModelMap` and `Environment`.
- Support for Kotlin autowired constructor with optional parameters.
- Kotlin reflection is used to determine interface method parameters.

Testing Improvements

- Complete support for JUnit 5's *Jupiter* programming and extension models in the Spring TestContext Framework.
  - `SpringExtension`: an implementation of multiple extension APIs from JUnit Jupiter that provides full support for the existing feature set of the Spring TestContext Framework. This support is enabled via `@ExtendWith(SpringExtension.class)`.
  - `@SpringJUnitConfig`: a composed annotation that combines `@ExtendWith(SpringExtension.class)` from JUnit Jupiter with `@ContextConfiguration` from the Spring TestContext Framework.
  - `@SpringJUnitWebConfig`: a composed annotation that combines `@ExtendWith(SpringExtension.class)` from JUnit Jupiter with `@ContextConfiguration` and `@WebAppConfiguration` from the Spring TestContext Framework.
  - `@EnabledIf`: signals that the annotated test class or test method is *enabled* if the supplied SpEL expression or property placeholder

evaluates to `true`.
  - `@DisabledIf`: signals that the annotated test class or test method is *disabled* if the supplied SpEL expression or property placeholder evaluates to `true`.
- Support for parallel test execution in the Spring TestContext Framework.
- New *before* and *after* test execution callbacks in the Spring TestContext Framework with support for TestNG, JUnit 5, and JUnit 4 via the `SpringRunner` (but not via JUnit 4 rules).
  - New `beforeTestExecution()` and `afterTestExecution()` callbacks in the `TestExecutionListener` API and `TestContextManager`.
- `MockHttpServletRequest` now has `getContentAsByteArray()` and `getContentAsString()` methods for accessing the content (i.e., request body).
- The `print()` and `log()` methods in Spring MVC Test now print the request body if the character encoding has been set in the mock request.
- The `redirectedUrl()` and `forwardedUrl()` methods in Spring MVC Test now support URI templates with variable expansion.
- XMLUnit support upgraded to 2.3.