What is Spring Framework 5?

Spring Framework 5 is a major version upgrade of the Spring Framework, several years in the making. It introduces a new non-blocking web framework called Spring WebFlux which uses Reactor to support the Reactive Streams API. It also offers a functional programming alternative to annotated controllers, first-class Kotlin language support, and first-class integration with JUnit 5. It is based on Java 8 language features, and designed to work with Java 9.

For more details see the reference documentation, or watch Spring Framework 5 Themes & Trends by Juergen Hoeller.

Does Spring Boot support Spring Framework 5?

Yes, the Spring Boot 2.x line is built on Spring Framework 5. Spring Boot 2.0 is expected to become generally available after Spring Framework 5.0, aiming for a timeframe around SpringOne Platform. Please use the latest Spring Boot 2.0 milestone for the time being, shipping with Spring Framework 5.0 already. For the latest release check the spring.io/spring-boot website or GitHub.

Why are we introducing Spring WebFlux?

Blocking threads consume resources. For latency-sensitive workloads which need to handle a large number of concurrent requests, the non-blocking async programming model is more efficient. This is particularly relevant for mobile applications and interconnected microservices, and generally for scenarios with many clients and uneven workloads.

The goal of Spring WebFlux is to offer Spring developers a non-blocking event-loop style programming model similar to node.js. For more details and demos, please watch Servlet vs Reactive Stacks in Five Use Cases and Reactive Web Applications with Spring Framework 5, both by Rossen Stoyanchev.

Can I keeping using Spring MVC with Spring Framework 5?

Absolutely, the Servlet-based Spring MVC remains included as a first-class choice alongside the new WebFlux framework. Upgrading to Spring Framework 5 does not require re-writing your applications to use Spring WebFlux. By all means, keep using Spring MVC if you are developing web apps that don't benefit from a non-blocking programming model, or that use blocking JPA or JDBC APIs for persistence (typically in combination with thread-bound transactions).

As a bonus, Spring MVC also supports reactive return types as an alternative to its DeferredResult feature now, allowing for asynchronous responses using the new WebClient or reactive datastore drivers even in traditional Servlet arrangements. However, please note that only a complete WebFlux stack delivers entirely non-blocking I/O within an event loop model.

What are Mono and Flux?

The WebFlux framework in Spring Framework 5 uses Reactor as its async foundation. This project provides the core types, Mono to represent a single async value, and Flux to represent a stream of async values. It also provides a library of operators for manipulating these values. For more information see the reactor-core and the reactive-streams-commons projects on GitHub.

How do I make all my code non-blocking?

For handlers to be fully non-blocking, you need to use reactive libraries throughout the processing chain, all the way to the persistence layer. Reactive Spring Data libraries are already available for Cassandra, MongoDB, Redis, and Couchbase. Please note that JPA and JDBC are inherently blocking APIs; we are still waiting for common ground around non-blocking relational database drivers.

Other Spring projects which are already reactive include Spring Security, Spring Vault, and Spring Cloud Stream.

What if there is no reactive library for my database?

One suggestion for handling a mix of blocking and non-blocking code would be to use the power of a microservice boundary to separate the blocking backend datastore code from the non blocking front-end API. Alternatively, you may also go with a worker thread pool for blocking operations, keeping the main event loop non-blocking that way.

What about interoperability with other reactive libraries like RxJava?

Reactor uses the same underlying Pub/Sub API as RxJava and other libraries built on Reactive Streams (a collaboration between engineers from Kaazing, Lightbend, Netflix, Pivotal, Red Hat, Twitter and many others).

Annotated handler methods in WebFlux (and to a limited degree even in Spring MVC) may declare RxJava 1.x or 2.x types as well, with the framework auto-adapting the corresponding Reactor publishers underneath the covers. This makes a lot of sense if you happen to have existing RxJava-based drivers and/or a preference for RxJava-based processing logic.

Can I still write controller methods with annotations?

Yes, both Spring MVC and Spring WebFlux support the same annotation-based programming model with @Controller, @RequestMapping, etc. Instead of taking ownership of the thread and and doing the work immediately, WebFlux controller methods operate on async Request and Response types.

Can I write controller methods without using annotations?

Yes, for developers who prefer to avoid the magic of annotations and reflection, Spring Framework 5 offers a new functional API to match routes with handler functions programmatically. This is made possible by the first class function support in Java 8, using lambdas or method references. For more information about how to compose handlers, see the blog post New in Spring Framework 5: Functional Web Framework by Arjen Poutsma.

Which HTTP servers will the WebFlux framework run on?

The WebFlux framework focuses on Tomcat and Jetty as well as Netty and Undertow ('undertow-core', no 'undertow-servlet' necessary). Since it is designed to support async programming, the framework never exposes the Servlet API - but it can adapt its Reactive Streams layer onto the non-blocking features of Servlet 3.1 containers underneath the covers (with first-class support for Tomcat and Jetty and best-effort adaptation to other Servlet containers at runtime).

What about HTTP client code?

The AsyncRestTemplate has been deprecated in favor of the new WebClient which provides a more fluent API, and is capable of both sync and async in one package. RestTemplate itself is not deprecated and there is nothing wrong with using it; the WebClient can be seen as its more modern successor

Will Spring Framework 5 work with Java 6 or Java 7?

No. Spring Framework 5 requires Java 8 or later. Please keep using Spring Framework 4.3 for Java 6/7 scenarios.

Does Spring Framework 5 work with the new Java 9 module system?

Yes, Spring Framework 5 ships with automatic module name entries in the manifests of our Spring Framework 5 jars. The public API surface of the Spring libraries remains unchanged.

Does Spring Framework 5 support Kotlin?

Yes, Spring Framework 5 officially supports Kotlin. More details are available in the Kotlin support documentation.
There are also two great blog posts about Kotlin support in Spring Framework 5: Introducing Kotlin support in Spring Framework 5 and Spring Framework 5 Kotlin APIs, the functional way, both by Sébastien Deleuze.

How do I upgrade to Spring Framework 5?

Please consult this wiki page for details on migrating to Spring Framework 5.

Where should I ask a technical question about Spring Framework 5?

Please post technical questions to StackOverflow, using tags like spring-webflux and project-reactor.

I want to contribute / get involved with Spring Framework 5, what should I do?

A great place to start might be our contributor guidelines