# Spring Framework Overview

Version 5.1.5.RELEASE

Spring makes it easy to create Java enterprise applications. It provides everything you need to embrace the Java language in an enterprise environment, with support for Groovy and Kotlin as alternative languages on the JVM, and with the flexibility to create many kinds of architectures depending on an application’s needs. As of Spring Framework 5.0, Spring requires JDK 8+ (Java SE 8+) and provides out-of-the-box support for JDK 9 already.

Spring将Java企业应用开发变得容易。它提供了在企业环境中使用Java语言所需的一切，并支持Groovy和Kotlin语言，以及根据应用程序的需要创建多种体系结构的灵活性。在SpringFramework5.0中，Spring需要JDK8以上(JavaSE 8以上)。

Spring supports a wide range of application scenarios. In a large enterprise, applications often exist for a long time and have to run on a JDK and application server whose upgrade cycle is beyond developer control. Others may run as a single jar with the server embedded, possibly in a cloud environment. Yet others may be standalone applications (such as batch or integration workloads) that do not need a server.

Spring支持广泛的应用场景. 在大型企业, 应用程序通常存在了很长一段时间,并且不得不运行在那些升级周期超出开发人员控制范围的服务器上. 其他的则可能以单个JAR形式运行并嵌入服务器, 例如在云环境中. 还有一些应用程序（如批处理或集成）可能不需要服务器.

Spring is open source. It has a large and active community that provides continuous feedback based on a diverse range of real-world use cases. This has helped Spring to successfully evolve over a very long time.

Spring是开源的. 它拥有一个庞大的活跃的社区，在基于各种真实世界用例上提供持续不断的反馈. 这已经帮助Spring在很长一段时间内不断进步.

## What We Mean by "Spring"

The term "Spring" means different things in different contexts. It can be used to refer to the Spring Framework project itself, which is where it all started. Over time, other Spring projects have been built on top of the Spring Framework. Most often, when people say "Spring", they mean the entire family of projects. This reference documentation focuses on the foundation: the Spring Framework itself.

Spring一词在不同的上下文中指的是不同的事物. 它可以是SpringFramework项目本身. 随着时间推移，其他Spring项目都建立在Spring框架之上. 大多数时候，人们提到Spring是指整个这一系列的项目. 这一篇参考文档则是基于SpringFramework框架本身.

The Spring Framework is divided into modules. Applications can choose which modules they need. At the heart are the modules of the core container, including a configuration model and a dependency injection mechanism. Beyond that, the Spring Framework provides foundational support for different application architectures, including messaging, transactional data and persistence, and web. It also includes the Servlet-based Spring MVC web framework and, in parallel, the Spring WebFlux reactive web framework.

Spring框架被分成了很多模块。应用程序可以根据需求自行选择。这其中核心容器最重要的，包含了配置模型和依赖注入机制。除此之外，Spring框架还为各种应用提供了一些基础支持，例如消息传递，事务性数据和持久化以及Web。当然也包括了以Servlet为基础的Spring MVC web框架和Spring WebFlux reactive web框架。

A note about modules: Spring’s framework jars allow for deployment to JDK 9’s module path ("Jigsaw"). For use in Jigsaw-enabled applications, the Spring Framework 5 jars come with "Automatic-Module-Name" manifest entries which define stable language-level module names ("spring.core", "spring.context" etc) independent from jar artifact names (the jars follow the same naming pattern with "-" instead of ".", e.g. "spring-core" and "spring-context"). Of course, Spring’s framework jars keep working fine on the classpath on both JDK 8 and 9.

关于模块的小贴士：Spring框架下的JAR包允许部署到JDK9的模块路径(“Jigsaw”)。对于启用了Jigsaw的应用程序来说//todo。当然，Spring框架在JDK8和JDK9下都运行良好，

## 2. History of Spring and the Spring Framework

Spring came into being in 2003 as a response to the complexity of the early [J2EE](https://en.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition) specifications. While some consider Java EE and Spring to be in competition, Spring is, in fact, complementary to Java EE. The Spring programming model does not embrace the Java EE platform specification; rather, it integrates with carefully selected individual specifications from the EE umbrella:

Spring于2003年诞生，为了解决早期J2EE规范的复杂性。虽然有些人认为JavaEE和Spring是竞争对手，但Spring实际上是JaveEE的补充。Spring并不包含JavaEE平台规范；相反，它从JavaEE中精心挑选了一些规范集成在一起，如下：

* Servlet API ([JSR 340](https://jcp.org/en/jsr/detail?id=340))
* WebSocket API ([JSR 356](https://www.jcp.org/en/jsr/detail?id=356))
* Concurrency Utilities ([JSR 236](https://www.jcp.org/en/jsr/detail?id=236))
* JSON Binding API ([JSR 367](https://jcp.org/en/jsr/detail?id=367))
* Bean Validation ([JSR 303](https://jcp.org/en/jsr/detail?id=303))
* JPA ([JSR 338](https://jcp.org/en/jsr/detail?id=338))
* JMS ([JSR 914](https://jcp.org/en/jsr/detail?id=914))
* as well as JTA/JCA setups for transaction coordination, if necessary.

The Spring Framework also supports the Dependency Injection ([JSR 330](https://www.jcp.org/en/jsr/detail?id=330)) and Common Annotations ([JSR 250](https://jcp.org/en/jsr/detail?id=250)) specifications, which application developers may choose to use instead of the Spring-specific mechanisms provided by the Spring Framework.

Spring框架还支持依赖注入和注解规范，应用开发人员可以选择这些来替代Spring框架所提供的。

As of Spring Framework 5.0, Spring requires the Java EE 7 level (e.g. Servlet 3.1+, JPA 2.1+) as a minimum - while at the same time providing out-of-the-box integration with newer APIs at the Java EE 8 level (e.g. Servlet 4.0, JSON Binding API) when encountered at runtime. This keeps Spring fully compatible with e.g. Tomcat 8 and 9, WebSphere 9, and JBoss EAP 7.

Spring框架5.0需要JavaEE 7（Servlet3.1+,JPA2.1+）以上版本支持，同时也集成了Java EE8的API（Servlet4.0,和Json Binding API）。这使得Spring与Tomcat 8和9,WebSphere 9,JBoss EAP7之间非常兼容。

Over time, the role of Java EE in application development has evolved. In the early days of Java EE and Spring, applications were created to be deployed to an application server. Today, with the help of Spring Boot, applications are created in a devops- and cloud-friendly way, with the Servlet container embedded and trivial to change. As of Spring Framework 5, a WebFlux application does not even use the Servlet API directly and can run on servers (such as Netty) that are not Servlet containers.

随着时间的流逝，Jave EE在应用开发中的角色已经演变了。在Java EE和Spring早期，应用程序主要用来部署在应用服务器上。现在，有了Spring Boot的帮助，应用程序以一种面向开发和云友好的方式创建的。对于Spring Framework 5来说，一个WebFlux应用程序并不需要Servlet API，它可以跑在非Servlet容器上（例如Netty)。

Spring continues to innovate and to evolve. Beyond the Spring Framework, there are other projects, such as Spring Boot, Spring Security, Spring Data, Spring Cloud, Spring Batch, among others. It’s important to remember that each project has its own source code repository, issue tracker, and release cadence. See [spring.io/projects](https://spring.io/projects) for the complete list of Spring projects.

Spring在持续创新和发展。除了Spring Framework之外，这还有Spring Boot,Spring Security,Spring Data,Spring Cloud,Spring Batch。重要的是这些项目都有自己的源代码仓库，问题追踪器和版本发布。可以访问[spring.io/projects](https://spring.io/projects) 查看Spring项目一览表。

3. Design Philosophy

When you learn about a framework, it’s important to know not only what it does but what principles it follows. Here are the guiding principles of the Spring Framework:

当你学习一门框架的时候，重要的不仅是了解它做了些什么，而且要知道为啥这样做。以下是Spring框架所遵循的原则：

* Provide choice at every level. Spring lets you defer design decisions as late as possible. For example, you can switch persistence providers through configuration without changing your code. The same is true for many other infrastructure concerns and integration with third-party APIs.
* 在各级提供选择。Spring会允许你尽可能得晚推迟设计决策。例如，你可以在不更改代码得情况下通过配置切换持久性提供程序。其他许多基础设置问题以及与第三方API得集成也是如此
* Accommodate diverse perspectives. Spring embraces flexibility and is not opinionated about how things should be done. It supports a wide range of application needs with different perspectives.
* 适应不同得观点，Spring拥抱灵活性并不固执得认为这事该应该怎么做。它可以从各种方面去支持各式各样得应用系统。
* Maintain strong backward compatibility. Spring’s evolution has been carefully managed to force few breaking changes between versions. Spring supports a carefully chosen range of JDK versions and third-party libraries to facilitate maintenance of applications and libraries that depend on Spring.
* 保持向后兼容性//todo
* Care about API design. The Spring team puts a lot of thought and time into making APIs that are intuitive and that hold up across many versions and many years.
* 关心API设计。Spring团队在API的设计上投入了大量的精力和时间，希望API是直观容易理解的。
* Set high standards for code quality. The Spring Framework puts a strong emphasis on meaningful, current, and accurate javadoc. It is one of very few projects that can claim clean code structure with no circular dependencies between packages.
* 对代码质量设置高标准。Spring Framework强调有意义，注重当前并且精准的Java文档。它是少数几个能够声明干净的代码结构，没有包之间的循环依赖的项目之一。

## 4. Feedback and Contributions

For how-to questions or diagnosing or debugging issues, we suggest using StackOverflow, and we have a [questions page](https://spring.io/questions) that lists the suggested tags to use. If you’re fairly certain that there is a problem in the Spring Framework or would like to suggest a feature, please use the [GitHub Issues](https://github.com/spring-projects/spring-framework/issues).

关于去哪提问，如何诊断或者调试问题，我们建议使用StackOverflow,我们也有问题页面，列出了建议使用的标签。如果你相当肯定Srping框架有有问题，或者建议一个特性，你可以使用GitHub Issues。

If you have a solution in mind or a suggested fix, you can submit a pull request on [Github](https://github.com/spring-projects/spring-framework). However, please keep in mind that, for all but the most trivial issues, we expect a ticket to be filed in the issue tracker, where discussions take place and leave a record for future reference.

如果你有解决办法或者建议性的修改，你可以GitHub上提交。//todo

For more details see the guidelines at the [CONTRIBUTING](https://github.com/spring-projects/spring-framework/blob/master/CONTRIBUTING.md), top-level project page.

关于跟多的细节请看Contributing上的指导。

## 5. Getting Started

If you are just getting started with Spring, you may want to begin using the Spring Framework by creating a [Spring Boot](https://projects.spring.io/spring-boot/)-based application. Spring Boot provides a quick (and opinionated) way to create a production-ready Spring-based application. It is based on the Spring Framework, favors convention over configuration, and is designed to get you up and running as quickly as possible.

如果你是刚刚接触Spring，你最好是创建一个SpringBoot的应用。SpringBoot能够快速创建基于Spring的应用程序。它配置简便，能够让你快速的搭起一个应用程序。

You can use [start.spring.io](https://start.spring.io/) to generate a basic project or follow one of the ["Getting Started" guides](https://spring.io/guides), such as [Getting Started Building a RESTful Web Service](https://spring.io/guides/gs/rest-service/). As well as being easier to digest, these guides are very task focused, and most of them are based on Spring Boot. They also cover other projects from the Spring portfolio that you might want to consider when solving a particular problem.

你可以使用start.spring.io来生成一个基础工程，或者跟着”Getting Started”来尝试。这些指导都非常好理解，也有明确的目标，大多数是基于SpringBoot。他们当然也有覆盖Spring家族其他项目，你在解决特定问题的时候会考虑到这些项目。