04 用思维导图拆解Spring Framework框架，让您事半功倍！

更新时间：2020-08-05 11:10:20

> 合理安排时间，就等于节约时间。——培根

背景

阅读源码绝对算得上是一件费时费力的工作，需要读者耗费大量的时间去完成。而作为开发人员，毕竟精力有限，实在没办法拿出太多的时间放在源码的阅读上。源码的复杂性。任何一款源码经历了多年的发展与提炼，其复杂程度可想而知。

当我们阅读源码的时候，大家都知道需要通过工具来跟踪代码的运行，如 SourceInsigh、IDEA、Eclipse、STS 等，进而去分析程序。

但是，当代码过于复杂，环环相扣绕来绕去的时候，跟进了几十个甚至几百个类和方法时，这时我们已经不知道自己所处的位置了，不得不再重来，但是一次又一次地，最终发现自己根本无法驾驭它，不得不放弃。有些源码发展多年，会遇到各种各样的问题，并对问题进行了解决，而其中有些问题对于我们来说甚至可以用莫名其妙来修饰，有时候根本想不出会在什么情况下发生。我们查阅各种资料，查询无果后，会失去耐心，最终放弃。

无论基于什么样的原因，放弃阅读源码始终不是一个明智的选择，因为你失去了一个跟大师学习的机会。而且，当你读过几个源码之后就会发现，它们的思想以及实现方式是相通的。这就是开源的好处。随着各种开源软件的发展，各家都会融合别家优秀之处来不断完善自己，这样，到最后的结果就是所有的开源软件从设计上或者实现上都会变得越来越相似，也就是说当你读完某个优秀源码后再去读另一个源代码，阅读速度会有很大提升。

那为什么选择 Spring 的源码来作为阅读的首选呢？

广泛性： Spring 使用的广泛性，使面试时，Spring 问题成为必选之一，而通过阅读代码理解 Spring 内部原理成为您脱颖而用的重要利器；

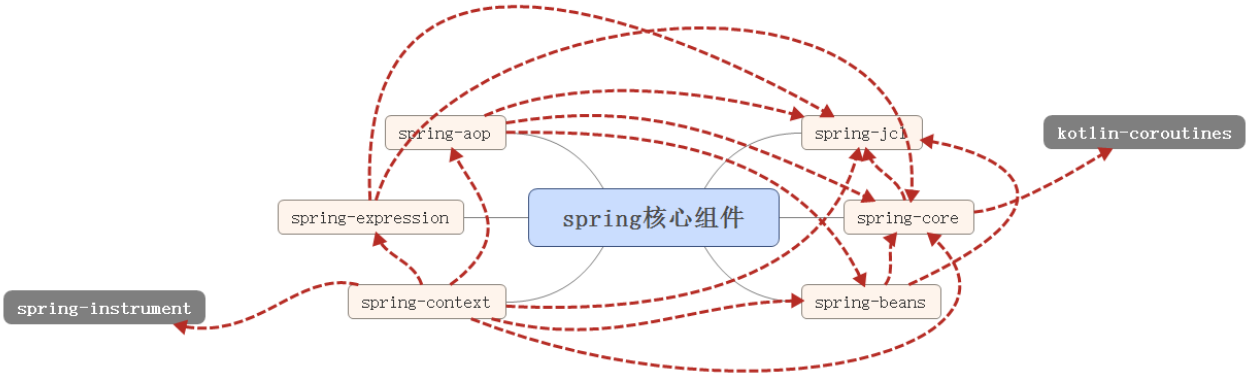
高质量代码： Spring 秉承的理念之一：高质量代码，使我们阅读代码的难度降低，适合当开发者写代码的范本。

> 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.

阅读代码，最好对整个 Spring 框架有全面的认识，可以抓住主要的，想要深入的代码，重点攻关，下面这篇文章将 Spring 各个模块直接的依赖管理整理出思维导图，并抽象出两大部分，分别是核心基础组件和高级应用组件，帮忙你更好的掌握 Spring 框架本身。

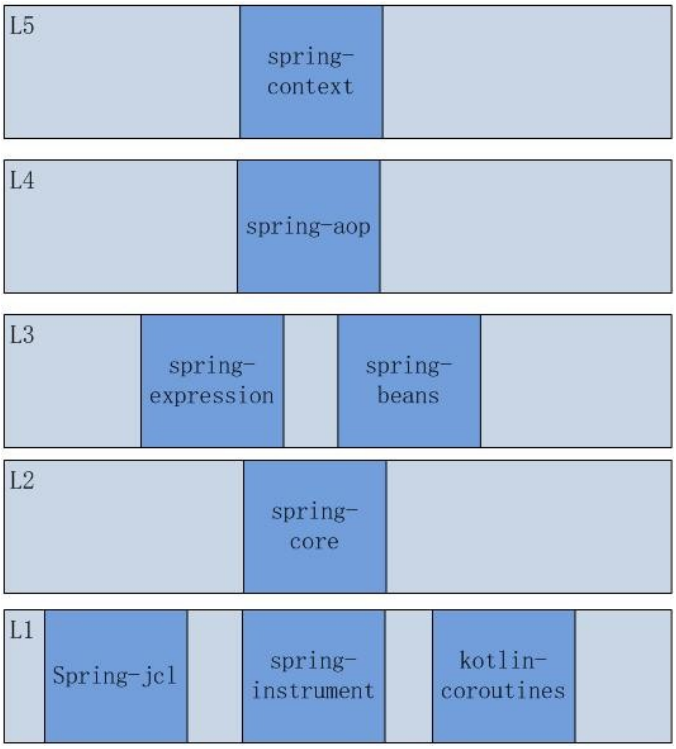
核心基础组件

Spring 基础组件包含六个，相应的关系如下图：



从箭头的流向可以归纳出从底到上的分层结构：

- spring-jcl、spring-instrument、kotlin-coroutines 是底层基础，不依赖 Spring；
- 其它组件，可以标记为 L1；
- spring-core 依赖于 spring-jcl、kotlin-coroutines 而又不依赖其它 Spring 组件，可以标记为 L2；
- spring-expression、spring-beans 依赖于 spring-jcl 和 spring-core 又不依赖其它 Spring 组件，可以标记为 L3；
- spring-aop 依赖于 spring-jcl、spring-core、spring-beans 而又不依赖其它 Spring 组件，可以标记为 L4；
- spring-context 依赖于 spring-jcl、spring-core、spring-expression、spring-beans、spring-aop，可标记为L5。



Spring-jcl： JCL 全称：Jakarta Commons Logging，Spring-jcl 采用了设计模式中的“适配器模式”，它对外提供统一的接口，然后在适配类中将对日志的操作委托给具体的日志框架；

Spring-core： Core 模块主要的功能是实现了反向控制 IOC（Inversion of Control）与依赖注入 DI（Dependency Injection）、Bean 配置以及加载。Core 模块中有 Beans、BeanFactory、BeanDefinitions、ApplicationContext 等几个重要概念；

Spring-expression： Spring 表达式语言，解析 Spring 表达式语言；

Spring-beans： 负责 Bean 工厂中 Bean 的装配，所谓 Bean 工厂即是创建对象的工厂，Bean 的装配也就是对象的创建工作。**重点：BeanFactory**；

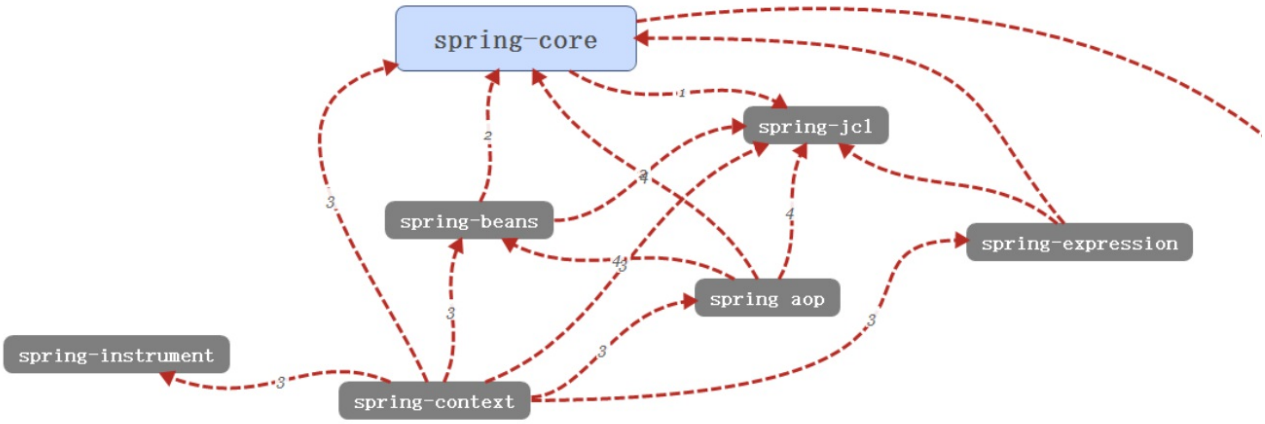
Spring-aop： Spring 提供了面向切面功能的模块；

Spring-context： Spring 的 IOC 容器，因大量调用 Spring Core 中的函数，整合了 Spring 的大部分功能。Bean 创建好对象后，由 Context 负责建立 Bean 与 Bean 之间的关系并维护。所以也可以把 Context 看成是 Bean 关系的集合，**重点：ApplicationContext**。

使用的其它组件：

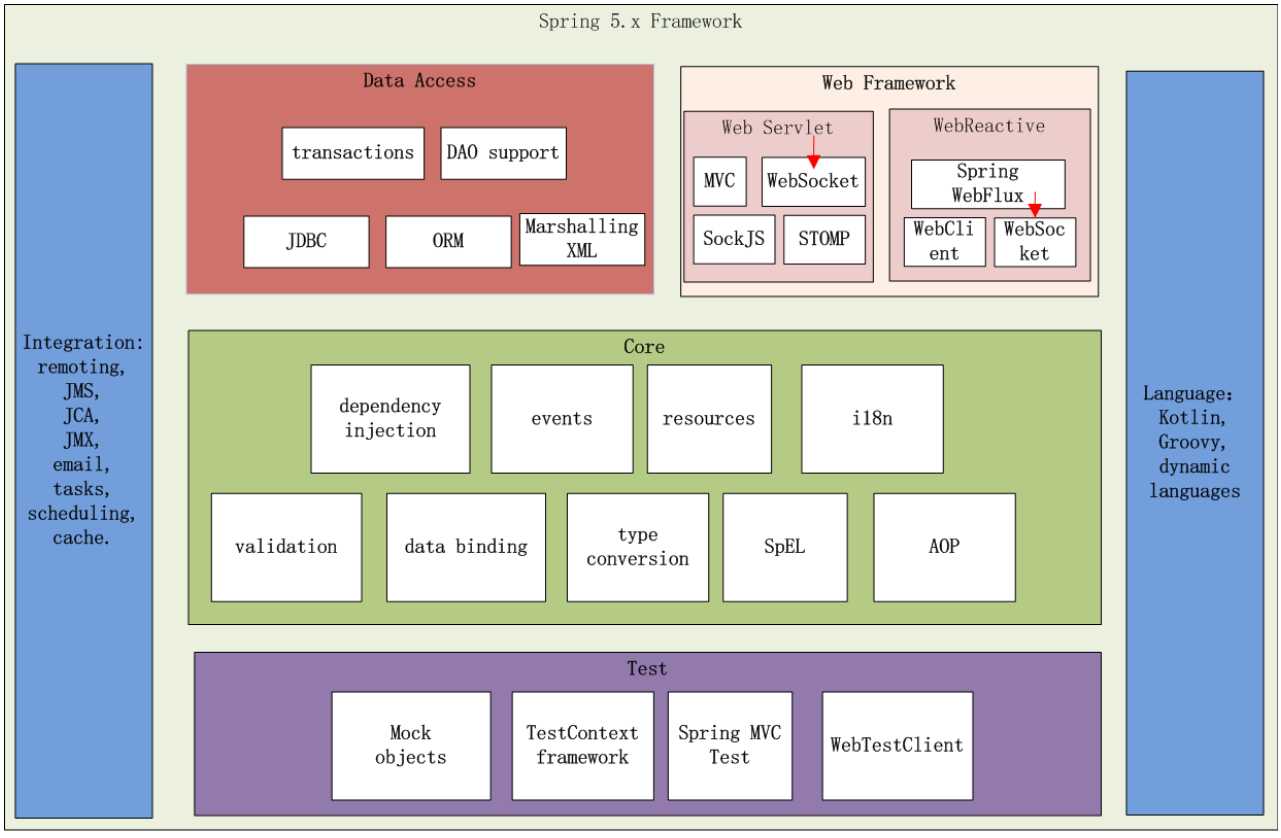
- **Spring-instrument**： 相当于一个检测器，提供对 JVM 以及对 Tomcat 的检测；
- **kotlin-coroutines**： 引入了协程。

为了方便查看，我们去掉了 Spring 核心组件的描述，仅仅描述核心组件之间的关系，如下图所示：



高级应用组件

本文根据 Spring5.x 官方最新文档画出的 Spring 的架构图如下：



可以看到 Spring 核心组件 core 位于所有组件的核心，它包含了 DI，事件，资源，国际化，验证，数据绑定，类型转换，SpEL，AOP 等主要功能，基于核心应用组件之上的组件有：数据访问（data access）、Web 框架、Spring 还支持动态脚本语言如 Groovy，也提供了 Kotlin 的支持、及其它各种框架及组件如调度，邮件，缓存，远程调用等的支持集成。最后，Spring 还提供了方便的测试支持 spring-test。

下面对各组件做一下详细介绍。

1. web 框架

- **Spring-web：** Web 上下文模块建立在应用程序上下文模块之上，为基于 Web 的应用程序提供了上下文；

- **Spring-webmvc：** 主要用于在遵守 Servlet 规范的前提下，将 Spring 框架集成到 Java web 应用中；
- **Spring-websocket**：Spring 在 4.0 后将 WebSocket 集成了进去，即 Spring-websocket 模块。它与 Java WebSocket API 标准（JSR-356）兼容，并且还提供额外功能；
- **Spring-webflux：** Spring Framework 5.0 中引入的新的反应式 Web 框架。 与 Spring MVC 不同，它不需要 Servlet API，完全异步和非阻塞， 并通过 Reactor 项目实现 Reactive Streams 规范。 并且可以在诸如 Netty， Undertow 和 Servlet 3.1+ 容器的服务器上运行。

2. 数据访问

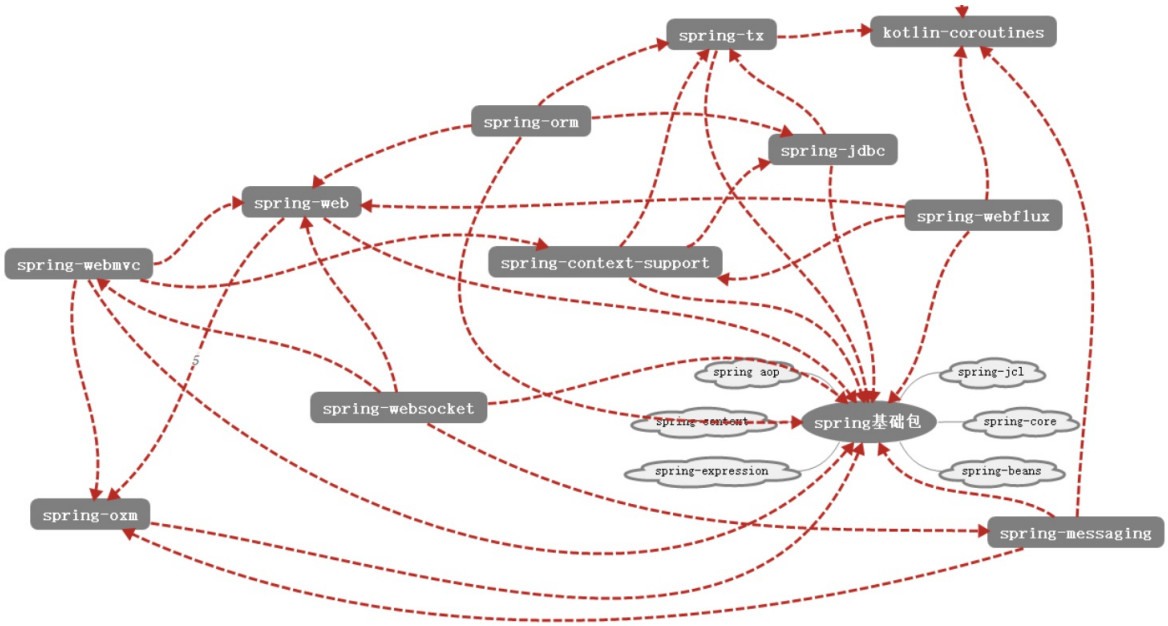
- **Transactions：** 事务管理
- **DAO Support：** 数据访问对象
- **JDBC：** jdbcTemplate
- **O/R Mapping：** jpa，hibernate
- **XML Marshalling：** xml 读取

3.消息框架

Spring-messaging

高级组件的关系

我们看一下他们之间的相互关系，完整的依赖图如下所示：



从图中可以看出来各组件以下依赖关系：

spring-web 依赖于 Spring 核心组件和 spring-oxm；

spring-webmvc 依赖于 Spring核心组件和 spring-web，spring-context-support，spring-oxm；

spring-websocket 依赖于 Spring 核心组件和 spring message，spring-web，spring-webmvc；

spring-webflux 依赖于 Spring 核心组件和 spring-context-support，kotlin-coroutines。

总结

Spring 框架本身包含的内容比较多，盲目的去阅读其源代码，只会陷入代码的沼泽中；源码 Spring 源码讲解战略战术：从战略上搞清楚 Spring 的各个组件及应用场景，然后从战术上根据自己的弱点进行加强练习，才可以练就一身神功。官方文档是我们了解 Spring 框架组件及应用场景的最好来源，通过使用思维导图来将官网文档进行浓缩精华，是一种值得推荐的方式。你瞧！你是否对 Spring 框架有了总体的认识了？

}

03 深入Spring5 的原理或者源码的道和术

05 Spring 5.x特性、设计理念及架构