课程目标

1、通过对本章内容的学习，可以掌握 Spring 的基本架构及各子模块之间的依赖关系。

2、了解 Spirng 的发展历史，启发思维。

3、对 Spring 形成一个整体的认识，为之后的深入学习做铺垫。

4、通过对本章内容的学习，可以了解 Spring 版本升级的规律，从而应用到自己的系统升级版本命名。

5、采用 Gradle 构建 Spring5 源码。

内容定位

Spring 使用经验 1-5 年，希望深入了解 Spring 源码的人群。

Spring 的前世今生

相信经历过不使用框架开发 Web 项目的 70 后、80 后都会有如此感触，如今的程序员开发项目太轻松了，基本只需要关心业务如何实现，通用技术问题只需要集成框架便可。早在 2007 年，一个基于 Java 语言的开源框架正式发布，取了一个非常有活力且美好的名字，叫做 Spring。它是一个开源的轻量级 Java SE（Java 标准版本）/Java EE（Java 企业版本）开发应用框架，其目的是用于简化企业级应用程序开发。应用程序是由一组相互协作的对象组成。而在传统应用程序开发中，一个完整的应用是由一组相互协作的对象组成。所以开发一个应用除了要开发业务逻辑之外，最多的是关注如何使这些对象协作来完成所需功能，而且要低耦合、高聚合。业务逻辑开发是不可避免的，那如果有个框架出来帮我们来创建对象及管理这些对象之间的依赖关系。可能有人说了，比如“抽象工厂、工厂方法模式”不也可以

帮我们创建对象，“生成器模式”帮我们处理对象间的依赖关系，不也能完成这些功能吗？可是这些又需要我们创建另一些工厂类、生成器类，我们又要而外管理这些类，增加了我们的负担，如果能有种通过配置方式来创建对象，管理对象之间依赖关系，我们不需要通过工厂和生成器来创建及管理对象之间的依赖关系，这样我们是不是减少了许多工作，加速了开发，能节省出很多时间来干其他事。Spring 框架刚出来时主要就是来完成这个功能。

Spring 框架除了帮我们管理对象及其依赖关系，还提供像通用日志记录、性能统计、安全控制、异常处理等面向切面的能力，还能帮我管理最头疼的数据库事务，本身提供了一套简单的 JDBC 访问实现，提供与第三方数据访问框架集成（如 Hibernate、JPA），与各种 Java EE 技术整合（如 Java Mail、任务调度等等）,提供一套自己的 Web 层框架 Spring MVC、而且还能非常简单的与第三方 Web 框架集成。从这里我们可以认为 Spring 是一个超级粘合大平台，除了自己提供功能外，还提供粘合其他技术和框架的能力，从而使我们可以更自由的选择到底使用什么技术进行开发。而且不管是 JAVA SE（C/S 架构）应用程序还是 JAVA EE（B/S 架构）应用程序都可以使用这个平台进行开发。如今的 Spring 已经不再是一个框架，早已成为了一种生态。SpringBoot 的便捷式开发实现了零配置，SpringCloud 全家桶，提供了非常方便的解决方案。接下来，让我们来深入探讨 Spring 到底能给我们带来什么？

一切从 Bean 开始

说到 Bean 这个概念，还得从 Java 的起源说起。早在 1996 年，Java 还只是一个新兴的、初出茅庐的编程语言。人们之所以关注她仅仅是因为，可以使用 Java 的 Applet 来开发 Web 应用，作为浏览器组件。但开发者们很快就发现这个新兴的语言还能做更多的事情。与之前的所有语言不同，Java 让模块化构建复杂的系统成为可能（当时的软件行业虽然在业务上突飞猛进，但当时开发用的是传统的面向过程开发思想，软件的开发效率一直踟蹰不前。伴随着业务复杂性的不断加深，开发也变得越发困难。其实，当时也是 OOP 思想飞速发展的时期，她在 80 年代末被提出，成熟于 90 年代，现今大多数编程语言都

已经是面向对象的）。

同年 12 月，Sun 公司发布了当时还名不见经传但后来人尽皆知的 JavaBean 1.00-A 规范。早期的 JavaBean 规范针对于 Java，她定义了软件组件模型。这个规范规定了一整套编码策略，使简单的 Java 对象不仅可以被重用，而且还可以轻松地构建更为复杂的应用。尽管 JavaBean 最初是为重用应用组件而设计的，但当时他们却是主要用作构建窗体控件，毕竟在 PC 时代那才是主流。但相比于当时正如日中天的 Delphi、VB 和 C++，它看起来还是太简易了，以至于无法胜任任何"实际的"工作需要。

复杂的应用通常需要事务、安全、分布式等服务的支持，但 JavaBean 并未直接提供。所以到了 1998 年 3 月，Sun 公司发布了 EJB 1.0 规范，该规范把 Java 组件的设计理念延伸到了服务器端，并提供了许多必须的企业级服务，但他也不再像早期的 JavaBean 那么简单了。实际上，除了名字叫 EJB Bean 以外，其他的和 JavaBean 关系不大了。

尽管现实中有很多系统是基于 EJB 构建的，但 EJB 从来没有实现它最初的设想：简化开发。EJB 的声明式编程模型的确简化了很多基础架构层面的开发，例如事务和安全；但另一方面 EJB 在部署描述符和配套代码实现等方面变得异常复杂。随着时间的推移，很多开发者对 EJB 已经不再抱有幻想，开始寻求更简洁的方法。

现在 Java 组件开发理念重新回归正轨。新的编程技术 AOP 和 DI 的不断出现，他们为 JavaBean 提供了之前 EJB 才能拥有的强大功能。这些技术为 POJO 提供了类似 EJB 的声明式编程模型，而没有引入任何 EJB 的复杂性。当简单的 JavaBean 足以胜任时，人们便不愿编写笨重的 EJB 组件了。

客观地讲，EJB 的发展甚至促进了基于 POJO 的编程模型。引入新的理念，最新的 EJB 规范相比之前的规范有了前所未有的简化，但对很多开发者而言，这一切的一切都来得太迟了。到了 EJB 3 规范发布时，其他基于 POJO 的开发架构已经成为事实的标准了，而 Spring 框架也就是在这样的大环境下出现的。

Spring 的设计初衷

Spring 是为解决企业级应用开发的复杂性而设计，她可以做很多事。但归根到底支撑 Spring 的仅仅是少许的基本理念，而所有的这些基本理念都能可以追溯到一个最根本的使命：简化开发。这是一个郑重的承诺，其实许多框架都声称在某些方面做了简化。而 Spring 则立志于全方面的简化 Java 开发。对此，她主要采取了 4 个关键策略：

1、基于 POJO 的轻量级和最小侵入性编程；

2、通过依赖注入和面向接口松耦合；

3、基于切面和惯性进行声明式编程；

4、通过切面和模板减少样板式代码；

而他主要是通过：面向 Bean(BOP)、依赖注入（DI）以及面向切面（AOP）这三种方式来达成的。

BOP 编程伊始

Spring 是面向 Bean 的编程（Bean Oriented Programming, BOP），Bean 在 Spring 中才是真正的主角。Bean 在 Spring 中作用就像 Object 对 OOP 的意义一样，Spring 中没有 Bean 也就没有 Spring 存在的意义。Spring 提供了 IOC 容器通过配置文件或者注解的方式来管理对象之间的依赖关系。

控制反转(其中最常见的实现方式叫做依赖注入（Dependency Injection，DI），还有一种方式叫“依赖查找”（Dependency Lookup，DL），她在 C++、Java、PHP 以及.NET 中都运用。在最早的 Spring 中是包含有依赖注入方法和依赖查询的，但因为依赖查询使用频率过低，不久就被 Spring 移除了，所以在 Spring 中控制反转也被直接称作依赖注入)，她的基本概念是：不创建对象，但是描述创建它们的方式。在代码中不直接与对象和服务连接，但在配置文件中描述哪一个组件需要哪一项服务。容器 （在 Spring 框架中是 IOC 容器）负责将这些联系在一起。

在典型的 IOC 场景中，容器创建了所有对象，并设置必要的属性将它们连接在一起，决定什么时间

调用方法。

依赖注入的基本概念

Spring 设计的核心 org.springframework.beans 包（架构核心是 org.springframework.core 包），它的设计目标是与 JavaBean 组件一起使用。这个包通常不是由用户直接使用，而是由服务器将其用作其他多数功能的底层中介。下一个最高级抽象是 BeanFactory 接口，它是工厂设计模式的实现，允许通过名称创建和检索对象。BeanFactory 也可以管理对象之间的关系。

BeanFactory 最底层支持两个对象模型。

1，单例：提供了具有特定名称的全局共享实例对象，可以在查询时对其进行检索。Singleton 是默认的也是最常用的对象模型。

2，原型：确保每次检索都会创建单独的实例对象。在每个用户都需要自己的对象时，采用原型模式。

Bean 工厂的概念是 Spring 作为 IOC 容器的基础。IOC 则将处理事情的责任从应用程序代码转移到框架。

AOP 编程理念

面向切面编程，即 AOP，是一种编程思想，它允许程序员对横切关注点或横切典型的职责分界线的行为（例如日志和事务管理）进行模块化。AOP 的核心构造是方面（切面），它将那些影响多个类的行为封装到可重用的模块中。

AOP 和 IOC 是补充性的技术，它们都运用模块化方式解决企业应用程序开发中的复杂问题。在典型的面向对象开发方式中，可能要将日志记录语句放在所有方法和 Java 类中才能实现日志功能。在 AOP 方式中，可以反过来将日志服务模块化，并以声明的方式将它们应用到需要日志的组件上。当然，优势就是 Java 类不需要知道日志服务的存在，也不需要考虑相关的代码。所以，用 Spring AOP 编写的应

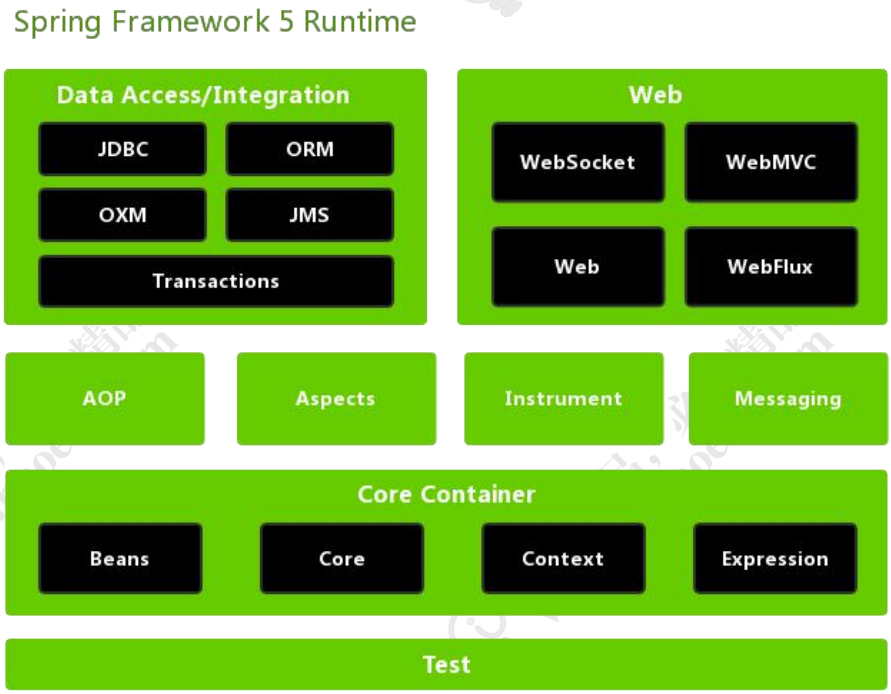
用程序代码是松散耦合的。

AOP 的功能完全集成到了 Spring 事务管理、日志和其他各种特性的上下文中。

AOP 编程的常用场景有 :Authentication(权限认证)、Auto Caching(自动缓存处理)、Error Handling（统一错误处理）、Debugging（调试信息输出）、Logging（日志记录）、Transactions（事务处理）等。

Spring5 系统架构

Spring 总共大约有 20 个模块 ,由 1300 多个不同的文件构成。而这些组件被分别整合在核心容器(Core Container）、AOP（Aspect Oriented Programming）和设备支持（Instrmentation）、数据访问及集成（Data Access/Integeration）、Web、报文发送（Messaging）、Test，6 个模块集合中。以下是 Spring 5 的模块结构图：



组成 Spring 框架的每个模块集合或者模块都可以单独存在，也可以一个或多个模块联合实现。每个模块的组成和功能如下：

核心容器

由 spring-beans、spring-core、spring-context 和 spring-expression（Spring Expression Language, SpEL） 4 个模块组成。

spring-core 和 spring-beans 模块是 Spring 框架的核心模块，包含了控制反转（Inversion of Control, IOC）和依赖注入（Dependency Injection, DI）。BeanFactory 接口是 Spring 框架中的核心接口，它是工厂模式的具体实现。BeanFactory 使用控制反转对应用程序的配置和依赖性规范与实际的应用程序代码进行了分离。但 BeanFactory 容器实例化后并不会自动实例化 Bean，只有当 Bean 被使用时 BeanFactory 容器才会对该 Bean 进行实例化与依赖关系的装配。

spring-context 模块构架于核心模块之上，他扩展了 BeanFactory，为她添加了 Bean 生命周期控制、框架事件体系以及资源加载透明化等功能。此外该模块还提供了许多企业级支持，如邮件访问、远程访问、任务调度等，ApplicationContext 是该模块的核心接口，她是 BeanFactory 的超类，与 BeanFactory 不同，ApplicationContext 容器实例化后会自动对所有的单实例 Bean 进行实例化与依赖关系的装配，使之处于待用状态。

spring-context-support 模块是对 Spring IOC 容器的扩展支持，以及 IOC 子容器。

spring-context-indexer 模块是 Spring 的类管理组件和 Classpath 扫描。

spring-expression 模块是统一表达式语言（EL）的扩展模块，可以查询、管理运行中的对象，同时也方便的可以调用对象方法、操作数组、集合等。它的语法类似于传统 EL，但提供了额外的功能，最出色的要数函数调用和简单字符串的模板函数。这种语言的特性是基于 Spring 产品的需求而设计，他可以非常方便地同 Spring IOC 进行交互。

AOP 和设备支持

由 spring-aop、spring-aspects 和 spring-instrument 3 个模块组成。

spring-aop 是 Spring 的另一个核心模块，是 AOP 主要的实现模块。作为继 OOP 后，对程序员影响最大的编程思想之一，AOP 极大地开拓了人们对于编程的思路。在 Spring 中，他是以 JVM 的动态代理技术为基础，然后设计出了一系列的 AOP 横切实现，比如前置通知、返回通知、异常通知等，同时，Pointcut 接口来匹配切入点，可以使用现有的切入点来设计横切面，也可以扩展相关方法根据需求进行切入。

spring-aspects 模块集成自 AspectJ 框架，主要是为 Spring AOP 提供多种 AOP 实现方法。

spring-instrument 模块是基于 JAVA SE 中的"java.lang.instrument"进行设计的，应该算是 AOP 的一个支援模块，主要作用是在 JVM 启用时，生成一个代理类，程序员通过代理类在运行时修改类的字节，从而改变一个类的功能，实现 AOP 的功能。在分类里，我把他分在了 AOP 模块下，在 Spring 官方文档里对这个地方也有点含糊不清，这里是纯个人观点。

数据访问与集成

由 spring-jdbc、spring-tx、spring-orm、spring-jms 和 spring-oxm 5 个模块组成。

spring-jdbc 模块是 Spring 提供的 JDBC 抽象框架的主要实现模块，用于简化 Spring JDBC 操作 。主要是提供 JDBC 模板方式、关系数据库对象化方式、SimpleJdbc 方式、事务管理来简化 JDBC 编程，主要实现类是 JdbcTemplate、SimpleJdbcTemplate 以及 NamedParameterJdbcTemplate。

spring-tx 模块是 Spring JDBC 事务控制实现模块。使用 Spring 框架，它对事务做了很好的封装，通过它的 AOP 配置，可以灵活的配置在任何一层；但是在很多的需求和应用，直接使用 JDBC 事务控制还是有其优势的。其实，事务是以业务逻辑为基础的；一个完整的业务应该对应业务层里的一个方法；如果业务操作失败，则整个事务回滚；所以，事务控制是绝对应该放在业务层的；但是，持久层的设计则应该遵循一个很重要的原则：保证操作的原子性，即持久层里的每个方法都应该是不可以分割的。所以，在使用 Spring JDBC 事务控制时，应该注意其特殊性。

spring-orm 模块是 ORM 框架支持模块，主要集成 Hibernate, Java Persistence API (JPA) 和 Java Data Objects (JDO) 用于资源管理、数据访问对象(DAO)的实现和事务策略。

spring-oxm 模块主要提供一个抽象层以支撑 OXM（OXM 是 Object-to-XML-Mapping 的缩写，它是一个 O/M-mapper，将 java 对象映射成 XML 数据，或者将 XML 数据映射成 java 对象），例如：JAXB, Castor, XMLBeans, JiBX 和 XStream 等。

spring-jms 模块（Java Messaging Service）能够发送和接收信息，自 Spring Framework 4.1 以后，他还提供了对 spring-messaging 模块的支撑。

Web 组件

由 spring-web、spring-webmvc、spring-websocket 和 spring-webflux 4 个模块组成。

spring-web 模块为 Spring 提供了最基础 Web 支持，主要建立于核心容器之上，通过 Servlet 或者 Listeners 来初始化 IOC 容器，也包含一些与 Web 相关的支持。

spring-webmvc 模块众所周知是一个的 Web-Servlet 模块，实现了 Spring MVC（model-view-Controller）的 Web 应用。

spring-websocket 模块主要是与 Web 前端的全双工通讯的协议。

spring-webflux 是一个新的非堵塞函数式 Reactive Web 框架，可以用来建立异步的，非阻塞，事件驱动的服务，并且扩展性非常好。

通信报文

即 spring-messaging 模块，是从 Spring4 开始新加入的一个模块，主要职责是为 Spring 框架集成一些基础的报文传送应用。

集成测试

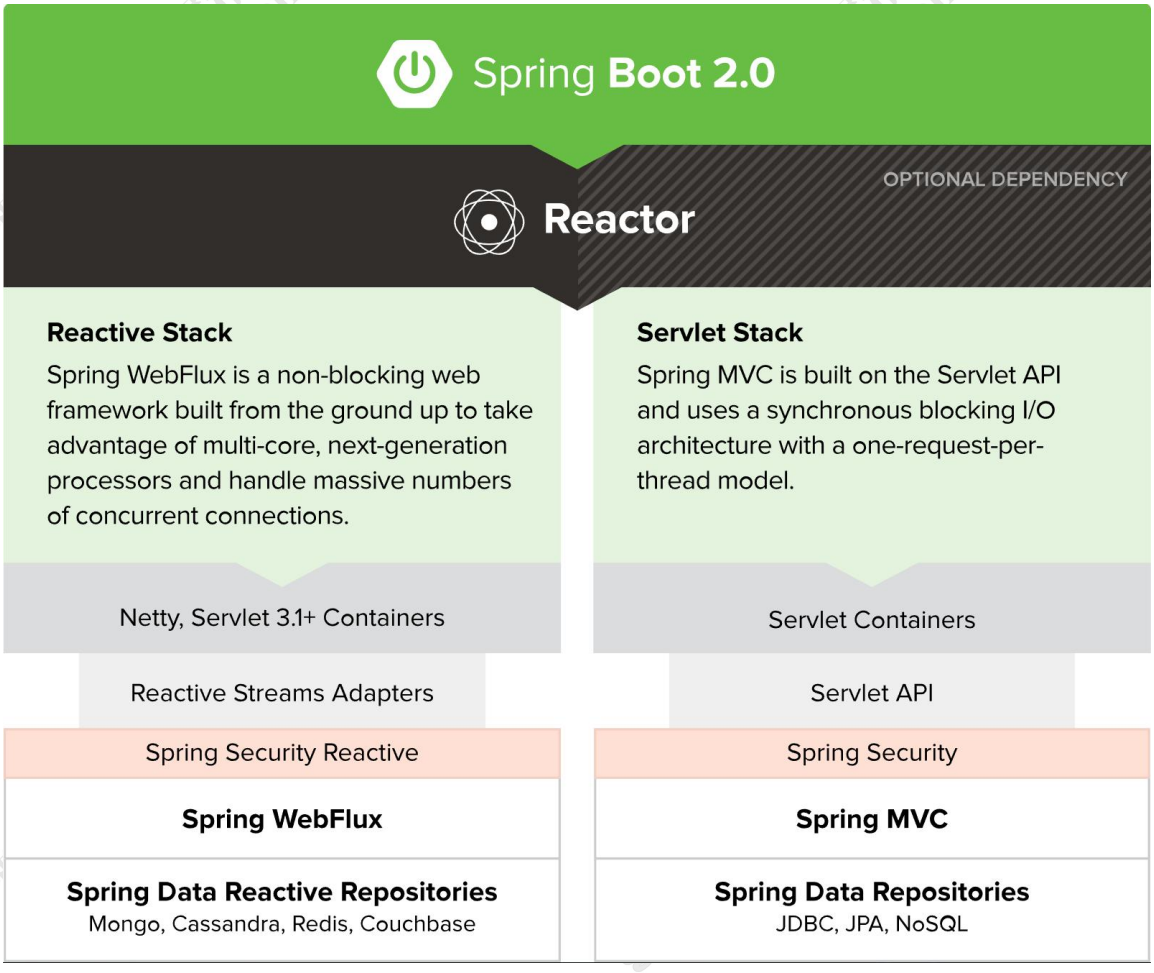
即 spring-test 模块，主要为测试提供支持的，毕竟在不需要发布（程序）到你的应用服务器或者连接到其他企业设施的情况下能够执行一些集成测试或者其他测试对于任何企业都是非常重要的。

集成兼容

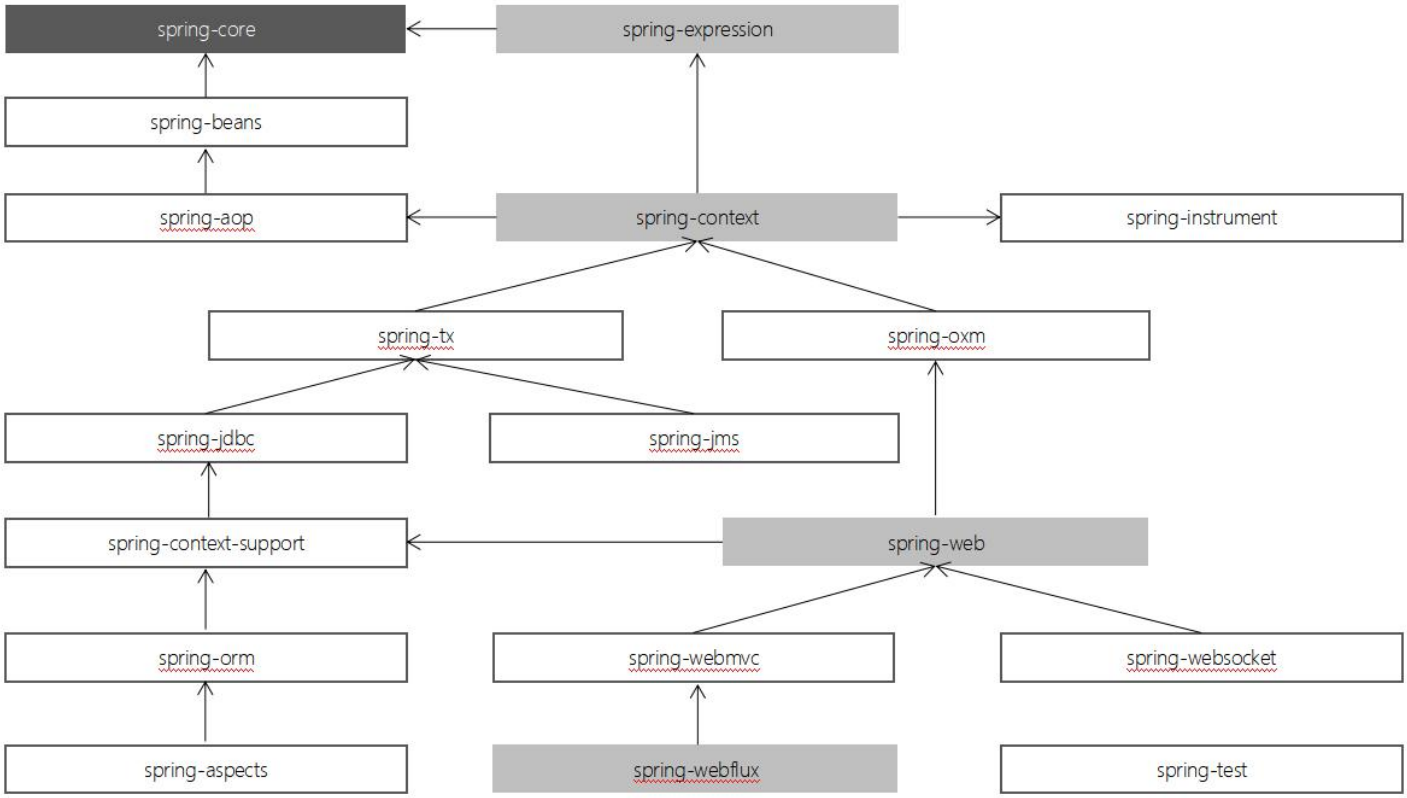
即 spring-framework-bom 模块，Bill of Materials.解决 Spring 的不同模块依赖版本不同问题。

各模块之间的依赖关系

Spring 官网对 Spring5 各模块之间的关系也做了详细说明：



我本人也对 Spring5 各模块做了一次系统的总结，描述模块之间的依赖关系，希望能对小伙伴们有所帮助。



接下来的课程中，我们将深入了解 Spring 的核心模块功能。基本学习顺序为：从 spring-core 入手，其次是 spring-beans 和 spring-aop，随后是 spring-context，再其次是 spring-tx 和 spring-orm，最后是 spring-web 和其他部分。

Spring 版本命名规则

常见软件的版本号命名

软件	升级过程	说明

Linux Kernel	0.0.1 1.0.0 2.6.32 3.0.18 ...	若用 X.Y.Z 表示，则偶数 Y 表示稳定版本，奇数 Y 表示开发版本。
Windows	Windows 98 Windows 2000 Windows XP Windows 7 ...	最大的特点是杂乱无章，毫无规律。
SSH Client	0.9.8	
OpenStack	2014.1.3 2015.1.1.dev8	

从上可以看出，不同的软件版本号风格各异，随着系统的规模越大，依赖的软件越多，如果这些软件没有遵循一套规范的命名风格，容易造成 Dependency Hell。所以当我们发布版本时，版本号的命名需要遵循某种规则，其中 Semantic Versioning 2.0.0 定义了一套简单的规则及条件来约束版本号的配置和增长。本文根据 Semantic Versionning 2.0.0 和 Semantic Versioning 3.0.0 选择性的整理出版本号命名规则指南。

语义化版本命名通行规则

该规则对版本的迭代顺序命名做了很好的规范，其版本号的格式为 X.Y.Z(又称 Major.Minor.Patch)，递增的规则为：

序号	格式要求	说明

X	非负整数	表示主版本号(Major)，当 API 的兼容性变化时，X 需递增。
Y	非负整数	表示次版本号(Minor)，当增加功能时(不影响 API 的兼容性)，Y 需递增。
Z	非负整数	表示修订号（Patch），当做 Bug 修复时(不影响 API 的兼容性)，Z 需递增。

详细的使用规则如下：

X, Y, Z 必须为非负整数，且不得包含前导零，必须按数值递增，如 1.9.0 -> 1.10.0 -> 1.11.0

0.Y.Z 的版本号表明软件处于初始开发阶段，意味着 API 可能不稳定；1.0.0 表明版本已有稳定的API。

当 API 的兼容性变化时，X 必须递增，Y 和 Z 同时设置为 0；当新增功能(不影响 API 的兼容性)或者 API 被标记为 Deprecated 时，Y 必须递增，同时 Z 设置为 0；当进行 bug fix 时，Z 必须递增。

先行版本号(Pre-release)意味该版本不稳定，可能存在兼容性问题，其格式为：X.Y.Z.[a-c][正整数]，如 1.0.0.a1，1.0.0.b99，1.0.0.c1000。

开发版本号常用于 CI-CD，格式为 X.Y.Z.dev[正整数]，如 1.0.1.dev4。

版本号的排序规则为依次比较主版本号、次版本号和修订号的数值，如 1.0.0 < 1.0.1 < 1.1.1 < 2.0.0；对于先行版本号和开发版本号，有：1.0.0.a100 < 1.0.0，2.1.0.dev3 < 2.1.0；当存在字母时，以 ASCII 的排序来比较，如 1.0.0.a1 < 1.0.0.b1。

注意：版本一经发布，不得修改其内容，任何修改必须在新版本发布！

商业软件中常见的修饰词

描述方式	说明	含义
Snapshot	快照版	尚不不稳定、尚处于开发中的版本
Alpha	内部版	严重缺陷基本完成修正并通过复测，但需要完整的功能测试
Beta	测试版	相对 Alpha 有很大的改进，消除了严重的错误，但还是存在一些缺陷

RC	终测版	Release Candidate（最终测试），即将作为正式版发布。
Demo	演示版	只集成了正式版部分功能升级，无法升级
SP	SP1	是 service pack 的意思表示升级包，相信大家在 windows 中都见过。
Release	稳定版	功能相对稳定，可以对外发行，但有时间限制
Trial	试用版	试用版，仅对部分用户发行
Full Version	完整版	即正式版，已发布。
Unregistered	未注册	有功能或时间限制的版本
Standard	标准版	能满足正常使用的功能的版本
Lite	精简版	只含有正式版的核心功能
Enhance	增强版	正式版，功能优化的版本
Ultimate	旗舰版	在标配版本升级体验感更好的版本
Professiona	专业版	针对更高要求功能，专业性更强的使用群体发行的版本
Free	自由版	自由免费使用的版本
Upgrade	升级版	有功能增强或修复已知 bug
Retail	零售版	单独发售
Cardware	共享版	公用许可证（IOS 签证）
LTS	维护版	该版本需要长期维护

软件版本号使用限定

为了方便理解,版本限定的语法简述为为 [范围描述]<版本号描述>

范围描述可选,必须配和版本描述确定范围,无法独立存在

< 小于某一版本号

<= 小于等于某一版本号

> 大于某一版本号

>= 大于等于某一版本号

= 等于某一版本号,没有意义和直接写该版本号一样

~ 基于版本号描述的最新补丁版本

^ 基于版本号描述的最新兼容版本

- 某个范围,他应该出现在两个版本描述中间,实际上语法应为 <版本描述>-<版本描述>,写在此处为了统一

严格来讲对 ~,^ 的表述需要结合具体的包管理工具和版本号规则来确定.但是对于一般使用记住如下原则：

^ 是确保版本兼容性时,默认对次版本号的限定约束

~ 是确保版本兼容性时,默认对补丁号的约束

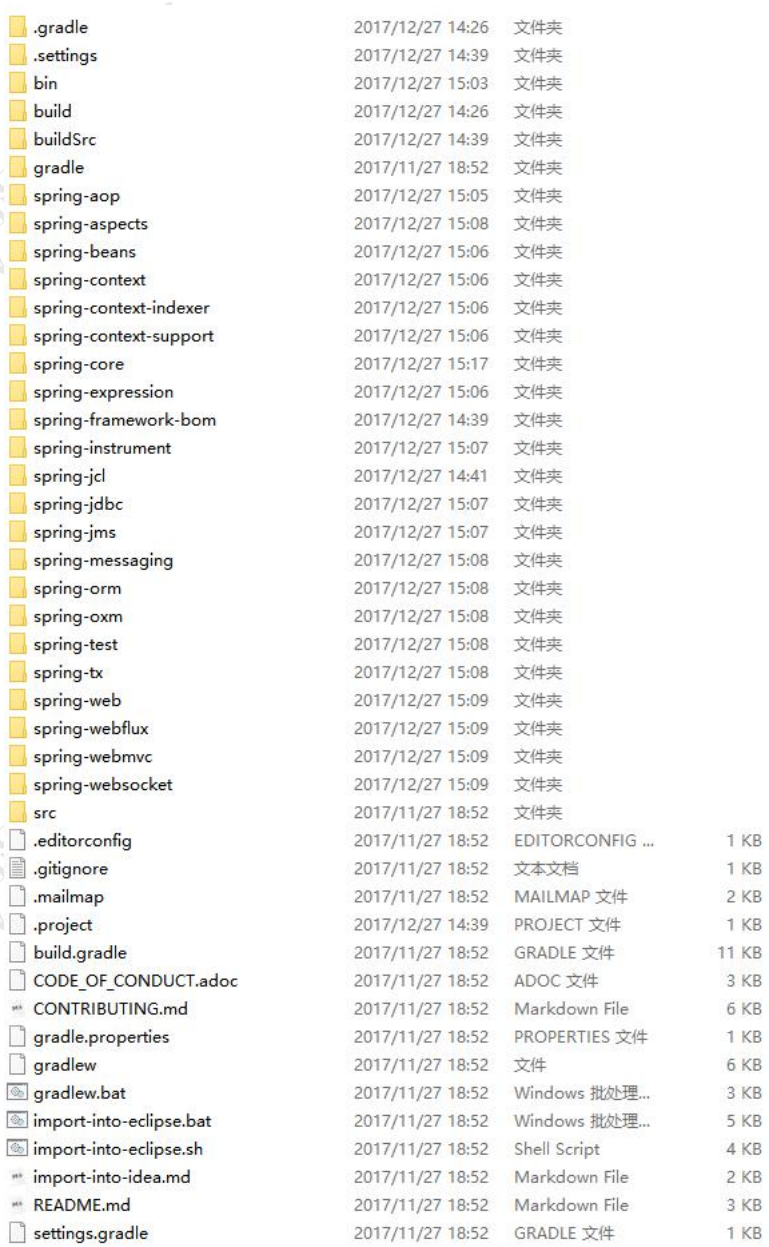
Spring 版本命名规则

描述方式	说明	含义
Snapshot	快照版	尚不不稳定、尚处于开发中的版本
Release	稳定版	功能相对稳定，可以对外发行，但有时间限制
GA	正式版	代表广泛可用的稳定版(General Availability)
M	里程碑版	(M 是 Milestone 的意思）具有一些全新的功能或是具有里程碑意义的版本。
RC	终测版	Release Candidate（最终测试），即将作为正式版发布。

Spring 源码下载及构建技巧

Spring5 源码下载注意事项

首先你的 JDK 需要升级到 1.8 以上。Spring3.0 开始,Spring 源码采用 github 托管，不再提供官网下载链接。这里不做过多赘述，大家可自行去 github 网站下载，我们使用的版本下载链接为：https://github.com/spring-projects/spring-framework/archive/v5.0.2.RELEASE.zip，下载完成后，解压源码包会看到以下文件目录：

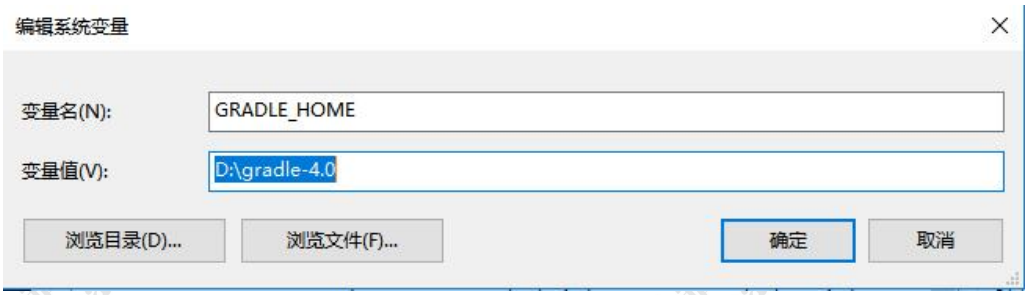


.gradle	2017/12/27 14:26	文件夹	
.settings	2017/12/27 14:39	文件夹	
bin	2017/12/27 15:03	文件夹	
build	2017/12/27 14:26	文件夹	
buildSrc	2017/12/27 14:39	文件夹	
gradle	2017/11/27 18:52	文件夹	
spring-aop	2017/12/27 15:05	文件夹	
spring-aspects	2017/12/27 15:08	文件夹	
spring-beans	2017/12/27 15:06	文件夹	
spring-context	2017/12/27 15:06	文件夹	
spring-context-indexer	2017/12/27 15:06	文件夹	
spring-context-support	2017/12/27 15:06	文件夹	
spring-core	2017/12/27 15:17	文件夹	
spring-expression	2017/12/27 15:06	文件夹	
spring-framework-bom	2017/12/27 14:39	文件夹	
spring-instrument	2017/12/27 15:07	文件夹	
spring-jcl	2017/12/27 14:41	文件夹	
spring-jdbc	2017/12/27 15:07	文件夹	
spring-jms	2017/12/27 15:07	文件夹	
spring-messaging	2017/12/27 15:08	文件夹	
spring-orm	2017/12/27 15:08	文件夹	
spring-oxm	2017/12/27 15:08	文件夹	
spring-test	2017/12/27 15:08	文件夹	
spring-tx	2017/12/27 15:08	文件夹	
spring-web	2017/12/27 15:09	文件夹	
spring-webflux	2017/12/27 15:09	文件夹	
spring-webmvc	2017/12/27 15:09	文件夹	
spring-websocket	2017/12/27 15:09	文件夹	
src	2017/11/27 18:52	文件夹	
.editorconfig	2017/11/27 18:52	EDITORCONFIG ...	1 KB
.gitignore	2017/11/27 18:52	文本文档	1 KB
.mailmap	2017/11/27 18:52	MAILMAP 文件	2 KB
.project	2017/12/27 14:39	PROJECT 文件	1 KB
build.gradle	2017/11/27 18:52	GRADLE 文件	11 KB
CODE_OF_CONDUCT.adoc	2017/11/27 18:52	ADOC 文件	3 KB
CONTRIBUTING.md	2017/11/27 18:52	Markdown File	6 KB
gradle.properties	2017/11/27 18:52	PROPERTIES 文件	1 KB
gradlew	2017/11/27 18:52	文件	6 KB
gradlew.bat	2017/11/27 18:52	Windows 批处理...	3 KB
import-into-eclipse.bat	2017/11/27 18:52	Windows 批处理...	5 KB
import-into-eclipse.sh	2017/11/27 18:52	Shell Script	4 KB
import-into-idea.md	2017/11/27 18:52	Markdown File	2 KB
README.md	2017/11/27 18:52	Markdown File	3 KB
settings.gradle	2017/11/27 18:52	GRADLE 文件	1 KB

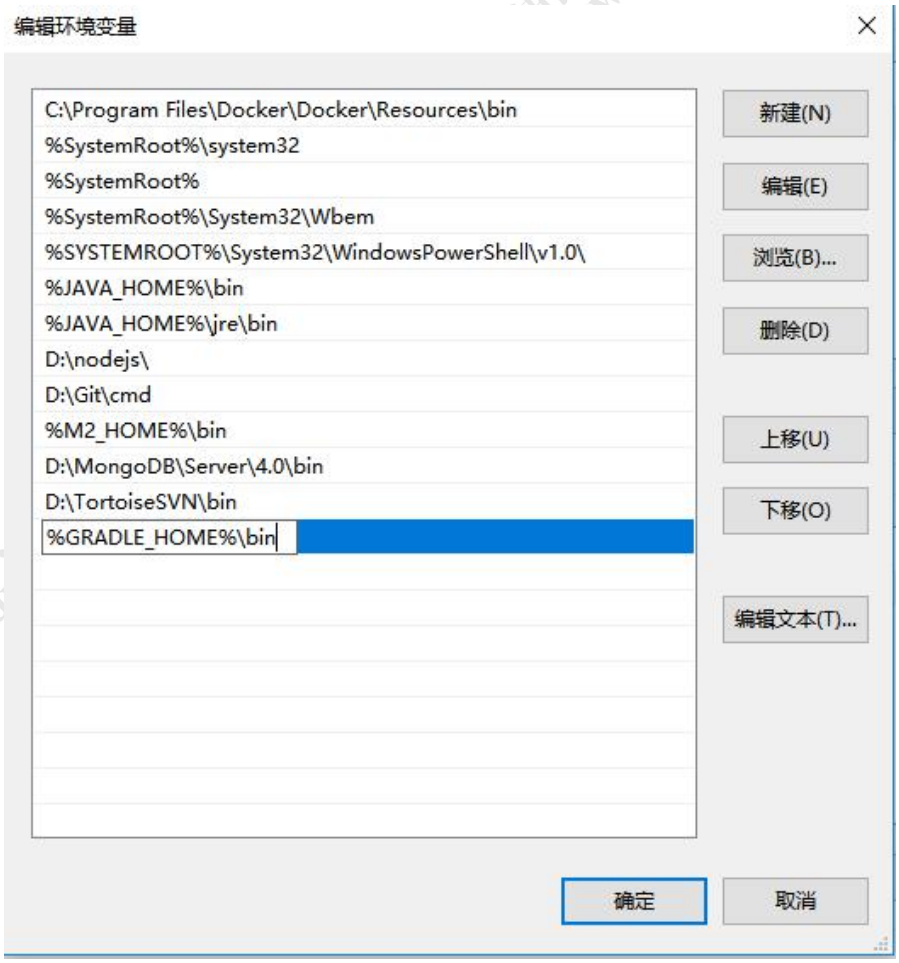
基于 Gradle 的源码构建技巧

由于 Spirng5 以后都是采用 Gradle 来编译，所以构建源码前需要先安装 Gradle 环境。Gradle 下载地址：https://gradle.org/releases，我使用的是 Spring5 官方推荐的版本 Gradle4.0,下载链接为：https://gradle.org/next-steps/?version=4.0&format=bin ，下载完成后按以下步骤操作，以 Windows 操作系统为例：

第一步：配置环境变量

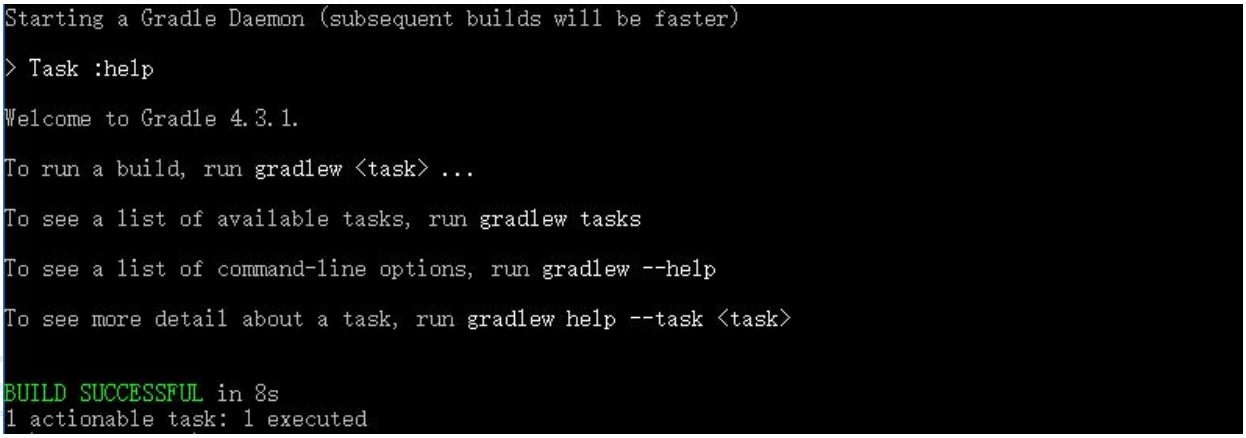


第二步：添加环境变量：Path：%GRADLE_HOME%\bin

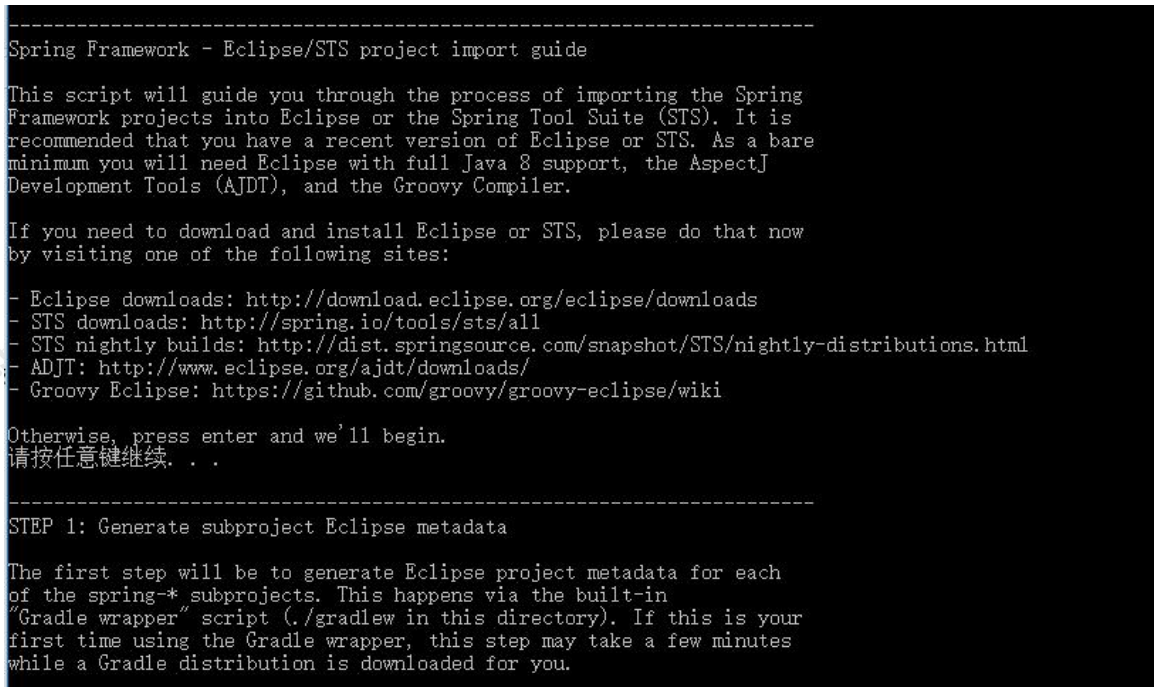


第三步：检测环境，输入 gradle -v 命令，得到以下结果：

第四步：编译源码,cmd 切到 spring-framework-5.0.2.RELEASE 目录，运行 gradlew.bat



第五步：转换为 eclipse 项目，执行 import-into-eclipse.bat 命令，构建前，请确保网络状态良好，按任意键继续。

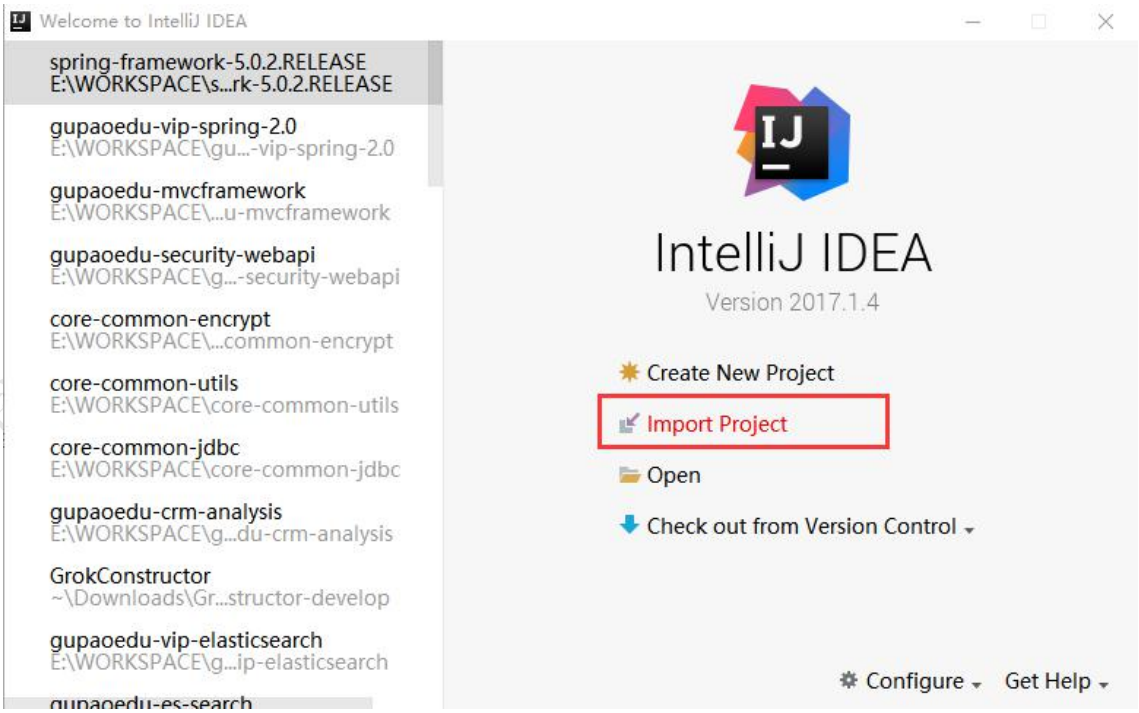


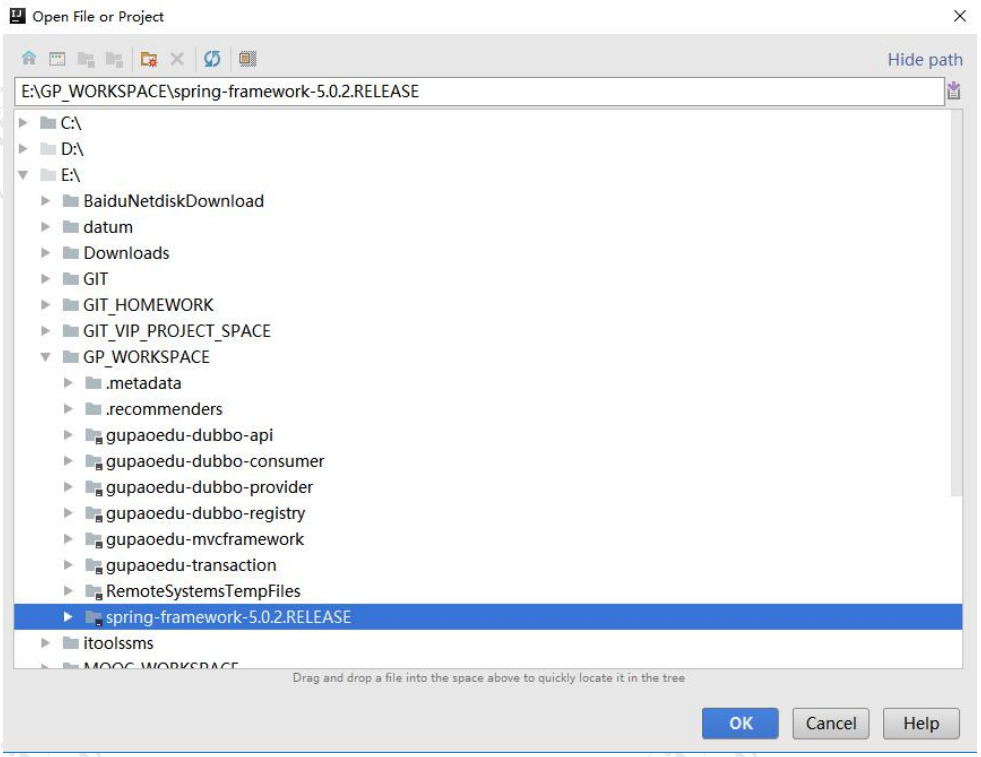
第六步：等待构建成功（若中途出现错误，大部分情况是由于网络中断造成的，重试之后一般都能解决问题），构建成功后，会出现如下界面：

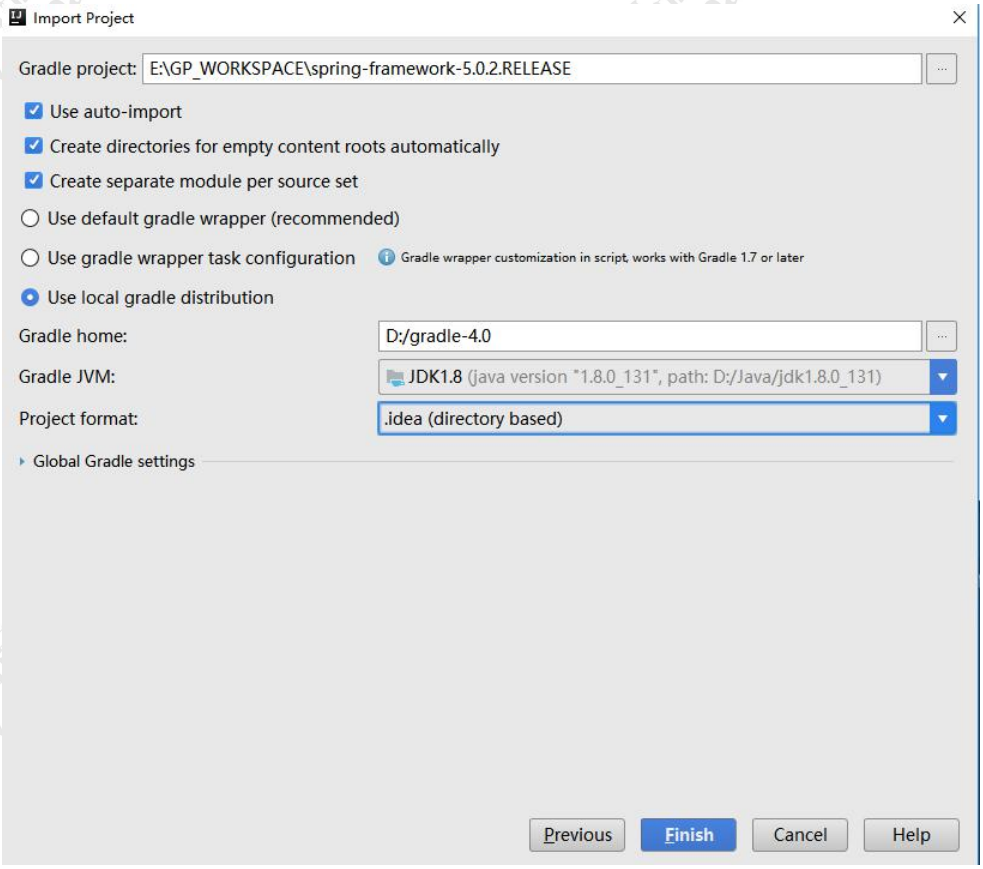
```
BUILD SUCCESSFUL in 6s
6 actionable tasks: 6 executed

------------------------------------------------------------------------
STEP 4: Import root project into Eclipse/STS

Follow the project import steps listed in step 2 above to import the
root project.

Press enter when complete, and move on to the final step.
请按任意键继续. . .

------------------------------------------------------------------------
STEP 5: Enable Git support for all projects

- In the Eclipse/STS Package Explorer, select all spring* projects.
- Right-click to open the context menu and select Team > Share Project...
- In the Share Project dialog that appears, select Git and press Next
- Check "Use or create repository in parent folder of project"
- Click Finish

When complete, you'll have Git support enabled for all projects.

You're ready to code! Goodbye!
```

到这一步为止，还在使用 Eclipse 的小伙伴已经可以将项目导入到 Eclipse 中了。而我们推荐使用的 IDEA 也比较智能，可以直接兼容 Eclipse 项目。接下来看下面的步骤：

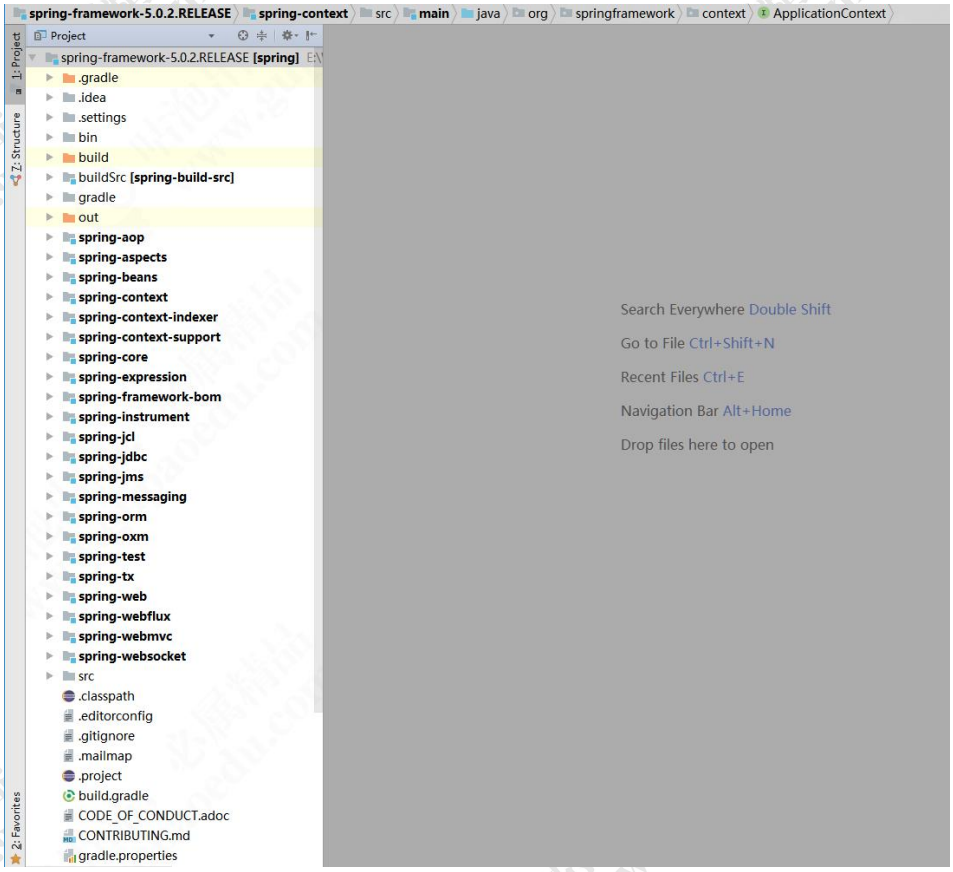
第七步：导入 IDEA。打开 IntelliJ IDEA，点击 Import Project，弹出如下界面，选择 spring-framework-5.0.2.RELEASE 文件夹:



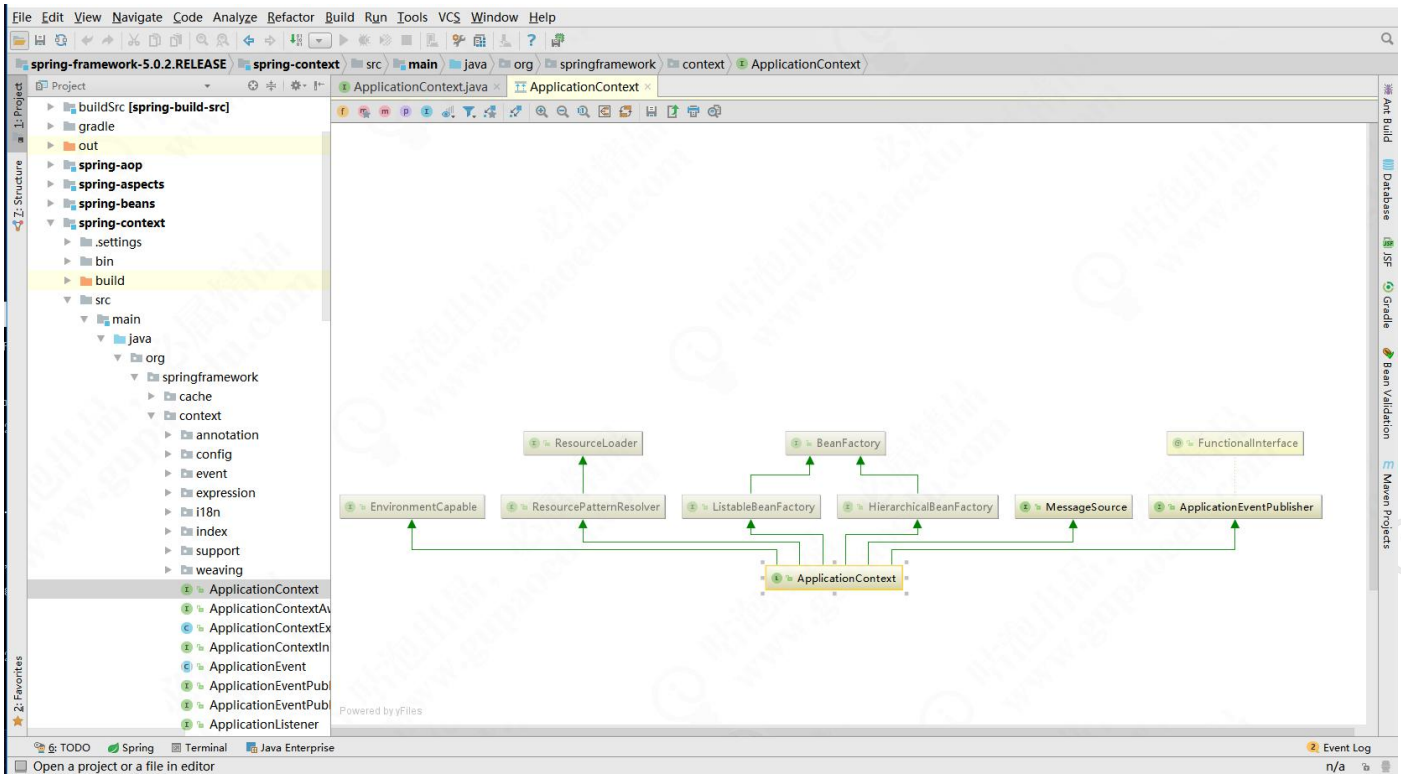




第八步：等待构建完成，在网络良好的情况下大约需要 10 分钟便可自动构建完成，你会看到如下界面：



第九步：在 IDEA 中，如果 Project 下的子项目文件夹变成粗体字之后，说明已经构建成功。还有一种验证方式是：找到 ApplicationContext 类，按 Ctrl + Shift + Alt + U，出现类图界面说明构建成功。

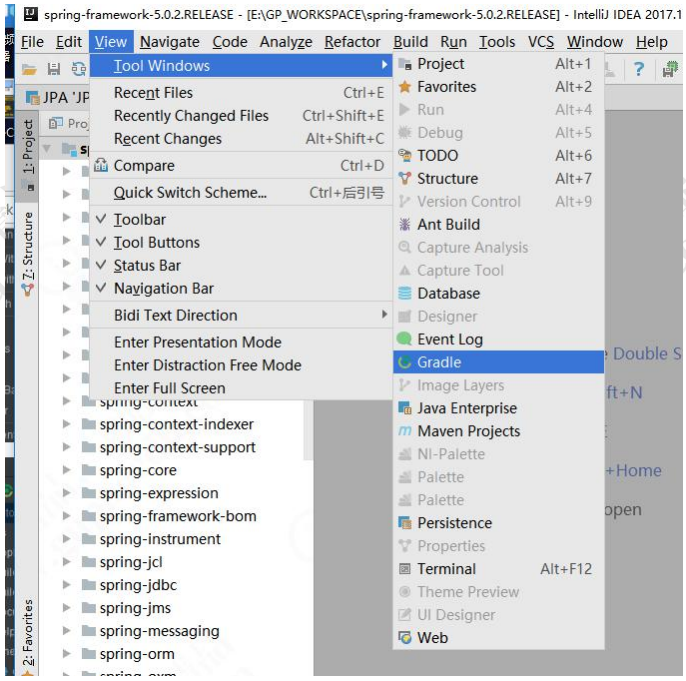


如果我已经将构建好的“spring-framework-5.0.2.RELEASE-中文注释版”，提交到了 Git 上，下载后导入到 IDEA 中开箱即用，无需重复构建。

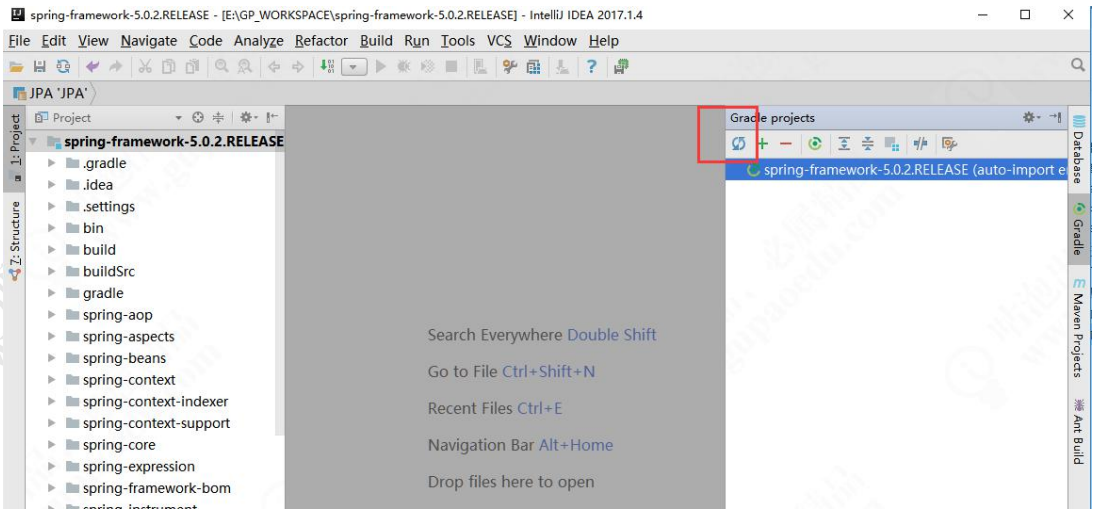
Gradle 构建过程中的坑

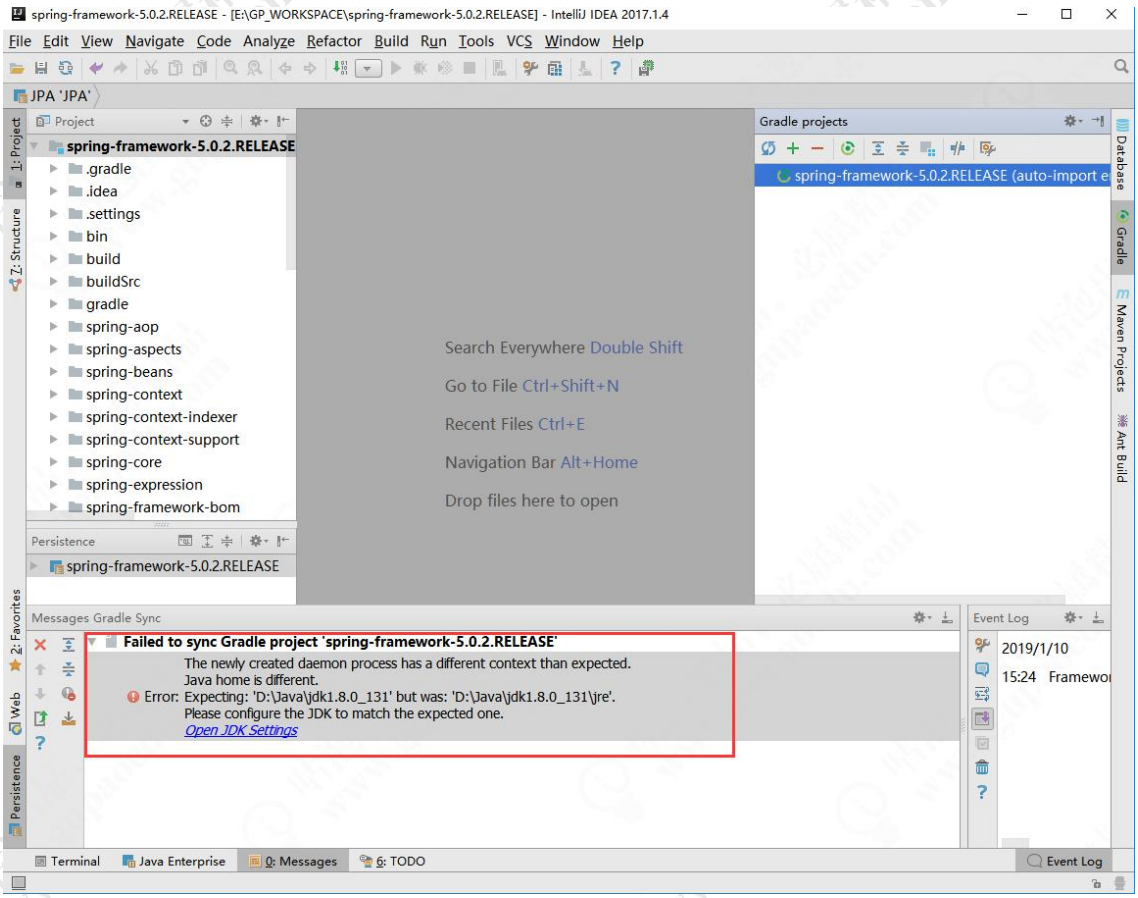
如果项目环境一直无法构建，项目文件夹没有变粗体字，类图无法自动生成。那么你一定是踩到了这样一个坑。

第一步：首先打开 View->Tool Windows -> Gradle



然后，点击右侧 Gradle 视图中的 Refresh，会出现如下的错误：





第二步：看错误，显然跟 Gradle 没有任何关系，解决办法：

1.关闭 IDEA，打开任务管理器，结束跟 java 有关的所有进程。

2.找到 JAVA_HOME -> jre -> lib 目录，将 tools.jar 重命名 tools.jar.bak。

3.重启 IDEA，再次点击 refresh，等待构建完成。