咕泡学院 JavaVIP 高级课程教案

深入分析 Spring 源码(第一阶段)

关于本文档

主题	咕泡学院 Java VIP 高级课程教案--深入分析 Spring 源码（第一阶段）
主讲	Tom 老师
适用对象	咕泡学院 Java 高级 VIP 学员及 VIP 授课老师
源码版本	v3.2.6.RELEASE

一、Spring 概述

Spring 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、高内聚。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、工厂方法设计模式"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。让我们来深入看一下 Spring 到底能帮我们做些什么？

二、一切从 Bean 开始

1996，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。但这些开发者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，当时也是面向对象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是面向对象的，当然这是后话了）。他们为 Applet 而来，为组件化而留。这便是最早的 Java。

同样在这一年的 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++，他看起来还是太简易了，以至于无法胜任任何"实际的"工作。

复杂的应用通常需要诸如事物、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 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，EJB Bean 已经和 JavaBean 没有任何关系了。

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 框架也是在这样的大环境下出现的。

2.1、Spring 设计伊始

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，而所有地这些的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、依赖注入以及面向切面这三种方式来达成的。

2.2、面向 Bean

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。

2.3、依赖注入

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 支持两个对象模型。

1，单例：模型提供了具有特定名称的对象的共享实例，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。对于无状态服务对象很理想。

2，原型：模型确保每次检索都会创建单独的对象。在每个用户都需要自己的对象时，原型模型最适合。

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。

2.4、面向切面

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。

Authentication 权限认证

Logging 日志

Transctions Manager 事务

Lazy Loading 懒加载

Context Process 上下文处理

Error Handler 错误跟踪（异常捕获机制）

Cache 缓存

2.5、常用的设计模式

代理模式：

静态代理、动态代理

JDK 动态代理：

CGLib 动态代理：

cglib.jar（全称 Code Generation Library 代码生成库）

asm.jar(全称 assembly，装配)

特点：

1、执行者、被代理人

2、对于被代理人来说，这件事情是一定要做的，但是我自己又不想做或者没有时间做，找代理。

3、需要获取到被代理的人个人资料。

穷举法：

租房中介：中介和你

火车票黄牛：黄牛和你

媒人：媒婆和你

明星经纪人：经纪人和明星

刘德华要开演唱会（长沙）、准备工作和善后工作

快递：

总结:做了一件什么事情呢？　字节码重组

可以做一件什么事情？ 可以在每一个方法调用之前加一些代码，在方法调用之后再加一些代码

AOP：事务代理（声明式事务，哪个方法需要加事务，哪个方法不需要加事务）

　　日志监听

service 方法

开启一个事务(open)

事务的执行（是由我们自己的代码完成的）

监听到是否有异常，可能需要根据异常的类型来决定这个事务是否要回滚还是继续提交(commit/rollback)

事务要关闭(close)

工厂模式：

特点：

1、隐藏复杂的逻辑处理过程，只关心执行结果。

简单工厂、工厂方法、抽象工厂

任督二脉

单例模式：

常用单例模式写法

2WH(人都是天生的惰性)

穷举法：

配置文件：如果不是单例（针对于某一种功能的配置）

（两个配置文件中的内容一样的，则有一个是浪费的

如果是不一样的，我们就不知道以哪个为准了）

直接上级领导：（对于你来说，如果有多个领导，你到底听谁的？ 选择恐惧症）

日历：（不严谨）在中国，是有两种日历同时存在的

阴历(农历，一般用于指导农业生产，二十四节气)

和阳历（公历，一般用于国际交流、企事业单位的工作指导）

（在万千世界中，即使是一片小小的树叶，他们都是长得不一样的）

是什么？为什么要有单例模式？单例具体怎么实现？

特点：

1、保证从系统启动到系统终止，全过程只会产生一个实例。
2、当我们在应用中遇到功能性冲突的时候，需要使用单例模式。

单例模式的七种写法：

委派模式：

要和代理模式区分开来

2WH

What:两个角色,受托人，委托人（社会上是平等关系）

公司里面：项目经理，普通员工(法律上平等的，工作的关系，各自的职责会不一样)

干活是我的，功劳是你的（最重要的特点）

项目经理（委托人）：主要职责是安排任务

普通员工（受托人）：执行任务

特点：

1、类似于中介的功能（委托机制）；
2、持有被委托人的引用。

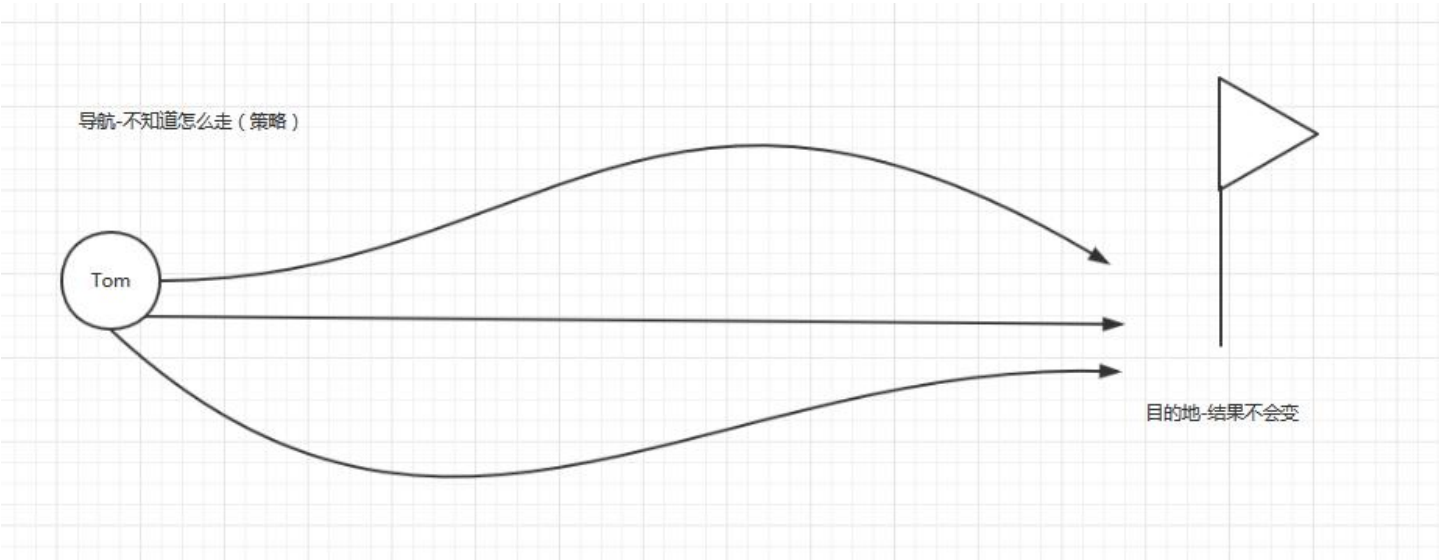
3、不关心过程，只关心结果。

Why? 主要目的就是隐藏具体实现逻辑

IOC 容器中，有一个 Register 的东西（为了告诉我们的容器，在这个类被初始化的过程中，需要做很多不同的逻辑处理，需要实现多个任务执行者，分别实现各自的功能）

保证结果的多样性，对于用户来是只有一种方法

策略模式：1、过程不同，但结果一样。



原型模式：1、数据内容完全一样，但实例不同。

```
------------------------
大圣本尊生日是：1495287417618
克隆大圣的生日是:1495287417646
大圣本尊和克隆大圣是否为同一个对象:false
大圣本尊持有的金箍棒跟克隆大圣持有金箍棒是否为同一个对象:false
```

```
<terminated> TestPrototype [Java Application] D:\Java\jre1.8.0_45\bin\javaw.exe (2017年5月20日
大圣本尊生日是：1495286717857
克隆大圣的生日是:1495286717857
大圣本尊和克隆大圣是否为同一个对象:false
大圣本尊持有的金箍棒跟克隆大圣持有金箍棒是否为同一个对象:true
```

模板模式：1、执行流程固定，但中间有些步骤有细微差别。

Spring 思想	应用场景（特点）	一句话归纳
AOP	1、Aspect Oriented Programming 2、找出多个类中有一定规律的代码，开发时拆开，运行时再合并。 3、面向切面编程，即面向规则编程。	解耦，专人做专事
OOP	1、Object Oriented Programming 2、归纳总结生活中一切事物。	封装、继承、多态
BOP	3、Bean Oriented Programming 4、面向 Bean（普通的 java 类）设计程序。	一切从 Bean 开始
IOC	1、Inversion of Control 2、将 new 对象的动作交给 Spring 管理，并由 Spring 保存已创建的对象。	转交控制权
DI/DL	1、Dependency Injection/Dependency Lookup 2、依赖注入、依赖查找，Spring 不仅保存自己创建的对象，而且保存对象与对象之间的关系。 3、注入即赋值，主要三种方式构造方法、set 方法、直接赋值。	先理清关系再赋值

设计模式	应用场景（特点）	一句话归纳
代理模式 Proxy	1、两个参与角色：执行者、被代理人。 2、对于被代理人来说，这件事情是一定要做的，但是我自己又不想做或者没有时间做，找代理。 3、需要获取到被代理的人个人资料。	办事要求人
工厂模式 Factory	1、对调用者隐藏复杂的逻辑处理过程，调用者只关心执行结果。 2、工厂要对结果负责，保证生产出符合规范的产品。	只对结果负责，不要三无产品。

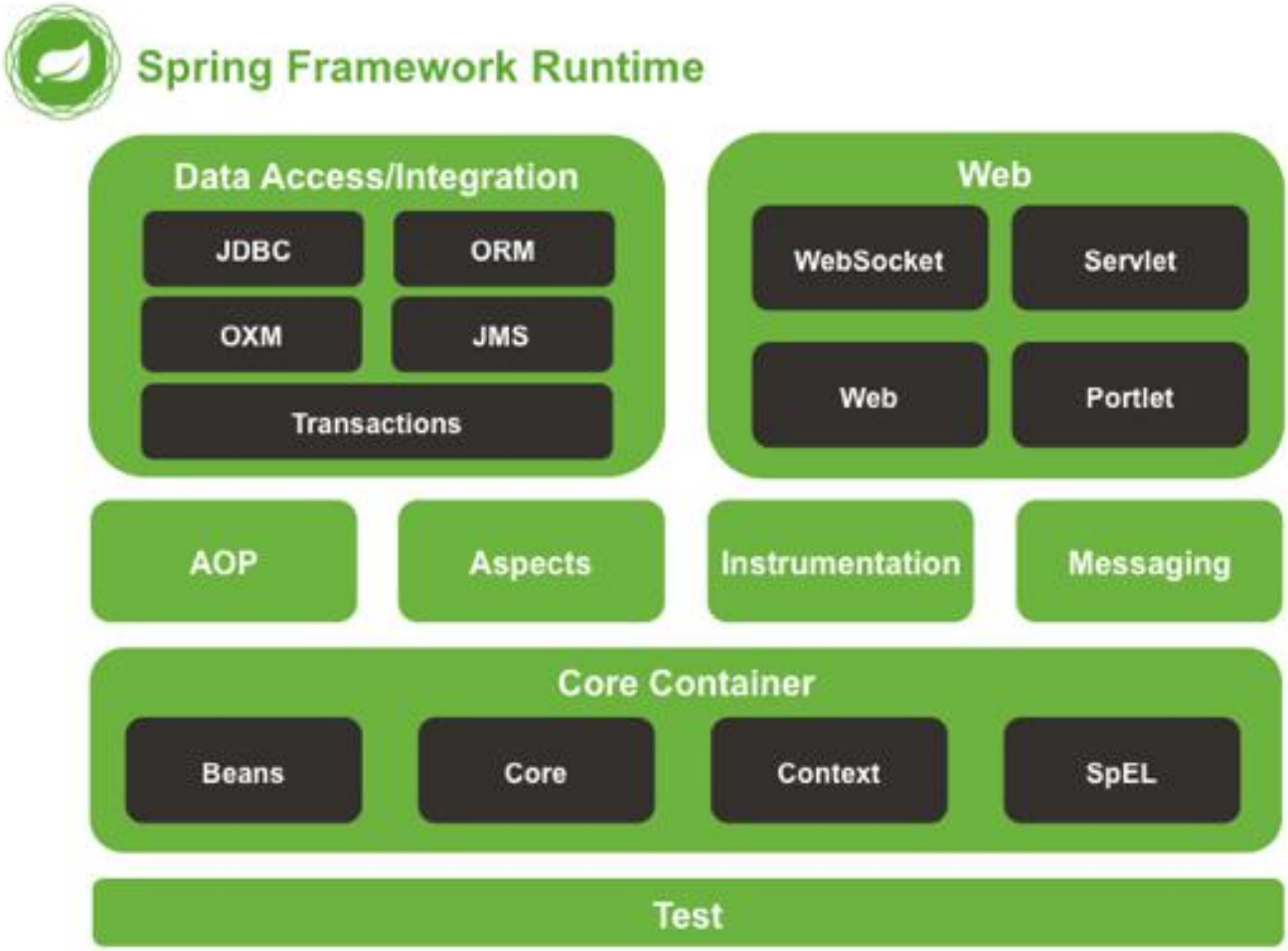
单例模式	1、保证从系统启动到系统终止，全过程只会产生一个实例。 2、当我们在应用中遇到功能性冲突的时候，需要使用单例模式。	保证独一无二
委派模式	1、两个参与角色，委托人和被委托人。 2、委托人和被委托人在权利上完全平等（即实现同一个接口） 3、委托人持有被委托人的引用。 4、不关心过程，只关心结果。	干活是你的（普通员工）， 功劳是我的（项目经理）。
策略模式	1、执行最终结果一样。 2、执行过程和执行逻辑不一样。	我行我素，只看结果。
原型模式	1、首先有一个原型。 2、数据内容相同，但对象实例不同（完全两个不同的内存地址）。	拔一根猴毛，吹出千万个。
模板模式	1、执行流程固定，但中间有些步骤有细微差别。	流程标准化，原料自己加。

三、俯瞰 Spring 架构设计

3.1、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 4 的系统架构图。



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：

1.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-beans 和 spring-core 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-contest 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-expression 模块是统一表达式语言（**unified 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。

2.AOP 和设备支持：由 spring-aop、spring-aspects 和 spring-instrumentation 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-instrumentation 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。**

3.数据访问及集成：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。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-jms 模块（**Java Messaging Service**）能够发送和接受信息，自 Spring Framework 4.1 以后，他还提供了对 spring-messaging 模块的支撑。

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。

4.Web：由 spring-web、spring-webmvc、spring-websocket 和 spring-webmvc-portlet 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。（**资料缺乏，这是个人理解**）

spring-webmvc-portlet 模块是知名的 Web-Portlets 模块(**Portlets 在 Web 门户上管理和显示的可插拔的用户界面组件。Portlet 产生可以聚合到门户页面中的标记语言代码的片段，如 HTML，XML 等**)，主要是为 SpringMVC 提供 Portlets 组件支持。

5.报文发送：即 spring-messaging 模块。

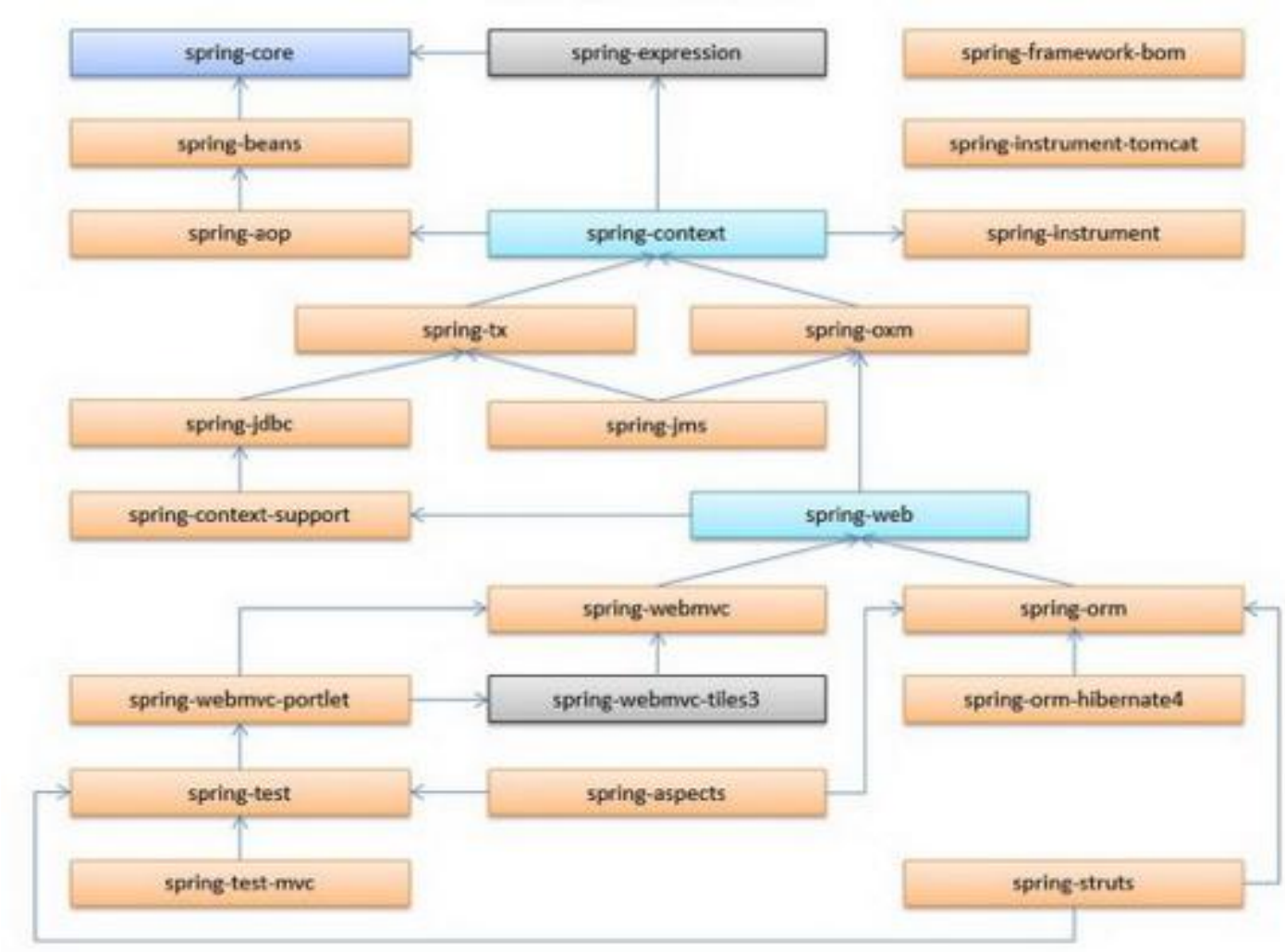
spring-messaging 是 Spring4 新加入的一个模块，主要职责是为 Spring 框架集成一些基础的报文传送应用。

6.Test：即 spring-test 模块。

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。

3.2、依赖关系

该图是 SPRING 3.2.X 的包结构，可以从中清楚看出 Spring 各个模块之间的依赖关系。



从图中可以看出，IOC 的实现包 spring-beans 和 AOP 的实现包 spring-aop 是整个框架的基础，而 spring-core 则是整个框架的核心，基础的功能都在这三个包里。

在此基础之上，spring-context 提供上下文环境，为各个模块提供粘合作用。

在 spring-context 基础之上提供了 spring-tx 和 spring-orm 包，而 web 部分的功能，都是要依赖 spring-web 来实现的。

由于 struts 框架自身的 Bug 一直没有修复，以及 Spring MVC 已经足够强大，所以在最新的 spring 4 中已经移除了 spring-struts 模块。

如果你想加入 spring 源码的学习，笔者的建议是从 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。

四、Spring 源码下载

第一步：https://github.com/spring-projects/spring-framework/releases/tag/v3.2.6.RELEASE

第二步：下载 gradle

http://downloads.gradle.org/distributions/gradle-1.6-bin.zip

第三步：解压,配置 GRADLE_HOME 和 Path

第四步：验证 gradle -v，环境变量是否正确

第五步：构建 eclipse 项目 gradle eclipse -x :eclipse

第六步：引入到 Eclipse 环境中