

Keep looking until you find it.

spring

- ★ 帮助读者**快速熟悉**Spring源码，以便对Spring源码进行**扩展**或**修改**，从而满足业务需求
- ★ 所有知识点均以**HelloWorld级别示例**为切入点，描述**简单**之后的复杂
- ★ 对于**复杂逻辑的讲解**采用**剥洋葱**似的方式，层层**分解复杂度**，便于读者理解和掌握

Spring

源码深度解析

● 郝佳 编著



人民邮电出版社
POSTS & TELECOM PRESS

Spring

源码深度解析

● 郝佳 编著



人民邮电出版社
北京

图书在版编目 (C I P) 数据

Spring源码深度解析 / 郝佳编著. -- 北京 : 人民邮电出版社, 2013.9
ISBN 978-7-115-32568-6

I. ①S… II. ①郝… III. ①JAVA语言—程序设计
IV. ①TP312

中国版本图书馆CIP数据核字(2013)第165971号

内 容 提 要

本书从核心实现和企业应用两个方面,由浅入深、由易到难地对 Spring 源码展开了系统的讲解,包括 Spring 的设计理念和整体架构、容器的基本实现、默认标签的解析、自定义标签的解析、bean 的加载、容器的功能扩展、AOP、数据库连接 JDBC、整合 MyBatis、事务、SpringMVC、远程服务、Spring 消息服务等内容。

本书不仅介绍了使用 Spring 框架开发项目必须掌握的核心概念,还指导读者如何使用 Spring 框架编写企业级应用,并针对在编写代码的过程中如何优化代码、如何使得代码高效给出切实可行的建议,从而帮助读者全面提升实战能力。

本书语言简洁,示例丰富,可帮助读者迅速掌握使用 Spring 进行开发所需的各种技能。本书适合于已具有一定 Java 编程基础的读者,以及在 Java 平台下进行各类软件开发的开发人员、测试人员等。

-
- ◆ 编 著 郝 佳
责任编辑 杜 洁
责任印制 程彦红 杨林杰
 - ◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子邮件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
北京艺辉印刷有限公司印刷
 - ◆ 开本: 800×1000 1/16
印张: 24.75
字数: 545 千字 2013 年 9 月第 1 版
印数: 1-3 000 册 2013 年 9 月北京第 1 次印刷
-

定价: 69.00 元

读者服务热线: (010)67132692 印装质量热线: (010)67129223
反盗版热线: (010)67171154

前言

源代码的重要性

Java 开发人员都知道，阅读源码是一个非常好的学习方式，在我们日常工作中或多或少都会接触一些开源代码，比如说最常用的 Struts、Hibernate、Spring，这些源码的普及与应用程度远远超过我们的想象，正因为很多人使用，也在推动着源码不断地去完善。这些优秀的源码中有着多年积淀下来的精华，这些精华是非常值得我们学习的，不管我们当前是什么水平，通过反复阅读源码能力能有所提升，小到对源码所提供的功能上的使用更加熟练，大到使我们的程序设计更加完美优秀。但是，纵观我们身边的人，能够做到通读源码的真的是少之又少，究其原因不外乎以下几点。

- 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，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函数后，这时我们已经不知道自己所处的位置了，不得不再重来，但是一一次又一次的，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，而这些问题有的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会发生。我们选择各种查阅资料，查询无果，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你会发现，他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，到最后的結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，速度会有很大提升。

以我为例，Spring 是我阅读的第一个源码，几乎耗尽了我将近半年的时间，其中各种煎熬可想而知，但是当我读完 Spring 再去读 MyBatis 只用了两周时间。当然，暂且不论它们的复杂程度不同，至少我阅读的时候发现有很多相通的东西。当你第一次阅读的时候，你的重点一定是在源码的理解上，但是，当你读完第一个源码再去读下一个的时候，你自然而然地会带着批判或者说挑剔的眼光去阅读：为什么这个功能在我之前看的源码中是那样实现的，而在这里会是这样实现的？这其中的道理在哪里，哪种实现方式更优秀呢？而通过这样的对比及探索，你会发现，自己的进步快得难以想象。

我们已经有些纠结了，既然阅读源码有那么多的好处，但是很多同学却因为时间或者能力的问题而不得不放弃，岂不是太可惜？为了解决这个问题，我撰写了本书，总结了自己的研究心得和实际项目经验，希望能对正在 Spring 道路上摸索的同仁们提供一些帮助。

本书特点

本书完全从开发者的角度去剖析源码，每一章都会提供具有代表性的实例，并以此为基础进行功能实现的分析，而不是采取开篇就讲解什么容器怎么实现、AOP 怎么实现之类的写法。在描述的过程中，本书尽可能地把问题分解，使用剥洋葱的方式一层一层地将逻辑描述清楚，帮助读者由浅入深地进行学习，并把这些难点和问题各个击破，而不是企图一下让读者理解一个复杂的逻辑。

在阅读源码的过程中，我们难免会遇到各种各样的生僻功能，这些功能在特定的场合会非常有用，但是可能多数情况下并不是很常用，甚至都查阅不到相关的使用资料。本书中重点针对这种情况提供了相应的实用示例，让读者更加全面地了解 Spring 所提供的功能，对代码能知其然还知其所以然。

本书按照每章所提供的示例跟踪 Spring 源码的流程，尽可能保证代码的连续性，使读者的思维不被打乱，让读者看到 Spring 的执行流程，旨在尽量使读者在阅读完本书后即使在不阅读 Spring 源码的情况下也可以对 Spring 源码进行优化，甚至通过扩展源码来满足业务需求，这对开发人员来说是一个很高的要求。本书就希望能帮助读者全面提升实战能力。

本书结构

本书分为两部分：核心实现和企业应用。

- 第一部分 核心实现（第 1~7 章）：是 Spring 功能的基础，也是企业应用部分的基础，主要对容器以及 AOP 功能实现做了具体的分析，如果读者之前没有接触过 Spring 源代码，建议认真阅读这个部分，否则阅读企业应用部分时

会比较吃力。

- 第二部分 企业应用(第8~13章): 在核心实现部分的基础上围绕企业应用常用的模块进行讨论, 这些模块包括 Spring 整合 JDBC、Spring 整合 MyBatis、事务、SpringMVC、远程服务、Spring 消息服务等, 旨在帮助读者在日常开发中更加高效地使用 Spring。

本书适用的 Spring 版本

截至完稿, Spring 已经发布了 4.0.0.M1 版本。本书虽然是基于 Spring 3.2 版本编写的, 但所讨论的内容都属于 Spring 的基础和常用的功能, 这些功能都经过长时间、大量用户的验证, 已经非常成熟, 改动的可能性相对较小。而且从目前 Spring 的功能规划来看, 本书所涉及的内容并不在 Spring 未来改动的范围内, 因此在未来的很长一段时期内本书都不会过时的。

感谢

创作的过程是痛苦的, 持续时间也远远超乎了我的想象, 而本以为自己对 Spring 已经非常的熟悉, 但是在写作的过程中还是会遇到各种各样的问题, 但是我很幸运我能坚持下来, 在这里我首先应该感谢爸爸妈妈, 虽然他们不知道儿子在忙忙碌碌地写些什么, 但是他们对我始终如一的支持与鼓励使我更加坚定信心, 在这里祝他们身体健康, 同时还要感谢我最好的朋友孙亚超在我低落时给予我的关心与问候, 当然要感谢的还有张雨绮同学, 虽然不是明星, 但是却有着堪比明星般的美丽笑容, 与她在一起的讨论总是让我受益匪浅, 同时也感谢妹子王晶对稿件提供的建议与意见。最后感谢郭维云、郝云勃、郝俊、李兴全、梁晓颖、陈森、孙伟超、王璐、刘瑞、单明、姚佳林、闫微微、李娇、时宇、李平、唐广亮、刘阳、黄思文、金施源等在整个编写过程中给予的支持与帮助。

联系作者

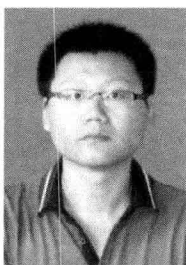
在编写本书过程中, 以“够用就好”为原则, 尽量覆盖到 Spring 开发的常用功能。所有观点都出自作者的个人见解, 疏漏、错误之处在所难免, 欢迎大家指正。读者如果有好的建议或者学习本书过程中遇到问题, 请发送邮件到 haojia_007@163.com, 希望能够与大家一起交流和进步。

在看得见的地方学习知识, 在看不到的地方学习智慧。祝愿大家在 Spring 的学习道路上顺风顺水。

作者

2013年7月

作者简介



郝佳

计算机专业硕士学位，曾发表过多篇论文先后被 EI、SCI 收录，2008 年辽宁省教育厅科技计划项目研究人之一；长期奋斗于 J2EE 领域，曾任职于某互联网公司软件架构师，擅长系统的性能优化，目前正在投身于开发一款基于 Java 并发多线程管理的开源框架；热衷于研究各种优秀的开源代码并从中进行总结，从而实现个人技能的提高，尤其对 Spring、Hibernate、MyBatis、JMS、Tomcat 等源码有着深刻的理解和认识。

目录

第一部分 核心实现

第 1 章 Spring 整体架构和环境搭建 2

- 1.1 Spring 的整体架构 2
- 1.2 环境搭建 4
 - 1.2.1 安装 GitHub 4
 - 1.2.2 安装 Gradle 5
 - 1.2.3 下载 Spring 6

第 2 章 容器的基本实现 10

- 2.1 容器基本用法 10
- 2.2 功能分析 11
- 2.3 工程搭建 12
- 2.4 Spring 的结构组成 13
 - 2.4.1 beans 包的层级结构 13
 - 2.4.2 核心类介绍 13
- 2.5 容器的基础
 - XmlBeanFactory 17
 - 2.5.1 配置文件封装 18
 - 2.5.2 加载 Bean 21
- 2.6 获取 XML 的验证模式 24
 - 2.6.1 DTD 与 XSD 区别 24
 - 2.6.2 验证模式的读取 26
- 2.7 获取 Document 28

- 2.7.1 EntityResolver 用法 29
- 2.8 解析及注册
 - BeanDefinitions 31
 - 2.8.1 profile 属性的使用 32
 - 2.8.2 解析并注册 BeanDefinition 33

第 3 章 默认标签的解析 35

- 3.1 bean 标签的解析及注册 35
 - 3.1.1 解析 BeanDefinition 37
 - 3.1.2 AbstractBeanDefinition 属性 55
 - 3.1.3 解析默认标签中的自定义标签元素 58
 - 3.1.4 注册解析的 BeanDefinition 60
 - 3.1.5 通知监听器解析及注册完成 63
- 3.2 alias 标签的解析 63
- 3.3 import 标签的解析 65
- 3.4 嵌入式 beans 标签的解析 67

第 4 章 自定义标签的解析 68

- 4.1 自定义标签使用 69

- 4.2 自定义标签解析 71
 - 4.2.1 获取标签的命名空间 72
 - 4.2.2 提取自定义标签处理器 72
 - 4.2.3 标签解析 74
- 5 第5章 bean的加载 78
 - 5.1 FactoryBean的使用 83
 - 5.2 缓存中获取单例bean 85
 - 5.3 从bean的实例中获取对象 86
 - 5.4 获取单例 90
 - 5.5 准备创建bean 92
 - 5.5.1 处理 override 属性 93
 - 5.5.2 实例化的前置处理 94
 - 5.6 循环依赖 96
 - 5.6.1 什么是循环依赖 96
 - 5.6.2 Spring 如何解决循环依赖 96
 - 5.7 创建bean 100
 - 5.7.1 创建bean的实例 103
 - 5.7.2 记录创建bean的ObjectFactory 112
 - 5.7.3 属性注入 115
 - 5.7.4 初始化bean 124
 - 5.7.5 注册 DisposableBean 128
- 6 第6章 容器的功能扩展 129
 - 6.1 设置配置路径 130
 - 6.2 扩展功能 130
 - 6.3 环境准备 132
 - 6.4 加载 BeanFactory 133
 - 6.4.1 定制 BeanFactory 135
 - 6.4.2 加载 BeanDefinition 136
 - 6.5 功能扩展 137
 - 6.5.1 增加 SPEL 语言的支持 138
 - 6.5.2 增加属性注册编辑器 139
 - 6.5.3 添加 ApplicationContextAwareProcessor 处理器 144
 - 6.5.4 设置忽略依赖 146
 - 6.5.5 注册依赖 146
- 6.6 BeanFactory 的后处理 146
 - 6.6.1 激活注册的 BeanFactory PostProcessor 147
 - 6.6.2 注册 BeanPostProcessor 153
 - 6.6.3 初始化消息资源 156
 - 6.6.4 初始化 ApplicationEvent Multicaster 159
 - 6.6.5 注册监听器 161
- 6.7 初始化非延迟加载单例 162
- 6.8 finishRefresh 165
- 7 第7章 AOP 167
 - 7.1 动态 AOP 使用示例 167
 - 7.2 动态 AOP 自定义标签 169
 - 7.2.1 注册 AnnotationAwareAspectJ AutoProxyCreator 170
 - 7.3 创建 AOP 代理 173
 - 7.3.1 获取增强器 176
 - 7.3.2 寻找匹配的增强器 186
 - 7.3.3 创建代理 187
 - 7.4 静态 AOP 使用示例 201
 - 7.5 创建 AOP 静态代理 203
 - 7.5.1 Instrumentation 使用 203
 - 7.5.2 自定义标签 207
 - 7.5.3 织入 209
- 第二部分 企业应用
- 8 第8章 数据库连接 JDBC 214
 - 8.1 Spring 连接数据库程序实现 (JDBC) 215
 - 8.2 save/update 功能的实现 217
 - 8.2.1 基础方法 execute 219
 - 8.2.2 Update 中的回调函数 223
 - 8.3 query 功能的实现 225

- 8.4 queryForObject 229
- 9 第 9 章 整合 MyBatis 231**
 - 9.1 MyBatis 独立使用 231
 - 9.2 Spring 整合 MyBatis 235
 - 9.3 源码分析 237
 - 9.3.1 sqlSessionFactory 创建 237
 - 9.3.2 MapperFactoryBean 的创建 241
 - 9.3.3 MapperScannerConfigurer 244
- 10 第 10 章 事务 254**
 - 10.1 JDBC 方式下的事务使用示例 254
 - 10.2 事务自定义标签 257
 - 10.2.1 注册 InfrastructureAdvisor AutoProxyCreator 257
 - 10.2.2 获取对应 class/method 的增强器 261
 - 10.3 事务增强器 269
 - 10.3.1 创建事务 271
 - 10.3.2 回滚处理 281
 - 10.3.3 事务提交 287
- 11 第 11 章 SpringMVC 291**
 - 11.1 SpringMVC 快速体验 291
 - 11.2 ContextLoaderListener 295
 - 11.2.1 ServletContextListener 的使用 295
 - 11.2.2 Spring 中的 ContextLoader Listener 296
 - 11.3 DispatcherServlet 300
 - 11.3.1 servlet 的使用 301
 - 11.3.2 DispatcherServlet 的初始化 302
 - 11.3.3 WebApplicationContext 的初始化 304
 - 11.4 DispatcherServlet 的逻辑处理 320
 - 11.4.1 MultipartContent 类型的 request 处理 326
 - 11.4.2 根据 request 信息寻找对应的 Handler 327
 - 11.4.3 没找到对应的 Handler 的错误处理 331
 - 11.4.4 根据当前 Handler 寻找对应的 HandlerAdapter 331
 - 11.4.5 缓存处理 332
 - 11.4.6 HandlerInterceptor 的处理 333
 - 11.4.7 逻辑处理 334
 - 11.4.8 异常视图的处理 334
 - 11.4.9 根据视图跳转页面 335
- 12 第 12 章 远程服务 340**
 - 12.1 RMI 340
 - 12.1.1 使用示例 340
 - 12.1.2 服务端实现 342
 - 12.1.3 客户端实现 350
 - 12.2 HttpInvoker 355
 - 12.2.1 使用示例 356
 - 12.2.2 服务端实现 357
 - 12.2.3 客户端实现 361
- 13 第 13 章 Spring 消息 367**
 - 13.1 JMS 的独立使用 367
 - 13.2 Spring 整合 ActiveMQ 369
 - 13.3 源码分析 371
 - 13.3.1 JmsTemplate 372
 - 13.3.2 监听器容器 376

第一部分 核心实现



第 1 章 Spring 整体架构和环境搭建

Spring 是于 2003 年兴起的一个轻量级的 Java 开源框架，由 Rod Johnson 在其著作《Expert One-On-One J2EE Development and Design》中阐述的部分理念和原型衍生而来。Spring 是为了解决企业应用开发的复杂性而创建的，它使用基本的 JavaBean 来完成以前只可能由 EJB 完成的事情。然而，Spring 的用途不仅限于服务器端的开发。从简单性、可测试性和松耦合的角度而言，任何 Java 应用都可以从 Spring 中受益。

1.1 Spring 的整体架构

Spring 框架是一个分层架构，它包含一系列的功能要素，并被分为大约 20 个模块，如图 1-1 所示。

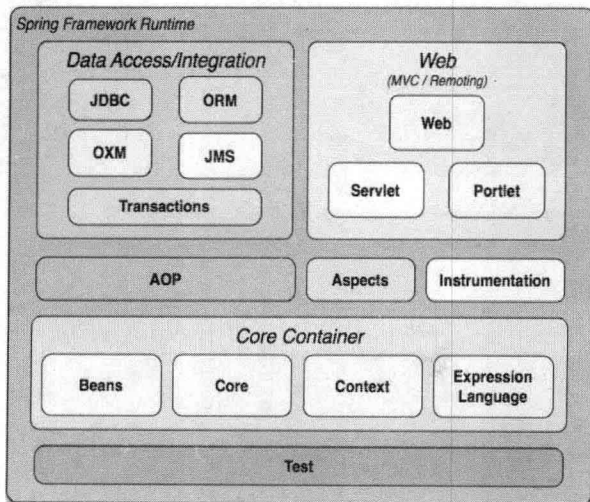


图 1-1 Spring 整体架构图

这些模块被总结为以下几部分。

(1) Core Container。

Core Container (核心容器) 包含有 Core、Beans、Context 和 Expression Language 模块。

Core 和 Beans 模块是框架的基础部分, 提供 IoC (转控制) 和依赖注入特性。这里的基础概念是 BeanFactory, 它提供对 Factory 模式的经典实现来消除对程序性单例模式的需要, 并真正地允许你从程序逻辑中分离出依赖关系和配置。

- Core 模块主要包含 Spring 框架基本的核心工具类, Spring 的其他组件都要使用到这个包里的类, Core 模块是其他组件的基本核心。当然你也可以在自己的应用系统中使用这些工具类。
- Beans 模块是所有应用都要用到的, 它包含访问配置文件、创建和管理 bean 以及进行 Inversion of Control / Dependency Injection (IoC/DI) 操作相关的所有类。
- Context 模块构建于 Core 和 Beans 模块基础之上, 提供了一种类似于 JNDI 注册器的框架式的对象访问方法。Context 模块继承了 Beans 的特性, 为 Spring 核心提供了大量扩展, 添加了对国际化 (例如资源绑定)、事件传播、资源加载和对 Context 的透明创建的支持。Context 模块同时也支持 J2EE 的一些特性, 例如 EJB、JMX 和基础的远程处理。ApplicationContext 接口是 Context 模块的关键。
- Expression Language 模块提供了一个强大的表达式语言用于在运行时查询和操纵对象。它是 JSP 2.1 规范中定义的 unified expression language 的一个扩展。该语言支持设置/获取属性的值, 属性的分配, 方法的调用, 访问数组上下文 (accessing the context of arrays)、容器和索引器、逻辑和算术运算符、命名变量以及从 Spring 的 IoC 容器中根据名称检索对象。它也支持 list 投影、选择和一般的 list 聚合。

(2) Data Access/Integration。

Data Access/Integration 层包含有 JDBC、ORM、OXM、JMS 和 Transaction 模块, 其中:

- JDBC 模块提供了一个 JDBC 抽象层, 它可以消除冗长的 JDBC 编码和解析数据库厂商特有的错误代码。这个模块包含了 Spring 对 JDBC 数据访问进行封装的所有类。
- ORM 模块为流行的对象-关系映射 API, 如 JPA、JDO、Hibernate、iBatis 等, 提供了一个交互层。利用 ORM 封装包, 可以混合使用所有 Spring 提供的特性进行 O/R 映射。如前边提到的简单声明性事物管理。

Spring 框架插入了若干个 ORM 框架, 从而提供了 ORM 的对象关系工具, 其中包括 JDO、Hibernate 和 iBatisSQL Map。所有这些都遵从 Spring 的通用事务和 DAO 异常层次结构。

- OXM 模块提供了一个对 Object/XML 映射实现的抽象层, Object/XML 映射实现包括 JAXB、Castor、XMLBeans、JiBX 和 XStream。
- JMS (Java Messaging Service) 模块主要包含了一些制造和消费消息的特性。
- Transaction 模块支持编程和声明性的事物管理, 这些事物类必须实现特定的接口, 并且对所有的 POJO 都适用。

(3) 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。所以，Spring 框架支持与 Jakarta Struts 的集成。Web 模块还简化了处理多部分请求以及将请求参数绑定到域对象的工作。Web 层包含了 Web、Web-Servlet、Web-Struts 和 Web-Portlet 模块，具体说明如下。

- Web 模块：提供了基础的面向 Web 的集成特性。例如，多文件上传、使用 servlet listeners 初始化 IoC 容器以及一个面向 Web 的应用上下文。它还包含 Spring 远程支持中 Web 的相关部分。
- Web-Servlet 模块 web.servlet.jar：该模块包含 Spring 的 model-view-controller (MVC) 实现。Spring 的 MVC 框架使得模型范围内的代码和 web forms 之间能够清楚地分离开来，并与 Spring 框架的其他特性集成在一起。
- Web-Struts 模块：该模块提供了对 Struts 的支持，使得类在 Spring 应用中能够与一个典型的 Struts Web 层集成在一起。注意，该支持在 Spring 3.0 中是 deprecated 的。
- Web-Portlet 模块：提供了用于 Portlet 环境和 Web-Servlet 模块的 MVC 的实现。

(4) AOP。

AOP 模块提供了一个符合 AOP 联盟标准的面向切面编程的实现，它让你可以定义例如方法拦截器和切点，从而将逻辑代码分开，降低它们之间的耦合性。利用 source-level 的元数据功能，还可以将各种行为信息合并到你的代码中，这有点像 .Net 技术中的 attribute 概念。

通过配置管理特性，Spring AOP 模块直接将面向切面的编程功能集成到了 Spring 框架中，所以可以很容易地使 Spring 框架管理的任何对象支持 AOP。Spring AOP 模块为基于 Spring 的应用程序中的对象提供了事务管理服务。通过使用 Spring AOP，不用依赖 EJB 组件，就可以将声明性事务管理集成到应用程序中。

- Aspects 模块提供了对 AspectJ 的集成支持。
- Instrumentation 模块提供了 class instrumentation 支持和 classloader 实现，使得可以在特定的应用服务器上使用。

(5) Test。

Test 模块支持使用 JUnit 和 TestNG 对 Spring 组件进行测试。

1.2 环境搭建

Spring 已经将源码从 svn 迁移到了 GitHub。而且也改为基于 Gradle 的构建来构建项目，它取代了之前的 Ant+Ivy 系统，所以要构建 Spring 源码环境首先要安装 GitHub 以及 Gradle。

1.2.1 安装 GitHub

首先读者需要到 GitHub 官网去下载安装包，其中 Windows 系统对应的版本下载地址为：

<http://windows.github.com/>，下载后双击进行安装。安装成功后，快捷菜单中会出现 GitHub 的菜单，如图 1-2 所示。

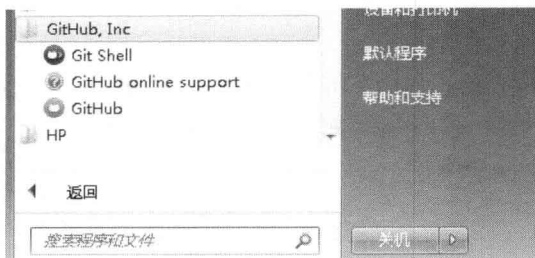


图 1-2 HitHub 安装成功后的启动菜单

1.2.2 安装 Gradle

Gradle 是一个基于 Groovy 的构建工具，它使用 Groovy 来编写构建脚本，支持依赖管理和多项目创建，类似 Maven，但比其更加简单轻便。Gradle 为 Ivy 提供了一个 layer，提供了 build-by-convention 集成，而且它还让你获得许多类似 Maven 的功能。你可以从 <http://www.gradle.org/downloads> 页面下载 Gradle，下载后将文件解压放到指定目录中（笔者放在了 C:\Program Files 目录下），然后开始进行环境变量的配置。

- (1) 根据对应目录创建 GRADLE_HOME 系统变量，如图 1-3 所示。
- (2) 将系统变量加入到 path 中，如图 1-4 所示。

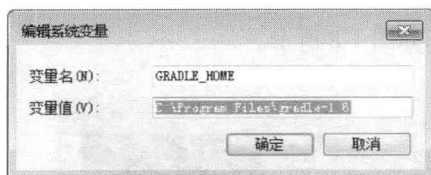


图 1-3 创建对应于 Gradle 的系统变量

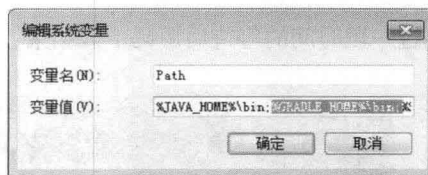


图 1-4 将 Gradle 对应的系统变量加入 path 中

- (3) 测试。

当完成系统变量的配置后打开命令窗口输入命令“gradle -version”，如果安装成功会出现 Gradle 对应的版本信息，如图 1-5 所示。

```
C:\Users\Administrator>gradle -version

Gradle 1.6
```

图 1-5 测试 Gradle 的环境变量配置

1.2.3 下载 Spring

因为 Spring 源码是通过 GitHub 进行管理的，所以我们首先打开 GitHub，单击快捷菜单中的“Git Shell”选项，如图 1-6 所示。

打开 GitHub 后，你可以通过 `cd` 命令将当前操作目录转换到我们想要存储源码的目录，例如，想要将下载的源码存储到 `e:\test` 下，则可以执行“`cd e:\test`”。

输入以下命令：

```
git clone git://github.com/SpringSource/Spring-framework.git
```

其中“`git://github.com/SpringSource/Spring-framework.git`”为 Spring 的源码地址。执行命令后便进入源码下载状态，如图 1-7 所示。

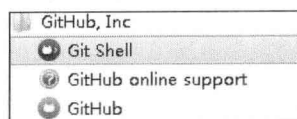


图 1-6 启动 GitHub 启动菜单

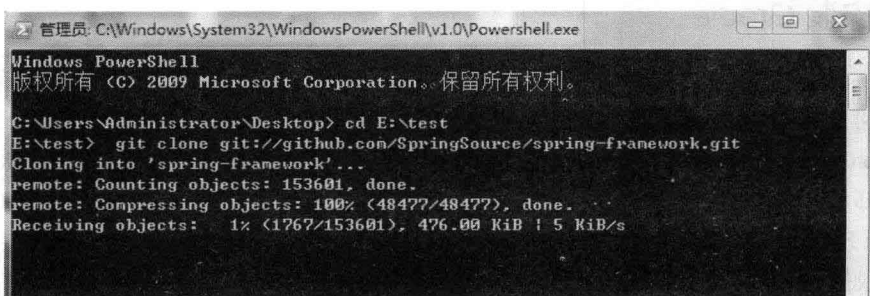


图 1-7 使用 GitHub 开始下载源码

经过一段时间的等待后源码下载结束，窗口状态如图 1-8 所示。

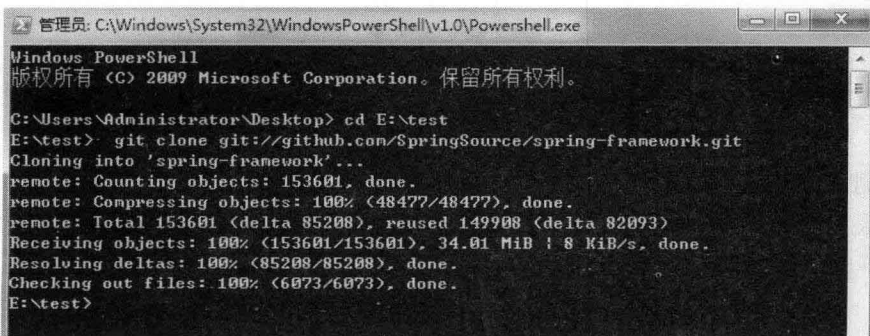


图 1-8 源码下载结束的窗口显示

而这时候我们去查看，对应的文件夹下已经存在了相应的源码信息，如图 1-9 所示。

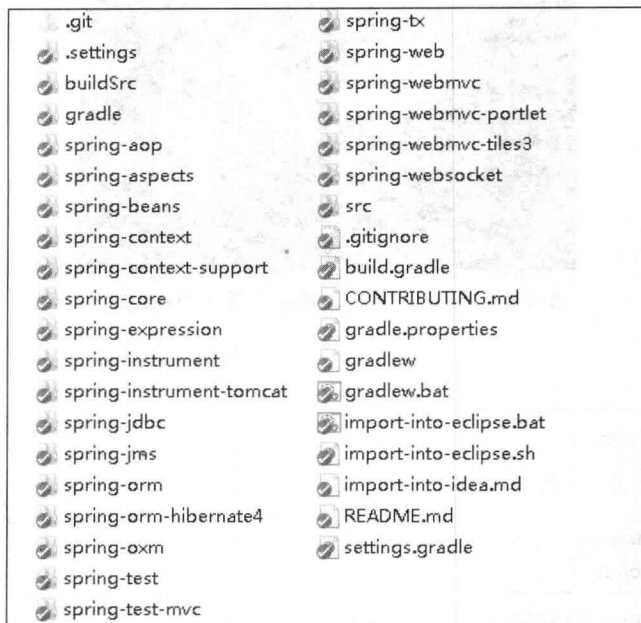


图 1-9 下载的 Spring 源码

但是当前的源码并不可以直接导入 Eclipse 中，我们还需要将源码转换为 Eclipse 可以读取的形式。网上有各种各样的方法，其中出现最多的是告诉大家将所有工程一次性的编译、导入，但是笔者并不推荐这样的方式，因为这样会耗费大量的时间，而且当存在编译错误的时候你不得不重新编译。笔者建议只对我们感兴趣的工程进行 Eclipse 工程转换，比如我们想要查看 Spring 事务部分的源码，打开命令窗口，将当前目录切换至源码所在目录，例如，这里是 Spring-tx 文件夹下，执行命令“gradle cleanidea eclipse”，当窗口出现如下状态说明已经开始执行转换过程，如图 1-10 所示。

```
D:\test\spring-framework\spring-tx>gradle cleanidea eclipse
:buildSrc:compileJava UP-TO-DATE
:buildSrc:compileGroovy UP-TO-DATE
:buildSrc:processResources UP-TO-DATE
```

图 1-10 Spring 源码转换至 eclipse 工程

经过一段时间后转换成功，如图 1-11 所示。

这时，我们再查看对应的文件夹会发现，已经出现了作为 Eclipse 工程所必须的.project 与.classpath 文件了，如图 1-12 所示。

打开 Eclipse，将工程导入，导入后如图 1-13 所示。