HZ BOOKS
华章IT

Web开发
技术丛书

看透 Spring MVC

源代码分析与实践

韩路彪 著

Spring MVC in Action
Source Code and Practice

- 国内资深Web开发专家根据Spring MVC最新技术撰写，基于实际生产环境，从基础知识、源代码和实战3个维度对Spring MVC的结构和实现进行详细讲解
- 全面介绍Spring MVC的架构、原理、核心概念和操作，通过案例完整呈现Tomcat的实现，系统总结Spring MVC九大组件的处理以及常用的技巧和最佳实践

机械工业出版社
China Machine Press

Web开发
技术丛书

看透 Spring MVC

源代码分析与实践

Spring MVC in Action
Source Code and Practice

韩路彪 著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

看透 Spring MVC：源代码分析与实践 / 韩路彪著 . —北京：机械工业出版社，2015.11
（Web 开发技术丛书）

ISBN 978-7-111-51668-2

I. 看… II. 韩… III. JAVA 语言－程序设计 IV. TP312

中国版本图书馆 CIP 数据核字（2015）第 233129 号

看透 Spring MVC：源代码分析与实践

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：秦 健 责任校对：董纪丽
印 刷：中国电影出版社印刷厂 版 次：2016 年 1 月第 1 版第 1 次印刷
开 本：186mm × 240mm 1/16 印 张：20
书 号：ISBN 978-7-111-51668-2 定 价：69.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88379426 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzit@hzbook.com

献　给

父亲韩志荣

前　言 Preface

当前网络正在改变人们生活的方方面面，从企业内部的管理和运营到我们个人的吃穿住行，所有这些都跟网络有着密切联系。不过这一切才刚刚开始，未来的网络将会给人们带来更多的惊喜，特别是在2015年“两会”中将“互联网+”纳入我国的发展战略之后，网络未来几年的高速发展更会超出我们的想象。

在网络技术中基于浏览器的B/S结构无论在PC端还是手机端都充当着至关重要的角色。PC端自不必说，手机中很多应用虽然是以APP的形式存在，但它采用的还是B/S结构，如今日头条、微信的朋友圈等，这些应用在内部封装了浏览器，后端仍然是Web站点。

在大型网站和复杂系统的开发中，Java无疑具有很大的优势，而在Java的Web框架中Spring MVC以其强大的功能和简单且灵活的用法受到越来越多开发者的青睐。

Spring MVC入门很简单，但是要想真正使用好却并非易事，而且现在也没有全面、深入的使用资料，以致在实际使用的过程中程序员经常会遇到各种各样的问题而不知道如何解决。对Spring MVC这样的开源项目来说，最好的学习方法当然是分析它的源代码，分析透源代码不仅可以让我们更灵活地使用Spring MVC来开发高质量的产品，而且可以学习到其中的很多优秀的编程技巧和设计理念。

本书除了分析Spring MVC的源代码，还系统地介绍了各种网站架构的演变以及Web开发中所涉及的协议和Tomcat的实现方法，现在很多程序员都想了解这方面的知识，但苦于缺乏通俗易懂的资料，而且这些也是程序员达到更高的层次所需要的知识。

通过本书你可以得到什么

- 系统学习网站的各种架构以及相应问题的解决方案。
- 零基础系统学习Web底层协议及其实现方法。
- 系统、深入地理解Spring MVC，为灵活开发高质量产品打下基础。
- 学习Spring MVC的编程技巧和设计理念，提高自己综合思考、整体架构的能力。
- 学习到笔者设计的一套分析源码的方法——器用分析法，古人说“授人以鱼不如授人以渔”，虽然这套方法并不复杂但是对于分析复杂的代码却非常有用。

当然，并不是说像看小说一样翻一遍本书就可以获得这么多东西，这需要大家真正沉下心来认真地去看，而且最好能对照着源代码去看。俗话说“磨刀不误砍柴工”，分析源代码就是磨刀的过程，是真正提升自己实力的过程，就像武术里的内功修炼一样，只有花足够的时间和精力才能到达一定的高度，这就是我们经常说的“功夫”，当功夫达到一定的高度时很多棘手的问题就可以轻而易举地解决了。

本书读者对象

- 有 Java 编程基础，想学习 JavaWeb 开发的读者。
- 有 JavaWeb 开发经验，想学习 Spring MVC 的读者。
- 有基础 Spring MVC 开发经验，想深入学习的读者。
- 有丰富 Spring MVC 开发经验，想学习 Spring MVC 底层代码的读者。
- 想自己开发 Spring MVC 插件的读者。

本书特点

- 本书从最底层的架构和协议开始讲解，即使没有太多开发经验的读者也可以理解，同时由于本书包含的内容全面而且深入，所以即使有丰富 Web 开发经验的读者读过之后也会有所收获。
- 本书采用了总分总的结构，首先概述全书内容，让大家在脑子里建立起整个框架，然后再对每个点展开分析，最后总结。这就好像一栋建筑，首先把它的整体结构展示给大家，然后再具体介绍每个细节，这样就可以让大家思路清晰而不至于迷失方向。这种模式最符合人的认知方式，所以不仅仅适用于学习，而且可以使用到别的很多地方，比如，进入一个新公司后（特别是大型公司），首先要了解一下公司都有哪些部门，各个部门之间是怎么协调配合的，弄明白整体结构之后再思考自己的业务，这样就可以理解得更加深，做得更好，如果有机会再多了解点其他部门的业务，这样成长得就会更快。
- 本书讲解的过程通俗易懂、深入浅出，对于不容易理解的内容，通过简单的例子让大家一目了然。在分析源代码的过程中还对一些代码分析了 Spring MVC 为什么要那么处理，那么处理有哪些好处，有些地方还为大家指出了需要注意的问题、可以实现的需求以及可以借鉴的东西等内容。

本书结构安排

本书一共分为四篇。

第一篇首先讲解了网站基础知识，包括网站架构的演变以及每种架构所针对的问题、Web 底层的协议以及简单的实现方法，最后分析了 Tomcat 的实现方法，这样可以让大家对 Web 有

整体而且深入的理解，从而为分析 Spring MVC 打下坚实的基础。

第二篇分析了 Spring MVC 的整体结构，帮助大家理解请求是怎么到 Spring MVC 中的，以及在 Spring MVC 中都做了些什么，这部分主要是帮大家建立框架，让大家对 Spring MVC 的整体结构了然于胸，在后面内容中只需要对具体的组件进行分析即可。

第三篇分别对 Spring MVC 中的 9 大组件进行了分析，这部分又分了两步：第一步先分析了每个组件的接口、作用和用法，让大家对每个组件有个大体的认识；第二步详细分析了 9 大组件的实现。

第四篇对 Spring MVC 的整体结构做了总结，并对异步请求的原理及用法做了补充。总结分为两步，首先是对 Spring MVC 的结构进行总结，并从更高的层次分析其设计理念；然后通过跟踪一个具体的请求帮助大家整体梳理请求的处理过程。异步请求是一块相对独立的内容，如果将其放入 Spring MVC 的分析过程中将增加大家对 Spring MVC 的理解难度，所以在最后对其进行单独讲解。

本书源代码可以到 kantou.excelib.com/springmvc 下载。

致谢

我最想感谢的就是我的父亲韩志荣，正是因为他的大力支持和背后的默默付出才让笔者可以将更多的时间和精力放在本书的创作上，从而让本书可以在保证质量的前提下以最快的速度跟大家见面。

虽然笔者已经尽了自己最大的努力，但是受水平所限，难免会有遗漏或者讲解不够准确的地方，还请大家批评指正。如果大家通过本书可以对 Web 开发、对 Spring MVC 的理解以及对设计的理念有些许收获，那将是笔者最感到欣慰的事情。

Contents 目　录

第一篇

网站基础知识

本篇主要给大家介绍网站的基础知识，为后面具体分析 Spring MVC 打下基础。内容主要包括架构的演变、Web 中涉及的协议、协议的实现方法、Java 中的 Servlet 以及对一个完整的产品 Tomcat 的分析等 5 部分。

本篇的很多内容，如底层协议和 Tomcat 的实现方法，在正常做开发的时候并不会直接使用到，不过理解了之后可以让我们在进行具体开发的时候更加得心应手，就好像数学中的基本运算，我们不需要知道原理也可以借助计算器计算出结果，但是如果明白了其中的原理就可以对计算带来很多帮助。比如，可以预先大概估计计算结果，当计算器的计算结果偏差很大时就可以看出来；可以使用一些简单的计算方法；还可以通过对具体内容的学习学到一些优秀思想，思想本身是很难学习的，需要通过一定的载体才可以传播，底层的知识就是这样的载体。

现在社会中普遍注重创新，其实创新是建立在扎实的基础之上的，如果没有扎实的基础就很难做出合理而且易用的创建成果。所以本篇的内容虽然在开发中一般不会直接使用到，但是对于提高自己的能力非常重要。

Chapter 1

第 1 章 网站架构及其演变过程

本章介绍网站的架构及其演变的过程。现在大型网站的架构变得越来越复杂，不过架构的演变过程并不是没有规律的，它们是在遇到相应问题之后为了解决问题才演变出来的。本章首先从软件的三大类型说起，然后介绍各种架构的演变过程及其背后的本质。

1.1 软件的三大类型

记得在上学的时候，计算机考试中很经典的一道题是“开电脑时应该先开主机电源还是先开显示器电源”，那个时代的软件主要以单机软件为主，如画图板、五笔打字等，当时学习使用电脑跟学习打字基本上是一个概念，那些不需要联网的单机软件就是最开始的软件。

后来有的程序需要统一管理软件中使用的数据，所以就将保存数据的数据库统一存放在一台主机中，所有的用户在需要数据时都要从主机获取，这时就分出了客户端和服务端，用户安装的软件叫客户端（Client），统一管理数据的主机中的软件就叫服务端（Server），这种结构就叫 CS 结构。再后来这种结构的服务端就不只是管理数据了，另外还可以处理一些业务逻辑，哪些业务放到客户端处理，哪些业务放到服务端处理就是见仁见智的问题了。业务放到服务端统一处理可以提供更好的安全性和稳定性而且升级比较容易，不过服务器的负担就增加了；业务放到客户端处理可以将负担分配到每个用户的机器上，从而可以节省服务器的资源，不过安全性和稳定性可能会有一些问题，而且升级也比较麻烦，每个用户安装的客户端程序都需要升级。另外，为了节省网络资源，通过网络传输的数据应该尽量少。CS 结构如图 1-1 所示。

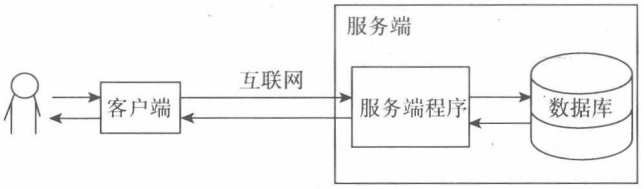

图 1-1　CS 结构图

CS 结构的程序已经可以完成网络通信了，不过使用起来还是有点麻烦，首先软件提供商需要同时开发客户端和服务端两套软件；其次每个用户在使用时都需要单独安装客户端软件，而且升级的时候也需要每个用户都进行升级。为了解决这个问题而设计了统一的客户端，而且默认安装在用户电脑里面，这就是我们电脑中的浏览器（Browser），而且一个浏览器可以访问所有同种类型的网站，当然它主要用作展示数据，具体业务处理是在不同的服务端进行的，这种结构就叫 BS 结构。BS 结构除了提供了统一的客户端，还根据相应协议和标准提供通用的服务器程序，服务器程序统一处理数据连接、封装和解析等工作。BS 结构如图 1-2 所示。

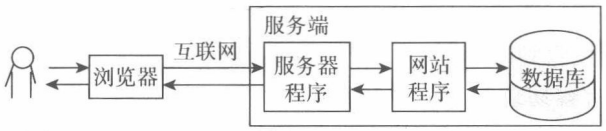

图 1-2　BS 结构图

这就是软件的三大类型：单机类型、CS 类型和 BS 类型。在这三种类型中，因为 BS 类型开发简单、使用方便而且功能强大，所以现在使用最广，当然并不是说 BS 结构是最好的，具体使用什么结构还需要根据实际的需求来决定，比如，现在我们电脑中的记事本、Office 以及压缩软件等都是单机软件，而它们使用得也非常广泛，另外 BS 结构虽然比 CS 结构在开发和使用上都简单，但是 BS 结构的灵活性和处理效率都不如 CS 结构，所以像 QQ、大型游戏等软件使用的还是 CS 结构。

1.2　基础的结构并不简单

前面介绍的 BS 结构是最基础的结构，不过即使这种最基础的结构的底层实现也并不简单，因为它需要通过互联网传输数据，而互联网是一个错综复杂的网络，其中包含的节点不计其数，而且每两个节点之间的距离以及连接的路线都是不确定的，数据在传输的过程中还可能会丢失，所以非常复杂。所有问题都有它对治的方法，对于复杂问题的对治方法就是将其分解成多个简单的问题，然后通过解决每个简单问题，最终解决复杂问题。BS 结构网络传输的分解方式有两种：一种是标准的 OSI 参考模型，另一种是 TCP/IP 参考模型。它们的分层方式及对应关系如图 1-3 所示。

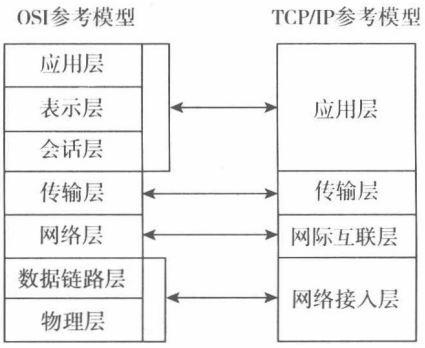

图 1-3 OSI 和 TCP/IP 分层模型及对应关系

OSI 参考模型一共分 7 层，不过它主要用于教学，实际使用中更多的是 TCP/IP 的 4 层模型。对于 TCP/IP 的 4 层模型可以简单地理解为：

- 网络接入层：将需要相互连接的节点接入网络中，从而为数据传输提供条件。
- 网际互联层：找到要传输数据的目标节点。
- 传输层：实际传输数据。
- 应用层：使用接收到的数据。

这种分层模型非常容易理解，就好像我们要在网上买东西，首先要确定自己所在的位置有相应的快递，这就相当于网络接入层，然后需要告诉卖家地址，地址就相当于网际互联层，快递送货相当于传输层，最后我们收到货物之后拆包使用就相当于应用层。

对于广泛使用的东西就需要制定相应的标准，没有规矩不成方圆，如果都按自己的想法去做就乱套了。对一个小作坊来说，做事情可以比较随意，但是一个大型公司就需要有很多制度来规范做事情的流程了。由于网络传输应用非常广泛，所以需要大家都遵守的规矩，不过网络传输中的这些规矩并不是强制性的，所以不叫制度也不叫标准而叫协议，其实 TCP/IP 参考模型也可以看作一种协议。BS 结构中 TCP/IP 模型中的网络接入层没有相应协议，网际互联层是 IP 协议，传输层是 TCP 协议，应用层是 HTTP 协议。

另外在 BS 结构中还使用到了 DNS 协议，而且在 HTTP 上层还有相关的规范，如 Java Web 开发中使用的是 Servlet 标准。

数据传输的本质就是按照晶振震动周期或者其整数倍来传输代表 0/1 的高低电平，传输过程中最核心就是各种传输协议，对直接连接的硬件来说就是各种总线协议，对网络传输来说就是网络协议，如果将传输的协议弄明白了，那么也就掌握了传输的核心，第 2 章会介绍 BS 结构中常用的协议和标准。下面先接着看网站架构的演变过程，开发一套前面介绍的那种 BS 结构的程序并非难事，特别是使用现在成形的框架来做就更加简单了，只需要写好核心的业务就可以了。不过这种基础架构的网站虽然可以用但并不代表好用，除了用户交互（那是另外一个话题），最重要的就是速度问题。如果打开一个连接的时间都可以喝完一杯咖啡，那样的系统能不能使用就看每个人自己的理解了。不过无论怎么理解，如果不是企业内部办公必

须使用的系统，也不是像 12306 那种具有垄断资源的系统，相信大部分人是不会有那个耐心去等待的。解决速度问题的核心主要就是解决海量数据操作问题和高并发问题，网站复杂的架构就是从这两个问题演变出来的。

1.3　架构演变的起点

基础架构中服务端就一台主机，其中存储了应用程序和数据库，刚上线时是没有问题的，当数据和流量变得越来越大的时候就难以应付了，这时候就需要将应用程序和数据库分别放到不同的主机中，其结构如图 1-4 所示。

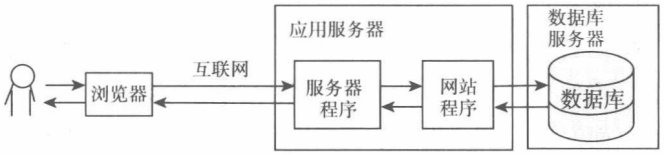

图 1-4　应用和数据分离结构图

1.4　海量数据的解决方案

现在无论是企业的业务系统还是互联网上的网站程序都面临着数据量大的问题，这个问题如果解决不好将严重影响系统的运行速度，下面就针对这个问题的各种解决方案进行系统介绍。

1.4.1　缓存和页面静态化

数据量大这个问题最直接的解决方案就是使用缓存，缓存就是将从数据库中获取的结果暂时保存起来，在下次使用的时候无需重新到数据库中获取，这样可以大大降低数据库的压力。

缓存的使用方式可以分为通过程序直接保存到内存中和使用缓存框架两种方式。程序直接操作主要是使用 Map，尤其是 ConcurrentHashMap，而常用的缓存框架有 Ehcache、Memcache 和 Redis 等。缓存使用过程中最重要问题是什么时候创建缓存和缓存的失效机制。缓存可以在第一次获取的时候创建也可以在程序启动和缓存失效之后立即创建，缓存的失效可以定期失效，也可以在数据发生变化的时候失效，如果按数据发生变化让缓存失效，还可以分粗粒度失效和细粒度失效。

多知道点

缓存中空数据的管理方法

如果缓存是在第一次获取的时候创建的，那么在使用缓存的时候最好将没有数据的缓存使用特定的类型值来保存，因为这种方式下如果从缓存中获取不到数据就会从数据库中获取，如果数据库中本来就没有相应的数据就不会创建缓存，这样将每次都会查询数据

> 库。比如有个专门保存文章评论的缓存，不同的评论按照不同文章的 Id 来保存，如果有一篇文章本来就没有评论，那么就没有相应的缓存或者缓存的值为 null，这样程序在每次调用这篇文章的评论时都会查询数据库。这就没起到缓存的作用，我们可以创建一个专门的类（如 NoComment）来保存没有评论的缓存，这样程序从缓存中查询后就可以知道是还没有创建缓存还是本来就没有评论内容。

不过缓存也不是什么情况都适用，它主要用于数据变化不是很频繁的情况。而且如果是定期失效（数据修改时不失效）的失效机制，实时性要求也不能太高，因为这样缓存中的数据和真实数据可能会不一致。如果是文章的评论则关系不是很大，但如果是企业业务系统中要生成报表的数据则问题就大了。

跟缓存相似的另外一种技术叫页面静态化，它在原理上跟缓存非常相似，缓存是将从数据库中获取到的数据（当然也可以是别的任何可以序列化的东西）保存起来，而页面静态化是将程序最后生成的页面保存起来，使用页面静态化后就不需要每次调用都重新生成页面了，这样不但不需要查询数据库，而且连应用程序处理都省了，所以页面静态化同时对数据量大和并发量高两大问题都有好处。

页面静态化可以在程序中使用模板技术生成，如常用的 Freemarker 和 Velocity 都可以根据模板生成静态页面，另外也可以使用缓存服务器在应用服务器的上一层缓存生成的页面，如可以使用 Squid，另外 Nginx 也提供了相应的功能。

1.4.2 数据库优化

要解决数据量大的问题，是避不开数据库优化的。数据库优化可以在不增加硬件的情况下提高处理效率，这是一种用技术换金钱的方式。数据库优化的方法非常多，常用的有表结构优化、SQL 语句优化、分区和分表、索引优化、使用存储过程代替直接操作等，另外有时候合理使用冗余也能获得非常好的效果。

表结构优化

表结构优化是数据库中最基础也是最重要的，如果表结构优化得不合理，就可能导致严重的性能问题，具体怎么设计更合理也没有固定不变的准则，需要根据实际情况具体处理。

SQL 语句优化

SQL 语句优化也是非常重要的，基础的 SQL 优化是语法层面的优化，不过更重要的是处理逻辑的优化，这也需要根据实际情况具体处理，而且要和索引缓存等配合使用。不过 SQL 优化有一个通用的做法就是，首先要将涉及大数据的业务的 SQL 语句执行时间详细记录下来，其次通过仔细分析日志（同一条语句对不同条件的执行时间也可能不同，这点也需要仔细分析）找出需要优化的语句和其中的问题，然后再有的放矢地优化，而不是不分重点对每条语