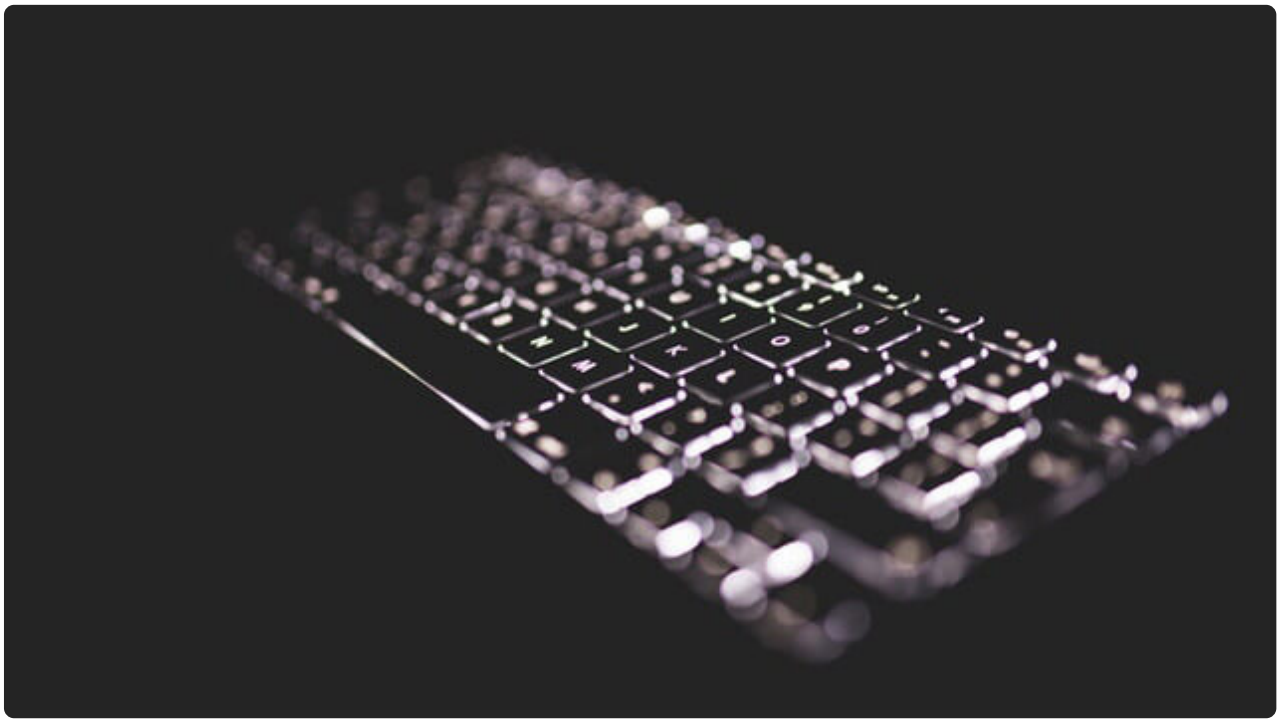
37 结束语

更新时间：2019-12-24 14:14:14



> 机会不会上门来找人，只有人去找机会。
>
> ——狄更斯

为期半年的 Java 并发专栏终于落下了帷幕。很感谢一直陪伴我走完这段历程的读者朋友，是你们的鼓励让我坚持完成写作；是你们的发问，让我钻研得更为彻底；是你们的坚持学习，坚定了我写作的信心。此外也很感谢慕课网的小编给我提出各种建议和鼓励。最后感谢我的爱人和父母，谢谢家人对家庭的照顾，让我有更多时间来学习和创作。

作为专栏最后一篇，我想从三个方面和大家聊一聊。

1. Java 并发编程的学习；
2. 关于学习 ；
3. 关于写作。

1、Java 并发编程的学习

Java 并发编程的书籍和教程市面上有很多，这反而造成大家在学习 Java 并发时不知道该选择哪本书籍。在这里我可以谈一谈我是如何学习的。我最开始看的是《Java 并发编程实战》。这本书绝对是经典中的经典。作者都是耳熟能详的大师。不过对于想要学习 Java 并发的新手来说，我并不推荐上来就学习此书。这一点在此书的序中也被提到：“本书并非是对并发的入门介绍”。对于新人，我推荐先看一些国内作者的书籍或者专栏，无论大纲安排还是语言运用，都更为符合国人的习惯，更为深入浅出。当你有了一定的并发基础后，一定要反复阅读《 Java 并发编程实战》。此时读此书，才能深刻理解其中的精髓。而不只是停留在语言层面的 “理解”。这本书的作者参与了 java 并发的设计和编写，所以作者对 java 并发的理解肯定无人能及。这本书的奇妙之处在于，随着你对 Java 并发学习的深入，每次读此书，都会汲取到更多的知识。但是如果你上来就通过此书学习并发，会让你有种云里雾里的感觉。

我们学习 Java 并发编程的最终目的和价值，还是运用于实战之中。所以关于 Java 并发编程的实战学习是必不可少的。虽然我在专栏中提到学习的目的之一是为了面试。但其实这也是被迫无奈。我们学习的任何新技术，如果不运用于生产，都是没有意义的。所以在学习完 Java 并发后，一定要学习并发的设计模式以及实战。另外最好能够看一看开源框架中对于多线程的使用，读一下源代码。然后在自己的代码中去尝试使用。

另外在 Java8 中，通过 lambda 表达式和流 API，可以很方便的并行处理流。这部分内容并不在本专栏范围内。但是 lambda 已经在 Java 领域应用的十分广泛，所以十分建议去认真学习，并且运用到日常开发中。

2、关于学习

写了这么久的并发编程，今天想聊聊并发之外的事情。首先想聊一聊我对学习的一些想法。

选择软件开发这个工作，也就意味着你需要终身学习。我做企业级开发已经 10 多年。还记得刚毕业 06 年左右，EJB 的书有很多。但过了没两年，EJB 就已经不是热点，取而代之的是满世界的 SSH 书籍，当时还是 Structs1。如何整合三大框架也算是个难题，居然也可以出本书来讲解。当时最流行的数据库是 Oracle，开源的数据库使用很少。那个时代，大家对框架还主要停留在使用层面。再往后，开始出现各种源代码分析的书籍，尤其是 Spring 居多，书很薄，但是卖的很贵。到了互联网时代，曾经的 SSH（Structs+Spring+Hibernate）变成了 SSM（Spring MVC+Spring+Mybatis）。再到后来的 Spring Cloud，微服务大行其道，分布式架构开始流行。与之而来的消息队列、分布式事务、容器等技术占据了主流。与此同时，大数据和人工智能单成一套体系，分别快速的发展。这些技术间也互有融合。云服务也给开发者带来了更多的便利，让开发者更能专注在软件开发本身。

从这个发展过程来看，不过 10 多年的功夫，技术已经换了一批又一批。从业者需要一直学习才能跟上技术的进步。这个行业是否过于辛苦了？其实并不是这样。有句话叫万变不离其宗。举个例子，无论是从 Struts1 到 Struts2，还是演变到 Spring MVC，都绕不开 HTTP 协议，都绕不开 MVC 设计模式，都绕不开 Servlet。如果只为了见效快，项目上用啥我就学啥，那么到最后一定是学不过来了。只有把根基打牢固，才能以不变应万变。无论用什么框架，底层使用的技术都是类似的。框架的应用固然重要，但是我们一定要再往下扎的深一点，扎的越深，学到的知识越不会轻易被淘汰。

我建议大家更多的学习底层知识和理论基础，不要把时间全花在框架学习上，那么只会让自己疲于奔命。底子打的牢靠，会让你从容应对技术的更新换代。

下面我想再谈谈如何学习一门新的技术。

首先，我们应该从实际需要出发，带着使用场景学习的效果更好。一般我会通过官网，了解这门技术的出现动机和他的优势，然后通过官网的文档目录对其有一个大概的了解。

接下来就可以跟着文档写一个 Hello World 程序，这样能让你对这门技术的工作机制有大概的了解。接下来会把这门技术最基础的知识点学习完成，当然是通过实践的方式，一步一步跟着教程和示例来做。一般来说我们学习新技术是因为项目上要使用，所以我们一般不会有特别充裕的时间让你从头学到尾。这个时候就需要你来分析一下你的使用场景，再结合文档看他通过什么方式解决你的问题。那么接下来就应该重点学习这些内容。其实就是带着我们的问题来看这门技术如何解决，然后再去针对性的学习。这样的学习是效果最好的。问题解决后，再花时间把其他章节学习完。我还是建议系统完整的学习，而不是用哪就只学那一小块。

其实好多时候我们学习完，并没有机会使用。学习这件事就变成了只有输入没有输出。不能形成闭环，效果自然也不会好。你会发现学习完如果不去使用，最多两个月就忘的一干二静了。如果项目中暂时用不到，我建议你换一种输出方式。比如可以把你学到的知识做分享，让自己掌握更牢固的同时，也帮助到其他同事。分享看似大家在从你这收获知识，但其实收获最大的还是分享者。不要怕讲的不够深入、不要怕自己会紧张、不要怕被同事问住。这些都无所谓，只要你自己认真准备了，你就有了收获。

如果你实在不愿意在众人面前分享。或者公司没有那么多分享的机会。你也可以选择写博客或者专栏。技术写作也是一项非常好的输出方式。而且你一个人就能完成，很适合比较宅的程序员。写作需要你把学习到的知识理解透彻，然后再换做自己的语言写出来。这个过程会让你把知识记忆的更为牢固。而且如果你以后有机会用到这门技术，只要翻看一下自己的博客，很快就能回忆起来。

既然聊到了写作，那么下面我就展开来说说写作这个事儿。

3、关于写作

我自认为还是个比较喜欢写作的人。但这半年的专栏写下来，也是让我疲惫不堪。我前不久参加了异步社区的作译者大会。还记得一个嘉宾分享时说到，选择了写作，意味着放弃了除写作外的一切。你没有时间看电影、看小说，没时间和朋友吃饭、聚会。甚至陪家人的时间都少了很多。我听完这位嘉宾的分享，十分身同感受。

我在写本专栏之初，想用一种新颖点的方式，通过漫画帮助理解记忆，但是没有充分考虑这部分的工作量。这导致开始写作后，占用了我大量的业余时间。除了写完文章外，每篇文章的配图还需要 1-2 个小时创作。于是我基本上半年多都是在 12 点以后才睡觉，周末经常是孩子上课，我在肯德基写文章。说实话，几度感觉要撑不下去了。尤其是写到 ConcurrentHashMap 源码分析。首先我要花大量时间把源码读懂，然后再通俗易懂的写出来。本来计划一节的内容，越写越多，最后写成了两节。这个时候工作上也很忙，我那段时间加班 9 点半到家，哄孩子睡觉到 10 点半，然后开始写文章到 1 点。早上 7 点起来送孩子上学。当时真的觉得自己要撑不下去，但没有退路，硬着头皮也得往前走。最后咬咬牙也就过来了。

很抱歉和大家诉了这么多苦。其实如果只是写博客，并没有这么辛苦。在写专栏前，我大概写了两年的技术博客，涉及 Kafka、zookeeoer 等系列文章。其实这次有机会写专栏，也是慕课网的小编通过博客联系到我的。而且如果没有写博客的经验积累，即使小编找到我，我也不敢就答应下来。写技术博客有几种方式，我比较喜欢写成系列的文章，这也是对学习最好的输出。如果只是记录解决了什么问题，那就更像是给自己看的笔记。

我写的第一个系列是 Kafka。起因是项目在实际使用，而我想要做一个 kafka 的分享。分享前我下了挺大的功夫学习和准备。分享过后，觉得短短两个小时并没有把我准备的所有东西讲出来。看了看手里为分享准备的材料，应该够支撑我写几篇 kafka 的文章。于是我就开始了 Kafka 系列教程的写作。当时工作不算很忙，基本上一两天就能写出一篇。一开始当然没什么点击量，不过我也不太在意，就一直坚持写。大概用了 20 天就完成了全部 9 篇文章。其实到我写完也没有多少点击量。后来又陆续写了 ZooKeeper、Spring Cloud、Kafka 源代码等内容，还有一些官方文档的翻译。渐渐的浏览量和关注量就上来了，还记得大概半年后我的关注量到了 100，当时还挺开心的。又过了半年多，关注量已经达到了 300，也获得了博客专家的称号。

其实说这么多，只是想讲一个道理。写作要坚持，另外要不忘初心。写作不是为了虚名和赚钱。写作的初衷很简单，就是为了巩固自己的知识和分享知识。只要你能坚持写作，你输出的知识，别人就一定能看到。千万不要写了几天觉得没人看，就放弃了。只要你坚持下去，是金子早晚会发光。退一步讲，即使没人看，也有利于你理解和记忆学习过的知识。如果想让更多的人搜索到你的文章，我可以分享个小窍门，起标题时换位思考，如果自己想学习这门技术时会如何搜索。举个例子，比如我想学习 Akka，可能我会搜索 Akka 教程、Akka 入门、Akka 实战、Akka 例子等。以上关键词就可以适当地放入你的标题中。那么能提高别人搜索到你文章的概率。

虽然很不想结束，但还是不得不说再见。希望这个专栏对你的并发编程学习有所帮助。大家关于软件开发或者职业生涯的任何问题都欢迎和我一块探讨。最后祝大家在技术探索的道路上一路顺利！

}

36 为多线程们安排一位经理—Master/Slave模式详解