Spring Boot + Spring MVC + Spring + MyBatis + 设计模式853769620

Spring MVC + Spring + MyBatis，j2ee就业班培训机构，他们其实都已经做的非常好了，我们就不用去讲了

Spring Boot，设计模式：快速学习 -> 技术储备 -> 模拟公司里的技术调研 -> 大面积的将这些技术运用到有真实复杂业务场景的项目中去

普通人的角度，人类角度，Spring Boot，然后去开始想，先写hello world，再看看官网，再网上搜索一些博客，系列性的，系统跟着敲一遍，找找视频课程，跟着做个小型的demo级的小项目 -> spring boot

# 1、不用Spring Boot的痛苦是什么？

（1）各种技术整合在一起，版本混乱，大量依赖自己去找，依赖冲突

（2）基于xml格式的配置文件，对各种技术框架进行大量的繁琐配置，mvc-servlet.xml，applicationContext.xml，mybatis-config.xml，web.xml

（3）web系统跑起来测一下，需要与tomcat等web容器整合起来才能测试

（4）单元测试的时候需要自己去选择和导入需要的各种测试组件的依赖，junit，hamcrest，mockito，很多组件

（5）部署打包的时候需要自己去配置打包插件

（6）部署应用上线之后，没法去对线上的应用，包括jvm堆栈等方方面面进行监控，没有方便的办法去看到这些东西

传统的以spring为核心的web系统开发，从启动项目、开发、测试、部署以及监控，都很麻烦，有大量需要手工做的事情

# 2、用了Spring Boot以后的好处是什么？

spring社区意识到了这些问题，开发人员一定是最讨厌繁琐的，希望的都是用技术提高工作效率。因此推出了spring boot框架，作为开发spring项目的一个脚手架（脚手架，类似于一个封装在各种技术之上的一个基础框架，基础模板）。脚手架帮助我们快速整合需要使用的技术框架，快速开发、测试以及部署和监控，节约我们的成本。如果没有spring boot，很多公司，特别是一些大公司，会将spring mvc 、mbatis、spring这些框架再度根据自己公司的开发经验、规范和实际情况，再度封装，进一步简化每个新项目使用框架的一个成本

Spring Boot是spring官方社区推出的，设计思想，架构思想，是更好的，更加优雅的

（1）spring boot负责统一各个依赖的版本，保证各种技术的版本之间兼容，自动引入需要的各种依赖。spring boot 1.5.9，在这个版本基础之上，你引入的spring、mybatis、spring mvc、redis、zookeeper、kafka、mongodb，等等各种技术，在spring boot1.5.9这个大版本的基础之上，其实所有技术的版本都是互相兼容的，省去了我们自己去寻找版本整合，解决不兼容问题的一个过程

（2）所有技术整合进来之后，不需要xml配置，spring boot全部是大量基于按照约定的自动配置，自动生成那些技术相关的一些bean，注入spring容器供使用，基于注解进行少量注释，基于application.properties，少量的配置即可

（3）spring boot支持内嵌的web容器，上来直接启动一个main方法就可以启动一个内嵌的tomcat web容器+web程序，快速上手测试，http://localhost:8080/

（4）一键引入需要的所有单元测试组件依赖，所有测试组件的版本兼容，支持controller、service、dao各种测试

（5）默认声明一个插件，自己给你把插件配置好了，支持打包成可以执行的jar包或者是war包

（6）系统上线之后，默认支持大量的线上应用的监控metrics，可以看到线上应用的jvm堆栈，等等信息

# 3、对于Spring Boot这种框架型技术该如何学习？

（1）spring core，那么多复杂的功能，你都学习了吗？spring mvc那么多复杂的功能，你都学习了吗？否，20%。

（2）你在工作中都会用到spring core和spring mvc，这些框架100%的功能，你都使用了吗？否，20%。

（3）在面试的时候，有面试官会揪着spring core和spring mvc的各种生僻冷门的功能、api/方法的细节，问你这个功能怎么回事，那个功能怎么回事吗？否，我面试从来不问任何一丁点的框架功能性的东西。

（4）技术分成五种：**框架性的技术，中间件的技术，架构，项目经验，底层技术**。

框架型的技术，28法则，20%的核心功能，就足够项目中的使用了。学了剩下80%的功能，几乎很少用到，而且功能性的东西，学习，几乎是零成本，你只要掌握了20%的功能，就已经学会了这个技术的基础了，如果你后面项目里需要80%的冷门功能，百度，官网，只要是个大脑正常的工程师，自己都可以搞定。

功能的学习，是最简单

空调，电视机，pc电脑，手机：学会用这些东西，当然要花点时间和成本咯，但是如果要了解这些东西是怎么制造出来， 专家

spring boot，spring，spring mvc，mybatis：学会用，几乎是非常的简单，只要你学会了一点，后面如果你要学剩下的东西，都是易如反掌

但是真正难的是什么（用电脑和使用电脑的区别）？spring boot的源码，设计思想，架构原理，技术亮点；spring core，spring mvc，mybatis，源码层面的

如果让你来手写一个类似于spring boot的框架， 你能写出来吗？

（5）重要的是，迅速掌握20%核心功能，务必熟练，然后就用这20%的核心功能，80%的项目，你都可以做了；接着对剩下百分之80的冷门功能，可以先看一看，项目不一定能用到，未来20%的特殊项目中有需要的时候，迅速通过百度+官网，找到80%的冷门功能中哪些是你需要使用的，快速学习一下，引入项目中来使用即可；最后有些冷门的东西，包括概念和功能，也许你一辈子都用不到。spring core那份官方文档里，可能有50%的东西，我工作10年+，项目，几乎从没用过。如果有些课程没讲到的冷门东西，你项目中真要使用，很简单，百度一下，或者看一下官网，这是工程师的必备能力，总不能饿永远靠着课程里教的东西在活着吧，学习能力很重要

（6）对于框架型技术来说，更重要的，是看源码，掌握其底层的设计原理和架构思想，包括技术亮点，提升自己的内功修为。

工作中用这个技术遇到报错，直接源码分析定位，不用借助他人，hold住全场；很厉害的，spring boot，spring ，mybatis，spring mvc，真正难的不是使用，难的是在工作中遇到了各种报错，看到了异常堆栈，傻眼了，没法处理。架构师，一条，hold住全场，你去大部分的公司，大公司，小公司，技术问题的方方面面，能hold住全场，任何问题出了你是公司的技术力量的制高点，最后一道防线，你一定能解决任何的问题。可以称之为技术专家，高级技术专家，资深技术专家，架构师。

提升你的技术内功的修为，大量的借鉴开源框架源码中的设计思想、架构思想、核心技术，应用到自己的系统设计中；读更多的优秀的开源框架的源码，你的内功修为就越高，你在设计自己的系统的时候，系统架构设计的更好

技术每隔几年就翻新一次，如果每次都是跟着学一些框架功能和使用，那就太low了，费力不讨好，如果不断研究源码，内功修为深厚，会发现大道相通，每次技术更新，你都能快速掌握新的技术。struts2+spring+hibernate -> spring mvc+spring+mybatis -> spring boot+spring mvc+spring+mybatis -> spring cloud+spring boot+spring mvc+spring+mybatis。每次出一个新技术，光是看一看功能，你就能猜测到背后的源码是怎么写的，站在这种内功修为的基础之上，你学习任何新技术，事半功倍，效率是普通的几倍。

对技术你掌握到了源码层面，面试有强大优势和竞争力，这是程序员的硬功夫，立身之本，出去面试可以甩开99%的普通工程师几条街。

学会用一个框架，只要不是傻子，都能学会，所以如果你工作很多年，仅仅只是会用一些技术框架，你觉得你的技术有什么竞争力可言？但是读懂大量的开源框架的源码，积累深厚的技术功底，这个是需要大量的时间去积累的，有一个老师能够带着你给你去讲解，90%的人靠自己读源码，不现实，功力不够，读不会，读不懂，最后没法读源码，能尝试读懂一点点源码，但是没法彻底吸收，融会贯通。源码读懂，吸收，融会贯通。核心竞争力，不是说普通人可以随随便便赶上你的了。

（7）按照这个思路学习框架型技术的效果

出去工作，20%的功能会用，没问题，上手开发一点问题都没有；出去面试，有源码兜底，强悍的竞争力

# 4、我们的课程对spring boot框架型技术的学习思路

（1）看一下网上别人的spring boot类技术课程，还有书籍，讲的大而全，spring boot的方方面面都有了。但是看了一遍，有用么？都是理论，根本不知道项目中怎么使用

书籍：大而全，什么都有，专注于完整的功能讲解

视频课程：大而全，什么都有，专注于完整的功能的讲解

如果这么去学习一门技术的话，如果我们这个课程跟他们一样，按照一样的思路去讲解spring boot技术，那我觉得我是失败的。因为我是不认可这种技术学习的思路的。你学到的就是一堆理论型的技术而已，你不知道在项目里应该怎么结合项目、业务需求、业务功能来使用，而且你光是学纯理论，过一段时间就忘了；如果你能在项目中大量的去实践这个技术，记忆和印象会非常的深刻

不能对框架型的技术学习，不能这样子学，大而全的，一方面很多东西，如果不能立刻用到项目中去，你的掌握很薄弱，你不知道这些技术怎么结合项目来使用；如果没有办法将学到的技术，立即投入项目中去使用，你很快会忘记，记忆会很不深刻

技术能力，其实都是围绕着我们的工作经历，项目经验，发展起来的。我做过这个东西，我学过这个东西。你的理解很深刻， 记忆很深刻，你实际知道这个东西是怎么在项目里用的。

技术的学习，无论是什么，都务必要跟项目和实践结合起来，否则的就是纸上谈兵，很容易遗忘，出去找工作，面试

（2）在每个阶段，针对这个阶段项目要做的事情，我们就学习这个框架需要使用的功能，过框架学习的时候，速度要快，别耗费太多精力在这里扣细节，框架学习的重点是快速学会后用到项目里去，大量开发业务需求，实践中来掌握

spring boot，会被拆分为多个阶段去学习，我们会完全按照公司里的随着项目演进，不断加深学习一门技术和不断实践的一个思路，带着大家去走

我们的项目是分成很多个阶段，阶段一，阶段一的技术挑战是比较小的，我们就去学习spring boot 20%的核心功能即可，我们快速几个小时学完以后，每个人都能上手开干，立即投入项目中，大量的去写业务功能，实践，加深你的印象

当然，阶段一，重点会剖析spring boot框架的核心源码，才是我们真正要花费实践在spring boot上的，竞争力

后面随着每个阶段的进展，比如我们的阶段二，要使用redis技术，然后我们就先学spring boot如何跟redis整合使用，立刻投入到项目中，大量跟项目整合来使用，印象会非常深刻，我们会同步剖析spring boot跟redis整合部分的源码

（3）随着项目阶段不断前进，对于框架型技术的学习会逐步完善，而且每一块的学习，都会是快速过功能，然后迅速用到项目中去实践的思路

（4）但是对核心框架的源码，必须100%要分析，因为这才是真正的重点和工程师的核心竞争力

我们整个技术体系里，框架型技术，实际上不多，spring boot，spring cloud

很多的技术，是中间件型的技术，redis、zookeeper、kafka。中间件型的技术，学习的思路就完全不一样了。非常非常重要的，务必是说把80%的核心功能都要掌握，然后尽可能在项目中将80%的核心功能都给实践了，同时尽量对重要的中间件，都要去剖析里面的源码。招聘需求的时候，深刻理解分布式的消息、缓存、存储、搜索，redis内核源码、zookeeper内核源码、kafka内核源码、elasticsearch内核源码。让大家不是仅仅只是学会用一个中间件技术，而是要从根本上的底层原理，去掌握分布式的缓存、分布式的协调、分布式的消息、分布式的搜索、分布式的存储。

5、我们这个小阶段学什么

阶段一，其实只要掌握spring boot 20%的核心功能，就足够了，立刻就开始动手开发，去做项目

（1）Spring Boot快速入门

（2）Spring Boot与Spring MVC+Spring+MyBatis的整合使用以及RESTful接口的支持

（3）Spring Boot框架配置

（4）Spring Boot单元测试

（5）Spring Boot打印日志

（6）Spring Boot应用部署

（7）Spring Boot应用监控

**按照公司的规范，编写一份Spring Boot的技术调研文档**

上面这些东西就完全可以支持我们阶段一的项目开发了，其实我们就是要用spring boot整合spring mvc+spring+mybatis来进行基础的开发，对外提供restful接口，进行单元测试，打印日志，最终部署一个单块的web应用，然后线上可以进行监控，仅此而已，完全足够我们用了

在开发完电商系统v1.0之后，我们来剖析一下spring boot使用到的这些部分的源代码，看源码，将从这里开始，同时还会讲解如何手写starter来支持任意技术与spring boot整合，比如公司自行研发的框架之类的东西

# 5、我们这个小阶段不学什么以及未来阶段学什么

（1）spring boot与thymeleaf、freemarker等模板技术的整合

这个阶段暂时不学习

纯后端放所有的java代码和前端模板页面，因为这个未来不是主流趋势了，主流趋势是前后端分离，我们后端工程师不再关注页面上的东西，纯碎关注后台架构，提供RESTful接口或者是rpc接口给前端工程师调用。大公司全部是这样，中小型公司未来主流也该是这样。如果有那种公司，还将前台页面包含在后端工程里，让java工程师写html模板，那赶紧走，这公司太没前途，技术太low。

未来只有那种非常小的项目，前台后台都是一个人写的项目，大学生的毕业设计，适合采用jsp、thymleaf模板放java工程里，后端工程师一个人写了

我们会提到一点，但是不会耗费大量精力去琢磨spring boot跟各种模板技术整合的细节

到了后面，做高并发系统架构的时候，一定会涉及到大型电商系统的复杂的页面静态化架构，那个时候我们再来结合具体的架构和项目讲解与模板技术的整合

（2）Spring Boot与Spring Data JPA的整合

不学习

JPA，Java Persistence API，java持久化API，ORM的概念，hibernate。就是不需要你写sql，直接你就是面向对象去操作和开发，从一个对象里面去get一个集合，此时hibernate会自动给发送sql查询数据。hibernate，性能很低，自动生成的sql都有问题。开发效率很高，很好维护，不需要写sql.

ORM框架很好，但是比较适合传统IT行业，性能要求不高，开发速度要求很高。我最近这么多年的工作，已经很少见到有用hibernate的了。

一般都是mybatis类的框架，支持写SQL，加上一些mybatis对sql的特殊功能的支持。java工程师精通写sql必须的，错误观点：“写sql不是java工程师的活儿，是dba的活儿，java工程师只要面向对象就可以了”。

暂时没看到需要学习这块的必要性

（3）Spring Boot与中间件技术的整合

mongodb、redis、elasticsearch、cache支持、分布式session支持、zookeeper、kafka

暂时不学习

spring boot整合mongodb，mongodb的核心概念以及CRUD的操作

现在学习这个干什么呢？学完就忘了。等到后面项目进展到要用mongodb、redis、elasticsearch的时候，自然就会学习如何Spring Boot与这些技术整合在一起使用了，这都不是事儿，学完了立即投入项目中，剖析他的源码

等项目进展到要搭建缓存架构的时候，自然会学习cache支持，项目做成分布式系统之后，要用到分布式session的时候，自然会学习分布式session的支持，引入zookeeper、kafka的时候，自然会学习如何整合

（4）spring boot与jdbcTemplate的整合

这种技术都没必要讲解了吧，包括jdbc的使用之类的，太原始了，spring boot为了大而全提供了这个支持，但是不意味着我们要讲解和使用，现在机会很少见到项目是用JdbcTemplate的

（5）spring boot与websocket、webservice的整合

这些其实都是spring boot提供的功能，同上，为了大而全，spring boot一定会提供大量的功能，但是我们不一定要使用。websocket是一个很不错的技术，常用于浏览器和服务端进行频繁大量的通信，就是浏览器需要主动跟服务端进行频繁的大量的交互，不断的刷新和更新数据。

不是普通的那种系统开发要用的技术，常见的场景是：社交聊天、弹幕、网页游戏、股票基金实时价格、体育实况更新、视频会议、在线教育，等等。

这个我们看后面课程具体的发展情况，如果有合适的场景，比如电商系统的聊天系统，让用户与客服沟通，可以使用websocket技术，等真正用到这个技术的时候，我们再来讲解

而至于webservice，未来都是微服务的天下了，这类webservice，SOA，技术以后可能会逐步退出历史舞台

（6）spring boot与安全技术的整合

spring security，oauth2

这个不会放这儿讲解，会放到后面专门的一大块安全性架构中去讲解，针对各种安全性需求，spring boot这块可以做什么

（7）spring boot对定时调度的支持

这个以后电商系统里一定会有定时调度的需求的，到时候再讲解

其实你能在网上看到的一些书籍或者视频课程里的东西，我们的整套课程后面是一定都会包含的，只不过采取我的授课思路，逐步演进式的学习

spring boot官方手册里的一些东西，随着我们系统的不断深入，如果要使用到一些东西，我都会带着大家不断的去学习使用的