第一个版本的简单思路介绍

（1）要把之前分析过的电商系统v1.0需求全部做出来，形成一个电商系统v1.0版本，认为是一个电商系统的雏形

（2）软件工程、项目管理、Spring Boot、设计模式

1、软件工程

软件工程，如果你是计算机科班出身的，尤其是大学的专业是软件工程这门专业的话，实际上你必定会有一门专业课，叫做《软件工程》。但是问题在于什么呢？软件工程不是那样学的，不是上来就一本特别抽象和高度归纳的一本老外写的书，你就能学会的。学习软件工程，靠的不是大学里的课本，你需要的是真正在简单了解整个软件工程的流程之后，快速的去运用整个软件工程的流程和思路去做一个项目出来。真正用软件工程做过项目之后，你立马就可以体会到软件工程是什么东西？

现在的一个问题，很多同学，可能上大学的时候都学习过软件工程这个东西，结果。。。。当时根本没弄明白是怎么回事，所以应付一下考试，直接就给忘了

然后更加坑爹的，你参加工作之后，刚开始你就是一个小兵，然后就是照着公司里的流程和规范在走，完全还是没法从全局上去把控什么叫做软件工程。但是问题在于说，如果我们这个课程，架构师，把控软件工程的流程和规范。

就我目前观察，反正市面上几乎所有的所谓的Java高级或者架构师的培训课程，都没有人去讲解软件工程的

一句话概括一下软件工程：就是从0开始如何完成一个软件项目，涵盖了做一个软件项目的所有的流程

（1）站在架构师的角度来制定软件工程流程和规范

你现在是一个架构师，你带了一个小团队，比如说几个人，去主导公司里的一个重要项目的开发

京东app的后台系统，刚开始3个人，一个leader带2个小弟

微信，刚开始的时候，就几个人

搜狗，就六七个人，就开始干了

百度，淘宝，阿里，刚开始就几个人

一个业务快速发展，需要的工程师越来越多，需要更多的人去开发不同的业务系统；一个系统访问量越来越大，职能拆分团队，dba团队，运维团队，后端团队，前端团队

做的第一件事情，一定是制定一份这个项目开发的软件工程的流程与规范，瞎做，第一步做什么？第二步做什么？如何一步一步的将一个项目给干出来？

这个过程，就是一个软件工程的流程 -> 每个步骤到底要怎么做，要按照什么统一的规范来做，软件工程的规范

一般来说，我在团队里面就会说是系统开发的流程和规范 -> 软件工程的流程和规范

第一件事情：业务调研，我们已经做完了。在公司里，如果在互联网行业里，这个业务调研，不是我们做的；产品经理的角色，跟系统的用户去聊，去调研需求，梳理出来一份产品设计文档，有的时候也可以叫做业务调研文档，业务调研 -> 产品设计

1. 业务调研：主要是完全跟技术无关，站在业务的角度去定义系统要干嘛
   1. 组织结构图：部门+岗位
   2. 业务流程图：泳道图，一级业务流程+二级业务流程
      1. 系统多个模块的整体业务流程
      2. 每个模块内部的业务流程
   3. 业务表单
   4. 业务需求
2. 需求分析：站在技术的角度，去分析系统要干嘛
   1. 用例图
      1. 用例表：用例名称+多个参与者+每个参与者对这个用例做的事情
      2. 用例描述+用例流程（泳道图）
      3. 用例图：参与者+用例，一个用例就是一个功能需求
      4. 用例划分
         1. 每个模块都划分出对应的用例来
         2. 从业务流程图的运行节点中，抽取用例
   2. 领域类图
      1. 主要根据业务表单来抽象
   3. 非功能需求
      1. 物理部署需求
      2. 实施需求
      3. 易用性需求
      4. 性能需求
      5. 可靠性需求
3. 概要设计：架构师一个人去做的
   1. 逻辑架构图：根据需求，画出来逻辑上，系统要长成什么样子
   2. 运行架构图：时序图、活动图（可选）、状态图（可选），系统的逻辑架构有了，系统跑起来是什么流程
   3. 物理架构图：组件图、配置图，系统长什么样子，系统跑起来流程是什么，系统真正怎么去部署
4. 概要设计评审
5. 详细设计：下推给项目组里的每个人去做
   1. 数据架构图
      1. 数据库ER模型图
      2. 数据库逻辑设计图
      3. 数据库物理设计图
   2. 接口设计
   3. 开发架构图：实现类图、包图
   4. 系统运行流程图（活动图）
   5. 测试用例设计
      1. 单元测试用例设计
      2. 冒烟测试用例设计
   6. 日志设计
6. 详细设计评审
7. 项目管理计划（每个人给出自己的排期计划）
   1. 活动图
   2. 网络图
   3. 进度计划（甘特图）
   4. 资源配置
8. 工程初始化
   1. 每个人本地的开发环境搭建
   2. 资源申请：机器、数据库、缓存、MQ，包含了各个环境的资源申请
   3. 数据库初始化
   4. 代码初始化和上传
9. 版本控制
10. 编码开发
11. 系统测试
    1. 单元测试（白盒测试）
    2. 冒烟测试
    3. 静态代码扫描
    4. 代码审查
    5. 集成测试（联调测试，也可能是RD自己干，不是QA干）
    6. 系统测试（QA去干，黑盒测试）
    7. 验收测试（让业务需求方来试用一下，验收一下，一般来说，互联网行业里做验收测试的，一般是产品经理，PM）
12. 系统上线
13. 系统运维：跟进bug的反馈，排查线上日志，解决bug

概括一下

我们这里的软件工程的流程，基本上就完全涵盖了标准软件工程的核心流程

**软件工程：安全性和可靠性（安全架构、可靠性架构，后面讲解），高级软件工程（SOA、分布式、嵌入式，有些到了微服务架构会详解，有些不是我们的范围，到了后面会讲解）**

软件项目管理：我们这里就要带着大家来实践了

作为一个架构师，为自己要负责的项目，定义好一整套的软件工程流程，同时对流程中的每个步骤定义好规范

（2）基于真实的电商系统项目来完整执行和实施一整套的软件工程流程和规范

2、项目管理

软件工程和项目管理，密不可分

因为是这样的，你现在已经知道整个软件的流程和每一个步骤的规范要做什么，要怎么做。**但是你如何带着一个团队，给他们分配好任务，带着他们一步一步的把这个流程全部做完呢？**

**要做到这一点，就是要依靠项目管理，对整个项目进行管理和把控，确保说带着一个team可以把你定义好的一整套东西全部做出来**

（1）**项目管理：项目管理计划、项目执行和监控、风险管理、变更管理、质量管理（配置管理+测试管理+缺陷管理）**，5大过程域、9大领域、44个定义（我不想这样去讲，涵盖的非常非常的完整）

我们第一个版本，不会去涵盖所有的完整的项目管理的东西，庞大了，抽取出来我们项目做的时候平时常用，最实用的一些东西，拎出来，带着大家在项目实战中去演练和体会。到了后面，随着我们的项目越来越复杂，项目管理的东西也会不断新增更多的内容进去。举个例子，就是说，比如说，你到了后面，可能一个大系统拆分给了多个小team去做，同时涉及到并行在执行的多个项目，多项目集群管理。

（2）瀑布式 vs 敏捷式

做项目，按照我刚才上面说的那一整套东西，包括软件工程流程+项目管理流程，就是非常经典的一套瀑布式的过程。瀑布式不是说就不好，其实是有其用处的，在我的个人经验里，我觉得，就是尤其是在一个项目从0到1的时候，采取瀑布式的方式来做，是挺靠谱的。需求变动是比较少的，还没有涉及到终端用户的反馈，其实就是一个初始产品的研发

但是从1到10的时候，此时系统已经上线了，面向终端用户去使用了，大量的需求迭代，并行的多个项目，直接会喷涌而来，此时瀑布式的方式来做，就不太灵活了，这个时候可能就需要用到敏捷式的开发流程和项目管理

我一开始在从0到1的时候，采取瀑布式的方式，大家可以学会

接着从1到10的时候，到微服务的环节，本来就要涉及到微服务的持续集成、持续交付、敏捷开发、敏捷项目管理

其实这个过程是很自然的，也是很合理的，因为我是经历过大量的项目，从0到1，再从1到10的，这个过程的转变，基本就是这样的

（3）站在互联网项目的角度进行项目管理

1）传统IT行业（华为、中兴、神州数码、联想、用友） vs 互联网行业（BAT、一线互联网公司）

传统IT行业的项目管理，跟互联网行业的项目管理， 大体上来说，是类似的，但是因为行业的不同，决定了很多东西还是会有差别的

2）互联网行业，没有架构师title，技术专家 -> 架构设计、团队管理、项目管理

传统IT行业，就我所知，拆分是比较细的，可能是这样的，一个team，有专门的做团队管理的leader，负责团队人员招聘、绩效；团队里有专门的架构师，就负责架构设计；项目经理，就是负责对设计好的架构，进行项目全生命周期的一个管理和负责

互联网行业，没有分的这么细的，PM、RD、QA

**PM负责设计产品 -> RD负责开发（后端RD，前端RD，数据RD） -> QA负责测试**

技术专家的职级，兼任了三个岗位，团队管理、架构设计、项目管理 -> 既要负责人员的招聘，打绩效，职级晋升评审；也要负责整体的技术规划，核心项目的架构设计；同时也要做重要的项目进行项目管理

（4）站在架构师的角度来制定项目管理规范

（5）作为一个架构师来完整管理和把控整个项目流程：模拟一个1+4的小team来进行项目管理

你一个架构师，带了4个小弟，带着4个小弟一块儿来干活儿

3、Spring Boot项目实战

以前的技术框架的大路货，通用货，每个人都会的东西，在2010年左右，是SSH2，Struts2+Spring+Hibernate

后来可能在2013年左右，大路货变成了，SSM，Spring Web MVC + Spring + Mybatis

每个公司，每个工程师，都必须掌握的一些东西，就是用这套最基础的框架来开发项目的

从去年开始兴起，2017年，Spring Boot，出现的原因是什么呢？每次要上手开发一个项目，都比较繁琐，自己要找一堆的框架版本如何兼容，在pom.xml里面配置依赖，接着对每个框架都要做xml配置文件的设置，mvc-servlet.xml、applicationContext.xml、mybatis-config.xml、web.xml、mapper.xml -> 框架整合完毕，可以开始开发了。为了要打包，还要引入maven打包的插件，配置好插件，让它可以打成一个有所有依赖的一个war包。本地测试，还要安装tomcat集成到eclipse，本地基于eclipse里的tomcat去部署和测试。

大家也都发现了，启动上手做这个基于spring的项目，太麻烦了，很多东西要弄，搭建一个工程要半天时间

Spring Boot项目，boot类似于一台电脑打开启动的意思，快速启动一个基于Spring Boot的项目

上来就是放两个简单的maven依赖，然后什么都不用配置，Spring Boot的starter依赖，直接自动给你配置好Spring MVC + Spring + MyBatis，然后就可以写代码了，写好代码之后，直接就可以打包部署，如果在本地测试，甚至可以直接运行一个main类就可以跑起来测试。如果要上线部署，直接打jar包或者是war包，打好之后就可以部署了。

传统的开发模式下，如果你要集成MQ、缓存、zookeeper、kafka之类的东西，都要做很多的事情

但是基于Spring Boot，其实就是一个脚手架的意思，给你提供了一堆的starter依赖，比如你要集成一个redis，可能就是引入一个starter依赖，然后自动就给你配置好你需要的一些东西，然后上手就可以开始开发了

在进行整个系统框架的搭建和整合的时候，基于Spring Boot可以大量减少工作，加速你的项目的启动

预料的一点，在未来的1~2年内，大路货就变成了Spring Boot + Spring MVC + Spring + MyBatis

基于目前最热最流行的Spring Boot技术来实战开发电商系统的v1.0版本

（1）Spring Boot + Spring Web MVC + Spring Core + MyBatis

（2）MySQL + Tomcat

（3）Maven + Git

**4、23种设计模式**

透彻的掌握设计模式，能够灵活的运用在系统的设计中，保证系统的代码是非常优雅和高度可扩展的，是一个架构师必备的基本能力

因为我见过的大量的烂代码，没有经过良好的分析和设计，就是胡乱写，胡乱实现需求，写出来的代码跟屎一样 -> 有些代码，写了1年之后，项目组里就没几个人能看懂了 -> 2年之后，基本就完全没人能看懂了 -> 最后只能推倒重做

烂代码，要对代码的逻辑做一些改动，重构一些实现机制，我天，看着一团乱麻的代码，自己都不知道如何下手 -> 即使要下手，也要对代码进行大量的改动，改动完了之后自己心里都没底。做系统维护的，加一些功能，修改一些功能，修复一个bug，效率非常之低。因为看不懂代码，代码太乱了，不知道怎么改，不知道怎么下手。

如果要写一手漂亮的好代码，设计模式的运用是至少的，可以将复杂的缠绕在一起的屎一样的代码，给转换成拆分的清晰、逻辑清爽、一眼就能看明白是怎么回事、改动的时候只要修改一点代码就可以了

将23种设计模式在我们的项目中全部实战、融合和体现，包括以后的版本，大量应用设计模式，确保写出来的代码扩展性良好，不要写屎一样的代码

你看市面上不少书和视频课程都是讲解设计模式的，但是有个问题，全部都是理论、demo案例来讲解的，你看完了以后，也很难将设计模式运用到你自己的项目里去

所以说，我们这次用真实复杂的电商业务来讲解项目，好处就在于，业务足够复杂，可以支撑我们将几乎架构师的所有技术都融入进去

**5、Spring Boot源码剖析**

我这里的一个思路，大家每跟着我学会一个技术，对这个技术一定不能只是简单会用，而是说要深入研究和掌握底层的原理，深刻理解这门技术

我最近给一些同学修改简历，包括我自己平时招人，招聘的时候，看简历，最怕的就是看到这种简历

（1）熟练掌握spring mvc、spring、mybatis等框架的使用

（2）熟练掌握spring cloud、dubbo、spring boot等框架的使用

（3）熟练掌握redis、zookeeper、kafka、mongodb、elasticsearch的使用

（4）熟练掌握maven和git的使用

（5）熟练掌握mysql、nginx、linux的开发和使用

薪资最多就是20k左右，不能再多了，这种人真的不行

这种简历，占到了市面上90%以上的简历

看到了我就头疼，很简单，看简历就知道你是什么水平了，没什么可以问的，基本上就是各种技术都会用，在项目里也许实际用过，但是也没什么挑战的一些项目

如果说你工作了3~5年，你的简历长成这样，说明你这今年的技术积累绝对不够

看看另外一份简历，一些BAT或者比较好的一些工程师的简历

（1）深入理解spring mvc、spring、mybatis等技术，研究过底层源码，熟悉其架构设计，能够进行开源框架的二次开发

（2）深入理解spring cloud、dubbo、spring boot等技术，研究过底层源码，对分布式系统的架构设计有着深刻的理解和掌握

（3）深刻理解redis的底层原理以及内核源码，对于多级缓存架构、缓存预热、缓存穿透、缓存雪崩等各种场景都有一定的理解，并且在项目中对缓存架构有深入的事件经验

这个人一看就是说，对技术有很强的追求，对各种技术有深入到源码的研究，对一些复杂的架构设计有实践经验，能做框架做二次开发，也能进行复杂的架构设计

30k，什么的，都可以要，30k~35k还是比较容易的一件事情

我期望的是，大家跟我学会的每个技术，都是深入的掌握的，掌握到了底层的源码层面，甚至可以二次开发，做各种复杂的大型架构的设计，做复杂的公司级的开源框架的二次开发，可以做复杂的公司级的基础中间件的架构设计和开发

光是会用Spring Boot，是没有什么值得骄傲的，因为使用非常之简单，其实就是个脚手架，各种常用技术通过一堆starter依赖给你自动配置好，精简xml配置，基于注解来配置，降低项目启动成本，加速项目开发

Spring Boot的核心在于其设计思想，在整合技术的时候，如何简化配置，提高开发效率，重点是其源码剖析，源码层面是如何做到这一点的，架构是什么，设计思想是什么，亮点是什么，我们可以借鉴和学习到什么

6、整体思路梳理

（1）完整贯穿和实施软件工程全流程，自己制定团队的软件工程规范

（2）完整把控整个项目全流程的项目管理

（3）在系统技术栈中，重点实践Spring Boot这门技术

（4）在系统设计中，重点实践23种设计模式的运用，如何写一手可扩展的漂亮代码

（5）最后深入剖析Spring Boot技术来结尾，加强自己的技术深度

（6）全程基于真实的电商项目v1.0业务需求来在项目实战中学习上面所有内容，项目实战驱动，保证最好的学习和吸收效果，同时将电商项目v1.0的业务需求全部开发出来，形成一个电商系统雏形