Java开发与技术挑战——关于技术的技术思考

发表于2015-05-19 17:28| 8575次阅读| 来源CSDN| 32 条评论| 作者王继辉

[特别策划](http://www.csdn.net/tag/%E7%89%B9%E5%88%AB%E7%AD%96%E5%88%92/news)[Java](http://www.csdn.net/tag/java/news)[《程序员》电子刊](http://www.csdn.net/tag/%E3%80%8A%E7%A8%8B%E5%BA%8F%E5%91%98%E3%80%8B%E7%94%B5%E5%AD%90%E5%88%8A/news)[编程语言](http://www.csdn.net/tag/%E7%BC%96%E7%A8%8B%E8%AF%AD%E8%A8%80/news)[Java20周年](http://www.csdn.net/tag/java20%E5%91%A8%E5%B9%B4/news)

**摘要：**Java语言已经走过二十个年头，岗位和身份的变化，也伴随着Java的同步成长，其中也包含了对技术理解的思考与变化。主要体现在四个阶段：开发工程师、系统架构师与技术主管、技术顾问、技术合伙人。

到目前为止，接触Java语言已经有将近10个年头。从趴在学校实验室中认领师兄提供的论坛系统开发任务，到参与WebERP构件化平台、网络资源聚合平台、电信增值业务平台、企业门户网站以及运营管理平台的研发，再到后面的掌上证券、软件商店、移动支付平台等移动互联网平台建设，不论是PC端，还是移动端，底层的技术一直都是用Java架构来进行实现的。期间参与过多次的Java技术培训与问题指导，以扑火队员的身份见证了形色各异的Java坑，从而不断提醒自己要跳出Java语言本身，去思考一些开发中的技术困惑和挑战。



**Java的成长经历**

Java语言已经走过二十个年头，岗位和身份的变化，也伴随着Java的同步成长，其中也包含了对技术理解的思考与变化。主要体现下面四个阶段：开发工程师、系统架构师与技术主管、技术顾问、技术合伙人（联合创始人）。

**开发工程师**

以前大学学的专业是数学而不是计算机，Java编程之路是从《ThinkinJava》这本书开始的，当时每天晚上十二点看到凌晨二三点，就这样持续将近3个月，面向对象的编程就这样入门了。Java语言上手非常容易，装一个IDE工具（Eclipse、Myeclipse等等），找一本参考书就可以开始，但是要想深入，还需要学习很多东西。从Java的常用包Java.lang.\*,Java.util.\*,Java.net.\*等等开始，逐渐开始学习dom4j、jdom、log4j、hibernate、spring、ibatis、struts等等以开源为主体的各种Java扩展技术，jsp、mvc、ssh等等也是经常挂在嘴边的技术，JSON与xml的封装和解析也是数据配置与交换中常用的技术手段，明白和使用正则表达式也是开发中一块比较兴奋的事情。运行环境Tomcat、JBoss、WebLogic等等都用过也是Java知识丰富的一个佐证，能用过就懂Web容器与应用服务器的区别。

由于Java广泛用于各种互联网的场景中，学习Java开发，还需要学习SQL语言以及各种商用关系型数据库MySQL、SQLserver、Oracle等等（NoSQL、memcached等各种非关系数据库）的安装与使用，在进行专业领域还要研究Lucence等等搜索引擎开源框架和组件。有些专注前端的可能还需要学习HTML、JavaScript以及jQuery之类的知识。

Java入门阶段接触最多的开发可能就是论坛系统、管理信息系统以及OA系统之类，深入学习后面临的项目和平台就比较专业了，比如软件商店、支付平台、广告平台等等。由于Java相关知识非常广泛、开源项目和平台众多，在这里一直是个学习者，Java二十周年技术人员学习的好平台。

**系统架构师与技术主管**

谈到Java优点时，大部人分都记得它有一个非常先进的垃圾回收机制，在对象不被使用时可以自动回收，不用开发人员关心，甚至把这个说成为是Java优越于C++等其他面向对象语言的一个重要特性，而在实际中，对象引用错误、JVM内存以及堆内存的分配设置不合理等等现象，往往会引发服务器应用出现outofmemory的错误，进而宕机。Java的垃圾回收函数System.gc()只能发送垃圾回收请求，实际执行时间仍旧不总是可靠的。在进行内存对象进行处理时，Java对象在强引用下是不可回收，使用Java的软引用（SoftReference）机制可以做到让内存空间不足时自动释放不用的内存对象，从而保证内存空间一直可控。

在Java应用开发中，逐渐开始使用设计模式的思想来进行关键模块的开发，单态模式、组合模式、适配器模式、观察者模式等等面向对象的特征也得到很好的发挥。基于Java架构设计中更多通过机制性保证来构建一个比较稳健可扩展的系统，比如无状态设计、同步异步分离、动静分离、模块服务化、多级缓存等等。

在技术选型上，像软件商店、移动支付平台，除移动端使用Android实现，服务器都可以使用Java进行实现，这样一方面可以重用原有程序积累，又可以通过Java的丰富开源组件来加快开发速度。但技术方面的选择上不再以Java为唯一技术选择，后端存储层面，可以把关系型数据库、非关系型数据库以及文件系统进行整合考虑，不再把单一的“Java+关系型数据库”作为软件系统的唯一设计标准。后台任务服务以及程序逻辑都可以用Python、Shell脚本进行实现也非常好。多种技术组合使用可以充分发挥每一种技术的优势，这也是技术选型的一个重要原则。

JavaEE的分层设计思想给软件开发提供了有利条件，根据前端、数据持久层、控制器层、代理层、组件层等视图可以有效地把工作模块分解给不同的团队成员，从而保证大型项目的任务分解与工作可跟踪。在产品进行迭代和升级时只需要修改受影响的代码部分就可以保证最大限度的代码稳定性。

**技术顾问**

Java技术输出主要涉及三个方面的内容：基础知识培训、基于性能优化的诊断分析、基于业务流程优化的架构重塑。Java基础知识培训主要定位在Java新手快速扫盲；基于性能优化的诊断分析主要集中在中小型公司，它们一般开发有一套自己的Java系统，但总出现系统问题或者访问量就是上不来（特别是它们购买的是性能比较好的服务器）；基于业务流程优化的架构重塑主要对业务上有新需求的Java平台改进，原有平台比较混乱，需要进行统一整合，它们的立足点不是因为现有平台不能运行而影响业务。在这个维度，Java语言本身被包装成服务，已经不再是开发某一个模块那么简单了。Java平台出现的不同坑也只有经历了才能知道。

**技术合伙人（联合创始人）**

以互联网为载体的创业公司往往喜欢招募PHP的开发人员，主要是因为网站类开发使用PHP比较快，同时也有一定量的开源平台可用。从技术合伙人的角度思考，重塑和定义业务、优化管理是使用技术比较普遍的场合，依托开源项目或原有积累进行二次开发就比较方便，对于资源比较丰富的企业可以选择完全从零开始进行自主开发。技术的应用和功能模块需要有一定的预见性和主动性，是不是Java语言实现并不是最重要的，业务自由度越大的时候对Java语言本身的依赖性就越小。

**有关技术挑战的思考**

Java语言的学习和运用一开始可以让我们更多的关注在Java技术应用本身，随着身份的变化和多元化的场景的出现让我们对Java又有更加深入的理解。那就是Java是一门编程语言，我们处理和解决问题关注的更多层面不是选择哪一门语言，而是如何更好地满足实际场景的需要。跳出编程语言来思考问题，可以有助于看清楚技术的优势和短板，方便发现和理解一直存在的技术困惑和挑战。这里先讲两个与Java项目有关的真实案例。

**酷派通行证Android应用工作交接的冲突**

酷派通行证Android版是一款用于酷派手机应用进行身份登录鉴权以及掌上证券支付的移动应用，以前是西安一个团队负责进行维护和升级，后因工作需要移交给北京所在团队，当时就指派一个比较年轻的同事接手这个项目。Android应用主要是用Java开发的，兼有少量的C语言开发，其中C语言部分主要用于进行私有协议数据交换，保证数据交换安全。代码交接过来，本地开发环境稍加配置就可以消除所有编译错误，但打包发布成独立APK后却怎么也无法正确调用（如图1效果所示），与西安同事进行过确认，所有的代码是完整可用的。

当接手工作的同事向我求助时，我当时也不清楚，只是从经验角度提供几个可能的原因让他进行分段验证，进而定位可能出现的原因，就这样持续了几天还没有解决，但深圳总部的手机项目已经反馈这应用有Bug出现，希望我们尽快进行解决。由于现在还没有本地运行通过，所以一直无法进行修复和更新版本。手机项目的入网、量产等环节都是有非常确定的时间节点，软件代表的邮件和电话也在不停地跟进这事，在这种压力下，接手这个工作的同事甚至产生不干了的想法，因为不知道这个问题是怎么回事，无法进行解决，更回答不了具体解决问题的时间点。这个问题持续两周，经过仔细测试和排查，发现问题的关键出现在编译环境上，交接前的编译环境和新的环境版本不同，从而导致高版本的编译程序无法运行在低版本的环境下，调整编译环境后问题自然解决。

遇到未知的问题和现象来临时普遍会表现的比较不自信，精神压力比较大，很难一直保持对Java专业知识的冷静和思考，容易产生放弃解决的矛盾心理。研发人员在进行产品研发时面对的问题往往是多方面的，并不是说一眼都能看出来是哪一行代码错误那么简单，没有找到原因之前可能是无解的，可以说是一种煎熬。



图1酷派通行证Android应用的冲突

**光合原品微信网关程序日志停止输出的异象**

因为光合原品业务需要，微信网关需要增加一个功能：每次用户关注微信公众号“光合原品”，给用户下行的提醒语要可以变化的。这种需求的简单做法就是后台维护一系列的关注回复语，使用Java的随机数函数机制来随机提取后台的提示语即可，程序开发完成后部署到生成环境，运行发现下面的异常：

2014-06-21 12:19:22 INFO [com.jiuxf.weixin.protocol.xml.MessageXMLUtil] jsonObject data is null

SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".

SLF4J: Defaulting to no-operation (NOP) logger implementation

SLF4J: See <http://www.slf4j.org/codes.html#StaticLoggerBinder> for further details.

后面的程序日志就彻底打印了，而这些程序在本地进行测试时运行也是正常的，这两个方面有点自相矛盾，也就是我们常说的悖论。

为了开发这个功能，程序中增加Redis为缓存处理中间件，用于临时存放关注缓存数据，缓存数据是以JSON的形式进行存放，程序中启用json-lib-2.4-jdk15.jar第三方jar用于完成对JSON的封装与解析。由于本地开发环境与服务器不一样，JSON封装解析运行还有其他依赖包（commons-beanutils-1.8.0.jar、commonscollections-3.1.jar、commons-lang-2.5.jar、commonslogging-1.1.1.jar、ezmorph-1.0.6.jar），jar冲突在Java平台下出现非常普遍，解决起来也非常让人无语，主要是不同的jar包设计者为了保证自己的程序运行不出现异常，往往会隐藏调用中的异常痕迹，导致看不到程序错误日志，程序问题定位难度很大。通过对jar的调用部分逻辑进行常量替换跟踪测试，发现问题依旧存在，jar冲突给予排除，还要继续寻找其他原因。经过调查和文件对比，真正的原因是没有在beanfactory.xml配置文件中新增的class声明，如图2所示。



beanfactory.xml的修改是手动进行修改的，遗漏了一条新增加的Service类。这种日志最奇怪的地方是程序出错，日志文件就不输出信息，传统的Java程序运行故障都会抛出Exception日志，专业人士一眼就看到问题所在，但没有错误提示想解决就难度大了。

**未知与悖论是技术人员无法回避的两大技术挑战**

上面提到的两个真实案例非常简单，其中包含了技术人员经常面临的两个挑战：未知和悖论。

未知是指人受特定的时间和空间的限制及影响，对于现有事物的认知产生一定的局限性，且处于一种迷茫的感知状态。未知体现在没有技术人员没有进行开发工作前的一种状态，这个状态中Java人员的心理不是不会写class，而是不知道当前工作与什么类和函数有关。遇到这种状态都会本能地使用搜索引擎查询相关资料，或与朋友同事进行交流，以获求最直接的帮助和解惑，但没有效果，因为他们也不知道。处在这个状态中，技术人员也会呈现不那么自信，不知道如何解决，不知道什么时候能解决，甚至会做出各种极端的举措。一定程度上，技术人员会写各种各样的Java代码，熟悉各种开发组件和开发包，关键是不知道自己要写什么，这就是未知的挑战。

另外一个挑战就是“悖论”了，这个必须打引号，说是悖论，其实并非真正的悖论，它只是当局者不能使用先验知识解释当前现象的一种错觉，代码都是对的，环境都是OK的，怎么就出现让自己不可理解的效果呢，实践证明这类问题在技术上往往都能解决。刚才说未知是技术人员没有开始工作前遇到的挑战，悖论往往是已经按照原有的技术讨论开展工作后的挑战，看似已经完成特定的任务，但结果却不是自己预期那样。在这个状态中，一开始可以按照理性的方式进行分析和确认问题，当几轮分析下来，可能就会走弯路（比如放弃和怀疑自己原有的知识结构的正确性，在错误的方式上不断循环验证等等），从而导致走进死胡同中出不来。

做技术研发时间越长，越要关注实现细节，利用完整的知识结构来验证每一个环节，悖论自然消除，未来的东西也是可以用已知的知识来解释和分析。技术人员面临的挑战和压力更多已经不是某一项新技术目前还不会怎么办，而是会原有技术的情况下出现的各种未知和悖论该怎么做。

**写在最后的话**

前文虽然解释了技术挑战的发生与特点，限于篇幅的限制尚不能展开叙述，为了更好地梳理开发中遇到的各种坑，正视各种看似荒诞无奇、甚至不可能出现的悖论现象，本人将在2015年撰写一本新书《技术的触动》（暂定书名），会结合自身真实经历，讲解各种具象（假设、一致性、技术认知、边界、依赖等等）的特征和内幕。有兴趣者可以关注我个人微信：hexi30nian（发送关注请备注来自CSDN）。细节问题也可以进行交流。

作者简介：王继辉网名河西三十年，北京光合原品科技有限公司联合创始人&CTO，有8年的互联网从业经验，在职期间曾提交通过20多件国家级发明专利，知识产权评审专家，CSDNCTO俱乐部会员与撰稿人。对分布式计算、软件架构、标签人格化等领域感兴趣。