链滴

老骥伏枥志在千里, 如今的 java 尚能饭否?

作者：washmore
原文链接：https://ld246.com/article/1498062409391
来源网站：链滴

<h2 id="写在前面">写在前面</h2>
从 1995 年第一个版本发布到现在，Java 语言已经在跌宕起伏中走过了 22 年，最新的 Java 版也已经迭代到 Java 9。当年 Java 语言的跨平台优势如今看来也只不过是家常小菜，Go、Rust 等语横空出世，进一步拓宽了编程语言的边界。当年发明 Java 语言的 Sun 公司早已被 Oracle 收购，Orac e 现在也正处于水深火热的云计算浪潮当中，甚至连 Java 之父 James Gosling 也加入了当今世界最的云计算公司 AWS。
Java 语言发展的这 20 年也正是全球互联网迅猛发展的 20 年，Java 语言同时也见证了电商浪潮移动互联网浪潮、大数据浪潮、云计算浪潮，所以在现今各大互联网公司身上都能看到 Java 的身影。/p>
纵看 Java 语言的发展，不禁让人联想到辛弃疾的一首词：
千古江山，英雄无觅，孙仲谋处。舞榭歌台，风流总被雨打风吹去。斜阳草树，寻常巷陌，人道奴曾住。想当年，金戈铁马，气吞万里如虎。元嘉草草，封狼居胥，赢得仓皇北顾。四十三年，望中记，烽火扬州路。可堪回首，佛狸祠下，一片神雅社鼓。凭谁问，廉颇老矣，尚能饭否？
TIOBE 的语言排行榜显示，自 2016 年初 Java 语言就出现了明显的下颓趋势，开发者社区也出了一些唱衰 Java 语言的论调，编者心中也有些许疑问：Java 老矣，尚能『饭』否？基于这样的背景，nfoQ 邀请到了 Java 资深专家张建锋来为大家解读 Java 语言的发展现状以及未来。
<h2 id="Java-语言的发展回顾">Java 语言的发展回顾</h2>
Java 语言源于 1991 年 Sun 公司 James Gosling 领导的的 Ork 项目，1995 年 Sun 公司正式名为 Java，并提出“Write once, Run anywhere"的口号。
1996 年 1 月 Java 1.0 发布，提供了一个解释执行的 Java 虚拟机，其时恰逢互联网开始兴起，Ja a 的 Applet 能在 Mozilla 浏览器中运行，被看作是未来的互联网语言。
1997 年 2 月 Java 1.1 发布，Java 语言的基本形态基本确定了，比如反射 (reflection), JavaBean 接口和类的关系等等，一直到今天都保持一致。然而，Java 最初的一些目标，如在浏览器中执行 App et，以及跨平台的图形界面 Awt 很快遭遇到负面的评价。
1998 年 12 月，Java 第一个里程碑式的版本，即 Java 1.2 发布了。这个版本使用了 JIT（Just in time）编译器技术，使得语言的可迁移性和执行效率达到最优的平衡，同时 Collections 集合类设计良，在企业应用开发中迅速得到了广泛使用。Sun 公司把 Java 技术体系分成三个方向，分别是 J2SE 面向桌面和通用应用开发），J2EE（面向企业级应用开发），J2ME（面向移动终端开发）。这个分影响非常久远，体现出主流语言设计者的思想：针对于不同的应用领域，在形态，API 集合等进行划。
2000 年 5 月，Java 1.3 发布，这个版本中 Corba 作为语言级别的分布式对象技术，成为 J2EE 一个技术前提。J2EE 受到 Corba 的设计的影响较大，早期 EJB 的 Home，接口和实现就是 Corba 在 C 语言的实现，被移植到 Java 语言之中。J2EE 中的 Servlet 规范获得了极大的成功，伴随着互联网兴起，和浏览器直接通过 HTTP 协议交互的 Servlet，和众多的 MVC 框架，成为 Web1.0 的网红。
2002 年 2 月，Java 1.4 发布，Java 语言真正走向成熟，提供了非常完备的语言特性，如 NIO 正则表达式，XML 处理器等。同年微软的.NET 框架发布，两者开始了为期十几年的暗自竞争。从语特性上来说，.NET 后发先至，一直处于优势。但 Java 依赖良好的开发者生态，绝大多数大型软件公的使用者众多和不断贡献，以及对 Linux 操作系统良好的支持，渐渐的在服务器端获得优势地位。</>
2004 年 9 月，Java 5 发布，Sun 不再采用 J2SE, J2EE 这种命名方式，而使用 Java SE 5, Java E 5 这样的名称。我认为 Java 5 是第二个里程碑式的版本。Java 语言语法发生很大的变化，如注解 (A notation)，装箱 (Autoboxing)，泛型 (Generic)，枚举 (Enum)，foreach 等被加入，提供了 java.util concurrent 并发包。Java 5 对于 Java 语言的推动是巨大的，特别是注解的加入，使得语言定义灵活很多，程序员可以写出更加符合领域定义的描述性程序。
2006 年 5 月， JavaEE 5 发布，其中最主要是 EJB3.0 的版本升级。在此之前，EJB2.X 版本被泛质疑，SpringFramework 创建者 Rod Johnson 在经典书籍“J2EE Development without EJB“，对 EJB2 代表的分布式对象的设计方法予以批驳。EJB3 则重新经过改造，使用注解方式，经过应用务器对 POJO 对象进行增强来实现分布式服务能力。在某种程度，可以说 EJB3 挽救了 JavaEE 的过消亡。
2006 年 12 月，Java 6 发布，这个语言语法改进不多，但在虚拟机内部做了大量的改进，成为个相当成熟稳定的版本，时至今日国内的很多公司依然以 Java6 作为主要 Java 开发版本来使用。同年 Sun 公司做出一个伟大的决定，将 Java 开源。OpenJDK 从 Sun JDK 1.7 版本分支出去，成为今天 penJDK 的基础。OpenJDK6 则由 OpenJDK7 裁剪而来，目前由红帽负责维护，来满足 Redhat Ente

prise Linux 6.X 用户的需要。
2009 年 12 月，JavaEE 6 发布，这个版本应该说是 JavaEE 到目前为止改进最大影响最深远的个版本。因为 JavaEE5 只有 EJB3 适应了 Java 注解语法的加入，而 EE6 全面接纳了注解。CDI 和 BenValidation 规范的加入，在 POJO 之上可以定义完备的语义，由容器来决定如何去做。Servlet 也级到 3.0 版本，并在接口上加入异步支持，使得系统整体效率可以大幅提高。EE 划分为 Full Profile 和 Web Profile，用户可以根据自己的需要选择不同的功能集。
在此之前，Oracle 已经以 74 亿美金的价格收购了 Sun 公司，获得了 Java 商标和 Java 主导权也收购了 BEA 公司，获得市场份额最大的应用服务器 Webogic。JavaEE 6 虽然是收购之后发布的版，但主要的设计工作仍然由原 Sun 公司的 Java 专家完成。
2011 年 7 月，Oracle 发布 Java 7, 其中主要的特性是 NIO2 和 Fork/Join 并发包，尽管语言上有大的增强，但我个人认为，自从 Oracle JDK（包括 OpenJDK7），Java 虚拟机的稳定性真正做到工业级，成为一个计算平台而服务于全世界。
2013 年 6 月，Oracle 发布 JavaEE 7，这个版本加入了 Websocket，Batch 的支持，并且引入oncurrency 来对服务器多线程进行管控。然而所有的子规范，算上可选项 (Optional) 总共有 40 多，开发者光是阅读规范文本就很吃力了，更不要说能够全局精通掌握。JavaEE 规范的本质是企业级用设计的经验凝结，每一个 API 都经过众多丰富经验的专家反复商议并确定。各个版本之间可以做到后兼容，也就是说，即使是 10 年前写的 Servlet 程序，当前的开发者也可以流畅的阅读源码，经过分代码调整和配置修改，可以部署在当今的应用服务器上。反过来，今后用 Servlet4 写的程序，浏器和服务器通信使用全新的 HTTP/2 协议，但程序员在理解上不会有障碍，就是因为 Servlet 规范的PI 非常稳定，基本没有大的变化修改。
2014 年 3 月，Oracle 发布 Java 8，这个版本是我认为的第三个有里程碑意义的 Java 版本。其最引人注目的便是 Lambda 表达式了，从此 Java 语言原生提供了函数式编程能力。语言方面大的特增加还有：Streams，Date/Time API, 新的 Javascript 引擎 Nashorn，集合的并行计算支持等，Jav 8 更加适应海量云计算的需要。
按照原来的计划，Java9 应该在今年 7 月发布，但因为模块化 (JPMS) 投票未通过的原因，推迟今年 9 月份发布。
JavaEE 8 也会在今年发布，预计的时间在 8-10 月。其中最主要更新是 Servlet 4.0 和 CDI 2.0 后者已经完成最终规范的发布和投票。
<h2 id="Java-社区情况介绍">Java 社区情况介绍</h2>
我们按照两个方面介绍 Java 社区情况。
Java User Group(JUG，Java 用户组) 目前全世界范围有 100 多个 JUG 组织，分布在各个大洲个国家，一般来说以地域命名。目前最有影响力的两个 JUG 分别是伦敦的 LJC(London Java Community) 和巴西的 SouJava，目前都是 JCP 的 EC(执行委员会) 成员。国内目前有 GreenTea JUG(北京和州)，Shanghai JUG，GuangDong JUG, Shenzhen JUG, Nanjing JUG 等。GreanTeaJUG 以阿里巴研发部门成员为核心，包括北京和杭州两地各个公司从事 Java 开发的研发人员，过去几年成功举了很多有业界影响力的活动，特别是邀请到众多国外的 Java 技术专家来分享知识，目前是国内最大的UG 开发者组织。
Java 开源社区 Java 是一门开放的语言，其开源社区也是参与者众多。最有名的应当数 Apache 区，目前已经拥有近 200 个顶级项目，其中绝大多数是 Java 语言项目。在 Java 生态圈中，具有重地位的如 Ant、Commons、Tomcat、Xerces、Maven、 Struts、Lucene、ActiveMQ、CXF、Cael、Hadoop 等等。很多技术时代，一大批 Java 项目加入，如 Web 时代的 Velocity、Wicket；JavEE 相关的 Tomee、OpenJPA、OpenWebBeans、Myfaces；WebService 时代的 jUDDI、Axis、SrviceMix；Osgi 时期的 Flex、Karaf；大数据时代的 HBase、Hive、ZooKeeper、Cassandra；云代的 Mesos、CloudStack 等等。
涉及到软件开发的方方面面，可以说当今几乎所有的中型以上 Java 应用中，都会有 Apache 开项目的身影。国内最早参与 Apache 社区的以国外软件公司国内研发团队成员为主，如红帽、IONA、ntel、IBM 研发中心等。如今国内互联网公司和软件公司也不断的参与，特别是开始主导一些 Apache 项目，如 Kylin 等。
JBoss 开源社区，包含了 50 多个 Java 开源项目，其中有 Hibernate、Drools、jBPM 等业界知开源项目，也有 Undertow、Byteman、Narayana 等名气不算大，但绝对是相应领域业界的顶级优项目。当前 JBoss 开源社区主要以企业应用中间件软件为主，RedHat 是主要的技术贡献力量。
Eclipse 开源社区，之前主要是包含 Eclipse IDE 的项目，后来也逐步进行多方面的扩展，比如 OGi，服务器等，目前一些知名 Java 项目，如 Jetty、Vertx 等都是 Eclipse 开源组织成员。此外 IOT 前是 Eclispe 的一个重点方向，在这里可以找到完整的 IOT Java 开发方案。

Spring 开源社区，以 SpringFramework 为核心，包括 SpringBoot、SpringCloud、SpringSecurity、SpringXD 等开源项目，在国内有广泛的应用场景。
<h2 id="目前大的玩家">目前大的玩家</h2>
Java 语言和品牌都是 Oracle 公司所有，所以 Oracle 公司是 Java 最主要的厂商。绝大多数 JSR(ava 规范提案) 的领导者都是 Oracle 的雇员。
Java 是一个庞大的生态圈，全世界的软件和互联网公司绝大多数都是 Java 用户，同时也可以参推动 Java 语言的发展。任何组织或者个人都可以加入 JCP（Java Community Process），并提交 JS 来给 JavaSE，JavaEE，JavaME 等提交新的 API 或者服务定义。Java 拥有当今最完备的语言生态，乎所有能想到的应用范围，都有软件厂商提出过标准化的构想，其中很多已经被接纳为 JSR 提案。如 JSR 总数已经都 400 多个。
JCP 是发展 Java 的国际组织，其中的执行委员会（EC）以投票的形式对 JSR 提案进行表决。目 EC 包括 16 个合约 (Ratified) 席位，6 个选举 (Elected) 席位和 2 个合伙 (Associate) 席位，以及 Oracle 作为所有者的永久席位。非永久席位每两年重新选举一次，每次选举为 24 个席位的一半，即为 12 个。
当前 EC 委员会中，对于 Java 起到最重要作用的，无疑是 Oracle，IBM 和 Redhat 三家公司。Oracle 自然不用说；Redhat 领导着 JavaEE8 中两项 JSR，并且在操作系统，Linux，虚拟化，云计算基础软件方面是产品领导者；IBM 是软硬件最大的厂商，拥有自己的 Unix 操作系统和 JDK 版本。这家软件厂商也是中间件厂商的强者，它们对于 Java 的影响是至关重要的。前不久投票被否决的 JSR 36(JPMS) 模块化提案，就是 Redhat 和 IBM 先后表示要投反对票，最后才没有通过的。
另外的几个重要的 Java 参与方分别包括：巨型互联网公司，以 Twitter 为代表；大型金融公司以高盛，瑞信为代表；强大的硬件产商，Intel，NXP，Gemalto 等；大型系统方案厂商，HP， Fijits；当然还有掌握先进 Java 技术的公司，如 Azul，Hazelcast，Tomitribe，Jetrains 等等。这些公司同对 Java 的发展起到关键作用。
<h2 id="GC-方面的进展">GC 方面的进展</h2>
JDK 中主要的 GC 分类有：

Serial，单线程进行 GC，在它进行垃圾收集时，必须暂停其他所有的工作线程，直到它收集结。

Parallel，相比 Serial 收集器，Parallel 最主要的优势在于使用多线程去完成垃圾清理工作，这可以充分利用多核的特性，大幅降低 GC 时间。

CMS(Concurrent Mark-Sweep)，是以牺牲吞吐量为代价来获得最短回收停顿时间的垃圾回收。实现 GC 线程和应用线程并发工作，不需要暂停所有应用线程。

G1(Garbage First Garbage Collector)，G 设计初衷是为了尽量缩短处理超大堆（大于 4GB）产生的停顿。相对于 CMS 的优势而言是内存碎片的产生率大大降低。

目前在 JDK8 中以上 4 种 GC 都可以使用，而在 JDK9 中 G1 GC 会成为默认的垃圾收集器。</>
在 OpenJDK 方面，Redhat 开源并贡献了 Shenandoah GC。这是一种新的 Java 虚拟机 GC 法，目标是利用现代多核 CPU 的优势，减少大堆内存在 GC 处理时产生的停顿时间。在使用大内存应用上使用，如 >20G 堆空间。Fedora24 以后，官方源中的 OpenJDK 即带有 Shenandoah 算，不过 JDK9 中还不会被加入。
无停顿的高性能 GC 就是 Azul 公司的 C4(Continuously Concurrent Compacting Collector) G 了，但只提供商业版本使用。
另外 IBM J9 中 Balanced GC，表现也很出色，能够保证相对一致的暂停时间而避免破坏性的长间停顿。Balanced GC 应用在各类 IBM 中间件产品之中。
<h2 id="Java-9-目前已经可以确认的特性介绍">Java 9 目前已经可以确认的特性介绍</h2>

Java9 中，最受人关注的新特性就是 Jigsaw 项目带来的模块化技术特性。
Java 语言一直缺乏语言级别的模块化能力，目前模块化技术通过 OSGi, JBoss Modules 等项目已经在服务端程序得到了广泛的应用。Java 在语言级别引入模块化能力，将极大的促进 Java 应用程组件化，模块化的改变。应用程序通过模块化拆分，可以做到更灵活的引入，加载，移除组件，占用少的内存，更适合云计算时代的要求。在 JDK9 EA（预览版）中，原有的 rt.jar 已经被划分为若干了 mod，通过模块内的 module-info.java 文件来声明模块间的引用关系。
然而，模块化改造是个渐进而适度的过程，Java9 为了可兼容 Java8 以前应用程序的运行，做出多的让步，模块定义严格性没有那么苛刻。各个厂商也有对自己现有系统可无缝运行在 Java9 上的商诉求。Java 模块化提案还得花更多的时间去讨论和修改。
Java9 中的 jshell 工具实现了 REPL，即读取，求值，打印，循环。这个工具可以使得开发者交式的使用 Java，方便于系统管理，调试，使用。可以想像到有了 jshell 后，Java 语言更加适合初学入门学习。
Jlink 工具和 AOT（预先编译技术）。一直以来，Java 运行方式是把程序编译成 class 文件，然通过 jvm 运行的。这种工作方式可以做到跨平台移植，在互联网时代初期，各种 Unix 繁荣和 Windos 在桌面的一统局面下，对于占据市场起到决定性作用。
然而到了今天，无论是大型互联网公司还是企业内部，x86 平台 64 位服务器已经成为主要的选。从运行效率考虑，可以把 java 程序编译成可执行的二进制文件，更加适应云计算和容器技术发展需要。
利用 jlink/jaotc 工具，可以把一个 Java 程序编译成可执行文件，在 Java9 推出时，可能只有 jaa.base 模块支持 AOT。
安全方面的加强。引入新的摘要算法 SHA-3，内置 ALPN 使得更好的支持 HTTP/2 协议，提供 TLS（数据包传输层安全性协议），可以保证 UDP 数据传输的安全，PKCS12 格式替代原有的 JKS 为 keystore 的默认格式。
此外，统一 JVM 日志 (Unified JVM Logging)，多版本共存 jar(Multi-release jar files)，接口部的私有方法 (Interface provate method) 等也是非常重要的新特性。
<h2 id="与其他语言的对比-Java-的优势">与其他语言的对比，Java 的优势</h2>
Java 是最好的语言么？不是，因为在每个领域都有更合适的编程语言。
C 语言无疑是现代计算机软件编程语言的王者，几乎所有的操作系统都是 C 语言写成的。C++ 面向对象的 C 语言，一直在不断的改进。
JavaScript 是能运行在浏览器中的语言，丰富的前端界面离不开 Javascript 的功劳。近年来的 Node.js 又在后端占有一席之地。Python 用于系统管理，并通过高性能预编译的库，提供 API 来进行科计算，文本处理等，是 Linux 必选的解释性语言。
Ruby 强于 DSL（领域特定语言），程序员可以定义丰富的语义来充分表达自己的思想。Erlang 是为分布式计算设计的，能保证在大规模并发访问的情况下，保持强壮和稳定性。Go 语言内置了并能力，可以编译成本地代码。当前新的网络相关项目，很大比例是由 Go 语言编写的，如 Docker、Kbernetes 等。
编写网页用 PHP，函数式编程有 Lisp，编写 iOS 程序有 Swift/ObjectiveC。
一句话概括，能留在排行榜之上的语言，都是好的语言，在其所在的领域能做到最好。
那么，Java 语言到底有什么优势可以占据排行榜第一的位置呢？
其一，语法比较简单，学过计算机编程的开发者都能快速上手。
其二，在若干了领域都有很强的竞争力，比如服务端编程，高性能网络程序，企业软件事务处理分布式计算，Android 移动终端应用开发等等。
最重要的一点是符合工程学的需求，我们知道现代软件都是协同开发，那么代码可维护性，编译检查，较为高效的运行效率，跨平台能力，丰富的 IDE，测试，项目管理工具配合。都使得 Java 成企业软件公司的首选，也得到很多互联网公司的青睐。
没有短板，容易从市场上找到 Java 软件工程师，软件公司选择 Java 作为主要开发语言，再在特的领域使用其他语言协作编程，这样的组合选择，肯定是不会有大的问题。
所以综合而言，Java 语言全能方面是最好的。
<h2 id="Java-未来方向的展望">Java 未来方向的展望</h2>
如今的 Java，已经在功能上相当丰富了，Java 8 加入 Lambda 特性，Java 9 加入模块化特性之，重要的语言特性似乎已经都纳入进来。如果要说值得考虑的一些功能，我觉得有以下几点：
模块化改造完毕之后，可能会出现更多专业的 JDK 发行软件商，提供在功能方面，比如针对于布式计算，机器学习，图形计算等，纳入相关的功能库作为文件。这样专业行业客户可以选择经过充优化后的 JDK 版本。

Java 语义上对“模式匹配”有更强的支持，如今的 switch 语句能力还是比较欠缺，可以向 Erlan, Scala 等语言借鉴。
多线程并发处理，Java 做的已经很好了。不过我个人觉得可以在多进程多线程配合，以及语言别数据管道表示上，可以进行改造和优化。
JDK9 会有 HTTP/2 client 端的能力，但毫无疑问会有更多更好的三方库出现，JDK 可以和这些方库通力合作，提供一个更好 API 界面和 SPI 参考实现。
目前 Java 在云计算方面遇到的最大问题还是占用内存过大。我个人认为从两个方面来看：

如果该应用的确是长时间运行的服务，可以考虑结构清晰的单体结构，算下来总的内存消耗并不比多个微服务进程占用的更多。

微服务应用，未来可以采用编译成本地代码的方式，并使用优化过的三方库，甚至本地 so 文件减少单个进程的过多内存占用。

安全框架更加清晰，SPI 可以允许三方库提供更强大更高效的安全功能。

JavaEE 方向则有更多的改进的地方：

EJB 重构目前的 Corba 分布通信基础，参考 gRPC 进行远程系统调用。

分解 EJB 规范，把 JVM 进程相关的特性，如注入 / 加强 / 事务 / 安全都统一到 CDI 规范中；对 JB 进行裁剪，保留远程访问特性和作为独立执行主体分布式对象能力。

加强 JMS 和 MDB，媲美 Akka 目前的能力。

JaxRS 适度优化，不必要依赖 Servlet，或者适度调整，来提供更大的能力。

JPA 借鉴 JDO，以及融入一部分特性，做到对 NoSQL 更良好的支持。

<h2 id="一些个人的心得和经验分享">一些个人的心得和经验分享</h2>
软件业有个 Hype Cycle 模型，有很多技术受到市场的追捧而成为明星，也有些身不逢时而备受漠。
EJB 是一个广泛被误解的技术，在企业应用分布式计算方面，EJB 给出了非常完备的技术体系。是目前所有的应用服务器都实现的不够好。对于目前打算转型微服务设计的架构师，EJB 也是一个非值得学习借鉴的技术。
Java 的慢是相对的，有些是当前实现的不够好。比如原来有人对 Java 的网络 IO 性能提出质疑然而稳定的 Netty 框架出现后，就没有人再怀疑 Java 处理网络 IO 的能力了，甚至在 JDK8 中自身的 NIO 也相当出色。要知道 Java 为了实现跨平台能力，采用的是各个操作系统的一个公共能力子集，且其设计哲学就是给出 API 框架，实现是可以自行实现和加载服务的。
Java 在处理界面方面，Swing 和 Swt 表现可圈可点（Idea 和 Eclipse 分别采用的图形基础库）JavaFX 已经运用到很多的行业软件上。在浏览器界面表现上，SpringMVC 在模板渲染页面方面使用最多；GWT 似乎使用者不多，但基于 GWT 的 Vaddin 在国外企业中用户众多，而且很多服务器管软件也用 GWT 写成；JSF 也在企业软件中得到广泛使用，状态信息直接在后端进行管理，配合 js 前框架，可以充分发挥各种技术的优势。

CDI 规范和 SpringFramework 在服务器程序中作用类似，Spring 是一套设计优良，完备的框，CDI 具有更强的可扩展性。通过对注解的语义定义，一家公司可以维护一套自己的组件描述语言，做到产品和项目之间的软件快速复用。CDI 是定义软件组件内部模型的最佳方式，只可惜了解的软件程师实在太少。
微服务架构在互联网应用，快速开发运维管理方面，配合容器技术使用，有很强的优势。但并不所有的应用场景都适合微服务：强事务应用系统，采用单体结构的软件体系设计，更容易从整体方面护，也能获得更优的性能。Java 语言无论在微服务还是单体结构，都有成熟稳定的软件架构供选择使。
转自 老骥伏枥志在千里，今的 java 尚能饭否？