Java 领域从传统行业向互联网转型你必须知道的事儿

我为什么要写这篇文章

武林中，”天下武功出少林”指各门各派的武功都与少林武学有一定的渊源，技术也是相同的道理，对于Java领域的应用而言，传统行业与互联网行业的技术都来自J2SE和J2EE的生态圈，但是两个行业的侧重点不同，传统行业侧重于严格的规范、复杂的流程、丰富的功能，因此或多或少的都会使用J2EE规范定义的技术，Appserver是J2EE规范的完全实现，因此，传统行业的企业级软件开发基本都是部署在Appserver上的，这样可以重复利用Appserver提供的通用功能而节省开发和实现的工作量，而后者更注重互联网产品的非功能质量需求，通常包括：高可用、高性能、安全性、可伸缩、可扩展等，互联中最流行的一句话是：天线武功唯”快”而不破，充分看出互联网企业里程序性能的重要性，为了达到较好的性能，高度抽象的J2EE技术已经没法满足需求，因此互联网技术更倾向于在简单的J2SE上发展具有互联网特色的技术栈，重新定义互联网级的开发工具、平台和技术栈。

由于笔者从传统的外企转型到互联网已经有3个年头，近两年来面试了很多来自传统行业的同行们，笔者发现这些同行们都有意向走进处于风口的互联网，但是由于传统行业使用的技术栈与互联网有所不同，不知从哪里开始入手准备和提高，尽管他们有强烈的学习和提高的愿望，本文就是给这些想从传统行业跨入互联网的小伙伴们准备的一篇导向性文章，帮助读者了解互联网的技术栈、了解互联网的侧重点、了解互联网的核心技术，并给出如何以传统行业的技术栈为基础快速掌握互联网的核心技术，其实，那只有一墙之隔，捅破那张窗户纸儿，一切都豁然开朗。

这里需要再次澄清，我并不认为互联网行业的技术要比传统技术深奥多少，这些技术跑不出J2SE和J2EE的生态圈，只不过高度抽象的J2EE技术由于性能上的局限性而被互联网撇弃而已，但是不得不承认的是两个行业的侧重点不同，传统行业侧重于规范，流程，功能的复杂性以及正确性，而互联网更侧重于“快”，这里的“快”有两方面的意思，一个是产品运行效率要高，响应速度要快，另外一个是开发效率要快，响应市场需求要快。从另外一个侧面说，传统行业一般关注一个复杂系统的功能完善和丰富，而互联网企业更关注一个简单的垂直业务的非功能质量，例如：高性能，可用性，高并发，可扩展，可伸缩，安全性等，那么，一个从业人员从传统行业到互联网行业，你到底还有多少距离？

小伙伴们从哪里开始入手互联网

这两年来面试下来看到了一个普遍的现象，来自于传统行业的技术人员，他们大多数掌握的技能是SSH，稍微资深一点的工程师对J2EE规范有所了解，他们仍然在使用J2EE规范的EJB, JPA, JMS, JCA, JAAS等技术，数据库基本上使用Oracle，DB2，Sqlserver等等。传统行业的开发人员基本实施“模块包揽制”，这得益于J2EE规范的完整性，以及Appserver提供了基本所有架构需要的功能，开发人员只需要将各个业务模块填入J2EE和Appserver提供给你的框架即可，因此，一个传统的开发人员会包揽一个模块从前台到后台所有的工作，这包括：HTML, JS, CSS, EJB, JPA, SQL, PLSQL等等。这些技术是不是一无是处，当然不是，反而是非常有价值的，那有了这些技术，我们是否可以一步跨入互联网，也不是，还需要以这些技术为基础，进一步扩展技术视野，对欠缺的技术广度和深度进行不足。

下面就学习传统行业技术人员拥有哪些技术积累，下一步又如何补充自己的知识面，成为能够胜任互联网行业的优秀技术人员呢？

消息队列

在传统行业，相信你一定用过JMS，作为J2EE规范的一部分，所有的Aappserver（Weblogic、Websphere、Jboss等）都有JMS的实现，那你一定知道JMS包含Queue和Topic两种Subject，你也知道Send/Receive和Publish/Subscribe两种收发模式，那在互联网为什么就不用这些呢？

原因主要有两个，一个是商业的Appserver都是收费的，然而，互联网提供的产品是免费的，互联网使用的产品也多是免费的，另外一个原因就是这些Appserver的实现性能差，有测评显示ActiveMQ比JbossMQ速度要高出10倍，在某些应用场景下ZeroMQ的速度要高出一个数量级，可达到微妙级别的延迟，有兴趣可以参考ZeroMQ的性能测试页面。

除此之外，一些开源的MQ的实现针对互联网业务，提供了除Queue和Topic的支持，还有partition，group，broker等更复杂的消息模型，具体参考Kafka, Kafka的设计具有使用简单、功能丰富、高性能等优点，不但天生具有持久、分片、复制等功能，而且在使用上对开发者和运维的体验也很好。对于Kafka的中间件设计，请参考我的博客文章简单易用的消息队列框架的设计与实现。

那么如果你在传统行业掌握了JMS规范定义的消息队列技术，你只需要再往前走一步，请深入学习开源的Kafka、RockitMQ、ActiveMQ、RabbitMQ、MemcacheQ、Redis、ZeroMQ、MSQ等。

缓存

在传统行业，相信大家都用过Oscache和Ehcache, 前者主要针对网页的缓存，后者主要针对数据库数据的缓存，通常可作为Hibernate的二级缓存，相信有些人还用过Jboss Cache，这是一个分布式企业级可实时复制的Cache，有些人在项目中也写了自己的缓存，甚至在一些项目中直接使用Hashtable作为缓存，其实这些缓存加速了特定场景下的数据访问，对你的项目成功起到了至关重要的作用。

但是互联网行业则从另外一个角度来使用缓存，主要应用场景有两个：第一，大量的数据需要集中保存，在服务的任意节点上可以访问缓存中的任意数据，也就是需要数据的中心存储，而且还要满足快速的查询需求的场景；第二，数据库读性能是有瓶颈的，廉价硬件机器上的单机Mysql读操作吞吐量在1000/s左右，大量的读查询会压垮数据库，这需要使用缓存来抗住读流量，通常应用在有热点数据的场景。

从这两个应用场景来看，互联网行业更关心分布式缓存，那数据如何分布呢？很简单，Hash或者一致性Hash，所以，咱们可不可以先把Oscache和Ehcache放一边，来研究一下Redis，Memcache或者淘宝的Tair呢？最简单的办法从Redis和Memcache的区别开始入手？可参考我的博客文章Redis vs. Memcache，这里博客文章只是简单的介绍了Redis和Memcache的区别，要想深入学习缓存技术推荐大家仔细阅读Redis设计与实现一书。

除了要学习分布式缓存，例如：Redis、Memcache本身的功能和技术点外，最主要的要有缓存分片的思想，在互联网里大多数的热数据都是缓存在缓存服务中的，这需要大量的缓存服务器，单台机器是不能满足需求的，那缓存分片是一个大话题，缓存分片的实现方式一般有如下3种：

> 1. 通过代理层实现，例如Codis，在代理层实现数据的路由，对应用层透明。
> 2. 客户端分片，可参考我的开源项目redic，实现简单、使用简单、支持分片、复制、失效转移等功能。
> 3. 缓存服务器支持的高可用模式，例如：Redis 3.x、Sentinel等。

服务框架

在传统行业，相信大家都使用EJB和Webservice来提供服务的导出和导入，有些个别传统行业不用APP服务器，仅仅使用JDK的RMI来导出和导入服务，但是为什么互联网偏偏不喜欢这些技术呢？Webservice使用重量级的SOAP协议，臃肿的XML满世界都是，性能上的去吗？ 那互联网用什么，互联网使用轻量级的RPC框架和RESTful服务，前者使用轻量级的序列化框架，例如：Google的ProtoBuffer, 还有Hessian和Burlap等序列化协议，后者则使用简单的HTTP协议，前者适合在内网做高性能的服务调用，而后者适合异构平台的服务调用，例如: 跨语言，跨防火墙，前后台之间等。RPC远程调用请参考阿里的Dubbo框架和Twitter的Finagle框架，至于Rest框架参请考Spring Web MVC，Spring Boot、Jersey，Apache CXF等。

在互联网的世界里，几乎所有的公司都实现了服务化，服务化导致的问题就是一致性问题，如何解决高并发系统的一致性呢？使用两阶段提交协议、三阶段提交协议、TCC？还是遵循ACID原理、CAP原理、BASE原理？如果我们保证的是最终一致性模型，我们都有哪些模式可以应用，请参考我的博客文章保证微服务架构一致性的公开课，这篇文章里有视频、PPT和完整的一致性保证的文章帮助大家学习一致性保证的最佳实践和完善的理论体系。

最近微服务变得越来越流行，微服务实际上是服务化的一个延续，是更细致化的服务化的架构，微服务的服务框架的代表是Spring Cloud，它与Netflix集成，提供了限流、熔

断、仓壁隔离、失效转移等为服务化中必不可少的高级特性，大家可以到官网文档进一步学习Spring Cloud相关技术。

数据库

在传统行业，大多数人开发人员都使用Oracle, DB2, Sqlserver数据库，其实，从功能和性能上来讲，他们都不亚于Mysql, 甚至比Mysql更优秀，但是Mysql是免费的，这使得Mysql得到互联网行业的青睐。

那么我们分析下，传统行业的人员在数据库方面欠缺什么吗？首先，Oracle和Mysql都使用B+树索引，原理相同，使用方法相同；Oracle支持行级锁，Mysql Innodb同样支持行级锁；Oracle Dataguard支持数据复制，Mysql也支持数据复制，但是Mysql的复制模式更灵活，并且支持主主配置。前面这些都是大同小异，如果你理解了相应的Oracle技术，你用很少的时间就可以掌握Mysql的相关技术。但是不同点是，Oracle虽然支持集群，通过增加服务节点的方式可以增加服务性能，但是集群的节点数量是有限的，并且数据存储是共享的，所以扩容基本采用垂直方式，然而使用Mysql则采用水平扩展，也就是需要进行手工的分区分表，对数据进行分而治之，以满足日益增长的读写压力以及数据存储压力。因此，如果想向互联网转行，一定要学好Mysql，推荐阅读《高性能Mysql》，这本书是必读的书籍，而且推荐每一个应用开发人员都要通读全书，而不是仅仅读其中与应用相关的那部分。

在互联网行业里面对性能追求到达了极致，因此会要求开发人员对数据库原理有所了解，其中最重要的部分就是索引，关于数据库的索引原理可参考文章MySQL索引背后的数据结构及算法原理。

负载均衡

刚才谈到，高并发系统，压力山大的时候怎么办？思想只有一个分而治之（ divide-and-conquer ）。因此，负载均衡则非常重要，传统行业以销售产品为盈利模式，因此，大多数项目在需要负载均衡的时候，多使用F5硬件负载均衡。

实际上传统的J2EE规范的EJB也可以分布式发布，通过JNDI的集成，也可以进行一定程度的负载均衡，但是这个负载均衡显得太重量级，用起来非常的不方便，效率也很低，并且和APP服务器绑定。

那么互联网呢？多采用软负载均衡，你必须了解LVS，nginx, Apache, Varnish, Haproxy等七层和三四层负载均衡原理和产品。

JVM

另外，在互联网行业做Java开发，一定要对JVM有所了解，并且进行深入的研究，例如：GC，类加载，Hotspot编译器，多线程、并发和锁，IO和NIO等。推荐阅读《深入理解Java虚拟机++JVM高级特性与最佳实践》，《深入理解Java7》，《Java Concurrency In Practice》，《Pro Java 7 NIO.2》，《Java Performance》等一系列深层次的JVM相关数据，

最好能阅读《The Java® Language Specification》和《The Java® Virtual Machine Specification》两本龙书，这些书籍都可以从Java核心必读书籍共享这篇文章下载。

大数据与云计算

作为一个IT从业人员，一定要跟上技术潮流，像云计算，大数据，CAP, BASE, 选主算法等概念不得不去了解，对于热点技术不得不研究，例如： Hadoop, Hbase, Zookeeper, Openstack, Dooker, Kafka, Storm等。

对于大数据技术推荐阅读Google三大论文，Mapreduce, GFS和BigTable，这三篇论文是大数据处理的鼻祖型论文，最核心的分布式存储、分布式处理、大文件的高效存储等技术都在这里，大数据鼻祖的三大论文可以从我的博客文章谷歌大数据的三驾马车下载。

性能评估和容量估算

如果你决定要来互联网一显身手，你必须学会性能评估和容量估算，这包括对前端机、缓存、消息队列、数据库等各个性能指标的估算，例如：吞吞量，响应时间，内存，CPU，IO，网络IO等。

性能和容量评估的方法论和典型案例可参考文章互联网性能与容量评估的方法论和典型案例。

为了确保架构设计的合理性，性能和容量评估是在架构设计初期完成的，用来证明架构方案可行，但是在项目实施中和实施后，还需要对项目的进行压测，来证明项目按照既定的目标而推荐和完成，关于性能测试的方法论和设计流程，我将会在后续文章中介绍给读者。

互联网架构方法论

在互联网行业里，处理大规模高并发的用户请求的核心思想只有一个，那就是“分而治之”，因此，通常业务被拆分为多个职责单一的服务，在某一个单服务里，业务逻辑并不复杂，但是对非功能质量需求的要求较高，这通常表现在性能、可用性等方面，因此互联网的架构设计中首要考虑的是非功能质量，这和传统行业注重功能和业务流程的情况有所不同，对于互联网行业中，架构设计的案例，可以参考这篇原创发号器Vesta的设计与实现如何设计一款多场景分布式发号器（Vesta），来了解互联网业务的架构设计的风格和思路。

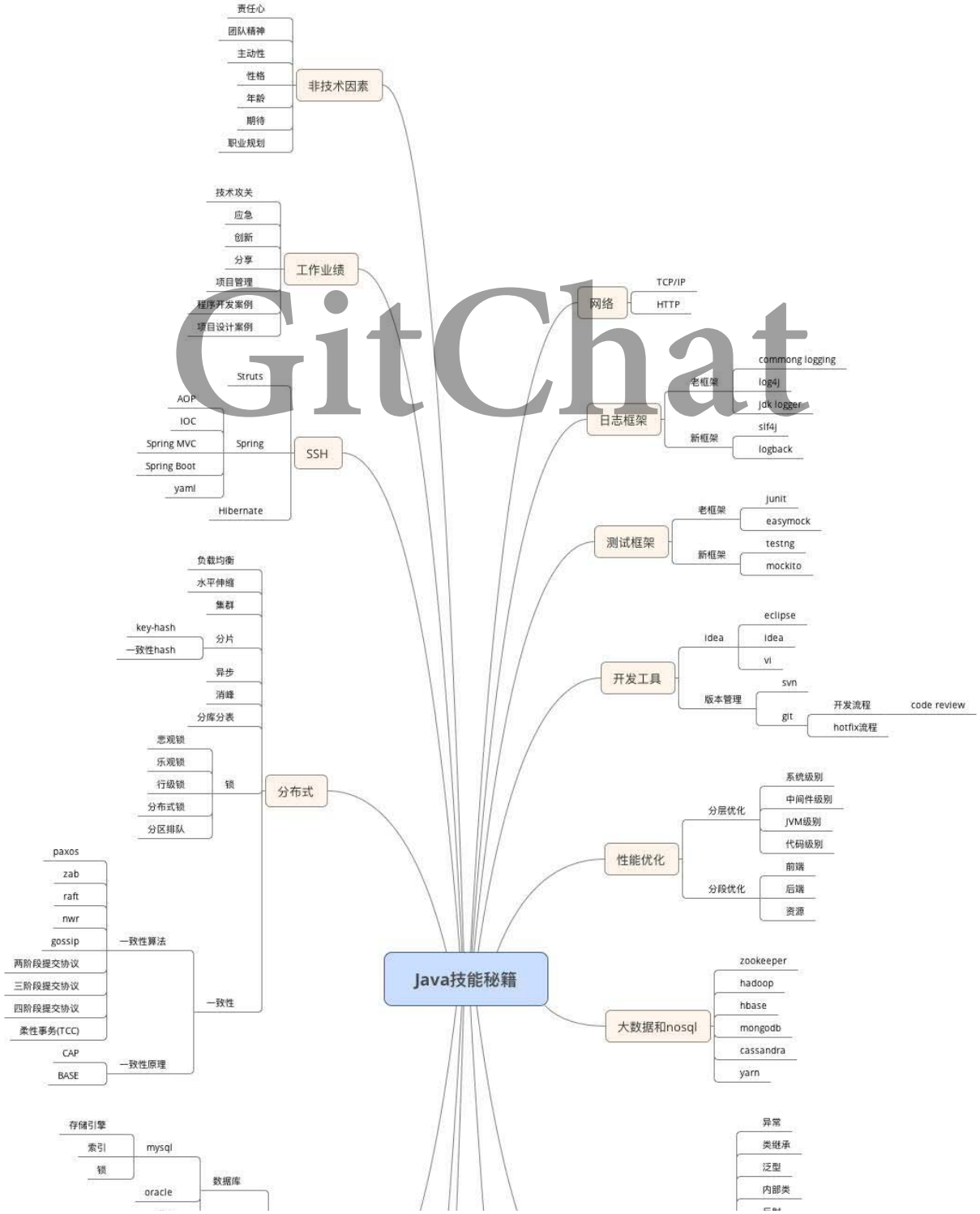
技术攻关和线上应急

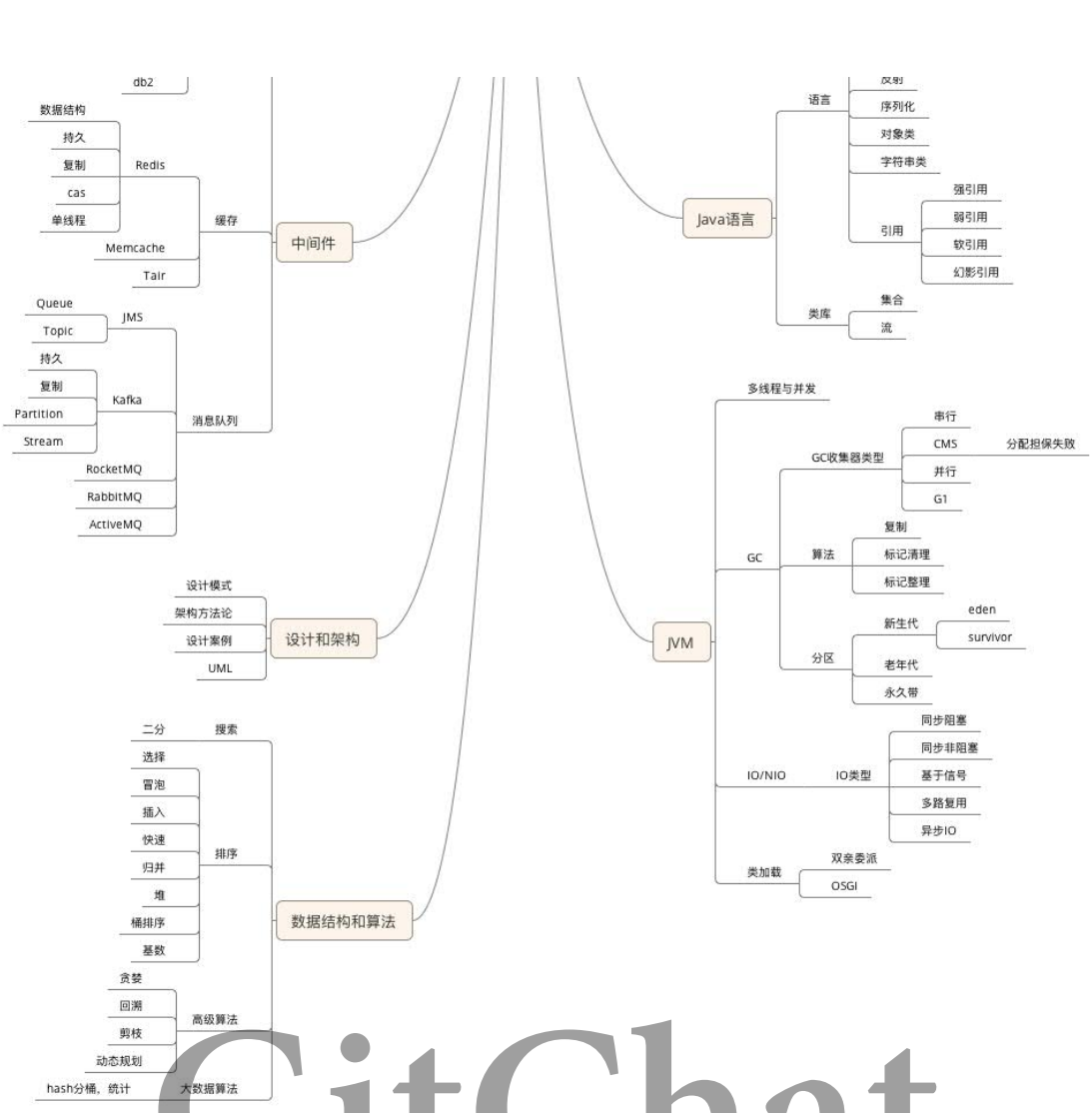
在互联网企业里，大多数产品都是针对用户端的，用户端的产品的特点是拥有海量的用户、产品要能够处理海量用户产生的大规模高并发的用户请求，因此会对产品的可用性比较敏感，在这种环境下，技术攻关和线上应急显得尤为重要，例如：如何解决线上线程卡死问题、如何解决OOM问题、如何解决服务超时问题等，可以参考如下两篇文章：

Java服务化系统线上应急和技术攻关，你必须掌握的Linux命令和Java服务化系统线上应急和技术攻关，你必须拥有的那些应用层脚本和Java虚拟机命令。

向这里看你会豁然开朗

希望这篇文章能够帮助更多的传统行业的从业人员转入互联网，在互联网的大舞台上展现你的才能，最后，附赠一张笔者在互联网行业里通过面试识别人才的《Java技能图谱》，大家可以根据其中的思维导图来深入学习各项知识点，每个知识点都需要系统的学习，或者看一本书或者查询相关的资料，切记要积累知识的广度的同事也要有一定的深度。




db2
数据结构
持久
复制
cas
单线程
Redis
Memcache
Tair
缓存
Queue
Topic
JMS
持久
复制
Partition
Stream
Kafka
RocketMQ
RabbitMQ
ActiveMQ
消息队列
中间件
设计模式
架构方法论
设计案例
UML
设计和架构
二分
搜索
选择
冒泡
插入
快速
归并
堆
桶排序
基数
排序
贪婪
回溯
剪枝
动态规划
高级算法
hash分桶，统计
大数据算法
数据结构和算法
反射
序列化
对象类
字符串类
强引用
弱引用
软引用
幻影引用
引用
语言
集合
流
类库
Java语言
多线程与并发
串行
CMS
分配担保失败
并行
G1
GC收集器类型
复制
标记清理
标记整理
算法
GC
eden
survivor
新生代
老年代
永久带
分区
同步阻塞
同步非阻塞
基于信号
多路复用
异步IO
IO类型
IO/NIO
双亲委派
OSGI
类加载
JVM