一个牛人给 java 初学者的建议

给初学者之一：浅谈 java 及应用

学 java 不知不觉也已经三年了

从不知 java 为何物到现在一个小小的 j2ee 项目经理

虽说不上此道高手，大概也算有点斤两了吧

每次上网，泡 bbs 逛论坛，没少去 java 相关的版面

总体感觉初学者多，高手少，精通的更少

由于我国高等教育制度教材陈旧，加上 java 自身发展不过十年左右的时间

还有一个很重要的原因就是 java 这门语言更适合商业应用

所以高校里大部分博士老师们对此语言的了解甚至不比本科生多

在这种环境下，很多人对 java 感到茫然，不知所措，不懂 java 能做什么

即便知道了 java 很有用，也不清楚该从哪里入手

所以就有了 java 入门难 这一说法

ok，那我们就从 java 到底能做什么聊起

先说什么是 java

java 是一种面向对象语言，真正的面向对象，任何函数和变量都以类(class)

封装起来

至于什么是对象什么是类，我就不废话了

关于这两个概念的解释任何一本面向对象语言的教材里面都有

知道了什么是 java，那自然就会对 java 能干什么感兴趣

在说 java 能做什么之前，先说 java 作为一个真正面向对象语言的优点

首先第一个，既然是真正的面向对象，那就要做到彻底的封装

这是 java 和 c++最大的不同，java 所有的源码以及编译后的文件都以类的形式存在

java 没有所谓的类外部定义，所有的函数（方法）以及变量（属性）都必须在类内部定义

这样就不会出现一个类被切割成这里一块那里一块的情况，c++就可以，不是么？

这样做使得整个程序的结构异常清晰，明了

其次第二个，最让人欢呼雀跃的是完全屏蔽了指针，同时引入了垃圾回收机制

任何一个写过 c/c++代码的人，都会对内存管理深恶痛绝

因为这使得我们不能把主要精力放在我们关心的事情上

而需要考虑计算机内部的一些事情，作为一个软件工程师

我想没有一个人愿意把大量的时间花在内存管理上，毕竟我们不是电子工程师

此时 java 的优势体现出来了，它完全屏蔽了内存管理

也就是说，如果你用 java 写程序，写出来的任何一个程序内存上的开销，都不受你控制

乍一看，似乎你受到了束缚，但实际上不是这样

因为虽然你的程序无法对内存进行管理，降低了一定的速度

但你的程序会非常非常的安全，因为你无法调用一个空指针

而不像以前写 c 的时候那样，成天因为空指针而担惊受怕

当然，如果你深入了解这一行，就会发现 java 其实也无法保证程序不去调用空的指针

但是它会在最大程度上避免空指针的调用

这已经很好了，安全，这是 java 的最突出的优点

第三个，虚拟机跨平台，这是 java 最大的特点，跨平台

可能所有人都知道 windows，但是不是所有人都知道 unix

和 java 一样，很多人都不知道 unix 这种操作系统干什么用

我不想多说 unix 的应用，这不是主要，但是我要说，大部分小型机

工作站，都跑在 unix 一族的操作系统上，比如 linux/solaris

unix 比起 windows 有一个最显著的特点，稳定，这就好比思科和华为

思科的机器慢但稳定，华为的机器快但不稳定，作为服务器这一端来说

要的皇怯卸嗫欤　　俏榷 ǎ　　評 nix 在服务器端还是非常有市场的

而且很重要的 windows 不安全，在 ms 的宣传中我想所有人都很少看到安全二字

因为 windows 操作系统针对的是 pc 用户，pc 死机就死机咯，大不了重启

瘟 95 最经常冒出来的就是蓝屏，在服务器这一端上因为 ms 没有自己的芯片

所以要做系统有些力不从心啊。扯远了，那么 java 可以做到在 windows 上编译

然后在 unix 上运行，这是 c/c++做不到的

那么说到这里，java 能做什么逐渐清晰起来

刚才说到了，java 程序有一个的特点是安全

这个安全是针对你的系统来说得，系统在跑了 java 程序之后会特别地稳定

而且还能跨平台，那么很明显，java 主要应用于除了 windows 操作系统以外所有的平台

比如手机，服务器

想想看，如果你写的程序要跑在手机上，而手机有多少款用的是 windows?

就算有，那如果你用 c/c++，是不是要针对每一款手机写一套程序呢？

累死，那跨平台的 java 就不用，做到编译一次，随时运行

同样，在服务器这一端，如果我想给一个网络门户站点，比如 sina

写一个应用程序，pc 的性能肯定无法满足 sina 这样大站点并发数量的要求

那么它就需要买服务器，那么服务器 ms 没有市场，而且 windows 很不安全

那么十之八九会买一个 sun/ibm 的机器，或者 hp，但不管是谁的机器

它装的操作系统也不会是 windows，因为 windows 太不安全了，而且多核

的支持太差了

这个有空再说，那么如果你要写一个程序在这样的机器上跑

难道我们就在这个机器上做开发么？当然不可能，一般程序员开发用的都是 pc，windows

那么该怎么办？写一个程序，然后再拿到服务器上去编译，去调试？

肯定不可能，所以我们就希望找到一个语言，编译完生成程序之后

在 pc 上调试，然后直接移植到服务器上去，那么此时，我们就会毫不犹豫地选择 java

因为在跨平台以及安全性来说，java 永远是第一选择

ok，下面说 java 的缺点

一慢，这其实是一种误区，这就好比 goto 语句一样

java 也抛弃了指针，虽然看上去似乎变慢了，但是在这个两三年硬件性能就能翻番的年代

速度已经不是我们关心的问题了，而且对于企业级的应用来说

没有什么比安全稳定更重要的，换句话说，我们可以忍受慢，但是不能忍受死机和蓝屏

而且越大型的应用，这种慢的劣势体现得越模糊

因为当系统项目越做越大，任何一个环节做不好都可能影响全局的情况下

安全尤其重要，而且就像 goto 语句一样

这种过分追求速度的主张会给系统开发和纠错以及维护带来无可挽回甚至

不可避免的损失

把内存交给计算机去管理吧，这种代价值得

我们做的不是 pc 游戏，没必要把内存的那一点点消耗当亲爹

二难看，又是一个误区，很多人甚至拿出 java swing 控件画出的界面来说

呵呵，其实 java 不是不能画得好看，IDEA 就是 java 写的 IDE，挺漂亮的

但为什么难看呢，是因为 swing 控件它本身就是 unix 时代的产物，swing 控件贴近 unix 界面

老外看 unix 界面其实挺顺眼的，他们就是吃 unix 饭长大的

而 unix 又是吃百家饭的，不像 ms 那么唯利是图，所以不怎么对中国人友好

加上我国又没有公司在做操作系统，所以看上去是不怎么顺眼

其实玩过 unix 的人都知道，unix 对中文的支持一直不怎么好

三我还没想到，其他人补充

给初学者之二：从 JDK 说起

在知道了 java 有什么优点，能做什么之后

就该说一下 java 该如何去学了

在说 java 如何去学之前，有必要把 java 的几个大方向做一个简单说明

早在五年前，嗯，应该说是六年前，也就是 99 年的时候

sun 公司做出了一个决定，将 java 应用平台做一个划分

毕竟在不同领域，语言应用特性是有区别的

针对不同领域内的应用,sun 公司可以发布相关高端标准来统一规范代码

这三大块就是 J2SE,J2EE 以及 J2ME

这个举措今天看来无疑是非常了不起的

正是由于这次革命性的发展，使 java 从一种小打小闹游戏性的语言

发展成为今天企业级应用的基础

这里要特别说明一下 J2SE J2EE J2ME 中 2 的意思

其实 2 就是英文单词 to 的谐音，就是 to 的意思

而不是 second edition，当然 java 2 本身版本号就是 1.2，也有点 2nd edition 的味道

说点题外的，sun 公司发布的 java 版本很有意思

虽然总是写是 1.X 但其实外界对这种版的说法也就是 X.0

比如 java 2,其实就是 java 1.2

1.3 其实就是 3.0，1.4 就是 4.0，现在所说的 5.0 其实就是 1.5

只是以前我们更习惯叫 1.X 而已

可能到了 5.0 以后，就全改叫 X.0 而不是 1.X 了

所以以后听到别人说 java 5.0，千万别惊讶，其实就是 1.5

在这三个 J2*E 中 J2SE 是基础，就是 java 2 的标准版(java 2 standard edition)

也就是最基础的 java 语言部分，无论学什么 java 技术，J2SE 都是必须掌握的

要使用 J2SE 就必须安装 JDK（java development kit）

JDK 在 sun 公司的主页上可以免费下载，下载后需要安装，具体安装流程看教材

JDK 包含有五个部分：核心 API，集成 API，用户界面 API，发布技术还有 java 虚拟机（JVM）

先说运行环境，运行环境最主要要说的就是 java 虚拟机（JVM）

前面我们说过 java 是跨平台的语言，那么如何做到跨平台呢？毕竟每种操作系统都是不同的

java 的设计者们提出了一个虚拟机的概念

在操作系统之上建立一个统一的平台，这个平台必须实现某些功能以支持程序的运行

如下图：

```
-----------------------------------
|                program                |
-----------------------------------
|                 JVM                   |
-----------------------------------
```

| UNIX | Windows | Linux | Solaris |..

程序员所写的每一个程序都先运行在虚拟机上

所有操作都必须经过虚拟机才能和操作系统交互

这样做不仅统一了所有操作系统，同时也保证了操作系统的安全

要死机的话，死的是虚拟机（JVM）而操作系统并不会受此影响

而我们所说的 java 运行环境指的主要是 JVM，其他的不说了，省略

下面说说 JDK(java development kit)的 API，其实用 JDK 来包括运行环境以及开发工具

个人感觉是不恰当的，因为这三个单词仅能说明开发工具，也就是几个标准的 API

而没有让人感觉到有运行环境的意思在里面，这是题外

那么什么是 API?

简单地说就是 Application Programming Interface，应用程序编程接口

在 java 里面，就是一些已经写好了的类打成的包

这又要解释什么是类什么是包了，简单说一下，包就是类的集合

一个包包含零个或多个类，嗯，具体的可以去看书

这些类是 java 的基础类，常用的类，可以简单理解成 java 的工具集

最后说一下 JDK 的发布技术，其实按我的理解，讲白了就是编译器

将.java 文件转换成.class 文件的一种技术

这三部分组成了 JDK，有了 JDK，就可以开发出 J2SE 应用软件了

最原始的只要用一个记事本写几行代码就可以了

但一般来说我们会使用效果比较好的开发工具，也就是 IDE

在 J2SE 这一块，特别推荐 JCreator 这款 IDE

sun 公司的产品，与 JDK 结合得几乎是天衣无缝，非常适合初学者使用

教材方面中文的推荐电子工业出版社出版的《java 教程》初级与高级篇各一本

还有就是《21 天学通 java》虽然有人说 21 天系列是烂书，但个人感觉对于 j2se，这本书翻译得已经很不错了，基本没有什么语法错误，语句也很通顺

最后要说的就是《thinking in java》

这本书自然很经典，说得比较细，只是我觉得不太适合初学者，其实也不难

初学者直接看也不成问题，但个人感觉还是找本教材也就是前面推荐的两款来看比较好

基本概念的理解还是用教材的，毕竟 thinking in java 有的版本翻译得很烂

而且个人建议还是看原版比较好，当然这几本都看最好了，但如果没时间至少精读其中一本，然后再看其他两本就可以，其实三本书内容也差不多但看问题的角度方式以及面向的读者也都不同，嗯，怎么说呢，找适合自己的吧

最后要说的是

由于虚拟机的存在，J2SE 的大多数软件的使用会比一般桌面软件慢一些
效果不尽如人意，现在大概只有 swing 控件还在使用吧，其它没怎么听说

J2EE&J2ME

这是 java 应用的重中之重，如果想拿高薪，最好把 J2EE 学好
记得以前在 csdn 上看过一个调查，月薪上万的程序员主要从事哪方面的工作
十个中有八个是做 J2EE 的，其他两个一个做 J2ME，还有一个做嵌入式
也许有些夸张，但也从某一方面说明 J2EE 人才的稀缺以及应用的广泛
所以如果想学 java，只精通 j2se 是永远不够的，至少还需要时间去了解其它两个 J2*E

给初学者之三：java 企业级应用之硬件篇

总算讲到企业级应用了，内容开始逐渐有趣起来

java 企业级应用分为硬件篇和软件篇
重点在软件，硬件是外延，严格地说与 java 没有必然联系
但是，由于 java 是网络语言，不了解硬件的一些基础知识
软件知道再多也没什么用，不要上了战场还不知道为什么而打仗
硬件是软件的基础，在这个前提下，有必要专门花一点点篇幅来聊一下硬件

硬件，简单地说就是我们实际生活中看得见摸得着的东西

也就是那些冰冷的机器，比如服务器，个人电脑还有网络交换机，路由器等等

那么先抛开网络设备不谈，先来说说计算机电脑的历史

在很早很早以前，人类创造了第一台电脑，那时候的电脑只是一台用来计算的机器

无比大，无比重，无比傻，除了算其它屁事不会做，没有所谓的人工智能与计算机网络

但是总算是诞生了，虽然以今天的眼光去看那时候的机器巨傻无比

只配叫做计算器而不是电脑，没有逻辑思维能力，只会死算

但千里之行，始于足下，反正是造出来了

然后随着时间的推移，制造业发展发展发展

电脑性能逐渐得到提升，速度快了起来，成本也逐渐低了下来

于是人们造出了第二台，第三台，第四台，第五台.....第 n 台计算机

人们就造出了无数台计算机并使其成为一种产品

逐渐应用于许多复杂计算领域，不仅仅是科研，许多生产领域也开始出现计算机的影子

然后又随着时间的推移，人们发现不可能把所有的计算机都做成一个样子

因为各行各业对计算机性能的要求各不相同

于是开始把计算机划分档次，最简单地是按照计算机的大小划分

就是教科书上写的大型机，中型机，小型机

//个人感觉这样分纯粹扯淡，还以为是小孩子玩球，分为大球，中球和小球

但是不管怎样，计算机不再是千篇一律一个样子了

按照性能的不同，在不同领域，出现了满足符合不同要求的计算机

几乎在同时，人们也开始考虑计算机之间通讯问题

人们开始考虑将不同的计算机连接起来，于是网线出现了，网络出现了

又随着网络的发展，出现了一下专门为了寻址而存在的机器

这就是路由器和交换机，然后又出现了一些公益性的组织或团体

他们制定了一系列的标准来规范以及管理我们的网络

于是 3w 出现了，计算机的网络时代来临了

嗯，说到这里，计算机发展到今天的历史大概说完了

我们来详细说说网络时代的计算机以及各个硬件供应商之间的关系

前面说到了，计算机分为大型机，中型机和小型机.....

但是现在市场上没有人这样分，要是出去买机器，对硬件供应商说

我要买一款中型机，或者说，我要买一款小型机，硬件供应商肯定会问问题

他们会问你买机器干什么用的？科学计算啊还是居家用，是作服务器啊还是图形设计

但不管怎样，简单地说大中小型机已经没有什么意义了

我们按照使用范畴来划分

简单划分为

服务器，工作站还有微机

服务器（server）

服务器涵盖了几乎所有的大型机以及大部分中型机甚至一些小型机

用通俗点话说　　衿骶褪悄掣龋　?4 小时不间断运行提供服务的机器

比如卖飞机票（中航信），比如酒店预定（携程）

比如提供门户站点相关服务（sina），比如电子商务（ebay，amazon，阿里巴巴）

这些服务对机器都有一些特定的要求，尤其强调安全和稳定

工作站（workstation）

工作站其实是图形工作站的简称，说白了，就是某种功能极其强大的计算机

用于特定领域，比如工程设计，动画制作，科学研究等

个人电脑/微机（pc）

计算机网络的最末端，这个应该不用我说太多了

网络时代的 pc 已经普及到千家万户

说完了分类，我们就来说说各个硬件供应商

首先是服务器还有工作站

这两类硬件供应商主要是以下三家

Sun,IBM 还有 HP(惠普)

然后是 PC

以前 IBM 还有 PC 事业部，现在被联想吞并了（蛇吞象）

现在国际市场上有联想和 DELL(戴尔)，目前戴尔还是国际老大

还有 HP 康柏

然后是网络，也就是路由器和交换机

这块市场嘛，Cisco(思科)Brocade(博科)还有 McDATA 三足鼎立

内核(CPU)

PC 内核

主要是 AMD 和 Intel，前者最近与 Sun 公司合作，Sun 也有一部分单双核

服务器用的是 AMD 的

服务器与工作站内核

这一块与硬件厂商绑定

还是 Sun,IBM,HP 三家自己生产

题外

在一些大型主机应用市场，比如卖飞机票

德国的汉莎，中国的中航信，香港的国泰用的都是尤利（美国的公司，英文名我忘了）

其它用的是 IBM 的机器，现在能做大型机的感觉似乎只有 IBM 可以

尤利已经快倒了，技术太落后了，现在他们的系统还是 fortran 写的，连 c 都不支持

要特别说明的是，一个超大型主机然后多个小终端/pc 的结构现在越来越没市场了

将来的趋势是用一整个包含多个服务器的分布式操作系统来取代这些大型主机

因为大型主机更新换代极其困难，一旦数据量超过了主机的处理能力

那么就要换主机，这个成本是极大的，但是如果用分布式操作系统

那就只需要增加小服务器就行了

硬件就大概说到这里，与大多数人没什么关系

因为大多数人压根不可能进入这些硬件领域，除非做销售

说了这么多，只是为了给软件部分打基础而已

//做嵌入式的除外

给初学者之四：java 企业级应用之软件篇

嗯，说过了硬件就该是软件了

这篇是这个系列的重中之重

首先我们来说说什么是软件，统一一下概念

所谓软件通俗地说就是一套计算机程序

实现了某些功能的计算机程序

在很早很早以前，一台计算机的软件是不分层次结构的

一台计算机只有一个系统，这个系统既是操作系统又是应用软件，与硬件紧密绑定

后来经过许多年的发展发展发展

人们把一些与硬件紧密相连的又经常用到必不可少的功能做到一套程序中去

这一套程序就被人们称做操作系统

另外一些可有可无的，不同工作适应不同环境的功能封装到另外一套程序中去

而这一系列程序被人们称作应用软件

如下图：

|应用软件：falshgat/IE/realplayer/winamp..|

|操作系统：UNIX/Windows/Linux/Solaris... |

前一篇我们知道，硬件分为服务器工作站与 pc

其实无论哪种硬件的软件，都有操作系统与应用软件

ok，那下面我们来谈应用软件

在现在企业级应用中，我们的应用软件一般分为三层

三层分别是表示层，业务逻辑层，数据持久层

|表示层|业务逻辑层|数据持久层|

我们来说说三层中的代表软件

表示层

这一层一般在客户端 pc 机上，最常见的是 IE 浏览器，这就是表示层的软件

表示层是直接与使用者交互的软件

业务逻辑层

这一层一般在服务器端，顾名思义，所有业务逻辑处理都在这一层完成

最典型的是 appserver，比如 IBM 的 websphere，BEA 的 weblogic 还有 tomcat/jboss 等

这一层也是三层中的重点，我们要说的大部分内容都是关于这一层的，这

个等会再说

这一层就叫做中间层

数据持久层

这一层典型的就是数据库，一般也在服务器端

但该服务器一般与装业务逻辑层软件的服务器分开

当然你也可以用 IO 输入输出流往硬盘上写东西

但没人会建议你这么做，因为这样做你的数据缺乏管理，不管怎样

这一层要做的就是保存数据，业务逻辑层软件一般不负责保留数据

或者说业务逻辑层只负责暂时储存数据，一关机，业务逻辑层数据全部 over 了

那么数据的持久化（也就是储存数据）就必须要在这一层完成

下面放着这些概念不谈，我们来说说将来的趋势

趋势一：

瘦客户端，很早很早以前，当时 C/S 模式也就是 client/server

客户端软件大行其道的年代，一个 pc 用户，是采用一个傻终端连接到服务器上

然后进行相应的操作，最典型的就是我们上 bbs 经常用的 c-term

这就是那个时代的产物，同样还有我国现行的机票定座用的 e-term

后来呢，浏览器变得非常流行，人们发现，浏览器也能传递一些数据

虽然这些数据并不像那些终端那样准确，但应付大多数日常需求足够了

于是人们就提出一个瘦客户端概念，也就是说，将来表示层所有的其他软

件疾挥?

我们唯一需要的就是一个网页浏览器，然后通过浏览器输入 ip 地址连接到服务器

然后进行相关的操作，由于网页浏览器一般每个操作系统都有自带一个

这样做就达到了给我们客户端瘦身的目的（不需要安装额外软件）

这样模式被称作 B/S 模式，也就是 browser/server 模式

但需要指出的是，虽然瘦客户端是趋势，但并不代表胖客户端没有市场

尤其是一些复杂的业务操作，还是浏览器这种简单软件无法胜任的

趋势二：

傻数据库，ok，首先，我承认，这个名词是我发明的，但我实在无法找到一个更好的表达

什么是傻数据库，如果谁对数据库有所了解的话，就知道，以前的数据库有自己的一套管理体系，甚至有自己的客户端，比如 oracle,mysql,sqlserver 都有

在某个管理工具上写什么 sql 语句查询数据库是我们以前常做的事

那么将来我们提倡的是：将所有的业务逻辑封装到业务逻辑层去

管理的事情由软件来做，由业务逻辑层的软件来做

所谓傻数据库就是说，将来的数据库什么事都不用做

只用把数据给我保存好就行了，那些复杂的业务逻辑什么外键什么关联

都没数据库什么事了，都交给业务逻辑层软件来做

这样做的好处就是：我们就不需要这些该死难懂又复杂的数据库系列管理

工具了

而且这些工具每个数据库都有自己的工具，完全不一样，乱七八糟，没有人喜欢面对他们

除了数据库维护人员，也就是 DBA，我们是软件工程师，维护的事让他们去做

而且严禁数据库维护人员改动数据库的数据，他们只做备份，必要时候恢复一下就是了

了解了这两个趋势之后，是不是有种砍头去尾保中间的感觉？

没错，未来的趋势就是中间件时代，中间件工程师将是未来计算机应用的主流

那再次统一一下概念，什么是中间件？

记得我上学的时候，看 ibm 的教材，看了半天中间件定义，就看懂记住一句话

中间件是做别人不愿意去做的事情，现在想想，狗屁定义，呵呵

什么是中间件，中间件是业务逻辑层的应用软件

是处理业务数据与客户端之间业务逻辑的一种应用软件

一种提供网络服务的服务器端应用软件

举个非常简单的例子，网上银行，某个人想用 IE 进入工行的账户，然后转帐

在这个例子中，客户端表示层显然是 IE，数据持久层显然是银行的核心数据库

那么中间件是什么？中间件就是提供这种服务的系统

这三层的划分如下

|表示层 | 业务逻辑层 | 数据持久层 |

| IE | 网上银行 | 数据库 |

给初学者之五：企业级应用之中间件

前面一篇简单介绍了一下应用软件的分层

下面重点介绍一下中间件，也就是业务逻辑层的软件结构

从本系列第二篇我们知道，java 程序是跑在虚拟机之上的

大致结构如下：

| grogram |

| 虚拟机 |

| 操作系统 |

也就是说操作系统先运行一个 java 虚拟机，然后再在虚拟机之上运行 java 程序

这样做的好处前面也说过了，就是安全，一旦出现病毒或是其他什么东西挂掉的是虚拟机，操作系统并不会受多大影响

这时候有人可能会问，为什么非要虚拟机？把操作系统当成虚拟机为什么不行？

可以，当然可以，但是这样做某一个应用软件的 bug 就可能造成整个操作系统的死亡

比如说我们在某个服务器上安装了一个收发电子邮件的软件和 java 虚拟机

那么一旦黑客通过收发电子邮件的软件入侵系统，那么操作系统就整个玩完

那么如果黑客通过 java 程序进行攻击的话，那么死的将会是虚拟机而不是操作系统

大不了虚拟机崩溃，而操作系统正常运行不受任何影响

举个简单例子，比如说最常见的是将数据库(DB)与中间件放在同一台服务器上

| program | |

----------| DB |

| 虚拟机 | |

| 操作系统 |

那么此时如果没有虚拟机，黑客病毒攻击中间件系统，就有可能造成操作系统的死亡

那此时数据库也有可能跟着一起玩完，那损失可就大咯

那如果此时有虚拟机，那么一旦被攻击，死的是虚拟机，操作系统与数据库不受任何影响

嗯，回顾完虚拟机，再来介绍中间件

在很早很早以前，任何一家企业，想要搭建一个局域网系统，他需要请许多个工程师

比如说我们想搭建一个网上银行，客户端用浏览器,后台数据库比如说用 oracle

那么搭建这样一个网上银行，可能需要用到多少个工程师，我们来算一算

首先，由于客户端用的是浏览器，我们需要一些了解网络通讯协议以及一些浏览器标准的网络工程师

其次，由于后台数据库用的是 oracle，那我们还需要请 oracle 的工程师，

因为数据库这一层每个数据库公司的接口什么都不一样

然后，我们还需要一些操作系统的工程师，因为我们的系统需要跟操作系统直接交互

最后，我们需要一些设计网上银行系统及其相关业务的工程师

太多了太多了，这样一个中间件队伍实在太庞大了，制作维护成本实在太高了

不仅如此，这样一个中间件就算做出来，他们所写的代码也只能满足这一家公司使用

其它公司统统不能再用，代码重用率极低，近乎不可能重用

毕竟这个系统中改动任何一个部分都有可能涉及到整个系统的改动

那么如何降低成本？

我举出了四组的工程师：

网络工程师，数据库工程师，操作系统工程师以及设计网上银行系统的业务工程师

除了最后一组设计网上银行的业务工程师之外，前面三组工程师是不是每一个项目都需要的？

就算不是每一个项目都需要，至少也是绝大多数项目需要的吧？

哪个项目能够脱离网络，数据库和操作系统？不可能，在这个时代已经很少很少了

好，那既然每个项目都需要，我们是不是可以用一个产品来取代这三组的工程师呢？

我们的业务工程师只需要遵循这个产品所提供的接口，进行相应的开发就行了

人们提出了一种叫做 appserver 也就是应用服务器的东西

应用服务器是干什么的？按官方的说法，应用服务器是包括有多个容器的软件服务器

那容器是什么？容器(Container)到底是个什么东西我想多数人还是不清楚

在说这个之前，先介绍一下组件

什么是组件，组件是什么？组件其实就是一个应用程序块

但是它们不是完整的应用程序，不能单独运行

就有如一辆汽车，车门是一个组件，车灯也是一个组件

但是光有车灯车门没有用，它们不能跑上公路

在 java 中这些组件就叫做 javabean，有点像微软以前的 com 组件

要特别说明的是，由于任何一个 java 文件编译以后都是以类的形式存在

所以 javabean 肯定也是一个类，这是毫无疑问的

好，那么容器里装载的是什么呢？就是这些组件

而容器之外的程序需要和这些组件交互必须通过容器

举个例子，IE 发送了一个请求给容器，容器通过调用其中的一个组件进行

相关处理之后

将结果反馈给 IE，这种与客户端软件交互的组件就叫做 servlet

但是组件有很多种，那么如何区分这些组件呢？

有多种管理办法，比如同是同样是 servlet，有些是通过 jsp 生成的

而有些是开发人员自己写的，那么通过 jsp 生成的 servlet 集中放在一个地方

而开发人员自己写的则需要在 xml 里面配置一些基本的参数

同时，不同组件有可能还需要继承一些特定的父类或者接口，这也是容器管理的需要

还有其他的一些组件，这里就不一一说明举例了

那么容器有很多种，按照他们装载的组件类型划分

比如有装 ejb 的 ejb 容器，有装 servlet 与 jsp 还有静态页面的 web 容器等等

//这种只含有 web 容器的应用服务器也被叫做 web 服务器

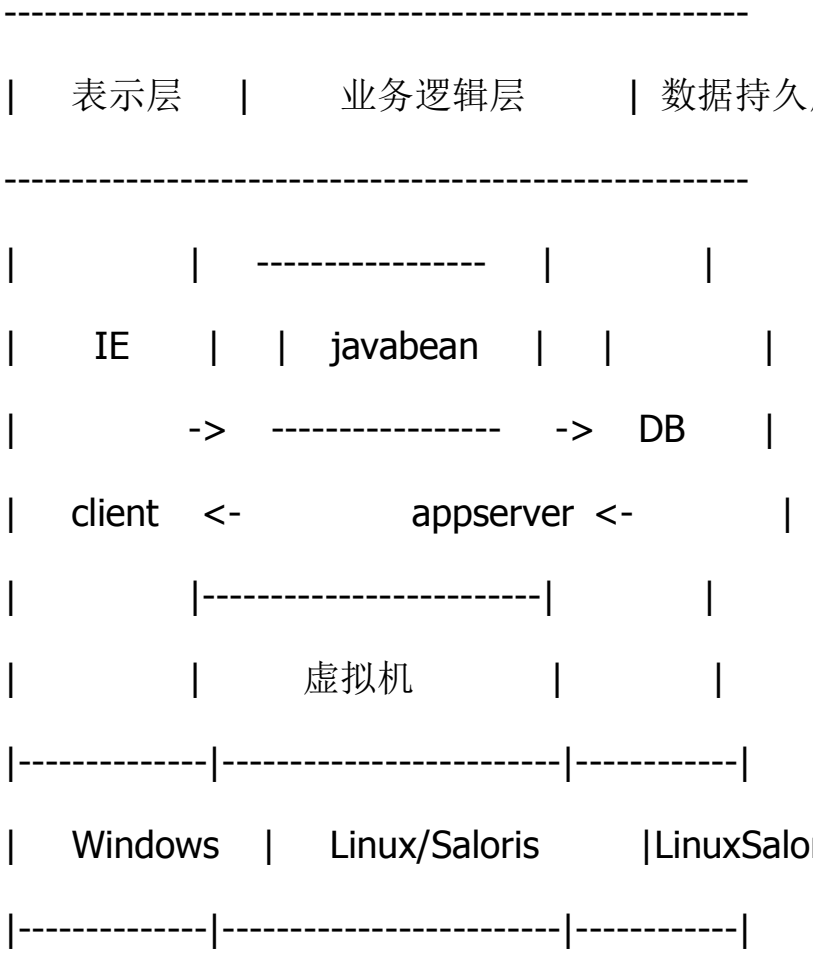
当表示层的应用软件通过网络向 appserver 发送一个请求的时候

appserver 自动找到相应容器中的组件，执行组件中的程序块，把得到结果返还给客户

而我们要做的事就是写组件也就是 javabean，然后放到 appserver 里面去就可以了

至于怎样与 IE 通讯，怎样截获网络上的请求，怎样控制对象的数量等等这些繁琐而无味的工作我们都不管，都由 appserver 去做吧，把注意力集中在业务逻辑上

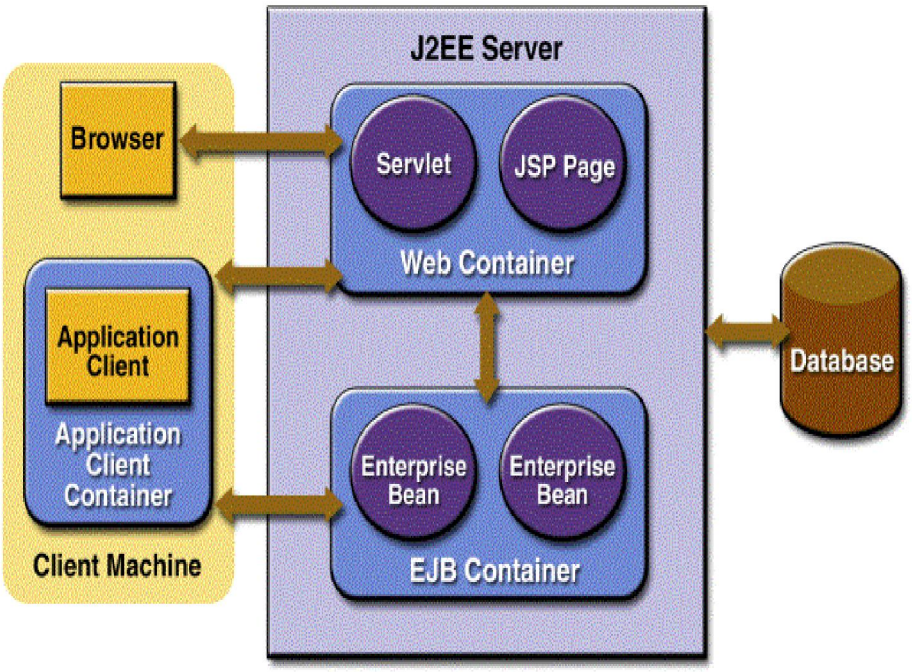
appserver 与其他相关软件的关系如下图：



图上可以看出：虚拟机负责处理中间件与操作系统之间的交互

appserver 则负责组件的管理以及与其他两层的业务交互

1 附图: image002.gif (76463 字节)



要说明的是上图中还包含有应用程序客户端容器(Application client contai ner)

管理应用程序客户端组件的运行，应用程序客户端和它的容器运行在客户机

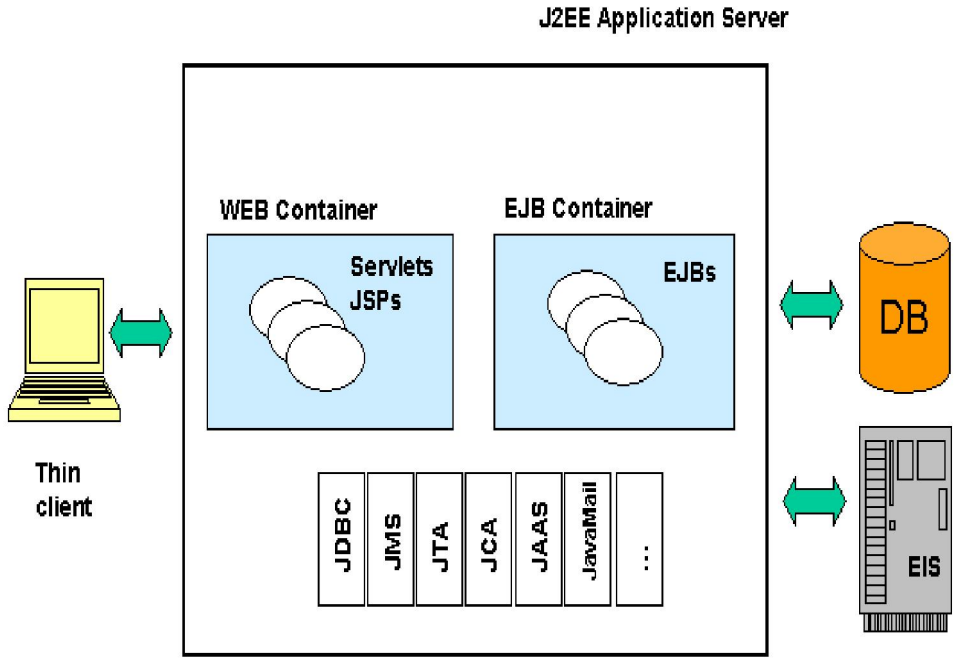
这种情况比较复杂一般说的是两个 server 之间的通讯

比如 jsp/servlet 容器在一个服务器上，而 ejb 容器在另外一个服务器上等等

这是分布式操作系统大面积应用的基础，这个以后再说

下面这张相对简单：

2 附图: j2ee.gif (8226 字节)



嗯，那么话题再回到中间件上去，什么是中间件？

appserver 就是所谓的中间件，但是中间件不仅有 appserver，还有其他的东西

换句话说，appserver 只是中间件的一种

而关于中间件有诸多规范以及遵循这些规范的模型

最流行的规范无非两种，一个是 j2ee 还有一个是.net

但是.net 几乎只有微软在用，所以很多人把.net 这个规范就当成是微软的中间件产品

也不为过，毕竟没几个公司喜欢跟着微软屁股后面跑的

给初学者之六：**java** 企业级应用之综合篇

我们知道中间件有很多种规范以及相关的模型

最流行的一个是 j2ee 还有一个是.net

那么各大公司关于这两套规范各有什么产品以及周边呢？

j2ee:

黄金组合

操作系统：Solaris

应用服务器：Weblogic

数据库：Oracle

开发工具：JBuilider/IntelliJ IDEA

优点：性能一级棒，大企业大公司做系统的首选，世界五百强几乎都是这套组合

缺点：极贵

超级组合，也是最安全最酷的黄金组合，硬件采用 SUN 公司的机器

但是 SUN 的服务器很贵，同等价格不如去买 IBM 的机器

SUN 的服务器支持 Solaris 的效果自然不用说，Solaris 号称是世界上最安全的操作系统

Oracle 也是世界上最安全，性能最优的数据库，Weblogic 是当今性能最优

的 appserver

JBuilder 和 IDEA 各有所长，JBuilder 是 Borland 公司的招牌之一

是当今世界上最流行的 java IDE，用 delphi 写的，但网络上评价似乎不是很好

IDEA 拥有插件功能，界面在所有 java IDE 中最为漂亮，东欧人开发的产品

东欧人严谨的作风在这个产品上体现得尤为突出，用 java 写的

IDEA 甚至号称自己被业界公认为是最好的 IDE//个人保留意见，没有最好只有更好

但我用 JBuilder 的时候发现了不少 bug，而至今还没有在 IDEA 上发现什么 bug

个人推荐 IDEA

价格方面，Solaris 开源，但是 SUN 的服务器比较贵，Weblogic 最高是 34 万

oracle 标准版要 18.6 万，企业版要 49 万，JBuilder 要 2.7 万左右

IDEA 零售价大概是 500 美金，也就是 5000 多元

另外，虽然理论上这些产品的综合性能要高于其他选择，但是必须看到

由于产商之间的利益冲突，比如 oracle 也有自己的 appserver，但是性能不怎样

使得这几种产品之间协作的性能要比预想中的要差一点点

--

开源系列

操作系统：-

应用服务器：JBoss

数据库：MySql

开发工具：Netbeans

优点：便宜，性能未必最佳，但是对付中小企业足够了

缺点：出了问题自己抗吧

嗯，这是 java 阵营最大的特色，免费免费，还有在开发工具这一栏 Eclipse 也是免费的

但后面要说，算了，换个有代表性的开源产品来

tomcat 仅有 web 容器而没有 ejb 容器，而 jboss 已经集成了 tomcat

也就是说下载了 jboss 之后，启动的同时也就启动了 tomcat

jboss 在 tomcat 基础之上多加了一个 ejb 容器，使得 jboss+tomcat 成为和 weblogic

websphere 之外又一个得到广泛应用的 appserver

现在大概是这样，中小型企业多用 jboss，如果应用小一点就用 tomcat

只有给那些大型企业做的项目，才会花钱去上一个 weblogic 或者 websphere

mysql 也是开源的数据库，做得非常不错，如果系统对数据库要求不高

或者安全要求不是非常严格，mysql 是一个非常不错的选择

开发工具方面，netbeans 是 sun 公司极力推广的一种 IDE

听说在北美市场使用量已经超过 eclipse 了

操作系统，软件再不用钱，服务器也要钱，看这台机器上跑什么操作系统就用什么了

--

IBM 套餐

操作系统：Linux

应用服务器：Websphere

数据库：DB2

开发工具：Eclipse/WebSphere Studio

优点：服务好，IBM 可以提供全套服务，也可以替客户承担风险

缺点：把机器数据全部交给 IBM，安全什么的都由不得你了

呵呵，IBM 全套产品，甚至包括硬件设备 IBM 的服务器

由于是一个公司的产品，各产品之间的协作自然不错

价格方面，Linux，DB2，Eclipse 都是开源产品，Websphere 目前零售价是 33.8 万人民币

IBM 服务器不错，可以考虑

--

.net：

微软阵营

操作系统：Windows

应用服务器：.net 应用服务器(好像叫 IIS)

数据库：SqlServer

开发工具：MS Visual Studio

优点：客户端的用户体验良好，和客户端诸多微软产品的兼容性强

缺点：离开了微软，寸步难行，和其他任何一家公司的产品都不兼容

微软的东西，怎么说呢，太专横了

微软所有的东西都是围绕着 windows 来做的

.net 其实已经可以实现跨平台了，但是微软出于自身商业考虑

在其应用服务器跨平台的实现上设置了种种障碍

而且针对 windows，微软做了大量的优化，可以这么看

.net 就是与 windows 捆绑的一套产品

所以有些人说，微软的产品离开了 windows，就是渣

而且.net 开源选择也少，安全性方面考虑，windows 本身就有一堆补丁要打了

sqlserver 也不安全，至于.net 到底安全不安全我不清楚，毕竟我没怎么用过

但整体考虑，感觉.net 不是大企业的首选，鉴于其浓厚的商业背景

也不是中小企业的首选，但是必须看到

客户端也就是微机 pc 市场已经完全被 windows 所垄断

所以在一些快速开发，还有和微软产品兼容性要求较高的领域，.net 还是比较有市场的

最后一个 visual studio 对它之前的版本兼容，且支持 c,c++,c#,vb 等语言

在其传统领域，比如写一些桌面软件等客户端应用上，.net 还是第一选择

--

最后要说明的是

这些组合不是固定不变的

由于 J2EE 得到了绝大多数 IT 企业的支持以及 JAVA 跨平台的特性

我们可以自由地定制个性化的组合

比如我可以选择 **windows+jboss+eclipse+oracle**

也可以选择 **solaris+websphere+IDEA+mysql**

等等，这些自由组合都是可以的，但是有一点必须说明

微软的东西，一般来说离开了 windows 就不能用

比如你选择了.net 应用服务器，那操作系统就必须是 windows

你选择了 sqlserver，那就必须在 windows 上用

还有就是遵循 j2ee 规范的所有的组件都可以在不同的应用服务器上互相移植

比如你可以在测试的时候用 jboss

而在正式投产上线的时候使用 websphere，只需要在配置文件中作相应改动即可

给初学者之七：java 企业级应用之术语篇

在了解完 J2ee 的相关周边产品之后需要深入 J2ee 规范内部去了解一下到底这些规范

这里介绍几个最常用的规范

再继续说下去之前有必要说几个常识

Java 的诞生

Java 之父 James Gosling 早年从 cmu 毕业之后

从事了一段时间的开发工作，后来意外碰到一个项目

这个项目要求他用 C++开发，但可爱的 JG 是天才，凡是天才在某方面特别突出的同时

必然有一些天生的缺陷，恩，或说共性，比如说懒，急躁和傲慢

JG 既然是天才，那就必然具备这些共性，JG 懒，以至于他学不好 C++

不仅他学不好，当年开发出 Java 的那个团队也都学不好 C++

他们急噪，以至于他们中有人甚至威胁以辞职的方式离开这个需要使用 CPP 开发的项目

他们傲慢，所以他们决定开发出一种新的语言来取代那个该死的 CPP

更可爱的是，他们一开始居然给这门语言起名 C++++--//没错，我没敲错

叫什么 C 加加 加加减减，意思是加上一些好东西，减去一些坏东西

天才的设定，有时候你会发现天才和傻瓜真的只有一线之隔

还好这个可爱的名字没有被继承下来，这些天才们给他们的产物起名叫 Oak//橡树

只是后来当他们去注册这个名字的时候，发现这个名字已经被注册了

于是在 Sun 公司的一个女职员//mm 就是心细，这个说法也是我们公司 mm 告诉我的

的提议下，把这个可爱的语言起名为 Java，就是他们当时喝的咖啡的名字

所以我们看到 Java 的标志就是一杯冒着热气的咖啡

JavaBean 了解完 Java 之后，再来说说什么是 JavaBean//华为面试题

JavaBean 是什么？ 咖啡豆

ja，更为科学点的解释是

用 java 语言编写的可重用的软件组件//组件的定义前面说过了，不再重复

很形象不是么？ 将 javabean 放入杯子//容器，还记得容器的概念么？web 容器，ejb 容器

就可以冲泡//编译 成咖啡，供客人们品尝//运行

完美的服务

下面进入正题 再谈容器

前面介绍过容器，我觉得有必要再补充一点

容器从某种意义上说其实就是一个可运行的 java 写的应用程序

犹如 c++/c 编译后生成的.exe 文件

不同的是 java 编译后的文件需要用命令行或者脚本启动执行

由于容器是由 java 写的，所以容器都能够跨平台

虽说如此，似乎大部分容器都针对不同的操作系统提供了不同的版本

但可以肯定的一点是，相同容器间的移植组件不需要重新编译

Servlet web 容器组件

Servlet 确切地说，就是 web 容器运行的 java 组件

与普通 javabean 不同的是，Servlet 定义了一系列方法//比如 init()和 destroy()

供容器调用，调用的主要目的是为了管理

当一个 request 请求被 web 容器截获之后，容器分析该请求地址

然后通过一个配置文件中的映射表//web.xml

调用相应的 Servlet 组件处理后将结果返还给客户端

JSP//Java Server Page

web 容器组件

Servlet 出现了之后，人们发现几乎没有办法用一个非常直观的方式去编写页面

毕竟页面是 html 语言编写的

而让我们用一种流程式的处理方式去逐行教计算机如何写 html 代码太困难

在这种情况下 JSP 应运而生，JSP 将 java 代码嵌入 html 代码内部

然后存成.jsp 文件，再由计算机编译生成 Servlet 储存起来//注意这个过程

所以 JSP 和 Servlet 对于 web 容器来说其实是一种东西，虽然它们编写遵循的标准有所不同

极大地简化了代码同时增加了代码的可读性，生产维护成本下降

值得一提的是，在制定 JSP 规范的过程中，借鉴了 ASP 的很多规范

写过 ASP 并熟悉 Java 语言的人应该能很快掌握 JSP

EJB//Enterprise JavaBean

ejb 容器组件

随着时间的推移，人们发现普通的 JavaBean 似乎并不能满足企业级应用的需要

最典型的就是虚拟机提供的垃圾回收收集机制也就是 GC 不够完善

可以优化的余地极大，在这种情况下，EJB 应运而生

EJB 和其它组件一样，不过遵循了某些规范而已

但是这些规范更多的是为充分利用机器并提高性能为主要目的的

举个简单例子

比如某个 web 服务器有 100 个用户同时连接上

由于网络连接是瞬时连接，所以很多时候并发数并没有 100 那么大

前一秒有可能有 30 个请求被发送过来并被处理

后一秒可以只有 10 个请求被发送过来并被处理

只有在非常非常极端的情况下才有可能发生 100 个请求同时被发送过来并被处理的情况

那么我们是否需要保留 100 个那么多个对象在服务器的内存里面去处理这些请求呢？

很显然，不需要，大多数时候//甚至可以说是所有时候，我不相信有那么极端的情况

我们只需要保存其中的 10-30%就够了，那么什么时候需要 20%，什么时候需要 50%

甚至 100%，这个过程就交给容器去管理，这就是 ejb 容器每天在干的事

管理内存中活跃的对象

恩，必须强调的一点是，由于使用的不成熟

我们经常把规范以及具体的应用两个名词混用

举个简单例子，我们说 Servlet，极有可能说的是 Servlet 规范

也有可能说的是一个具体的 Servlet，这个就要看情况而定了

EJB，JSP 也是如此

JDBC

和数据库的连接

这个严格说来是数据库产商需要关心的事

关于 AppServer 如何与数据库的连接

但是也需要开发人员做一点事，因为 AppServer 不知道什么时候组件需要用到数据库

同时也需要开发人员告诉 AppServer 他们使用的是什么数据库，ip 地址等等

JDBC 就是关于这一套东东的规范

包括数据库的产商应提供什么样的接口

AppServer 应用服务器应该如何去连接

开发人员应该如何去配置这些连接等等

还有一些数据源，连接池等概念参考相关数据在此就不再赘述

其它的规范比如 JMX 等确切地说与开发人员关联并不大了

这类高级应用只对 AppServer 应用服务器产商重要

也不再罗嗦了

记得听说过这样一种说法

大一时候不知道自己不知道 大二时候知道自己不知道 大三时候不知道自己知道 大四时候知道自己知道 为什么呢，因为大一时候刚进大学，什么都不懂，很正常，大家都一样

大二或者大三时候开始接触知识，虽然还是不懂，但慢慢地开始学习，开始积累

过了一段时间，知道自己知道了//也就是前一种说法的大四，后一种说法的大三

开始屁癫，开始拽得不得了，觉得自己怀才不遇，千里马难寻伯乐的那种感觉

有些人是大四毕业了以后开始拽，悟性高一点的，大三就开始拽，因人而异

这几乎是每一个初学者经过一段时间学习后的必然阶段

不管如何，总之开始入门了，这也不是坏事

但最后每个人都会知道自己不知道的，也就是后一种说法的大四阶段

//前一种说法里面的那些家伙估计要到工作以后才能明白

因为任何一门学科都博大精深，要是能在两三年之内就统统搞懂

那不是在吹牛就是坐井观天，java 如此，c 如此，c++也是如此

那么到了本系列的第七集，可爱的读者应该处在什么阶段呢？

恭喜，在看完这篇文章之后，你就基本处于知道自己不知道的那种阶段

离拽起来还有那么一段距离，因为你们毕竟还没有学习和积累一定的基础知识

但是骗骗外行，蒙蒙国企那些吃闲饭的管理人员问题不大

给初学者之八：java 高级应用之框架篇

没错，我没敲错

之所以不再声称是企业级应用而称之为高级应用 是因为下面要讲的东西属于纯民间性质

是 java 具体应用的上层建筑，可用可不用，没有人强迫你用

首先给框架//framework 下一个定义

我想读者你可能听说过.net framework 这个概念

没错，我们将要说的 framework 也和这个 framework 差不多

所不同的是.net framework 的竞争对象是 j2ee 那一系列标准

而我们将要说到的几个框架则应用在 j2ee 的不同层面

单就单个框架而言，没有.net framework 管得那么多

但是却要比它精专多了，而且总量加起来，也远比微软那一套框架要广泛得多

回到正题，框架是什么？

软件工程之所以被叫做软件工程就是因为有那么一批人觉得可以用工程学

里面

那些管理 Project 的方法来管理软件从开发到维护这一系列流程

那么在建筑工程里面框架是什么？

现在建筑多采用钢筋混凝土结构，注意里面一个很重要的词汇：钢筋

托福阅读中曾有一题听力就是关于钢筋结构的诞生，在美国

恩，现代建筑中多在建筑起来之前，先用钢筋搭建出一个框架出来

然后往钢筋中间填入混凝土，从而形成一个完成的建筑

而今天要说到的框架就是这么一个东西在每一个软件中间的实现

框架就是那么一个通过预先写好代码从而帮我们建立起一个软件结构的这么一个东西

这里提一下框架与规范//主要指 J2ee 规范也就是官方标准的区别

从某种意义上说，J2ee 规范本身就是一个框架

无论是 web 容器也好，还是 ejb 容器也好，它们都开发了一部分通用的代码

并且帮助我们搭建起来了一个软件结构，我们要做的就是往里面填入组件

比如 ejb/servlet/jsp 等等

没错，要这么理解也没错，但是为了避免混乱，我们还是严格区分开来

本文中将要提到的框架如无特别说明，就是指的是非官方标准的框架

规范是规范，而框架是建立在规范之上的一种东西

可以说是标准的延续，或者说是民间的尝试，总之是这么一个非官方的东西

说到这里顺便提一下 JCP 组织也就是 Java Community Process/Java 社区

当初 Sun 公司在 java 发布之初，为了提倡开源和共项

同时也出于一个提出合理的标准的目的，而让广大的开发者参与标准的制定

而成立了这样一个社区，现在还健在，网址是 jcp.org

每一个新的规范发布之前都会在这个社区广泛讨论，最终对规范的制定产生巨大的影响

其中就包括企业级的参与者，相当有名的 JBoss 以及我国的金碟公司都是其中的成员

下面介绍一下几个相当著名的框架，必须要指出的是，虽然框架大多开源但并不代表所有的框架都开源，比如.net framework，但是 java 框架大多数开源

言归正传

Struts

表示层框架，名字来源于飞机的金属框架

可能有读者会提问了

表示层不是客户端么？

没错，但是语言这东西，众口烁金，别人都这么说你就不好不这么说了

最早表示层说的是客户端，后来随着时间的发展

人们也把服务器端直接与客户端//比如 IE

打交道的那部分也称为表示层//JSP+Servlet

那么表示层框架是干什么的呢？

早先大规模应用 JSP 的时候，人们发现，JSP 里面充斥着逻辑代码与数据

可读性极差，于是人们借用很早很早以前的 MVC 模式的思想

把表示层组件分为 V-Viewer，也就是 JSP

M-Model 模型，一般来说是一个 JavaBean

C-Controller 控制器，一般来说是一个 Servlet

所有人通过 JSP 和服务器打交道，发送请求，Viewer 把这个请求转发给 Co
ntroller

Controller 通过调用一个 Model 来处理该请求，然后返回数据到 Viewer

这么一个过程，从而达到数据与逻辑的剥离，增强代码可读性，降低维护
成本

而帮助人们实现这一系列东西的就是 Struts 框架，就是这么一个东西

Struts 的竞争对手主要是产商们极力倡导的 JSF 也就是 Java Server Faces

但是由于 Struts 出道时间早，所以应用比较多

JSF 则是产商们大力支持，前景看好

对于这一层来说，在 JSP 的 html 代码中出现的 java 语句越少越好

因为 java 代码越少说明页面处理的业务逻辑越少，也越合理

这也是 Struts 最初的目的，记住这话

Spring 大名鼎鼎的 Spring 框架

有人曾说 2005 年一片叫春之声，指的就是该框架

Spring 起源于 Rod Johnson 的《Expert One-on-One J2EE Design and D

evelopment》一书

Rod Johnson 认为，J2ee 里面的那一套//尤其是 ejb

太重了，对于单机的系统来说，没有必要使用那么复杂的东西

于是就开始设计并引导 Spring 小组开发出这样一个构架

不能不说他是个天才，因为的的确确不是所有的系统都是跨多服务器的

没有必要把一个简单的系统设计得那么复杂//天才的那几个共性又体现出来了

Spring 从诞生之日起就是针对 EJB 的，力争在不少应用上取代 EJB

而它也确实达到了这个目的

现在包括 WebLogic 等主流应用服务器还有主流 IDE 都开始逐渐接受该框架

并提供相应支持

提到 Spring 就不能不说控制反转 Ioc//Inversion of Control

和依赖注射 DI//Dependency Injection

什么叫控制反转呢？

套用好莱坞的一句名言就是：你呆着别动，到时我会找你。

什么意思呢？就好比一个皇帝和太监

有一天皇帝想幸某个美女，于是跟太监说，今夜我要宠幸美女

皇帝往往不会告诉太监，今晚几点会回宫，会回哪张龙床，他只会告诉太监他要哪位美女

其它一切都交由太监去安排，到了晚上皇帝回宫时，自然会有美女出现在皇帝的龙床上

这就是控制反转，而把美女送到皇帝的寝宫里面去就是注射

太监就是是框架里面的注射控制器类 BeanFactory，负责找到美女并送到龙床上去

整个后宫可以看成是 Spring 框架，美女就是 Spring 控制下的 JavaBean

而传统的模式就是一个饥渴男去找小姐出台

找领班，帮助给介绍一个云云，于是领班就开始给他张罗

介绍一个合适的给他，完事后，再把小姐还给领班，下次再来

这个过程中，领班就是查询上下文 Context，领班的一个职能就是给客户找到他们所要的小姐

这就是 lookup()方法，领班手中的小姐名录就是 JNDI//Java Naming and Directory Interface

小姐就是 EJB，饥渴男是客户端，青楼是 EJB 容器

看到区别了么？饥渴男去找小姐出台很麻烦，不仅得找，用完后还得把小姐给还回去

而皇帝爽翻了，什么都不用管，交给太监去处理，控制权转移到太监手中去了

而不是皇帝，必要时候由太监给注射进去就可以了

看到 Spring 的美妙了吧，Spring 还提供了与多个主流框架的支持

可以和其它开源框架集成

Hibernate

名字取材自 ORM 最早的一句玩笑话//ORM 就是 OR-Mapping

说用了 ORM 之后，程序员就可以去冬眠了，而不需要操心那么多事

这里不得不说的是，该框架由于做得太好，以至于被 J2ee 招安，成为 EJB 3.0 的一部分

替代原有 EJB2.X 里面关于 Entity Bean 而成为 EJB ORM 的工具

这里解释一下 ORM//OR-Mapping

中文名对象关系映射

什么意思呢？我们知道传统的数据库都是关系型的

一条条记录以表格的形式储存，而表与表之间充斥着是关系/关联

比如说一个人，名字 zhaoce，性别男，年龄 23 那么数据库中是这么储存的

姓名 性别 年龄 zhaoce m 23 某女 f 22

而实际应用服务器中的实体都是以对象的形式存在，一个个对象

zhaoce 是以这种形式存在的

```
Human human=new Human();
human.setName("zhaoce")
human.setSex("m");
human.setAge(23);
```

这样的，那么我们知道，传统的 JDBC 是通过一个二维字符串将数据取出

需要我们自己将其包装成对象，在存入的时候，我们还需要将对象拆开

放入 sql 语句中//Insert into Huamn values('zhaoce','m',23)

然后执行该 sql 语句

太麻烦太麻烦，ORM 理念的提出改变了这一切，ORM 认为，这些东西应该由框架来做

而不是程序员，程序员做他该做的，不要为这种破事分心，还测试半天

于是就出现了 Hibernate，JDO，TopLink 等等，甚至.net 里面也有 ADO.ne
t

过去一段时间是 Hibernate 和 JDO 争风，现在看来 Hibernate 逐渐成为主流
并被官方接纳

成为规范标准之一，替代掉原来 EJB2.X 的 ORM EntityBean

TopLink 则是 Oracle 公司推出和 Oracle 数据库结合的一种 ORM

商业用软件，贵且复杂，不过正在逐渐开放

而象表示层一样，这一种专门面对数据层的代码也被称为数据持久层

所以数据持久层这一概念有时不仅仅指代数据库

关于 ORM，最高的境界应该是在 java 代码中不出现任何一句的 sql 语句

注意，是不包括 sql 语句，Hibernate 的 hql 以及 ejb 的 ejb-ql 不算在内

至于出现不出现 hql/ejb-ql 等替代 ql，这要视具体情况而定，不过最好也是
不出现

当然最后所说的过分理想的情况往往不现实，总之一句话

以 sql 为代表的 ql/*还有 hql,ejbql 等*/语句在代码中出现得越少越好

记住这话，现在未必能够理解，学了以后就懂了

这三个是目前最为常用的框架 而目前光已公布的框架就>500

还在不停增加中，不可能一一列举，有兴趣的可以去看相应文档 要指出的
是框架不是应用程序

只是一堆组件的有序复合，应用时不能脱离于应用服务器单独存在

给初学者之九：收尾

最后一篇介绍几个常见的概念

设计模式

这可不仅是 java 独有

我看的书就是 c++和 smalltalk 例子的

先说说什么是设计模式

模式是什么？模式是经验的总结，潜规则的抽象

什么意思呢？比如说我们坐飞机，上飞机前需要经过几个步骤

什么安检领取登机牌之类的，这一套流程能不能改呢？

可以，但为什么几乎全世界的航空公司登机前都是这一套流程呢？

因为航空公司经过长期实践之后得出了一堆结论和经验

并认为这样做才是最安全，或说是最有效率的

这就是模式，模式是编程高手之间交流的桥梁

两个编程高手通过统一命名的模式了解对方的思想

当然不借助模式可不可以？当然可以，只是模式无处不在，你不知道而已

又比如吃饭，每吃一口饭，我们要先端碗，拿筷子，张嘴，塞饭入口，咀嚼最后吞咽

这就是一套模式，我们给这套模式命名为吃饭

那么当老爸叫吃饭的时候，我们就能明白什么意思

而不用老爸进来呓呓啊啊并比画上半天，哑语也不是这么用的

这就是模式，已知的模式有 400 多种//好象更多，不记得了

比如数据库有数据库的设计模式，编程有编程的模式等等

面向对象有常用的 21 种模式，需要掌握，主要分为创建，行为，结构三类

J2ee 有 J2ee 的模式，Sun 公司出了一本书叫《J2EE 核心模式》可以拿来看看

必需要指明的是，模式不是规范，比如吃饭模式

没有人规定你吃饭非得要那么吃，你可以端碗，上抛，张嘴在下落后连碗一起吞咽

这也可以，只要你愿意，同样，只要你愿意，你就可以不遵循模式

模式之外还有反模式，学模式不可定势，不要学死，活学活用，无招胜有招才是最高境界

JavaDoc

文档工具，极其好用

可以根据注释自动生成 HTML 文档

Ant

98 年，有一位程序员在从欧洲飞回美国的飞机上想到了这么一个东西

从而改变了整个世界，他的名字叫 James Duncan Davidson

组织管理工具，可以这么描述它

比如你想在编译之后自动再次生成 JavaDoc

那么你只需要编辑 Ant 脚本//对，就像 Windows 脚本那样

然后批处理就可以了，不过现在 Ant 已经广泛集成到 IDE 中去

不需要自己手动编写，不过如果想要炫炫，据说此招百试不爽

JUnit

测试工具，Unit 家族可不只有 JUnit

还有其它版本的，这个不细说，具体实践一下就明白了

POJO

//Plain Old Java Object

就是传统的 Java 对象，也就是一个 JavaBean

由虚拟机来掌握其生死

常用的两个管理构架/规范是 Spring 和 EJB 容器

命名由来是某人//名字我忘了

觉得我们使用了太多的规范，以至于我们都忘记了纯粹的 java 对象

以至于我们都忽略了它的存在，所以叫了这么一个名字

以唤醒人们的记忆，这个意义上来说 EJB 其实不能算是 POJO

毕竟遵循了一堆的接口，但是不管怎样，接口归接口，还是没有继承类

没有被强加什么//遵循可以写空方法假遵循

所以说还是 POJO 也对

但是由于这种东西缺乏管理，不象 Servlet 有专门的容器管理并继承了一定的类

而没有管理的对象在虚拟机中是很危险的，因为垃圾回收机制各个虚拟机

不同

而且也不怎样，极有可能长时间不回收，这样在企业级的应用中呢

就有可能造成内存大量被占用从而死机，毫无疑问，这种机制需要优化

这种优化就是通过 EJB 容器或者 Spring 构架来实现

这么做还有一个好处就是迫使程序员对每一个类做封装

强迫他做管理，以达到防止内存泄露的目的，内存泄露最经常出现的错误就是

引用未释放，引用最典型体现在 new 这个关键字上，new 得越多引用得越多

随着时间地增长，有可能导致循环，不停 new new new new new.....

其中哪怕只要有一个 new 处理不当，虚拟机无法回收内存

那就极有可能完蛋，而且这种小 bug 越是在大的项目越是难以找到

有可能因为一个人而影响整个项目组，所以不妨记住我的一条经验

好的系统框架不应该在业务逻辑流程中出现 new 关键字

现在不理解也无所谓，将来有一天会明白的

SOA

面向服务的构架

不说太多，这个属于上上层建筑

不过不妨记住我的一句话，可以帮助理解这个概念

面向什么就是对什么做封装

面向对象就是对对象做封装

面向服务类似，剩下的靠悟性

反射

1.4 新增功能，非常强大

通过反射，程序可以解析出类本身的属性也就是变量

//注意这里说的属性不是.net 里面的属性，我不喜欢微软造的新名词，乱

还有行为也就是方法，然后通过 invoke()方法调用该方法

甚至可以新增对象等，java 首创，本是其它语言所没有的

后来被微软抄了去，利用该功能，开源框架广泛受益并大量采用，近乎疯狂地使用

具体就不说了，最后要指出的是，有一种说法是利用反射会降低效率

在早期的时候，的确是，现在不会了，放心使用

容器

5.0 以后的版本在 J2SE 中都出现了容器

各位甚至可以自己尝试用标准库去使用容器

推荐网站

www.javaeye.com //java 视线论坛，Hibernate 国内的权威

dev2dev.bea.com //bea 的 dev2dev 社区，用 WebLogic 首选的好去处

www-128.ibm.com/developerworks //ibm developer works 社区，ibm 产品的老家

www.jdon.com //j 道，Jboss 国内相对讨论会多一点的地方，有自己的框架

www.matrix.org.cn //matrix，有自己的框架，很清新的论坛

jcp.org //JCP，前面说到过了

sourceforge.net //开源的东西几乎这里都可以找到，除 java 外还有游戏共享等

saloon.javaranch.com //我常去，人气不错

www.apache.org //阿帕奇老家

www.jboss.com //Jboss 和 Hibernate 老家

www.springframework.org //Spring 老家

www.wiki.org //非常好的百科站点，可惜国内被封，创始人加入了 Eclipse zone

www.google.com //你要的这里有，不信？输入关键字再按一下那个靠左的白色按钮试试

书籍

《Thinking in Java》 //实话说，一般，尤其是翻译后的版本，原版还行

《Java 教程》 //电子工业出版社出版的那本，上下册，很厚，但翻译得不错

《21 天学通 Java》 //入门极好，但是《21 天学通 j2ee》极烂，不要买

《Mastering EJB》 //翻译过的书质量我不清楚，估计不怎样，请看原版书籍

《精通 Hibernate》 //看清楚作者，孙卫琴，其它人的别买

其它的可以不用了，网络上的远比书上来得多，来得好，虽然也来得杂

最后的建议

一，不要做一个浮躁的人

二，学好英语，很重要

三，阅读源代码和文档

四，共享源代码，不要做一个功利的人

五，热爱 Java