Java源码阅读的真实体会

刚才在论坛不经意间，看到有关源码阅读的帖子。回想自己前几年，阅读源码那种兴奋和成就感(1)，不禁又有一种激动。   
源码阅读，我觉得最核心有三点：技术基础+强烈的求知欲+耐心。   
  
说到技术基础，我打个比方吧，如果你从来没有学过Java，或是任何一门编程语言如C++，一开始去啃《Core Java》，你是很难从中吸收到营养的，特别是《深入Java虚拟机》这类书，别人觉得好，未必适合现在的你。   
虽然Tomcat的源码很漂亮，但我绝不建议你一开始就读它。我文中会专门谈到这个，暂时不展开。   
  
强烈的求知欲，我认为是阅读源码的最核心驱动力。我见到绝大多数程序员，对学习的态度，基本上就是这几个层次(很偏激哦)：   
1、只关注项目本身，不懂就baidu一下。   
2、除了做好项目，还会阅读和项目有关的技术书籍，看wikipedia。   
3、除了阅读和项目相关的书外，还会阅读IT行业的书，比如学Java时，还会去了解函数语言，如LISP。   
4、找一些开源项目看看，大量试用第三方框架，还会写写demo。   
5、阅读基础框架、J2EE规范、Debug服务器内核。   
  
大多数程序都是第1种，到第5种不光需要浓厚的兴趣，还需要勇气：我能读懂吗？其实，你能够读懂的。   
  
耐心，真的很重要。因为你极少看到阅读源码的指导性文章或书籍，也没有人要求或建议你读。你读的过程中经常会卡住，而一卡主可能就陷进了迷宫。这时，你需要做的，可能是暂时中断一下，再从外围看看它：如API结构、框架的设计图。   
  
我就说说如何读Java源码，以及我曾经的阅读感悟。   
  
**Java源码初接触**  
如果你进行过一年左右的开发，喜欢用eclipse的debug功能。好了，你现在就有阅读源码的技术基础。   
我建议从JDK源码开始读起，这个直接和eclipse集成，不需要任何配置。   
  
可以从JDK的工具包开始，也就是我们学的《数据结构和算法》Java版，如List接口和ArrayList、LinkedList实现，HashMap和TreeMap等。这些数据结构里也涉及到排序等算法，一举两得。   
面试时，考官总喜欢问ArrayList和Vector的区别，你花10分钟读读源码，估计一辈子都忘不了。   
  
然后是core包，也就是String、StringBuffer等。   
如果你有一定的Java IO基础，那么不妨读读FileReader等类。我建议大家看看《Java In A Nutshell》，里面有整个Java IO的架构图。Java IO类库，如果不理解其各接口和继承关系，则阅读始终是一头雾水。   
Java IO 包，我认为是对继承和接口运用得最优雅的案例。如果你将来做架构师，你一定会经常和它打交道，如项目中部署和配置相关的核心类开发。   
  
读这些源码时，只需要读懂一些核心类即可，如和ArrayList类似的二三十个类，对于每一个类，也不一定要每个方法都读懂。像String有些方法已经到虚拟机层了(native方法)，如hashCode方法。   
  
当然，如果有兴趣，可以对照看看JRockit的源码，同一套API，两种实现，很有意思的。   
如果你再想钻的话，不妨看看针对虚拟机的那套代码，如System ClassLoader的原理，它不在JDK包里，JDK是基于它的。JDK的源码Zip包只有10来M，它像是有50来M，Sun公司有下载的，不过很隐秘。我曾经为自己找到、读过它很兴奋了一阵。   
  
**Java Web开发源码**  
在阅读Tomcat等源码前，一定要有一定的积累。我的切实体会，也可以说是比较好的阶梯是：   
1、写过一些Servlet和JSP代码。注意，不是用什么Struts，它是很难接触到Servlet精髓的。用好Struts只是皮毛。   
2、看过《Servlet和JSP核心编程》   
3、看过Sun公司的Servlet规范   
4、看过http协议的rfc，debug过http的数据包   
如果有以上基础，我也不建议你开始读Tomcat源码。我建议你在阅读Tomcat源码前，读过Struts源码，Struts源码比 WebWork要简单得多。这个框架是可以100%读懂的，至少WebWork我没有100%读懂。我曾经因为读懂了Struts源码，自己写过一个 Web框架。   
  
当然，在读Struts框架前，最好看过它的MailReader等demo，非常非常不错的。   
如果你做过一些Struts项目，那么读它时就更得心应手了。   
在读Struts前，建议看看mvnforum的源码，它部分实现了Struts的功能，虽然这个BBS做得不敢恭维。   
  
如果你读过Struts，再开始考虑Tomcat源码阅读吧。   
不过，我还是不建议直接读它，先读读onJava网站上的系列文章《How Tomcat Works》吧，它才是Tomcat的最最简易版。它告诉你HttpServletRequest如何在容器内部实现的，Tomcat如何通过 Socket来接受外面的请求，你的Servlet代码如何被Tomcat容器调用的(回调)。   
学习JSP，一定要研读容器将JSP编译后的Servlet源码。   
为什么我总是称呼Tomcat为容器，而不是服务器？这个疑问留给大家吧。   
  
如果你一定要读Tomcat，那么就读Jetty吧。至少它是嵌入式，可以直接在eclispe里面设置断点debug。虽然Tomcat也有嵌入式版本。   
  
**Java数据库源码阅读**  
我建议，先读读Sun的JDBC规范。   
我想你一定写过JDBC的代码，那么这时候可以开始阅读源码了。   
如果了解JDBC规范(接口)，那么它的实现，JDBC Driver就一定要开始了解，我的建议是，读读mysql的jdbc驱动，因为它开源、设计优雅。在读mysql的JDBC驱动源码时，建议看看 mysql的内幕，官方正好有本书，《Mysql Internals》，我五年前读过一部分。比如你可以知道mysql的JDBC驱动，如何通过socket数据包(connect、query)，给这个C++开发的mysql服务器交互的。   
  
通过上面的阅读，你可以知道，你的业务代码、JDBC规范、JDBC驱动、以及数据库，它们是如何一起协作的。   
如果你了解这些内幕，那么你再学习Hibernate、iBatis等持久化框架时，就会得心应手的。   
  
读过JDBC驱动，那么下一步一定要读读数据库了。而正好有一个强大的数据库是用Java开发的，Hsqldb。它是嵌入式数据库，比如用在桌面客户端软件里，如Mail Client。   
我四年前为此写过一篇小文，就不介绍了。   
  
**Java通讯及客户端软件**  
我强烈推荐即时通讯软件wildfire和Spark。你可以把wildfire理解成MSN服务器，Spark理解成MSN客户端。它们是通过XMPP协议通讯的。   
我曾经在一个项目中，定制过Spark，当然也包括服务端的一些改动。所以它们的源码我都读过。   
我之所以推荐它们。是因为：   
1、XMPP够轻量级，好理解   
2、学习Socket通讯实现，特别是C/S架构设计   
3、模块化设计。它们都是基于module的，你既可以了解模块化架构，还可以了解模块化的技术支撑：Java虚拟机的ClassLoader的应用场景。   
4、Event Driven架构。虽然GUI都是Event驱动的，但Spark的设计尤其优雅   
  
这么说吧，读它们的源码，你会为做一名程序员而自豪，因为无论是他们的架构设计还是代码,都太漂亮了。   
  
**Java企业级应用**  
当然了，就是Hibernate、Spring这类框架。   
在读Spring源码前，一定要先看看Rod Johnson写的那边《J2EE Design and Development》，它是Spring的设计思路。注意，不是中文版，中文版完全被糟蹋了。   
在读Hibernate源码前，一定要读读Gavin King写的那本《Hibernate in Action》，同时，应该再读读Martin Fowler写的《企业应用架构模式》，它专门谈到持久化框架的设计思路。当你觉得这两本书读透了,再去看它们源码吧。   
而且，在读源码前，你会发现它们用到很多第三方Jar包，二三十个，你最好把那些Jar包先一个个搞明白。   
  
说到企业应用，一定会涉及到工作流。我当年读过jBPM的源码，网上有介绍jBPM内核的文章(银狐)。我感觉它的内核也就两千行，不要害怕。我曾经阅读jBPM源码的博客。   
当然了，读工作流源码，前提是一定要对其理论模型有深入的了解，以及写过一些demo、或做过一些项目。   
  
我上面介绍的这些，是我自己读过的，也适合一般人阅读。   
我也读过一些非Java源码，感觉不错，也推荐给大家：   
**dojo源码** 它的架构设计得很优雅，仿Java的import和extends。但实际应用起来一塌糊涂。我们当年基于这个开发了自己的框架，不过我不是主力。   
  
**Flex源码** Flex 08年底刚刚开源后，我就用它做过一个中型项目，应该说是国内的技术先行者。当时市面没有有深度的书，也没有开源项目。我纯粹是看Flex的Help文档和源码，把项目搞定的。两三年过去了，现在觉得系统设计得蛮优雅的。   
  
好了，先介绍到这里。   
上面说到的这些Java源码，我都是4年前、甚至更早读过的。技术变化这么快，像互联网的高速发展，催生很多高性能、分布式数据库，如hadoop。我一看，发现自己已经落伍了。   
这几年，想必已经出现了很多优秀的框架，大家不妨分享出来。   
  
  
**题后记**  
这三年，一直在创业，主要是技术应用，偏业务，源码阅读不多，但很欣赏那些专注于技术的狂热者。   
现暂别创业，进入一家电子商务公司，负责其B2C网站的改版和运营。