【Java入门】如何掌握Java（J2EE篇） PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/145/2021_2022__E3_80_90Java_E5_85_A5_c104_145554.htm J2EE，作为开发mission-critical的企业级应用的一整套规范的整合平台。其规范之多、内容之广，从而给开发J2EE应用带来了很多“ 麻烦 ”。比如，为实现内容的RDBMS存储，我们可能的方法有JDBC、Entity Beans、JDO、O/R Mapping工具（TopLink、Hibernate）、XML-DBMS、JAXB等方法（其中一些方法不是J2EE规范所包含的）。因此，为实现J2EE各层（至少有表示层、控制层、商业逻辑层等3层）以及层与层之间的耦合，J2EE系统架构师需要考虑的问题会很多。加上，J2EE本身的快速发展，给架构、开发具有工业强度的J2EE应用带来一些难题。同时，软件开发技术从来就没有“ 银弹 ”，所以J2EE技术也不是万能的。但是，如果我们在结合具体商业需求的基础上，合理的应用好J2EE技术，其结果可想而知。本文试探从自己以往的项目经验来探讨开发J2EE应用应该遵循的几点准测入手，以起到抛砖引玉的作用。如果能达到这一点要求，则万分激动。本文结合JBoss 3.2.1下的J2EE应用开发为例展开论述。1，结合商业需求选择合理的架构 如果脱离商业需求，而单独的讨论技术本身的优势是不够的。各项技术都有产生的特定背景，其中很多都是来自工业需求而触动的。一般而言，企业信息系统（EIS）都要求自己稳定、安全、可靠、高效、便于维护。同时，各个企业信息系统都有自己独特的要求，可能有些时候需要考虑与原有遗留系统的集成，所以了解各个企业信息系统具体的商业需求对于整个系统的架构显得很关键

。比如，如果待开发的J2EE应用系统中使用到的数据大部分来自于外在数据源；而这些数据可能是通过JDBC直接从外在数据源导入到待开发的J2EE系统的Database中。对于这种情形，如果在开发过程中，仅仅使用JDBC来操作数据库，对于小强度（并发访问用户少、数据流量少）的情形，显然是比较合适的；但如果，并发访问用户较多、数据流量大，对Database层使用较为频繁的情形，则显得有些力不从心。因此，对于这种需求，我们可以考虑采用Entity Beans with Caches。打个比方，在JBoss 3.2.1中对于Entity Beans的Cache策略有多种，这时可以考虑使用，，即“ Standard CMP 2.x EntityBean ”，方式并采用“ D ”类型的commit-option来保证Entity Beans的内容与数据源的同步，并使得系统的性能得到大大改善（同直接使用JDBC相比）。其中，可以将一些Entity Beans设置为read-only，以改善性能。当然，在这里也可以采用其他一些O/R Mapping技术，比如TopLink。再比如，考虑这样一种情形：如果待开发的企业信息系统使用到的数据都是由系统本身生成和操作的，则建议采用：CMP Entity Beans技术。Entity Beans给大家的印象很坏，这可能与EJB 1.1给大家留下的坏映象有关吧。但是，EJB 2.0(或者说2.1)得到了很大的改善，Local Interfaces、CMR、Read-Only、Session Fa&ccedil.ade模式给Entity Beans注入了活力。当然，并发用户多、数据流量很大时才会体现出使用Entity Beans的优势。其中，有一点很关键：要注重Entity Beans技术的性能调优，各个应用服务器都有自己的一套性能调优方案。对于JBoss 3.2.1，配置文件standardjboss.xml提供了Entity Beans技术调优的入口。比如，Bean Lock策略的合理使用对于Entity

Beans的调优就显得很重要。这样使得，我们可以更加关注于系统的商业逻辑，而不只是底层的Database（EJB调优处于EJB Container中，因此我们处在J2EE性能的高端，而不是底端，即Database层。同时，Database层的调优使得J2EE系统的数据库移植性大打折扣。）。简而言之，要结合各个系统的特定需求和状况给出具体的技术架构方案，而不能孤单的论述技术本身的好坏。2，Framework的合理选用设计模式在J2EE应用系统中扮演着重要的角色。因此，有一个问题摆在大家面前，是自己来实现具体的设计模式，还是借助于Third-party Framework。如果贵公司不大，或者说公司不想在J2EE基础应用Framework投入很多精力，选用现有的较为成熟的、稳定、与现有J2EE Specification兼容的技术框架会比较明智。一般而言，Framework本身，或者说J2EE平台本身都是实现并优化了具体的设计模式、规则，比如业务代理、Service Locator（包括Web Tier和EJB Tier各自的服务定位器，起到统一管理有限资源、Cache相关资源的作用，便于系统移植）、Front Controller、DAO等等。现有的J2EE Framework比较丰富。比如：Struts: 对于实现了Model 2类型的Framework，对于现在以及将来（随着JSF规范、技术的成熟），选用她是一种明智之举。目前，Struts已经发展到1.1版本。其内在的MVC主线、对后端数据操作方式没有限定、集合了Apache Jakarta项目组的优秀相关项目的精华，可谓是开发J2EE应用的佳品。同时，对于具有.NET Web Forms功能的下一代J2EE平台技术JSF而言，Struts本身可考虑到与JSF的兼容和集成性。比如，通过JSP呈现表示层、Servlet呈现控制层、EJB呈现数据存储层。各层之间，可以通过值对象、HTTP相关对象来通讯，实

现J2EE相关技术的完美应用。Log4j: 我想对于习惯采用“System.out.println(“ ”).”的读者而言，Log4j是大家的福音。尽管Java 2 Standard Edition也具备java.util.logging包来保证日志的输出，但Log4j的简单、高效、灵活已经成了很多项目的选择。日志，在某种程度上可以考验系统的稳定性、正确性，所以采用可配置的Log4j（目前，Log4j已经考虑到了与java.util.logging包的兼容性）是不会错的。比如，JBoss 3.2.1本身就是借助于Log4j来管理日志的。realMethods: 可能有些读者还不知道这一款杀手锏。那好，这里就简要作一介绍。realMethods是一开发J2EE应用的Framework，她不同于Struts(主要在于实现Model 2，J2EE应用前端)；realMethods对于J2EE应用的各个层面都有详尽、高效的支持。同时，realMethods以前还是商用软件，现在已经成为了Open Source的产品，因此现在可以参看其全部源代码。BC4J: Oracle公司推出的用于Java的商业组件。其内容和外在的特点和优势，不言而寓。当然，类似的Framework很多很多。作为开发J2EE应用的团队而言，我们需要对各种Framework加以筛选，选择适合项目需求、团队、公司发展方向的框架。一般情况下，待开发的目标产品不宜采用过多的Framework。其一，J2EE各个技术发展很快，过多的Framework使得系统的后续升级、维护不利；其二，可以借鉴其中的好的一面，比如研究realMethods实现的相应的设计模式，并改造她以适合我们的项目需求；其三，Framework本身会有变动，如果选用过多，会给开发团队加重负担，从而不利于项目管理。有选择的使用现有的成熟Framework能提升大家的开发效率、开发水平。