Java 项目开发实用案例

综合运用Java Web应用开发的热点技术

巧妙通过Cache实现高效海量访问系统

探密Framework并开发自主Web框架

赵如意
徐丁立
曹金利
王金明
编著

科学出版社
www.sciencep.com

Java 项目开发实用案例

赵如意　徐丁立
曹金利　王金明　编著

科学出版社
北京

内 容 简 介

本书是以项目案例为导向的实践指导书。首先，用一章的篇幅阐述J2SE5.0的特性和Java的主流开发框架，开发工具等，而后，分五章介绍了五个开发项目：电子相册系统、人力资源管理系统、高海量访问系统、公司办公系统和日常管理系统。在项目开发体验的基础上，本书又针对开发框架做了详细的描述，同时，在附录中介绍了Java编程的约定。阅读本书可以领略项目开发的特点和奥秘。

本书所附光盘中含每个案例的完整工程文件和相关说明文档，读者可以根据每一章所在的目录下的说明文件安装、配置和运行对应的工程项目。

本书可作为大专院校计算机及相关专业学生的选修课教材，或毕业设计的指导教材，对从事项目开发的专业技术人员也有一定参考价值。

图书在版编目(CIP)数据

Java项目开发实用案例/赵如意等编著. —北京：科学出版社，2006

ISBN 7-03-017242-6

Ⅰ. J … Ⅱ. 赵… Ⅲ. JAVA语言-程序设计 Ⅳ. TP312

中国版本图书馆CIP数据核字（2006）第048873号

责任编辑：王淑兰 庞海龙／责任校对：柏连海

责任印制：吕春珉／封面设计：耕者设计工作室

科学出版社 出版

北京东黄城根北街16号

邮政编码：100717

铭浩彩色印装有限公司 印刷

科学出版社发行 各地新华书店经销

*

2006年7月第 一 版 开本：787×1092 1/16

2006年7月第一次印刷 印张：29 1/2

印数：1—4 000 字数：685 000

定价：45.00元（含光盘）

（如有印装质量问题，我社负责调换〈环伟〉）

销售部电话 010-62136131 编辑部电话 010-62130750（HI01）

前　言

Java 自 1995 年诞生以来，其应用在全球呈一路飙升的趋势，在企业级的服务器端应用更加倍受青睐。尤其在安全性和规模性要求较高的银行、证券、金融等领域，Java 已成为各种解决方案的首选。在一些知名的求职网站，Java 程序员的需求量最多。显然，写一本饱含 Java 开发人员经验的《Java 项目开发实用案例》，无论对初入程序开发之道的新手，还是正从事软件开发职业的开发人员都是有益的。

在 2003 年末，许多人已经感觉到 IT 业求职的压力了。当时 IT 行业属于 2001 年网络泡沫以后的冰点时期，求职场上招聘要求都是要有几年开发经验的人员。求职的严酷形势使人们清醒感到在学校学的内容和真正工作所需的技术存在“学不致用”的问题。大学里面做的项目即便是类似的内容（比如 B/S 结构开发等），也只是随心所欲地去写，根本没有一个整体构架以及共通化以供代码复用的观念——这是缺乏具体的项目经验的人都会有的情况。

本书作者应科学出版社之邀，为已有一定 Java 基础而又缺乏完整项目经验的读者写一本案例教程。由于本书是以实例为主，因此召集一些有丰富工作经验的同行合著。大家都想把书写好，各种构思也比较多，撰写过程中发现写一本真正有价值的书也并不是一件容易的事情。尽管如此，大家坚持不懈地努力，终于将一本内容颇为丰富的书完成了。希望读者通过阅读本书能在项目开发实践过程中有所收获。

本书不是一本 Java 基础教程，书中没有论述基本的 Java 语法以及具体的 API。读者为了能流畅地阅读本书，需要了解以下知识：

首先需要有 Java 面向对象编程的基本概念，熟悉 Java 的基本语法和常用的 API。了解 HTTP 以及 Web 的基本知识，具有初步的编写简单 applet 的能力。

本书重点在于介绍用 Java 开发的项目案例。通过阅读本书，读者可以了解如何设计一个适合项目需求的框架，如何设计并实现能适合多人协作的框架，并获得一定的概览完整项目的能力。

本书是以项目案例为导向的实践指导书。随书光盘的内容包括各个案例（第 2 章到第 7 章）的完整的工程文件和相关说明文档，其中源码的重点部分在书中均有详细分析和说明。读者可以根据每一章所在的目录下的说明文件安装、配置和运行对应的工程项目。由于本书的第一章是 Java 相关的知识、特点、框架和工具的概述，因此在随书光盘中没为本章安排电子文件。

本书的作者和写作分工如下：

本书 1.2 节“J2SE5.0 新特性”简介、第 5 章“公司办公信息管理系统”和附录“Java 编程约定”由赵如意编写。他目前正在北京航空航天大学计算机学院软件开发环境国家重点实验室攻读硕士学位，师从校长李未院士，曾在联想软件设计中心、IBM 中国研究中心（CRL）和 IBM 中国软件开发中心（CSDL）参加相关项目的开发工作。读者可通过 zhaoruyi@gmail.com与其联系。

本书1.3节“主流开发框架概览”、第3章“人力资源管理信息系统”和第7章“框架探秘”由曹金利编写。他是中国最大的技术社区——CSDN社区Java版块IntelliJ IDEA子版版主，现在NCS公司就职。

本书第4章“高效海量访问系统”由王金明编写。他是某公司的Java架构师并在CSDN社区作Java版主。

本书1.1节“Java漫谈”、第2章“电子相册系统”和第6章“日程管理系统”由徐丁立编写。2004年他于东南大学毕业后一直在联迪恒星（南京）信息系统有限公司从事项目开发工作，并是CSDN社区Java版设计模式子版版主。徐丁立的个人Blog是http://blog.csdn.net/dlxu，上面有他自己学习Java的历程，对读者可能会有一定的参考作用。

本书的实例大部分是几位作者在工作之余写下的代码，很多架构风格可以在实际项目开发时参考。对缺乏实际项目经验的在校生和Java初学者也有一定的学习价值。由于作者水平有限，还望读者批评指正。最后，祝您阅读愉快，并取得自己想要的收获。

作　者

2006年4月

目　录

第 1 章　日积月累……1
1.1　Java 漫谈……2
1.1.1　Java 语言诞生及其发展……2
1.1.2　Java EE 的发展与展望……9
1.2　J2SE 5.0（Tiger）新特性简介……10
1.2.1　简易开发……10
1.2.2　可扩展性和性能……15
1.2.3　监视和管理能力……15
1.2.4　桌面客户端……17
1.2.5　其他新特性……19
1.3　主流开发框架概览……20
1.3.1　Struts……20
1.3.2　Spring……21
1.3.3　Hibernate……23
1.3.4　Webwork2……25
1.4　主流工具使用指南……25
1.4.1　JBuilder……25
1.4.2　Eclipse……30
1.4.3　Java Studio Standard……35
1.4.4　Tomcat……41
1.4.5　数据库服务器……42
第 2 章　电子相册系统……51
2.1　概述……52
2.2　案例简介……52
2.3　案例设计……52
2.3.1　需求分析……52
2.3.2　数据库设计……53
2.4　开发实现……54
2.4.1　Framework 采用决策……54
2.4.2　Web 应用的核心 Web.xml……54
2.4.3　Struts 的配置核心 Struts-config.xml 文件……57
2.4.4　自定义 Taglib……59
2.4.5　图片的显示……70
2.4.6　文件上传操作……73

第 3 章　人力资源管理信息系统……76
3.1　概述……77
3.2　案例简介……77
3.2.1　案例背景……77
3.2.2　功能需求……78
3.2.3　开发环境说明……78
3.3　分析设计……78
3.3.1　需求分析……78
3.3.2　设计思路……78
3.3.3　功能分析……78
3.3.4　框架设计……79
3.3.5　模块划分……79
3.3.6　数据库设计……79
3.4　开发实现……82
3.4.1　系统框架图……82
3.4.2　配置文件说明……82
3.4.3　功能模块……93
第 4 章　高效海量访问系统……141
4.1　概述……142
4.2　案例简介……142
4.3　需求分析……143
4.4　功能分析……144
4.4.1　发帖功能……144
4.4.2　帖子浏览功能……144
4.4.3　Hibernate Cache Provider……145
4.5　数据库设计……145
4.6　开发实现……145
4.6.1　Framework 采用决策……145
4.6.2　Hibernate Cache Provider 实现……154
4.6.3　发帖功能……168
4.6.4　浏览功能……177
第 5 章　公司办公信息管理系统……180
5.1　概述……181
5.2　案例简介……181
5.3　分析设计……181
5.3.1　系统目标设计……181
5.3.2　系统结构设计思想……182
5.3.3　系统功能设计……183
5.3.4　系统工作流程设计……185

5.3.5 数据库需求分析和逻辑设计 ····· 185
5.3.6 数据库结构创建 ····· 186
5.3.7 数据库结构的概念模式 ····· 192
5.4 开发实现 ····· 193
5.4.1 配置 server.xml 和 Web.xml ····· 193
5.4.2 自定义数据库连接池及其实现 ····· 198
5.4.3 Java Mail 及认证的实现 ····· 209
5.4.4 Java Bean 及 MD5 应用 ····· 211
5.4.5 系统用户身份验证子系统 ····· 214
5.4.6 公司员工信息管理子系统 ····· 222
5.4.7 公司财务信息管理子系统 ····· 254
5.4.8 公司内部信息交流子系统 ····· 291
第 6 章 日程管理系统 ····· 338
6.1 概述 ····· 339
6.2 案例简介 ····· 339
6.3 案例设计 ····· 339
6.3.1 需求分析 ····· 339
6.3.2 数据库设计 ····· 341
6.4 开发实现 ····· 347
6.4.1 Framework 采用决策 ····· 347
6.4.2 Web.xml 文件 ····· 347
6.4.3 Struts-config.xml 文件 ····· 350
6.4.4 从系统启动开始 ····· 352
6.4.5 自定义 Taglib ····· 362
6.4.6 扩展 Framework 构造 ····· 369
6.4.7 一个画面的业务流程 ····· 388
6.4.8 Tiles 框架页面的搭建 ····· 391
6.4.9 批处理的业务流程 ····· 392
第 7 章 框架探秘 ····· 394
7.1 Framework 简介 ····· 395
7.2 某框架简介 ····· 396
7.3 分析设计 ····· 397
7.4 逻辑设计 ····· 399
7.4.1 配置部分 ····· 399
7.4.2 代码部分 ····· 404
附录 Java 编程约定 ····· 455

第1章
日积月累

学习目标

1. 了解 Java 语言的诞生和发展；
2. 了解 Java EE 的发展与展望；
3. 了解 Java SE 5.0（Tiger）的新特性；
4. 了解主流开发框架；
5. 了解主流开发工具。

学习重点

1. Java 语言的诞生和发展；
2. Java EE 的发展与展望；
3. Java SE 5.0（Tiger）的新特性；
4. 主流开发框架（Struts、Spring、Hibernate、WebWork2）；
5. 主流软件工具（JBuilder、Eclipse、Java Studio Standard、Tomcat、Oracle、MySQL、SQL Server）。

1.1 Java 漫谈

1.1.1 Java 语言诞生及其发展

1. Java 的诞生

1991 年，SUN MicroSystem 公司的 James Gosling 等人，为了能在电子消费类产品上进行交互操作而开发了一个名叫 Oak 的软件。当时大家并没有太关注它。后来，SUN 公司加深对 Oak 的研究，并于 1995 年把 Oak 命名为 Java，于是 Java 诞生了。随着 Internet 的迅速发展，Java 开始不断展现它的魅力，特别是 Java 的 Applet 技术成为了一种时尚。当时各大浏览器厂商争相把支持 Applet 作为自己的卖点。美国著名的“PC Magazine”杂志把 Java Applet 评为 1995 年十大优秀科技产品。微软首席架构师比尔·盖茨先生在悄悄观察一段时间以后，也不无感慨地说：“Java 是这么长时间以来最为卓越的程序设计语言。”当时 Java 的运用主要体现在 Applet 上，其他方面则运用得不是很深入，所以很多人也担心 Java 是否会昙花一现。但到后来，Java 2 的诞生彻底打消了人们的这种疑虑。

2. Java 的特点

我们通常所说的广义上的 Java 技术并不仅仅只局限于 Java 这种编程语言，而是整个 Java 平台加上 Java 编程语言。Java 编程语言作为一种程序开发语言，具有以下几种优点：

（1）可面向对象

Java 是一种面向对象的编程语言，Java 所有的程序都是由类（Class）组成。而且 Java 拥有继承机制，子类可以使用父类中提供的方法，实现代码的复用。不过 Java 抛弃了复杂的多重继承，取而代之，采用了接口这种新的方法来达到多重继承的效果。

（2）网络性

Java 的诞生是伴随着互联网的发展的，所以对于网络应用 Java 有非常好的处理能力。比如通过 JDK 所提供的类库处理 TCP/IP 协议，可以迅速开发网络应用程序。

（3）健壮性

无论是在编译还是运行时，Java 都会对可能出现的问题进行检查。比如对于类型转换，对编译类型进行严格检查，对于向下类型转换，编译器要求强制显式声明转换，否则不予通过编译。在 Exception 的处理中，如果未对那些可能出现的可捕捉的异常进行处理，那么编译就无法通过，会强制要求程序员对这些异常进行处理。Java 中出现了“垃圾回收”这一概念，虚拟机会自行调度以决定什么时候把哪些不会用到的对象从内存中清除，达到内存释放的目的，降低程序员因为疏忽或错误造成大量内存未及时释放而导致系统崩溃的可能性。Java 摒弃指针而采用了对象的引用来代替，有效防止了程序员因指针使用不当而造成访问到内存中不该访问的资源，甚至修改了那些资源而造成系统的崩溃。

（4）安全性

Java 的安全机制一向很好，在 Java 刚开始应用时，Applet 的安全就被众人称赞。在 Java 2 以后，采用了双亲委派访问模式，使得指定的对象只能访问指定的资源。

（5）并发处理性

Java 的一大特点就是多线程并发处理能力。Java 的多线程 API 功能非常强大，特别在并发程序设计大师 Doug Lea 的设计下，Java 的多线程 API 的性能非常出色。

（6）动态执行

java.lang.reflect 包提供了 Java API 中的反射功能，可以在程序运行过程中动态决定执行什么方法，调用什么 Class，或者生成什么实例。不过要注意的是，在 Java 早期的版本中（JDK 1.3 以及以前的版本），反射所带来的性能开销是非常大的，所以如果基于这些版本的 JDK 环境进行开发的话，建议尽量少用反射机制，从而提高系统的性能。

（7）API 丰富，便于开发

Java 在诞生时期，开发者们便给 Java 设计了大量的 API。在最新的 JDK 5.0 中，打开 SUN 公司提供的 Java Doc API 参考，便可看到丰富的 API，种类十分齐全，功能也很强大。这些 API 都经过了长期测试几乎没有 bug，程序员可以放心使用它们敏捷开发所需功能。

（8）平台无关性

这点也许应该是 Java 的最大特色了。Java 程序的执行不再像 C 语言或者 C++语言那样直接编译成可执行文件，而是先编译为 bytecode，它是一种特殊的二进制文件。这种文件在 Java 虚拟机中可以执行，所以程序只要编写一次，再编译为 bytecode，然后在不同操作系统下的 Java 虚拟机上运行，运行的结果就基本是一致的。早在 Java 产生初期，SUN 公司就提出了“一次编写，到处运行”的口号。的确，Java 实现了这个诺言。在如今的企业级开发中，在 Windows 操作系统上开发，在 UNIX 操作系统上实际运行的案例比比皆是。这也许就是 Java 最大魅力之所在。

3. Java 2 开放式平台的诞生

当然，Java 本身也在不停发展。Java 拥有的开放式 Java Community Process（JCP）决定了其发展。Java 2 出现以后，Java 便进入了一个全新的时期。1999 年，Sun 公司推出了以 Java 2 平台为核心的 J2EE、J2SE 和 J2ME 三大平台。随着三大平台的迅速推进，世界上已有一股强大的 Java 应用浪潮。在 2005 年的 SUN ONE 大会上，把 J2SE、J2EE、J2ME 中的“2”去掉，称为 Java SE、Java EE 和 Java ME。下面将分别简要介绍这三大平台。

（1）Java SE

Java SE（原 J2SE）就是 Java Standard Edition 的英文缩写，主要针对的是桌面软件的开发。目前，桌面软件开发有三种图形界面 Framework。第一种是 AWT，这个历史比较悠久，早在 JDK 1.0 的时候就已经拥有了。AWT 的原理是找出各种平台下图形控件的交集，然后用 AWT 分别调用各个平台下控件的本地代码来达到实现 GUI 界面的目的。不过由于其控件是每个操作系统的交集，所以控件种类较少，功能也并不是很强，AWT 已经基本不被使用了。目前普遍使用的主要是以下两种图形界面 Framework：第一种是 Swing，这是 SUN 公司提供的，在 Java 2 以后正式加入了 Java 的公开 API。Swing 的实现原理是取所有平台下图形控件的并集，然后用 Java 在虚拟机的层面上模拟出那些控件。这样一来所带来的好处就是图形控件种类比较丰富，功能也比较强大，而且由于是用纯

Java 实现的，所以跨平台性很好。但是由于所有控件都是由 Java 虚拟机来模拟，所以代码运行速度比较慢，不过这种情况已得到改变。在 2004 年 SUN 推出了 JDK 5.0 后，Swing 的运行速度已大大提高，据 SUN 公司透露，JDK 6 的 Swing 运行速度将更上一层楼。GUI 的第三种 Framework 是 IBM 推出的 SWT，这也是当时为了解决 Swing 效率较低而设计的一种比较高效的 Framework。SWT 的原理折中了 AWT 和 Swing，如果本地平台中拥有你所想用的控件，那么 SWT 就自动调用本地代码来生成，如果本地平台没有你要用的这种控件，那么 SWT 则仿照 Swing 来模拟出这种控件的功能。因此 SWT 在性能上优于前两者，大名鼎鼎的 Eclipse 就采用了 SWT，Eclipse 比同规模的用 Swing 写的 Java 程序执行效率高。不过进入了 JDK 5.0 以后，笔者使用了一段时间的体会是，两者的运行速度相差无几，而 Swing 的界面比 SWT 美观，Sun 公司也极力推荐用 Swing，因为 Swing 是纯 Java 的，在跨平台上效果会比较好。

（2）Java EE

Java EE（原 J2EE）就是 Java Enterprise Edition 的缩写。顾名思义，就是 Java 的企业版本，也是 Java 中最为复杂，应用最为广泛的一项技术。关于 J2EE，在后叙章节中会有更详细的介绍。

（3）Java ME

Java ME（原 J2ME）是 Java Micro Edition 的缩写。其实 Java 的设计初衷也许就是想按照 Java ME 的路线来发展。当初 SUN 公司开发 Java 的目的，是为了在各种复杂的电子设备中提供一个通用的技术，让同样的软件可以在不同的电子设备中顺利运行。也许是“有心栽花花不开，无心插柳柳成荫”吧，Java 在互联网上大显身手，先是在 Applet 中展现其魅力，后又在 Java EE 的 Web Service 中大红大紫。但 Java ME 一直不温不火，很长一段时间以来，Java ME 都只被当作开发手机游戏的工具，它的潜在能力还没有被完全挖掘出来。不过，随着网络的不断发展，机顶盒、手机等智能设备的硬件条件也越来越好，在这些领域中 Java ME 应该会大显身手。

4. Java EE 的诞生背景

虽然 Java 发明出来的本意是成为消费电子产品开发的编程语言，而却在企业级应用中找到了自己的重要位置，率先在网络中得到了广泛应用。这虽然看似偶然，但是仔细分析一下也是必然的。

Java 从一开始就是跨平台的，可以忽略各种操作平台的异构性。只要写好 Java 程序，就可以在任何平台下运行。这也就是 SUN 公司倡导的“一次编写，到处运行”的宗旨。

虽然在个人电脑上 Windows 占据了绝大多数操作系统的份额，不过在服务器方面，则并不是 Windows 的天下。Windows 的特点就是易用，娱乐性非常好，不过 UNIX 具有公认的优良的稳定性和安全性，它更适合作为服务器的操作系统（现在也有不少服务器用 Linux）。为了编写这些操作系统下的软件，传统的做法就是在个人电脑上先写好源代码，然后送到 UNIX 下来编译生成 UNIX 下的本地可执行代码，最后再运行。这样做带来了很大的可移植性问题，比如拿 C 语言来说，在不同操作系统下，同样的代码表现是不一致的，有可能在 Windows 下一段代码经过编译后运行产生的是 A 这个结果，但是到了 UNIX 下后，因为 UNIX 下没有某些在 Windows 下才有的特性，所以就无法产生 A 这

个结果了。

不过，Java 的出现改变了这个局面，因为 Java 的原理就是在每个不同的操作平台上都实现一个平台本身的虚拟运行环境，也就是 Java 虚拟机，所有的 Java 程序都是运行在这个 Java 虚拟机上的。所以不论环境平台变成什么样，只要实现于这个平台的虚拟机是符合 Java 虚拟机规范的，那么这个 Java 程序就可以正常运行，达到了“一次编写，到处运行”的目的。正是由于这个特性，使得 Java 特别适合于大型企业级软件开发。因为人们可以在 Windows 上进行程序开发，开发完以后可以把编译好的代码直接在 UNIX 服务器上运行，因为 Java 代码在其所有的虚拟机上运行出的结果基本都是一样的（有可能多线程中线程调度模式不一致），所以我们可以很方便地在 Windows 中开发，在 UNIX 下部署，大大提高了生产力。

5. Servlet 的应用

Servlet 技术的出现也许是 SUN 公司的一贯风格，SUN 喜欢把具有一定应用范围和特性的程序叫做 XXXlet，比如 Applet，以及后来在 JSP 中出现的 Scriptlet、J2ME 中的 Midlet 等。Servlet 就是一种特殊的小程序，它接收一个请求（request），返回一个应答（response）。Servlet 的子类 HttpServlet 是 Servlet 在 HTTP 协议下的实现。众所周知，HTTP 是一个无状态协议，是最简单的请求—应答机制，请求和应答之间不再有别的任何联系。HttpServlet 接收了客户端传来的 Request，然后从 Request 中取得用户所传递过来的信息。再经过自身的处理后把返回的数据用 Response 返回，返回的格式多种多样，有时候是返回一个页面，有时候是返回一个指令，指示浏览器转向另外一个页面，也有时候返回一个二进制流来达到下载文件的目的等。

Servlet 是整个 Web 应用程序中最基础的部分，相比起类似功能的其他技术，由于 Servlet 是编译过的代码，所以 Servlet 的性能比之前类似功能的 CGI 和使用解释方式运行的 ASP 高很多。可以说自从有了 Servlet，高效的动态 Web Service 才得以实现。

6. JSP 的出现

随着 Servlet 的发展，人们发现 Servlet 在解决 Web Service 中的问题时很多情况是在 Java 程序中嵌入 HTML 网页代码，这给维护修改带来了极大的不便。因为在通常情况下一个网页设计师对编程并不怎么擅长。同样的，一个程序员对网页设计也不会非常擅长，同时擅长网页设计和程序设计的人员是很有限的。为了让网页的动态显示更为直观，SUN 公司推出了 JSP 技术。

JSP 跟传统的 HTML 网页比较类似，主题框架由 HTML 页面构成，中间夹杂 JSP 特有的代码，称为 Scriptlet。JSP 的运行机制就是当用户访问 JSP 页面时，Web Server（也就是通常所说的容器）便根据 JSP 页面代码自动生成相应的 Servlet 代码，并在后台进行编译，所以用户访问的也是一个特殊的 Servlet。其实，如果说 Servlet 是 Java 代码中嵌入 HTML 代码的话，那么 JSP 页面就是在 HTML 页面中嵌入 Java 代码。本质上看，当初的这种 JSP 形式和 Servlet 也差不多，只不过换了种形式而已。如果页面比较复杂，在进行网页修改维护的时候，任务将十分艰巨。所以对某些常用的功能，通常把它封装到一个 Class 中，并且使用 XML 配置符进行定义，也就是自定义的 Tag。

自定义 Tag 的出现让 JSP 程序清洁了很多，许多重复性的代码从页面程序中消失了，取而代之的是一个个封装得很整齐的 Tag。可以说，如果 Tag 在开发初期经过了精心的设计，那么可以非常大地提高开发效率，减少许多不必要的开发时间。笔者曾经开发过一个项目，在客户端已经实现了一套 Tag，这些 Tag 的功能足够强到我们都不需要写程序就可以完成这个 MIS 系统，所有可能出现的需求都可以用这些 Tag 完成。由于 Tag 是通过测试才使用的，所以 bug 比较少，开发速度大大提高，而且 bug 出现频率比通常的 JSP 开发低很多。

7. JDBC 等技术的出现

任何稍微有点规模的软件都不可能离开数据库。J2EE 被定位为企业级的软件开发技术，数据库操作更显得尤为重要。于是专门针对数据库操作的 Java API 出现了，这就是 JDBC。由于数据库种类多种多样，JDBC 其实是由一套接口组成的，由各种数据库开发商自行提供实现。而我们写数据库操作方法的时候，在代码中操作的都是诸如 Connection、PreparedStatement 以及 ResultSet 等公共接口，每个数据库驱动都提供了对应的开发接口，同时都符合这套规范，所以 JDBC 实现了操作时的数据库无关性。

8. JNDI 和 RMI

众所周知，在 Java 虚拟机中，Java 的对象是以 Object 的形式存在的，但是如果想把虚拟机 A 中的一个 Object 以某种形式存储起来，然后通过网络发送到另外一台计算机上，那么这台计算机上的 Java 虚拟机 B 拥有刚才虚拟机 A 中的那个 Object。听起来好像不可思议，因为一个 Object 可以说也是一个数据结构，这种数据结构是抽象虚拟的，而网络中传输的二进制流则是实际化的，这种抽象虚拟的对象该如何传递呢？不过这些已经不用我们操心了，SUN 公司提供了一种叫做序列化的技术。它能够把抽象的 Object 变为一个二进制流，可以存放于一个文件中，可通过网络进行传输。自然的，也有一种反序列化的技术，可以从刚才序列化后得到的二进制流再重新在 Java 虚拟机中组装为原来的对象。也许有些项目经验并不是很丰富的读者可能并不是很看重这套技术，但是这项技术是非常有用的。因为现在项目的开发规模越来越大，很多项目不再是运行在一台服务器上的，而是运行在一系列的服务器集群上。对于这些服务器集群，每个服务器都拥有各自至少一个 Java 虚拟机。如果把这些服务器看成一个个分散的服务器的话，那么业务在开发时所考虑的内容就会异常复杂，考虑的内容一多，自然也就很容易出错。但是如果把这些服务器集群只是看作一台单独的服务器，开发的时候也并不用在意多服务器这类的条件，序列化和反序列化这套技术就显得不可或缺了。通过这套技术，可实现在不同的 Java 虚拟机中共享对象的目的。这样看来，好像这些对象就都存在于一个单独的 Java 虚拟机中一样。开发人员可不必专注于软件如何编码来适应服务器集群，而是让序列化机制帮助在多个虚拟机中共享 Java 对象。

JNDI 则是在整个 Java 系统中（不一定只有一个 Java 虚拟机）实现存放管理所有资源的机制。可以通过查询名字来取得指定的资源，而不必考虑这个资源究竟是在哪个 Java 虚拟机上。打个比方，一般情况下，数据库信息都是存放在一个叫做 DataSource 对象里的，从 DataSource 可以取得 Connection 等重要信息。但是在一个分布式系统中（也就是

前面所说的服务器集群），DataSource 肯定就存放在某台主机上，其他主机都是从这台主机上取得。那么其他主机如何得知 DataSource 在哪台主机上呢，这就用到了 JNDI 技术。首先给 DataSource 命名，然后把名字以及 DataSource 的物理地址存储于 JNDI 中。当某个服务器需要用到 DataSource 的时候，便根据名字在 JNDI 中查找，JNDI 把物理地址返回给这个服务器，然后服务器从物理地址中取得 DataSource 并加以利用。

RMI 则利用了上述两种技术实现远程的控制调用。一个 RMI 调用分为两部分，分别为主动方和被动方。主动方的程序叫做 Stub，被动方的程序叫做 Skeleton。当 A 服务器（主动方）需要调用 B 服务器（被动方）中某个方法的时候，A 调用存放于 A 中的 Stub，而这个 Stub 则通过 JNDI 查找到 Skeleton 的所在。A 中把参数通过序列化传送到 B 中，在 B 中反序列化为从 A 中传递的参数的形式；然后 Skeleton 使用参数在 B 中进行业务处理，把处理完的结果通过序列化和反序列化的形式再传送给主动方。这就完成了一个典型的 RMI 操作。

9. EJB 和 JMS

EJB 一开始推出的目的就是为了针对大型企业级应用解决方案的。可以想象，当两个服务器之间互相通信操作的时候，处理互相通信操作的代码是必要的。但是这和企业级的业务逻辑并没有什么太大的关系，而且只要涉及到服务器之间的操作，就都需要这些互相操作的代码。所以我们可以专门把这些代码抽出来，这就是中间件的构想。不过在实现的时候要充分考虑到业务逻辑和共同代码的分离性，要准确知道哪些是业务逻辑，哪些是 Base。

EJB 在 1.0 以及 2.x 中主要由三部分构成，分别是实体 Bean（entity Bean），会话 Bean（session Bean），以及消息驱动 Bean（message driven Bean）。下面分别简单介绍一下。

（1）实体 Bean

它是用来对应数据库中的数据的，数据库中每个表在 Java 中可以对应一个 Class，Class 中的每个 Field 对应数据库中对应表的每个 Column，每张表中一条记录可以对应这个 Class 的一个 Object。我们可以在对一个 Java 的 Object 操作，其中 Object 中 Field 的值产生变化，则对应这个 Field 在数据库中的 Column 的值也发生变化。这样就实现了数据库与 Java 对象之间的一一映射。不过由于实体 Bean 过于复杂，操作起来比较麻烦，而且运行速度缓慢，因此备受批评。在 EJB 3.0 中对实体 Bean 这部分做了比较大的变动，借鉴了 Hibernate 的成功经验，改用 POJO 作为数据持久化的方法。

（2）会话 Bean

顾名思义，就是指它的生命周期比较短，一般就是一个用户会话的时长，一般就几分钟。而不像实体 Bean 那样有非常长的生命周期（一个实体 Bean 有可能在服务器中存在一年或者更长的时间）。不过不能因此就看轻会话 Bean 的功能和作用。会话 Bean 分为有状态的会话 Bean 和无状态的会话 Bean。无状态的会话 Bean 有点类似于 Java 的 Class 中的静态方法，这个 Bean 本身没有状态，只要给它一个参数，它就会运行产生固定的结果。比如一个计算乘法结果的会话 Bean 就是无状态的。当输入乘数与被乘数，它就会自动返回相乘后的结果，而不管你是什么人，现在是什么时间之类的状态。所以这种 Bean 一般在应用服务器中专门维护一个无状态会话 Bean 池，需要使用的时候，就从这个池里

面提取一个无状态会话 Bean，当用户使用完毕后，此无状态会话 Bean 便重新放回无状态会话 Bean 池。而有状态会话 Bean 则无法采用上述机制。有状态会话 Bean 的使用是采用钝化的方法来保证的，比如一个计算客户折扣的会话 Bean，由于每个客户之前买过什么东西是不一定的，所以折扣也是会变化的。这时候需要给每个客户专门定制一个会话 Bean，但是内存中也放不下那么多会话 Bean，这时应该怎么办呢？答案很简单，我们利用序列化机制，通过一些算法计算出那些会话 Bean 短时间内不再使用，便把这些会话 Bean 通过序列化机制序列化为二进制流储存在硬盘或者数据库中。等以后需要使用的时候再通过 Java 提供的反序列化机制，把这些二进制流恢复成有状态的会话 Bean，对其再进行利用。

（3）消息驱动 Bean

消息驱动 Bean（MDB）是 EJB 2.0 之后新增的一种 Bean。它不像前面两种 Bean 要求操作是同步的。从名字上看，感觉和 Windows 的消息驱动机制有相似之处，都是通过发送消息到一个消息队列，然后系统再从消息队列中取得需要的信息，再进行处理。这样，就不必要求操作即时同步了。自然，EJB 中也提供了相应的机制来保证声明必需运行的消息一定会被执行。消息驱动 Bean 跟 JMS 关系比较紧密，可以参考下面对 JMS 的简介部分增强对消息驱动 Bean 的理解。

（4）JMS

JMS（Java message service）是用来访问企业消息系统的 API，用来让整个系统中的 Java 应用程序进行消息交换，并且通过提供标准的产生、发送、接收消息的接口简化企业应用的开发。在 EJB 的消息驱动 Bean 中就采用了 JMS 的方式实现。JMS 通信主要采用两种方式，第一种是点对点的方式，也就是类似于通常大家聊天的 QQ 模式，有一个发送者，也有一个接收者。另外一种通信模式就是发布/订阅（publish/subscribe）模式，有点类似于订阅报纸的感觉，由一个服务者发布服务，类似于报社，有许多其他的用户在对这个服务者发出的服务进行定制，类似于我们到邮局订阅报纸；然后发出服务的人就发出消息，所有定制的人都会收到消息。两种通信模式的共同点都是可以异步发送消息，而最大的不同点只有一个，就是点对点方式只能有一个接收者，而发布/订阅方式则可以有多个接收者。

10. SOAP，Web Service 以及 SOA 服务

SOAP（simple object access protocol，简单对象访问协议）是在非集中、分布环境中交换信息的轻量级协议。它是基于 XML 协议的。SOAP 的传输方式主要以 HTTP 协议为主，不过也不局限于 HTTP 协议。SOAP 由于只是一个利用 HTTP 等通信协议和 XML 的技术，所以也被别人称为“没有任何发明的发明”。SOAP 允许任何类型的对象（或代码），在任何平台上，以任何一种语言相互通信。目前，已在多个平台上，以许多种语言实现了 SOAP。正是由于 SOAP 的这个特性，所有它在 Web Service 中有很大的用武之地。众所周知，在 Web 平台上，多种操作系统，多种语言编写的软件并存，当一个服务器发布服务的时候，通常只能与特定的软件，特定的客户段进行联系。比如，现今比较流行的网络游戏，一般有一到多个游戏服务器，这些服务器发布网络游戏服务器端服务，这些服务也通常只能与特定的游戏客户端通信，所以这些服务属于专有的，而不是一个通用

的服务。而 SOAP 的使用则是一种通用的服务，SOAP 利用 XML 来传递信息。HTTP 用于实现 SOAP 的 RPC 风格的传输，而 XML 是它的编码模式。SOAP 本身并不以任何平台和任何编程语言为载体。所以每一种平台，每一种语言只要编写适合解析 SOAP 的借口，就可以与 SOAP 进行通信，利用 SOAP 可以与另外一种平台，以及别的编程语言所发布的服务进行交互式通信。这也是实现现今很流行的 SOA 架构[①]的很重要的组成部分。

1.1.2 Java EE 的发展与展望

1. J2EE 的更名

J2EE 发展到 2005 年可以说是有了一个新的起点，在 2005 年 6 月的 SUN ONE 大会上，也就是 Java 诞生 10 周年的大会上，决定了把沿用了 7 年的 J2XE 中的“2”去掉，改为 Java XE。目前的 J2EE 1.4 名称并不改变，而以后的新的版本要变更，比如下一版本的企业版的 Java 名称改为 Java EE 5。

2. EJB 3.0 的出现

EJB 3.0 就要正式推出。EJB 3.0 相对于 EJB 2.x 来说变更很大，特别在数据持久化方面。众所周知，EJB 2.x 中的数据化持久技术采用的是 Entity Bean。虽然功能强大，但是过于繁琐，不易学习，开发工作量较大。为了编写一个 Entity Bean 要编写许多其他的东西，如果采取 CMP（容器管理持久化）的话还要进行额外的配置。如果你采取 BMP（Bean 管理持久化）技术，Entity Bean 中的代码则更为繁杂，而且由于 EJB 中过于繁琐的调用机制，所以普遍反映采用 Entity Bean 的执行效率比较低下。加之由于 Hibernate 的成功应用，Hibernate 的作者 Gavin King 进入 EJB 3.0 的专家组，他参考了 Hibernate 的经验后，EJB 3.0 决定采用 POJO 作为数据持久化方式。

3. SOA[①]的普及应用

在程序语言出现的最开始，首先是面向过程的编程，后来面向对象的编程（OOP）出现，于是类似的面向对象的设计（OOD）、面向对象的架构（OOA）相继如雨后春笋一样冒了出来。现在又多了一项，即面向服务的架构（service-oriented architecture，SOA）。

SOA 是一个组件模型。在应用程序中，我们可以按照功能划分不同的单元。我们称一个单元为一个服务。我们把这些服务分割开，中间采用定义好的接口进行连接。这些接口是采用中立的方式定义的，不依赖于任何特定的操作系统、硬件设备和编程语言。这种特征也被称为各种服务之间的松耦合。至此，大家就会想到刚才提及的 SOAP 了。是的，SOAP 就拥有担任这些接口的一个很好的条件。事实上，SOAP 也的确是 SOA 的一个重要实现。

SOA 其实也不算一个全新的概念，不过它是传统的面向对象编程的一个替代。传统的面向对象的编程是紧耦合的，这项技术已经存在 20 多年，在可预见的未来，还会继续存在好长一段时间。不过由于松耦合可以更加灵活地满足业务要求，而且当每个服务内

① SOA（service oriented arkitecture 面向服务的架构）。一种把整个大系统尽量解耦，成为相对独立的一个个小的模块（也就是我们所说的服务）。这样可以达到系统最大的可重复利用性以及可维护性。