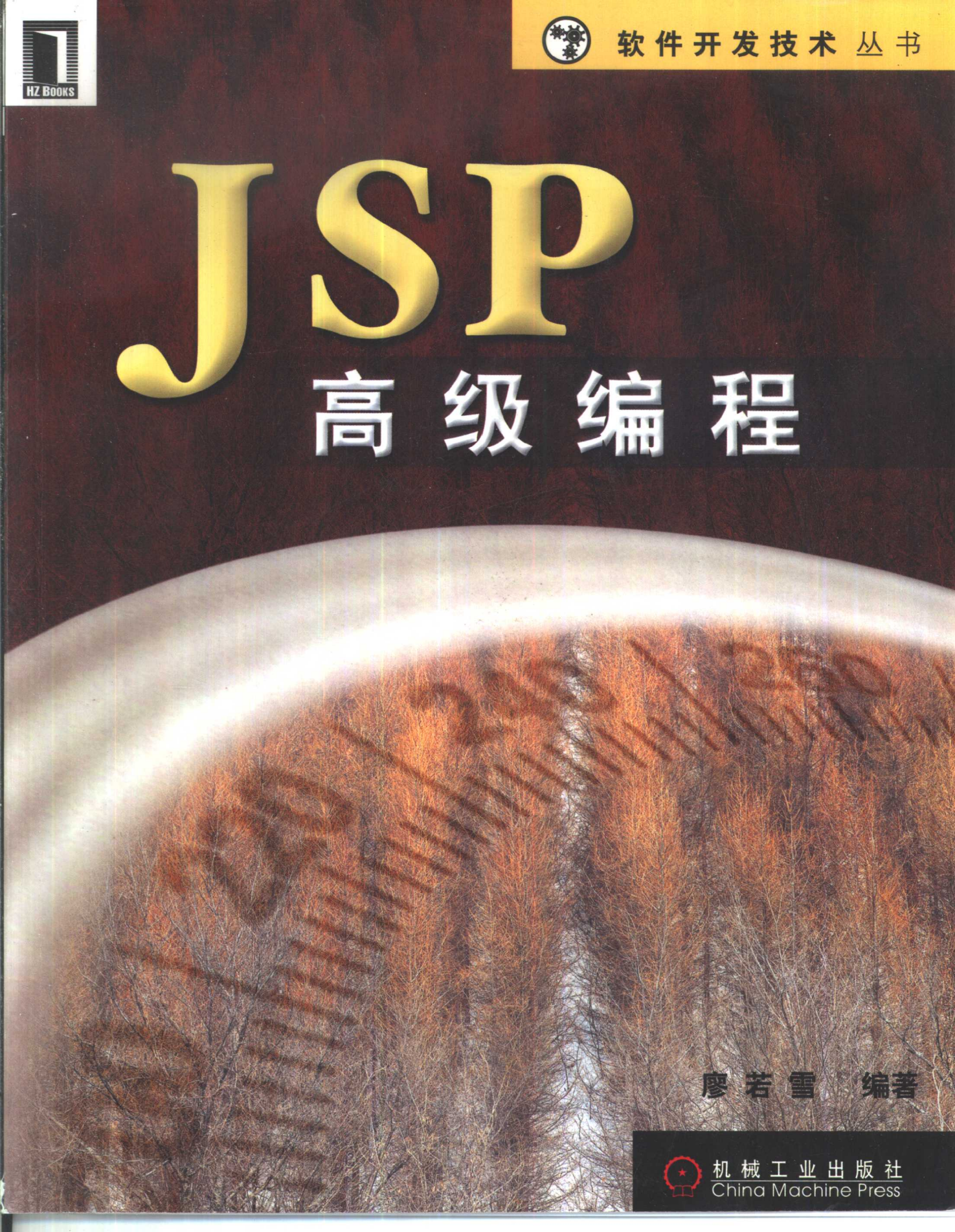
HZ BOOKS
软件开发技术丛书
JSP
高级编程
廖若雪 编著
机械工业出版社
China Machine Press

软件开发技术丛书

JSP 高级编程

廖若雪 编著

机械工业出版社
China Machine Press

JSP是一种如日中天的新型Internet/Intranet开发语言，可以在多种操作系统平台和多种Web服务器下使用。

本书从最基础的JSP开始，循序渐进地介绍了JSP 开发技术，并涵盖了许多高级主题，如需要在企业级Web应用中使用的特性——Enterprise JavaBeans、JDBC 2.0、数据库连接池和自定义标签库等。

本书既适合初学者阅读，也适合具有一定JSP基础的开发人员深入研究使用。

图书在版编目（CIP）数据

TP393.4/07

JSP高级编程 / 廖若雪编著. -北京：机械工业出版社，2001.3
（软件开发技术丛书）
ISBN 7-111-08685-6

I. J… II. 廖… III. 计算机网络－程序设计 IV. TP393

中国版本图书馆CIP数据核字（2001）第00688号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）
责任编辑：程代伟 刘立卿
北京昌平第二印刷厂印刷 · 新华书店北京发行所发行
2001年3月第1版 · 2001年6月第2次印刷
787mm × 1092 mm 1/16 · 28.25印张
印数：6 001-8 000册
定价：45.00元

前　言

JSP是SUN公司推出的一种新型的Internet/Intranet开发语言，和前一代Internet/Intranet开发语言(ASP、PHP)相比，JSP在以下几个方面有了重大的突破：

1) 通过JSP的扩展标签库和JavaBeans功能，网站逻辑和网站界面可以完美地分离。

2) 使用Enterprise JavaBeans，可以轻松地在JSP开发的Web中实现事务、安全、会话等企业级应用所需要的功能。

3) JDBC2.0提供了不同的数据库产品无关的数据库连接方式，更重要的是，数据库连接池提供了一种比普通的数据库连接方式效率高得多的连接方式。

JSP的语法和Java的基本上是相同的，有Java基础的读者可以很快学会如何使用JSP，而没有Java语言基础的读者，只要循序渐进地阅读本书，一样可以成为JSP编程的高手。

本书主要分为两个部分：

第一部分为JSP基础部分。通过这一部分的学习，读者可以掌握JSP的基本使用方法，学会如何使用JSP来开发一般的中、小型Web应用。这一部分使用常见的Apache Group的Tomcat作为JSP引擎的例子。

第二部分为JSP高级应用部分。这一部分主要讲述如何使用JSP进行大型Web应用的开发。为了方便读者学习，本书还专门讲述了SUN公司的J2SDKEE和BEA公司的Weblogic应用服务器的基本使用方法。

JSP可以在各种操作系统和各种Web服务器下使用，其代码基本上不需要任何改动就可以使用。本书为了适应大多数读者的情况，使用了Windows操作系统作为例子，具体的试验平台如下：

Windows 2000 Advanced Server

Apache 1.3.14

Internet Information Server 5.0

Tomcat 3.1

J2SDKEE 1.2

BEA Weblogic 5.1

除了上述平台，书中的代码还在如下平台上进行了测试：

Redhat Linux 6.1

Apache 1.3.12

Tomcat 3.1

BEA Weblogic 4.51

数据库系统主要使用了Microsoft SQL Server 7.0，部分代码使用了MySQL。

由于作者水平有限，本书难免有许多不尽人意之处。敬请广大读者和专家同行不吝赐教，批评指正。

作者的电子邮件地址为：lrxape @ 263.net
主页为：risosir.home.chinaren.com

廖若雪
2000.11

目　录

第二部分 高级JSP技术

第一部分 JSP 入 门

第1章 概 述

1.1 Java技术

Java是一种简单易用、完全面向对象、具有平台无关性且安全可靠的主要面向Internet的开发工具。自从1995年正式问世以来，Java的快速发展已经让整个Web世界发生了翻天覆地的变化。随着Java Servlet的推出，Java在电子商务方面开始崭露头角，最新的Java Server Page技术的推出，更是让Java成为基于Web的应用程序的首选开发工具。

要学习Java技术中的Java Server Page，Java基础是必不可少的。本书将在第2章为没有Java基础的读者简单讲解Java的基础语法和Java Beans等，它们是在学习JSP之前必须掌握的Java知识。这里，先回顾一下Java的发展历程，然后讲解几个后面将要用到的重要概念。

1.1.1 Java技术的发展

Java技术是由美国Sun公司倡导和推出的，Java技术包括Java语言和Java Media APIs、Security APIs、Management APIs、Java Applet、Java RMI 、JavaBeans、JavaOS、Java Servlet、JDBC、JNDI、Enterprise JavaBeans等，下面是Java技术的发展简述。

1990年，Sun公司由James Gosling领导的小组设计了一种平台独立的语言Oak，主要用于为各种家用电器编写程序。

1995年1月，Oak 被改名为Java，1995年5月23日，Sun公司在Sun World'95上正式发布Java和HotJava浏览器。

1995年8月至12月，Netscape公司、Oracle公司、Borland公司、SGI公司、Adobe公司、IBM公司、AT&T公司、Intel公司获得Java许可证。

1996年1月，Sun公司宣布成立新的业务部门——JavaSoft部，以开发、销售并支持基于Java技术的产品，由Alan Baratz任总裁。同时推出Java开发工具包JDK(Java Development Kit) 1.0，为开发人员提供编制Java应用软件所需的工具。

1996年2月，Sun发布Java芯片系列，包括picoJava、MicroJava和UltraJava，并推出Java数据库连接JDBC (Java Database Connectivity)。

1996年3月，Sun公司推出Java WorkShop。

1996年4月，Microsoft公司、SCO公司、苹果电脑公司(Apple)、NEC公司等获得Java许可证。Sun公司宣布苹果电脑、HP、日立、IBM、微软、Novell、SGI、SCO、Tamdem等公司将把Java平台嵌入到其操作系统中。

1996年5月，HP公司、Sybase公司获得Java许可证。北方电讯公司宣布把Java技术和Java微处理器应用到其下一代电话机中的计划。5月29日，Sun公司在旧金山举行第一届JavaOne世界Java开发者大会，业界人士踊跃参加。Sun公司在大会上推出一系列Java平台新技术。

1996年8月，Java WorkShop成为Sun通过互联网提供的第一个产品。

1996年9月，Addison-Wesley和Sun推出Java虚拟机规范和Java类库。

1996年10月，德州仪器等公司获得Java许可证。Sun提前完成JavaBeans规范并发布。发布第一个Java JIT(Just-In-Time)编译器，并打算在Java WorkShop和Solaris操作系统中加入JIT。10月29日，Sun发布Java企业计算技术，包括JavaStation网络计算机、65家公司发布的 85个Java产品和应用、7个新的Java培训课程和Java咨询服务、基于Java的Solstice互联网邮件软件、新的Java开发者支持服务、演示HotJava Views、Java Tutor、完成Java Card API等。Sun宣布完成Java Card API规范，这是智能卡使用的第一个开放API。Java Card规范将把Java能力赋予全世界的亿万张智能卡。

1996年11月，IBM公司获得JavaOS和HotJava许可证。Novell公司获得Java WorkShop许可证。Sun和IBM宣布双方就提供Java化的商业解决方案达成一项广泛协议，IBM同意建立第一个 Java检验中心。

1996年12月，Xerox等公司获得Java或JavaOS许可证。Sun发布JDK 1.1、Java商贸工具包、JavaBeans开发包及一系列Java APIs。推出一个新的Java Server产品系列，其中包括Java Web Server、Java NC Server和 Java Server Toolkit。Sun发布100%纯Java计划，得到百家公司的支持。

1997年1月，SAS等公司获得Java许可证。Sun交付完善的JavaBeans开发包，这是在确定其规范后不到8个月内完成的。

1997年2月，Sun和ARM公司宣布同意使JavaOS能运行在ARM公司的RISC处理器架构上。Informix公司宣布在其Universal Server和其他数据库产品上支持JDK 1.1。Netscape公司宣布其Netscape Communicator支持所有Java化的应用软件和核心APIs。

1997年3月，HP公司获得Java WorkShop许可证，用于HP-UX操作系统。西门子AG公司等获得Java许可证。日立半导体公司、Informix公司等获得JavaOS许可证。Novell公司获得Java Studio许可证。Sun发售JavaOS 1.0操作系统，这是一种在微处理器上运行Java环境的最小、最快的方法，提供给Sun的JavaOS许可证持有者使用。Sun发售HotJava Browser 1.0，这是一种Java浏览环境，可以方便地按剪裁来编制专用的信息应用软件，如客户自助台和打上公司牌号的网络应用软件。Sun推出JDK 1.1.1。

1999年6月，Sun 发布JDK 1.3和Java Web Server 2.0。

1.1.2 JavaBeans

什么是JavaBeans？JavaBeans就是Java的可重用组件技术。ASP通过COM来扩充复杂的功能，如文件上载、发送email以及将业务处理或复杂计算分离出来，成为独立可重复利用的模块。JSP通过JavaBeans实现了同样的功能扩充。JSP 对于在 Web 应用中集成 JavaBeans 组件提供了完善的支持。这种支持不仅能缩短开发时间（可以直接利用经测试和可信任的已有组件，避免了重复开发），也为 JSP 应用带来了更多的可伸缩性。JavaBeans 组件可以用来执行复杂的计算任务，

或负责与数据库的交互以及数据提取等。

在实际的JSP开发过程中，读者将会发现，和传统的ASP或PHP页面相比，JSP页面将会是非常简洁的。由于JavaBeans开发起来简单，又可以利用Java语言的强大功能，许多动态页面处理过程实际上被封装到了JavaBeans中。

1.1.3 JDBC

JDBC是用于执行SQL语句的Java应用程序接口，由一组用Java语言编写的类与接口组成，在JSP中将使用JDBC来访问数据库。JDBC是一种规范，它让各数据库厂商为Java程序员提供标准的数据库访问类和接口，这样就使得独立于DBMS的Java应用程序的开发工具和产品成为可能。一般的Java开发工具都带有JDBC-ODBC桥驱动程序，这样，只要是能够使用ODBC访问的数据库系统，也就能够使用JDBC访问了。有趣的是，不同于ODBC是Open Database Connectivity的简称，JDBC并不是Java Database Connecivity的简称，而是SUN的注册商标，至少官方说法是这样的。

1.1.4 J2EE

电子商务和信息技术的快速发展以及对它们的需求给应用程序开发人员带来了新的压力。必须以比以前更少的金钱、更少的资源来更快地设计、开发企业应用程序。

为了降低成本，并加快企业应用程序的设计和开发，J2EE 平台提供了一个基于组件的方法，来设计、开发、装配及部署企业应用程序。J2EE 平台提供了多层的分布式的应用模型、组件再用、一致化的安全模型以及灵活的事务控制。您不仅可以用比以前更快的速度向市场推出创造性的客户解决方案，而且您的平台独立的、基于组件的J2EE 解决方案不会被束缚在任何一个厂商的产品和API 上。

1. J2EE 规范定义了以下种类的组件

- 应用客户组件。
- Enterprise JavaBeans 组件。
- Servlet及JavaServer Pages（JSP 页面）组件（也被称作Web 组件）。
- Applet。

一个多层的分布式的应用模型意味着应用逻辑被根据功能划分成组件，并且可以在同一个服务器或不同的服务器上安装组成J2EE 应用的这些不同的组件。一个应用组件应被安装在什么地方，取决于该应用组件属于该多层的J2EE 环境中的哪一层。这些层是客户层、Web层 、 业务层及企业信息系统层（EIS ）等。

(1) 客户层

J2EE 应用可以是基于Web 的，也可以是不基于Web 的。在一个基于Web 的J2EE 应用中，用户的浏览器在客户层中运行，并从一个Web服务器下载Web 层中的静态HTML 页面或由JSP或Servlet 生成的动态HTML 页面。在一个不基于Web 的J2EE 应用程序中，一个独立客户程序不运行在一个HTML 页面中，而是运行在其他一些基于网络的系统（比如手持设备或汽车电话）中，Applet 程序在客户层中运行，并在不经过Web 层的情况下访问Enterprise Beans。这个不基

于Web 的客户层可能也包括一个JavaBeans 类来管理用户输入，并将该输入发送到在企业层中运行的Enterprise Beans类来进行处理。根据J2EE 规范，JavaBeans 类不被视为组件。

为J2EE 平台编写的JavaBeans 类有实例变量和用于访问实例变量中的数据的“get 和set 方法”。以这种方式使用的JavaBeans 类在设计和实现上通常都是简单的，但是它们必须符合JavaBeans 规范中列出的命名和设计约定。

(2) Web 层

J2EE Web 组件可以由JSP 页面、基于Web 的Applet以及显示HTML 页面的Servlet组成。调用Servlet或者JSP 页面的HTML 页面在应用程序组装时与Web 组件打包在一起。就像客户层一样，Web 层可能包括一个JavaBeans 类来管理用户输入，并将输入发送到在业务层中运行的Enterprise Beans 类来进行处理。运行在客户层的Web 组件依赖容器来支持诸如客户请求和响应及Enterprise Bean 查询等。

(3) 业务层

作为解决或满足某个特定业务领域（比如银行、零售或金融业）需要的逻辑的业务代码由运行在业务层的Enterprise Beans 来执行。一个Enterprise Beans 从客户程序处接收数据，对数据进行处理（如果需要），再将数据发送到企业信息系统层存储起来。一个Enterprise Beans 还从存储中检索数据，并将数据送回客户程序。运行在业务层的Enterprise Beans 依赖于容器来为诸如事务、生命期、状态管理、多线程及资源存储池等提供通常都非常复杂的系统级代码。业务层经常被称作Enterprise JavaBeans （EJB ）层。业务层和Web 层一起构成了3 层J2EE应用的中间层，而其他两层是客户层和企业信息系统层。

(4) 企业信息系统层

企业信息系统层运行企业信息系统软件，这层包括企业基础设施系统，例如企业资源计划（ERP）、大型机事务处理（mainframe transactionprocessing ）、数据库系统及其他遗留信息系统（legacy informationsystems ）。J2EE 应用组件因为某种原因(例如访问数据库) 可能需要访问企业信息系统。J2EE 平台的未来版本将支持Connector 架构，该架构是将J2EE 平台连接到企业信息系统上的一个标准API。

(5) 查询服务

因为一个J2EE 应用程序的组件是单独运行的，并且往往在不同的设备上运行，因此，需要一种能让客户层和Web 层代码查询并引用其他代码和资源的方法。客户层和Web 层代码使用Java 命名和目录接口（JNDI ）来查询用户定义的对象（例如Enterprise Beans ）、环境条目（例如一个数据库驱动器的位置）、企业信息系统层中用于查找资源的JDBC DataSource对象，以及消息连接。

(6) 安全和事务管理

诸如安全和事务管理这样的应用行为可以在部署时在Web 和Enterprise Beans 组件上进行配置。这个特征将应用逻辑从可能随装配而变化的配置设定中分开了。

J2EE 安全模型允许配置一个Web 或Enterprise Beans 组件，使系统资源只能由授权的用户访问。例如，一个Web 组件可以被配置成提示输入用户名和密码。一个Enterprise Beans 组件可以被配置成只让特定团体中的成员调用其某些方法。或者，一个Servlet 组件可以被配置成让某个

组织中的所有人都能访问其某些方法，同时只让该组织中的某些享有特权的人访问其中一些方法。同样是该Servlet 组件，可以针对另外一个环境而被配置成让每个人都能访问其所有方法，或者仅让选定的少数人访问其所有方法。

J2EE 事务模型使得能够在部署时定义构成一个单一事务的方法之间的关系，以使一个事务中的所有方法被处理成一个单一的单元。这是我们所希望的，因为一个事务是一系列步骤，这些步骤要么全部完成，要么全部取消。例如，一个Enterprise Beans 可能有一组方法，使我们可以通过从第一个账户借出并存入第二个账户的方式而将钱从第一个账户转移到第二个账户。我们希望全部的操作被作为一个单元对待，这样，如果在借出之后存入之前发生了故障，该借出操作被取消。事务属性是在装配期间定义在一个组件上的。这使得能将来自多个应用组件的方法归到一个事务中，这说明，我们可以轻易变更一个J2EE 应用程序中的应用组件，并重新指定事务属性，而不必改变代码或重新编译。在设计应用组件时，要记住，尽管Enterprise Beans 有一个可使应用组件的容器自动启动多步事务的机制，但是Applet 和应用的客户容器可能并不支持这一点。然而，Applet 和应用客户容器总是能够调用支持这一点的一个Enterprise Beans。还应当注意，JSP 页面和Servlet 没有被设计成是事务的，它们通常应当将事务工作交给一个Enterprise Beans 来完成。然而，如果事务工作在一个JSP 页面或Servlet 中是必须的，那么此种工作也应当是非常有限的。

(7) 可重用应用组件

J2EE 组件（Applet 、应用的客户、Enterprise Beans 、JSP 页面及Servlet ）都被打包成模块，并以Java Archive （JAR ）文件的形式交付。一个模块由相关的组件、相关的文件及描述如何配置组件的配置描述文件组成。例如，在组装过程中，一个HTML 页面和Servlet 被打包进一个模块之中，该模块包含HTML文件、Servlet 组件及相关的配置描述文件，并以一个Web ARchive（WAR ）文件的形式交付，该WAR 文件是一个带.war 扩展名的标准JAR 文件。模块的使用使得利用相同组件中的某些组件来组装不同的J2EE 应用程序成为可能。例如，一个J2EE 应用程序的Web 版可能有一个Enterprise Beans 组件，还有一个JSP 页面组件。该Enterprise Beans 组件可以与一个应用客户组件结合，以生成该应用程序的非Web 版本。这不需要进行额外的编码，只是一个装配和部署的问题。并且，可重用组件使得将应用开发和部署过程划分成由不同的角色来完成成为可能，这样，不同的人或者公司就能完成封装和部署过程的不同部分。

2. J2EE 平台定义了如下角色

(1) J2EE 产品提供商

设计并使J2EE 平台、API 和在J2EE 规范中定义的其他特征能被其他公司或人购得的公司。

(2) 应用组件提供商

创建用于J2EE 应用程序的Web 组件、Enterprise Beans 组件、Applet 或应用客户程序的公司或个人。在装配过程中，应用组件文件、接口及类被打包进一个JAR 文件中。

(3) 应用程序装配商

从组件提供商获得应用组件JAR 文件，并将它们组装成一个J2EE 应用的Enterprise Archive（EAR）文件的公司或个人，这种文件是一个带.ear扩展名的标准文件。应用装配商提供与该应用程序相关的整体信息，并使用验证工具来检验EAR 文件的内容是正确的。组装和部署信息存

储在一个基于文本的配置描述文件中，此种文件使用XML 标记来标记该文本。应用装配商可以使用一个能通过交互式选择来正确添加XML 标记的装配和配置工具来编辑该配置描述文件。

(4) 部署商

部署（deploy）J2EE 应用程序的公司或个人。其职责包括设定事务控制、安全属性，并根据应用组件提供商提供的指示来标明一个Enterprise Beans 是自己处理自身的存储，还是由一个容器来处理等。部署涉及配置和安装。在配置过程中，部署商遵循应用组件提供商提供的指示来解决外部依赖问题，定义安全设定，以及分配事务属性。在安装过程中，部署商将应用组件安装到服务器上，并生成容器特定的类和接口。

(5) 系统管理员

配置并管理运行J2EE 应用程序的计算环境和网络基础设施，并监督运行环境的人员。

(6) 工具提供商

生产被组件提供商、装配商及部署商使用的用于进行开发、组装和打包的工具的公司或个人。

(7) 设计用户界面和引擎

在为J2EE 应用程序设计用户界面和后端引擎时，需要决定让该程序是基于Web，还是不基于Web。在做出这个决定时，我们可能希望考虑平台配置、下载速度、安全、网络流量和网络服务。

例如，包含有用户界面并且经常被大量用户访问的一个Applet 可能需要花很长的时间才能被下载下来，这让用户沮丧。然而，如果知道该Applet要运行在一个公司的内部网内的受控环境中，那么，在这种情况下，该Applet 将拥有一个完全可接受的下载速度。另一个考虑是，繁重的处理应当在哪里执行。例如，如果客户程序在一个蜂窝电话或呼机中执行，服务器应当完成尽量多的计算和数据处理，而客户程序只应显示结果就可以了。然而，设计在一个强大的台式机平台上运行的大型财务分析系统则应当在客户机上完成其复杂计算。应用的客户程序和Applet用户界面通常都是用Swing API 创建的，该API 可从标准版Java2平台中获得。Swing API 提供了一整套GUI 组件（表格、树形结构、按钮等），这些组件可以被用来实现一种比用一个典型的HTML 页面所能实现的更为交互的体验。Swing 也支持HTML 文本组件，这个组件可以被用来显示来自一个服务器的响应。客户程序可以直接访问Enterprise Beans 层或企业信息系统层。但应谨慎实现这种程序。绕过EJB 层的程序可以使用JDBC API 来访问一个关系型数据库，但应被限制于对数据库表格进行维护等管理任务上。

(8) 设计基于Web 的应用程序

基于Web 的应用程序是基于浏览器的，并且，如果它们运行在Internet上，就可能被全世界的人访问。当设计一个基于Web 的应用程序时，不仅需要决定用什么来处理内容和应用逻辑（HTML 、XML 、JSP 页面及Servlet），而且还应当考虑使该应用程序国际化。一个国际化的基于Web 的应用程序向用户提供了选择一种语言，然后根据该选定语言加载应用的正文的方式。对被支持的每种语言而言，应用正文都被存储在一个外部文件中，并且与另外一个文件的关键词相对应。应用代码使用这些关键词及选定的语言来加载正确的文本。国际化API 还提供类来根据选定的语言格式化日期和金钱。一旦制订了使应用程序国际化的细节，就可以决定用什么

来实现它了。总的来说，一个基于Web 的应用程序使用HTML 来显示数据; 用XML来定义数据以使其可被另一个程序读取并处理; 使用JSP 页面或Servlet来管理用户与业务层或存储层之间的数据流。

可以在J2EE 平台上实现的基于Web 的应用程序有四种。从简单到复杂排列，它们是：

- 基本HTML。
- 带基本JSP 页面或Servlet 的HTML。
- 带JavaBeans 类的JSP页面。
- 将应用逻辑根据功能划分成区域的高度结构化的应用。

当设计一个基于Web 的应用程序时，需要决定用什么来建立它。如果是从建立一个简单的应用程序开始着手，并且以后会给该应用程序添加功能，那么，设计就应当适应今后发展的需要。

(9) 模型、视图和控制器架构

在基于组件的J2EE 平台充分内置了灵活性的情况下，剩下的问题可能是如何组织应用程序以实现简单高效的应用程序升级和维护，以及如何让不懂程序代码的人员避开程序数据。答案就在模型、视图和控制器架构（MVC ）的使用之中。MVC 这样的架构是一个描述重现的问题及其解决方案的设计范式，但每次问题重现时，解决方案都不会完全相同。

MVC 设计范式包括三种对象：模型（model ）提供应用业务逻辑（Enterprise Beans 类）；视图（view ）则是其在屏幕上的显示（HTML 页面、JSP 页面、Swing GUI ）；控制器则是Servlet 、JavaBeans 或SessionBeans 类，它用于管理用户与视图发生的交互。我们可以将控制器想像成处在视图和数据之间，对视图如何与模型交互进行管理。通过使视图完全独立于控制器和模型，就可以轻松替换前端客户程序。并且，通过将控制器和模型代码保持在视图之外，那些不理解这些代码的人员就不能改变他们不应改变的东西。

将控制器和模型分开就可以在不影响模型的情况下改变控制器，也可以在不影响控制器的情况下改变模型。例如，如果应用的前端是一个HTML 页面，HTML 专家就可以更新它。如果使用一个JSP页面，将控制器的代码放到一个JavaBeans 或SessionBeans 类中，或使用动作标记（action tag），这样，JSP 页面就仅包含JSP 代码了。

本书将在第二部分讲解如何使用J2EE来建立企业级的Web应用。

1.1.5 EJB

EJB就是前面说的Enterprise JavaBeans。EJB上层的分布式应用程序是基于对象组件模型的，低层的事务服务使用了API技术。EJB技术简化了用JAVA语言编写的企业应用系统的开发、配置和执行。EJB的体系结构规范由Sun Microsystems公司制定。

EJB技术定义了一组可重用的组件：Enterprise Beans。可以利用这些组件像搭积木一样建立分布式应用程序。当你把代码写好之后，这些组件就被组合到特定的文件中去。每个文件有一个或多个Enterprise Beans，在加上一些配置参数。最后，这些Enterprise Beans被配置到一个装了EJB容器的平台上。客户能够通过这些Beans的Home接口，定位到某个Beans，并产生这个Beans的一个实例。这样，客户就能够调用Beans的应用方法和远程接口。

EJB服务器作为容器和低层平台的桥梁，管理着EJB容器和函数。它向EJB容器提供了访问系统服务的能力。例如：数据库的管理和事务的管理，或者其他的Enterprise的应用服务器。

J2EE 应用程序中的Enterprise Beans

当编写管理特定业务功能（比如追踪雇员资料或进行复杂财务计算）的J2EE 应用程序时，请将完成这些任务的业务逻辑放置在EJB 层的Enterprise Beans 中。通过这种方式，就可以使代码集中在解决手边的业务问题，而利用Enterprise Beans 容器来支持低层服务，比如状态管理、事务管理、线程管理、远程数据访问和安全等。将业务逻辑与低层系统逻辑分开意味着容器可以在运行时创建和管理Enterprise Beans。按照规范编写的任何Enterprise Beans，都可以根据其在一个特定的J2EE 应用程序中将被如何使用，来对其事务管理或安全属性进行配置，并可以被部署到任何一个与规范兼容的容器中。可重用组件使不必改变和重新编译Enterprise Beans 代码成为可能。一个Enterprise Beans 由接口和类组成。客户程序通过Enterprise Beans的Home 和远程接口来访问Enterprise Beans 的方法。Home 接口提供了创建、删除和定位Enterprise Beans 的方法，而远程接口则提供了业务方法。在部署时，容器由这些接口来创建类，使客户能够创建、删除、定位或调用位于Enterprise Beans 上的业务方法。Enterprise Beans 类提供了业务方法、创建方法和查询方法的实现。如果Enterprise Beans 管理它自己的持久性的话，还为其生命期方法提供了实现。有两种Enterprise Beans ： Entity Beans 和Session Beans。

一个Session Beans 代表与客户程序的一个短暂会话，而且可能执行数据库读写操作。一个Session Beans 可能会自己调用JDBC，或者它可能使用Entity Beans来完成此种调用。在后面这种情况下，这个Session Beans 是该Entity Beans 的客户。一个Session Beans 的域包含会话状态，而且是短暂的。如果服务器或者客户程序崩溃，该Session Beans 就丢失了。这种模式通常被用于像PL/SQL 这样的数据库程序设计语言上。

一个Entity Beans 代表一个数据库中的数据及作用于该数据的方法。在一个关系型数据库中的雇员信息表中，每一行都有一个Beans 来代表。Entity Beans 是事务的，并且是长寿命的。只要数据留在数据库中，Entity Beans 就存在。这个模式可以被很容易地用于关系型数据库，而不仅仅限于对象数据库。

Session Beans 可以是有状态的，也可以是无状态的。一个有状态的Session Beans 包含代表客户程序的会话状态。该会话状态是该Session Beans实例的域值加上这些域值所引用到的所有对象。有状态Session Beans 并不代表在一个持久数据存储中的数据，但是，它可以代表客户程序访问和更新数据。无状态Session Beans 没有用于某个特定客户程序的任何状态信息。它们通常被用于提供不保持任何特定状态的服务器端行为。无状态Session Beans 要求更少的系统资源。一个提供一种一般服务，或用于表示被存储的数据的一个被共享的视图的业务对象是无状态Session Bean的一个例子。

因为Enterprise Beans 占用可观的系统资源和带宽，你可能希望将某些业务对象构造成数据访问对象或值对象。数据访问对象完成诸如代表客户程序访问数据库等工作。值对象用于代表容纳数据字段并提供简单的“get 和set ”方法来访问这些数据的一个结构。另外，可以将程序构造成使用Enterprise Beans 在客户和EJB 层的其他部分之间承担通信的任务。

对于一个使用容器管理的持久性来访问关系型数据库的Enterprise Beans，并不要求在Beans