WANG LUO BIAN CHENG BIAO ZHUN JIAO CHENG

J2EE

网络编程标准教程

WANG LUO BIAN CHENG BIAO ZHUN JIAO CHENG

主编 田雪松

上海科学普及出版社

J2EE网络编程标准教程

主　编　田雪松

上海科学普及出版社

图书在版编目（CIP）数据

J2EE 网络编程标准教程 / 田雪松主编. —上海：上海科学普及出版社，2003.12
ISBN 7-5427-2620-X

Ⅰ.J… Ⅱ.田… Ⅲ.JAVA 语言—程序设计—教材
Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2003）第 104708 号

策　　划：铭　政
责任编辑：徐丽萍

J2EE 网络编程标准教程

主　　编：田雪松
出版发行：上海科学普及出版社（上海中山北路 832 号　邮政编码 200070）
经　　销：各地新华书店
印　　刷：北京市燕山印刷厂
开　　本：787×1092　1/16
印　　张：20
字　　数：502 千字
版　　次：2004 年 1 月第 1 版　　2004 年 1 月第 1 次印刷
定　　价：26.00 元
书　　号：ISBN 7-5427-2620-X / TP·499

内 容 提 要

本书是一本全面、系统、深入阐述 J2EE 技术的最新教材和科技参考书。全书在详细讲解 J2EE 体系结构的基础上，对 J2EE 涉及的几项关键技术也进行了深入的探讨，包括 XML、RMI、JNDI、JDBC、EJB、Servlet、JSP 和 JSTL 等。在讲解这些技术的同时，还列举了大量的应用实例，使读者可以在实践操作中迅速掌握 J2EE 的关键技术。书中所列举的实例均由 JBuilder 工具创建，使用的 J2EE 服务器为 WebLogic。

为了保证本书讲解内容的权威性与先进性，本书所有内容大部分都与 J2EE 规范保持一致。书中还加入了最新引入 J2EE 规范的 EJB 查询语言、JSP 标记文件等内容，是学习 J2EE 新技术不可多得的参考资料。

本书既适合于有一定 Java 语言基础的 J2EE 初学者，又可以作为 J2EE 程序员的参考资料。没有 Java 语言基础的初学者，也可以结合其他 Java 语言教材一起使用本书。

前　言

自从 Sun 公司在 1996 年正式发布 Java 1.0 以来，Java 在业界的影响变得越来越大，尤其是在 1999 年 Sun 公司推出三个版本的 Java 2 平台后，Java 的应用更是深入到了各行各业之中。其中，尤为引人注目的是 Java 2 平台的企业版，也就是本书将要介绍的 J2EE，由于其自身所具有的优点，目前已经广泛地应用到基于 Web 的各种大型应用系统中。

由于 J2EE 具有良好的应用前景，目前无论是在国内还是在国外都兴起了一股学习 J2EE 的热潮。但是，国内介绍 J2EE 技术的图书并不多，仅有的几本也大多是国外相关专著的译本。本书就是在这种情况下诞生的，它是一本基于 J2EE 规范的、全面介绍 J2EE 技术的专业图书。

全书共分 9 章，是按照 J2EE 的体系结构组织编排的。其中：

第 1 章介绍了 J2EE 的基础知识，包括 J2EE 的概念和体系结构等内容，同时还对 EJB、JSP 的技术基础 XML、RMI 和 JNDI 进行了简要的介绍；第 2 章介绍了开发 J2EE 应用系统所使用的工具 JBuilder，以及 J2EE 服务器 WebLogic 的使用；第 3 章介绍 J2EE 体系结构的最底层，即 EIS 层，其中，着重讲解了 JDBC 技术；第 4 章到第 6 章，介绍的是 J2EE 体系结构的业务层，这一部分是本书的核心内容，主要讲解了 EJB 技术；第 7 章到第 9 章介绍的是 J2EE 的表述层，主要讲解了 Servlet 技术和 JSP 技术。全书按照 J2EE 的体系结构，从 EIS 层到表述层，层层展开，对读者理解 J2EE 十分有利。

本书所有实例都是使用 JBuilder 开发出来的，使用的 J2EE 服务器是 WebLogic。这两个工具在实际的 J2EE 开发中是较为流行的一种组合，这也十分有利于读者将 J2EE 技术应用到实际工作中。

本书由田雪松主编，参加本书编写和制作的还有于清源、赵福杰、任立功、崔慧勇、王文增、杨艳秋、李建慧等。此外，在本书的编写过程中，也得到了许多 J2EE 专家的指点，在此一并表示感谢。由于编者水平有限，书中疏漏之处在所难免，敬请广大读者批评指正。联系网址：http://www.china-ebooks.com。

编　者

2003 年 11 月

目　录

目录

目录

第 1 章 J2EE 基础知识

近年来，随着计算机与通信技术的迅猛发展，基于网络的大型应用系统也变得越来越复杂。这些系统往往是运行在异构的硬件平台上，并且需要通过网络进行数据交换。开发这样的系统，不仅要求系统开发人员对系统中的各个环节十分了解，还要求他们必须能够处理通信过程中出现的与硬件相关的许多问题。这就大大增加了系统开发的难度，降低了软件的开发效率。J2EE 正是在这种情况下诞生的，由于它能很好地解决大型应用系统中所面临的各种问题，所以受到人们越来越多的重视。

J2EE 的核心思想有两个：一个是分层的思想，另一个就是组件的思想。软件分层并不是在 J2EE 中才提出的概念，早在人们提出软件工程思想时分层的概念就已经广为人知了。J2EE 的高明之处就是在于它把层分得更细、更科学。J2EE 将一个系统按照功能分成了四个独立的层，即数据层（EIS 层）、业务层、表述层和客户层。每一层又定义了相应的组件开发规范（数据层除外），通过组件来实现层的功能。企业在开发应用系统时，可以根据系统的需求选择其中的几层进行组件开发，然后再将这些组件组合起来就可以实现系统的功能了。软件分层有利于软件开发的分工，使大型系统中所需的技术相互隔离。某一层的组件开发人员只需要掌握本层的组件技术就可以了，而不必关心其他层的技术内容。

不仅如此，为了提高组件的可重用性，人们又对 J2EE 的业务层、表述层和客户层进行了第二次分层。这一次的分层将每一层又分成了三个相互依赖的部分，即服务器、容器和组件。组件运行于容器之上，而容器运行于服务器之上，服务器又运行在一个特定的操作系统之上。所以，这里的分层与整个系统的分层是不同的。整个系统的层与层之间是相互独立的，最后通过接口相互连接实现整个系统的功能，而服务器、容器和组件则是相关的，它们共同协作实现层的功能。在实际应用中，服务器与容器常常是结合在一起的，它们向组件提供一些底层的服务，使组件可以彻底的和平台隔离开。组件不再需要处理与平台相关的细节问题，而只包含与业务相关的逻辑内容，从而提高了组件的可移植性和可重用性。现在，市场上已经有很多这样的服务器出现了，例如，WebLogic、WebSphere、Tomcat、JBoss 等。所以在使用 J2EE 进行开发时，程序员只要根据需求开发出实现业务逻辑的组件，然后再将它们部署到这些服务器上就可以了。

学习 J2EE 一定要掌握上面所说的两个思想，这样才不会对其中的技术感到迷惑。读者可能会对前面讲到的一些概念感觉比较生疏，下面就来介绍这些基本概念。除此之外，还将介绍 J2EE 的部分核心技术。

1.1 J2EE 概述

J2EE 是 Sun 公司在 Java 技术的基础上提出的企业级应用解决方案，因此，要想了解什么是 J2EE 就必须知道什么是 Java。大部分 Java 初学者可能会认为 Java 和 C++一样，仅仅是

一种语言而已，但实际上这只是一种狭义的理解。从目前的发展趋势来看，Java 已经成为一门十分庞杂的技术体系，这个技术体系是以 Java 语言为核心，此外，还包括了 Java Applet、RMI-IIOP、Java IDL/CORBA、JavaBeans、Servlet、JSP、JSTL、JDBC、JNDI、EJB、JavaMail 等，而 J2EE 正是在 Java 语言的基础上整合了这些关键技术而形成的一个新平台。为了帮助理解 J2EE 的含义，首先来了解一下 Java 和 J2EE 的历史。

1.1.1 J2EE 的发展简史

讲 J2EE 的历史，就不能不提 Sun 公司。Sun 公司的名称是 Stanford University Network 的简写，中文的意思就是“斯坦福大学网络公司”。Sun 公司起初是由包括 Scott McNealy 在内的四名在斯坦福大学和加州大学 Berkeley 分校的研究生在 1982 年 2 月正式注册创建的。Sun 公司最初是以工作站的设计制作为业务重点，6 个月后便开始创收盈利。目前，Sun 公司在全球的雇员人数达 2 万 6 千余人，而 Scott McNealy 也成为 Sun 公司的首席执行官。

1990 年，为了开拓消费类电子产品市场，Sun 公司开始了一个 Green 项目，开发用于电器中的软件。一开始，他们采用 C++语言开发，但是 C++语言自身存在的一些问题使它并不适合电器软件开发。于是该项目的领导人 James Gosling 便在 C++语言的基础上创建了一种全新的语言叫 Oak（橡树），这就是 Java 语言的前身。

到了 1994 年，Oak 成了一门相当成熟的语言，而此时网络技术也已经如火如荼地发展起来。Gosling 意识到 Oak 非常适合 Internet 编程，于是在当年秋天用 Oak 开发了一个早期的 Web 浏览器，即 WebRunner（HotJava 的前身）。这个系统展示了 Oak 的广阔市场前景，在业内引起了巨大的轰动。

由于 Oak 已是 Sun 公司另一个语言的注册商标，所以 Oak 在 1995 年正式更名为 Java。Java 是太平洋上一个盛产咖啡的岛屿的名称，因此许多 Java 语言文档中都会有咖啡的图标。1996 年 1 月，Sun 公司正式发布了 Java 1.0，1998 年夏末又推出了 Java 2.0，从此，Java 就走上了发展的坦途。1999 年，为了将 Java 2 的应用拓展到各个领域中，Sun 公司推出了三个版本的 Java 2 平台，这就是 J2ME、J2SE 和 J2EE，J2EE 由此诞生了。

1.1.2 Java 2 平台版本

通过上面的介绍可以知道，J2EE 原来只是 Java 2 平台的一个版本而已。下面就来介绍这三个版本的 Java 2 平台的具体含义及相互之间的区别：

🕮 Java 2 平台微型版（J2ME）

J2ME 是 Java 2 Platform Micro Edition 的首字母简写，意思是 Java 2 平台微型版，适于开发小型设备和智能卡上的应用系统，如手机和掌上电脑的操作系统等。

🕮 Java 2 平台标准版（J2SE）

J2SE 是 Java 2 Platform Standard Edition 的首字母简写，意思是 Java 2 平台标准版，适于创建普通台式电脑上的应用系统，如 PC 机、小型工作站的应用软件等。

Java 2 平台企业版（J2EE）

J2EE 是 Java 2 Platform Enterprise Edition 的首字母简写，意思是 Java 2 平台企业版，适于创建服务器端的大型应用软件和服务系统。

专家指点

这里“企业版”的含义绝不是说它只适于企业，而是代表了一种规模，详见 1.1.3 小节。

实际上，平常人们所说的 Java 语言就是 J2SE 的一部分。此外，J2SE 还包括标准版的运行环境和软件开发工具包。J2ME 和 J2EE 都是在 J2SE 的基础上创建的，这也是 J2SE 为什么叫标准版的原因了。J2ME 是 J2SE 的精简版，它在最大限度上简化了 Java 语言，并使运行环境高度优化；而 J2EE 则是在 J2SE 的基础上进行扩充，以满足在企业级应用中的需求。

Sun 公司就是通过这三个版本的 Java 2 平台，使 Java 的应用得到了极大的扩展。小到家电设备中的嵌入式芯片、手机和掌上电脑，大到各种应用服务器现在都可以使用 Java 2 进行开发了。

1.1.3 深入理解 J2EE

可以说，J2EE 就是在 J2SE 的基础上进行了一定的扩充，以满足在企业级应用的 Java 2 平台。但是，只是这样解释还不能够完整地揭示 J2EE 的内涵。比如，J2EE 是适用于企业级应用的，那么什么是企业级应用呢？再比如，J2EE 是一种平台，那么什么又是平台呢？本节将深入探讨 J2EE 的这些概念。

业务逻辑

在 J2EE 中，业务逻辑是一个十分重要的概念，可以将业务逻辑理解成是一个业务过程。例如，银行用户在提款机上取款就是一个比较复杂的业务过程，实现这个业务过程可能要包括许多的步骤，如读取用户的账号和密码并到银行的数据库中核对、读取取款金额并到数据库中扣除等。如果需要通过软件来实现这些业务过程，就必须把它们概括成业务逻辑。业务逻辑抽象了这些过程，方便了人们通过软件来实现这些业务过程。业务逻辑是一个商务软件的核心内容，要想高效地实现商务软件的功能就必须设计好它的业务逻辑内容。

企业级应用

J2EE 是 Java 2 平台企业版的意思，许多初学者都不明白这里的“企业版”是什么意思。企业版绝不是指它只适合于企业使用，而是代表了一种规模，这种规模的应用为企业级应用。必须强调的是，这并不是说只有企业级应用才可以使用 J2EE，而是说 J2EE 适合于企业级应用。如果使用 J2EE 开发企业级应用，可以大幅地提高效率、节省时间和成本。如果非要在一个简单的桌面系统中使用 J2EE 技术，也是没什么不行的，但是这样做就没有什么意义了。

简单来说，企业级应用就是大规模的应用。这种大规模的应用一般都是要处理大量底层

数据的，因此能对数据库进行维护和访问是必不可少的。除此之外，企业级应用一般还具有以下一些特征：

- 系统一般有许多的使用者，需要有很长的生命期，所以应用系统必须要稳定可靠。
- 组件往往分布在异构的计算环境中，所以应用系统必须可以跨平台。
- 对系统的可维护性、可扩展性与可重用性有很高的要求。
- 需要有事务管理、安全管理以及线程管理等功能。

但是必须指出，典型的企业级应用往往并不是从空白开始，而是在企业原有的“企业信息系统”（EIS，Enterprise Information System）的基础上进行更新。在实际的应用开发中，企业往往已经使用早期的计算机技术实现了企业管理与业务流程的信息化，但随着新技术与新需求的涌现，原有的系统渐渐显得不够完善，因此企业可能就希望升级现有的系统，以丰富它的功能。还有一种情况就是企业早期已经部分地实现了企业管理与业务流程的信息化，而现在又希望在原来的基础上加入新的业务内容。例如，一个书店，早期可能已经使用关系数据库实现了对书店内书目的管理，用户在店内购物时可以通过这个系统查询店内的各种书目，如果在此基础上要实现网上购书的话，就可能利用到原有的图书管理系统，若只是使用新技术而将原有的 EIS 完全废除，则将造成极大的浪费，同时效率也不高。使用 J2EE 就能完全解决这些问题，它采用多层的结构，能从 EIS 中读取数据并进行处理，这种多层的结构将在 1.2 节中进行介绍。

专家指点

这里“企业”的概念是指使用由 J2EE 技术开发出来的系统的单位，并不一定是一个公司或工厂，也可能是一个学校、一个事业单位或者政府机构等。EIS 只是对企业原有系统的一种代称，不同企业的 EIS 是不一样的。

平台

在 Java 2 以前，Java 一直定位为一种语言，到 Java 2 时，Sun 公司实际上已经将 Java 定位成一种平台了，所以 Java 2 就是以三种平台的方式进行发布的。平台与语言有什么不同呢？举个简单的例子，C++是一种语言，而 Windows 则是一个平台。C++编写出来的程序必须要在 Windows 上运行，而不能跨过操作系统直接运行在电脑上。Java 则不同，由于包含了 Java 虚拟机（JVM）和 Java 运行环境（JRE），Java 几乎是不依赖于任何操作系统的。更进一步，如果在 Pentium 处理器中嵌入 JVM，那么 Java 语言就可以像汇编语言一样直接运行在处理器上了。也就是说，Java 平台使得用户可以直接在平台上进行开发，并在平台上运行。整个平台可以很容易地从一台机器移植到另一台机器上，而不用做任何更改。

此外，J2EE 还制定了一组规范，将 J2EE 的平台责任划分成七大部分以实现软件开发的分工。每个部分都由独立的供应商或专业人员来完成，从而大大提高了开发效率。这七部分是：

- 平台供应商：提供 J2EE 平台，包括组件容器、平台 APIs 等；
- 组件供应商：提供应用程序组件，包括 HTML 页面设计人员、JSP 程序员、EJB 开发人员等；
- 组件装配人员：组装由组件供应商提供的组件，最后形成 EAR（Enterprise Archive）

文件；

- 部署人员：将装配好的组件部署到容器上；
- 系统管理员：管理和配置部署好的系统；
- 工具供应商：提供开发组件所使用的工具；
- 系统组件供应商：提供系统组件。

学习 J2EE 的程序员，一般就是充当了组件供应商的角色，主要职责是开发 Servlet/JSP 组件和 EJB 组件。如果进一步分工，程序员也分为 Servlet/JSP 组件开发人员和 EJB 组件开发人员。程序员只需要关心自己将要开发的组件技术，而无需学习其他组件的开发技术。这就使得程序员能够集中精力，成为这一领域内的专家。

此外，J2EE 的部署也是一门十分重要的学问。如果不把开发出来的组件部署到服务器上，那么这些组件就无法发挥它们的作用。记住，组件是不能够独立运行的，必须要部署到相应的服务器上才行。J2EE 程序员往往也需要掌握组件部署的技术，这些在本书的后续章节中将会有所介绍。对于其他部分的角色责任，J2EE 规范中也都制定了相应的内容以保证与其他角色之间的协调。这些不是本书的重点，基本不会介绍。读者如果对这些内容感兴趣，可以到 Sun 公司的网站上去下载相应的规范。

1.2 J2EE 体系结构

前面曾经介绍过 J2EE 的分层思想，在 J2EE 的规范中这种分层被称为 J2EE 的体系结构（J2EE Architecture）。在企业级应用系统中，体系结构的概念是十分重要的，它决定了系统开发的难易程度以及系统开发出来后的可移植性和可维护性。以往人们使用最多的是两层体系结构，有过程序开发经验的人可能都知道 C/S（Client/Server）模式，这实际上就是一种典型的两层体系结构。在这种结构中，Server 一般都是运行在服务器上，并实现了一定功能（如数据存储等）的服务进程；而 Client 则运行在客户机上，通过网络调用 Server 所提供服务的客户端进程。

对于企业级应用，这种模式的弊端是显而易见的。首先，由于业务逻辑没有从服务器或客户机上独立出来，所以它必须与 Server 端程序或 Client 端程序结合在一起。如果将业务逻辑与 Server 端结合在一起，必然会降低系统的可移植性。这种模式的应用系统通常称为瘦客户端的应用系统。例如，如果 Server 端使用 MS SQL Server 数据库，那么在 Server 端开发的应用程序就必然是与 SQL Server 数据库相关的。而当企业需要将 Server 端程序应用在 Oracle 数据库上时，那整个 Server 端程序就必须要全部重写，以适应数据库的变化。所以这样的 Server 端程序，可移植性是极差的。

业务逻辑与 Client 端结合在一起的应用系统，通常称为瘦服务器端的应用系统。这种模式的应用系统也存在着同样的问题。不仅如此，由于客户端可能不止一个，它的维护问题更加复杂。例如，当客户端的业务逻辑需要更改时，必须对客户端程序进行升级。当客户端程序升级后，就必须要花费大量的人力和财力将它们重新装到每一台客户机上。如果客户机比较少还好办，但是如果客户机成百上千，并且分布在世界各地，那成本就相当的可观了。

因此近年来，越来越多的人开始提倡使用多层的体系结构。多层的体系结构将业务逻辑

从客户端和服务器端独立出来，将整个系统划分为数据服务器、应用逻辑服务器和客户端机器。客户端机器通过网络调用应用逻辑服务器的服务，而应用逻辑服务器又调用数据服务器中的数据进行相应的处理，最后将结果返回给客户。

初学者对于体系结构的思想都会感到迷惑，不知道软件体系结构中每一层的功能是什么，也不知道它们的功能是如何实现的。下面就来介绍 J2EE 的体系结构，这在整个 J2EE 的学习中是相当重要的基础内容。

1.2.1 J2EE 多层体系结构

J2EE 的体系结构是多层的分布式体系结构，应用逻辑按功能划分为组件（组件的概念见 1.2.2 节），组件根据各自所在的层分布在不同的机器上。当然，如果要把这些组件都装在同一台机器上也是没什么不可以的，这要根据实际情况决定。事实上，Sun 公司设计 J2EE 的初衷正是为了解决两层模式的弊端。图 1-1 描述的就是这种多层的体系结构。

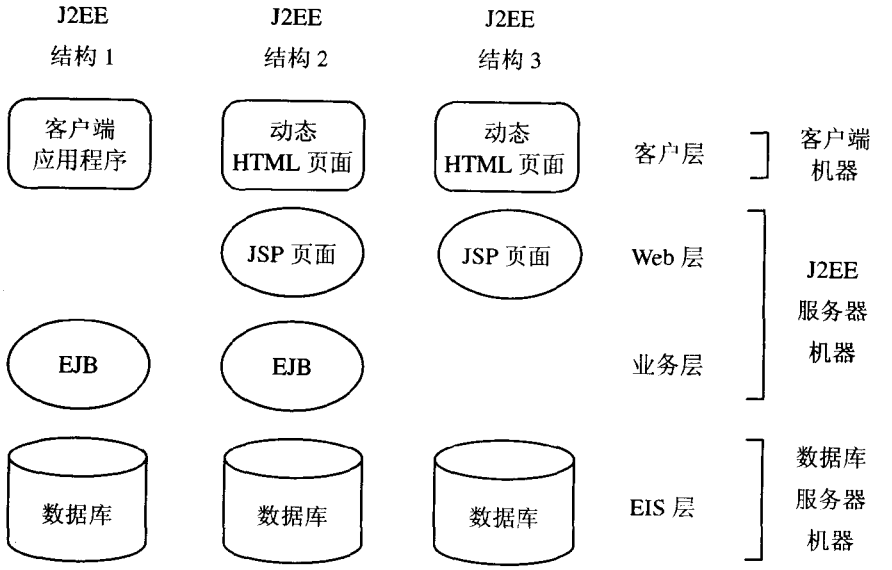

图 1-1 J2EE 多层体系结构

由图 1-1 可以看出，J2EE 常用的体系结构有三种。其中最典型的是结构 2 中所示的四层结构，它用到了所有的 J2EE 组件。但是在实际的应用中，往往并不需要应用所有的组件，有时只需要用 EJB 而不需要 JSP，如结构 1 所示；有时则只需要用 JSP 而不需要 EJB，如结构 3 所示。具体采用哪种结构完全是按照系统的需要进行选择的。下面就来看看这种多层的体系结构是如何解决两层结构中存在的问题。

EIS 层

EIS 层也就是数据层，它是整个应用系统将要处理的数据源泉。一般情况下，这一层就是一个关系数据库，也可能是企业原有的一个信息系统。例如，一个网上购物站点，它的 EIS 层可能就是一个存储了客户信息、订单、库存商品等内容的关系数据库。系统在处理客户端

请求时，将从 EIS 层中读取数据。所以 J2EE 程序开发人员必须要具备数据库的基础知识，例如，SQL 语言、数据库事务管理等。

在 J2EE 中，提供数据库访问功能的是 JDBC API。通过这些应用程序接口，程序可以轻松地访问各种关系数据库。本书第 3 章将对 JDBC 进行详细的介绍。

业务层

业务层是 J2EE 体系结构的核心部分，所有与应用系统相关的业务逻辑都在这一层中实现。在 J2EE 中实现业务层功能的组件是 EJB，它使用固定的组件结构将业务逻辑封装在组件中，并且使用容器实现了底层的一些细节功能。

因此，在实际应用中，开发人员会将大部分的精力集中在 EJB 的设计实现上。一旦 EJB 的设计确定下来，整个系统的功能与结构也就确定下来了。所以学习 J2EE 的重点是要学会如何开发 EJB，并且还要知道如何将它们部署到服务器上。当然在一些应用系统中也可能选择不使用 EJB，而采用 JSP 或 Servlet 来实现业务逻辑，但是这样的应用系统往往是不易维护的。

Web 层

Web 层也称为表述层，因为它描述了将要发送到客户端浏览器的内容是如何显示的。Web 层的组件要运行在 Web 服务器上，它会根据客户端发送过来的请求进行处理，然后将结果以 HTML 文件的方式发送给客户端。客户端的浏览器会根据发送回来的 HTML 文件，将结果显示在浏览器窗口中。

在 J2EE 中实现 Web 层功能的组件是 Servlet/JSP 组件，它们在 J2EE 中的作用是十分重要的。JSP 与 Servlet 在本质上是一样，因为它最后要被翻译成一个 Servlet 文件后才能在服务器上运行。JSP 目前的发展趋势正在走向结构化，也就是将它所包含的请求处理内包装在标记文件或 Servlet 类中，而在 JSP 中将只描述页面是如何显示的。Struts 技术与 JSTL 就是这种发展趋势的产物。

客户层

在 J2EE 中，客户层已经变得十分简单。它所要实现的功能只是提交用户请求，并显示服务器处理的结果。在实际的开发中，客户层往往使用的就是一个浏览器软件，如 Internet Explorer、NetScape 等。由于这些软件在市场上已经十分流行，而且在网络十分普及的今天，几乎每一台计算机上都会装有浏览器。所以，客户层的软件基本上已经不用再进行开发了。

当然，在不使用 Servlet/JSP 的情况下，也是需要开发客户端的软件的。但是由于客户端的软件只是简单的实现结果的显示，所以就算要进行开发，它的实现和维护也是十分简单的。

1.2.2 组件、容器与服务器

如果将 J2EE 的体系结构看成是对应用系统的一种横向分层，那么每一层又分成组件、容器和服务器则可以看成是对每层的纵向分层。纵向分层使得组件与平台细节彻底隔离开来，使组件不需要包含与底层操作系统相关的内容，所以大大提高了组件的可移植性和可重

用性。不仅如此，这种纵向分层还导致了软件开发的又一次分工，产生了容器与服务器开发商。容器与服务器开发商将组件中的一些共同功能剥离出来，使组件的功能和结构更单纯，从而提高了组件开发的效率。

组件

组件可以定义为一种自治的、提供外部公共接口的、动态可用的事物处理过程，组件可以用来构建其他组件或者应用程序。可见，组件是对可重用代码的一种封装，这些代码可以用来执行应用程序的一些功能。

组件与函数、对象有些相似，因为它们都是对一定功能的实现和封装，但是它们的区别也是明显的。组件与函数的区别在于函数是无状态的，即函数调用不能保存上一次调用结束后的任何信息，而组件则能够保存客户调用的信息；组件与对象的区别在于组件提供了动态可用的接口、属性和操作，而对象则是静态的。

J2EE 规范中定义了 3 类组件（共 5 种），这些组件在开发完成后，可以部署到相应的容器上。这 3 类组件是：

- 客户端组件：应用程序客户端组件（application clients）或 Applet 组件。
- Web 组件：JSP 或 Servlet 组件。
- 业务逻辑组件：EJB 组件。

这些组件分别对应于 J2EE 四层结构中的相应层，并实现了该层的功能。客户端组件应用于客户层，一般用于显示服务器返回的处理结果。由于它与 J2SE 中的应用程序及 Applet 相近，所以不是 J2EE 的重点。而 Web 组件与业务逻辑组件由于实现了 Web 层与业务逻辑层的功能，所以是 J2EE 中最重要的两种组件。

容器与服务器

组件是不能独立运行的，必须要为它提供相应的运行环境，为组件提供运行环境的就是容器。因此 J2EE 中的任何组件在运行前都必须部署到容器中去，部署的过程实际上就是指定容器的配置，告诉容器应该提供怎样的服务给组件才能保证组件良好运行。容器是组件与底层平台之间的接口，也就是说组件是容器与外界进行联系的媒介。组件与组件之间是不能直接进行沟通的，它们之间的信息交换必须要通过容器进行。

那么组件是如何在容器上运行的呢？这里就要讲一下容器回调的概念了。一般情况下，程序员在编写组件时可以通过容器的各种接口调用容器提供的服务。但是在许多时候，容器也必须要调用组件的接口，来完成用户的请求。这种由容器调用组件接口的调用方式就称为回调。组件在容器中运行的过程，实际上就是容器回调组件的过程。

首先，组件在部署到容器上之后，容器会根据组件的组成文件生成相应的对象实例。当用户的请求发送过来时，容器会对它们进行一些先期的处理，这包括身份验证、负载平衡等等。然后，容器就会根据请求的内容回调组件中的相应方法，通过组件中的这些方法来完成用户的请求。

所以，组件虽然是用来实现层功能的，但是它却必须要为容器提供规范中约定的调用接口。了解这一点，对于理解 J2EE 组件的一些开发规范是十分有帮助的。初学者往往认为组件中定义的方法都是直接为用户定义的，并且用户可以直接调用这些方法，但这是不正确的。记住，在 J2EE 组件中编写的大部分方法都是提供给容器的回调接口，容器会根据用户的请