希赛®IT技术讲堂 CSAI.cn Java篇

Oracle +JSP 系统应用开发

阮国明　边 伟　等编著

机械工业出版社
China Machine Press

编写委员会

组编：希赛顾问团

顾问：张友生

主审：邓子云

编委（按姓名拼音排序）：

边　伟	陈亿春	崔海波	方海光	冯向科
葛志春	郭　莹	赫　斌	黄　婧	黄少年
李　刚	刘　毅	陆秉炜	刘志成	娄嘉鹏
聂艳明	阮国明	孙鸿飞	施　游	唐　俊
唐天广	王　军	吴吉义	吴伟敏	薛大龙
王红安	王　冀	王　勇	谢　顺	杨　森
张晓燕	张立东	朱小平		

第一篇

Java 与 JSP 基础知识导学

第1章

JSP 相关技术概述

本章专家知识导学

本章内容将是学习后续章节的基础。读者可通过本章了解与 JSP 相关的各种基本概念，如果对 JSP 相关技术已经比较熟悉的读者，可跳过本章继续学习。

JSP 技术由 Sun 公司提出。利用它可以很方便地在页面中生成动态的内容，使网络应用程序可以输出多姿多彩的动态页面。JSP 技术通常与 Java Servlet 技术结合，可以在 HTML 页面或者其他标记语言中加入 Java 代码，如 XML 中内嵌了 Java 代码段，或调用外部 Java 组件。JSP 作为一个前端处理工具，可以使用 JavaBeans 和 EJB(Enterprise JavaBeans)进行完美地结合，实现复杂的商业逻辑和动态功能。

在一个典型的数据库应用中，JSP 页面将会调用某些 JavaBean 或者 Enterprise JavaBean 组件，这些组件可以通过 JDBC 直接或间接地访问数据库。JSP 页面在运行之前要被转换成 Java Servlet，转换过程是按需进行的，有时可能会提前进行，然后它可以处理 HTTP 请求，并生成响应信息。JSP 技术为编写 Servlet 程序提供了更为便利的途径。

另外，JSP 页面和 Servlet 程序是可以相互操作的。也就是说，JSP 页面可以包含从 Servlet 程序输出的内容，可以将内容输出到 Servlet 程序。反过来 Servlet 程序也可以包含从 JSP 页面输出的内容，并且可以将内容输出到 JSP 页面中。

本章主要讲解与 Java 技术及 JSP 技术相关的基本概念，读者需要重点了解本章涉及的各个知识点。

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。具有强大功能的 JSP 技术的推出，更是让 Java 成为 Web 应用程序的首选开发工具。

【专家提示】 要学习 JSP，Java 基础是必不可少的。本书将在第 2 章中为没有 Java 基础知识的读者简单介绍 Java 的基础语法、JavaBeans 和 Java Servlet 的基础知识，它们是在学习 JSP 之前必须掌握的 Java 基础知识。

1.1.1 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 E-mail 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 JavaBean 实现了同样的功能扩充。JSP 对于在 Web 应用中集成 JavaBean 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。JavaBean 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访问等。

【专家提示】 和传统的 ASP 或 PHP 页面相比，JSP 页面是非常简洁的。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都被封装到了 JavaBeans 中。

1.1.2 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 程序员提供了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 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。

1.1.3 J2EE

本小节将从 5 个方面对 J2EE 进行比较全面的介绍，其中包括 J2EE 的概念、它的优势、J2EE 典型的四层模型、其框架结构，以及 J2EE 的 13 种核心技术。本小节将分门别类地对 J2EE 中的服务、组件、层次、容器及 API 做详细的介绍，希望通过本小节内容的介绍，读者将会对 J2EE 有一个更清晰的认识。

1. J2EE 的概念

目前，Java 2 平台有 3 个版本，它们是适用于小型设备和智能卡的 Java 2 平台 Micro 版(Java 2 Platform Micro Edition，J2ME)、适用于桌面系统的 Java 2 平台标准版(Java 2 Platform Standard Edition，J2SE)及适用于创建服务器应用程序和服务的 Java 2 平台企业版(Java 2 Platform Enterprise Edition，J2EE)。

J2EE 是一种利用 Java 2 平台来简化企业解决方案的开发、部署和管理相关的复杂问题的体系结构。J2EE 技术的基础就是核心 Java 平台或 Java 2 平台的标准版，J2EE 不仅巩固了标准版中的许多优点，例如“编写一次、随处运行”的特性、方便访问数据库的 JDBC API、CORBA 技术以及能够在 Internet 应用中保护数据的安全模式等，同时还提供了对 EJB(Enterprise JavaBeans)、Java Servlets API、JSP(Java Server Pages)以及 XML 技术的全面支持。其最终目的就是成为一个能够使企业开发者大幅缩短投放市场时间的体系结构。J2EE 体系结构提供中间层集成框架用来满足无须太多费用而又需要高可用性、高可靠性以及可扩展性应用的需求。通过提供统一的开发平台，J2EE 降低了开发多层应用的费用和复杂性，同时提供对现有应用程序集成强有力的支持，完全支持 Enterprise JavaBeans，有良好的向导支持打包和部署应用，添加目录支持，增强了安全机制，提高了性能。

2. J2EE 的优势

J2EE 为搭建具有可伸缩性、灵活性、易维护性的商务系统提供了良好的机制。

(1)保留现存的 IT 资产

由于企业必须适应新的商业需求，利用已有的企业信息系统方面的投资，而不是重新制定全盘方案就变得很重要。因此，一个以渐进的(而不是激进的，全盘否定的)方式建立在已有系统之上的服务器端平台机制将是公司所需求的。J2EE 架构可以充分地利用用户原有的投资，例如一些公司使用的 BEA Tuxedo、IBM CICS、IBM Encina、Inprise VisiBroker 以及 Netscape Appli-

cation Server。由于 J2EE 拥有广泛的业界支持和一些重要的"企业计算"领域供应商的参与，使得这一切将成为可能。每一个供应商都对现有的客户提供了不用废弃已有的设备，就可进入可移植的 J2EE 领域的升级途径。由于基于 J2EE 平台的产品几乎能够在任何操作系统和硬件配置上运行，因此，现有的操作系统和硬件也能被保留使用。

(2)高效的开发

J2EE 允许公司把一些通用的、很繁琐的服务端任务交给中间件的供应商去完成，这样开发人员可以将精力集中在如何创建商业逻辑上，相应地缩短了开发时间。高级中间件供应商提供以下一些复杂的中间件服务：

- **状态管理服务**。让开发人员编写更少的代码，不用关心如何管理状态，这样能够更快地完成程序开发。
- **持续性服务**。让开发人员不用对数据访问逻辑进行编码就能编写应用程序，能生成更轻巧、与数据库无关的应用程序，这种应用程序更易于开发与维护。
- **分布式共享数据对象 CACHE 服务**。让开发人员开发高性能的系统，极大地提高整体部署的伸缩性。

(3)支持异构环境

J2EE 能够开发部署在异构环境中的可移植程序。基于 J2EE 的应用程序不依赖任何特定的操作系统、中间件和硬件。因此设计合理的基于 J2EE 的程序只需开发一次就可以部署到各种平台。这在典型的异构企业计算环境中是十分关键的。J2EE 标准也允许客户订购与 J2EE 兼容的第三方现成的组件，把它们部署到异构环境中，节省了由自己制订整个方案所需的费用。

(4)可伸缩性良好

企业必须选择一种服务器端平台，这种平台能够提供极佳的可伸缩性以满足那些在其系统上进行商业运作的大批新客户。基于 J2EE 平台的应用程序可被部署到各种操作系统上，例如可被部署到高端 UNIX 与大型机系统，这种系统单机可支持 64 ~ 256 台处理器(这是 NT 服务器所望尘莫及的)。J2EE 领域的供应商提供了更为广泛的负载平衡策略，能消除系统中的瓶颈，允许多台服务器集成部署。这种部署可高达数千台处理器，实现可高度伸缩的系统，以满足未来商业应用的需要。

(5)稳定的可用性

一个服务器端平台必须能全天候运行以满足公司客户、合作伙伴的需要。因为 Internet 是全球化的、无处不在的，即使在夜间按计划停机也可能造成严重损失，如果是意外停机，将会造成灾难性后果。J2EE 部署到可靠的操作环境中，它们支持长期的可用性。一些 J2EE 部署在 Windows 环境中，客户也可以选择健壮性能更好的操作系统如 Sun Solaris、IBM OS/390。最健壮的操作系统可达到 99.999% 的可用性或每年只需 5 分钟的停机时间。这是实时性很强的商业系统理想的选择。

3. J2EE 的四层模型

J2EE 使用多层的分布式应用模型，应用逻辑按功能划分为组件，各个应用组件根据它们所在的层分布在不同的机器上。事实上，Sun 公司设计 J2EE 的初衷正是为了解决两层模式(Client/Server)的弊端。在传统模式中，客户端担当了过多的角色而显得臃肿，在这种模式中，第一次部署的时候比较容易，但难于升级或改进，可伸展性也不理想，而且经常基于某种专有的协议——通常是某种数据库协议，这使得重用业务逻辑和界面逻辑非常困难。现在 J2EE 的多层企业级应用模型已将两层模型中的不同层面拆分成许多层。一个多层化应用能够为不同的每种服务提供一个独立的层，以下是 J2EE 典型的四层结构(参见图 1-1)：

- 运行在客户端机器上的客户层组件；
- 运行在 J2EE 服务器上的 Web 层组件；
- 运行在 J2EE 服务器上的业务逻辑层组件；
- 运行在数据库服务器上的企业信息系统(Enterprise Information System)层软件。

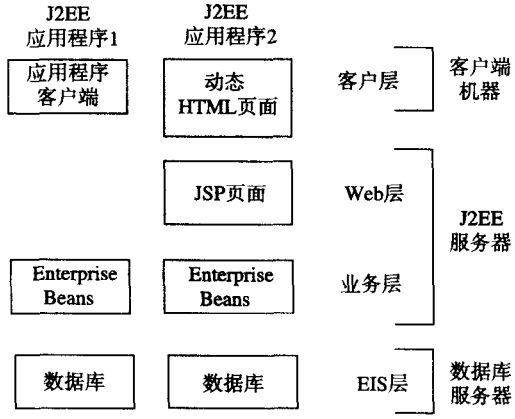

图 1-1 J2EE 典型的四层结构

(1)J2EE 应用程序组件

J2EE 应用程序是由组件构成的，J2EE 组件是具有独立功能的软件单元，它们通过相关的类和文件组装成 J2EE 应用程序，并与其他组件交互。J2EE 说明书中定义了以下的 J2EE 组件：

- 应用客户端程序和 applets 是客户层组件；
- Java Servlet 和 JavaServer Pages(JSP)是 Web 层组件；
- Enterprise JavaBeans(EJB)是业务层组件。

(2)客户层组件

J2EE 应用程序可以是基于 Web 方式的，也可以是基于传统方式的。

Web 层组件可以是 JSP 页面或 Servlet，按照 J2EE 规范，静态的 HTML 页面和 applets 不算是 Web 层组件。

如图 1-2 所示的客户层那样，Web 层可能包含某些 JavaBean 对象用来处理用户输入，并把输入发送给运行在业务层上的 Enterprise Bean 来进行处理。

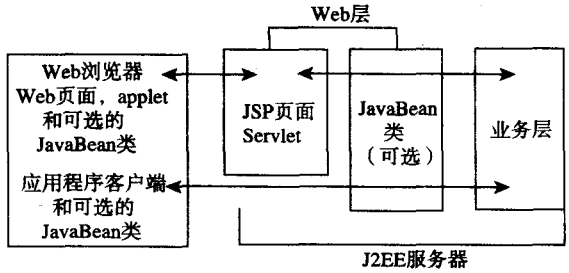

图 1-2 客户层组件

(3)业务层组件

业务层代码的逻辑用来满足银行、零售、金融等特殊商务领域的需要，由运行在业务层上的 Enterprise Bean 进行处理。图 1-3 说明了一个 Enterprise Bean 是如何从客户端程序接收数据，进行处理(如果有必要的话)，并发送到 EIS 层存储的，这个过程也可以逆向进行。

有三种企业级的 Bean：会话(session)Beans、实体(entity)Beans 和消息驱动(message-driven)Beans。会话 Bean 表示与客户端程序的临时交互，当客户端程序执行完后，会话 Bean 和相关数据就会消失。相反，实体 Bean 表示数据库的表中一行永久的记录，当客户端程序中止或服务器关闭时，就会有潜在的服务保证实体 Bean 的数据得以保存。消息驱动 Bean 结合了会话 Bean 和 JMS 的消息监听器的特性，允许一个业务层组件异步地接收 JMS 消息。

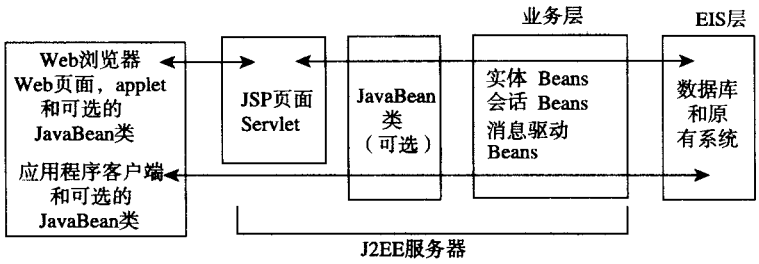

图 1-3　业务层组件

(4)企业信息系统层

企业信息系统层处理企业信息系统软件，包括企业基础建设系统，例如企业资源计划(ERP)，大型机事务处理、数据库系统和其他的遗留信息系统，例如，J2EE 应用组件可能为了数据库连接需要访问企业信息系统。

4. J2EE 的结构

基于组件具有平台无关性的 J2EE 结构使得 J2EE 程序的编写十分简单，因为业务逻辑被封装成可复用的组件，并且 J2EE 服务器以容器的形式为所有的组件类型提供后台服务。由于不用自己开发这种服务，所以可以让开发人员集中精力解决手头的业务问题。

(1)容器和服务

容器设置定制了 J2EE 服务器所提供的内在支持，包括安全、事务管理、JNDI(Java Naming and Directory Interface)寻址及远程连接等服务，以下列出了最重要的几种服务。

- J2EE 安全(security)模型可以让您配置 Web 组件或 Enterprise Bean，这样只有被授权的用户才能访问系统资源。每一客户属于一个特别的角色，而每个角色只允许激活特定的方法。您应在 Enterprise Bean 的布置描述中声明角色和可被激活的方法。由于拥有这种声明性的方法，您不必编写加强安全性的规则。
- J2EE 事务管理(transaction management)模型让您指定组成一个事务中所有方法间的关系，这样，一个事务中的所有方法将被当成一个单一的单元，当客户端激活一个 Enterprise Bean 中的方法时，容器介入一管理事务。因为有容器管理事务，所以在 Enterprise Bean 中不必对事务的边界进行编码。要求控制分布式事务的代码会非常复杂。您只需在布置描述文件中声明 Enterprise Bean 的事务属性，而不用编写并调试复杂的代码，容器将读此文件并为您处理此 Enterprise Bean 的事务。
- JNDI 寻址(JNDI Lookup)服务向企业内的多重名字和目录服务提供了一个统一的接口，这样，应用程序组件可以访问名字和目录服务。
- J2EE 远程客户连接(remote client connectivity)模型管理客户端和 Enterprise Bean 间的低层交互。当一个 Enterprise Bean 创建后，一个客户端可以调用它的方法，就像它和客户端位于同一虚拟机上一样。
- 生存周期管理(life cycle management)模型管理 Enterprise Bean 的创建和移除，一个 Enter-

prise Bean 在其生存周期中将会历经几种状态。容器创建 Enterprise Bean，并在可用实例池与活动状态中移动它，最终将其从容器中移除。即使可以调用 Enterprise Bean 的 create 及 remove 方法，容器也将会在后台执行这些任务。

- 数据库连接池(database connection pooling)模型是一个有价值的资源。获取数据库连接是一项耗时的工作，而且连接数非常有限。容器通过管理连接池来缓和这些问题。Enterprise Bean 可从池中迅速获取连接。在 Bean 释放连接后可为其他 Bean 使用。

(2)容器类型

J2EE 应用组件可以安装部署到以下几种容器中去，如图 1-4 所示。

- EJB 容器管理所有 J2EE 应用程序中企业级 Bean 的执行，Enterprise JavaBean 和它们的容器运行在 J2EE 服务器上。
- Web 容器管理所有 J2EE 应用程序中 JSP 页面和 Servlet 组件的执行。Web 组件和它们的容器运行在 J2EE 服务器上。
- 应用程序客户端容器管理所有 J2EE 应用程序中应用程序客户端组件的执行。应用程序客户端和它们的容器运行在 J2EE 服务器上。
- Applet 容器是运行在客户端机器上的 Web 浏览器和 Java 插件的结合。

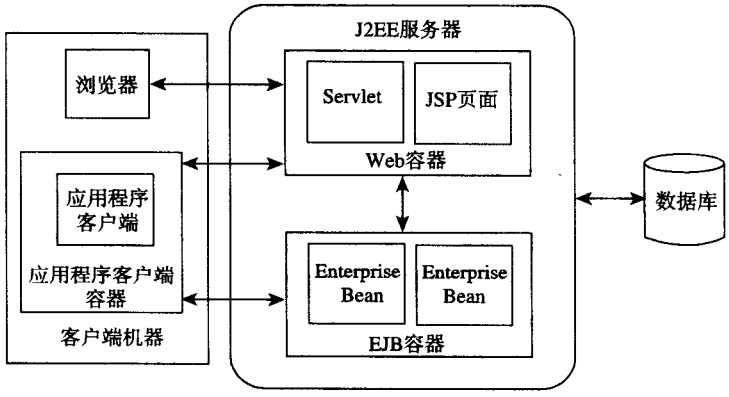

图 1-4 EJB 容器

5. J2EE 的核心 API 与组件

J2EE 平台由一整套服务(services)、应用程序接口(API)和协议构成，它对开发基于 Web 的多层应用提供了功能支持。下面对 J2EE 中的 13 种技术规范进行简单的描述(限于篇幅，这里只能进行简单的描述)：

(1)JDBC(Java Database Connectivity)

JDBC API 为访问不同的数据库提供了一种统一的途径，像 ODBC 一样，JDBC 对开发者屏蔽了一些细节问题。另外，JDCB 对数据库的访问也具有平台无关性。

(2)JNDI(Java Name and Directory Interface)

JNDI API 被用于执行名字和目录服务。它提供了一致的模型来存取和操作企业级的资源如 DNS 和 LDAP、本地文件系统，或应用服务器中的对象。

(3)EJB(Enterprise JavaBean)

J2EE 技术之所以赢得媒体广泛重视，原因之一就是 EJB。EJB 提供了一个框架来开发和实施分布式商务逻辑，由此很显著地简化了具有可伸缩性和高度复杂的企业级应用的开发。EJB 规范定义

了 EJB 组件在何时如何与它们的容器进行交互作用。容器负责提供公用的服务，例如目录服务、事务管理、安全性、资源缓冲池以及容错性。值得注意的是，EJB 并不是实现 J2EE 的唯一途径。由于 J2EE 的开放性，使得某些厂商能够以一种和 EJB 平行的方式来达到同样的目的。

(4) RMI(Remote Method Invoke)

正如其名字所表示的那样，RMI 协议调用远程对象上的方法。它使用了序列化方式在客户端和服务器端传递数据。RMI 是一种被 EJB 使用的更底层的协议。

(5) Java IDL/CORBA

在 Java IDL 的支持下，开发人员可以将 Java 和 CORBA 集成在一起。他们可以创建 Java 对象并使之在 CORBA ORB 中展开，他们还可以创建 Java 类并作为和其他 ORB 一起展开的 CORBA 对象的客户。后一种方法提供了另外一种途径，通过它 Java 可以用于将新的应用和旧的系统相集成。

(6) JSP(Java Server Pages)

JSP 页面由 HTML 代码和嵌入其中的 Java 代码所组成。服务器在页面被客户端请求后对这些 Java 代码进行处理，然后将生成的 HTML 页面返回给客户端的浏览器。

(7) Java Servlet

Servlet 是一种小型的 Java 程序，它扩展了 Web 服务器的功能。作为一种服务器端的应用，当被请求时开始执行，这和 CGI Perl 脚本很相似。Servlet 提供的功能大多与 JSP 类似，不过实现的方式不同。JSP 通常是在大多数 HTML 代码中嵌入少量的 Java 代码，而 Servlet 全部由 Java 写成并且生成 HTML。

(8) XML(Extensible Markup Language)

XML 是一种可以用来定义其他标记语言的语言，它用于在不同的商务过程中共享数据。XML 的发展和 Java 是相互独立的，但是，它和 Java 具有的相同目标正是平台独立性。通过将 Java 和 XML 的组合，您可以得到一种完美的具有平台独立性的解决方案。

(9) JMS(Java Message Service)

JMS 是用于和面向消息的中间件相互通信的应用程序接口(API)。它既支持点对点的域，又支持发布/订阅(publish/subscribe)类型的域，并且提供对下列类型的支持：经认可的消息传递，事务型消息的传递，一致性消息和具有持久性的订阅者支持。JMS 还提供了另一种方式来对您的应用与旧的后台系统相集成。

(10) JTA(Java Transaction Architecture)

JTA 定义了一种标准的 API，应用系统由此可以访问各种事务监控。

(11) JTS(Java Transaction Service)

JTS 是 CORBA OTS 事务监控的基本实现。JTS 规定了事务管理器的实现方式，该事务管理器是在高层支持 Java Transaction API(JTA)规范，并且在较底层实现 OMG OTS Specification 的 Java 映像。JTS 事务管理器为应用服务器、资源管理器、独立的应用以及通信资源管理器提供了事务服务。

(12) JavaMail

JavaMail 是用于访问邮件服务器的 API，它提供了一套邮件服务器的抽象类，不仅支持 SMTP 服务器，也支持 IMAP 服务器。

(13) JAF(JavaBeans Activation Framework)

JavaMail 利用 JAF 来处理 MIME 编码的邮件附件。MIME 的字节流可以被转换成 Java 对象，或者转换自 Java 对象。大多数应用都可以不需要直接使用 JAF。

1.1.4 EJB

EJB 就是前面说的 Enterprise Java Beans。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 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您的分布式应用程序。当您把代码写好之后，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，并添加上一些配置参数。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，并产生该 Beans 的一个实例，这样，客户就能调用 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。

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，应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，可以使代码集中在解决手边的业务问题上，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：实体 Beans(Entity Beans)和会话 Beans(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。在后一种情况下，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 就会丢失。这种模式通常被用于像 PL/SQL 这样的数据库程序设计语言中。

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，或用于表示被存储数据的被共享的视图业务对象是无状态 Session Beans 的一个例子。

因为 Enterprise Beans 占用可观的系统资源和带宽，您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 的代码中使用任何 JDBC2.0 API 来进行数据库访问，因为容器完成了这些工作。然而，如果使用 Beans 管理的持久性，或者要访问一个非关系型数据库的企业信息系统，那么就必须在 Beans 中提供相应的代码来完成这些工作。在 Enterprise Beans 使用 Beans 管理的持久性来访问一个数据库的情况下，必须使用 JDBC2.0 API 代码来实现该 Enterprise Beans 的生命期方法，以便处理数据的加载和存储，以及运行时在系统和持久数据存储之间维持数据的一致性。

一个使用 Beans 管理的持久性的 Enterprise Beans，或一个需要访问企业信息系统的 Web 组件必须提供合适的代码。这些代码可能是用于进行数据库访问的 JDBC2.0 API；用于访问一个特定企业信息系统的企业信息系统 API；用于抽象企业信息系统 API 的复杂性和低层细节的一个访问对象；或者是用于访问企业信息系统资源的一个连接对象。

尽管 Web 层使用 HTTP 或 HTTPS 来在各层之间传输数据，但是 EJB 层使用的是 RMI-IIOP。RMI-IIOP 是一个完整的分布式计算协议，能让任何访问 Enterprise Beans 的客户层程序或 Web 层程序直接访问 EJB 层的服务。这些服务包括用于查找和引用 Enterprise Beans 的 JNDI，发送和接收异步消息的 Java Message Service(JMS)，以及用于关系型数据库访问的 JDBC。

【思考 1】 JavaBeans 组件模型的特点是什么？EJB 组件模型特点的特点又是什么？EJB 组件模型与 JavaBeans 组件模型相比有何不同？

1.1.5 Java Servlet

Java Sevlet 是 JSP 技术的基础，而且大型的 Web 应用程序的开发需要 Java Servlet 和 JSP 配合才能完成，本小节简单介绍 Servlet 的相关知识，Servlet 的开发将在后面的章节讲述。

Servlet 这个名称大概源于 Applet，现在国内的翻译方式很多，本书为了避免误会，决定直接采用 Servlet 而不做任何翻译，读者如果愿意，可以称之为"小服务程序"。Servlet 其实和传统的 CGI 程序、ISAPI、NSAPI 等 Web 程序开发工具的作用是相同的，在使用 Java Servlet 以后，用户不必再使用效率低下的 CGI 方式，也不必使用只能在某台固定 Web 服务器平台运行的 API 方式来动态地生成 Web 页面。许多 Web 服务器都支持 Servlet，即使不直接支持 Servlet 的 Web 服务器也可以通过附加的应用服务器和模块来支持 Servlet。得益于 Java 的跨平台特性，Servlet 也是平台无关的，实际上，只要符合 Java Servlet 规范，Servlet 是完全平台无关且是 Web 服务器无关的。由于 Java Servlet 内部是以线程方式提供服务，不必对每个请求都启动一个进程，并且利用多线程机制可以同时为多个请求服务，因此 Java Servlet 效率非常高。

【专家提示】 Java Servlet 也不是没有缺点，与传统的 CGI、ISAPI、NSAPI 方式相同，Java Servlet 是利用输出 HTML 语句来实现动态网页的，如果用 Java Servlet 来开发整个网站，动态部分和静态页面的整合过程简直就是一场恶梦。这就是为什么 Sun 公司还要推出 Java Server Pages 的原因。

1.2 JSP 技术

前面说过，Java Servlet 的最大缺点就在于没有把网站的逻辑和页面的输出分开，导致整个

Servlet 代码混乱不堪。为了解决 Java Servlet 的这种缺点，Sun 公司推出了 Java Server Pages (JSP)。

1.2.1 JSP 技术概述

按照脚本语言是服务于某一个子系统的语言这种概念，JSP 应当看作是一种脚本语言，然而，作为一种脚本语言，JSP 又显得过于强大了，在 JSP 中几乎可以使用全部的 Java 类。

作为一种基于文本的、以显示为中心的开发技术，JSP 提供了 Java Servlet 的所有好处，并且，当与一个 JavaBeans 类结合在一起时，JSP 提供了一种使内容和显示逻辑分开的简单方式。分开内容和显示逻辑的好处是，更新页面外观的人员不必懂得 Java 代码，而更新 JavaBeans 类的人员也不必是设计网页的行家里手，就可以用带 JavaBeans 类的 JSP 页面来定义 Web 模板，以建立一个由具有相似外观的页面组成的网站。JavaBeans 类完成数据提供，这样在模板中就没有 Java代码，这意味着这些模板可以由一个 HTML 编写人员来维护。当然，也可以利用 Java Servlet 来控制网站的逻辑，通过 Java Servlet 调用 JSP 文件的方式来将网站的逻辑和内容分离。本章后面将对这种分离网站的逻辑和内容的设计方法做一些更深入的描述。

在选择使用一个 Java Servlet，还是一个 JSP 页面时，要记住的一点是，Java Servlet 是一个程序设计工具，它最适用于不需要频繁修改的低级应用功能；而 JSP 页面则通过以显示为中心的描述性方法将动态内容和逻辑结合在一起。对于使用一个 JSP 页面的简单的基于 Web 的应用程序，可以使用定制标记或者 Scriptlet，而不是使用 JavaBeans 类来将内容与应用逻辑结合起来。定制标记被打包到一个标记库中，并被引入到一个 JSP 页面中。Scriptlet 是直接嵌入在 JSP 页面中的很小的 Java 代码段。

一般来说，在实际的 JSP 引擎中，JSP 页面在执行时是编译式，而不是解释式的。解释式的动态网页开发工具(如 ASP、PHP3 等)由于速度等原因已经满足不了当前大型电子商务应用的需要，传统的开发技术都在向编译执行的方式改变，如 ASP→ASP + ; PHP3→PHP4。尽管 JSP 的规范中并没有要求实际的 JSP 引擎要使用编译式的执行方式，但一般是不会使用解释的方式来执行 JSP 页面的。通常说来，JSP 页面一般是翻译为 Servlet 的 Java 源文件，再经过 Java 编译器编译为 Servlet 的 Class 文件。为什么要编译为 Servlet 呢？据说是为了让原先的 Servlet 引擎可以直接服务于 JSP，而 JSP 引擎就仅仅需要将 JSP 转译为 Servlet 就可以了。这里要注意的是，JSP 规范中并没有规定如何将 JSP 页面转译为 Servlet，因此，不同的 JSP 引擎转译的结果也是不一样的。在 JSP 文件转译为 Servlet 以后，每次客户机(通常是用户的 Web 浏览器)向服务器请求该 JSP 文件的时候，服务器将检查自上次编译后 JSP 文件是否有改变，如果没有改变，就直接执行 Servlet，而不用再重新编译，其效率是相当高的。一般来说，JSP 文件的编译是在第一个用户访问到这个 JSP 页面时发生，而该用户通常是开发人员自己，这样，正式放在服务器上让用户访问的 JSP 文件一般都已经有了对应的编译好的 Servlet 了。许多服务器都进行了使 JSP 文件在第一个用户访问之前就预先编译好的设置，这样一来，效率就更高了。在第 4 章中，我们将展示一个简单的 JSP 文件对应的 Servlet。

在 JSP 规范中，并没有明确要求 JSP 中的程序代码部分(称为 Scriptlet)一定要用 Java 来编写，实际上，有一些 JSP 引擎采用的就是其他脚本语言，如 EMAC-Script、Web 等等。实际上这几种脚本语言也是构建在 Java 上面，编译为 Servlet 来实现的。根据 JSP 规范，完全和 Java 没有任何关系的 Scriptlet 也是可以的，不过，由于 JSP 的强大功能主要在于能和 JavaBeans、Enterprise JavaBeans 一起工作，所以即使是 Scriptlet 部分不使用 Java，编译成的执行代码也应该是与 Java 相关的。

1.2.2 JSP 的优势及与其他 Web 开发工具的比较

与传统的 CGI 相比较，JSP 有相当的优势。首先，在速度上，传统的 CGI 程序需要使用系统的标准输入输出设备来实现动态网页的生成，而 JSP 是直接和服务器相关联的。对于 CGI 来说，每一个访问就需要新增加一个进程来处理，进程不断地建立和销毁对于作为 Web 服务器的计算机将是不小的负担。其次，JSP 是专门为 Web 开发而设计的，其目的是为了建立基于 Web 的应用程序，其中包含了一整套的规范和工具。使用 JSP 技术可以很方便地将一大堆 JSP 页面组合成为一个 Web 应用程序。

与 ISAPI 和 NSAPI 相比较，JSP 的开发速度要快得多，开发难度也要小得多，在编译为 Java Servlet 以后，配合目前最新的 JIT(JustInTime)的 Java 解释器，其执行速度也慢不了多少。而且，ISAPI 和 NSAPI 这种和 Web 服务器过于紧密结合的技术在使用时一旦出现错误，很容易使 Web 服务器崩溃，而 JSP 就没有这个缺点。

JSP 的真正对手是 ASP 和 PHP，ASP(Active Server Page)是一个 Web 服务器端的开发环境，利用它可以产生和执行动态的、互动的、高性能的 Web 服务应用程序。ASP 采用脚本语言 VBScript(JavaScript)作为自己的开发语言。

PHP 是一种跨平台的服务器端的嵌入式脚本语言。它大量地借用 C、Java 和 Perl 语言的语法，并耦合 PHP 自己的特性，使 Web 开发者能够快速地编写动态地产生页面。它支持目前绝大多数数据库。另外，PHP 是完全免费的，您可以从 PHP 官方站点(http://www.php.net)自由下载。而且您可以不受限制地获得源码，甚至可以从中加进您自己需要的东西。

有人做过试验，对这三种语言分别做循环性能测试及访问 Oracle 数据库测试。在循环性能测试中，JSP 只用了令人吃惊的 4 秒钟就结束了 20000×20000 的循环。而 ASP、PHP 进行的是 2000×2000 循环测试(少一个数量级)，却分别用了 63 秒和 84 秒。数据库测试中，三者分别对 Oracle8 进行 1000 次 Insert、Update、Select 和 Delete，JSP 需要 13 秒，PHP 需要 69 秒，ASP 则需要 73 秒。

JSP 技术具有如下的特点：

1. 将内容的产生和显示进行分离

使用 JSP 技术，Web 页面开发人员可以使用 HTML 或者 XML 标识来设计和格式化最终页面。使用 JSP 标识或者小脚本来产生页面上的动态内容。产生内容的逻辑被封装在标识和 JavaBeans 群组件中，并且捆绑在小脚本中，所有的脚本在服务器端执行。如果核心逻辑被封装在标识和 Beans 中，那么其他人，如 Web 管理人员和页面设计者，能够编辑和使用 JSP 页面，而不影响内容的产生。在服务器端，JSP 引擎解释 JSP 标识，产生所请求的内容(例如，通过访问 JavaBeans 群组件，使用 JDBC 技术访问数据库)，并且将结果以 HTML(或者 XML)页面的形式发送回浏览器。这有助于作者保护自己的代码，同时确保了任何基于 HTML 的 Web 浏览器的完全可用性。

2. 强调可重用的群组件

绝大多数 JSP 页面依赖于可重用且跨平台的组件(如，JavaBeans 或者 Enterprise JavaBeans)来执行应用程序所要求的更为复杂的处理。开发人员能够共享和交换执行普通操作的组件，或者使得这些组件为更多的使用者或者用户团体所使用。基于组件的方法加速了总体开发过程，并且使得各种组织在他们现有的技能和优化结果的开发努力中得到平衡。

3. 采用标识简化页面开发

Web 页面开发人员不会都是熟悉脚本语言的程序设计人员。JSP 技术封装了许多功能，这些

功能是在易用的、与 JSP 相关的 XML 标识中进行动态内容产生所需要的。标准的 JSP 标识能够访问和实例化 JavaBeans 组件，设置或者检索群组件属性，下载 Applet，以及执行用其他方法更难于编码和耗时的功能。

通过开发定制化标识库，JSP 技术是可以扩展的。今后，第三方开发人员和其他人员可以为常用功能建立自己的标识库。这使得 Web 页面开发人员能够使用熟悉的工具以及像标识一样的执行特定功能的构件来工作。

JSP 技术很容易整合到多种应用体系结构中，以利用现存的工具和技巧，并且扩展到能够支持企业级的分布式应用。作为采用 Java 技术家族的一部分，以及 J2EE 的一个成员，JSP 技术能够支持高度复杂的基于 Web 的应用。

由于 JSP 页面的内置脚本语言是基于 Java 程序设计语言的，而且所有的 JSP 页面都被编译成为 JavaServlet，因此 JSP 页面就具有 Java 技术的所有好处，包括健壮的存储管理和安全性。

作为 Java 平台的一部分，JSP 拥有 Java 程序设计语言“一次编写，各处执行”的特点。随着越来越多的供应商将 JSP 支持加入到他们的产品中，您可以使用自己所选择的服务器和工具，来修改已有工具或服务器而不会影响目前的应用。

1.3 用 JSP 开发 Web 的几种主要方式

JSP 作为 J2EE 的一部分，既可以用于开发小型的 Web 站点，也可以用于开发大型的、企业级的应用程序，本节将讲述对于不同规模的 Web 系统，使用 JSP 进行开发的不同方式。

1.3.1 直接使用 JSP

对于最小型的 Web 站点，可以直接使用 JSP 来构建动态网页，这种站点最为简单，所需要的仅仅是简单的留言板、动态日期等基本的功能。对于这种开发模式，往往是将所有的动态处理部分都放置在 JSP 的 Scriptlet 中。

1.3.2 JSP + JavaBeans 模式

中型站点面对的是数据库查询、用户管理和小量的商业业务逻辑。对于这种站点，不能将所有的东西全部交给 JSP 页面来处理。在单纯的 JSP 中加入 JavaBeans 技术将有助于这种中型网站的开发。利用 JavaBeans，将很容易完成如数据库连接、用户登录与注销、商业业务逻辑封装等任务。例如，将常用的数据库连接写成一个 JavaBeans，既方便使用，又可以使 JSP 文件变得简单而清晰，通过封装，还可以防止一般的开发人员直接获得数据库的控制权。

本章我们主要介绍这种开发方式，在第 2 章中，将简要讲述如何开发 JavaBeans，因此，没有 JavaBeans 基础的读者不用担心。

1.3.3 JSP + JavaBeans + Servlet 模式

无论用 ASP 还是 PHP 开发动态网站，长期以来都存在一个比较棘手的问题，就是网站的逻辑关系和网站的显示页面很难分开。经常可以看见一些夹杂着 if…then…、caseselect 或 if{…}以及大量显示用的 HTML 代码的 ASP、PHP 页面，即使是有着良好编程习惯的程序员，也无法阅读这样的内容。

另一方面，动态 Web 的开发人员也在抱怨，将网站美工设计的静态页面和动态程序合并的过程是一个异常痛苦的过程。如何解决这个问题呢？在 JSP 问世以后，笔者的一位朋友认为 Servlet 已经完全可以被 JSP 代替，然而，事实是当 Servlet 不再担负动态页面生成的任务后，开

始担负起决定整个网站逻辑流程的任务。在逻辑关系异常复杂的网站中，借助于 Servlet 和 JSP 良好的交互关系和 JavaBeans 的协助，完全可以将网站的整个逻辑结构放在 Servlet 中，而将动态页面的输出放在 JSP 页面中来完成。在这种开发方式中，一个网站可以由一个或几个核心的 Servlet 来处理网站的逻辑，通过调用 JSP 页面来完成客户端(通常是 Web 浏览器)的请求。

1.3.4 J2EE 开发模型

J2EE 是建立在 Java 2 平台上的企业级应用的解决方案。J2EE 技术的基础便是 Java 2 平台，不但拥有 J2SE 平台的所有功能，同时还提供了对 EJB、Servlet、JSP、XML 等技术的全面支持，其最终目标是成为一种支持企业级应用开发的体系结构，用来简化企业解决方案的开发、部署和管理等复杂问题。事实上，J2EE 已经成为企业级开发的工业标准和首选平台。

在 J2EE 开发模型中，整个系统可以分为三个主要的部分，如图 1-5 所示。

1. 客户层

视图就是用户界面部分，在 Web 应用程序中也就是 HTML、XML 和 JSP 页面。这个部分主要处理用户看到的东西，动态的 JSP 部分处理用户可以看见的动态网页，而静态的网页则由 HTML 和 XML 输出。

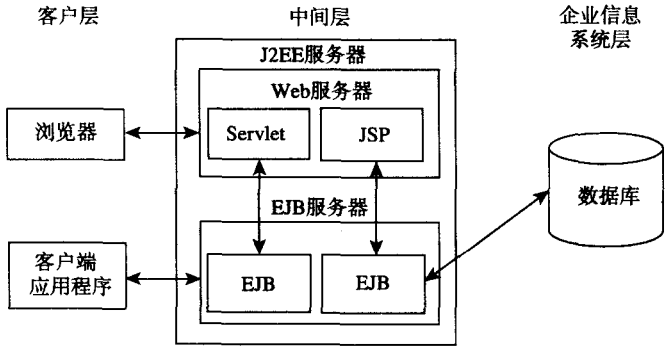

图 1-5　J2EE 开发模型的三个主要部分的关系

2. 控制器

控制器负责网站的整个逻辑，它用于管理用户与视图发生的交互。可以将控制器想象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，也就是说，网页制作人员将可以独立自由地改变 Web 页面，而不用担心影响这个基于 Web 的应用程序的功能。在 J2EE 中，控制器的功能一般是由 Servlet、JavaBeans、EJB 中的 Session Bean 来担当的。

3. 模型

模型是指应用业务逻辑部分，这一部分的主要角色是 EJB，借助于 EJB 强大的组件技术和企业级的管理控制，开发人员可以轻松地创建出可重用的业务逻辑模块。

下面通过假设一个企业应用的 J2EE 实现，来了解各种组件和服务的应用。假设应用对象是计算机产品的生产商/零售商的销售系统，这个销售系统能够通过自己的网站发布产品信息，同时也能将产品目录传送给计算机产品交易市场。销售系统能够在线接受订单(来自自己的 Web 网站或者来自计算机产品交易市场)，并随后转入内部企业管理系统进行相关的后续处理。

如图 1-6 所示，这个企业应用可以这种方式架构。该企业应用的核心是产品目录管理和产品

定购管理这两个业务逻辑，使用 EJB 加以实现，并部署在 EJB 容器中。由于产品目录和定购信息都需要持久化，因此使用 JDBC 连接数据库，并使用 JTA 来完成数据库存取事务。

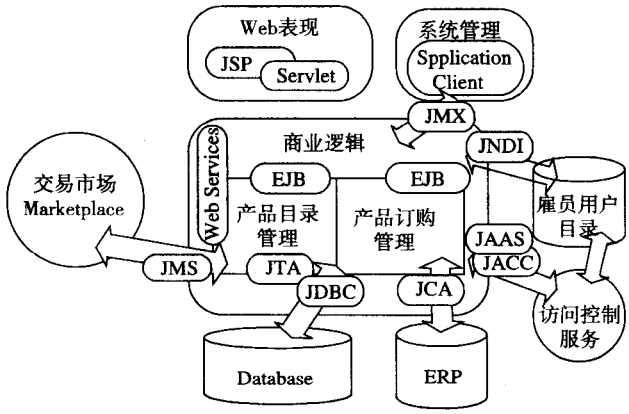

图 1-6 J2EE 应用示例

然后使用 JSP/Servlet 来实现应用的 Web 表现：在线产品目录浏览和在线定购。为了将产品目录发送给特定的交易市场，使用 JMS 实现异步的基于消息的产品目录传输。为了使得更多的其他外部交易市场能够集成产品目录和定购业务，需要使用 Web Services 技术包装商业逻辑的实现。由于产品定购管理需要由公司内部雇员进行处理，因此需要集成公司内部的用户系统和访问控制服务以方便雇员的使用，使用 JACC 集成内部的访问控制服务，使用 JNDI 集成内部的用户目录，并使用 JAAS 进行访问控制。由于产品订购事务会触发后续的企业 ERP 系统的相关操作（包括仓储、财务、生产等），需要使用 JCA 连接企业 ERP。

最后为了将这个应用纳入到企业整体的系统管理体系中去，使用 Application Client 架构了一个管理客户端（与其他企业应用管理应用部署在一台机器上），并通过 JMX 管理这个企业应用。

1.4 小结

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 JavaMediaAPI、SecurityAPI、ManagementAPI、JavaApplet、JavaRMI、JavaBeans、JavaOS、JavaServlet、JDBC、JNDI、Enterprise JavaBeans 等。Java 技术也是在不断发展中的新技术。

JavaBeans 就是 Java 的可重用组件技术。JSP 通过 JavaBean 来扩充复杂的功能，如文件上载、发送 E-mail 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。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。

J2EE 平台提供了一个基于组件的方法，用来设计、开发、配置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。采取开发模式不但降低成本，还能加快企业应用程序的设计和开发，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。EJB 就是 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。

Java Servlet 是 JSP 技术的基础，而且大型的 Web 应用程序的开发需要 JavaServlet 和 JSP 配