

# 第一部分 JSP 入门

## 第一章 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，Macrosoft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、Tandem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 Connectivity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 Bean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 transaction processing）、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 Bean 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 Bean 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：

- B基本HTML。
- B带基本JSP 页面或Servlet 的HTML。
- B带Java Beans 类的JSP页面。
- B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，并且认为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Session Beans 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 tags），这样，JSP 页面就仅包含JSP 代码了。

本书将在第二部分中讲解如何使用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你的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 sessionBeans 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 Bean 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 JDBC 2.0 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，那么就必须要在 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 JDBC 2.0 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 JDBC 2.0 API；或是用于访问一个特定企业信息系统的企业信息系统 API；或是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 Bean 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。

### 1.1.6 Java Servlet

Java Servlet 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，这里简单介绍 Servlet 的相关知识，Servlet 的开发将在第二部分讲述。

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 程序和 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个固定 Web 服务器平台运行的 API 方式来动态生成 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 的跨平台的特性，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，不必对于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。

但 Java Servlet 也不是没有缺点，和传统的 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 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 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论述，JSP 应当被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，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的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。本章我们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的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了，传统的开发技术都在向编译执行的方式改变，如ASP、ASP+；PHP3、PHP4。而尽管JSP的规范书中并没有要求实际的JSP引擎要使用编译式的执行方式，但估计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规范书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）向服务器请求这一个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，而这第一个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。许多服务器都有设置，可以使JSP文件在第一个用户访问之前就预先编译好，这样看来，效率就更高了。后面在第4章中，将展示一个简单的JSP文件对应的Servlet。

在JSP规范书中，并没有明确要求JSP中的程序代码部分（称为Scriptlet）一定要用Java来写，实际上，有一些JSP引擎就是采用的其他脚本语言，如：EMAC-Script、WebL等等，但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。按照JSP规范书，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

和传统的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。而且对于 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，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。

和ISPAI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（Just In Time）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，还有即将问世的ASP+，在Web技术方面ASP、PHP和JSP的比较见表1-1。

注意：这里的ASP指ASP3.0，JSP指JSP规范书1.1中指出的规范，PHP指PHP4。

表1-1 ASP、JSP、PHP的比较

	ASP	JSP	PHP
Web服务器	IIS、PWS	Apache、IIS、PWS、Netscape Server、iPlanet等	Apache、IIS、PWS、Netscape Server等等
运行平台	Windows	各种UNIX（Solaris、Linux、AIX、IRIX等）、Windows、MacOS等	各种UNIX（Solaris、Linux、AIX、IRIX等）、Windows
组件技术	COM	JavaBeans、EJB	COM、JavaBeans
自定义TAG语法	无	有	无
开放性	无	多家合作，包括 SUN、IBM、BEA Weblogic、Netscape、Oracle	自由软件
脚本语言支持	VBScript、JScript	Java、EMAC-Script、WebL等	PHP
建立大型Web应用程序	可以	可以	不宜
程序执行速度	快	极快	极快
学习难度	低	较低	低
Session管理	有	有	有
统一的数据库连接	有、ADO、ODBC	有、JDBC	无
后缀名	asp	jsp	php、php3、phps

### 1. Web服务器和运行平台

ASP目前仅仅被支持于Microsoft Internet Information Server（IIS）和Personal Web Server（PWS），由于IIS和PWS仅仅有Windows下的版本，故ASP目前只能在Windows平台下使用。尽管有第三方的插件号称可以在UNIX下使用ASP，但对基于COM组件技术的ASP来说，在没有COM支持的UNIX平台下只能是一个“玩具”。

JSP仅仅是一个规范，尽管通过前面的论述可以得出JSP一般要用Java来实现的论断，但作为跨平台的语言，Java可以在许多平台下使用。这样，JSP也就显而易见的是跨平台的了。目前的

JSP的确可以在多种Web服务器和操作系统下使用。如 Apache Web Server和Microsoft IIS等。

Apache Web Server是世界上占有率最高的 Web服务器产品，可以在包括 SUN Solaris、IBM AIX、SGI IRIX、Linux和Windows在内的许多操作系统下运行。Apache Web Server下JSP的实现可以通过免费的 Apache Jserv和GNUJSP、Jakarta-Tomcat实现，也可以使用商业的 JRUN (LiveSoftware)、Weblogic (BEA)、Websphere (IBM) 来实现。

Microsoft IIS本身不直接支持JSP，但可以通过JRUN、Weblogic、Websphere来实现。

还可以使用应用服务器添加JSP支持的Netscape Enterprise Server及由之发展而来的可以直接支持JSP的iPlanet Web Server等等。

PHP本身就对各种操作系统和 Web服务器做了支持，PHP目前可以作为 Apache的一个附加模块直接编译进入 Apache中去，由于 Apache支持多种操作系统，PHP相应地也就可以在各种操作系统上实现。PHP也可以CGI方式或ISAPI方式插入到IIS或PWS中去。

## 2. 组件技术

ASP和JSP对组件技术的支持已经很完善了，而PHP直到前不久才开始支持COM和JavaBeans。但支持也不是很完善，如果 PHP不能在将来完善对组件技术的支持，在大型 Web应用程序方面将很难与JSP和ASP竞争。但由于PHP技术本身的易学易用，加上众多的函数支持和开放源代码的特性，在中小型Web站点的开发上，PHP还是会占有一席之地的。

其实，JSP本身对于ASP和PHP并没有明显的优势，JSP的强大是因为其后面有强大的 Java技术做支持。包括JavaBeans和J2EE技术在内的Java技术是JSP强大生命力的所在。

Microsoft最新推出的ASP+技术和ASP技术相比有了许多激动人心的进步，但是从企业级应用的角度看，JSP技术仍然有相当的优势。有理由认为，在将来的 Web开发中，中小型站点将出现JSP、ASP+和PHP三分天下的局面，但是对于大型的电子商务站点，JSP及J2EE技术将成为首选。

## 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、动态日期等基本功能。对于这种开发模式，一般可以将所有的动态处理部分都放置在JSP的Scriptlet中，就像一般使用PHP或ASP开发动态网页一样。

### 1.3.2 JSP+JavaBeans

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、商业业务逻辑封装的任务。如：将常用的数据库连接写为一个Java Beans，既方便了使用，又可以使JSP文件简单而

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。

本书的第一部分将主要讲述这种开发方式，在第 2 章，将简要讲述如何开发 JavaBeans，没有 JavaBeans 基础的读者不用担心。

### 1.3.3 JSP+JavaBeans+Servlet

无论用 ASP 还是 PHP 开发动态网站，长期以来都有一个比较重要的问题，就是网站的逻辑关系和网站的显示页面不容易分开。常常可以看见一些夹杂着 if.....then.....、case select 或是 if {.....} 和大量显示用的 HTML 代码的 ASP、PHP 页面，即使是有着良好的程序写作习惯的程序员，其作品也几乎无法阅读。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和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 Servlet 在不再担负动态页面生成的任务以后，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，一个网站可以有一个或几个核心的 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。后面我们将可以看到，在 J2EE 模型中，Servlet 的这项功能可以被 EJB 取代。

### 1.3.4 J2EE 开发模型

在 J2EE 开发模型中，整个系统可以分为三个主要的部分：

#### 1. 视图

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

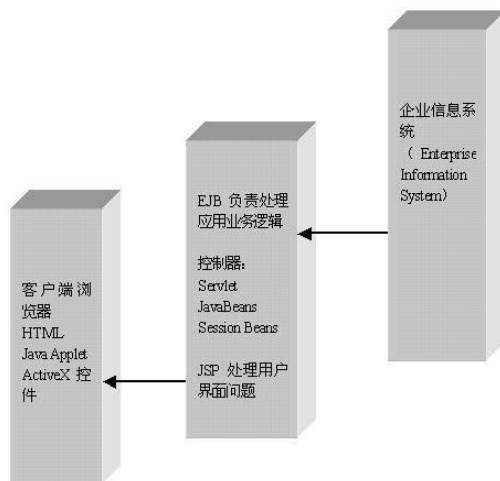


图 1-1



## 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，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、Enterprise JavaBeans 中的 SessionBean 来担当的。

## 3. 模型

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

图 1-1 说明了上述三者在 J2EE 开发模型中的相互关系。

# 1.4 本书用到的软件及获取方法

## 1. JDK

Java Development Kit 即 Java 开发工具包，进行 JSP 开发工作必须要有 JDK。本书使用版本 1.3，可以在 <http://java.sun.com> 找到。

同样，在 SUN 的主页上还可以找到一份 JDK 的中文文档，读者可能会觉得使用本地化的文档更方便。

## 2. Jakarta-Tomcat

Apache Jakarta 项目组开发的基于 GPL 自由软件协议的 JSP 引擎，配合 JDK 就可以搭建起一个最简单的 JSP 试验平台。开放源代码，使用起来比较简单，缺点是不支持 J2EE 技术，无法使用 EJB。下载地址：<http://jakarta.apache.org>。需要说明的是，原先的 GNUJSP 项目的开发人员现在已经完全转向了 Jakarta-Tomcat 项目，不建议使用 ApacheJServ+GNUJSP 搭建 JSP 平台，如果一定要使用，也可以在 <http://www.apache.org> 找到需要的软件。

## 3. Apache Web Server

仅仅使用 Tomcat 来搭建 JSP 的实用平台是不够的，一般小型站点使用 JSP 都是采用 Apache+Tomcat 的方式，因为作为独立的 Web Server，Tomcat 的效率显然还不够，也不具有 Apache 的众多实用功能。Apache Web Server 也可以在 <http://www.apache.org> 下载。

## 4. J2SDKEE

在 SUN 的主页上可以找到被称为 J2SDKEE 的 JavaTM 2 SDK，Enterprise Edition 开发工具。使用 J2SDKEE 可以用最小的开销获得一个 J2EE 的开发平台。下载地址：<http://java.sun.com/j2ee>。

## 5. IBM Websphere

要想开发基于 J2EE 技术的 Web 应用程序，最好是使用一个商业化的应用服务器（Application Server），IBM 的 Websphere 作为应用服务器在国内使用得比较多，文档和例子也不少，学习起来比较方便。IBM 公司提供免费的试用版供下载，可以在 IBM 公司的主页上找到。IBM 公司在国内也发布了不少光盘版的试用版，在各大 FTP 站点上也有下载。

## 6. BEA Weblogic

作为业界领先的应用服务器提供商，BEA Weblogic应用服务器在世界上的市场占有率遥遥领先于其他对手，不过 BEA Weblogic应用服务器的使用难度大于 IBM Websphere，而且文档几乎都是英文的，国内读者可能会觉得比较困难。BEA Weblogic应用服务器在 BEASYS 的主页上提供试用版本的下载地址为：<http://www.beasys.com>或<http://www.Weblogic.com>。

#### 7. UltraEdit

UltraEdit并不是一个Java工具，但笔者认为它是一个比较好的JSP写作工具，在UltraEdit的主页上可以得到UltraEdit的试用版和支持Java2和JSP的语法文件地址为：<http://www.ultraedit.com>。

读者亦可以从其他的JSP引擎入手学习JSP，如LiveSoftware JRUN、Inprise Application Server、ServletExec等等，也有许多好的Java开发工具和文本编辑器可以使用，如：Inprise Jbuilder 3.5、IBM Visual Age For Java、Symantec Visual Café、EditPlus、JavaPad等等。

最后说明一下本书的代码写作规范：

类名：类名由几个单词构成时，将这几个单词紧靠在一起，且大写每一个单词的首字母。

其他：和类名相仿，但是第一个单词的首字母小写。

每个程序块统一缩进一个Tab制表符。但是对于那种长代码较多的代码段，统一取消缩进。

注释单独列为一行。

每行代码长度不超过60个字符（为了出版的需要）。

如：

```
class TestClass{
    int a;
    void TestFuncs() {
        .....
    }
}
```