Professional Java Servlets 2.3

Java Servlets 2.3

编程指南

〔美〕 John Bell
Tony Loton 等著

马树奇 等译

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

wrox 乐思
PROGRAMMER TO

Professional Java Servlets 2.3

Java Servlets 2.3 编程指南

〔美〕 John Bell, Tony Loton 等著

马树奇 等译

電子工業出版社

Publishing House of Electronics Industry

北京 · BEIJING

内 容 提 要

本书旨在介绍如何使用Java Servlets技术生成功能强大并且可以移植的企业应用程序组件，如何使用servlet来控制应用程序运行的流程、跟踪应用程序用户、截获及修改请求和响应信息，以及如何与Web服务交互。

本书适用于熟悉Java语言及Java核心API的开发者，是美国多位Java技术专家的经验总结与成果汇集。

版权贸易合同登记号：01-2002-0170

图书在版编目（CIP）数据

Java Servlets 2.3编程指南/（美）贝尔（Bell，J.）等著；马树奇等译. - 北京：电子工业出版社，2002.7

书名原文：Professional Java Servlets 2.3

ISBN 7-5053-7738-8

Ⅰ. J… Ⅱ. ①贝… ②马… Ⅲ. Java语言—程序设计 Ⅳ. TP312

中国版本图书馆CIP数据核字（2002）第043643号

责任编辑：李 莹

印 刷：北京天竺颖华印刷厂

出版发行：电子工业出版社 http://www.phei.com.cn

北京市海淀区万寿路173信箱 邮编：100036

北京市海淀区翠微东里甲2号 邮编：100036

经 销：各地新华书店

开 本：787×1092 1/16 印张：37.75 字数：970千字

版 次：2002年7月第1版 2002年7月第1次印刷

定 价：56.00元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换，若书店售缺，请与本社发行部联系。联系电话：（010）68279077

序

欢迎学习专业Java Servlets 2.3。本书旨在介绍如何使用Java Servlets技术生成功能强大并且可以移植的企业应用程序组件，以及如何在Web应用程序中应用。

近些年来，人们通常把分布式业务服务中的交互活动放在Web层，并且使用servlet作为Web应用程序的动力引擎和组件框架。为此，我们将介绍如何使用servlet来控制应用程序运行的流程、跟踪应用程序用户、截获及修改请求和响应信息，以及与Web服务交互。

随着Servlet技术规范2.3版的开发（最终于2001年9月完成），专家组进行了一系列的修改，其中包括：

- 添加过滤处理
- 溶入侦听或者存活期事件（lifecycle）
- 使用J2SE 1.2作为Web容器的基层系统平台
- 在技术规范中采用Javadoc API定义
- 提高国际化处理水平
- 增强Java Archive（JAR）相关性
- 改善类装入器
- 还有其他许多修改，如一种新型经过改善的错误属性，以及会影响HTTPS请求的新的安全属性
- 引入新类和方法的一些变化，并且不建议使用javax.servlet.http.HttpUtils类

Servlets 2.3 API技术规范也是Java 2 Platform、Enterprise Edition（J2EE）1.3技术规范中的一个关键部分，在本书中你将会看到，它在J2EE系统平台的应用程序中作为控制器扮演着非常重要的角色。

随着内容的展开，我们将详细介绍这些概念，同时使用充分而完整的工作示例来说明它们的使用。

本书适合的对象

本书针对的是已经熟悉了Java语言及Java核心API的开发者。同时，本书假设读者已经熟悉了HTML和XML的一些基本知识，但这并不是基本要求。我们将使用最新版本的技术规范，这就是Java Servlets 2.3版。

Servlet很少独立使用，而本书也并不宣称这里就覆盖了所有领域的各种情况，特别是与其他Java技术及API如JDBC、JNDI和JavaServer Pages相关的领域。读者可以参考Wrox Press公司出版的《Professional Java Server Programming J2EE 1.3 Edition》（ISBN 1-861005-37-7），该书对于整个J2EE系统平台进行了出色的介绍。

本书介绍的内容

本书的结构如下：

- 第1章综述如何把servlet融合到企业应用中，以及该把它们用于什么场合。
- 第2章和第3章介绍Servlet 2.3 API。我们将学习servlet的存活期，并且理解如何使用及生成HTTP请求和响应信息。
- 尽管我们在前面的几章中也会运行一些应用程序，但第4章才介绍Web应用程序的结构以及如何在一个Web服务器上部署它们。
- 第5章～第7章介绍servlet的一些强大特性，包括如何维护会话、如何保持servlet以及什么是过滤器（filter）。
- 第8章介绍JavaServer Pages（Java服务器页面，JSP），这是一种对Java Servlets构成有益补充的技术。
- 第9章～第11章介绍在生产环境中部署Web应用程序时将会遇到的一些问题，以及跟踪servlet中出现问题的一些调试技术，同时了解如果不考虑类装入和同步处理的效果可能出现什么问题。
- 第12章和第13章介绍Web应用程序设计对系统性能和可维护性会产生怎样的影响。我们将研究各种不同的模式来生成更好的应用程序，介绍一些技术和工具，以便提高Web应用程序的性能和扩展能力。
- 最后一章也就是第14章介绍如何使用servlet作为代理程序访问来自Web服务（web service）的信息。

学习本书内容需要什么条件

本书中的大部分代码都已经用Java 2 SDK 1.3版（此SDK可以从网址http://java.sun.com/j2se/1.3/下载）和Apache Tomcat 4（可以从http://jakarta.apache.org/tomcat/下载）测试过。但是运行一些章节的示例的时候还需要额外一些软件。

本书中有几章的内容需要访问数据库。在这些章节中我们使用的是MySQL（版本3.23）以及MM.MySQL JDBC驱动程序（版本2.0.6）。可以从网址http://www.mysql.com下载上述两个软件。下载的文件中包含有完整的安装指导。

有几章还需要其他一些软件：

- Java 2 SDK 1.4版——可以从网址http://java.sun.com/j2se/1.4/下载
- The JavaBeans Activation Framework——可以从网址http://java.sun.com/products/javabeans/glasgow/jaf.html下载
- XML Parsers: 我们已经使用了Apache Xerces——可以从http://xml.apache.org/xerces-j/index.html下载，以及Apache Xalan XSLT处理器——可以从http://xml.apache.org/xalan-j/下载

- Apache SOAP 2.2——可以从http://xml.apache.org/soap/index.html下载
- Apache AXIS——可以从http://xml.apache.org/axis/index.html下载

本书中的代码可以在一台连网的计算机中运行（也就是说，可以通过本地浏览器从http://localhost/上查看）。

本书中的全部源代码可以从下列网址下载：

http://www.wrox.com/

约定

为了帮助读者从文本内容中获取更多的知识并且跟上内容的发展，我们在本书中使用了一系列的约定。

例如：

用黑体字标出重要的、不能忘记的信息，这些内容直接与周围的文本相关。

用这种楷体字辅助说明当前的讨论内容。

我们还以不同的方式显示代码。方法的定义和属性显示样式如下：

```
protected void doGet(HttpServletRequest req, HttpServletResponse resp)
                    throws ServletException, IOException
```

示例代码的显示样式如下：

```
In our code examples, the code foreground style shows new, important,
    pertinent code
while code background shows code that's less important in the present context,
    or has been seen before.
```

客户支持

我们非常珍视来自读者的意见，希望了解你对本书有什么想法——喜欢什么、不喜欢什么、想的是什么——这些对我们今后的工作都会有好处。你可以把自己的意见通过电子邮件发送给feedback@wrox.com。请注意在电子邮件消息中列出本书的标题。

如何下载本书的示例代码

访问Wrox站点http://www.wrox.com/时，只需简单地通过Search（搜索）机制从书目列表中找出自己需要的名称。点击Code列中的Download链接或者在相关书目的详细信息页面上点击Download Code即可进行下载。

在我们的站点上可供下载的文件均经过WinZip进行了压缩。把附件存储在计算机硬盘上的文件夹中时，需要使用一个解压缩程序如WinZip或PKUnzip来展开这些文件。展开这些文件之后，各章的代码通常会展开到各章对应的文件夹中。当开始展开处理时，应保证自己

的软件（WinZip、PKUnzip等）使用了对应的文件夹名字。

勘误

我们已经尽了最大的努力来保证在内容或者代码中尽可能没有错误。但是人无完人，错误总难避免。如果你发现我们的书中有什么错误，如代码错误，请给予我们相关的反馈消息，我们将不胜感激。因为这些勘误信息足可以为其他读者节省数小时以解决可能遇到的困难。当然你也是在帮助我们提供更高质量的信息。只要给地址support@wrox.com发送电子邮件即可，你的信息将得到审核，如果正确，我们会登出相关页的勘误信息，或者在本书后续版本中使用。

如果想在Web站点中查看勘误信息，请进入网址http://www.wrox.com/，简单地通过我们的Advanced Search或者标题列表找出相关书目，点击位于相关书目详细页面封面图形下面的Book Errata链接即可找到。

电子邮件支持

如果你想直接就本书中的一些问题询问了解本书详细情况的专家，可以发电子邮件到support@wrox.com，列出书名以及ISBN中的最后4个编号。典型的电子邮件如下所示：

- 在Subject（主题）字段中列出书名、ISBN的后4位，有问题的页码。
- 你的姓名、联系信息以及问题消息正文。

我们不会给你发垃圾邮件。我们需要这些详细信息是为了节省你我双方的时间。当你发送一封电子邮件消息时，该消息会经过下列支持过程：

- 客户支持（Customer Support）——首先把消息送到我们的客户支持技术人员手中，他们拥有最经常问及情况的文件，将回答关于本书或者Web站点等直接问题。
- 编辑（Editorial）——比较深入的问题会转发给负责本书相关事务的技术编辑。他们具有编程经验或者关于特定产品的经验，能够回答相关主题的详细技术问题。
- 作者（Author）——最后，当出现了编辑无法回答的问题时（当然这种情况极少出现），他们会把相关请求转给作者。我们会保护作者在写作过程中不受任何打扰，但我们也很愿意把有价值的信息转给他们。Wrox公司的所有作者都会为其著作的技术支持提供帮助。他们将通过电子邮件与客户及编辑联系，发出响应信息，以使所有读者受益。

Wrox公司的技术支持过程只对直接与我们出版的书籍有关的问题做出解答。对于不属于此类范围的问题，可以从我们的http://p2p.wrox.com/论坛获取支持。

p2p.wrox.com

如果想与作者进行对等讨论可以加入P2P邮件群。我们这个系统提供了程序员到程序员（programmer to programmer™）之间的联系，除了我们的一对一电子邮件支持系统以外，可以通过邮件群、论坛和新闻组来实现。如果你向P2P发出一个查询，那么请相信此消息肯定会被许多Wrox作者及其他业界专家看到，他们都参与我们的邮件群。在p2p.wrox.com，有许多不同的邮件群可以提供帮助，不论你是在阅读本书，还是在开发自己的应用程序。特别针对

本书的邮件群是j2ee和pro_java_server。

如果想订阅这样的邮件群，可以按照下列步骤操作：

1. 进入http://p2p.wrox.com/
2. 从左侧菜单栏中选择适当的分类
3. 点击你想要加入的邮件群
4. 遵照其中的指示进行订阅，并且填写你自己的电子邮件地址和口令
5. 对收到的要求确认的电子邮件进行答复
6. 使用订阅管理器来加入更多的邮件群以及设置你自己的电子邮件首选项

我们会提供最好的支持

你可以选择加入邮件群或者接收他们的每周摘要。如果你没有时间、设施来接收邮件群，那么可以搜索我们的在线档案。垃圾邮件会被删除，你自己的电子邮件地址由专门的Lyris系统保护。关于加入或者离开邮件群的指示及其他一般邮件群查询，可以发电子邮件给listsupport@p2p.wrox.com。

目　录

第1章　企业中的servlet

Java是一种成熟的技术，可以按照不同的系统平台来划分。目前比较流行的是Java 2 Platform, Standard Edition（Java 2系统平台标准版，J2SE），它提供了核心的Java API和Java Virtual Machine（Java虚拟机，JVM），以及一些开发工具，如Java编译器等。但是本书中我们将关注Java 2 Platform, Enterprise Edition（Java 2系统平台企业版，J2EE）。实际上，我们只集中研究J2EE的一部分，这就是Java Servlet 2.3 API。

J2EE建立在J2SE的基础上，为开发和部署企业应用程序提供API和服务。将J2SE和J2EE的服务和库结合在一起有助于开发独立于系统平台、基于Web的Java应用程序。J2EE系统平台还有助于服务器厂商提供相关的环境来部署和运行J2EE应用程序。

- 在本章中，我们首先考虑分布式应用中的企业结构以及Web应用程序开发中涉及到的各层。
- 接着，观察J2EE结构怎样能够通过Web容器和Java servlet为Web开发提供便利，并且讨论servlet开发者希望在其应用程序中包含的J2EE服务和库。
- 讨论在Web应用程序中使用遵守Servlet 2.3技术规范的Java servlet会有哪些优点。
- 讨论容器厂商和开发者在实现Servlet技术规范时扮演的角色，介绍Tomcat Web容器/服务器，这两者是Servlet技术规范的参考实现。
- 在本章结束的时候，我们将观察在现代企业应用程序中servlet需要扮演的角色，以此说明它们在企业应用程序开发过程中的相关性和重要性。

但在我们详细介绍servlet之前，先来从较高的层面观察一下企业应用程序的结构。

企业应用程序的结构

当今企业应用程序的结构和基础设施是千差万别的。一个企业系统可能包含1960年延续过来的老式大型机，并且与现代系统配合使用。在过去的10年间，网络和因特网已经把老式系统与现代系统集成在一起了。

企业中之所以要包含老式系统是为了避免把这些系统上的核心业务处理转移到现代系统时花费大量的资金。但是由于需要继续维护老式系统、引入新系统以及把两者与来自其他机构的系统合并起来，因此企业系统变得非常复杂。

我们来考虑一家金融服务公司，它已经存在30年了。30年前，该公司把其核心业务处理定义在一台大型机上，使用COBOL语言完成。多年以后，该公司经过了多次收购与合并，后来加入的其他那些公司的业务处理也与原有的系统集成起来。这些业务都采用当前最先进的技术开发出了相关应用程序。结果是，现在该公司有了一个非常复杂的结构，把许多不同的硬件和软件系统链接在一起。当然，这些系统不仅拥有内部链接，还有通向Web的外部链接。

Web所在的分布式环境可供职员们在自己办公室里的一台PC机前工作，与互连的任意其他系统或者资源交互，既可以在业务范围内，也可以在业务范围之外。外部系统（例如供货商的系统）也包含在网络中，并且以此与企业员工和系统通过Web通信。这就提出一个问题，只有建立一种公共语言才能使这些系统相互通信。

网络和协议

在开发分布式计算系统的时候，基层的物理连接是建立这些系统的基础。如果去掉这些网络，分布式应用程序将失去其大部分的使用价值。基本的网络结构对开发者经常是透明的。Java语言的“write once, run anywhere（只编写一次，可以在任何地方应用）”这种思想提供一种强大的网络API工具，使得开发分布式应用程序更容易，也使我们能够了解进行开发的网络拓扑。当我们开始在分布式环境中处理servlet的时候，应该首先简单观察一下计算机之间的物理连接以及系统链路。

在一个网络系统中，所有彼此链接的系统都是互连的，要么通过有线网络，要么通过无线网络。网络就是计算机系统及其上运行的软件之间的一个通信链路。在图1.1中显示了三种基本的网络拓扑：

- 星型（star）拓扑用于把多台计算机连接到一个中心点，这个中心点经常称为集线器（hub）
- 环型（ring）拓扑用于把多台计算机连接成一个封闭的环形，在这里每一台计算机都与接下来的计算机连接起来，直到形成一个封闭的环
- 总线型（bus）拓扑则把多台计算机连接到一条共享的介质上，整个系统通过这条线路彼此通信

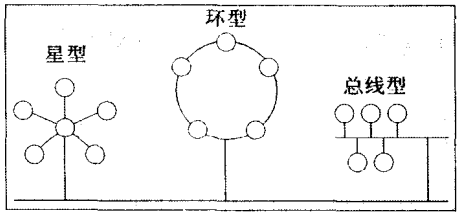

图1.1

当然，这些并不是在网络中可以实现的拓扑结构的全部。上述拓扑主要用于局域网（LAN）中。它们是最普通的结构，大多数其他备选的拓扑则是这几种型式的衍生或者结合。随着新的网络技术和通信技术的发展，如蓝牙（Bluetooth）技术，企业网络的拓扑还会变得更复杂。要求网络（以及网络管理员）支持日新月异的技术也是一个持续不断的挑战。

网络被设计成一种通信渠道，可供不同的以及在其他方面不兼容的系统能够连接到相同的网络并且进行通信。我们都知道自己的网络怎样在相同的网络中包含不同的操作系统和不同的设备。我们可以使用一个浏览器来与一台服务器通信，而该服务器可能运行在完全不同的另外一种操作系统上。因此，问题是不兼容的系统如何彼此通信？

任何资源都可以加入网络中，只要它们能够使用网络允许的协议进行通信即可。

协议（protocol）就是一组双方达成一致的通信规则。人们开发了许多协议，用来规定公共标准和消息格式，使不同的系统能够交换信息和数据。这些协议被设计为提供特定的服务，并且采用了分层结构以便提供（相对）可靠的网络服务，如图1.2所示。

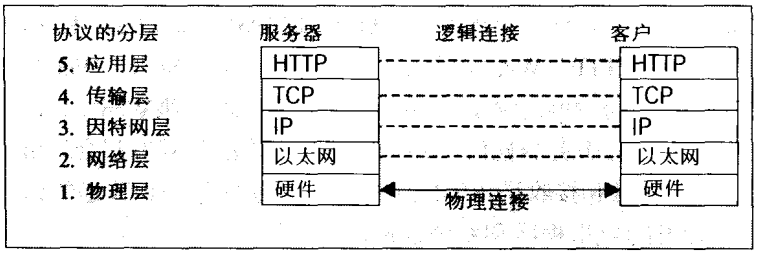

图1.2

在物理层，硬件组成的物理连接出现在彼此通信的计算机之间。这一部分基本上利用网卡和网线（或者无线）来连接系统，为计算机提供传输消息的介质。

在网络层，网络实现了诸如以太网（Ethernet）这样的协议以便实现计算机之间的通信。这一部分负责把数据分成数据帧，并且通过网络进行传送。它还定义了诸如要发送的数据容量、进行数据拆分和组装较大消息的格式以及如何处理网络故障等事宜。

从一个系统发送到另一个系统的消息包装在这种分层的协议中，然后通过网络发送到另一个系统。接收这些消息的系统会从协议的各层中解开相关的消息。从逻辑上看，每个对应的层都与另一计算机上对应的层进行通信。这些层中的每一层在最基本的水平上提供了系统之间的桥梁，位于其上方的协议层则为它们提供附加的服务。

这种情况有点像一种可靠形式的翻译，即顶层采用自己的语言进行通信，但这些消息必须被翻译成一系列中间语言或者编码才能进行交换。接收者接着把此消息通过一系列预定的翻译器和解码器发送过来，直到此回复消息再次被翻译成发送者使用的语言。

TCP/IP

Transmission Control Protocol（传输控制协议，TCP）和Internet Protocol（网际互连协议，IP）层一般被合在一起，因为这两种协议提供了互补的服务。TCP/IP协议并不是惟一可以在这些层上使用的协议，但是随着Web的发展，它们已经成为Web上进行通信的标准协议。

Internet Protocol定义了数据块（也称为分组，即packet）如何进行格式化，还定义了把这些分组传递到其目的地的机制。这项协议使用互连计算机的IP地址来对网络上传输的数据进行路由选择。这是一种比较不可靠的办法，因为数据可能会丢失或者到达目的地的顺序可能会乱。Transmission Control Protocol则为应用层提供了一种面向连接的通信服务。这个层提供了其下面的IP层所缺少的可靠性服务，可以保证所有的数据都会被接收到（如果必要时可以重新发送），并且按照正确的顺序组装起来。

TCP/IP协议结合起来提供了一种可靠的服务，其中的各层各负其责地提供了可靠的连接，供其上的应用层使用。

HTTP

通信中的应用层通常由HTTP即Hypertext Transfer Protocol（超文本传输协议）提供。也可以使用其他协议，如电子邮件协议POP3、SMTP和IMAP，以及File Transfer Protocol（文件传输协议，FTP），但HTTP是Web应用程序中占主导地位的应用。Web容器和J2EE应用程序都需要支持HTTP作为请求和响应协议。由于这是最普通的顶级Web协议，因此应用程序通过使用此协议可以与大部分服务器通信，并且保证其中的消息双方都会理解。

HTTP提供了一个发送和接收请求的既定格式，并且作为一种普通的语言可以被不同系统上使用不同语言开发的应用程序和系统理解。

为什么不使用远程过程调用

Remote Procedure Calls（远程过程调用，RPC）是一种可供客户向服务器上的一个程序发出特定请求的机制（根据需要传递过去必要的变量）。服务器接着会把结果返回给发出相应请求的客户。发出调用请求的客户必须遵守预定的格式。

在Java中，RPC是在Remote Method Invocation（远程方法调用，RMI）中实现的，通过这种方式，Java应用程序可以根据为RMI方法调用定义的过程来调用远程服务器上的一个类。实现手段是使用一个接口，在这个接口上定义想要调用的远程对象。接着由系统向相关远程对像做出相应的请求，在接口的帮助下像对待一个本地对象一样进行处理。Java使用厂商提供的自定义协议实现来传递这些请求。Sun公司使用建立在TCP/IP上的Java Remote Method Protocol。另外RMI方法调用也可以基于IIOP（Internet Inter-Orb Protocol）在Java应用程序之间实现（RMI-IIOP），这是一种独立于编程语言的技术，允许使用一个远程接口与任何兼容对象请求代理（object request broker，ORB）进行交互。

CORBA（Common Object Broker Request Architecture，普通对象代理请求结构）提供了一种独立于语言的机制，用于调用远程应用程序上的方法或者过程。

RMI和CORBA在具体实现上比HTTP要复杂得多，这也使得它们不适合于许多Web应用程序。HTTP是分布式环境中理所当然的选择，并且对于主要的Web应用程序而言（包括所有类型的客户），它相当出色，具有最好的可靠性和灵活性，因此也得到了最广泛的支持。

HTTP和RPC都要求客户和服务器理解共同的请求－响应过程。但是，如果更新了RPC类型的请求中包含的类，那么就经常需要更新客户和服务器类，即使相关改变只影响到其中一方。有了HTTP，只要双方同意请求的格式，服务器更新就不需要客户方进行任何修改（反之亦然）。

HTTP是Web上的标准语言，比其他协议得到了更多的服务器和客户的支持。因此它成为大部分Web和servlet开发中使用的理想协议。

分层结构

在老式的客户－服务器两层结构开发模型中，客户应用程序直接与数据源连接，这种模型目前已经在很大程度上被取代了。企业应用程序正在朝更流行的多层应用程序发展，这种程序的结构扩展到了三层以上。应用程序在逻辑部分上分离与合并成许多不同的层，这样做具有许多重要的优点：

- 通过把相关功能模块化成特定的层，使相关规则和功能封装在一起，从而获得更好的维护性，也利于开发
- 模块化还在基于组件的开发中提高了业务处理、信息表示以及其他逻辑处理的灵活性和重用性
- 具有特定技术水平的开发者可以把注意力集中于特定层次的逻辑问题上（例如数据库专家可以集中处理数据库层的问题），而各层之间的合同定义了各层之间的关系和它们可以从其他层得到什么服务

三层结构的示意图如图1.3所示。

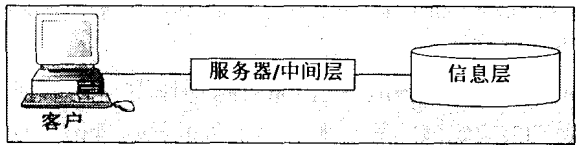

图1.3

客户层（client tier）是为用户描述或者显示数据及信息的地方。

服务器（server）或者中间层（middle tier）负责对客户的数据和/或操作请求做出响应。这里通常有核心的应用处理逻辑。但是，中间层经常又被分成两个不同的层：

- 业务层（business tier）负责保存业务处理逻辑。这一部分关注的是忠实地实现应用程序的业务规则。
- Web层（web tier或表示层，即presentation tier）则保留用来表示或包装客户业务数据。该层对客户请求做出响应，并且把它们转发到业务层，由业务层应用业务处理逻辑来处理请求。

业务层将返回请求得到的结果（数据或其他响应信息），并且Web层将为客户准备响应信息，如图1.4所示。

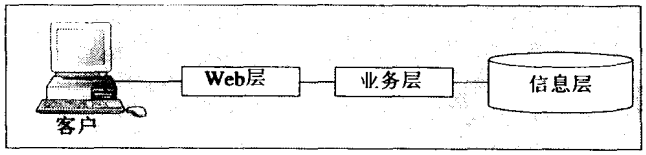

图1.4

本书主要讨论在Web应用程序中进行的servlet开发工作，这种servlet工作蕴含在应用程序模型的Web层中。