HZ BOOKS
华章科技

BrainySoftware

华章专业开发者丛书

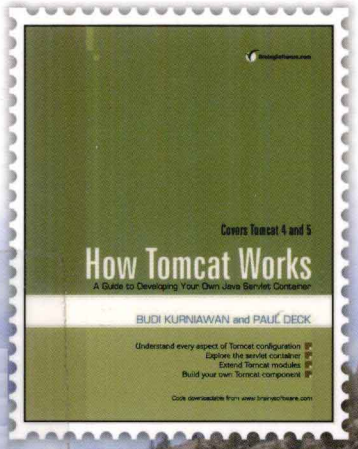
深入理解Tomcat的必读之作
欲知其然，更欲知其所以然

深入剖析 Tomcat

How Tomcat Works
A Guide to Developing Your Own Java Servlet Container

（美） Budi Kurniawan
Paul Deck 著

曹旭东 译



机械工业出版社
China Machine Press

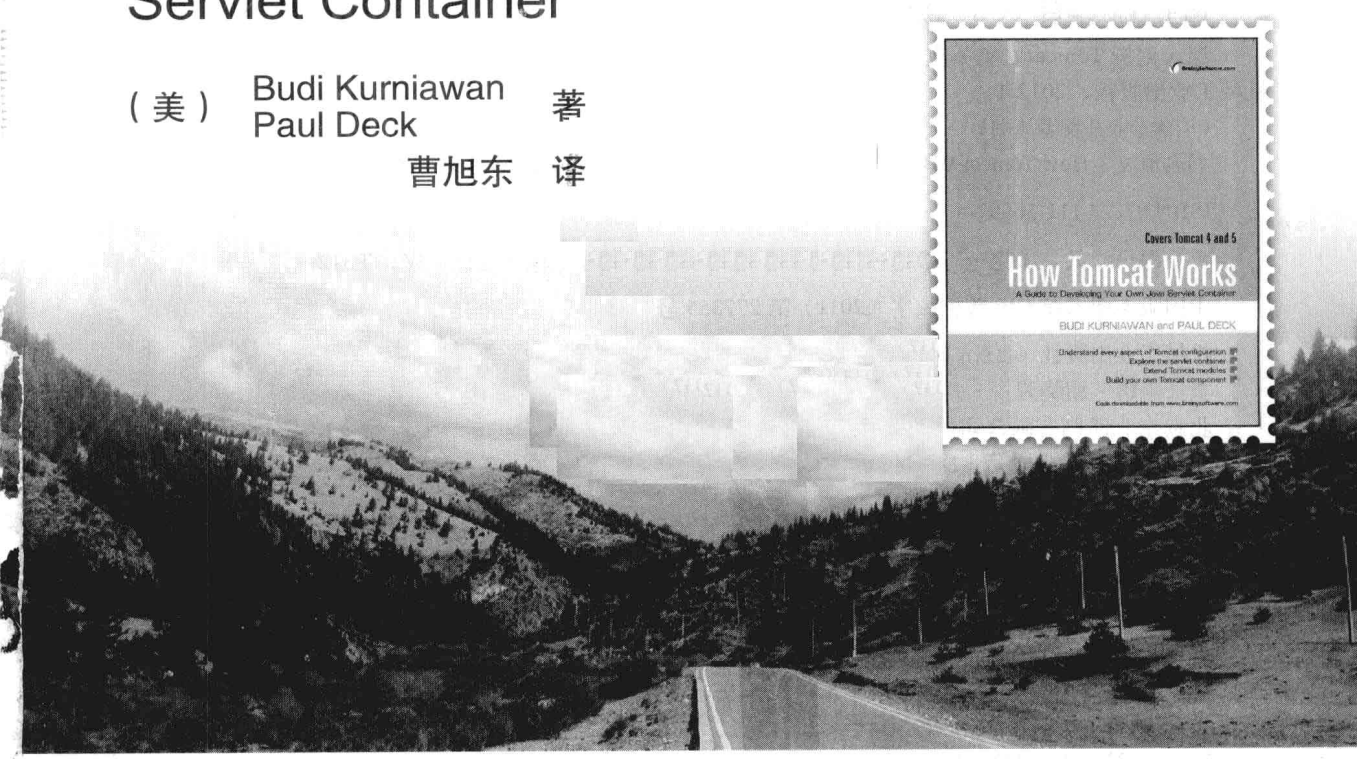
华章专业开发者丛书

深入剖析 Tomcat

How Tomcat Works
A Guide to Developing Your Own Java Servlet Container

（美） Budi Kurniawan
Paul Deck 著

曹旭东 译



机械工业出版社
China Machine Press

本书深入剖析 Tomcat 4 和 Tomcat 5 中的每个组件，并揭示其内部工作原理。通过学习本书，你将可以自行开发 Tomcat 组件，或者扩展已有的组件。Tomcat 是目前比较流行的 Web 服务器之一。作为一个开源和小型的轻量级应用服务器，Tomcat 易于使用，便于部署，但 Tomcat 本身是一个非常复杂的系统，包含了很多功能模块。这些功能模块构成了 Tomcat 的核心结构。本书从最基本的 HTTP 请求开始，直至使用 JMX 技术管理 Tomcat 中的应用程序，逐一剖析 Tomcat 的基本功能模块，并配以示例代码，使读者可以逐步实现自己的 Web 服务器。

本书版权登记号：图字：01-2011-7874

图书在版编目（CIP）数据

深入剖析 Tomcat /（美）克尼亚万（Kurniawan, B.），（美）德克（Deck, P.）著；曹旭东译. —北京：机械工业出版社，2012.1

（华章专业开发者丛书）

书名原文：How Tomcat Works：A Guide to Developing Your Own Java Servlet Container

ISBN 978-7-111-36997-4

Ⅰ. 深… Ⅱ. ①克… ②德… ③曹… Ⅲ. JAVA 语言－程序设计 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2011）第 277355 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：谢晓芳

北京京北印刷有限公司印刷

2012 年 2 月第 1 版第 1 次印刷

186mm×240mm • 22.25 印张

标准书号：ISBN 978-7-111-36997-4

定价：59.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：（010）88378991；88361066

购书热线：（010）68326294；88376949；68995259

投稿热线：（010）88379604

读者信箱：hzjsj@hzbook.com

译 者 序

Tomcat 是 Apache 软件基金会（Apache Software Foundation）的一个顶级项目，由 Apache、Sun 和其他一些公司及个人共同开发，是目前比较流行的 Web 服务器之一。作为一个开源的、小型的轻量级应用服务器，Tomcat 深受广大程序员的喜爱，具有占用系统资源少，扩展性好，支持负载平衡与邮件服务等开发应用系统常用的功能；而且它还在不断地改进和完善中，任何一个感兴趣的程序员都可以更改它或在其中加入新的功能。

虽然 Tomcat 易于使用，便于部署，但 Tomcat 本身是一个非常复杂的系统，包含了很多功能模块。这些功能模块密切合作，各司其职，构成了 Tomcat 的核心结构。作者从最基本的 HTTP 请求开始，直至使用 JMX 技术管理 Tomcat 中的应用程序，逐步深入，逐一剖析 Tomcat 的基本功能模块，并配以示例代码，使读者可以逐步实现自己的一个 Web 服务器。

当然，本书并不能完全覆盖 Tomcat，书中并没有包含 Tomcat 的太多设计思路及具体的实现细节，而是更注重对 Tomcat 基本结构的分析介绍。在每一章中都有与本章内容相关的示例应用程序，帮助读者更好地理解该章的内容。

本书由曹旭东翻译，由于时间仓促，加上译者水平有限，书中难免有疏漏之处，望广大读者予以指正。

曹旭东

前　言

欢迎阅读本书。本书剖析了 Tomcat 4.1.12 版本和 Tomcat 5.0.18 版本的基本结构，并解释了其 servlet 容器 Catalina 的内部工作原理。Catalina 是开源、免费的，也是最受欢迎的 servlet 容器之一。Tomcat 本身是一个复杂的系统，包含了许多不同的组件。若你想学习 Tomcat 的工作方式，应该从了解这些组件开始。本书描述 Tomcat 的总体结构，并针对每个组件建立一个简单的版本，使你更好地理解组件的工作机制，之后对真实组件进行描述。

“本书结构”一节会对全书的章节设置做一个总体介绍，并说明构建的应用程序的总体结构。在“准备必需的软件”一节，说明需要下载使用哪些软件，如何为代码创建目录结构等。

本书读者对象

本书适合于所有使用 Java 技术工作的开发人员。

- 如果你是一名 JSP/Servlet 程序员或 Tomcat 用户，并想了解 servlet 容器是如何工作的，那么本书很适合你；
- 如果你想加入 Tomcat 开发团队，那么本书很适合你，因为你首先要学习已有的代码是如何工作的；
- 如果你不是一名 Web 开发人员，但对软件开发很有兴趣，那么你可以从本书中学习到一个大型应用软件（如 Tomcat）是如何设计和开发的；
- 如果你想对 Tomcat 进行配置或定制，你应该阅读本书。

为了更好地理解本书所讲述的内容，你需要理解 Java 中的面向对象编程知识，以及 Servlet 编程方面的知识。如果你对后者还不熟悉，那么你学习起来可能会有些困难。你可以先学习一下 Servlet 编程方面的知识，例如看一下 Budi 的《Java for the Web with Servlets, JSP, and EJB》一书。为了使你更好地理解本书的内容，每一章的开头都会有一段与该章内容相关的背景信息的介绍。

servlet 容器是如何工作的

servlet 容器是一个复杂的系统，但是，它有 3 个基本任务，对每个请求，servlet 容器会为其完成以下 3 个操作：

- 创建一个 request 对象，用可能会在调用的 Servlet 中使用到的信息填充该 request 对象，如参数、头、cookie、查询字符串、URI 等。reqeust 对象是 javax.servlet.ServletRequest 接口或 javax.servlet.http.ServletRequest 接口的一个实例。
- 创建一个调用 Servlet 的 response 对象，用来向 Web 客户端发送响应。response 对象是 javax.servlet.ServletResponse 接口或 javax.servlet.http.ServletResponse 接口的一个实例；

• 调用 Servlet 的 service() 方法，将 request 对象和 response 对象作为参数传入。Servlet 从 request 对象中读取信息，并通过 response 对象发送响应信息。

当你阅读具体的章节时，你会看到关于 servlet 容器 Catalina 的详细描述。

Catalina 框图

Catalina 是一个成熟的软件，设计和开发得十分优雅，功能结构也是模块化的。上一节“servlet 容器是如何工作的”中提到了 servlet 容器的任务，基于这些任务可以将 Catalina 划分为两个模块：连接器（connector）和容器（container）。

图 I-1 很简单，在后续的章节中，你会逐个接触到所有的组件。

图 I-1 Catalina 的主要模块

现在，回到图 I-1，这里的连接器负责将一个请求与容器相关联。它的工作包括为它接收到的每个 HTTP 请求创建一个 request 对象和一个 response 对象。然后，它将处理过程交给容器。容器从连接器中接收到 request 对象和 response 对象，并负责调用相应的 Servlet 的 service() 方法。

但是请记住，上面所描述的处理过程只是 Catalina 容器处理请求的整个过程的一小部分，犹如冰山的一角，在容器中还包括很多其他的事情要做。例如，在容器调用相应的 Servlet 的 service() 方法之前，它必须先载入该 Servlet 类，对用户进行身份验证（如果有必要的话），为用户更新会话信息等。因此，当你发现容器使用了很多不同的模块来处理这些事情时，请不要太惊讶。例如，管理器模块用来处理用户会话信息，载入器模块用来载入所需的 Servlet 类等。

Tomcat 4 和 Tomcat 5

本书涵盖了 Tomcat 4 和 Tomcat 5 两个版本。下面是这两个版本的一些区别之处：

• Tomcat 5 支持 Servlet 2.4 和 JSP 2.0 规范，Tomcat 4 支持 Servlet 2.3 和 JSP 1.2 规范；

• Tomcat 5 默认的连接器比 Tomcat 4 默认的连接器执行效率更高；

• Tomcat 5 使用共享线程来执行后台任务，而 Tomcat 4 的组件使用各自的线程执行后台任务，因此，相比于 Tomcat 4，Tomcat 5 更节省资源；

• Tomcat 5 不再使用映射器组件来查找子组件，因此，代码更简单。

本书结构

本书共 20 章，前两章概述了全书内容。第 1 章介绍了 HTTP 服务器是如何工作的，第 2 章介绍了一个简单的 servlet 容器。第 3 章和第 4 章着重于连接器的说明，第 5~20 章介绍容器中的各个组件。下面是每一章的内容简介。

注意 每一章都配有一个应用程序用于对该章所介绍组件进行实际应用的说明。

第 1 章：本书从介绍一个简单的 HTTP 服务器开始。为了建立一个可以运行的 HTTP 服务器，你需要了解 java.net 包下 Socket 类和 ServerSocket 类的内部运行机制。该章有详细的背景信息介绍，使你可以理解该章中应用程序的运行机制。

第 2 章：阐明一个简单的 servlet 容器是如何工作的。该章有两个与 servlet 容器有关的应用程序，可以服务于静态资源的请求和简单 Servlet 的请求。此外，你会学习到如何创建 request 对象和 response 对象，并将它们传递给被请求的 Servlet 的 service() 方法。此外，在该 servlet 容器中有一个可以运行的 Servlet，可以从 Web 浏览器中进行调用。

第 3 章：将对 Tomcat 4 中的默认连接器的精简版进行说明。该章中建立的应用程序可以作为一个学习工具，有助于理解在第 4 章中讨论的连接器。

第 4 章：对 Tomcat 4 的默认连接器进行说明。该连接器已经弃用，而是推荐使用另一个称为 Coyote 的执行速度更快的连接器。但是，原先的默认连接器更简单、更容易理解。

第 5 章：讨论容器模块。容器由 org.apache.catalina.Container 接口表示，共有 4 种类型的容器，分别是 Engine、Host、Context 和 Wrapper。该章提供了两个分别与 Context 和 Wrapper 相关的应用程序。

第 6 章：对 Lifecycle 接口进行说明。该接口定义了 Catalina 组件的生命周期，并提供了一种优雅的方法来通知其他组件在该组件中发生了某种事件。此外，Lifecycle 接口提供了一种优雅的机制来启动和关闭 Catalina 中所有的组件，只需要启动 / 关闭一次即可。

第 7 章：介绍日志记录器组件，该组件用于记录错误消息和其他的相关消息。

第 8 章：对载入器组件进行介绍，载入器是 Catalina 中的重要模块，负责载入 Servlet 和 Web 应用程序中所需要的其他类。该章还将说明 Web 应用程序的重载是如何实现的。

第 9 章：介绍管理器组件。该组件负责在会话管理中管理会话。该章介绍了几种不同类型的管理器，并说明了管理器组件是如何持久化一个 session 对象的。在该章末尾，介绍如何使用 StandardManager 实例构建一个使用 session 对象保存数据的应用程序。

第 10 章：该章将讨论 Web 应用程序的安全限制，以限制对某些内容的访问。你会学习到一些与安全相关的实体，如主体、角色、登录配置和身份验证等。该章有两个应用程序，分别在 StandardContext 对象中安装了验证器阀，并使用基本验证来对用户进行身份验证。

第 11 章：对 org.apache.catalina.core.StandardWrapper 类进行了详细阐述，该类表示 Web 应用程序中的一个 Servlet 对象。该章还说明了过滤器和 Servlet 的 service() 方法是如何调用的。该章的应用程序使用 StandardWrapper 实例来表示实际的 Servlet 对象。

第 12 章：介绍 org.apache.catalina.core.StandardContext 类，该类表示一个 Web 应用程序。该章还说明了如何对一个 StandardContext 对象进行配置（这是在接收到 HTTP 请求时进行的）、如何支持 Web 应用程序的自动重载，以及 Tomcat 5 如何通过一个共享线程来执行其相关联组件中的周期性任务。

第 13 章：对另外两种容器（Host 和 Engine）进行说明。该章对这两种容器的标准实现 org.apache.catalina.core.StandardHost 和 org.apache.catalina.core.StandardEngine 进行了说明。

第 14 章：对服务器组件和服务组件进行介绍。服务器组件提供了一种优雅的机制来启动或关闭整个 servlet 容器，服务组件可作为一个容器和一个或多个连接器的持有者。该章通过应用

程序来说明如何使用服务器组件和服务组件。

第 15 章：说明如何通过 Digester 组件来对 Web 应用程序进行配置。Digester 是 Apache 软件基金会的一个开源项目。即使你对这个不熟悉也没有关系，该章会简要介绍 Digester 库，说明如何使用该库来将 XML 文档中的节点转换为 Java 对象。该章然后说明 Tomcat 是如何通过 ContextConfig 对象来对 StandardContext 对象进行配置的。

第 16 章：对 Tomcat 中的关闭钩子进行说明。不论用户如何关闭 Tomcat（即通过发送关闭命令，或是突然直接关闭控制台），通过使用关闭钩子，Tomcat 总是可以执行一些清理工作。

第 17 章：对使用批处理文件和 Shell 脚本来启动和关闭 Tomcat 进行说明。

第 18 章：对部署器组件进行说明，该组件负责部署和安装 Web 应用程序。

第 19 章：对一个特殊接口（ContainerServlet）进行说明。使用该接口，Servlet 对象可以访问 Catalina 中的内部对象。此外，该章会对用来管理已部署应用程序的 Manager 应用程序进行说明。

第 20 章：对 JMX 进行说明，并阐述 Tomcat 是如何为其内部对象创建 MBean，并使这些内部对象可托管的。

每一章的应用程序

每一章都会有一个或多个应用程序用来解释 Catalina 中的某个特定组件的使用方法。一般情况下，在该应用程序中你会找到该组件的精简版，或是为说明如何使用 Catalina 组件而编写的代码。在每一章的应用程序中编写的所有的类和接口都在 *ex[章号].pyrmont* 包下。例如，第 1 章的应用程序类会在 *ex01.pyrmont* 包下。

准备必需的软件

本书的应用程序会运行在 J2SE 的 1.4 版本下。源文件的压缩包可以从作者的网站 www.brainysoftware.com 上下载。该压缩包包含了 Tomcat 4.1.12 的源代码，以及本书中的应用程序的代码。假设你已经安装了 J2SE 的 1.4 版本，而且环境变量 path 也已经包含了 JDK 的路径，那么只需执行下面的步骤。

1）解压缩 zip 文件。解压缩后会有一个名为 HowTomcatWorks 的新文件夹。HowTomcatWorks 是工作目录，其下有一些子目录，包括 lib（包含所有必需的库文件）、src（包含所有源文件）、webroot（包含一个 HTML 文件和三个 Servlet 示例文件）和 webapps（包含示例应用程序）；

2）进入到工作目录中，编译 Java 文件。若你使用 Windows 操作系统，则运行 win-compile.bat 批处理文件。若你使用 Linux 操作系统，则执行下面的命令（必要时，不要忘记用 chmod 命令修改文件的执行权限）：

```
./linux-compile.sh
```

注意 更多的信息可以在压缩包的 Readme.txt 文件中找到。

目　录

第①章

一个简单的 Web 服务器

本章介绍 Java Web 服务器是如何运行的。Web 服务器也称为超文本传输协议（HyperText Transfer Protocol，HTTP）服务器，因为它使用 HTTP 与其客户端（通常是 Web 浏览器）进行通信。基于 Java 的 Web 服务器会使用两个重要的类：java.net.Socket 类和 java.net.ServerSocket 类，并通过发送 HTTP 消息进行通信。本章先介绍 HTTP 协议和这两个类，然后介绍一个简单的 Web 服务器。

1.1 HTTP

HTTP 允许 Web 服务器和浏览器通过 Internet 发送并接收数据，是一种基于“请求 - 响应”的协议。客户端请求一个文件，服务器端对该请求进行响应。HTTP 使用可靠的 TCP 连接，TCP 协议默认使用 TCP 80 端口。HTTP 协议的第一个版本是 HTTP/0.9，后来被 HTTP/1.0 取代，随后 HTTP/1.0 又被当前版本 HTTP/1.1 取代。HTTP/1.1 定义于 RFC（Request for Comment，请求注解）2616 中，可以从 http://www.w3.org/Protocols/HTTP/1.1/rfc2616.pdf 下载。

注意 本节简单地介绍 HTTP 1.1，目的是帮助你了解 Web 服务器应用程序发送的消息。若你对此有兴趣，想了解更多信息，请阅读 RFC 2616。

在 HTTP 中，总是由客户端通过建立连接并发送 HTTP 请求来初始化一个事务的。Web 服务器端并不负责联系客户端或建立一个到客户端的回调连接。客户端或服务器端可提前关闭连接。例如，当使用 Web 浏览器浏览网页时，可以单击浏览器上的 Stop 按钮来停止下载文件，这样就有效地关闭了一个 Web 服务器的 HTTP 连接。

1.1.1 HTTP 请求

一个 HTTP 请求包含以下三部分：

- 请求方法——统一资源标识符（Uniform Resource Identifier，URI）——协议 / 版本
- 请求头
- 实体

HTTP 请求的示例如下所示：

```
POST /examples/default.jsp HTTP/1.1
Accept: text/plain; text/html
Accept-Language: en-gb
Connection: Keep-Alive
Host: localhost
User-Agent: Mozilla/4.0 (compatible; MSIE 4.01; Windows 98)
```

```
Content-Length: 33
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate

lastName=Franks&firstName=Michael
```

请求方法——URI——协议/版本部分出现在请求的第一行，

```
POST /examples/default.jsp HTTP/1.1
```

其中 POST 是请求方法，/examples/default.jsp 表示 URI，HTTP/1.1 表明请求使用的协议及其版本。

每个 HTTP 请求可以使用 HTTP 标准中指定的诸多请求方法中的一种。HTTP 1.1 支持的 7 种请求方法包括：GET、POST、HEAD、OPTIONS、PUT、DELETE 和 TRACE。其中 GET 和 POST 是 Internet 应用中最常用的两种请求方法。

URI 指定 Internet 资源的完整路径。URI 通常会被解释为相对于服务器根目录的相对路径。因此，它总是以“/”开头的。统一资源定位符（Uniform Resource Locator，URL）实际上是 URI 的一种类型（参见 http://www.ietf.org/rfc/rfc2396.txt）。协议版本指明了当前请求使用的 HTTP 协议的版本。

请求头包含客户端环境和请求实体正文的相关信息。例如，请求头可能会包含浏览器使用的语言，请求实体正文的长度等信息。各个请求头之间用回车/换行（Carriage Return/LineFeed，CRLF）符间隔开。

在请求头和请求实体正文之间有一个空行，该空行只有 CRLF 符。这个空行对 HTTP 请求格式非常重要。CRLF 告诉 HTTP 服务器请求实体正文从哪里开始。在有些 Internet 编程书籍中，CRLF 被认为是 HTTP 请求的第 4 部分。

在前面的 HTTP 请求的示例中，请求实体正文很简单，如下所示：

```
lastName=Franks&firstName=Michael
```

当然，在一个典型的 HTTP 请求中，HTTP 请求实体正文也可以很长。

1.1.2 HTTP 响应

与 HTTP 请求类似，HTTP 响应也包括三部分：

- 协议——状态码——描述
- 响应头
- 响应实体段

下面是一个 HTTP 响应的示例：

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/4.0
Date: Mon, 5 Jan 2004 13:13:33 GMT
Content-Type: text/html
Last-Modified: Mon, 5 Jan 2004 13:13:12 GMT
Content-Length: 112
```

```
<html>
<head>
<title>HTTP Response Example</title>
</head>
<body>
Welcome to Brainy Software
</body>
</html>
```

HTTP 响应头的第一行与 HTTP 请求头的第一行类似。第一行指明了使用的协议是 HTTP/1.1，请求发送成功（状态码 200 表示请求成功），一切都正常执行。

从上面的示例中可以看出，它与请求头类似，响应头也包含了一些有用的信息。响应实体正文是一段 HTML 代码。响应头和响应实体正文之间由只包含 CRLF 的一个空行分隔。

1.2 Socket 类

套接字是网络连接的端点。套接字使应用程序可以从网络中读取数据，可以向网络中写入数据。不同计算机上的两个应用程序可以通过连接发送或接收字节流，以此达到相互通信的目的。为了从一个应用程序向另一个应用程序发送消息，需要知道另一个应用程序中套接字的 IP 地址和端口号。在 Java 中，套接字由 java.net.Socket 表示。

要创建一个套接字，可以使用 Socket 类中众多构造函数中的一个。其中一个构造函数接收两个参数：主机名和端口号。

```
public Socket (java.lang.String host, int port)
```

其中参数 host 是远程主机的名称或 IP 地址，参数 port 是连接远程应用程序的端口号。例如，想要通过 80 端口连接 yahoo.com，可以使用下面的语句创建 Socket 对象：

```
new Socket ("yahoo.com", 80);
```

一旦成功地创建了 Socket 类的实例，就可以使用该实例发送或接收字节流。要发送字节流，需要调用 Socket 类的 getOutputStream() 方法获取一个 java.io.OutputStream 对象。要发送文本到远程应用程序，通常需要使用返回的 OutputStream 对象创建一个 java.io.PrintWriter 对象。若想要从连接的另一端接收字节流，需要调用 Socket 类的 getInputStream() 方法，该法会返回一个 java.io.InputStream 对象。

下面的代码段创建了一个套接字，用于与本地 HTTP 服务器进行通信（127.0.0.1 表示一个本地主机），发送 HTTP 请求，接收服务器的响应信息。以下代码创建了一个 StringBuffer 对象来保存响应信息，并将其输出到控制台上。

```
Socket socket = new Socket("127.0.0.1", "8080");
OutputStream os = socket.getOutputStream();
boolean autoflush = true;
PrintWriter out = new PrintWriter(
  socket.getOutputStream(), autoflush);
BufferedReader in = new BufferedReader(
  new InputStreamReader( socket.getInputStream() ));
```

```
// send an HTTP request to the web server
out.println("GET /index.jsp HTTP/1.1");
out.println("Host: localhost:8080");
out.println("Connection: Close");
out.println();

// read the response
boolean loop = true;
StringBuffer sb = new StringBuffer(8096);
while (loop) {
  if ( in.ready() ) {
    int i=0;
    while (i!=-1) {
      i = in.read();
      sb.append((char) i);
    }
    loop = false;
  }
  Thread.currentThread().sleep(50);
}

// display the response to the out console
System.out.println(sb.toString());
socket.close();
```

注意 为了从Web服务器上获取正确的响应信息，需要发送一个遵循HTTP协议的HTTP请求。如果你已经阅读了前一节中关于HTTP的描述，你应该已经可以理解以上代码中关于发送HTTP请求的方法。

注意 可以使用本书中的com.brainysoftware.pyrmont.util.HttpSniffer类来发送HTTP请求，并显示响应信息。要使用该Java程序，需要连接到Internet。但是，要注意的是，防火墙可能会使程序失败。

ServerSocket类

Socket类表示一个客户端套接字，即，当想要连接到远程服务器应用程序时创建的套接字。但如果你想要实现一个服务器应用程序（例如一个HTTP服务器或FTP服务器），你需要另一种方法。因为服务器必须时刻待命，它并不知道客户端应用程序会在什么时候发起连接。正因如此，需要使用java.net.ServerSocket类，这是服务器套接字的实现。

ServerSocket类与Socket类并不相同。服务器套接字要等待来自客户端的连接请求。当服务器套接字收到了连接请求后，它会创建一个Socket实例来处理与客户端的通信。

要创建一个服务器套接字，可以使用ServerSocket类提供的4个构造函数中的任意一个。需要指明IP地址和服务器套接字侦听的端口号。典型情况下，IP地址可以是127.0.0.1，即服务器套接字会侦听本地机器接收到的连接请求。服务器套接字侦听的IP地址称为绑定地址。服务器套接字的另一个重要属性是backlog，后者表示在服务器拒绝接收传入的请求之前，传入的连接请求的最大队列长度。

ServerSocket类的其中一个构造函数的签名如下：

```
public ServerSocket(int port, int backLog, InetAddress bindingAddress);
```