网络编程系列丛书

杜佳荣　马建红　滕振宇　编著

Java 网络编程技术与实践

书附光盘中给出案例源代码、案例开发和运行过程的全程多媒体讲解视频

本书主要内容

- Java网络编程基础
- Java与TCP网络协议开发
- Java与UDP通信协议开发
- Java Applet编程
- JMF播放器实现
- FTP客户端程序开发
- 基于RMI的网络应用设计
- 网络五子棋、网络白板实例
- 网络聊天室及文件上传和下载
- 邮件管理及订单查询系统
- EJB网络应用程序开发
- 基于EJB的学生选课系统

清华大学出版社

网络编程系列丛书

Java 网络编程技术与实践

杜佳荣　马建红　滕振宇　编著

清华大学出版社
北　京

内 容 简 介

本书本着理论结合实际的原则，通过诸多案例来分析各种 Java 网络程序的设计思想和开发步骤。全书共分 15 章，分别讲解了 Java 网络编程基础、TCP 和 UDP 协议应用、Applet 技术、基于 JMF 协议实现在线播放多媒体功能、开发基于 FTP 协议的文件下载程序、基于 RMI 协议实现远程调用，以及网络五子棋、网络白板、Java 聊天室、文件上传和下载、邮件系统、订单查询等程序的开发方法和基于多层网络架构程序的开发过程。

本书突出实用，实例丰富，不仅可作为高等院校计算机相关专业师生的参考教材，对于广大程序设计人员也有很大的参考价值。

图书在版编目(CIP)数据

Java 网络编程技术与实践/杜佳荣，马建红，滕振宇　编著. —北京：清华大学出版社，2008.6
(网络编程系列丛书)

ISBN 978-7-302-17514-8

Ⅰ.J…　Ⅱ.①杜…②马…③滕…　Ⅲ.Java 语言—程序设计　Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2008)第 061043 号

责任编辑：王　定
封面设计：久久度文化
版式设计：康　博
责任校对：胡雁翎
责任印制：孟凡玉

出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
社　总　机：010-62770175　　**邮　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn

印 刷 者：北京市世界知识印刷厂
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185×260　**印　张**：35.5　**字　数**：820 千字
附光盘 1 张
版　　次：2008 年 6 月第 1 版　　**印　　次**：2008 年 6 月第1次印刷
印　　数：1～5000
定　　价：59.00 元

Java 语言运行在诸多操作系统的虚拟机上，具有跨平台的特性，用它可以屏蔽各种操作系统的差异。

跨平台特性对编写网络程序来说，是非常有帮助的。用 Java 语言开发出来的实现网络功能的项目，能在各种操作系统之间较容易地实现网络通信功能，进而能在网络间协作性地完成各种操作。

在本书的各章节中，大量使用了 JDK 提供的类库，通过使用这些已经被封装的类以及其中包含的方法，我们可不必把太多的精力放在具体的网络底层的实现方式上，而可以把项目开发中的各种资源集中到“实现网络应用逻辑”这个重要方面。

本书共分为 15 章，其中第 1 章介绍了 Java 网络编程基础，第 2 章和第 3 章详细讲述了当前网络应用中最基本的 TCP 和 UDP 协议。通过阅读这些章节，可以了解网络开发的各种难点和“基于协议”的网络程序开发方式。

在本书的第 4 章里，分析了在网页中嵌入声音和动画的 Applet 技术，第 5 章则详细分析了基于 RTP 协议实现 JMF 在线播放器的方法，第 6 章则介绍了如何利用 Java 语言开发基于 FTP 协议的文件下载程序，而本书的第 7 章介绍了基于 RMI 协议实现远程调用的方法。这些章节里的一些技术点，应用范围都很广，这将是我们进一步学习的基础。

在后面的第 8 章到第 10 章里，分别通过案例，生动地介绍了综合运用各种协议实现“网络五子棋”、“网络白板”和“Java 聊天室”的方法和过程，第 11 章介绍了基于 COS 组件的文件上传和下载应用开发，第 12 章介绍了如何使用网页方式实现 JSP 邮件系统，第 13 章介绍了基于 SOAP 协议的订单查询应用开发。

在本书的最后两章(第 14 章和第 15 章)则说明了如何利用 EJB 开发基于多层网络架构的程序。这两章不仅讲述了基于 EJB 项目的开发方式，还通过基于 EJB 的案例分析了“基于组件”的开发思想。

本书不仅注重各种网络协议理论，更侧重于各种网络开发的思想，书中的诸多网络应

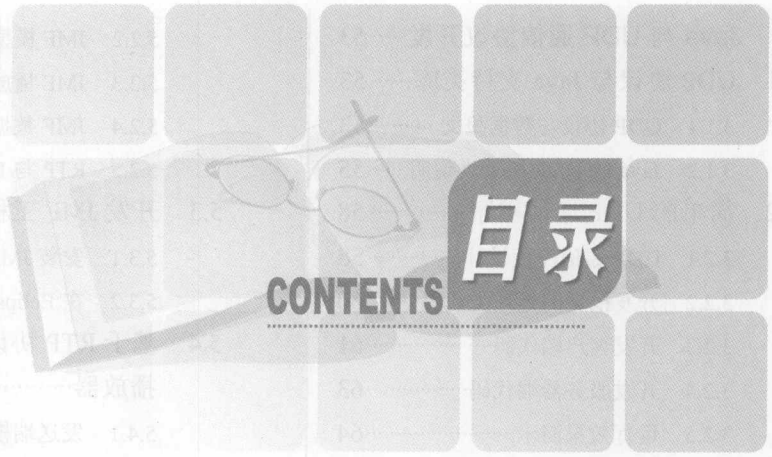
用程序只要稍加修改就可直接使用。

本书不仅适合高等院校计算机相关专业的师生学习和参考之用，更对高校计算机专业的学生进行毕业设计有非常大的参考价值，同时也可以作为广大计算机编程爱好者的自学参考用书。

本书主要由杜佳荣、马建红、滕振宇执笔，在编写本书程序的过程中，得到了马晓冬、程炜杰、赵晓坤的大力支持，在此表示衷心的感谢。此外，参与本书编写的人员还有余亚洲、侯义东、马子明、方静飞、彭飞、王平辉、何勇琪、张袁俊、刘益群、彭涛、钱磊、刘海钧、王勇、王晖、仇多杰、王亮、常杨、杨宝功、斯维、方义、曹宁等人，在此深表感谢。

由于时间仓促，编者水平有限，书中不足之处在所难免，敬请读者批评指正。

编　者



CONTENTS 目录

CHAPTER 1

Java 网络编程起步

随着互联网经济的迅猛发展，越来越多的 IT 公司开始关注基于网络的应用。

网络编程的难点在于，要保证程序能在安装不同类型操作系统的主机上运行，并且，各主机能通过这些 Java 程序无差别地进行通信，这就要求开发网络程序的语言具有能在很大程度上实现跨平台的特性。

幸运的是，Java 的一个很重要的特性是平台无关性，即在开发编译代码后，能无差别地运行在异构的操作系统上。所以，Java 语言与网络编程有着天然的联系，用 Java 语言能相对容易地编写基于网络的应用项目。

1.1 Java 开发环境概述

在讲解 Java 网络编程之前，我们有必要了解 Java 语言的开发环境。

Java 语言的开发环境叫作 Java Development Kit，意思为“Java 开发包”，JDK 是它的简写。从功能角度来看，JDK 主要包含四个基本的组件，具体说明如下。

- Javac——它是 Java 语言的编译器，用它可以把扩展名是.java 的源程序转成扩展名是.class 的字节码文件。
- Jar——它是打包工具，用它将相关的类文件打包成一个 jar 包，jar 包比起.class，使用起来更加便利。
- Javadoc——它是 Java 程序文档生成器，用它可以从源代码的注释中提取内容，生成 html 格式的帮助文档。
- Jdb——它是用来查错的 Java 小工具。

此外，JDK 开发包里，还包括完整的 JRE(Java Runtime Environment，Java 运行环境)，

其中包括了用于各种运行环境的支持库类，也包括了给开发员使用的补充库，如实现国际化的库和 IDL 库等。

1.1.1 安装和配置 JDK

通过刚才的介绍，可知 Java 的开发和编译工具都集中在 JDK 包里，所以首先要安装和配置 JDK 环境。

可以按照如下步骤安装和配置 JDK 开发环境。

1. 安装 JDK

安装 JDK 前，首先需要获得 JDK 的安装文件。

在 Sun microsystems 的官方网站：http://java.sun.com/javase/downloads/index.jsp 上(注：大家也可以从 http://java.sun.com/首页里，根据指示，依次进入下载页面)，可以得到 JDK 安装文件，目前最新的版本是 JDK 6.0。

在这个版本的 JDK 中，提供了 J2SE 标准版的开发环境和工具，以及 JRE 扩展类库。由于 JDK 具有运行在操作系统层面上，依次屏蔽各操作系统的差异，从而实现跨平台的特性，所以，对于不同种类的操作系统，需要选择不同版本的 JDK 安装程序。由于本书主要是在 Windows 环境下介绍 Java 的网络编程，所以这里下载 Windows 平台的 JDK 安装程序。

下载成功之后，用鼠标双击运行该应用程序，启动 JDK 安装程序，根据提示信息，就可以完成 JDK 的安装。

在选择安装路径的时候，约定 JDK 的安装目录为“C:\Java\jdk1.6.0”。在 JDK 安装完成之后，安装程序会再次要求输入 JRE 的安装路径，我们同样设置 JRE 的安装目录为“C:\Java\jre1.6.0”。

最后，当所有的部分都安装完成之后，系统提示安装成功。安装完成后，就可以查看描述 JDK 内容的自述文件。

一般 JDK 安装完成之后，建议重新启动电脑。JDK 安装成功之后，并不是说 JDK 环境就配置好了，我们还需要设置环境变量。

2. 设置环境变量

在操作系统里，右击桌面上的“我的电脑”图标，在弹出的菜单里选择“属性”命令，打开系统属性页面，在其中选择“高级”属性页，单击其中的“环境变量”按钮，打开环境变量设置页面，如图 1-1 所示。

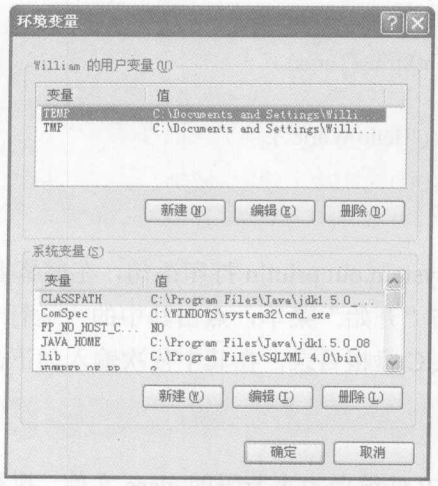


图 1-1 设置环境变量

在系统变量中寻找 JAVA_HOME 变量，如果没有则单击“新建”按钮，新建 JAVA_HOME 变量，如图 1-2 所示。

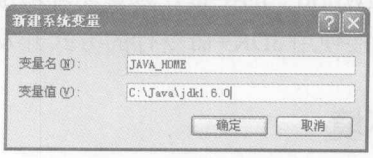


图 1-2 新建环境变量

这里配置变量名为 JAVA_HOME，变量值为刚才安装 JDK 的目录 C:\Java\jdk1.6.0。用这个变量，可以指定 Java 的安装环境，并可以设置后面的 classpath 和 path 环境。

可以按照同样的方法分别设置 Path 和 classpath 变量，同样地，如果该变量存在，则在变量值的最后添加；如果不存在，则直接添加。

添加的 Path 变量值为“%JAVA_HOME%\bin;%JAVA_HOME%\lib;”，添加完成后，如果在命令行里运行“javac”等 JDK 命令，系统会自动到上述路径去查找 javac.exe 执行文件。

添加的 classpath 变量值为“%JAVA_HOME%\lib;”，添加完成后，JDK 虚拟机系统会自动到这个路径下查找编译所需要的 Java 开发包。

至此，完成了 JDK 开发环境的配置工作。

3. 测试 JDK 配置

为了测试安装和配置的 JDK 环境是否正确，在 C 盘根目录下建立一个名为“HelloWorld.java”的文件，请注意这里的大小写必须完全一致。

然后用记事本或其他文本工具编辑这个文件，代码如下：

```
public class HelloWorld
```

```
{
    public static void main(String[] args)
    {
        System.out.println("Hello World");
    }
}
```

这段代码主要是通过 System.out.println 打印语句，在屏幕上输出“HelloWorld”字符。

编写保存完成后，点击“开始”菜单，点击其中的“运行”选项，并输入“cmd”命令，启动命令行窗口，进入 C 盘根目录，在其中依次输入如下命令代码：

```
javac HelloWorld.java
```

Javac 命令用来把.java 文件编译为字节码的.class 文件，执行完此命令后，会在 C 盘根目录下出现 HelloWorld.class 文件，在此基础上运行如下命令：

```
java HelloWorld
```

如果在命令行里能显示“HelloWorld”的字符，则说明环境配置成功了。

需要说明的是，不推荐直接使用 JDK+命令行的方式开发 Java 程序。

1.1.2 安装 Eclipse 环境

如果所有的代码都要在记事本中编辑，这将会是一种低效的事情，这时就需要类似于 Visual Studio 那样的集成开发环境。

Eclipse 作为一款 Java 的开发集成软件，拥有非常方便的实用方法和实用的功能，同时，它也具有添加各种插件的功能。通过添加插件，可以方便地在 Eclipse 里添加各种 Java 的扩展 API 包，比如支持 EJB 开发的 Jboss 组件，支持 Java 手机开发的 J2ME 插件。所以，Eclipse 是目前较为流行的 Java 开发软件。

可以按如下的步骤，安装和配置 Eclipse 环境。

1. 安装 Eclipse

要安装 Eclipse 软件，首先需要到官方网站上下载。Eclipse 的下载链接可以从官方网站：http://www.eclipse.org/downloads/中获得，目前最新的版本为 Eclipse SDK 3.2.1。

Eclipse 同样有多个下载的版本可供选择，这里同样选择 Windows 版本。(注：目前支持 Windows 版本的 Eclipse 并不支持最新的 Vista 系统。)

Eclipse 下载之后为一个 Zip 压缩包，它不需要安装，直接把它解压缩到指定的目录便可以使用。

这里约定解压缩的目录为 D:\eclipse。完成解压缩之后，如果 JDK 环境一切正常，就可以通过运行 D:\eclipse 下的 eclipse.exe 文件，启动 Eclipse 程序。

Eclipse 启动之后会根据系统信息自动配置 JDK 环境，所以不需要手动设置。

2. 添加 Eclipse 插件

从官方网站上下载的 Eclipse 一般为英文版，对于初次使用 Eclipse 的读者来说，可能会有一定的困难，所以通常还需要下载一个支持中文的补丁包。

在网址 http://download.eclipse.org/eclipse/downloads/下，可以找到 Eclipse 的语言包“3.2.1_Language_Packs”(或者可以到搜索引擎下搜索 Eclipse 语言包的下载地址)，点击之后进入语言包选择的页面，如图 1-3 所示。

SDK Language Packs	Windows	Linux (x86/GTK 2) and Solaris (Sparc/GTK 2)	Description
NLpack1 - German, Spanish, French, Italian, Japanese, Korean, Portuguese (Brazil), Traditional Chinese and Simplified Chinese.	NLpack1-eclipse-SDK-3.2.1-win32.zip	NLpack1-eclipse-SDK-3.2.1-gtk.zip	These are required if SDK was downloaded
NLpack2 - Czech, Hungarian, Polish and Russian	NLpack2-eclipse-SDK-3.2.1-win32.zip	NLpack2-eclipse-SDK-3.2.1-gtk.zip	
NLpack2a - Danish, Dutch, Finnish, Greek, Norwegian, Portuguese, Swedish and Turkish.	NLpack2a-eclipse-SDK-3.2.1-win32.zip	NLpack2a-eclipse-SDK-3.2.1-gtk.zip	
NLpackBidi - Arabic and Hebrew	NLpackBidi-eclipse-SDK-3.2.1-win32.zip	NLpackBidi-eclipse-SDK-3.2.1-gtk.zip	

图 1-3　选择语言包的界面

根据提示信息，下载 NLpack1-eclipse-SDK-3.2.1-win32.zip 文件。

文件下载完成之后，将其解压缩，生成 features 和 plugins 文件夹。将这两个文件夹中的内容分别复制到 Eclipse 所在的目录(比如：D:\eclipse\features 和 D:\eclipse\plugins)中去，如果复制的时候存在同名，则将其覆盖。复制完成之后，把 D:\eclipse\configuration 文件夹中除了 config.ini 文件以外的所有文件或文件夹删除。最后重新运行 Eclipse，就能看到中文版的开发界面了。

1.2　网络通信常用协议

网络通信的层次结构和网络通信的协议，是开发网络程序的基础。目前在网络编程方面，最常用的是 TCP/IP 和 UDP 通信协议，而其他的一些诸如 RMI、SOAP 和 FTP 等协议，可以说都是构建在这两者之上的。

通过这些协议，网络通信的各主机可以用一种统一而非杂乱的规范，高效便捷地相互发送和接收消息，以此完成各种网络操作。

1.2.1 TCP/IP 网络通信模型

如果在现实生活中，不同的主机间用不同的方式传输数据(比如在中国，在底层通信光缆上用高电压表示“1”，用低电压表示“0”，而美国反之)，网络通信将会变的杂乱无章，从而无法管理。

所以美国在 1977 年到 1979 年间，建立了 TCP/IP 网络通信模型，用来规范网络信息传输的各项动作。基于 TCP/IP 协议的网络层次结构如图 1-4 所示。

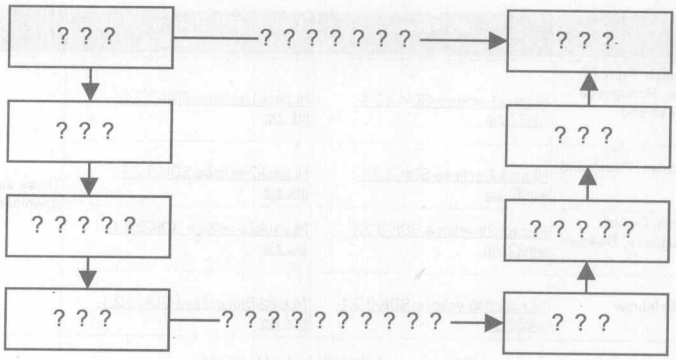


图 1-4 TCP/IP 协议的网络层次结构图

在上图中，物理层负责以二进制数据流的形式在物理信道上传输数据，数据链路层负责把数据流打包解包，网络层主要负责为数据包选择发送到目的主机的路由，而传输层为用户提供点对点的数据传输服务。

从 TCP/IP 层次结构图中，我们还可以看到，高层(比如传输层)是调用低层(比如网络层)提供的服务来实现网络通信动作的，而低层一般是提供具体的网络互连服务。

1.2.2 TCP 与 UDP 通信协议

一般来说，Java 语言大多是在 TCP/IP 网络互连层次结构上的传输层编程，而不关心底层的数据传输方式。

在传输层里主要有基于 TCP 和 UDP 这两种协议的传输数据流的方式。

其中，TCP 是 Tranfer Control Protocol 的缩写，是一种面向连接的保证可靠传输的协议。在传输数据流前，通信双方必须建立一条虚拟的信道，在此信道上能以差错率较低的形式传输数据流。

而 UDP 是 User Datagram Protocol 的缩写，是一种无连接的协议。同 TCP 协议不同，使用 UDP 方式传输数据时，每个数据段都是一个独立的信息，包括完整的源地址或目的地址，它在网络上以任何可能的路径传往目的地，因此能否到达目的地，到达目的地的时间以及内容的正确性都不能保证。

对比这两种协议，可以看到，使用 UDP 时，每个数据段中都包含了完整的地址信息，因此不需要建立发送方和接收方的连接。而对于 TCP 协议，由于它是一个面向连接的协议，所以多了建立连接的时间。

另外，使用 UDP 传输数据时对数据是有大小限制的，每个被传输的数据包必须限定在 64KB 之内。而 TCP 没有这方面的限制，一旦连接建立起来，双方就可以在信道里，按统一的格式传输大量的数据。并且，UDP 是一个不可靠的协议，发送方所发送的数据包并不一定以相同的次序到达接收方。而 TCP 是一个可靠的协议，它确保接收方完全正确地获取发送方所发送的全部数据。

在 Java 语言里，使用 Socket(套接字)模型来封装网络传输的具体动作，而在 Socket 模型里，也提供了对应的方法实现以 TCP 和 UDP 方式传输数据的功能。

关于 Socket 通信的知识以及代码，本书将在第三章和第四章详细讲述。

1.3 I/O 流与网络通信

从 TCP/IP 网络层次结构模型上，我们可以看到，不管在传输层里传输何种格式的数据，这些数据终究要在物理层以二进制数据流的形式传送。

所以，Java 的 I/O 体系同网络编程密切相关，在这里将讲述 I/O 流的基本知识，并给出 I/O 操作的基本案例，在后文中将具体调用 I/O 类提供的功能，实现在网络间传输数据流的动作。

1.3.1 Java 输入流与输出流概述

I/O(Input/Output)是计算机的输入/输出接口，在 Java JDK 库的 java.io 提供了全方位的 I/O 接口，其中主要包括：文件读写、内存读写和标准设备输出等。

Java 中 I/O 是把所有的输入/输出数据抽象成“流”的形式，即所有的输出数据可以被串行化地写入输出流，应用程序里可以从输入流读入待接收的数据。

事实上，Java 的 I/O 体系分为 Input/Output 和 Reader/Writer 两类，它们都可以用来保存和管理输入和输出的数据。并且，在 java.io 包里，所有的 I/O 类都是配对的，即有一个 Input 类就有一个对应的 Output 类。这样的划分方式，能让 Java 程序员使用同样的接口，操作不同数据类型的输入/输出流。

1.3.2 代码示例

下面将通过一个具体的实例，来观察 Java 里 I/O 类的工作方式。代码如下：