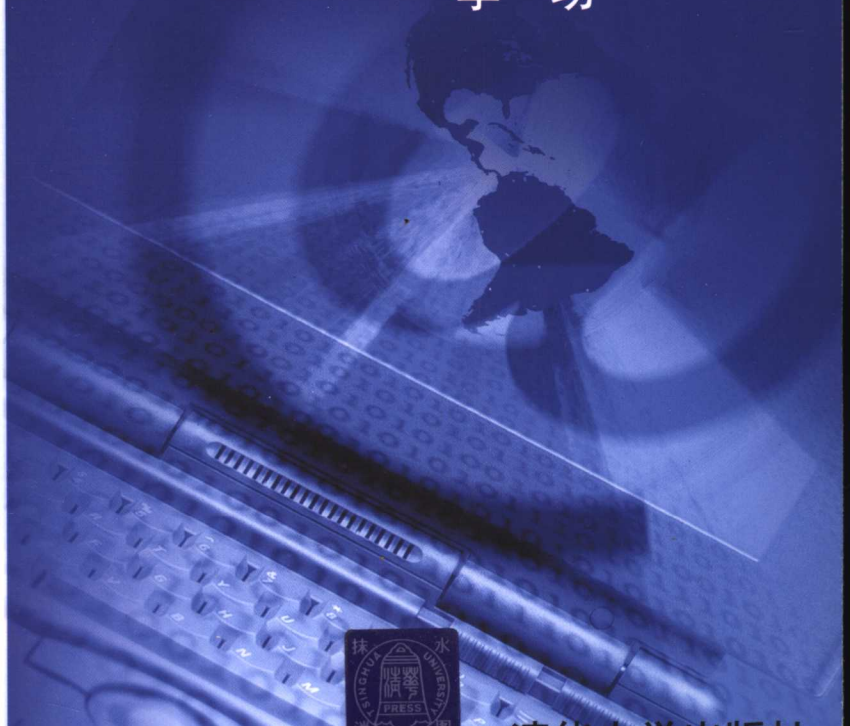
# 精通

# Java 网络编程

- Java环境配置
- Java的输入/输出系统及其工作方式
- Java多线程应用程序的优势
- Internet的工作方式、体系结构和TCP/IP协议栈
- 使用UDP和TCP编写客户和服务器应用程序
- 使用扩展JavaMailAPI访问E-mail
- 数据库连接的实现
- RMI和CORBA分布式计算技术
- Java安全性保证
- JSP，Applet和Servlet的基本概念

汪晓平
俞　俊　编著
李　功



清华大学出版社

# 精通 Java 网络编程

汪晓平　俞俊　李功　编著

清华大学出版社

北　京

## 内 容 简 介

本书清晰地介绍了联网的基本原理，在进行网络编程时要掌握的主要概念，以及在联网时可能遇到的问题和 Java 解决方案，并通过多个实例详尽地介绍了如何运用网络编程技术在 Java 平台上编写应用程序。

本书内容包括 Java 环境配置，Java 的输入/输出系统及其工作方式，Java 多线程应用程序的优势，Internet 的工作方式、体系结构和 TCP/IP 协议栈，Java 环境下使用 UDP 和 TCP 编写客户与服务器应用程序，使用扩展 JavaMailAPI 访问 E-mail，Java 下数据库连接的实现，RMI(远程方法调用)和 CORBA 分布式计算技术，Java 安全性的保证，以及 JSP, Applet 和 Servlet 的基本概念。

本书力求创新，给读者以实用和最新的技术与技巧，适合初中级编程人员作为自学教材或参考书。读者在阅读本书时，可以通过访问 http://www.tupwk.com.cn 下载本书相关程序代码。



**图书在版编目(CIP)数据**

精通 Java 网络编程/汪晓平，俞俊，李功编著. —北京：清华大学出版社，2005.9
ISBN 7-302-11481-1

Ⅰ. 精…　Ⅱ. ① 汪…② 俞…③ 李 …　Ⅲ. Java 语言—程序设计　Ⅳ. TP312

中国版本图书馆 CIP 数据核字(2005)第 086923 号

**出 版 者**：清华大学出版社　　**地　　址**：北京清华大学学研大厦
http://www.tup.com.cn　　**邮　　编**：100084
**社 总 机**：010-62770175　　**客户服务**：010-62776969
**组稿编辑**：王　定
**文稿编辑**：鲍　芳
**封面设计**：久久度文化
**版式设计**：康　博
**印 装 者**：北京市昌平环球印刷厂
**发 行 者**：新华书店总店北京发行所
**开　　本**：185×260　**印张**：36　**字数**：831 千字
**版　　次**：2005 年 9 月第 1 版　2005 年 9 月第1 次印刷
**书　　号**：ISBN 7-302-11481-1/TP · 7535
**印　　数**：1 ~ 4000
**定　　价**：49.80 元

# 前　言

本书旨在介绍和解释网络技术的基本概念，并讨论 Java 网络编程实践技巧。通过本书的阅读，读者可以提高 Java 网络编程的速度，以及更好地运用在软件开发中所学的技巧。如果用户已经拥有另一种语言的某些编写网络程序的经验，那么将发现那些编程经验依旧可以应用在 Java 中。本书可以被看作是一本速成指南，同时也是 Java 网络编程 API 方面的参考书。

基于以上思想，我们精心编写了本书。书中通过大量经典的实例，由浅入深、循序渐进、由点到面、系统地讲解了 Java 编程各方面的知识。

在本书中读者将学会怎样运用网络编程技术在 Java 平台上编写应用程序。Java API 提供了在 Internet 上通信的很多方法，包括从发送数据包和数据流到实现 HTTP 等高层应用协议，以及分布式计算机制。本书可以帮助读者在短时间内掌握 Java 网络编程技巧，随心所欲地运用 Java 编写自己需要的程序。

全书分为 15 章，共 30 多个实用例子。同时每章针对技术难点还给出了具体的代码。

第 1 章介绍了 Java 环境配置，包括环境变量的设置以及 Java 编程中用到的开发环境 JBuilder 9。通过该部分的学习，使读者能够顺利地配置好整个运行环境，同时熟悉 JBuilder 9 的使用，为后面章节的学习打好基础。

第 2 章介绍了 Java 的输入/输出系统及其工作方式，内容涵盖了基本输入/输出类、装饰类、新输入/输出流等。

第 3 章介绍了 Java 线程的概念，同时讲解使网络程序可以并发执行多个任务的多线程应用程序的优势。

第 4 章介绍了 Internet 的工作方式、体系结构和 TCP/IP 协议栈。

第 5 章介绍了如何使用 URL 检索数据，包括 URL 的基本概念和具体的应用方式。

第 6 章介绍了客户端和服务器端的基本概念，以及如何使用 UDP(User Datagram Protocol，用户数据包协议)和 TCP(Transport Control Protocol，传输控制协议)编写客户端应用程序。

第 7 章是在第 6 章的基础上介绍服务器端应用程序的编写过程，以及将客户端和服务器端合并起来共同考虑设计。

第 8 章介绍了安全套接字 SSL 的基本概念和应用。

第 9 章介绍了如何用扩展的 JavaMailAPI 设计 E-mail。

第 10 章介绍了如何使用 JDBC 实现 Java 与数据库的连接。

第 11 章介绍了包括 RMI(远程方法调用)和 CORBA 的分布式计算技术。

第 12~15 章分别介绍了 JSP，Applet，EJB 和 Servlet 的基本概念，并通过实例帮助读者较快地理解。

本书由汪晓平主编，俞俊、李功执笔。在编写过程中，我们力求精益求精，但难免存在一些不足之处，如果读者使用本书时遇到问题，请发 E-mail 到 lgpolly@21cn.com，或 hnwangd@163.com。

另外，读者可以访问 http://www.tupwk.com.cn 下载本书的相关程序代码。

编　者

2005 年 7 月

# 目　录

# 第1章 Java环境配置

Java语言的优势之一是它的跨平台特性，不管所使用的底层操作系统是什么类型，只要此操作系统安装了Java虚拟机，那么用Java编写的程序就可以在此平台上顺利运行。本章主要介绍如何配置Java环境。

## 1.1 Java环境的配置

下面介绍Java常用开发环境J2SDK的安装配置。

### 1.1.1 SDK的下载和安装

本节中采用的J2SDK的版本是1.4.2，用户可以从http://java.sun.com/j2se/1.4.2/download.html上免费下载，如图1-1所示为下载网址。

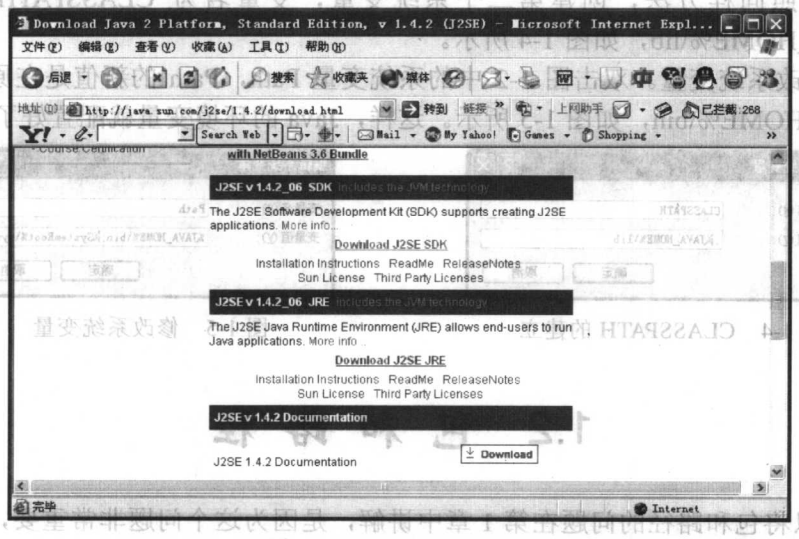



图1-1 J2SDK下载网址

J2SDK 的安装过程较为简单，只要按照程序的提示逐步操作就可以了。安装完成后，需要配置环境变量。

## 1.1.2 环境变量的配置

在 Windows XP 的桌面上右击“我的电脑”，在弹出的快捷菜单中选择“属性”命令，会弹出“系统属性”设置对话框。选择“高级”选项卡，并单击“环境变量”按钮，打开如图 1-2 所示的“环境变量”对话框，然后按以下步骤进行环境变量的配置。

(1) 创建第一个系统变量。在图 1-2 中单击“新建”按钮，弹出如图 1-3 所示的“新建系统变量”对话框。设置“变量名”为 JAVA_HOME，“变量值”为 C:\j2sdk\j2sdk1.4.2 (如果用户将 J2SDK 安装在 C:\j2sdk 目录下)。

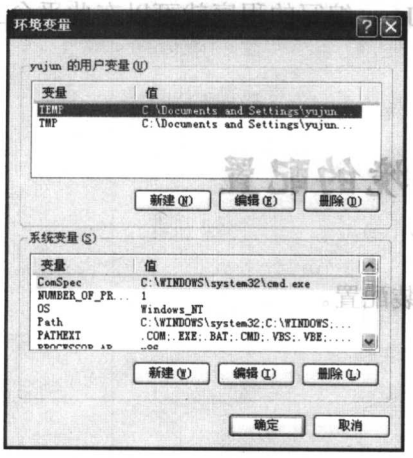



图 1-2 环境变量

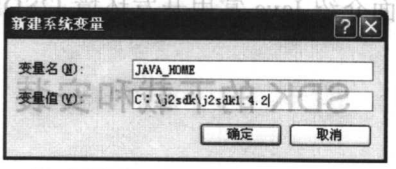



图 1-3 新建系统变量

(2) 按照同样方法，创建第二个系统变量，变量名为 CLASSPATH，变量值为;%JAVA_HOME%\lib，如图 1-4 所示。

(3) 修改系统变量。双击图 1-2 中的系统变量 Path，Path 的新值是在原有值前加上%JAVA_HOME%\bin，如图 1-5 所示。这样，Java 的环境变量就配置好了。

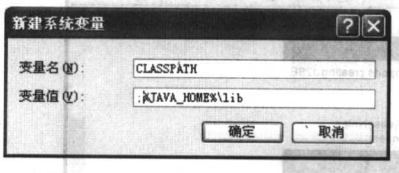



图 1-4 CLASSPATH 的建立

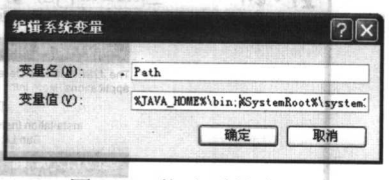



图 1-5 修改系统变量

# 1.2 包和路径

之所以将包和路径的问题在第 1 章中讲解，是因为这个问题非常重要，很多时候造成程序运行错误往往不是程序本身的问题，而是包设置的问题。

## 1.2.1　类路径

当编写了一个 HelloWorld 程序，编译运行后出现了 Can't find class HelloWorld 或者 Exception in thread "main" java.Lang.NoSuchMethod Error：main.这样的提示语言是什么原因？现在一起来看一看 Java 程序的运行过程。Java 是通过 Java 虚拟机来解释运行的，也就是通过 Java 命令 javac 编译生成的。class 文件就是虚拟机要执行的代码，称之为字节码(bytecode)，虚拟机通过 classloader 来装载这些字节码，也就是通常意义上的类。Classloader 要知道 java 本身的类库及用户自己的类须要有默认值(当前路径)；或者要有一个用户指定的变量来表明，这个变量就是类路径(classpath)，或者在运行的时候传参数给虚拟机，这也就是指明 classpath 的 3 个方法。编译的过程和运行的过程大同小异，只是一个是找出来编译，另一个是找出来装载。

实际上 Java 虚拟机是由 Java luncher 初始化的，也就是 Java (或 java.exe)这个程序来做的。虚拟机按以下顺序搜索并装载所有需要的类。

(1) 引导类：组成 Java 平台的类，包含 rt.jar 和 i18n.jar 中的类。

(2) 扩展类：使用 Java 扩展机制的类，都是位于扩展目录($JAVA_HOME/jre/lib/ext)中的.jar 档案包。

(3) 用户类：开发者定义的类或者没有使用 java 扩展机制的第三方产品。必须在命令行中使用-classpath 选项或者使用 CLASSPATH 环境变量来确定这些类的位置。上面所说的用户自己的类就是特指这些类。

一般来说，用户只需指定用户类的位置，引导类和扩展类是“自动”寻找的。

用户类路径就是一些包含类文件的目录，.jar 和.zip 文件的列表，至于类具体怎么找，因为牵扯到 package 的问题，下面将会介绍，暂时可认为只要包含了这个类就算找到了它。根据平台的不同分隔符略有不同，类 Unix 系统基本上都是“:”，Windows 系统多是“;”。其可能的来源有以下这些。

(1) “:”，即当前目录，这个是默认值。

(2) CLASSPATH 环境变量，一旦设置，将默认值覆盖。

(3) 命令行参数 -cp 或者 -classpath，一旦指定，将上面两者覆盖。

由-jar 参数指定的.jar 档案包，就把所有其他的值覆盖，所有的类都来自这个指定的档案包中。生成可执行的.jar 文件还需要其他一些知识，比如 package，还有特定的配置文件。

现在举个 HelloWorld 的例子来说明。先做以下假设：

(1) 当前目录是 HelloWorld(或 c:\HelloWorld，以后都使用前一个)。

(2) JDK 版本为 1.4.2(Windows 下的)。

(3) Path 环境变量设置正确，这样在任何目录下都可以使用 Java 工具。

(4) 文件是 HelloWorld.java，内容如下：

```
public class HelloWorld
{
public static void main(String[] args)
```

```
    {
    c:\>java HelloWorld
```

到现在为止，只设置了 Path。当前路径就是指.class 文件在当前目录下：

```
[HelloWorld]$ javac HelloWorld.java  //这一步不会有多大问题
[HelloWorld]$ java HelloWorld        // 这一步可能就会有问题
```

如果出了像开头那样的问题，首先确定不是由于输错命令而出错。如果没有输错命令，那么接着输入：

```
c:\HelloWorld>echo %CLASSPATH%
```

看看 CLASSPATH 环境变量是否设置了，如果设置了，那么用以下命令：

```
c:\HelloWorld> set CLASSPATH=
```

来使它为空，然后重新运行。这次用户类路径默认的是“.”，所以应该不会有相同的问题了。还有一个方法就是把“.”加入到 CLASSPATH 中，如：

```
c:\HelloWorld> set CLASSPATH=%CLASSPATH%;.
```

同样也可以成功。

当程序需要第三方的类库支持，而且比较常用，就可以采用这种方法。比如常用的数据库驱动程序，写 Servlet 需要的 Servlet 包等。设置方法就是在环境变量中加入 CLASSPATH，然后就可以直接编译运行了。还以 HelloWorld 为例，例如要在根目录中运行它，那么直接在根目录下执行：

```
c:\>java HelloWorld
```

如果 CLASSPATH 没有改动的话，这样肯定会出错。那么怎么改呢？前面说过，用户类路径就是一些包含所需要的类的.jar 包和.zip 包。现在没有生成包，所以只好把 HelloWorld.class 所在的目录加到 CLASSPATH，根据前面的做法，再运行一次，成功了，换个路径，又成功了！用户不仅仅可以直接运行其中的类，当要 import 其中的某些类时，也可以同样处理。

随着系统的不断扩充，如果都加到这个环境变量里，那么这个变量会越来越臃肿。看看下面一个方法：

```
c:\> javac -classpath aPath\aPackage.jar;. myJava.java
c:\> java  -cp aPath\aPackage.jar;. myJava
```

前面提到了扩展类，Java 的扩展类就是应用程序开发者用来扩展核心平台功能的 Java 类的包(或者是 native code)，虚拟机能像使用系统类一样使用这些扩展类。如果把包放入扩展目录里，这样 CLASSPATH 也就不用设了，也就不用指定了，确实可以正确运行，但是个人认为这样不好，不能什么东西都往里搁，用户可以根据需要放入一些标准的扩展包，如 JavaServlet，Java3D 等。

## 1.2.2　包(package)

Java 中的“包”是一个比较重要的概念， package 是这样定义的：

> **Definition:** A package is a collection of related classes and interfaces that provides access protection and namespace management.

也就是：一个包就是一些提供访问保护和命名空间管理的相关类与接口的集合，使用包的目的就是使类容易查找使用，防止命名冲突，以及控制访问。

这里不过多讨论关于包的内容，只讨论和编译、运行、类路径相关的内容。至于包的其他内容，请读者自己查阅相关文档。

简单一点来说，包就是一个目录，下面的子包就是子目录，这个包里的类就是这个目录下的文件。下面用一个例子来说明。

首先创建目录结构如下 PackageTest/source/，以后根目录指的就是 PackageTest 目录，源程序放在 source 目录下。源程序如代码 1-1 所示。

**代码　1-1**

```
//PackageTest.java
package pktest;
import pktest.subpk.*;
public class PackageTest
{
    private String value;
    public PackageTest(String s)
    {
        value = s;
    }
    public void printValue()
    {
     //打印出包信息
        System.out.println("Value of PackageTest is " + value);
    }
    public static void main(String[] args)
    {
     //获取包信息
        PackageTest test = new PackageTest("This is a Test Package");
        test.printValue();
        PackageSecond second = new PackageSecond("I am in
          PackageTest");
        second.printValue();
        PackageSub sub = new PackageSub("I am in PackageTest");
        sub.printValue();
```