Java 2 Developer's Handbook

Java 2

高级开发指南

〔美〕 Philip Heller
Simon Roberts 著

邱仲潘 等译

SYBEX

PHEI

電子工業出版社
Publishing House of Electronics Industry
URL:http://www.phei.com.cn

Java 2 Developer's Handbook

Java 2高级开发指南

〔美〕 Philip Heller 著
Simon Roberts

邱仲潘 等译

电子工业出版社

内 容 提 要

本书是《Java高级开发指南》的升级版，由两位具有丰富教学经验的Java专家编写。本书包含了大量Java的最新功能，这是目前最高级的Java开发指南。

全书共分三大部分。第一部分，基础篇，介绍Java环境、小程序和应用程序、组件、布局管理器和移植性等问题。第二部分，高级课题，介绍图形、线程、动画、文件和流、数据库访问、分布式对象、内容/协议处理器等问题。第三部分，新API，介绍JDK 2（Java开发工具库）中的新功能，包括Java基础类（JFC）组件、二维API、Java Beans和安全性。最后还有一个附录，为常见问题的解答。

本书适合于Java高级开发人员及用户。

书　　名：**Java 2高级开发指南**

著　　者：〔美〕Philip Heller　Simon Roberts

译　　者：邱仲潘　等

责任编辑：高　华

印 刷 者：北京天竺颖华印刷厂

装 订 者：三河金马印装有限公司

出版发行：电子工业出版社出版、发行

北京市海淀区万寿路173信箱　邮编：100036　发行部电话：68279077

北京市海淀区翠微东里甲2号　邮编：100036　发行部电话：68207419

URL:http://www.phei.com.cn

经　　销：各地新华书店经销

开　　本：787×1092　1/16　印张：38.125　字数：970千字

版　　次：1999年6月第1版　1999年6月第1次印刷

书　　号：ISBN 7-5053-4999-6/TP·2470

定　　价：62.00 元

著作权合同登记号　图字：01-98-2482

致　谢

感谢Tom McGinn、Mike Ernest、Josh Krasnegor、James Casaletto、Charlotte Jordan、Georgianna Meagher和Mike Bridwell的支持。感谢Russel Taylor、Devon Tuck、Tim Lindholm和Urs Eberle的丰富知识和耐心的解释。

感谢图书策划人Suzanne Rotondo、Maureen Adams、Kim Crowder和Krista Reid-McLaughlin，经理编辑Laura Arendal，项目编辑Raquel Baker和编辑Mar; lyn Smith。感谢技术编辑Tim Russell和Matthew Fiedler对我们的手稿进行了技术处理，感谢生产主管Rebecca Rider的校读和组织技巧，感谢电子出版专家Kate Kaminski娴熟而迅速的排版。

特别感谢Emily Roberts提供的精彩图形（见本书选配光盘文件javadevhdbk\ch06\Emily.gif）。

前 言

在前言中，我们想简单介绍一下作者、读者和本书。

首先作个自我介绍，我们一个住在西海岸、两个住在东海岸、一个住在英格兰。我们不仅居住的地域分布很广，接触的面也很广。我们的职业是Java教员，我们向上千位学员教授过Sun公司的Java课。

我们说，同学们好，请翻开教材第一页。学生会说，这些我们已经知道了，但怎么通过TCP/IP将小程序连接到数据库呢？上练习课时，我们说，请做第33页的练习题。学生会说，我上次已经做过了，可我编写自己的布局管理器时却行不通，为什么呢？

换句话说，我们接触了Java的实际应用。我们伴着上千名Java学员进行了学习和体验。遇到麻烦时，他们就会告诉我们。理论行不通时，我们就会展开讨论。

我们用手电筒、鸡毛掸子、API页面和源代码探索了Java的各个边边角角。本书将把我们见到的东西告诉你。

我们最初为Java 1.1编写了高级开发指南。出现JDK 2时，我们觉得有必要改写，因此作了大量修改，以便读者了解这个先进工具。但本书的基本思路没有改变，就是不仅介绍这个Java工具库，而且介绍如何使用这个工具。

下面轮到你们自我介绍了，编写本书时，我们心中的读者是谁呢？

我们假设您已经读过第一本Java书籍或用过第一个Java类程序，已经知道一些理论，知道Java语言的语法，知道对象，知道大部分软件包，已经学会在API文档中查找类和方法说明，已经编写小程序和应用程序，已经对Java的一般问题有所了解，准备进入更深层次的问题了。我们要提供的正是这时所要的工具。

如果上一段反映了你对Java的掌握程序，则本书适合你使用。这是一本深入的书籍，没有关于Internet历史的介绍，没有关于对象多态的说明，James Gosling（创立Java的人）的大名也不再提及。

前言就讲这么多，很高兴与你相识，下面就可以介绍本书的内容了。

本书的内容

编写本书时，我们遵循了几个原则。第一，每一章先分析相关背景材料，然后进入正题。如果你已经对基础有所了解，可以跳过第二或第三节。

使用示例程序代码时，我们想尽办法使它切合正题，讲到线程时，示例程序是用于说明线程的，而不是卖弄我们的技巧。阅读长长的源代码比较费劲，所以我们尽可能将代码段缩短，但在实际编程中，为了使程序完整，有时需要很长的程序。本书的程序代码力求简短，但同时又保证其功能的完整性。

本书的组织

本书分成三个部分。

- 第一部分，基础篇（第1章～第5章），介绍Java环境、小程序和应用程序、组件、布局管理器和移植性等问题。尽管这些问题是基本问题，但介绍得有一定深度和广度。
- 第二部分，高级课题（第6章～第14章），介绍图形、线程、动画、文件和流、联网、数据库访问、分布式对象和内容/协议处理器等问题。
- 第三部分，新API（第15章～第18章），介绍JDK 2（Java开发工具库）中的新功能。这部分介绍Java基础类（JFC）组件、二维API、JavaBeans和安全性。

本书还有一个附录，包含常见问题的答案。

规范

本书通过各种规范的使用，更合理地表达各种信息。通篇用下面所示的提示、说明和警告以引起读者对特定问题的注意。

提示：这是提示，包含具体编程技巧。

说明：这是说明，包含重要问题的解释。

警告：这是警告，引起对错误、设计疏忽和其它问题的注意。

本书用等宽字体表示印刷程序代码、URL和文件与目录名。

本书选配光盘

本书中所有源代码和类文件都放在本书选配光盘中，我们尽可能将示例程序设计成小程序，并包括简单的Web页面，便于读者运行。有时这个办法行不通，有时则需要用应用程序来执行（特别是在与网络和文件系统相关时）。这些仍然可以从本书选配光盘中调用。

本书选配光盘中有个目录叫javadevhdbk，其中一个子目录包含本书每章的源代码，它的应用程序可以通过命令行执行，而小程序则要通过浏览器执行。每个小程序有个简单的html文件，文件名类似于小程序的子类名。建议用小程序浏览器浏览小程序。

本书引进一种新的示例程序，称为Labs（实验室）。本书不介绍Labs的源代码（但本书选配光盘中有提供）。这些labs的学习不是通过阅读源代码，而是通过执行小程序运行。例如，第4章布局管理器部分有个Lab叫做GridBagLab。为了直观地掌握GridBag布局管理器，唯一的办法是建立许多配置，混合并匹配各种栅格包限制数值的组合。其中一种生成多个组合的办法是编写大量程序代码。但是，使用GridBagLab，只要使用用户接口选择限制值，单击取一个钮，再看看结果即可。这样，就可以把Labs作为教学工具，在很短的时间内得到直观的体验。

本书选配光盘的修订和纠正内容放在出版社的Web站点http://www.Sybex.com。

本书和本书选配光盘一起带你开发健全而独立于平台的Java程序。作者很高兴看到本书问世，希望读者从中得到乐趣和实惠。

目 录

第一部分 基 础 篇

第1章 Java技术

- Java的网络支持
- Java类装入与初始化
- Java安全机制
- Java线程
- Java颜色、字体与组件支持
- GUI组件布局

Java程序运行在真实平台上实现的虚拟机中。Java程序的运行要靠环境的大量支持。

本书首先介绍Java环境问题：联网、类、安全性、文件系统、属性、线程和屏幕显示。编程人员充分理解这些材料有助于开发程序并理解Java开发工具库（JDK）的工作。

Java与联网

Java环境中最明显的组件是网络。小程序是在Internet或Intranets上活动的。Java包括下列联网支持：

- 小程序和应用程序通过类的启动在网络上用UDP或TCP/IP协议进行通信。
- Java的类装载器是用于实现网络操作而设计的。
- Java的安全机制是用于防止来自网络上的攻击的。
- 小程序中隐含的许多限制都是为了防止不良网络行为的。

为了建立健全的网络小程序和应用程序，一定要充分理解Java联网类提供的功能和Java安全模型所隐含的限制。第10章联网部分介绍Java的联网功能，第18章介绍Java的安全支持。

Java类

每一行可执行Java码都属于一个或另一个类定义。Java用类表示一切，如钮、小程序、URL、文件，甚至类本身。本节介绍类的装入及类装入在Java安全机制中所起的作用。

类的装入

考虑下列最简单的小程序：

```
public class SimplestApplet extends java.applet.Applet
{
}
```

这段程序代码绝对是什么也干不了的，但研究这个什么也不干的过程却很有意义。首先，作为小程序，应当在HTML页面中进行介绍，才能让浏览器下载和执行：

```
...
<APPLET CODE=SimplestApplet.class WIDTH=200 HEIGHT=150>
</APPLET>
...
```

用户浏览这个页面时，浏览器的HTML翻译器对其进行分析，APPLET标签表明要装入Java类。

缺省情况下（没有CODEBASE标签时），浏览器寻找与HTML页面同一目录中的小程序，即寻找文件SimplestApplet.class，如果找到该文件，则将其装入浏览器中。

这时小程序定义文件进行全面安全检查，通过字节码验证器检查类定义是否符合安全标准。例如，每个操作码都要有效，并且带有正确数量的有效变元，必须遵循访问限制，不能有上溢和下溢的操作数堆栈。

说明： Java编译器将Java源代码转变成独立于平台的字节码，可以将其想象成JVM（Java虚拟机）的机器语言。和实际硬件的机器语言一样，字节码也是一系列指令。每个指令包括一个操作码和一些操作数。

待字节码验证器验证合格后，浏览器中的Java运行环境必须建立SimplestApplet的内部表达（Internal representation）。由于每个Java类（除Object）都是从其它类扩展而来的，所以建立内部表达要两个步骤：

- 建立上级类的表达（如果还没有），这个过程可以递归（除非上级类是Object）。
- 子类所属的新数据和功能的表达。

第二步很容易：类装载器只要从网络上装入SimplestApplet类即可。第一步要表达上级Applet类。假设浏览器的Java运行环境中还没有Applet类的表达，则必须装入这个类。但Applet来自Panel类的扩展，Panel来自Container，而Container又来自Component，所以这些类都要装入。那么，Java是从何处寻找要装入的类呢？Java用下列搜索方法寻找类的定义：

- 搜索Java运行系统自己的类定义集。
- 如果没找到，则按Class Path指示的路径搜索本地文件系统中的其它位置。
- 如果还没找到，则搜索远程Web服务器。

Java用这个搜索方法保证安全和效率。下面会对这些步骤进一步介绍。

核心Java API

每个Java运行环境都有自己的标准JDK。有些厂家把这些文件直接放在上面，有些厂家

则把这些文件放在比较隐蔽的地方。这些文件通常是捆绑到一起的，但不进行压缩。无论其形式如何变化，每个运行环境都知道如何找到自己的系统类。

这个方法确保常用类文件随手可得。例如，每个Java程序都需要Object的定义，每个带GUI的Java程序都需要Component的定义。有些类虽然用得较少（如Line NumberInputStream），但一旦需要，一定要保证装入正确的标准版本类。

Class path类

如果系统中找不到所要的类文件，则运行时搜索本地机上的其它位置。如果设置了环境变量CLASS PATH，则根据其中列出的顺序搜索所列的所有目录。CLASSPATH是一列路径名，在DOS平台上用分号隔开，在其它机器上用冒号隔开。如果没有设置CLASSPATH，则只搜索当前工作目录。

下列命令在Unix机上设置CLASSPATH：

```
setenv CLASSPATH /w/x/java/classes:/y/z/morejava/classes
```

下列命令在DOS机上做同样的工作：

```
SET CLASSPATH=C:\w\x\java\classes;C:\y\z\morejava\classes
```

远程类

如果本地机上找不到applet小程序所要的类，则下一步要搜索Web服务器（如果本地机上找不到应用程序所要的类，同时又没有Web服务器，则搜索失败）。

缺省情况下，类文件应放在存放Web页面的同一目录中。这可以用CODEBASE标签覆盖。关于CODEBASE标签和软件包位置的细节，请看第5章。

可以把远程小程序类存放在Java档案（JAR）文件中（JAR文件是个容器（container）文件，与zip格式兼容）。这个方法有几个优点，特别是在小程序代码涉及多个类文件时：

- 一个JAR文件比多个类文件容易安装和管理。
- JAR文件中的所有类文件可以在一次操作中下载到客户浏览器上，而不用JAR时，多个类文件需要用多次连接进行传输。
- JAR文件可以用加密密钥进行数字签名。如果浏览器探测到来自信任源的小程序，则可以放松安全限制。例如，浏览器可以让信任小程序读取或写入本地文件系统。

第2章会介绍如何修改APPLET标签以指定JAR文件，第18章会介绍签名JAR文件。

静态初始化

类装入并通过字节码验证器检查后，类声明的任何静态变量都被分配存储空间。如果静态声明中包括初始化语句，则也在此时完成初始化工作。例如，如果类声明为：

```
Static int rev=6;
```

则分配int的空间并初始化为6。

Java编译器允许类中存在静态代码段，即码中没有方法名、参数和返回值，只有一段静态程序代码。例如，类定义可能如下：

```
Class MyClass extends MyOtherClass
{
    int             i,j;
    Static double   d=123.456;

    static
    {
      System.out.println("MyClass was just loaded.");
      d = d * 2.1;
    }
      ...
}
```

类定义中可以有多个静态代码段。静态变量初始化之后，按类定义中的顺序执行静态代码段。和静态方法一样，静态代码段只能引用所在类中的静态实例变量。

静态代码段样子古怪，难于阅读，许多编程人员对其不熟悉。不管是否需要，静态代码段总是执行，这就增加了维护的风险，所以除非确有必要，否则别用静态代码段。静态代码段的优势是运行得较早，越早越好。

静态块最常见的用途是在包含本地方法的类中。本地方法动态调用装入的库，而在本地调用发生之前将库完全装入对性能极为有利。为此，带本地方法的类中具有静态代码段，启动System.loadLibrary()调用。

静态代码在类中的另一合法用途是初始化类的静态方法要使用的静态变量。显然，这种初始化无法在构造器（constructor）中进行，因为可能不会调用构造器。如果初始化很简单，则可以在声明行中进行：

```
public class SimpleStatic
{
    static int i = 4;
    . . .
}
```

但有些初始化需要执行更复杂的代码。下列代码段显示了用静态块进行涉及捕获异常状态的初始化：

```
public class SimpleStatic2
{
    static Socket sock;
    static boolean sockOK = false;
    . . .
    static
    {
        try
        {
            sock = new Socket("dragonfly", 8765);
            sockOK = true;
        }
```

```
            catch (IOException x)  { }
        }
        . . .
    }
```

Java安全性支持

Java有许多保护机制，使客户机免遭恶意小程序的袭击。前面介绍过字节码验证器，它能防止Java运行环境装入有害类文件。一个好的Java编译器不会产生不符合字节码验证器标准的操作码，但字节码也可能是敌人炮制的，这就可以用字节码验证器来判别。

下面几节介绍其它几种安全机制，第18章会详细介绍Java安全支持。

浏览器与安全管理器

Java小程序对客户机资源的访问比应用程序要少。这不是小程序的天性，而是出于浏览器的限制。

每个浏览器都构造SecurityManager类的实例。进行某个操作之前（如文件或套接访问），包装这个操作的各个类必须取得安全管理器的许可权限。安全管理器提供或拒绝对各种功能的访问，其中文件系统和网络访问是其所控制的两个主要功能。

人们常说“小程序不能从本地或远程文件系统读取或写入”，这个说法不确切，实际情况是几乎所有支持Java的浏览器都有个安全管理器，它不让小程序进行本地或远程访问。这个限制是浏览器厂家作出的业务决定，而不是小程序内在的限制。但对于允许和不允许哪种小程序的访问则有一个普遍认可的看法。大多数现有浏览器（包括所有Netscape Navigator产品）都实现了安全管理器。这些安全管理器不让小程序进行本地或远程读取。不让运行小程序的机器作为TCP/IP服务器，只在服务器提供小程序本身时让运行小程序的机器作为TCP/IP客户机。

防止伪装

除了字节码验证器和安全管理器外，上面介绍的类装入机制是另一道防线，它防止一种称为伪装（spoofing）的攻击。

伪装就是建立与标准类同名的类，希望冒名顶替，最可能被伪装的类是控制敏感资源访问的类，在Java中，首当其冲的是SecurityManager类。可以看出，类装入算法能防止Security Manager和其它标准JDK类的假冒。

假冒者建立一个小程序作为诱饵，并把它放在服务器上的Web页面中。假冒者还建立假冒的SecurityManager.class文件，提供对任何地方的读取和写入权限。小程序、假冒Security Manager.class文件和Web页面都放在服务器上的同一目录中。假冒者希望这个假Security Manager被下载和被作为正版。

但这个攻击不会成功，因为这个假冒类不会被装入。浏览器要装入Security Manager类时，首先寻找自己仓库中的类，如果Security Manager能被马上找到，就不需要进一步搜索，假冒版本无入侵津。

Java与文件系统

小程序通常不能访问本地文件系统，但应用程序则能够访问。各种不同路径名和权限规则产生了单平台程序中没有的问题。Unix用前斜杠作为路径分隔符，支持多权限；Windows 95/98用反斜杠，要求盘号和绝对路径名，支持较少权限。

java.io.File类包装了本地规则的访问，允许Java应用程序以独立于平台的方式使用文件系统，路径分隔符和权限问题隐藏在编程人员的视线后边。FileDialog类向用户提供了独立于平台的文件选择对话框。第9章将详细介绍Java和文件系统。

Java属性

Java程序无法从本地机上读取环境变量，这是因为并不是所有平台都支持环境变量。Java提供了另一种指定环境变量的方法，称为属性（Properties）。

Java属性类似于X资源。小程序从浏览器提供的表中读取属性和属性值，而应用程序从文件或命令行取得属性和属性值。由于安全原因，小程序对本地机属性的访问受到限制。第5章将进一步介绍属性的使用和限制。

Java线程支持

由于Java环境是多线程的，编程人员可以用Java的线程建立多线程应用程序和小程序。

有些Java版本用常规机器的线程支持机制。而有些则从头开始建立自己的线程支持机制。不同平台有不同的线程支持，结果其线程表现也各不相同，主要的差别在于有的版本是分时性的，有的版本则不是。编写成功的可移植Java线程程序有赖于对不同模式的了解，并要能够建立在各种情况下均能安全运行的代码。第7章线程部分将涉及这个问题。

Java屏幕显示

编程人员都知道屏幕显示对用户的重要性。本书稍后会详细介绍屏幕显示的各个方面，这里先简单介绍颜色、字体和GUI组件。

Java颜色模式

Java颜色模式支持24位颜色，其中包括8位alpha（不透明度），提供含有160万种颜色的调色板，但是目前运行Java的机器很少有支持这么多颜色的硬件。实时运行系统要设法将请求的颜色转变为系统所能提供的颜色。

有三种颜色映射的方法。

- 采用高级计算机，实际支持真彩色。
- 将请求颜色映射到最接近的实际颜色，这种情况下，许多不同的红、绿、蓝颜色组合映射到基础平台中的同一种颜色。

· 用配色图案蒙骗肉眼。

本书选配光盘上的小程序TrueColor显示了Java颜色在读者机器上的着色方法。源代码在javadevhdbk\ch01\TureColors.java，小程序在javadevhdbk\ch01\truecolors.html。这个小程序显示3个256×256象素的矩形，矩形左边是不加蓝色时用红和绿的不同强度配成的各种颜色，红色从上由下由零到最大，绿色从左到右从零到最大；中间矩形是绿和蓝的混合，右边矩形是红和蓝的混合。

第6章将详细介绍颜色模式。

Java字体支持

和颜色一样，字体也随平台的不同而不同。Java端口支持Helvetica、Times-Roman、Courier和Dialog字体，但可以将其随意映射为任何实际字体。

说明： 在JDK 1.1中，Helvetica字体被称为SansSerif，Times-Roman被称为Serif，Courier被称为Monospaced。

Java用java.awt.Font类包装字体行为和映射。Font的构造器建立对象的实例，并请求基础视窗系统建立字体本身。字体通常都缓存在视窗系统内，所以同一字体的多次构造也无妨大雅。例如，如果小程序多次调用下列构造器：

```
Fon bigfont=new Font("Serif", Font.ITALIC, 55);
```

首次调用时，建立Font的一个实例，这是在瞬间完成的。此外，被请求的视窗系统生成55点斜体Helvetica，这很费时，特别是在X平台上。下次调用该构造器时，建立新对象，对视窗系统的请求立即返回，因为55点斜体Helvetica已经存在。

Font类有取得实例的族、样式和字号的方法，但通常用处不大。返回的并不是视窗系统的实际族、样式和字号，而只是传递到Font构造器的请求值。

第5章将详细介绍字体问题。

组件布局

GUI组件（如按钮和文本字段）在不同平台间更是大有不同。这就产生了组件布局问题。OK钮在某个平台上可能是50个象素宽，而在另一个平台上则为56个象素。程序要考虑每个可能的钮尺寸（以及滚动条大小、文本字段大小，等等）才能生成优美的布局。有时，组件出人意料地大，会把屏幕区搞得乱七八糟。

Java利用延迟技术解决这个问题，在组件生成时才确定选择。Java将精确布局的工作包装在不同的布局管理器类中。编程人员不必指明组件的准确位置，只要确定布局的策略和实现这个策略的布局管理器结构组合。第4章布局管理器中详细介绍这些结构。

组件和同级件

JDK 2引入的新Java基础类（JFC）维护一致的跨平台外观，而旧的抽象窗口工具库（AWT）组件则借用常规机器的外观和功能。

Java组件（例如钮）只是个对象，是某个类的实例。java.awt软件包隐藏了一个复杂的机制，能启动组件在本地机的视窗系统上表示组件本身。java.awt.Toolkit类是Button之类的