HZ BOOKS

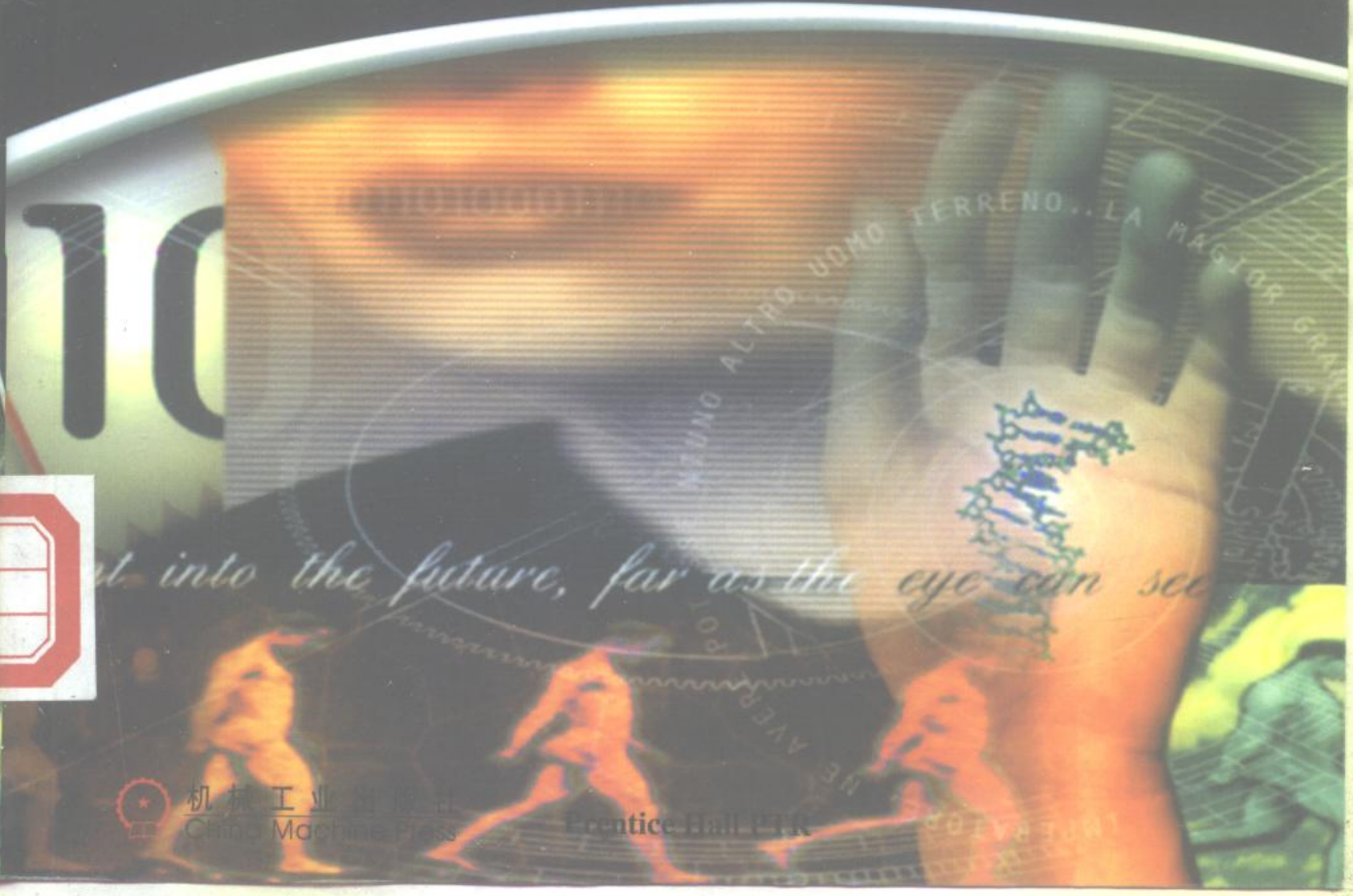
软件开发技术丛书

Sun microsystems

Jini 核心技术

Core Jini

(美) W. Keith Edwards 著　　王召福　任 鸿　刘作伟　等译



机械工业出版社
China Machine Press
Prentice Hall PTR

软件开发技术丛书

Jini 核心技术

（美） W.Keith Edwards 著
王召福 任鸿 刘作伟 等译

机械工业出版社
China Machine Press

本书全面讲解了Jini技术，包括发现、租借、远程事件、事务等主要概念，并提供真正的分布式连网技术以及可用于实际开发的Jini服务和应用程序技术。书中附有程序代码。

本书适用于程序设计人员、网络技术人员。

本书版权登记号：图字：01-1999-2345

图书在版编目（CIP）数据

Jini核心技术 /（美）爱德华兹（Edwards, W. K.）著；王召福等译. -北京：机械工业出版社，2000.7

（软件开发技术丛书）

书名原文：Core Jini

ISBN 7-111-08072-6

I. J… II. ①爱… ②王… III. 分布型网络 - 连接技术 IV.TP393.03

中国版本图书馆CIP数据核字（2000）第37428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：郭东青

北京市密云县印刷厂印刷 · 新华书店北京发行所发行

2000年7月第1版第1次印刷

787mm × 1092 mm 1/16 · 29.25印张

印数：0 001-5 000册

定价：59.00元

译者的话

Jini 技术是来自SUN公司的新的分布式系统技术，它是分布计算领域的一个重要发展，是一种构建分布式系统的全新方式。Jini技术建立在Java 软件框架之上，支持随意连入网络访问网络上服务的设备，也适用于企业分布计算。

本书的作者W.Keith Edwards 对Jini内部概念有着深入的理解，他在本书中介绍了Jini技术的体系结构和正式规范，对其中的核心概念——发现、查找、租借、远程事件、事务进行了详尽的分析。另外本书中提供的例子代码可以很好地说明 Jini技术，把它们组合起来，就可以应用到真正的Jini网络。

本书原著结构清晰，行文流畅，说理透彻，是一本难得的全面介绍Jini技术的好书。我们通过辛勤的翻译工作，尽快把它介绍给广大中文读者。无论是Java程序员，还是关心自然连入分布式系统的设备或企业计算的技术人员，都应当关注本书的内容，通过阅读本书彻底掌握Jini技术。

本书由王召福主持翻译，另外参加本书翻译工作的还有任鸿、刘作伟、刘学勤、刘志军、丁江峰、蒋永、李晓奇、王冬玲、程志全、李宇卓、刘琼；陈晓红女士负责了全书的录入校对工作。

2000年4月

序 一

20年前，我还是加州大学伯克利分校的学生，正致力于研究后来广泛流行的UNIX操作系统，那时的计算机都是由专家使用，软件也都只与相应的应用相关。在随后的20年中，我们经历了个人计算机从产生到广泛应用的过程，不过如今系统的复杂性也大大超过了20年前的系统，它曾给使用者带来了很多的不方便。我们需要一种更好的构造新系统的方法，以减少这些障碍，使计算设备与人们的使用更好地结合起来。

大约10年前，我见到一本名为《Lignes d'horizon》的书，我和我的同事John Gage仔细地阅读了该书，而且John还把它从法语翻译成英文供个人使用。在那本书中我们看到了作者对未来计算世界的构想，即“游牧式”计算，这和我们需要的更简单、更可靠、更加以人为中心的计算方式非常一致。正是这本书启发我们开始着手设计未来世界的计算基础框架，用于想象中的“游牧式”的设备和逐渐普及的嵌入式计算。

这就有了Jini。Jini使用Java语言提供一个简单的分布计算基础框架。分布计算由于可能发生局部失败而与众不同，如果我们不处理失败，那么系统在增大时将变得更加不可靠，而不是随冗余部分的增加而更加可靠。Jini提供的工具可以处理局部失败和分布计算中原本就存在的有限带宽和延迟问题。

正如那句名言“网络就是计算机”所描述的，未来的计算世界是连接的世界。Java为网络世界带来了可靠的面向对象的程序设计方式，它的快速崛起反映了面向对象技术相对于结构化程序设计的强大，以及它的现代软件设计思想带来的高可靠性和高生产率。Jini扩展了Java在分布式应用方面的优势，它被构造为一组相互通信的Java应用程序，使得用户不必理解底层支撑操作系统的细节就可以理解分布式编程的意义。

Jini很简单，我们在几天内就可以把它以及分布编程技术的原理向Java编程人员讲解清楚。通过学习Jini的思想以及远程对象、租借、分布式事件和事务的使用，读者不仅可以掌握Jini的基本内容，而且可以学习到分布式程序设计的基本原则。本书将同时面向学生、写产品应用程序的专业程序员、以及对这项新的分布计算方案和平台技术感兴趣的其他读者。

Bill Joy

阿斯本科罗拉多

1999年6月

序　二

Jini技术的早期历史有充分的文档记载，至少也有很多众所周知的轶事。在过去的6～12个月间跟踪了这项技术发展的人都知道，Billy Joy是最初的发起者并指明了方向；Jim Waldo如同首席建筑师，是真正的创建者。没有他们，这项真正的革命性技术不可能存在。

大部分人都知道，Jini技术的基础工作始于Sun公司的微系统实验室，它是在波士顿市郊的Sun公司东海岸机构的一部分。Jim Waldo在实验室中从事大规模的分布式系统研究，他与Ann Wollrath、Peter Jones以及Ken Arnold一起，研究并发展了Jini的基本概念。这些早期提出的用于解决分布式计算中相关问题的概念，如租借、分布式事件和轻量事务，是与标准方法中隐藏本地与远程系统中差别（不幸的是这是标准方法）的做法完全不同的思想。

Waldo、Wollrath、Jones和Arnold是Jini工程队伍的核心，这个队伍最终从实验室转移到JavaSoft产品组以研究如何把一个以语言为中心的方案应用到在实验室中形成的分布计算概念。这次合作的结果是，Ann Wollrath创建了Java语言中最令人感兴趣和有用的Java RMI，Ken Arnold创建了JavaSpaces技术和Jini轻事务模型。

关于Jini的早期历史，人们可能不大了解的是，当最初Bill Joy要求工作组扩展早期的工作以开发一个新的分布计算平台时，开发队伍暂时移出了JavaSoft，以便和可能扼杀新的激进思想的大公司策略隔离开。事实上，Jini诞生也得益于Sun微系统公司的人才部门，就是在这时候，Bob Scheifler（代表×国际财团）、Bryan O'Sullivan和Mark Hodapp分别以设计师和管理者的身份加入了Jini队伍，他们正是项目的发展和繁荣所必需的人，这个开发团队最终发展到由15个软件开发工程师组成。直到今天，这个位于东海岸的团队仍然保持小规模、团结和高效率的特点，Jini队伍是我见过的由最出色的人和最出色的智慧组成的团体。

在1998年夏天以前，很少有人（内部的和外部的）知道Jini计划的存在。就如同Java语言一样，这项工程的前景至少有几个月的时间是不明朗的。为了这项工程，Bill Joy和Mike Clary拿他们全部的学术声望冒险。最终由于这些人的杰出表现、由于Jini团队的敬业精神以及Sun的CEO Scott McNealy的远见卓识，这项工程得以发展起来。

Jini工程的成功再次证明了Sun所具有的以革命性的方式进行思考和行动的能力，而其他公司不愿使自己时常暴露于商业风险之中。正像Java（当时叫做Oak）传奇般地从要取消的项目中发展起来一样，Jini也曾面临同样的境遇，它从被推迟取消的项目中发展起来。这个项目能发展到目前的情况，完全归功于Sun公司提倡创新的文化。

应该注意的是，Jini技术从根本上说不仅是技术设计，而且是一种商业模式。Sun公司共享源码认证（SCSL）的概念源于Jini工程，可能有人说这样的行为还不够勇敢，但将这项关键技术共享源码，的确需要很大的勇气和远见，尤其是这样的举措还和商业社会中传统模式的安全思想背道而驰。SCSL在开发界代表一种“自愿请缨”的行动，有了它，任何人都可以参与到Jini的发展演化过程中。

我于1997年加入Jini工程。当时我在另一家公司为一项基于远程通信的大型分布式系统设计和实现中间件，那项工作令人非常兴奋，并且工作环境在我的经历中是最好的；而在有人

和我联系让我加入Jini开发组时，Jini工程正处于起步阶段，并且其未来很难确定。虽然这项工程当时的状况可能使人们不理解为什么我要加入这个其未来飘摇不定的项目，但是在仅仅听了五分钟关于Jini远景的描述后，我就知道了那正是我要去的地方。我记得自己曾想（带着对Jon Landau和Bruce Springsteen的歉意）："我看到了软件的未来，那就是Jini。"尽管向我描述这些概念时Jini只是处在早期的阶段（并且向我描述概念的人还只是一个管理人员），Jini对我来说已是如此的美妙和引起兴奋，我把它看作可能成为某种改变世界的一部分的大好时机。

当我应邀审校本书时，我的第一反应是不情愿。的确，审校一本书是很费力的；但是更重要的是，在新的技术被关注时，总是有很多低质量的书籍试图靠最新"热门"技术来牟利。开始我以为这本书也不外是此类书籍。但当我听说这本书入不仅选了Prentice Hall的著名"Core"系列，并且这本书的作者是Xerox的Palo Alto研究中心（PARC）的Keith Edwards时，我感觉到了这本书潜在的特别之处。令我高兴的是，这本书没有让我失望。

Keith Edwards清楚地理解了Jini技术内部的概念。他深入地分析了Jini的规范，并且以一种对初学者和老手都可接受的方式表述了规范中反映的设计思想。Keith同时还准确描述了Jini关于分布式计算的基本原理，强调了本地和远程系统的不同，这些描述与Jini的设计体系完全一致。Keith对Jini中的基本概念，如发现、查找、租借、分布式事件和事务的描述都非常出色，即使有经验的读者也会从中获益。

本书覆盖了作为合格的"Jini公民"为创建运行在Jini环境中的客户和服务程序必须掌握的所有知识。书中的解释全面易懂，并附有准确说明这些重要概念的例子程序。尽管多数关于Jini技术的出版物侧重于强调设备在Jini环境中的角色，但Keith以他精心编写的解释和范例，说明了Jini技术不仅可以用于设备，而且还可以应用于企业环境。

因此，无论你是一名Java程序初学者还是有经验的程序员，也无论你关心的是设备还是企业环境，本书都会使你对Jini技术有一个详尽彻底的理解。下面就让我们进入Keith Edwards的介绍，欣然领略这一构建分布式系统的全新方式。

Brain Murphy
Sun微系统公司
伯灵顿，马萨诸塞
1999年6月

前　言

本书给出了关于Jini技术的详尽的指南。Jini技术是来自sun微系统公司的新的分布式系统技术，它很有可能会改变我们创建网络软件的方式。在某种意义上，Jini代表了一种变化，即从设备网络化的世界（指计算机只具有彼此会话的能力）转变成一个真正分布式系统的世界——通过网络连接的各部分组成群体一起协同工作。

连网设备和真正的分布式系统有着十分重要的区别。简单的连网系统只具有简单的通信模式（通常是点对点和客户/服务器的形式），并且一般是静态的、长期存在的实体；而另一方面，分布式系统由很多部分构成，这些组成部分的任何一个都可以自由加入和退出，不必停止整个系统，并且系统能以可靠和可预见的方式对其环境和组成部分的变化作出响应。一言以蔽之，有了Jini，我们的世界中"系统"不再是由独立的网络设备组成，而是由这些设备组成的协同工作的集体。

本书有两个宗旨。第一，它提供对Jini核心技术的全面介绍，这些介绍对于需要理解Jini的Java开发者，以及其他想要了解Jini是如何与世界协调起来以及Jini为什么如此出色的关心技术的读者都十分有用。第二，本书深入讨论了如何利用Jini创建可工作的软件。后一个宗旨面向各类开发者，包括为PDA等小设备编制Jini驱动软件的开发者、创建基于LAN的网络系统如ch型办公/家庭办公（SOHO）和远程办公/家庭办公（ROHO）系统的开发者，甚至还有企业范围内网络服务甚至Internet本身应用的开发者。

本书的这两个目标体现在本书的前两部分中。第一部分介绍Jini的历史，论述了Jini的最终定位（尤其是强调了Jini为什么与众不同），并且就如何将其应用到实际的领域提供了大量的细节描述。第二部分深入阐述了Jini中的各个核心概念，比如租借、查找、发现和事务等，同时给出了说明Jini各个特定方面的例子。本书最后一部分是附录和参考资料。

本书的基本思路是让开发者通过阅读书中的程序来逐步学习。只要翻阅一下，读者一定能注意到，本书——尤其是第二部分，提供了大量的例子程序，其中一些是"独立的"Jini程序，它们用尽可能少的代码行来介绍Jini的基本概念，其他一些是实用程序，它们是Jini API的补充，Jini程序员可以把它们当作工具箱的一部分。

纵观全书，我提供的很多例子程序都有一个共同的主题：把它们组合到一起，就可以应用到真正的Jini网络，彼此可以互操作。这种策略本身与Jini的设计策略相一致（用大量相互配合的应用程序一起工作提供某项服务），并且使用多个离散的程序段而不是用单一的几千行的应用程序来说明概念是一种好方式。这些例子中的每一个都是用来说明Jini体系结构中一个或多个关键技术，并且读者可以根据需要进行扩展，以应用于不同的领域，从各种小型智能设备到企业级的软件系统。

本书没有对Java本身进行介绍。Jini是位于Java之上的一层，非常类似于Java基本类库（JFC）或Java的数据库互连（JDBC）。像其他层一样，Jini引入了一些新概念，是Java编程的模型的扩展，但是Jini的内核仍然是纯Java实现。本书假定读者熟悉Java编程，但不要求熟悉Java核心类库或语言的功能。

有一个例外是Java远程方法调用（RMI）系统。RMI在Jini中广泛应用。事实上，Jini使用的只是在Java 2（也称为JDK1.2）中出现的一些新特征，如RMI激活框架。因此，本书的附录A给出了RMI的简要介绍，以方便那些不熟悉RMI技术的读者。

本书的安排

本书的第一部分“基础”，提供了Jini的一些必要的背景知识，这对那些仅仅想了解Jini、知道它如何工作的读者十分有益。当然对于那些想构建实际的、可运行的Jini系统的开发人员，它也是必读的部分。这一部分的各章详细描述了Jini与经典的网络系统和分布式系统的区别。只有真正了解了这些差异，开发者才能着手编写Jini风格的软件。

第1章“一种新的计算模式”着重描述了Jini的由来和动机，在本章读者将了解到为何Jini是Java最初承诺的真正实现：对于各种各样的软件和硬件，Jini使它们简单地无需管理地组合在一起协同工作。本章同时还介绍了如何获取和安装Jini软件。

第2章“分布式系统”简单介绍了分布式系统的基本知识，以及为何Jini有别于传统的分布式系统软件。即使是熟练的网络程序员，也不应当跳过本章。Jini是一种相当独特的分布式系统，了解Jini的起源以及它要解决的问题是十分有意思的。

第3章“Jini模型”介绍了Jini的基本概念。所幸的是这些概念并不多——只有五个，了解这些概念并融会贯通，将会使以后各章的阅读更加轻松。

第4章“部署方案”介绍了部署Jini服务的各种方案。对于Jini软件的开发者来说，部署方案也就是指部署的目标。Jini大概是Sun发布的Java类库中最独特的一个，因为它被设计为支持那些可能不具有嵌入式Java虚拟机的设备。Jini能用于连接那些只有有限计算能力的设备，同时也可以用于连接运行完整Java的大型服务器或其他机器。另外这一章将帮助读者理解设计Jini应用时可用的选项。

前4章组成了本书的第一部分，它们对Jini进行了全面的介绍。本书第二部分是“基于Jini的开发”，将深入介绍Jini的特征，主要面向编写新的Jini服务程序或制造Jini设备的专业Java开发人员。以“深入理解”作标题的各章主要是深入剖析Jini技术某一特定的部分，其他章节主要是一些展示Jini在现实世界中的应用的大例子。

第5章“Jini起步”以一系列程序开始，向读者介绍Jini的概念。这一系列程序是一组“Hello,World”风格的程序，用于说明诸如查找、发现、租借以及远程事件等概念。本章是本书中第一次由浅入深地介绍Jini软件的创建，几乎覆盖了所有Jini的基本思想，并且介绍了一个RMI对象激活框架的例子。

第6章“深入理解：发现”深入地介绍Jini的发现协议。发现是Jini服务向外部世界通告自己的存在，以及Jini客户应用获知有哪些可用的服务集群或“群体”的手段。我们将深入到协议内部去了解它们，理解其中的关键机制以更有效地使用它们。学习完本章后，读者会对这些协议有深刻的了解，在需要时足以亲自实现它们。

第7章“使用属性描述服务”讨论了Jini中属性的概念。属性是与服务相关联，为服务提供描述性信息的对象，它甚至还可以提供附加的功能（如用户界面）来扩展服务的行为。本章介绍了如何使用属性以及如何把属性与JavaBeans相结合。

第8章“深入理解：使用查找服务”接着讨论Jini应用生命周期中的下一阶段——使用Jini查找服务。查找指应用如何从一个特定的群体中找到特定的服务，在本章中，我们将分析Jini

的服务和客户是如何使用查找的，并学习如何使用一些“方便的”高层API简化服务的查找责任。另外本章特别关注了如何将Jini群体连接成更大结构的问题。

到此为止，读者应该对如何构造可参与到Jini发现和查找过程中的、有一定功能和用途的客户应用有了基本的理解。通过这样的客户程序，可以找到并浏览网络上的服务。第9章“一个Jini查找服务浏览器”介绍了一个较大的应用例子，以实际应用前几章介绍的思想。这个例子是个浏览器，通过它用户可以寻找查找服务以及在查找服务中注册的其他服务，并且允许用户浏览和控制这些服务的属性。本章的代码实际是创建了一个用于显示和使用服务信息的工具包，读者可以在自己的应用程序中重用它们。

第10章“深入理解：租借”深入介绍了Jini租借的概念，也就是指Jini如何管理远程系统所拥有的资源。租借还是使Jini在大型分布式系统中具有“自修复”能力的关键。在本章，将讨论租借的优点，并介绍在租借编程中一再提及的一些常用语。

第10章主要是讨论租借的思想，并且重点介绍了客户如何使用租借，而第11章“输出被租借资源”，则主要讨论Jini服务如何实现租借，这是租借概念中的另外一个重要部分。通过这两章的学习，读者就可以全面了解Jini概念，先向客户输出租借的资源，然后客户利用租借的使用者API来控制它。

在前面各章讨论的发现、查找和租借是开发基本Jini服务的基础，而第12章“良性的服务”总结了要提供一个有用且可用的服务必须遵循的特别步骤。本章还讨论了服务管理、加入协议（用来管理服务如何与它所在的群体交互）以及如何为服务提供用户界面。

在讨论了如何构造好的服务之后，我们可以考虑创建一个完整的、复杂的Jini应用了。第13章“一个完整的例子：打印服务”给出了一个很大的服务例子，允许客户连接它打印文档。该程序支持管理、持久性和通过属性的自描述，可作为读者编写其他服务的基础。当然，这个服务也可以与本书中的其他服务相互连接。

第14章“深入理解：远程事件”介绍了Jini如何支持应用程序间的异步通知。Jini扩展了Java本地事件模型，使事件可以在不同JVM的对象间传递。远程事件模型与JavaBeans事件模型有不同的语义，本章将讨论为什么如此，并研究使用远程事件模型时的一些习惯用法。远程事件模型通过创建“适配器”并将其插入到事件管道以提供诸如事件的转发和存储等服务。本章给出了一个“事件邮箱”的例子，它在很多Jini群体中都是很有用的服务。

上面各章主要是涵盖了Jini的内核和基础，随后的两章稍有不同。它们本质上讨论的是Jini架构之上的系统服务。

第15章“JavaSpaces”介绍了Sun的JavaSpaces服务，该服务提供了一个特别有用的对象存储引擎。很多使用Jini的服务都需要一种方式存储持久性数据，以与其他服务共享使用，JavaSpaces提供了一种方便的基于对象的存储方式。另外JavaSpaces甚至可以作为一种新的分布计算模式的基础，本章简要讨论了JavaSpaces的编程模型和思想。

第16章“分布式事务”讨论了作为Jini中最重要的技术概念之一的事务。事务是协同一组操作，使它们要么全部成功完成，要么全部不进行的一种方法。对数据库编程的了解有助于了解事务的本质，即防止局部失败。Jini的事务模型源于数据库系统的两阶段提交协议，但也有些不同之处，本章讨论这些差别，并且说明了如何在Jini中使用事务。

最后，本书的两个附录提供了一些有用的参考资料和背景材料。附录A是RMI技术的基础介绍，特别地，这些特征已被添加到了Java 2中。RMI对象激活框架提供了随时从永久存储器

中恢复和激活对象的机制，这在Jini应用中广泛使用，对它的理解非常重要。一般说来，要想深入理解Jini技术，RMI的思想是必要的基础。

附录B介绍了在开发Jini应用时一些非常有用的系统特性，通过控制和调整这些特性，使用者可以控制Jini的行为和一些基本的Java特性。

本书的约定

本书的一些约定可以帮助读者更好地使用本书。通览全书，读者可以发现很多涉及各个方面的标注，这些标注称为注意、警告、提示，它们非常重要，是书中不可缺少的一部分。“警告”是为了使读者可以在开发Jini应用时避开一些潜在的陷阱；书中的例子可能包含了未完全实现的或行为与规范中不一致的Jini API部分。“提示”可以帮助读者尽可能用通用和有效的方式实现特定的设计，例子中可能包含一些共同的常用方法和模式，可以使开发工作更轻松。“注意”是一些可以使读者对Jini的某些方面有更深认识的额外信息，或仅仅是一些谈资。

从网上获取例子程序

本书所有的例子程序都被收集在一起，读者可以从ftp://ftp.prenhall.com/pub/ptr/sunsoft_books.w-053/corejini处下载。将下载的文件保存起来后，可以解压缩得到各代码文件。对于Windows系统，可以使用Winzip这样的解压缩工具，而在Solaris或其他的UNIX平台下可以使用unzip corejini.zip命令。

解压缩的过程将在相应目录下建立corejini目录。为了与本书其余部分保持一致，建议在Windows系统把文件内容解压缩到C:\files\corejini，而在Solaris系统解压缩到/files/corejini。在corejini目录中含有与各章相关的子目录。

反馈

任何书籍都不是十全十美的，尤其像这样一本探讨Jini这样的新技术的书更是如此。尽管本书经过了多次审校，但也不能保证排除了所有的错误。毫无疑问，它还有很大的改进余地。如果读者确实发现了正文或例子程序的“错误”，或者认为某些改进可以使本书对其他读者更有益，敬请发电子邮件到：kedwards@parc.xerox.com.

参考信息

本书很多章中的“参考读物”的部分，给出了一些可使读者进一步增长Jini知识的书籍、论文以及Internet网络上其他的可用资源。

最重要的，读者应该考虑访问Prentice Hall的CoreJini Web页面，以获取关于本书的更新和修订信息：http://www.phptr.com

目　录

附 录

第一部分 基 础

第1章 一种新的计算模式

主要内容：

- Jini的历史。
- Sun的Jini认证模型。
- 获取和安装Jini。
- 以命令行方式运行Jini服务。

计算机真是一种奇妙的设备，它们所能提供的计算能力、存储量以及计算速度，在几年前都是不可想象的。这种耳熟能详的说法，只是证明了我们计算机业界所发生的惊人变化。

尽管计算机发展如此迅速，当前计算机的基本结构与20世纪50年代相比仍然没有大的变化：CPU、内存以及磁盘；尽管我们可以使用计算机完成很多在40年前不可思议的事情，比如说游戏、平衡我们的收支预算等，但是从根本上说，我们仍然同我们的前辈们一样管理机器：安装软件、运行应用程序、管理系统的磁盘资源（总是不够用的资源）。一个50年代的主机系统管理员，可以毫不费力地理解我们现在的工作。事实上，我们已经变成了系统管理员——每个人都要管理自己的计算机，做那些前一辈系统管理员熟知的工作。计算机速度的提高、体积的减小，并没有为我们管理、安装和使用计算机带来质的变化。

与电话系统相比，桌面计算机的处境就显得比较暗淡。无论是容量还是连入网络的电话机数量，电话网络从一开始就一直保持指数级的增长。人们可以在世界上任何一个地方，使用标准电话线与另外的地方建立起适于语言或者数据传输的通道。尽管系统的复杂性也在以指数级增长，但对用户来说却感觉不到，我们对电话系统的感觉不会因为有外国的用户拨来了电话而变得更加复杂。尽管越来越多的地方连入了网络，但是连接任何一个地方的接口都是类似的。安装和维护自己的电话也是再容易不过了——只要把新设备插入到接口中就可以连通世界了。电话网络中最末端的一部分属于我们自己，因此我们自己就可“管理”（如果这个词还有意义的话），一切“工作正常”。

如果希望自己的机器更有用，或者说更可用，那么随着连入网络的设备的数量大量增加，我们必须在可靠性、方便配置和方便管理等方面达到电话系统那样的水平。对于电话系统，我们最常做的管理工作就是偶尔更换一个话筒，可是为什么我们当前的计算机和计算机网络却需要如此繁琐的工作才能正常工作呢？

1.1 Jini的历史

从某种意义上说，Jini的历史就是Java的历史。Java的最初目标是在各种面向用户的设备之间交换数据和代码位。Java语言最初被称为Oak，1990年出现在Sun微系统公司的实验室，设计这种语言的目的是实现一种为嵌入式处理器编写程序的可移植方法。不过，在项目完成