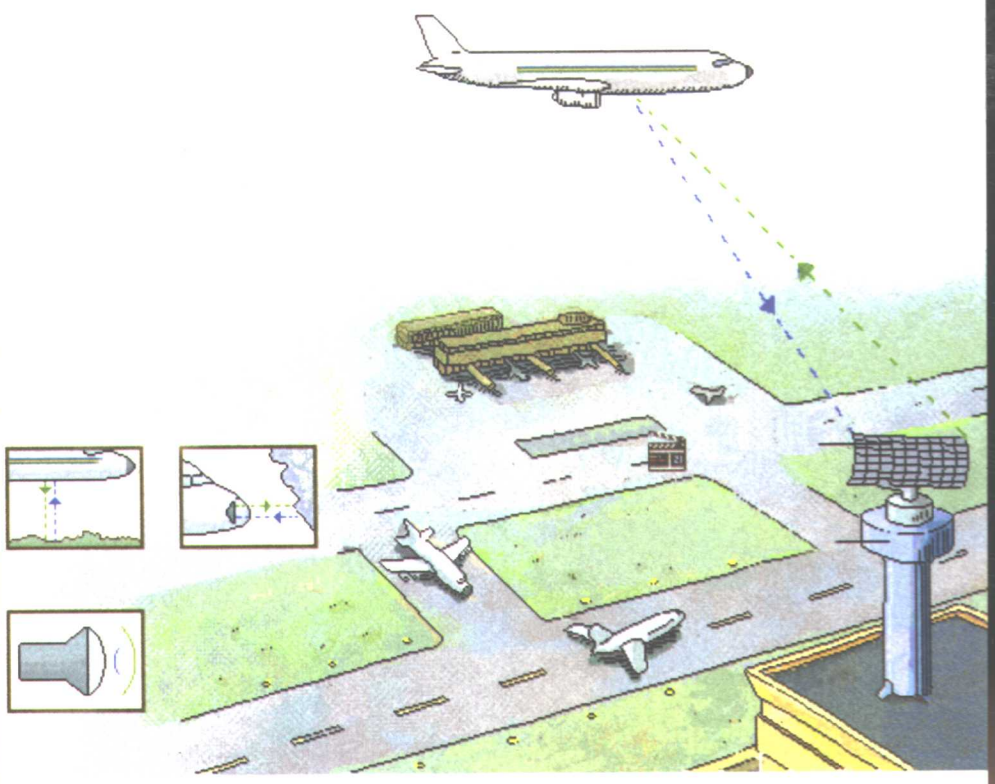
HZ BOOKS

JavaSpaces Example by Example

Sun 公司核心技术丛书

JavaSpaces 程序设计



(美) Steven L. Halter 著

钟鸣 石永平 等译

机械工业出版社 China Machine Press

PH PTR Pearson Education 培生教育出版集团

Sun公司核心技术丛书

JavaSpaces程序设计

JavaSpaces Example by Example

（美） Steven L. Halter 著

钟 鸣 石永平 等译

刘晓霞 审校

机械工业出版社
China Machine Press

PH PTR
Pearson Education
培生教育出版集团

JavaSpaces是建立在Jini之上的一种技术，它可以作为一个Jini服务，作为一种共享分布式通信的机制，还可作为一种存储对象的机制。本书详细介绍了JavaSpaces技术，举例讲解JavaSpaces的应用方法，并对JavaSpaces与Jini的关系进行了讨论。本书内容翔实、讲解透彻，适合有一定Java工作经验的程序设计人员参考。

本书版权登记号：图字：01-2002-2192

图书在版编目（CIP）数据

JavaSpaces程序设计/（美）霍尔特（Halter,S.L.）著；钟鸣等译. -北京：机械工业出版社，2002.6
（Sun公司核心技术丛书）
书名原文：JavaSpaces Example by Example
ISBN 7-111-10329-7

Ⅰ.J… Ⅱ.①霍… ②钟… Ⅲ.Java语言-程序设计 Ⅳ.TP312

中国版本图书馆CIP数据核字（2002）第032619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：于杰琼 张鸿斌
北京忠信诚胶印厂印刷·新华书店北京发行所发行
2002年6月第1版第1次印刷
787mm×1092mm 1/16·15.25印张
印数：0 001-4 000册
定价：35.00元

前　言

撰写分布式程序设计的书籍很难，涉及过分布式程序设计环境的人对这句话应该不会感到吃惊。好在程序设计技术和工具在逐渐发展，逐渐满足分布式程序设计的需求，这件工作也渐渐容易起来了。特别是JavaSpaces技术的引进使分布式计算环境简单化后，撰写这一类的书籍就容易一些了。

JavaSpaces技术是进行分布式计算的一种简单机制。在分布式计算应用程序中，JavaSpaces提供了对象的提供者和请求者可用来方便地进行通信的共享虚拟空间。这允许以Java对象的形式对任务、请求和信息进行简单的交换。JavaSpaces还提供了永久地建立和保存对象的能力。

JavaSpaces是一种Jini技术服务，这表示使用JavaSpaces的开发人员也可以利用各种Jini服务，比如说利用事务处理和通知等。

JavaSpaces以一种非常简单的接口提供了这些功能。但是，有效地利用这种简单的接口需要大量的概念和工具。本书将利用一组密切相关的例子，提供坚实的基础知识，把大家引导到正确使用JavaSpaces技术的方向上。

关于本书

本书面对的读者是希望学习JavaSpaces的中、高级程序设计员。读者需要有一定的Java工作经验，最好知道一点Jini概念（但不是必需的）。

本书将尽量给出所讲述的每个概念的完整例子。

本书结构

第一部分“JavaSpaces基础知识”，讲述使用JavaSpaces的基础知识。在第1章“关于JavaSpaces”中，介绍什么是JavaSpaces，本书中怎样研究它们以及为什么大家都想使用它们。关于“什么是JavaSpaces”，在给出某些定义的同时，还给出了相应的JavaSpace接口。关于“为什么大家都想使用它们”，给出了使用JavaSpaces的一些高度概括的描述，后面的章节中还要进一步说明。例子里包括将JavaSpaces用作一种分离通信（decoupled communication）、应用程序构造和并行计算的机制。

在第2章“获得和安装JavaSpaces”中，介绍获得JavaSpaces（以及Jini本身）并启动和运行的方法。第一次做这件事是一种挑战性的体验。这一章介绍安装Jini和JavaSpaces并运行一个简单的应用程序，详细介绍某些常见问题，如授权配置问题和不正确的代码库设置等。虽然有经验的Jini开发人员可以跳过这一章，但对于初学者来说，这一章是极有价值的。

在第3章“JavaSpaces基础”中，着手尝试实际使用JavaSpace并编写一些简单的程序。该章介绍一种定位一个JavaSpace的方法。然后介绍什么是Entry以及怎样使用它。在这里，每个概念都用简单的例子说明。

在第4章“JavaSpaces的更多介绍”中，将进一步介绍涉及JavaSpaces的更多高级内容。再次说明，每个内容都以一个简单例子的形式进行介绍。后面的章节中大量利用了这些内容。该章的目的是使读者熟悉这些内容及相应的程序块。

第二部分“分布式程序设计”介绍怎样将第一部分所学的知识用于编写分布式应用程序。第5章“分布式介绍”开始探索在分布式环境中进行程序设计的方法。分布式程序设计不同于单机（进程）中的程序设计。该章介绍了与分布式环境程序设计有关的某些概念，并利用一种连接表的概念详细地研究了分布式与非分布式数据结构之间的某些区别。

第6章“同步问题”讨论防止进程死锁的机制。这一章介绍锁定和共享问题，并给出了相应的机制，如处理资源同步的信号量等。

第7章“公平共享资源”给出了利用（并结合）前几章所介绍的许多内容的更为复杂的例子。例子应用程序给出了一个较小的定单处理系统。利用这个程序来说明怎样以一种公平可靠的方式共享数据的思想。

第三部分“进入更高层次”介绍怎样使用第一、第二部分所述的机制，帮助满足进入实际应用所必需的要求。第8章“并行计算”介绍可能会在哪些方面遇到性能问题，以及JavaSpaces有助于提高性能的方法。存在着许多改进（或降低）分布式应用性能的方法。该章讨论改善性能的某些常见技术，如确定何种计算能够从分布得到好处，以及怎样分布数据等。

第9章“安全性问题”研究某些与JavaSpaces有关的安全问题。目前不存在标准的Jini安全模型。但这并不是说，应用程序就没有安全性了。这一章介绍当前怎样将Java的某些安全功能用于JavaSpaces。讨论怎样设置策略文件并使用某些有助于应用程序级安全的简单协议。

第10章“总结”对某些内容做进一步的补充。在这一章中，还介绍一些使用或可能会使用JavaSpaces的领域。

作者介绍

Steven L.Halter是Imation公司的高级软件工程师，他在进行软件研究和开发的罗切斯特实验室工作。与别人合作撰写过《Enterprise Java Performance》一书。他曾经担任过IBM软件系统部的软件工程师，而且曾经是San Francisco性能组的成员。他对对象永久性和对象基础结构颇有研究，拥有12项已颁布的美国专利。他还做过IBM AS/400的系统软件体系结构和设计方面的工作。他在爱荷华州立大学取得过计算机科学学士学位和博士学位。

本书英文版书名：JavaSpaces Example by Example
英文版书号：ISBN 0-13-061916-7
原书出版社网址：www.phptr.com

参加本书翻译的人员有：钟鸣、石永平、王君、张文、魏允韬、田晓涛、耿娜、何江华、梅刚、孙登峰、樊伟。全书由刘晓霞审校。

目　录

第三部分 进入更高层次

第一部分

JavaSpaces基础知识

现在，关于程序设计人员是否需要了解分布式程序设计已经不是一个问题了。新的问题是，编写一个分布式应用程序需要付出多大的努力。JavaSpaces技术降低了编写分布式应用程序的复杂性，减轻了编程人员的负担。它使程序设计人员能够将较多的时间花在详细的应用程序设计之上，而不是花在分布式环境上。尽管这样，分布式环境中的程序设计仍然与传统单机程序设计环境不同。

在本书的这个部分中，介绍JavaSpaces。JavaSpaces技术提供了一个在分布式环境中进行程序设计的简单机制。本书前四章逐步介绍JavaSpaces技术及其怎样使用。这样做有助于跳过不必要的难点。由于环境和设置问题导致失败的地方有很多，在第一次遇到时要判断出这些地方是相当令人头痛的。

阅读完前四章后，就具备了使用JavaSpaces技术和更进一步理解本书其余内容的基本思想和概念了。

原书空白页

第1章

关于JavaSpaces

- 介绍JavaSpaces
- 使用JavaSpaces的理由

本章概要地介绍两个内容。首先介绍什么是JavaSpaces。从非常具体的（且简单的）接口定义到JavaSpaces究竟处于什么位置这种更为抽像的问题进行介绍。

在具有这种总体性的了解之后，我们将对JavaSpaces应用做某种高层次的描述，这些描述在后面的章节中还要进一步扩充。本章举例说明了怎样将JavaSpaces用作一种分离通信、应用程序构造和并行计算的机制。

1.1 什么是JavaSpace

“什么是JavaSpace”这个问题的答案有多种。根据提问题的角度不同，实际上对其有不同的理解。

可以从以下几个方面来了解JavaSpace：

- 纯对象风格。
- 作为Jini服务。
- 具有共享分布式通信的机制。
- 对象存储机制。

从纯对象的观点，所有JavaSpace都是稍后要考察的重要的JavaSpace接口的实现。它确实是得出那些有趣结果的一个非常小的接口。

这里是插入JavaSpaces相关术语的一个好地方。在本书中谈到“JavaSpaces技术”或JavaSpaces时，一般指的是JavaSpace的实现。在仅看到JavaSpace或space（空间）时，应该理解为JavaSpaces的一个具体运行实例。

从Jini的观点来看，JavaSpace是一个利用Jini基础结构并向其他Jini客户机和服务提供其功能的一个Jini服务。关于JavaSpaces怎样适应Jini世界还要碰做更多的介绍。

JavaSpaces提供了一种完成共享分布式计算的机制。这可能是它所提供的一种最重要的功能了。

JavaSpaces还提供了一个非常有趣且简单的对象存储机制。这并不是说它们是一种对象数据库（它们的确不是），但这确实是一个非常有用的功能。

下面将从这些方面来介绍JavaSpace。这些方面能在概念上为理解JavaSpace提供一个很好的开端。

1.1.1 JavaSpace接口

JavaSpace实际的接口定义很简短紧凑，请参阅程序清单1-1。

程序清单1-1　JavaSpacejava

```
package net.jini.space;

import net.jini.core.entry.*;
import net.jini.entry.*;
import net.jini.core.transaction.*;
import net.jini.core.event.*;
import net.jini.core.lease.*;
import java.rmi.*;

 public interface JavaSpace {
    Lease write(Entry entry, Transaction txn, long lease)
        throws TransactionException, RemoteException;

    long NO_WAIT = 0;
Entry read(Entry tmpl, Transaction txn, long timeout)
   throws UnusableEntryException, TransactionException,
          InterruptedException, RemoteException;

Entry readIfExists(Entry tmpl, Transaction txn,
                   long timeout)
   throws UnusableEntryException, TransactionException,
          InterruptedException, RemoteException;

Entry take(Entry tmpl, Transaction txn, long timeout)
   throws UnusableEntryException, TransactionException,
          InterruptedException, RemoteException;

Entry takeIfExists(Entry tmpl, Transaction txn,
```

```
                    long timeout)
        throws UnusableEntryException, TransactionException,
               InterruptedException, RemoteException;

    EventRegistration notify(Entry tmpl, Transaction txn,
                             RemoteEventListener listener,
                             long lease,
                             MarshalledObject handback)
        throws TransactionException, RemoteException;

    Entry snapshot(Entry e) throws RemoteException;
}
```

正如所见，程序清单1-1中列出的七个方法可用来提供某些非常复杂的行为的机制。不过，在遇到复杂问题时，最好从简单的地方入手。幸运的是，这些接口都很简单。

1. Entry

在学习实际的方法前，应该对Entry（项）类给予某种特殊的关注。请注意，每个方法都以一个项为参数，七个方法中有五个返回一个项。显然，在项和JavaSpaces之间具有相当重要的联系。这种联系就是，项是你放入一个JavaSpace或从一个JavaSpace取出的东西。

下面是net.jini.core.entry.Entry的接口定义：

```
package net.jini.core.entry;
public interface Entry extends java.io.Serializable {
}
```

这甚至比JavaSpace接口更简单，其中根本就没有方法。Entry接口是marker接口的一个例子。它本身不增加任何特定的指示功能。它所提供的是一个特定的类可放入一个空间的指示。

请注意，Entry接口位于程序包net.jini.core.entry之中，但JavaSpace接口在net.jini.space中。Entry接口并不仅仅用作JavaSpace用法的一个标记。Entry实际上提供了可供任何Jini服务（在1.1.2节“Jini和JavaSpaces”中作为一个可搜索的程序块学习）使用的公共接口。

除了提供关于哪些类可以放入空间的指示外，项还定义了一个JavaSpace实现怎样使用它的项实例的语义。

在建造一个实现项的类时，需要遵循以下几条规则：

- Entry子类的每个字段必须是公共的。（字段可以是非公共的，但不能把它们保存在空间中。）
- 字段不能为简单字段。它们必须是对象。
- 字段必须是可串行化的。
- 必须提供一个公共的无参数的构造函数。

第3章中介绍了许多Entry的内容，但简要地说，在那里这些规则是为了提供搜索较大项组的简单有效的机制。这实际是说，JavaSpace是什么？是一个实现Entry接口的类实例的集合。

现在回到JavaSpace接口本身的方法上。

2. Read

read方法用于在JavaSpace中查找项。从本质上来说，它提供了一种搜索JavaSpace的方法。

```
Entry read(Entry tmpl, Transaction txn, long timeout)
   throws UnusableEntryException, TransactionException,
          InterruptedException, RemoteException;
```

第一个参数是一个项，用作执行搜索的模板。如果项的某个字段为空，则空间中相同类型的任意项内的同一个字段都将匹配。这里，“类型”一词用来表示匹配项可以是与模板相同的类或者是模板的子类。

如果模板中的一个字段不为空，则相同类中其他项内的字段必须精确匹配。第3章将详细介绍匹配的确切含义。如果找到一个匹配，就会返回匹配项。如果空间中有不止一个匹配项，则空间可以返回任意匹配项。至于返回哪个项不能保证。这表示不应该指望读项的顺序会有什么对应关系（如到达的顺序等）。

第二个参数提供了一个Transaction实例，应该在其下执行读操作。第4章将介绍怎样对JavaSpaces使用事务处理。

最后一个参数为长整型，以毫秒为单位，此值说明在read方法中对于匹配项的出现在空间中要等待多久。这表示，如果一个匹配项在第一次调用read方法时不在空间中，则read方法将等待此项被添加到空间中，等待时间为此参数给出的超时值。如果等待时间超过超时值后还没有项匹配，则返回空。

3. readIfExists

readIfExists方法是一个非常类似于read的方法。它具有与read完全相同的参数和返回值。不过，readIfExists在这些参数的使用上稍有不同。它也是用来搜索空间的，而且使用相同的匹配模板实例的规则。read和readIfExists的不同之处在于超时值的处理。

readIfExists方法试图匹配作为第一个参数传入的模板项。如果未匹配，它立即返回，而不是像read方法那样等待匹配项。既然它立即返回，那么还要超时参数干什么呢？

这个问题涉及JavaSpaces怎样处理事务。一个匹配项可能位于空间中，但可能在至今尚未完成的事务处理下写过。这表示此匹配项并不真正对readIfExists方法可见。超时参数指出readIfExists等待完成此未结束的事务处理要等多久。

因此，read方法要等待直至找到匹配项或超时期满。而readIfExists方法仅在恰好存在与未结束事务处理之下的匹配项相同的匹配项时等待。

第4章将详细介绍JavaSpaces和事务处理的相互作用。

4. take

take方法也具有与read方法相同的参数和返回值。它使用项模板的相同匹配规则，其超时值与read的超时值类似，即等待直到匹配项出现为止。但其重要的差别在于，如果找到一个匹配项，不仅返回给方法的调用者，而且还从空间中删除它。

另外，如果多个客户机调用take方法且它们匹配空间中相同的项，则只有一个客户机取得该项，而其他客户机得到空返回值。

5. takeIfExists

就像readIfExists方法对应于read方法一样，takeIfExists方法对应于take方法。即，它的超时参数指出等待具有一个匹配项的未结束的事务处理完成要等待多长时间。

6. write

write方法用于把项放入空间。

```
Lease write(Entry entry, Transaction txn, long lease)
        throws TransactionException, RemoteException;
```

write以希望放入空间的项作为第一个参数。请注意，任意种类的项都可以写入一个空间。write方法也使用write所属的Transaction实例以及一个租用参数。

第4章将深入介绍租用参数，简单地说，租用表示write的调用者希望项在空间内呆多久，以毫秒为单位。

write方法的返回值为一个Lease实例。这允许调用者对将项保留在空间中有某些控制权。

7. notify

notfiy方法提供感兴趣的项被写入某个空间时到得通知的异步机制。

```
EventRegistration notify(Entry tmpl, Transaction txn,
                         RemoteEventListener listener,
                         long lease,
                         MarshalledObject handback)
        throws TransactionException, RemoteException;
```

第一个Entry参数指出匹配空间中的项时使用的模板。匹配规则与read方法的相同。与read方法不同的是notify指出调用者对只要写匹配项就得到通知感兴趣，而不是对调用时在那里的项感兴趣。

RemoteEventListener参数告诉空间送回事件给谁。在写一个新项到匹配模板的空间时，此空间发送一个事件到RemoteEventListener，以便它可以处理此事件。

Handback参数作为事件数据的组成部分被发送到监听程序。这给notify方法的请求者提供了

与Listener实例通信的一种途径。

Lease参数指出调用者在接收通知时等待多长时间，时间以毫秒表示。

EventRegistration返回的值为notify方法的调用者（如Lease实例）管理其注册提供某些信息。

第4章将介绍通知及怎样使用这些通知。

8. snapshot

它是一种优化空间性能的方法。

```
Entry snapshot (Entry e) throws RemoteException;
```

在利用相同的模板项对一个空间重复调用方法时，snapshot方法有助于提高程序的性能。snapshot工作的方式是你对传递希望优化性能的模板的空间调用它。

调用之后返回一个Entry实例，此Entry实例代表传入的项。基本上，空间会记住此新项实际上是旧项。在传递的这个新代表项调用空间时，避免了串行处理的大量开销。在模板项很大且串行化代价很高时，这样做可以极大地改善性能。

有一件需要注意的重要事情是，这仅对在其上调用snapshot方法的空间有效。如果在另一个不同的空间上调用某个方法并传递快照项，则新空间将不承认快照项代表原模板项。

另一个要点是取回的项与传入的项根本不同。不应该将新项与已经拥有的任何项进行比较。

1.1.2 Jini和JavaSpaces

Jini是JavaSpaces建立在其上的一种技术基础。如果对JavaSpace技术怎样适应Jini没有正确地理解，实际上是不可能在JavaSpace中进行编程的。

Jini提供了一种可以在其上建立分布式计算系统的基础。这听上去可能会有似曾相识的感觉，因为前面我们曾经说过JavaSpaces是一种分布式计算的机制。

对Jini进行透彻的讨论已经超出了本书的范围。如果希望对Jini有更好的理解，W.Keith Edwards撰写的《Core Jini》(Prentice Hall出版社, 2001出 版)一书是一本很好的参考书籍。Web站点www.jini.org也是了解Jini的更多信息的好所在。

本书的目的是提供足够的信息和背景知识以说明JavaSpaces适合在Jini内的哪些地方使用，并说明对JavaSpace程序员是必需的或将被证明是极为有用的Jini机制。

图1-1图示了JavaSpaces与Jini的关系。

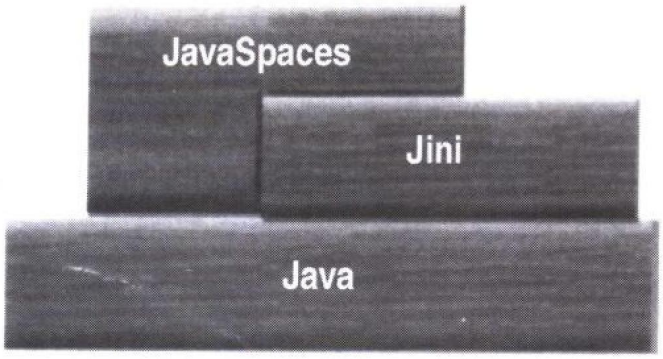


图1-1 JavaSpaces与Jini

JavaSpace是一个Jini服务。Jini服务对其他Jini服务和客户机提供功能。它通过Jini查找机制使自己可为潜在的用户所用。（第3章将详细介绍查找功能。）

因为JavaSpace是一个Jini服务，客户机可以将可从JavaSpace得到的功能与其他Jini服务和库机制相结合。当然，也可以使用所有其他的Java功能。作为Jini组成部分的优点在于Jini自身的基本功能及其他可得到的服务（如JavaSpaces本身）所提供的手段。

Jini的基本功能可分为五个方面：

- Discovery（发现）
- Lookup（查找）
- Leasing（租用）
- Event（事件）
- Transaction（事务处理）

从一个非常高的层次上看，这些功能拥有定义良好的任务。提供Discovery和Lookup作为查找东西的手段。Jini中提供的Discovery协议就是用来找到Lookup服务的东西。Lookup服务是用来找到其他Jini服务的机制。

作为分布式计算的支持机制提供了Leasing、Event和Transaction。在前面关于JavaSpaces的段落中已经简略地提到过Leasing。JavaSpaces的write方法返回一个net.jini.core.lease.Lease实例。Lease接口是作为Jini的组成部分提供的。Leasing表示事物（包括程序服务）都具有生存期这个概念。通过使不活动的事物具有自动消失的能力，Jini能够自动消除可能的废物。

Events（正如前面的“Notify”一节所述）提供一种异步通知有关成分的手段。当发出事件的一个服务中某个感兴趣的事情发生时，客户机寄存器希望接收到相应的事件。这样客户机程

序可以继续完成其他任务而不用等待该事件的发生。

Transaction提供防止部分故障的机制。这个机制决定加入（恰当地加入）一个事务处理的所有操作都成功，或者全都失败。这有助于防止出现不一致情况。

另一个对JavaSpaces应用程序有重要影响的机制是动态代码下载。这使服务和客户机能够利用直到实际运行时才会遇到的类。

1.1.3 共享分布式计算

租用、事件和事务处理的产生是由分布式程序设计的特性所决定 的。相对于一般的本地计算来说，分布式计算环境中的基础环境易出错。

因此，不应该假定服务总是存在，而是应该预见到分布式环境中不可避免地会存在问题。分布式进程通信的“标准”方法是使它们互相联络，然后直接互相传递消息。这些消息对程序员来说可以作为远程方法调用或数据包出现，但要点是在进程间建立直接连接。图1-2示出了两个进程的直接通信。

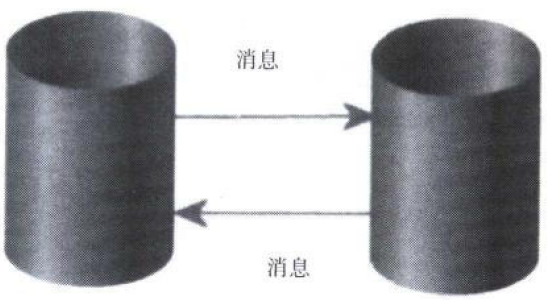


图1-2　进程对进程的通信

JavaSpaces引入了一种不同的模型。JavaSpace提供了进行通信的一种中介模型。图1-3示出这种模型。

初看上去，这似乎在分布式系统中又引入了一个可能出问题的环节。但实际功能是分离了进程。不用操心特定进程通信的细节，进程1（图1-3中）所要操心的是写一个项到JavaSpace。进程2无需关心项是怎样进入JavaSpace的，它只要取走它们，然后做自己的工作即可。

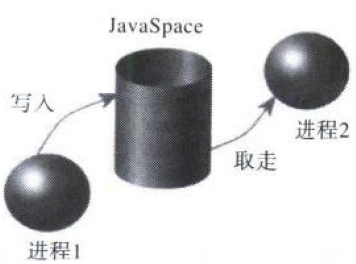


图1-3　利用JavaSpace进行进程通信