# RMI的基本介绍

## 简介

1. Java RMI（Java Remote Method Invocation），即远程方法调用，是 Java中一种用于**实现远程过程调用的应用程序编程接口**。可以**使客户机上运行的程序可以调用远程服务器上的对象**。**远程方法调用特性能够在网络环境中分布操作**。RMI 全部的宗旨就是尽可能简化远程接口对象的使用。
2. RMI（Remote Method Invocation）为远程方法调用，是**允许运行在一个 Java 虚拟机的对象调用运行在另一个 Java 虚拟机上的对象的方法**。这两个虚拟机可以是运行在相同计算机上的不同进程中，也可以是运行在网络上的不同计算机中。
3. RMI 能让一个 Java 程序去调用网络中另一台计算机的 Java 对象的方法，那么调用的效果就像是在本机上调用一样。RMI 的基础是接口，RMI 构架基于一个重要的原理：定义接口和定义接口的具体实现是分开的。

## RMI的应用

为分布式 Java 应用程序之间的远程通信提供服务，提供分布式服务。

* 1. Spring中实现了RMI

1. 在服务器端定义服务的接口，定义特定的类实现这些接口；
2. 在服务器端使用 org.springframework.remoting.rmi.RmiServiceExporter 类来注册服务；
3. 在客户端使用 org.springframework.remoting.rmi.RmiProxyFactoryBean 来实现远程服务的代理功能；
4. 在客户端定义访问与服务器端服务接口相同的类。

## RMI的局限性

1. RMI 目前使用 Java 远程消息交换协议 JRMP（Java Remote Messaging Protocol）进行通信。JRMP 是专为 Java 的远程对象制定的协议，由于 JRMP 是专为 Java 对象制定的，因此，RMI 对于用非 Java 语言开发的应用系统的支持不足。不能与用非 Java 语言书写的对象进行通信（意思是**只支持客户端和服务器端都是 Java 程序的代码的远程调用**）。
2. 由于客户机和服务器都是使用 Java 编写的，二者平台兼容性的要求仅仅是双方都运行在版本兼容的 Java 虚拟机上。

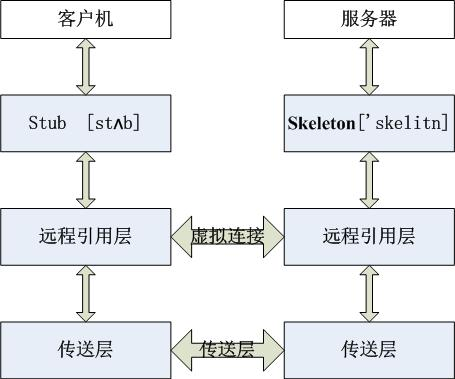
## RMI的参数

当调用远程对象上的方法时，客户机除了可以将原始类型的数据作为参数一外，还可以将对象作为参数来传递，与之相对应的是返回值，可以返回原始类型或对象，这些都是通过 Java 的对象序列化（serialization）技术来实现的。（换而言之：**参数或者返回值如果是对象的话必须实现 Serializable 接口**）

## RMI的实现组成部分

1. 远程服务的接口定义
2. 远程服务接口的具体实现
3. 桩（Stub）和框架（Skeleton）文件
4. 一个运行远程服务的服务器
5. **一个 RMI 命名服务**，它允许客户端去发现这个远程服务
6. 类文件的提供者（一个 HTTP 或者 FTP 服务器）
7. 一个需要这个远程服务的**客户端程序**

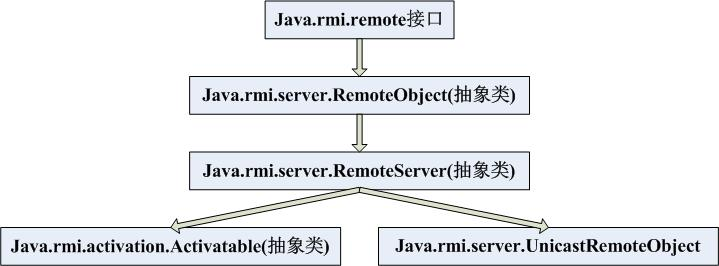
## 体系结构



1. 桩 / 框架（Stub/Skeleton）层：客户端的桩和服务器端的框架。
2. 远程引用（remote reference）层：处理远程引用行为。
3. 传送层（transport）：连接的建立和管理，以及远程对象的跟踪。

# RMI 类和接口

## Remote接口



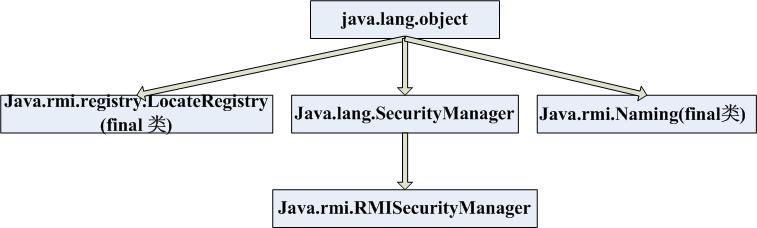
1.1 Public interface Remote{}只作为标记接口存在，不定义任何方法。

* 1. 在 RMI 中，**远程接口（服务端）声明了客户端可以从远程 Java 虚拟机中调用的方法集。**需要满足以下要求：

1. 远程接口必须直接或间接扩展 Java.rmi.Remote 接口，且必须声明为 public，除非客户端与远程接口在同一包中
2. 在远程接口中的方法在声明时，除了要抛出与应用程序有关的异常之外，还必须包括 RemoteException（或它的超类，IOExcepion 或 Exception）异常
3. **在远程方法声明中，作为参数或返回值声明的远程对象必须声明为远程接口**，而非该接口的实现类。

1.3 RemoteObject 抽象类实现了 Remote 接口和序列化 Serializable 接口，它和它的子类**提供 RMI 服务器函数**。

## 实现类和异常





* 1. LocateRegistry final() 类**用于获得特定主机的引导远程对象注册服务器程序的引用（即创建 stub）**，或者创建能在特定端口接收调用的远程对象注册服务程序。

1. 服务端

SomeService servcie=……；// 远程对象服务

1. 客户端

//返回去指定端口的远程对象的调用

Registry registry=LocateRegisty.getRegistry(int port)；

SomeService servcie=(SomeService)registry.lookup(“I serve”);//写入请求信息

Servcie.requestService();//发送请求服务

2.2 Naming类提供了存储和获取**对远程对象注册表中远程对象的引用**的方法。

(1) 服务端

**//服务端创建远程对象**

Registry registry=LocateRegistry.createRegistry(int port);

Naming.rebind(“service”,service);**//将指定名称重新绑定到一个远程对象。**

(2) 客户端

Naming.lookup(String url)// **返回与指定的name相关联的远程对象的引用**

url 格式如下 "rmi://localhost/"+ 远程对象引用

* 1. RMISecurityManager 类

1. 在 RMI 引用程序中，如果没有设置安全管理器，则只能从本地类路径加载 stub 和类，这可以确保应用程序不受由远程方法调用所下载的代码侵害
2. 在从远程主机下载代码之前必须执行以下代码来安装 RMISecurityManager:S

System.setSecurityManager（new RMISecurityManager（））；