JAVARMI简介及实例分析 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/145/2021_2022_JAVARMI_E7_AE_80_c104_145605.htm 分布式对象技术主要是在分布式异构环境下建立应用系统框架和对象构件。在应用系统框架的支撑下，开发者可以将软件功能封装为更易管理和使用的对象，这些对象可以跨越不同的软、硬件平台进行互操作。目前，分布式互操作标准主要有Microsoft的COM/DCOM标准、Sun公司的Java RMI标准和OMG组织的CORBA标准。 Java RMI简介 远程方法调用（RMI，Remote Method Invocation）是jdk1.1中引入的分布式对象软件包，它的出现大大简化了分布异构环境中Java应用之间的通信。 要使用RMI，必须构建四个主要的类：远程对象的本地接口、远程对象实现、RMI客户机和RMI服务器。RMI服务器生成远程对象实现的一个实例，并用一个专有的URL注册。RMI客户机在远程RMI服务器上查找服务对象，并将它转换成本地接口类型，然后像对待一个本地对象一样使用它。 下面是一个简单的RMI实例，RMI客户机通过RMI服务器提供的方法输出一个语句。例子虽然很简单，但掌握了Java RMI调用的基本原理和方法，在实现复杂应用时，我们需要做的也只是完善远程对象的实现类而已。RMI实例分析 1.远程对象的本地接口声明（RMIOperate.java） 该类仅仅是一个接口声明，RMI客户机可以直接使用它，RMI服务器必须通过一个远程对象来实现它，并用某个专有的URL注册它的一个实例。 远程接口扩展 java.rmi.Remote 接口。 除了所有应用程序特定的例外之外，每个方法还必须在 throws 子句中声明 java.rmi.RemoteException

（或 RemoteException 的父类）。Hello.java/* * @author javamxj （CSDN Blog) 创建日期 2004-12-27 */import java.rmi.*.// RMI本地接口必须从Remote接口派生public interface Hello extends Remote { // 接口中的具体方法声明，注意必须声明抛出RemoteException String sayHello(String name) throws RemoteException.}2.远程对象实现类 这个类应实现RMI客户机调用的远程服务对象的本地接口，它必须从UnicastRemoteObject继承，构造函数应抛出RemoteException异常。 HelloImpl.java /* * @author javamxj （CSDN Blog) 创建日期 2004-12-27 */import java.rmi.*.import javax.rmi.PortableRemoteObject.public class HelloImpl extends PortableRemoteObject implements Hello { /* 构造函数 */ public HelloImpl() throws RemoteException { super(). } /* 实现本地接口中声明的sayHello()方法 */ public String sayHello(String message) throws RemoteException { System.out.println("我在RMI的服务器端，客户端正在调用sayHello方法。 "). System.out.println("Hello " message). return message. }}3.RMI服务器类 该类创建远程对象实现类HelloImpl的一个实例，然后通过一个专有的URL来注册它。所谓注册就是通过Java.rmi.Naming.bind()方法或Java.rmi.Naming.rebind()方法，将HelloImpl实例绑定到指定的URL上。HelloServer.java/* * @author javamxj （CSDN Blog) 创建日期 2004-12-27 */import java.rmi.*.public class HelloServer { public static void main(String[] args) { // 在服务器端设置安全机制 /* if (System.getSecurityManager() == null) { System.setSecurityManager(new RMISecurityManager()). } */ try {

System.out.println("开始 RMI Server ..."). /* 创建远程对象的实现实例 */ HelloImpl hImpl = new HelloImpl(). System.out.println("将实例注册到专有的URL "). Naming.rebind("HelloService", hImpl). System.out.println("等待RMI客户端调用..."). System.out.println(""). } catch (Exception e) { System.out.println("错误: " e). } }}请注意有关 rebind 方法调用的下列参数： 第一个参数是 URL 格式的 java.lang.String，表示远程对象的位置和名字。 需要将 myhost 的值更改为服务器名或 IP 地址。否则，如果在 URL 中省略，则主机缺省值为当前主机，而且在 URL 中无需指定协议（例如“ HelloServer ”）。 在 URL 中，可以选择提供端口号：例如“ //myhost:1234/HelloServer ”。端口缺省值为 1099。除非服务器在缺省 1099 端口上创建注册服务程序，否则需要指定端口号。 第二个参数为从中调用远程方法的对象实现引用。 RMI 运行时将用对远程对象 stub 程序的引用代替由 hImpl 参数指定的实际远程对象引用。远程实现对象（如 HelloImpl 实例）将始终不离开创建它们的虚拟机。因此，当客户机在服务器的远程对象注册服务程序中执行查找时，将返回包含该实现的 stub 程序的对象。 