**一、RMI概念：**

RMI(Remote Method Invocation)远程方法调用是一种计算机之间利用远程对象互相调用实现双方通讯的一种通讯机制。使用这种机制，某一台计算机上的对象可以调用另外一台计算机上的对象来获取远程数据。RMI是Enterprise JavaBeans的支柱，是建立分布式Java应用程序的方便途径。在过去，TCP/IP套接字通讯是远程通讯的主要手段，但此开发方式没有使用面向对象的方式实现开发，在开发一个如此的通讯机制时往往令程序员赶到乏味，对此PRC(Remote Procedure Call)应运而生，它使程序员熔岩地调用远程程序，但在面对复杂的信息传讯时，RPC依然未能很好的支持，而且RPC未能做到面向对象调用的开发模式。针对PRC服务遗留的问题，RMI出现在世人面前，它被设计成一种面向对象的通讯方式，允许程序员使用远程对象来实现通信，并且支持多线程的服务，这是一次远程通讯的革命，为远程通信开辟新的里程碑。

1. **RMI的开发步骤：**
2. 先创建远程接口及声明远程方法，注意这是实现双方通讯的接口，需要继承Remote。
3. 开发一个类来实现远程接口及远程方法，值得注意的是实现类需要继承UnicastRemoteObject。
4. 通过javac命令编译文件，通过java -server命令注册服务，启动远程对象。
5. 最后客户端查找远程对象，并调用远程方法。
6. **Java RMI之HelloWorld：**

1、定义一个远程接口，必须继承Remote接口，其中需要远程调用的方法必须抛出RemoteException异常

**import** java.rmi.Remote;

**import** java.rmi.RemoteException;

**public** **interface** IHello **extends** Remote {

//简单的返回“Hello World！"

**public** String helloWorld() **throws** RemoteException;

//一个简单的业务方法，根据传入的人名返回相应的问候语

**public** String sayHelloToSomeBody(String someBodyName) **throws** RemoteException;

}

1. 远程的接口实现

**import** java.rmi.RemoteException;

**import** java.rmi.server.UnicastRemoteObject;

**public** **class** HelloImpl **extends** UnicastRemoteObject **implements** IHello {

//因为UnicastRemoteObject的构造方法抛出了RemoteException异常，因此这里默认的构造方法必须写，必须声明抛出RemoteException异常

**protected** HelloImpl() **throws** RemoteException {

**super**();

}

//简单的返回“Hello world!”字样

@Override

**public** String helloWorld() **throws** RemoteException {

**return** "Hello world";

}

//一个简单的业务方法，根据传入的人名返回相应的问候语

@Override

**public** String sayHelloToSomeBody(String someBodyName) **throws** RemoteException {

**return** "你好，" + someBodyName + "!";

}

}

3、创建RMI注册表，启动RMI服务，并将远程对象注册到RMI注册表中。

**import** java.rmi.RemoteException;

**import** java.rmi.registry.LocateRegistry;

**public** **class** HelloServer {

**public** **static** **void** main(String[] args) {

**try** {

//创建一个远程对象

IHello rhello = **new** HelloImpl();

//本地主机上的远程对象注册表Registry的实例，并指定端口为8888，这一步必不可少（Java默认端口是1099），

//必不可缺的一步，缺少注册表创建，则无法绑定对象到远程注册表上

LocateRegistry.*createRegistry*(8888);

//把远程对象注册到RMI注册服务器上，并命名为RHello

//绑定的URL标准格式为：rmi://host:port/name(其中协议名可以省略，下面两种写法都是正确的)

Naming.*bind*("rmi://localhost:8888/RHello", rhello);

//Naming.bind("//localhost:8888/RHello", rhello);

System.***out***.println(">>>>>INFO:远程IHello对象绑定成功！");

} **catch** (RemoteException e) {

System.***out***.println("创建远程对象发生异常！");

e.printStackTrace();

} **catch** (MalformedURLException e) {

System.***out***.println("发生重复绑定对象异常！");

e.printStackTrace();

} **catch** (AlreadyBoundException e) {

System.***out***.println("发送URL出现异常！");

e.printStackTrace();

}

}

}

4、客户端测试，在客户端调用远程对象上的远程方法，并返回结果

**import** java.net.MalformedURLException;

**import** java.rmi.Naming;

**import** java.rmi.NotBoundException;

**import** java.rmi.RemoteException;

**public** **class** HelloClient {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

IHello rhello;

**try** {

//在RMI服务注册表中查找名称为RHello的对象，并调用其上的方法

rhello = (IHello)Naming.*lookup*("rmi://localhost:8888/RHello");

System.***out***.println(rhello.helloWorld());

System.***out***.println(rhello.sayHelloToSomeBody("张三"));

} **catch** (MalformedURLException | RemoteException | NotBoundException e) {

e.printStackTrace();

}

}

}

5、打开一个命令提示符窗口运行RMI服务端程序：

java -server HelloServer

6、打开另一个命令提示符窗口运行RMI客户务端程序：

java HelloClient

1. **总结**

从上面的过程来看，RMI对服务器的IP地址和端口依赖很紧密，但是在开发的时候不知道将来的服务器IP和端口如何，但是客户端程序依赖这个IP和端口。

这也是RMI的局限性之一。这个问题有两种解决途径：一是通过DNS来解决，二是通过封装将IP暴露到程序代码之外。

RMI的局限性之二是RMI是Java语言的远程调用，两端的程序语言必须是Java实现，对于不同语言间的通讯可以考虑用Web Service或者公用对象请求代理体系（CORBA）来实现。