

Java学习笔记（十六）——Java RMI

【前面的话】

最近过的好舒服，每天过的感觉很充实，一些生活和工作的技巧注意了就会发现，其实生活也是可以过的如此的有滋有味，满足现在的状况，并且感觉很幸福。

学习Java RMI的原因是最近在使用dubbo框架做一个系统，所以对这Java RMI进行学习，做一些笔记，[基础性文章，选择性阅读](#)。

【定义】

**Java RMI:** Java远程方法调用，即Java RMI (Java Remote Method Invocation) 是Java编程语言里，一种用于实现远程过程调用的应用程序编程接口。它使客户机上运行的程序可以调用远程服务器上的对象。远程方法调用特性使Java编程人员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。

我们知道远程过程调用（Remote Procedure Call, RPC）可以用于一个进程调用另一个进程（很可能在另一个远程主机上）中的过程，从而提供了过程的分布能力。Java 的 RMI 则在 RPC 的基础上向前又迈进了一步，即提供分布式对象间的通讯。

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。

【JavaRMI】

一、工作原理

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。通俗的讲：A机器上面有一个class，通过远程调用，B机器调用这个class 中的方法。

RMI，远程方法调用（Remote Method Invocation）是Enterprise JavaBeans的支柱，是建立分布式Java应用程序的方便途径。RMI是很容易使用的，但是它非常的强大。

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。下面我们通过具体的例子，建立一个简单的远程计算服务和使用它的客户端程序

二、RMI包含部分：

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

三、RMI的用途？

RMI的用途是为分布式Java应用程序之间的远程通信提供服务，提供分布式服务。

目前主要应用时封装在各个J2EE项目框架中，例如Spring，EJB（Spring和EJB均封装了RMI技术）

在Spring中实现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

四、RMI的局限？

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端都是Java程序的代码的远程调用）。

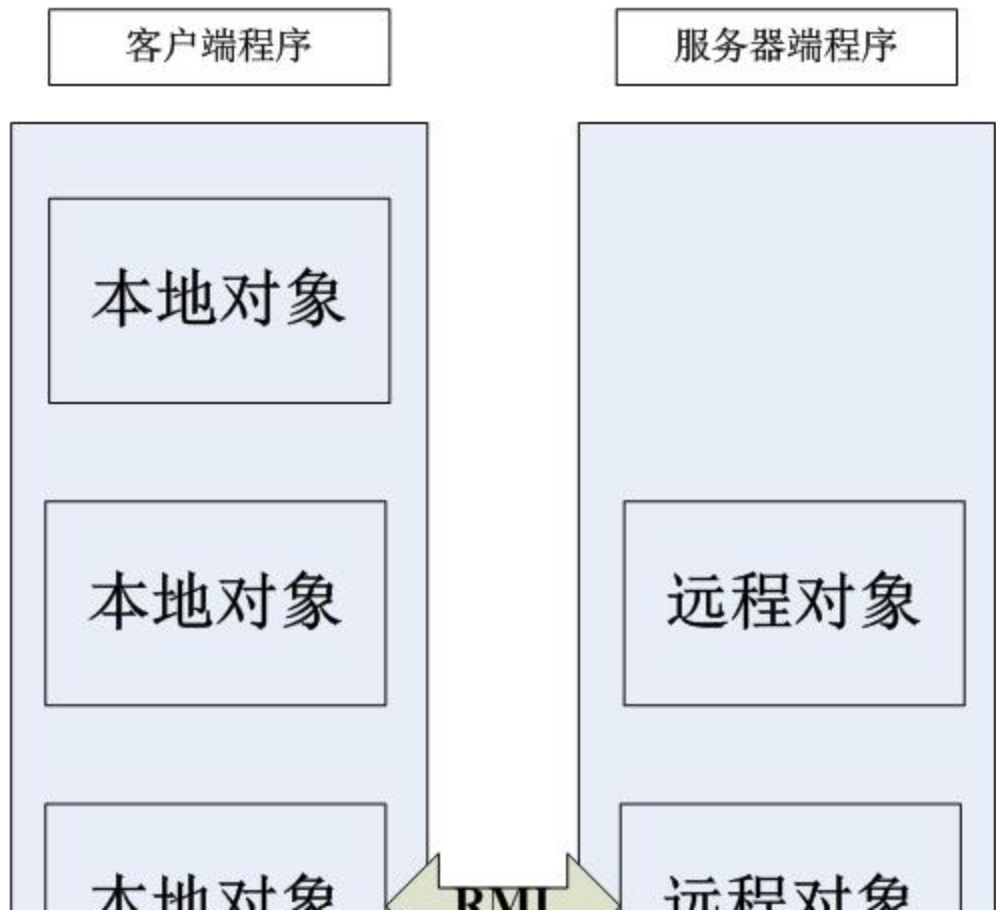
五、RMI的使用局限？

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。

六、RMI调用远程方法的参数和返回值

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。（换言之之：参数或者返回值如果是对象的话必须实现Serializable接口）

七、RMI应用程序的基本模型



公告

昵称：牲口TT  
园龄：5年6个月  
粉丝：131  
关注：22  
+加关注

< 2019年4月 >						
日	一	二	三	四	五	六
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	1	2	3	4
5	6	7	8	9	10	11

搜索

找找看

谷歌搜索

常用链接

我的随笔  
我的评论  
我的参与  
最新评论  
我的标签

17  
推荐

2  
反对

随笔分类(45)

- [01]总结计划(3)
- [02]Java笔记(23)
- [03]Spring(3)
- [04]产品学习 (1)
- [05]项目管理(4)
- [06]实用性书籍(2)
- [07]思考文学(4)
- [08]算法学习(1)
- [09]测试(4)

随笔档案(45)

- 2016年3月 (1)
- 2016年1月 (1)
- 2015年12月 (1)
- 2015年11月 (1)
- 2015年8月 (3)
- 2015年7月 (1)
- 2015年6月 (1)
- 2014年4月 (9)
- 2014年3月 (9)
- 2014年2月 (11)
- 2014年1月 (2)
- 2013年11月 (5)

最新评论

1. Re:Java学习笔记（十六）——Java RMI  
@温暖的陌生人没啥问题,他说的打包成Jar,其实就是 虚拟远程对象,利用client端的虚拟对象来获取远程对象的值...  
--HeIloWorld
2. Re:Java学习笔记（十六）——Java RMI  
你这代码有错误  
--温暖的陌生人
3. Re:Java学习笔记（十六）——Java RMI  
我在将服务端发布到 阿里云遇到几个问题，借此地备注分享一下00. 阿里云实例，安全组问题，需要开放端口哦。01. 返回内网问题，需要在服务端添加：System.setProperty("java.rmi.....  
--凌海森
4. Re:Java学习笔记（十六）——Java RMI  
这代码你自己 运行过？  
--凌海森
5. Re:项目管理学习——《构建之法》读书笔记  
@ff引用“4，如果作为大学教程，MSF流程章节，其实没有必要单独成为一章。”额，我看的时候还是觉的MSF挺有必要的。不过我赞同之前另外一个读者的看法：建议插入界限清晰的“部分”，有许多书采用这种.....  
--SoftwareTeacher

阅读排行榜

1. Java学习笔记（十六）——Java RMI(14867)
2. Java学习笔记（十八）——Java DTO(10625)
3. 《如何阅读一本书》读书笔记(8168)
4. 反思自己(4046)
5. Java学习笔记（十九）——Java 日志记录 AND log4j(3069)

评论排行榜

1. 反思自己(71)
2. 壹贰壹肆(10)
3. 《如何阅读一本书》读书笔记(7)
4. Java学习笔记（八）——java多线程(6)
5. Java学习笔记（十六）——Java RMI(4)

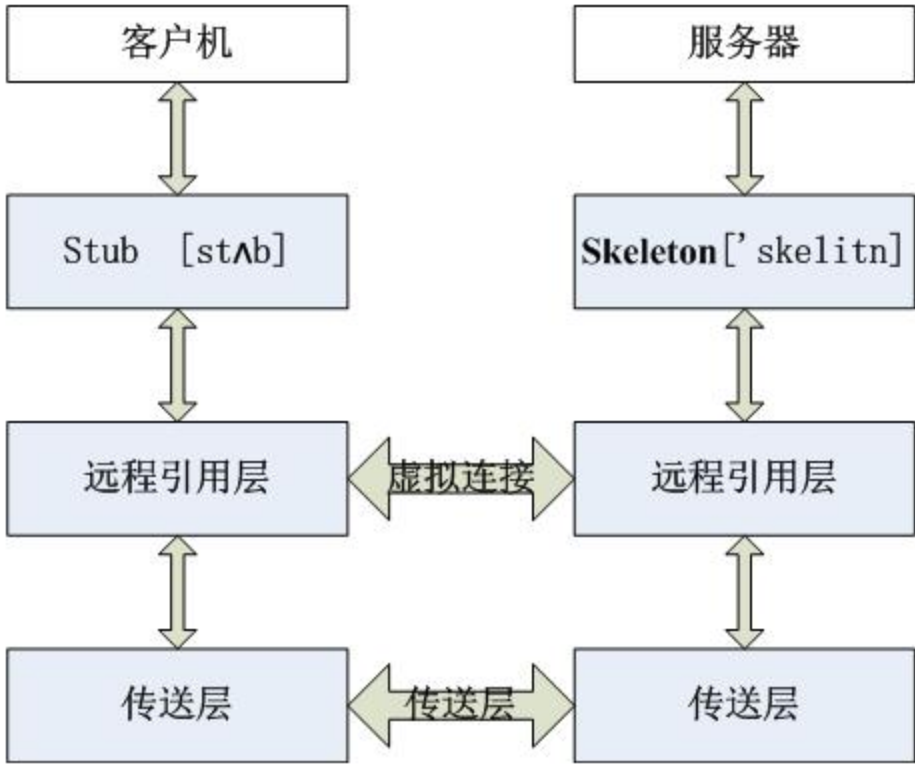
推荐排行榜

1. 反思自己(50)
2. 《如何阅读一本书》读书笔记(29)
3. Java学习笔记（十六）——Java RMI(17)
4. Java学习笔记（十九）——Java 日志记录 AND log4j(9)
5. 项目日志：搭建项目，项目日志(166)



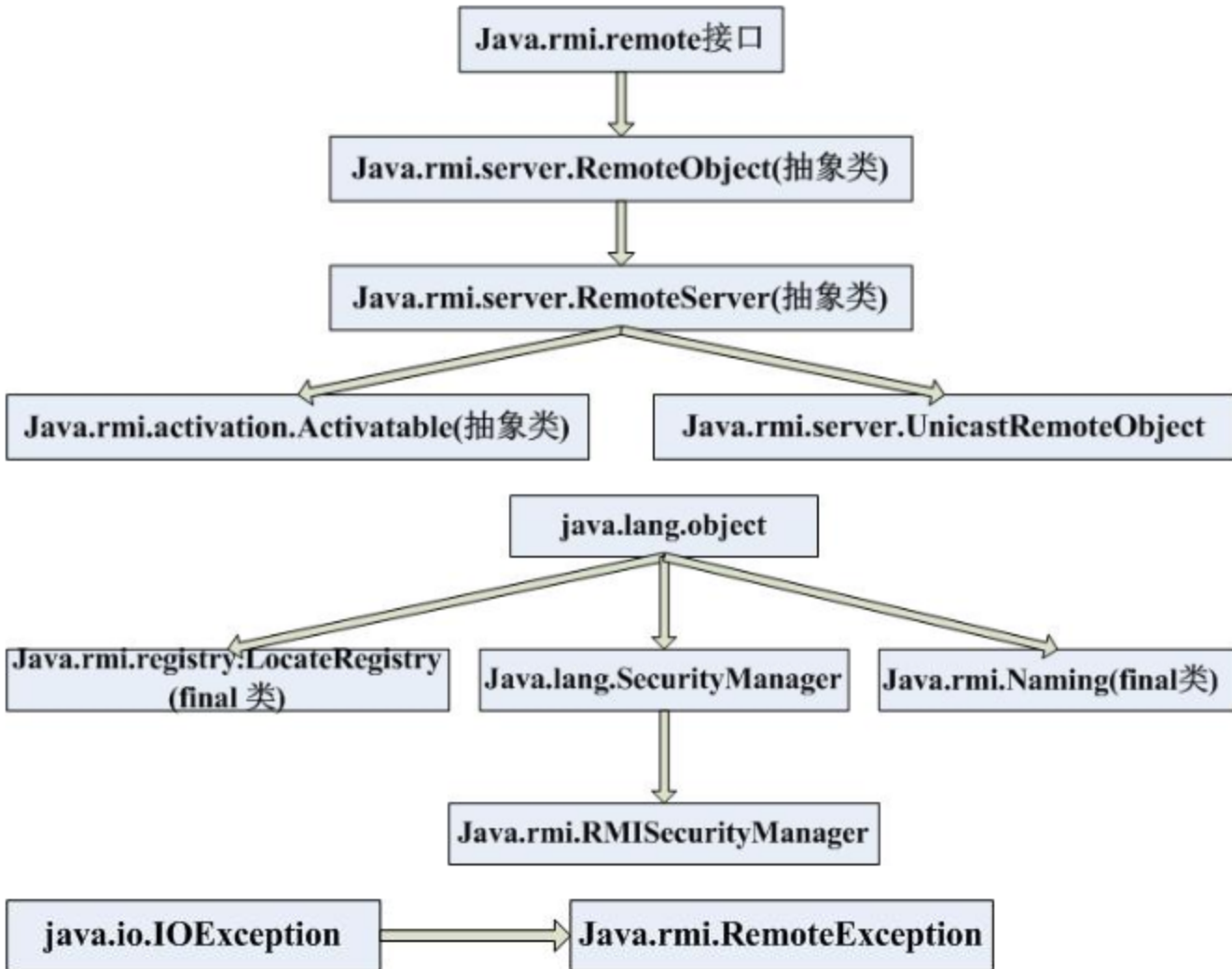


八、RMI体系结构



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

九、RMI类和接口 (完成一个简单RMI需要用到的类)。



(一) Remote接口: 是一个不定义方法的标记接口

```
Public interface Remote{}
```

在RMI中, 远程接口声明了可以从远程Java虚拟机中调用的方法集。远程接口满足下列要求:

- 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, 除非客户端于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一场之外, 还必须包括RemoteException (或它的超类, IOException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。

(二) 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。

(三) 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。

服务器端: 向其他客户机提供远程对象服务

```
SomeService servcie= .....; //远程对象服务
```

1. Registry registry=LocateRegisty.getRegistry(); //Registry是个接口, 他继承了Remote, 此方法返回本地主机在默认注册表端口 1099 上对远程对象 Registry 的引用。
2. getRegistry(int port) 返回本地主机在指定 port 上对远程对象 Registry 的引用;
3. getRegistry(String host) 返回指定 host 在默认注册表端口 1099 上对远程对象 Registry 的引用;
4. getRegistry(String host, int port) 返回指定的 host 和 port 上对远程对象 Registry 的引用
5. registry.bind("I serve",service);// bind(String name,Remote obj) 绑定对此注册表中指定 name 的远程引用。name : 与该远程引用相关的名称 obj : 对远程对象 (通常是一个stub) 的引用
6. unbind (String name) 移除注册表中指定name的绑定。
7. rebind (String name,Remote obj) 重新绑定, 如果name已存在, 但是Remote不一样则替换, 如果Remote一样则丢弃现有的绑定
8. lookup(String name) 返回注册表中绑定到指定 name 的远程引用, 返回Remote
9. String[] list() 返回在此注册表中绑定的名称的数组。该数组将包含一个此注册表中调用此方法时绑定的名称快照。

客户端: 向服务器提供相应的服务请求。

```
Registry registry=LocateRegisty.getRegistry() ;  
SomeService servcie=(SomeService)registry.lookup( "I serve" );  
Servcie.requestService();
```

(四) Naming类和Registry类类似。

客户端:

```
Naming.lookup(String url)  
  
url 格式如下"rmi://localhost/"+远程对象引用
```

服务器端:

```
Registry registry=LocateRegistry.createRegistry(int port);  
Naming.rebind("service",service);
```

(五) RMISecurityManager类

在RMI引应用程序中, 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

在从远程主机下载代码之前必须执行以下代码来安装RMISecurityManager:

```
System.setSecurityManager (new RMISecurityManager () ) ;
```

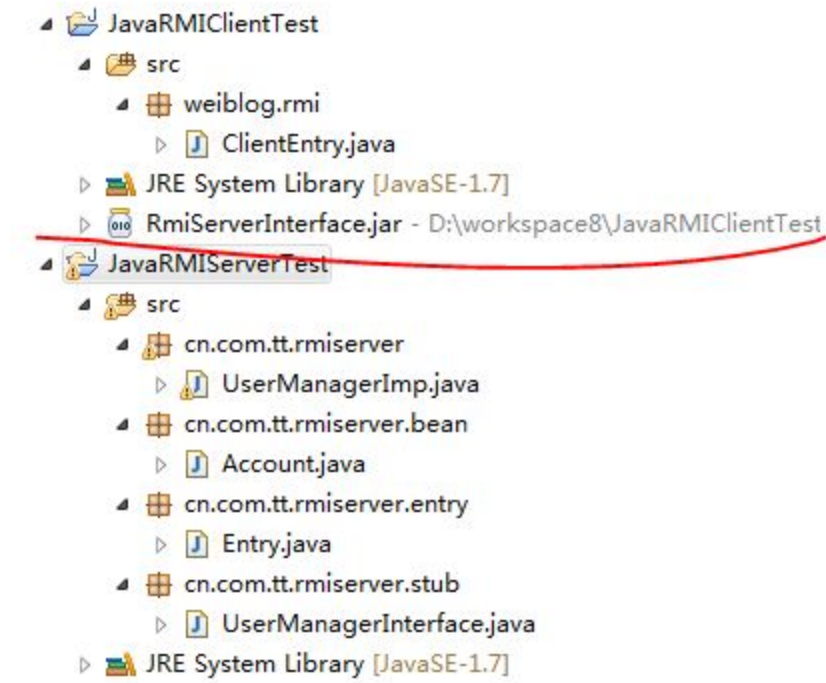
## 十、demo开发

为了编写一个demo，我们分为两部分，一部分是server端的代码，一部分是client端的代码，client端的代码主要是为了使用server端的代码。当然这个代码是非常简单的，只是为了说明问题，现实的使用会使比较复杂的。

### (一) 我们的目的

建立一个server端的java project，包含远程端的代码，定义接口，定义接口实现，然后在建立一个client端的java project，通过RMI使用远端服务中的方法。

### (二) 我们的代码结构



### (三) 远程服务代码

#### 1. 远程服务的接口定义

第一步就是建立和编译服务接口的Java代码。这个接口定义了所有的提供远程服务的功能，下面是源程序：

UserManagerInterface.java

```
1 package cn.com.tt.rmiserwer.stub;
2
3 import java.rmi.Remote;
4 import java.rmi.RemoteException;
5
6 import cn.com.tt.rmiserwer.bean.Account;
7
8 public interface UserManagerInterface extends Remote{
9     public String getUsername() throws RemoteException;
10    public Account getAdminAccount() throws RemoteException;
11 }
```

接口必须继承Remote类，每一个定义地方法都要抛出RemoteException异常对象。

#### 2. 接口的具体实现

第二步就是对于上面的接口进行实现：

UserManagerImp.java

```
1 package cn.com.tt.rmiserwer;
2
3 import java.rmi.RemoteException;
4
5 import cn.com.tt.rmiserwer.stub.UserManagerInterface;
6 import cn.com.tt.rmiserwer.bean.Account;
7
8 public class UserManagerImp implements UserManagerInterface {
9     public UserManagerImp() throws RemoteException {
10
11     }
12     private static final long serialVersionUID = -3111492742628447261L;
13
14     public String getUsername() throws RemoteException{
15         return "TT";
16     }
17     public Account getAdminAccount() throws RemoteException{
18         Account account=new Account();
19         account.setUsername("TT");
20         account.setPassword("123456");
21         return account;
22     }
23 }
```

3. 定义一个bean，实现implements Serializable序列化接口。也就是可以在client和server端进行传输的可序列化对象。

Account.java

```
1 package cn.com.tt.rmiserwer.bean;
2
3 import java.io.Serializable;
4
5 public class Account implements Serializable,Cloneable{
6     private static final long serialVersionUID = -1858518369668584532L;
7     private String username;
8     private String password;
9
10    public String getUsername() {
11        return username;
12    }
13    public void setUsername(String username) {
14        this.username = username;
15    }
16    public String getPassword() {
17        return password;
18    }
19    public void setPassword(String password) {
20        this.password = password;
21    }
22 }
```

#### 4. 定义server端的主程序入口。

Entry.java

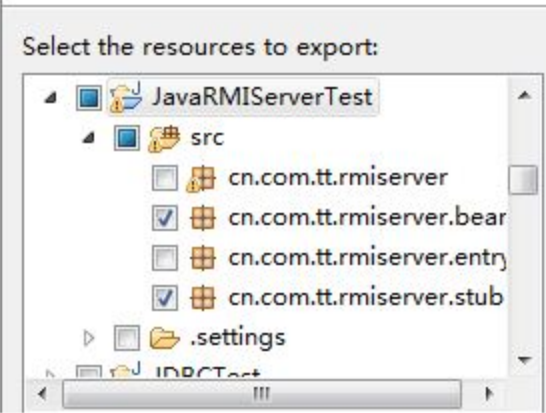
```
1 package cn.com.tt.rmiserwer.entry;
```



```
2
3 import java.rmi.AlreadyBoundException;
4 import java.rmi.RemoteException;
5 import java.rmi.registry.LocateRegistry;
6 import java.rmi.registry.Registry;
7 import java.rmi.server.UnicastRemoteObject;
8
9 import cn.com.tt.rmserver.UserManagerImp;
10 import cn.com.tt.rmserver.stub.UserManagerInterface;
11
12 public class Entry {
13     public static void main(String []args) throws AlreadyBoundException,
RemoteException{
14         UserManagerImp userManager=new UserManagerImp();
15         UserManagerInterface userManagerI=
(UserManagerInterface)UnicastRemoteObject.exportObject(userManager,0);
16         // Bind the remote object's stub in the registry
17         Registry registry = LocateRegistry.createRegistry(2002);
18
19         registry.rebind("userManager", userManagerI);
20         System.out.println("server is ready");
21     }
22 }
```

(四) 客户端代码

1. 把Server端的Account类和接口UserManagerInterface 导出Export成jar包，命名为：RmiServerInterface.jar。导入到client中。
2. 项目——右键——Export——java——jar file——next——选择Account类和接口UserManagerInterface——命名为：RmiServerInterface.jar如下图：



3. 新建一个java Project，导入jar包，编写客户端代码。
4. 代码

```
ClientEntry.java

1 package weiblog.rmi;
2
3 import java.rmi.NotBoundException;
4 import java.rmi.RemoteException;
5 import java.rmi.registry.LocateRegistry;
6 import java.rmi.registry.Registry;
7
8 import cn.com.tt.rmserver.stub.UserManagerInterface;
9
10 public class ClientEntry {
11
12     public static void main(String []args){
13
14         try {
15             Registry registry = LocateRegistry.getRegistry("localhost",2004);
16             UserManagerInterface userManager =
(UserManagerInterface)registry.lookup("userManager");
17             System.out.println("用户名是"+userManager.getAdminAccount().getUsername()
18                 +"密码"+userManager.getAdminAccount().getPassword());
19         } catch (RemoteException e) {
20             // TODO Auto-generated catch block
21             e.printStackTrace();
22         } catch (NotBoundException e) {
23             // TODO Auto-generated catch block
24             e.printStackTrace();
25         }
26     }
27 }
28
29 }
```

5. 先运行服务器端代码， 然后运行客户端代码， 就会显示运行结果， 客户端可以运行多次， 每次都可以取得服务器端的对象。如果要再次运行客户端代码就需要更改端口号， 如果不更改就会显示端口号被占用。


【参考资料】

1. demo参考的下面文章：  
[RMI网络编程开发之二 如何搭建基于JDK1.5的分布式JAVA RMI 程序](#)
2. 理论知识参考的下面文章：  
[RMI入门教程](#)  
[java\\_RMI技术讲解](#)

【后面的话】

快要放假了多么值得高兴。

时至今日都是我咎由自取，学java那是自找，与任何人无关。大学生活过的平顺，造就了我轻信java，编写众多bug的脾气，导致今日岌岌可危的地步，我今天愿意承担一切后果。其实，我很感谢你们让我在测试的时候发现bug，而不是到了生产环境，我必须重新梳理我所造成的bug，坦然面对每个bug。我，在学java的路上编写了大量的bug。我辜负了c和c++，辜负了c#，辜负了程序员和码农的称呼，辜负了所有以为我可以写出漂亮代码的人。对不起，请接收我发自内心的歉意和忏悔。晚上和周末本来有一个温暖美的生活，可是这一切被我学习java给打破了，我学java的行为不配得到原谅，我造成的无休止的bug现状也难以修改，但我还是想修改，我必须修改，这是我今日之后的生活。至于我自己，已咎由自取，愿日后不在产生bug。java程序员。——我真是越来越喜欢自黑了。哈哈，让我笑一会。



牲口TT  
关注 - 22  
粉丝 - 131

加关注

« 上一篇: [Java学习笔记 \(十五\) ——javadoc学习笔记和可能的注意细节](#)  
» 下一篇: [Java学习笔记 \(十七\) ——Java序列化](#)

posted @ 2014-04-02 11:05 牲口TT 阅读(14873) 评论(4) 编辑 收藏

评论列表		
#1楼	2018-10-30 16:03 凌海森	
66	这代码你自己 运行过? ?	支持(0) 反对(0)
#2楼	2018-10-30 18:57 凌海森	
66	我在将服务端发布到 阿里云遇到几个问题，借此地备注分享一下  00. 阿里云实例，安全组问题，需要开放端口哦。  01. 返回内网问题，需要在服务端添加: System.setProperty("java.rmi.server.hostname", "阿里云外网地址");  02. java.rmi.NoSuchObjectException: no such object in table 需要给导出的对象来一个强引用，否则被GC回收了  03. 服务器防火墙问题，一个服务注册端口，还有一个通讯端口（随机分配），所以需要 把随机端口固定，并在防火墙里打开。 UserManagerInterface userManagerI = (UserManagerInterface) UnicastRemoteObject.exportObject(userManager, 8500); // 这里就不要写0了	支持(0) 反对(0)
#3楼	2019-03-19 16:34 温暖的陌生人	
66	你这代码有错误	支持(0) 反对(0)
#4楼	2019-04-04 14:45 HelloWorld	
66	@ 温暖的陌生人 没啥问题,他说的打包成jar,其实就是 虚拟远程对象,利用client端的虚拟对象来获取远程对象的值	支持(0) 反对(0)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，访问网站首页。

- 【推荐】超50万C++/C#源码：大型实时仿真组态图形源代码
- 【推荐】IT职业生涯指南，Java程序员薪资翻3倍的秘密
- 【推荐】专业便捷的企业级代码托管服务 - Gitee 码云
- 【博客】工作996，生病ICU，程序员不加班就没前途吗？

百度智能云 | <<>> 开年采购季

云服务器**64.2元/年**  
限量开抢

立即抢购 →

- 相关博文:
- [JavaRMI](#)
  - [java学习笔记 \(十六\)](#)
  - [Java RMI 学习笔记](#)
  - [Java RMI](#)
  - [javarmi](#)

SPREAD JS

纯前端表格控件  
华为、海信、金蝶等企业都在使用>>

GrapeCity

- 最新新闻:
- 骁龙850平台加持 新款HUAWEI MateBook E来了
  - 支付宝蚂蚁森林里种树！5亿人种下1亿棵真树
  - 手机靓号网上司法拍卖：千人围观 最高拍到391万元
  - 京东物流抢食顺丰：取消底薪只是开始 买地建仓不能少
  - 直击|华为：目前超100万企业用户和开发者选择华为云
- » [更多新闻...](#)