<http://blog.csdn.net/a123demi/article/details/51179083>

版权声明：本文为博主原创文章，转载请注明原地址

目录[(?)[+]](http://blog.csdn.net/a123demi/article/details/51179083)

# 一.  绪论

[**spring**](http://lib.csdn.net/base/javaee)为各种远程访问技术提供集成工具类。Spring远程访问通过使用普通POJOs，能更容易的开发远程访问服务。目前，Spring远程访问的主要技术如下：

1.      远程调用RMI(Remote Method Invocation): 通过使用RmiProxyFactoryBean 和 RmiServiceExporter，并且，Spring支持两个传统的RMI（使用 **[Java](http://lib.csdn.net/base/java" \o "Java 知识库" \t "_blank)**.rmi.Remote接口和java.rmi.RemoteException）和通过RMI调用器实现的暴露远程调用（支持任何Java接口）。

2.      Spring的HTTP调用器(Spring’sHttp Invoker): Spring提供了一种特殊的允许通过HTTP进行Java串行化的远程调用策略，支持任意Java接口（就像RMI调用器）。相对应的支持类是 HttpInvokerProxyFactoryBean和 HttpInvokerServiceExporter。

3.      Hessian: 通过HessianProxyFactoryBean 和 HessianServiceExporter，可以使用Caucho提供的基于HTTP的轻量级二进制协议来透明地暴露服务。

4.      Burlap: Burlap是Caucho的另外一个子项目，可以作为Hessian基于XML的替代方案。Spring提供了诸如BurlapProxyFactoryBean 和 BurlapServiceExporter 的支持类。

5.      JAX RPC: Spring通过JAX-RPC为远程Web服务提供支持(J2EE1.4's web service API)。

6.      JAX-WS：Spring通过JAX-WS为远程Web服务提供支持(thesuccessor of JAX-RPC, as introduced in [**Java**](http://lib.csdn.net/base/java)EE 5 and [**java**](http://lib.csdn.net/base/java)6)。

7.      JMS：远程访问通过类JmsInvokerServiceExporter和JmsInvokerProxyFactoryBean使用JMS的底层协议实现。

本文主要实现基于RMI(远程方法调用)的远程访问技术。

# 二. 远程访问------RMI

# 1. RMI远程访问基本流程

    1). 服务端定义远程访问接口;

    2). 服务端通过RmiServiceExporter暴露服务接口

    3). 客户端定义与服务端已暴露的相同接口

    4). 客户端通过RmiProxyFactoryBean调用服务接口

## 2.  RMI---服务器

### 1).服务端定义远程访问接口

1. <span style="font-family:Times New Roman;"><span style="font-family:Microsoft YaHei;font-size:18px;">**package** com.lm.core.service;
3. **import** java.util.List;
5. **import** com.lm.core.entity.User;
7. **public** **interface** UserRmiService {
8. List<User> getUserByAcount(String name,String password);
10. **void** insert(User user);
11. }
12. </span></span>

### 2). 接口实现

1. **<span** style="font-family:Times New Roman;"**><span** style="font-family:Microsoft YaHei;font-size:18px;"**>**package com.lm.core.service.impl;
3. import java.util.ArrayList;
4. import java.util.List;
6. import org.springframework.beans.factory.annotation.Autowired;
8. import com.lm.core.entity.User;
9. import com.lm.core.mapper.UserMapper;
10. import com.lm.core.service.UserRmiService;
12. public class UserRmiServiceImpl implements UserRmiService {
14. @Autowired
15. private UserMapper userMapper;
16. @Override
17. public List**<User>** getUserByAcount(String name, String password) {
18. System.err.println("获取用户信息:" + name + password);
19. return new ArrayList**<User>**();
20. }
21. @Override
22. public void insert(User user) {
23. System.err.println("开始插入用户信息:" + user.toString());
24. }
26. }**</span></span>**

### 3). spring配置暴露接口

1. **<span** style="font-family:Times New Roman;"**><span** style="font-family:Microsoft YaHei;font-size:18px;"**>**<!-- spring rmi-->
2. **<bean** name="urseRmiService" class="com.lm.core.service.impl.UserRmiServiceImpl"**/>**
3. **<bean** name="rmiTest" class="org.springframework.remoting.rmi.RmiServiceExporter"**>**
4. **<property** name="service"ref="urseRmiService" **/>**
5. **<property** name="serviceName"value="urseRmiServiceTest" **/>**
6. **<property** name="serviceInterface"value="com.lm.core.service.UserRmiService"**/>**
7. **<property** name="registryPort"value="1021" **/>**
8. **</bean></span></span>**

### 4). 客户端定义相同暴露的接口

1. **<span** style="font-family:Times New Roman;"**><span** style="font-family:Microsoft YaHei;font-size:18px;"**>**packagecom.lm.core.service;
3. importjava.util.List;
5. importcom.lm.core.entity.User;
7. public interfaceUserRmiService {
8. List**<User>** getUserByAcount(Stringname,String password);
10. void insert(User user);
11. }**</span></span>**

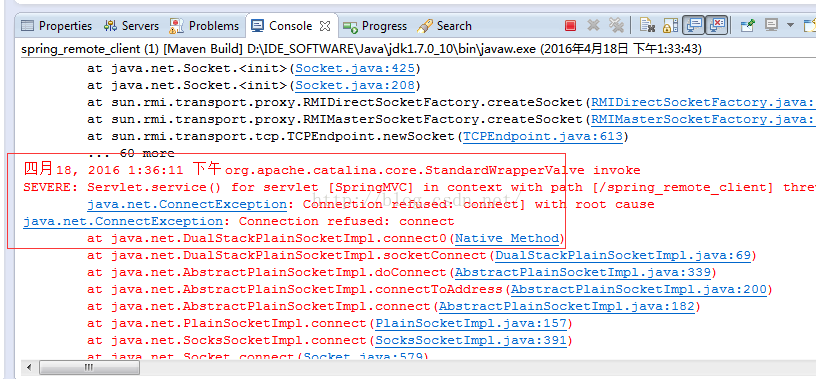
### 5). Spring配置远程访问接口信息

1. **<span** style="font-family:Times New Roman;"**><span** style="font-family:Microsoft YaHei;font-size:18px;"**>**  <!-- spring rmi -->
2. **<bean** id="clientRmiTest" class="org.springframework.remoting.rmi.RmiProxyFactoryBean"**>**
3. **<property** name="serviceUrl"value="rmi://127.0.0.1:1021/urseRmiServiceTest"**/>**
4. **<property** name="serviceInterface" value="com.lm.core.service.UserRmiService"**/>**
5. **<property** name="lookupStubOnStartup" value="false"**/>**
6. **<property** name="refreshStubOnConnectFailure" value="true"**></property>**
7. **</bean></span></span>**

lookupStubOnStartup：预查找远程对象 默认为true

refreshStubOnConnectFailure：是否刷新远程调用缓存的stub

注意：如果上述两项不配置，当服务器未开启，客户端无法打包，会有拒绝连接异常。

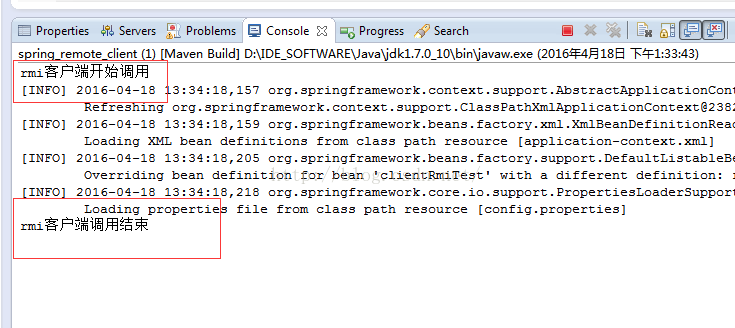
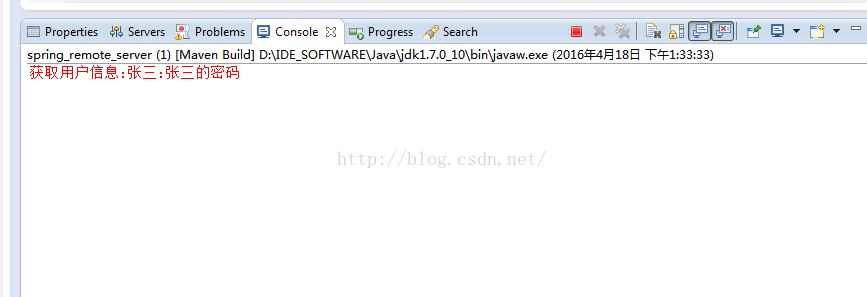


### 6). 客户端调用服务器接口

1. **<span** style="font-family:Times New Roman;"**><span** style="font-family:Microsoft YaHei;font-size:18px;"**>**package com.lm.web.controller;
3. import org.springframework.context.ApplicationContext;
4. importorg.springframework.context.support.ClassPathXmlApplicationContext;
5. import org.springframework.stereotype.Controller;
6. import org.springframework.web.bind.annotation.RequestMapping;
7. import org.springframework.web.bind.annotation.ResponseBody;
9. import com.lm.core.service.UserRmiService;
10. import com.lm.web.vo.BaseMapVo;
12. @Controller
13. @RequestMapping(value = "/user")
14. public class UserController {

17. @SuppressWarnings("resource")
18. @RequestMapping(value = "/rmiTest")
19. @ResponseBody
20. public BaseMapVo rmiTest(String name, String password) {
21. BaseMapVo vo = new BaseMapVo();
22. System.out.println("rmi客户端开始调用");
23. ApplicationContext ctx = new ClassPathXmlApplicationContext(
24. "classpath:application-context.xml");
25. UserRmiService rmi = (UserRmiService) ctx.getBean("clientRmiTest");
26. rmi.getUserByAcount("张三", ":张三的密码");
27. System.out.println("rmi客户端调用结束");
28. vo.setRslt("sucess");
29. return vo;
30. }
31. }**</span></span>**

### 7).输出和结果



上述就为本文所有内容，

代码下载<http://download.csdn.net/detail/a123demi/9494528>

参考文献：<http://docs.spring.io/spring/docs/2.5.x/reference/remoting.html>