<http://blog.csdn.net/shirdrn/article/details/6359254>

版权声明：本文为博主原创文章，未经博主允许不得转载。

使用[**spring**](http://lib.csdn.net/base/javaee)对RMI的支持，可以非常容易地构建你的分布式应用。在服务端，可以通过Spring的org.springframework.remoting.rmi.RmiServiceExporter可以暴露你的服务；在客户端，通过org.springframework.remoting.rmi.RmiProxyFactoryBean可以使用服务端暴露的服务，非常方便。这种C/S模型的访问方式，可以屏蔽掉RMI本身的复杂性，如服务端Skeleton和客户端Stub等的处理细节，这些对于服务开发和服务使用的人员来说，都是透明的，无需过度关注，而集中精力开发你的商业逻辑。

下面通过一个例子，说明如何通过Spring集成RMI。

**服务端发布服务**

我们定义了服务接口，服务端实现该服务接口来完成其复杂的逻辑，客户端可以通过该接口调用服务端暴露的服务，如下所示：

**[java]** [view plain](http://blog.csdn.net/shirdrn/article/details/6359254) [copy](http://blog.csdn.net/shirdrn/article/details/6359254)

1. **package** org.shirdrn.spring.remote.rmi;
3. **public** **interface** AccountService {
4. **int** queryBalance(String mobileNo);
5. String shoopingPayment(String mobileNo, **byte** protocol);
6. }

服务实现，示例如下所示：

**[java]** [view plain](http://blog.csdn.net/shirdrn/article/details/6359254) [copy](http://blog.csdn.net/shirdrn/article/details/6359254)

1. **package** org.shirdrn.spring.remote.rmi;
3. **import** org.apache.log4j.Logger;
5. **public** **class** MobileAccountServiceImpl **implements** AccountService {
7. **private** **static** **final** Logger LOG = Logger.getLogger(MobileAccountServiceImpl.**class**);
8. **public** **int** queryBalance(String mobileNo) {
9. **if** (mobileNo != **null**)
10. **return** 100;
11. **return** 0;
12. }
14. **public** String shoopingPayment(String mobileNo, **byte** protocol) {
15. StringBuffer sb = **new** StringBuffer().append("Your mobile number is /"").append(
16. mobileNo).append("/", protocol type is /"").append(protocol)
17. .append("/".");
18. LOG.info("Message is: " + sb.toString());
19. **return** sb.toString();
20. }
21. }

服务端发布服务，供客户端进行（远程方法）调用，Spring配置server.xml如下所示：

**[xhtml]** [view plain](http://blog.csdn.net/shirdrn/article/details/6359254) [copy](http://blog.csdn.net/shirdrn/article/details/6359254)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<beans** xmlns="http://www.springframework.org/schema/beans"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:p="http://www.springframework.org/schema/p"
4. xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.5.xsd"**>**
6. **<bean** id="serviceExporter" class="org.springframework.remoting.rmi.RmiServiceExporter"**>**
7. **<property** name="serviceName" value="MobileAccountService" **/>**
8. **<property** name="service" ref="accountService" **/>**
9. **<property** name="serviceInterface"
10. value="org.shirdrn.spring.remote.rmi.AccountService" **/>**
11. **<property** name="registryPort" value="8080" **/>**
12. **<property** name="servicePort" value="8088" **/>**
13. **</bean>**
15. **<bean** id="accountService" class="org.shirdrn.spring.remote.rmi.MobileAccountServiceImpl" **/>**
17. **</beans>**

上面配置，指定了暴露的服务的名称，通过serviceName属性注入到RmiServiceExporter中，服务名称为MobileAccountService，客户端通过该服务名称就能够进行调用。

下面启动服务端，发布服务，如下所示：

**[java]** [view plain](http://blog.csdn.net/shirdrn/article/details/6359254) [copy](http://blog.csdn.net/shirdrn/article/details/6359254)

1. **package** org.shirdrn.spring.remote.rmi;
3. **import** org.springframework.context.support.ClassPathXmlApplicationContext;
5. **public** **class** RmiServer {
7. **public** **static** **void** main(String[] args) **throws** InterruptedException {
8. **new** ClassPathXmlApplicationContext("org/shirdrn/spring/remote/rmi/server.xml");
10. Object lock = **new** Object();
11. **synchronized** (lock) {
12. lock.wait();
13. }
14. }
15. }

**客户端调用服务**

客户端配置client.xml如下所示：

**[xhtml]** [view plain](http://blog.csdn.net/shirdrn/article/details/6359254) [copy](http://blog.csdn.net/shirdrn/article/details/6359254)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<beans** xmlns="http://www.springframework.org/schema/beans"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:p="http://www.springframework.org/schema/p"
4. xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.5.xsd"**>**
6. **<bean** id="mobileAccountService" class="org.springframework.remoting.rmi.RmiProxyFactoryBean"**>**
7. **<property** name="serviceUrl" value="rmi://192.168.1.103:8080/MobileAccountService" **/>**
8. **<property** name="serviceInterface"
9. value="org.shirdrn.spring.remote.rmi.AccountService" **/>**
10. **</bean>**
12. **</beans>**

配置中，将一个serviceUrl和serviceInterface注入给RmiProxyFactoryBean，即可进行远程方法调用。调用示例如下所示：

**[java]** [view plain](http://blog.csdn.net/shirdrn/article/details/6359254) [copy](http://blog.csdn.net/shirdrn/article/details/6359254)

1. **package** org.shirdrn.spring.remote.rmi;
3. **import** org.apache.log4j.Logger;
4. **import** org.springframework.context.ApplicationContext;
5. **import** org.springframework.context.support.ClassPathXmlApplicationContext;
7. **public** **class** RmiClient {
9. **private** **static** **final** Logger LOG = Logger.getLogger(RmiClient.**class**);
11. **public** **static** **void** main(String[] args) {
12. ApplicationContext ctx = **new** ClassPathXmlApplicationContext(
13. "org/shirdrn/spring/remote/rmi/client.xml");
14. AccountService accountService = (AccountService) ctx
15. .getBean("mobileAccountService");
16. String result = accountService.shoopingPayment("13800138000", (**byte**) 5);
17. LOG.info(result);
18. }
20. }

可见，实现远程访问变得非常容易。