[Spring HttpInvoker远程调用的例子](http://blog.csdn.net/liuhui_306/article/details/17284357)

2013-12-12 16:49 442人阅读 [评论](http://blog.csdn.net/liuhui_306/article/details/17284357#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/liuhui_306/article/details/17284357#report)

 分类：

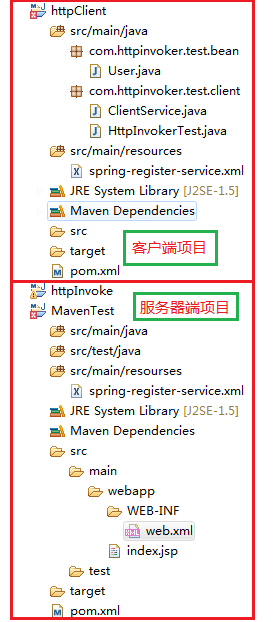


Spring

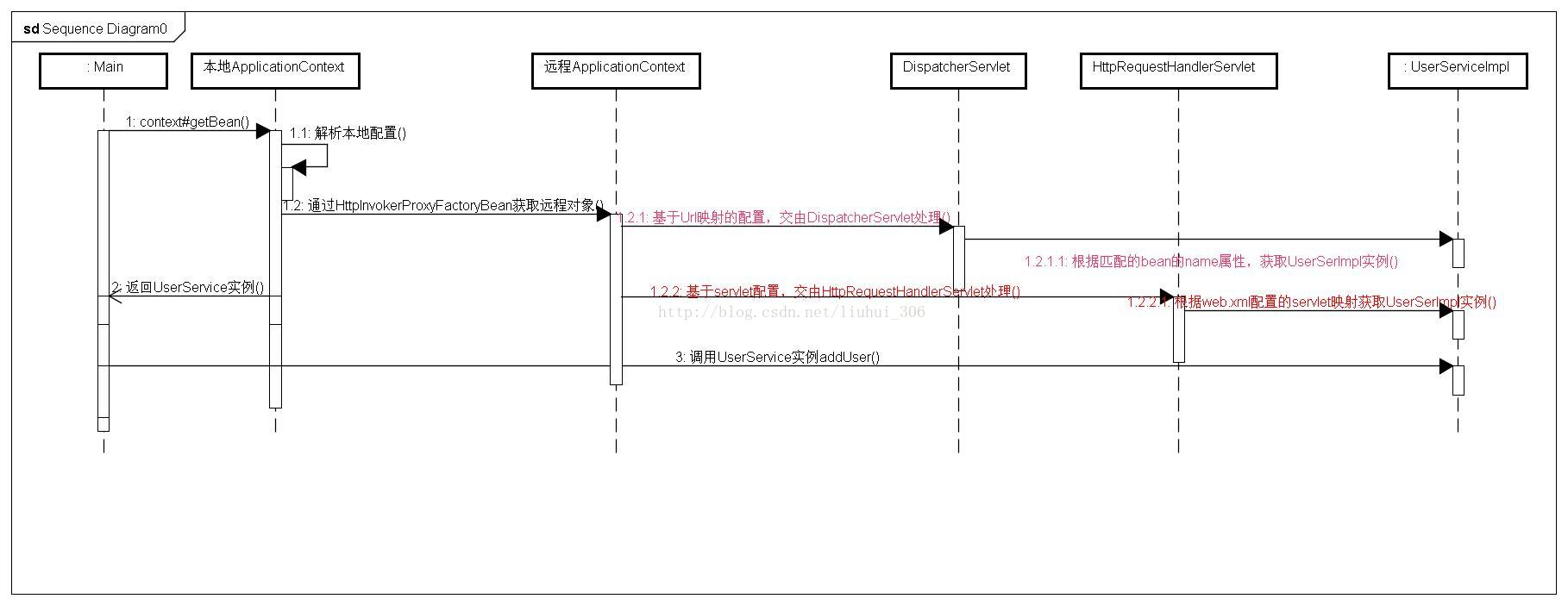
版权声明：本文为博主原创文章，未经博主允许不得转载。

HttpInvoker的实现原理和java的RMI实现原理相同，都是基于Http协议，将java对象序列化以后以流的方式在客户端和服务器端进行传输。

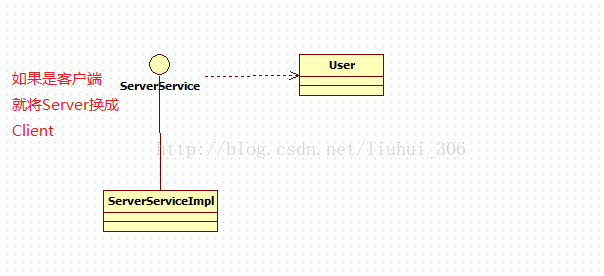
例子的项目结构图：



执行流程图如下：



类图如下：



涉及到的类及接口：

客户端：

**[html]** [view plain](http://blog.csdn.net/liuhui_306/article/details/17284357) [copy](http://blog.csdn.net/liuhui_306/article/details/17284357)



1. 接口
2. package com.httpinvoker.test.client;
4. import com.httpinvoker.test.bean.User;
6. /\*\*
7. \* 远程调用的客户端
8. \*/
9. public interface ClientService {
11. /\*\*
12. \* 添加一个账户
13. \* @param account
14. \*/
15. void addAccount(User account);
17. /\*\*
18. \* 查询一个账户
19. \* @return
20. \*/
21. User findAccount();
23. }
25. 测试类
26. package com.httpinvoker.test.client;
28. import org.springframework.context.support.ClassPathXmlApplicationContext;
30. import com.httpinvoker.test.bean.User;
32. public class HttpInvokerTest {
33. public static void main(String[] args) {
34. ClassPathXmlApplicationContext ctx = new ClassPathXmlApplicationContext(
35. new String[] {"spring-register-service.xml" });
37. ClientService clientService = (ClientService)ctx.getBean("accountService");
39. User account = new User("张三", "zhangsan", "男");
41. clientService.addAccount(account);
43. User account2 = clientService.findAccount();
44. System.out.println("客户端查询到的用户的名字为 : " + account2.getName());
45. }
46. }

服务器端：

**[java]** [view plain](http://blog.csdn.net/liuhui_306/article/details/17284357) [copy](http://blog.csdn.net/liuhui_306/article/details/17284357)



1. 接口
2. **package** com.httpinvoker.test.server;
4. **import** com.httpinvoker.test.bean.User;
6. /\*\*
7. \* 远程调用的客户端
8. \*/
9. **public** **interface** ServerService {
11. /\*\*
12. \* 添加一个账户
13. \* @param account
14. \*/
15. **void** addAccount(User account);
17. /\*\*
18. \* 查询一个账户
19. \* @return
20. \*/
21. User findAccount();
23. }
25. 实现
26. **package** com.httpinvoker.test.server;
28. **import** com.httpinvoker.test.bean.User;
30. **public** **class** ServerServiceImpl **implements** ServerService {
32. **public** **void** addAccount(User account) {
33. System.out.println("用户 : " + account.getName() + " 添加成功!");
34. }
36. **public** User findAccount() {
37. User account = **new** User("张三", "zhangsan", "男");
38. System.out.println("用户 : " + account.getName() + " 查找成功!");
39. **return** account;
40. }
41. }

公共的User对象

**[java]** [view plain](http://blog.csdn.net/liuhui_306/article/details/17284357) [copy](http://blog.csdn.net/liuhui_306/article/details/17284357)



1. **package** com.httpinvoker.test.bean;
3. **import** java.io.Serializable;
5. **public** **class** User **implements** Serializable{
6. **private** **static** **final** **long** serialVersionUID = 1L;
8. **private** String name;
9. **private** String passwd;
10. **private** String sex;
12. **public** User() {
13. }
15. **public** User(String name, String passwd, String sex) {
16. **this**.name = name;
17. **this**.passwd = passwd;
18. **this**.sex = sex;
19. }
21. **public** String getName() {
22. **return** name;
23. }
25. **public** **void** setName(String name) {
26. **this**.name = name;
27. }
29. **public** String getPasswd() {
30. **return** passwd;
31. }
33. **public** **void** setPasswd(String passwd) {
34. **this**.passwd = passwd;
35. }
37. **public** String getSex() {
38. **return** sex;
39. }
41. **public** **void** setSex(String sex) {
42. **this**.sex = sex;
43. }
44. }

客户端spring的配置：

**[html]** [view plain](http://blog.csdn.net/liuhui_306/article/details/17284357) [copy](http://blog.csdn.net/liuhui_306/article/details/17284357)



1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<beans** xmlns="http://www.springframework.org/schema/beans" xmlns:p="http://www.springframework.org/schema/p" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:aop="http://www.springframework.org/schema/aop" xmlns:tx="http://www.springframework.org/schema/tx" xmlns:context="http://www.springframework.org/schema/context" xsi:schemaLocation="
3. http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
4. http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.0.xsd
5. http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.0.xsd
6. http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd"**>**
7. **<bean** id="accountService" class="org.springframework.remoting.httpinvoker.HttpInvokerProxyFactoryBean"**>**
8. **<property** name="serviceUrl"**>**
9. **<value>**http://localhost:8080/MavenTest/userHttpInvokerService**</value>**
10. **</property>**
11. **<property** name="serviceInterface"**>**
12. **<value>**com.httpinvoker.test.client.ClientService**</value>**
13. **</property>**
14. **</bean>**
16. **</beans>**

服务器端 spring的配置

**[html]** [view plain](http://blog.csdn.net/liuhui_306/article/details/17284357) [copy](http://blog.csdn.net/liuhui_306/article/details/17284357)



1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<beans** xmlns="http://www.springframework.org/schema/beans" xmlns:p="http://www.springframework.org/schema/p" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:aop="http://www.springframework.org/schema/aop" xmlns:tx="http://www.springframework.org/schema/tx" xmlns:context="http://www.springframework.org/schema/context" xsi:schemaLocation="
3. http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
4. http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.0.xsd
5. http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.0.xsd
6. http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd"**>**
8. **<bean** id="accountService" class="com.httpinvoker.test.server.ServerServiceImpl"**>**
9. **</bean>**
11. **<bean** name="userHttpInvokerService" class="org.springframework.remoting.httpinvoker.HttpInvokerServiceExporter"**>**
12. **<property** name="service" ref="accountService"**/>**
13. **<property** name="serviceInterface" value="com.httpinvoker.test.server.ServerService"**/>**
14. **</bean>**
16. **</beans>**

web.xml的配置:

<servlet>  
     <servlet-name>userHttpInvokerService</servlet-name>  
     <servlet-class>org.springframework.web.context.support.HttpRequestHandlerServlet</servlet-class>  
 </servlet>  
   
 <servlet-mapping>  
     <servlet-name>userHttpInvokerService</servlet-name>  
     <url-pattern>/userHttpInvokerService</url-pattern>  
 </servlet-mapping>

注意：<http://localhost:8080/MavenTest/userHttpInvokerService>中的/userHttpInvokerService是服务器端的servlet所匹配的servlet-name所对应的bean的name属性值。