Dubbo 源码分析 ---【2】RPC 实现原理

作者：zsr251
原文链接：https://ld246.com/article/1541581813424
来源网站：链滴

RPC原理浅析

实现简单RPC的基础模块：

1. 处理网络连接通讯的模块：负责连接建立、管理和消息的传输。
2. 编解码的模块：将网络通讯传输的字节码反序列化为使用的对象，将对象序列化为字节码
3. 服务端：暴露要开放的服务接口
4. 客户端：调用服务接口的一个代理实现，这个代理实现负责收集数据、编码并传输给服务器然后等结果返回

简单实现

Dubbo作者梁飞的简单实现 RPC框架几行代码就够了]

框架代码

```
/*
 * Copyright 2011 Alibaba.com All right reserved. This software is the
 * confidential and proprietary information of Alibaba.com ("Confidential
 * Information"). You shall not disclose such Confidential Information and shall
 * use it only in accordance with the terms of the license agreement you entered
 * into with Alibaba.com.
 */
package com.alibaba.study.rpc.framework;

import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.lang.reflect.InvocationHandler;
import java.lang.reflect.Method;
import java.lang.reflect.Proxy;
import java.net.ServerSocket;
import java.net.Socket;

/**
 * RpcFramework
 *
 * @author william.liangf
 */
public class RpcFramework {

    /**
     * 暴露服务
     *
     * @param service 服务实现
     * @param port 服务端口
     * @throws Exception
     */
    public static void export(final Object service, int port) throws Exception {
        if (service == null)
            throw new IllegalArgumentException("service instance == null");
```

```
        if (port <= 0 || port > 65535)
            throw new IllegalArgumentException("Invalid port " + port);
        System.out.println("Export service " + service.getClass().getName() + " on port " + port);
        ServerSocket server = new ServerSocket(port);
        for(;;) {
            try {
                final Socket socket = server.accept();
                new Thread(new Runnable() {
                    @Override
                    public void run() {
                        try {
                            try {
                                ObjectInputStream input = new ObjectInputStream(socket.getInputStrea
());
                                try {
                                    String methodName = input.readUTF();
                                    Class<?>[] parameterTypes = (Class<?>[])input.readObject();
                                    Object[] arguments = (Object[])input.readObject();
                                    ObjectOutputStream output = new ObjectOutputStream(socket.getOu
putStream());
                                    try {
                                        Method method = service.getClass().getMethod(methodName, para
eterTypes);
                                        Object result = method.invoke(service, arguments);
                                        output.writeObject(result);
                                    } catch (Throwable t) {
                                        output.writeObject(t);
                                    } finally {
                                        output.close();
                                    }
                                } finally {
                                    input.close();
                                }
                            } finally {
                                socket.close();
                            }
                        } catch (Exception e) {
                            e.printStackTrace();
                        }
                    }
                }).start();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }

    /**
     * 引用服务
     *
     * @param <T> 接口泛型
     * @param interfaceClass 接口类型
     * @param host 服务器主机名
     * @param port 服务器端口
```

```
     * @return 远程服务
     * @throws Exception
     */
    @SuppressWarnings("unchecked")
    public static <T> T refer(final Class<T> interfaceClass, final String host, final int port) throw
 Exception {
        if (interfaceClass == null)
            throw new IllegalArgumentException("Interface class == null");
        if (! interfaceClass.isInterface())
            throw new IllegalArgumentException("The " + interfaceClass.getName() + " must be in
erface class!");
        if (host == null || host.length() == 0)
            throw new IllegalArgumentException("Host == null!");
        if (port <= 0 || port > 65535)
            throw new IllegalArgumentException("Invalid port " + port);
        System.out.println("Get remote service " + interfaceClass.getName() + " from server " +
ost + ":" + port);
        return (T) Proxy.newProxyInstance(interfaceClass.getClassLoader(), new Class<?>[] {interf
ceClass}, new InvocationHandler() {
            public Object invoke(Object proxy, Method method, Object[] arguments) throws Thro
able {
                Socket socket = new Socket(host, port);
                try {
                    ObjectOutputStream output = new ObjectOutputStream(socket.getOutputStrea
());
                    try {
                        output.writeUTF(method.getName());
                        output.writeObject(method.getParameterTypes());
                        output.writeObject(arguments);
                        ObjectInputStream input = new ObjectInputStream(socket.getInputStream());
                        try {
                            Object result = input.readObject();
                            if (result instanceof Throwable) {
                                throw (Throwable) result;
                            }
                            return result;
                        } finally {
                            input.close();
                        }
                    } finally {
                        output.close();
                    }
                } finally {
                    socket.close();
                }
            }
        });
    }

}
```

定义服务接口

```
/*
 * Copyright 2011 Alibaba.com All right reserved. This software is the
 * confidential and proprietary information of Alibaba.com ("Confidential
 * Information"). You shall not disclose such Confidential Information and shall
 * use it only in accordance with the terms of the license agreement you entered
 * into with Alibaba.com.
 */
package com.alibaba.study.rpc.test;

/**
 * HelloService
 *
 * @author william.liangf
 */
public interface HelloService {

    String hello(String name);

}
```

实现服务

```
/*
 * Copyright 2011 Alibaba.com All right reserved. This software is the
 * confidential and proprietary information of Alibaba.com ("Confidential
 * Information"). You shall not disclose such Confidential Information and shall
 * use it only in accordance with the terms of the license agreement you entered
 * into with Alibaba.com.
 */
package com.alibaba.study.rpc.test;

/**
 * HelloServiceImpl
 *
 * @author william.liangf
 */
public class HelloServiceImpl implements HelloService {

    public String hello(String name) {
        return "Hello " + name;
    }

}
```

暴露服务

```
/*
 * Copyright 2011 Alibaba.com All right reserved. This software is the
 * confidential and proprietary information of Alibaba.com ("Confidential
 * Information"). You shall not disclose such Confidential Information and shall
 * use it only in accordance with the terms of the license agreement you entered
```

```
 * into with Alibaba.com.
 */
package com.alibaba.study.rpc.test;

import com.alibaba.study.rpc.framework.RpcFramework;

/**
 * RpcProvider
 *
 * @author william.liangf
 */
public class RpcProvider {

    public static void main(String[] args) throws Exception {
        HelloService service = new HelloServiceImpl();
        RpcFramework.export(service, 1234);
    }

}
```

引用服务

```
/*
 * Copyright 2011 Alibaba.com All right reserved. This software is the
 * confidential and proprietary information of Alibaba.com ("Confidential
 * Information"). You shall not disclose such Confidential Information and shall
 * use it only in accordance with the terms of the license agreement you entered
 * into with Alibaba.com.
 */
package com.alibaba.study.rpc.test;

import com.alibaba.study.rpc.framework.RpcFramework;

/**
 * RpcConsumer
 *
 * @author william.liangf
 */
public class RpcConsumer {

    public static void main(String[] args) throws Exception {
        HelloService service = RpcFramework.refer(HelloService.class, "127.0.0.1", 1234);
        for (int i = 0; i < Integer.MAX_VALUE; i ++) {
            String hello = service.hello("World" + i);
            System.out.println(hello);
            Thread.sleep(1000);
        }
    }

}
```

总结

1. 使用阻塞的socket IO流来进行server和client的通信
2. 端对端的，用端口号来直接进行通信
3. 参数的序列化也是使用的最简单的objectStream
4. 方法的远程调用使用的是jdk的动态代理