1 什么是 RMI

- 我们说的 RMI 一般都是指 Java RMI
- Remote Method Invocation：远程方法调用
- 使一个 Java 虚拟机中的对象可以调用另一个 Java 虚拟机上的对象的方法(就像调用本地对象方法一样)
- 两个虚拟机可以运行在同一机器中的不同进程中，也可以运行在网络上的其它机器中

2 RMI 的适用场景

- 当需要远程调用另一个虚拟机中的对象方法时（必须两端都是 Java 应用）

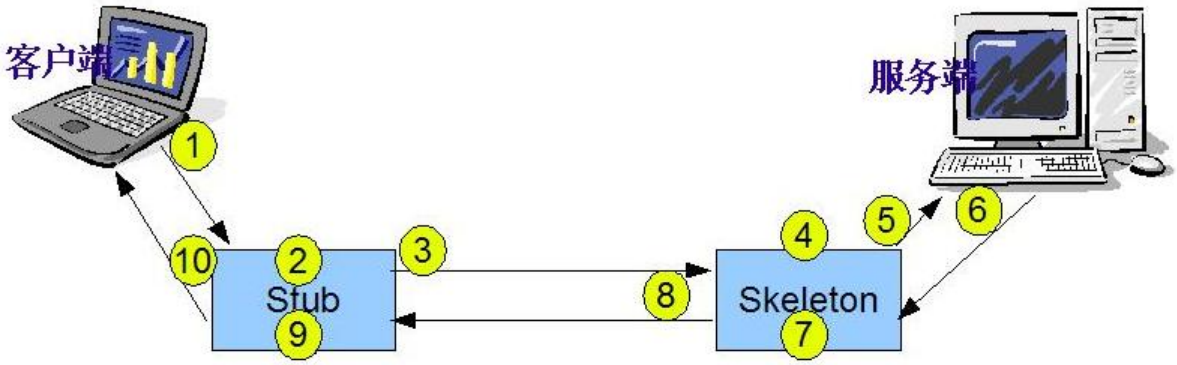
3 为什么要用 RMI

- RMI 封装了所有底层通信细节，使得开发人员只需专注于开发业务功能

4 RMI 核心概念

- RMI框架采用**代理**来负责客户与远程对象之间通过Socket进行通信的细节
- RMI框架为远程对象分别生成了客户端代理和服务器端代理
- Stub(存根)：客户端代理类，
- Skeleton(骨架)：服务器端代理类
- stub 和 skeleton 的讲解，自己实现一个stub和skeleton程序

4.1 RMI 实现过程



整个远程方法调用的过程

1、客户端由于需要调用远程主机上的方法
2、经过 Stub 的代理，编码这个请求方便网络传输
3、网络上传输
4、收到 Stub 请求命令，转换成服务端可以识别的请求
5、发送请求到服务端
6、经过服务端处理后结果，发送到 Skeleton 编码
7、 Skeleton 把结果编码，方便在网络上传输
8、网络上传输
9、解码结果
10、客户端获得远程方法调用的结果

4.2 Skeleton及Stub获取

- 通过继承 UnicastRemoteObject 类，即可隐式的获取 Skeleton及Stub
- 通过 rmic 工具生成：rmic xx.class，将生成 xx_Stub.class，xx_Skel.class（不推荐）,UnicastRemoteObject 首先判断是否有静态的存根类，如果没有才动态创建 Stub

5 示例代码

```
//Remote 接口是一个标志接口，标志这个接口中所包含的方法从非本地虚拟机上调用
public interface StudentService extends Remote {
    //由于方法会通过网络调用，所以必须抛出 远程异常来说明该方法有风险
    Student getStudentList() throws RemoteException;
}
public class StudentServiceImpl extends UnicastRemoteObject implements StudentService {
    @Override
    public Student getStudentList() {
        return new Student(1,"wk");
    }
    // 该构造期必须存在，因为集继承了UnicastRemoteObject类，其构造器要抛出RemoteException
    public StudentServiceImpl() throws RemoteException {
        super();
    }
}
public class StudentServer {
    public static void main(String[] args) throws Exception{
        //创建一个远程对象，实质上隐含了是生成stub和skeleton,并返回stub代理引用
        StudentService stu = new StudentServiceImpl();
```

```
        StudentService stu0 = new StudentServiceImpl0();
        //在本地主机上创建和导出注册表实例，并在指定的端口上接受请求
        //Java默认端口是1099，缺少注册表创建，则无法绑定对象到远程注册表上
        LocateRegistry.createRegistry(1099);

        //把远程对象注册到RMI注册服务器上----------

        //JDK 1.3及以下
        //Naming.rebind("stu",stu);//绑定的URL标准格式为：rmi://host:port/name(其中
rmi://host:port都可以省略)

        //JDK 1.3 以上
        Context namingContext = new InitialContext();//初始化命名上下文
        //把远程对象注册到RMI注册服务器上，注意：rmi 协议名不能省略，host:port可以省略
        namingContext.rebind("rmi:stu",stu);
        namingContext.rebind("rmi:stu0",stu0);
        System.out.println("服务器已启动！");

    }
}
public class StudentClient {
    public static void main(String[] args) throws Exception{
            //两种方式都可以获得远程对象
            StudentService stu = (StudentService) Naming.lookup("stu");
            Context namingContext = new InitialContext();
            StudentService stu0 = (StudentService) namingContext.lookup("rmi:stu");
            System.out.println(stu0.getStudentList().toString());
            System.out.println(stu.getStudentList().toString());

        StudentService stu01 = (StudentService) namingContext.lookup("rmi:stu0");
        System.out.println(stu01.getStudentList().toString());

    }
}
```

参考

github 源码地址 stub 和 skeleton 的讲解，自己实现一个stub和skeleton程序 RMI 报错：javax.naming.NoInitialContextException: specify class name environment system property Java RMI远程方法调用详解 java项目中rmi远程调用实例 RMI底层实现原理 从懵逼到恍然大悟之Java中RMI的使用 Head First 设计模式 —— 代理模式