1 项目简介

- 测试Java RMI 远程调用功能
- 通过 JDK 1.3 及以下版本的方法（Naming.rebind()）来注册远程对象可以正常运行
- 通过 JDK 1.3 以上版本的方法 (new InitialCOntext()j.rebind()) 来注册远程对象报错

2 报错代码

```
public class StudentServer {
    public static void main(String[] args) throws Exception{
        StudentService stu = null;
        //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
        LocateRegistry.createRegistry(1099);
        stu = new StudentServiceImpl();
        //JDK 1.3及以下
        //Naming.rebind("stu",stu);//这里可以正常注册

        //JDK 1.3 以上
        Context namingContext = new InitialContext();
        namingContext.rebind("stu",stu);//在这里报错！
        System.out.println("服务器已启动！");
    }
}
```

3 报错代码跟踪

- 从 rebind() 方法开始调试

```
public void rebind(String name, Object obj) throws NamingException {
    getURLOrDefaultInitCtx(name).rebind(name, obj);
}
protected Context getURLOrDefaultInitCtx(String name)
    throws NamingException {
    //如果已经有默认的上下文了，就直接返回默认上下文
    if (NamingManager.hasInitialContextFactoryBuilder()) {
        return getDefaultInitCtx();
    }
    //通过 name 得到上下文的主题（这里是通过分割 name 字符串中的 :和/ 符号，来得到主题的！！）
    String scheme = getURLScheme(name);
```

```
    if (scheme != null) {
        //如果有主题，则通过主题来创建上下文
        Context ctx = NamingManager.getURLContext(scheme, myProps);
        if (ctx != null) {
            return ctx;
        }
    }
    //否则返回默认主题
    return getDefaultInitCtx();//因为 scheme 为null，错误代码运行到这里！
}
```

- 获取默认上下文出错

```
protected Context getDefaultInitCtx() throws NamingException{
    if (!gotDefault) {//这里 gotDefault 为 false，因为在 new InitialContext()，没有调用这个
方法，所以 gotDefault 是默认的false
        defaultInitCtx = NamingManager.getInitialContext(myProps);
        gotDefault = true;
    }
    if (defaultInitCtx == null)
        throw new NoInitialContextException();

    return defaultInitCtx;
}
protected Context getDefaultInitCtx() throws NamingException{
    if (!gotDefault) {
        defaultInitCtx = NamingManager.getInitialContext(myProps);//进入这里出错
        gotDefault = true;
    }
    if (defaultInitCtx == null)
        throw new NoInitialContextException();
    return defaultInitCtx;
}
public static Context getInitialContext(Hashtable<?,?> env)
    throws NamingException {
    InitialContextFactory factory;
    InitialContextFactoryBuilder builder = getInitialContextFactoryBuilder();//这里
builder为null
    if (builder == null) {
        // 没有相关工厂被创建，获取环境变量中的初始化上下文工厂的 类名(className)
        // Get initial context factory class name
        String className = env != null ?
            (String)env.get(Context.INITIAL_CONTEXT_FACTORY) : null;
        // 因为没有找到对应的工厂类名，所以报错！！
        if (className == null) {
            NoInitialContextException ne = new NoInitialContextException(
                "Need to specify class name in environment or system " +
                "property, or as an applet parameter, or in an " +
                "application resource file:  " +
                Context.INITIAL_CONTEXT_FACTORY);
            throw ne;
        }
        //代码略
```

```
}
}
```

4 报错原因分析

- 从JDK1.3以上版本开始，RMI的命名服务API被整合到JNDI（Java Naming and Directory Interface，Java名字与目录接口）中
- 通过 JNDI 来注册远程，必须要在 name 中指定主题为 rmi，即 name = "rmi:..."，这样 JNDI 才可以通过获取到的 rmi 主题，来创建相应的上下文。正确代码如下：

```
    Context namingContext = new InitialContext();
    namingContext.rebind("rmi:stu",stu);//name 的完整形式为： "rmi:ip/port/name"，默认为localhost/1099
```

- 因为 JDK 1.2 之前调用的是 java.rmi.Naming 类，这里就已经明确了主题是 rmi，所以正常运行

5 其它错误原因

- 不能在main方法测试 JDNI 的tomcat 连接数据源

参考

github 源码地址 Java RMI远程方法调用详解 java项目中rmi远程调用实例 Java RMI 远程方法调用