

一、RMI基础

1.1、什么是RMI

RMI (Remote Method Invocation, 远程方法调用) 是Java的一组拥护开发分布式应用程序的API。RMI使用Java语言接口定义了远程对象, 它集合了Java序列化和Java远程方法协议(Java Remote Method Protocol)。

简单地说, 原先的程序仅能在同一操作系统的方法调用, 通过RMI可以变成在不同操作系统之间对程序中方法的调用。

RMI依赖的通信协议是JRMP。JRMP: Java远程方法协议 (Java Remote Method Protocol, JRMP) 是特定于Java技术的、用于查找和引用远程对象的协议。

RMI对象是通过序列化方式进行传输的。

1.2、RMI中的三个角色

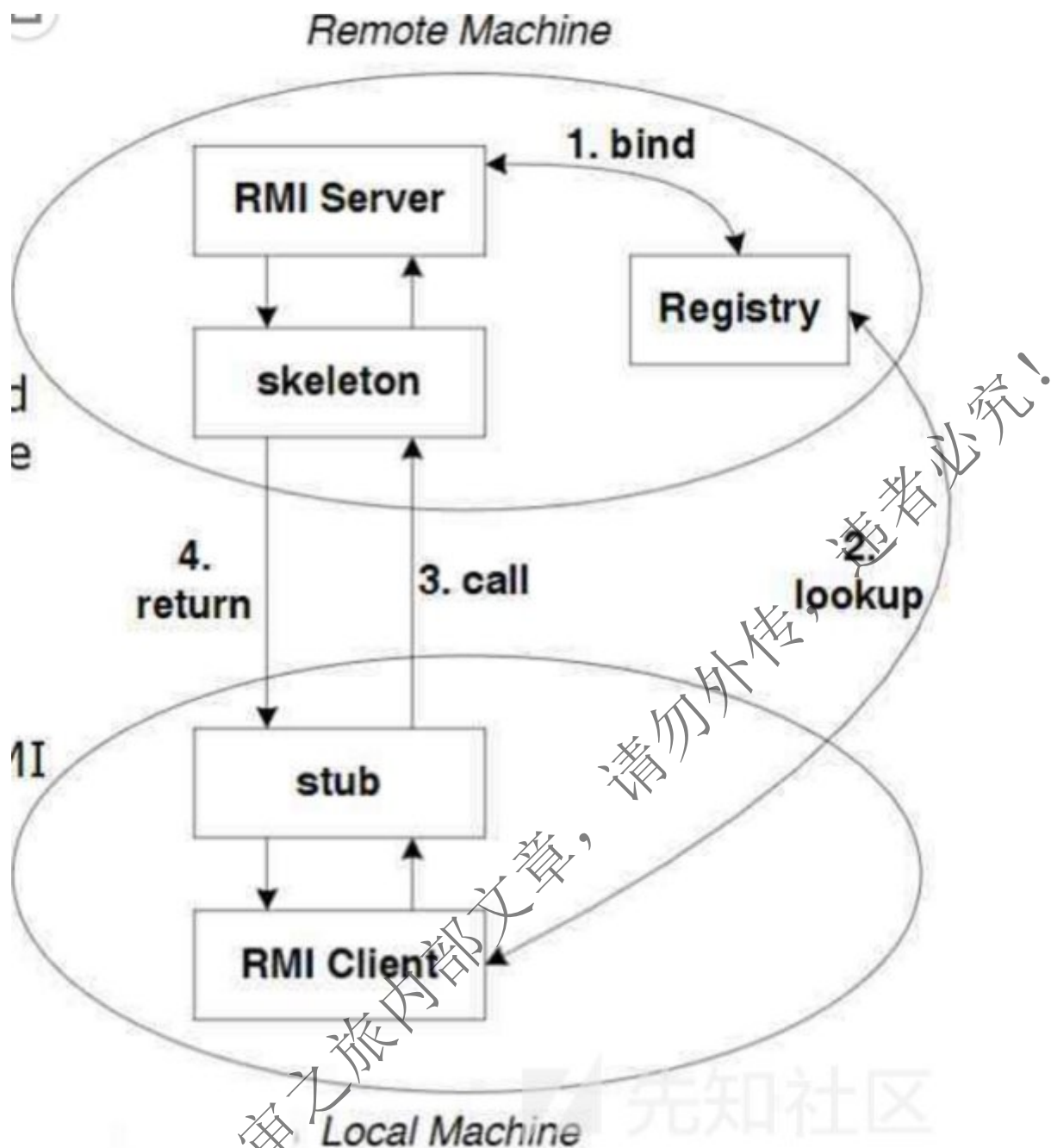
RMI中涉及到三个角色, 它们分别为服务端 (Server), 注册中心 (Registry) 和客户端 (Client), 下面是他们的作用。

服务端 (Server): 负责将远程对象绑定至注册中心。

注册中心 (Registry): 服务端会将远程对象绑定至此。客户端会向注册中心查询绑定的远程对象。

客户端 (Client): 与注册中心和服务端交互。

插入一张来自互联网的图片, 直观展示了他们的关系。



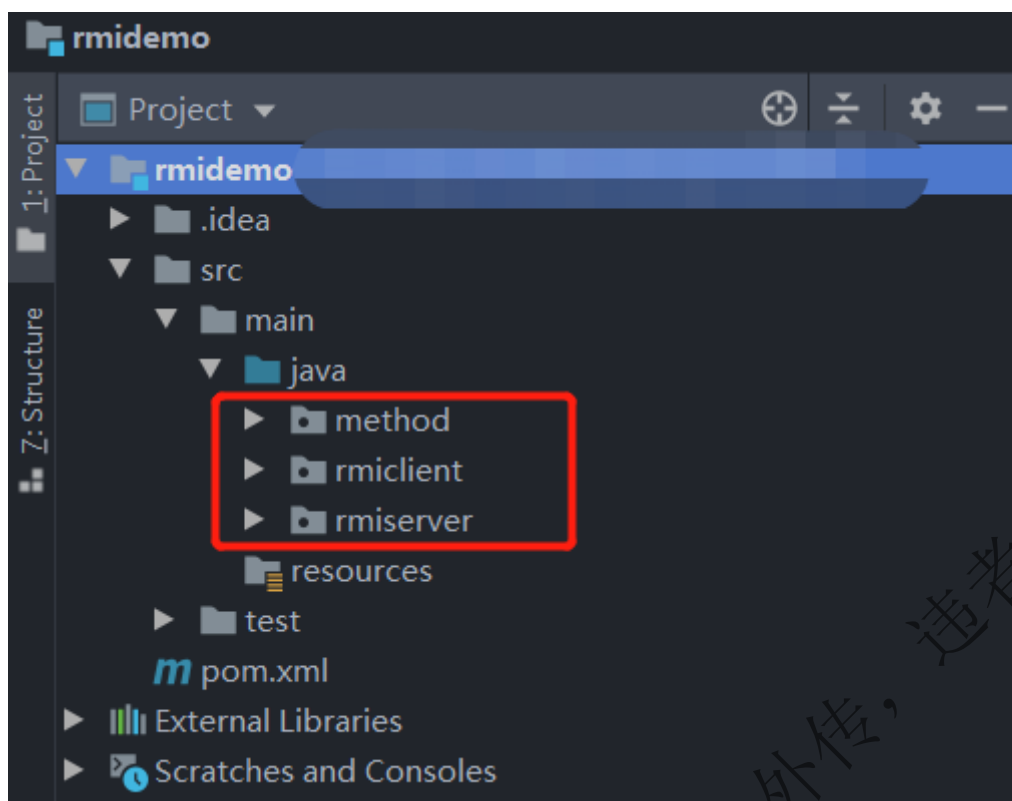
存根/桩(Stub): 客户端侧的代理, 每个远程对象都包含一个代理对象stub, 当运行在本地Java虚拟机上的程序调用运行在远程Java虚拟机上的对象方法时, 它首先在本地创建该对象的代理对象stub, 然后调用代理对象上匹配的方法。

骨架(Skeleton): 服务端侧的代理, 用于读取stub传递的方法参数, 调用服务器方的实际对象方法, 并接收方法执行后的返回值。

⚠ 注意: 在低版本的JDK中, Server与Registry是可以不在一台服务器上的, 而在高版本的DK中, Server与Registry只能在一台服务器上, 否则无法注册成功。比如jdk8u121这个分界线。

1.3、示例代码

老规矩, 先创建一个名为 `rmidemo` 的工程文件。并在java目录下创建三个包, 分别名为 `method`, `rmiclient`, `rmiserver`。最终目录结构如下图所示:



RMI代码编写步骤如下：

1. 创建远程接口及声明远程方法 (SayHello.java)
2. 实现远程接口及远程方法 (继承UnicastRemoteObject) (SayHelloImpl.java)
3. 启动RMI注册服务，并注册远程对象 (RmiServer.java)
4. 客户端查找远程对象，并调用远程方法 (RmiClient.java)
5. 执行程序：启动服务端RmiServer，运行客户端RmiClient进行调用

我们对上面流程进行拆解，编写对应的代码。

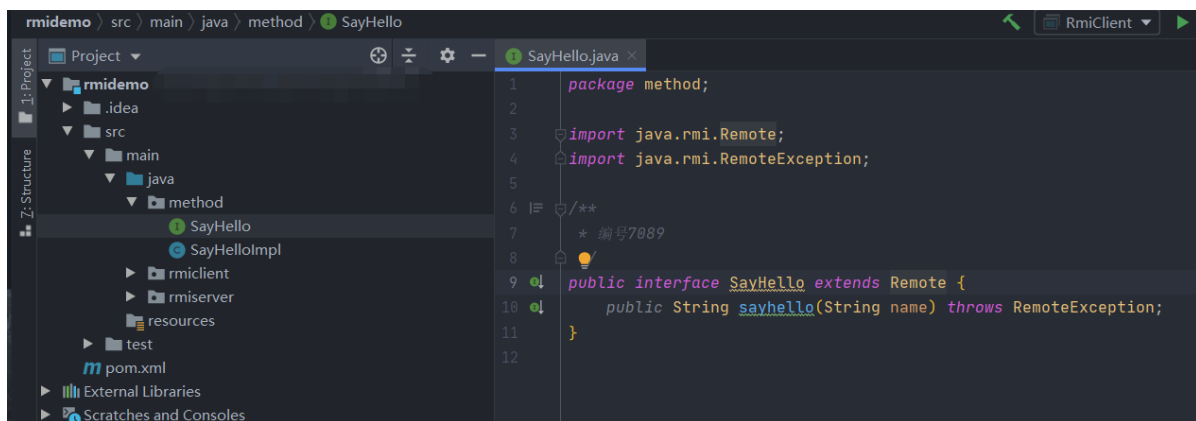
①、创建远程接口及声明远程方法

一定要先创建声明一个远程方法，才能用于后续的远程方法调用。在这里我们以经典的Hello World为例。远程方法中的接口均要继承 `Remote`，实际上 `Remote` 类中没有任何代码，继承也仅是为了说明该是接口使用于远程方法。

在 `src.main.java.method` 目录下新建一个名为 `SayHello` 的java Interface，并键入以下代码，最终如下图所示：

```
package method;
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface SayHello extends Remote {
    public String sayhello(String name) throws RemoteException;
}
```



②、实现远程接口及远程方法

此步骤编写远程方法中具体实现代码。需要注意的是，它必须继承 `UnicastRemoteObject` 类，表明其可以作为远程对象，并可以被注册到注册中心，最终可以让客户端远程调用。

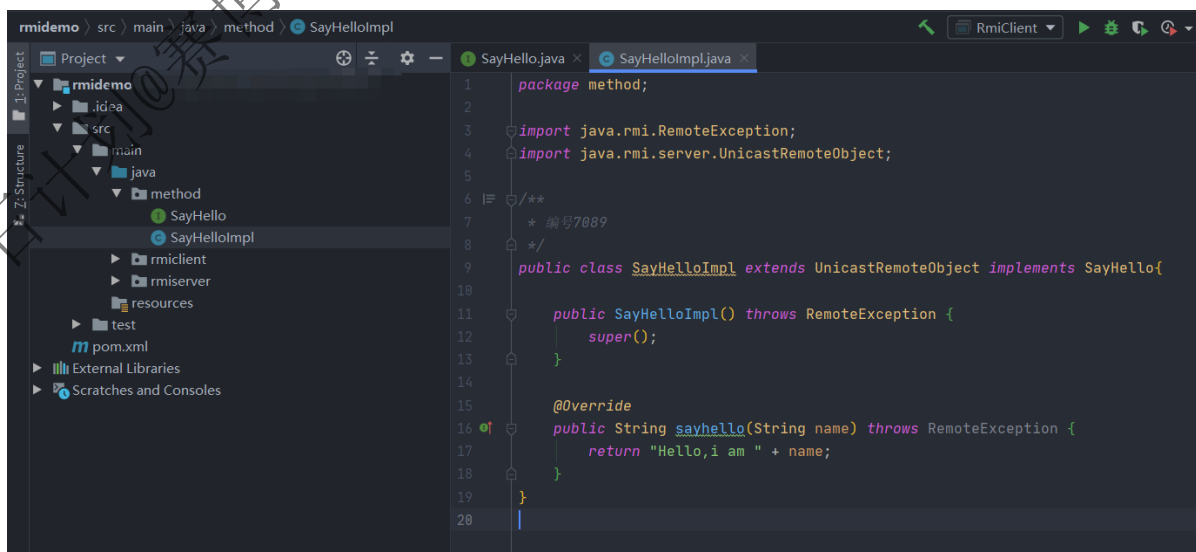
在 `src.main.java.method` 目录下新建一个名为 `SayHelloImpl` 的 Java Class，并键入以下代码，最终如下图所示：

```
package method;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class SayHelloImpl extends UnicastRemoteObject implements SayHello{

    public SayHelloImpl() throws RemoteException {
        super();
    }

    @Override
    public String sayHello(String name) throws RemoteException {
        return "Hello,i am " + name;
    }
}
```



③、启动RMI注册服务，并注册远程对象

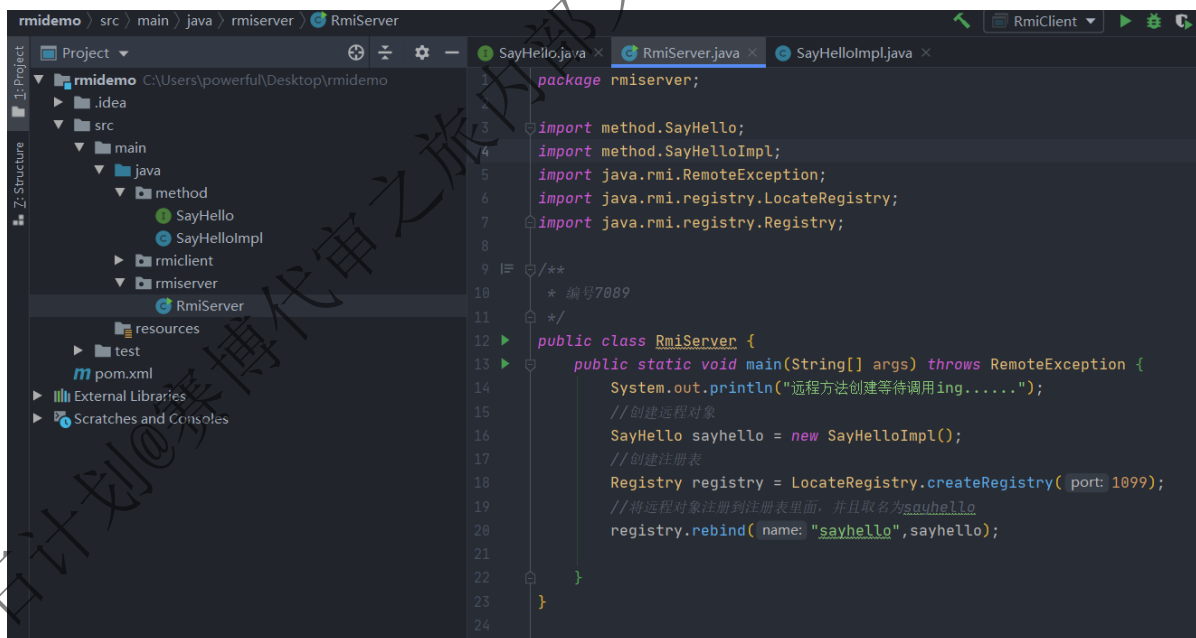
这个步骤我们是编写RMI服务端代码，需要将上面编写的远程方法注册到注册中心去。

在 `src.main.java.rmiserver` 目录下新建一个名为 `RmiServer` 的Java Class，并键入以下代码，最终如下图所示：

```
package rmiserver;

import method.SayHello;
import method.SayHelloImpl;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class RmiServer {
    public static void main(String[] args) throws RemoteException {
        System.out.println("远程方法创建等待调用ing.....");
        //创建远程对象
        SayHello sayhello = new SayHelloImpl();
        //创建注册表
        Registry registry = LocateRegistry.createRegistry(1099);
        //将远程对象注册到注册表里面，并且取名为sayhello
        registry.rebind("sayhello", sayhello);
    }
}
```



④、客户端查找远程对象，并调用远程方法

这个步骤我们是编写RMI客户端代码，主要是获取到注册中心代理，查询具体注册的远程方法并调用。

在 `src.main.java.rmiclient` 目录下新建一个名为 `RmiClient` 的Java Class，并键入以下代码，最终如下图所示：

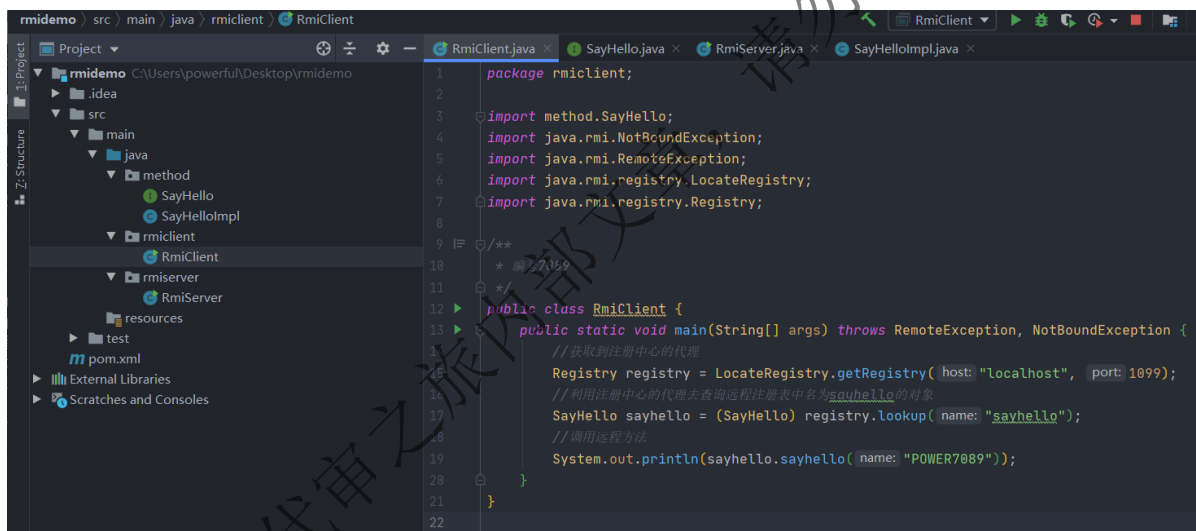
```
package rmiclient;
```

```

import method.SayHello;
import java.rmi.NotBoundException;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

/**
 * 编号7089
 */
public class RmiClient {
    public static void main(String[] args) throws RemoteException,
    NotBoundException {
        //获取到注册中心的代理
        Registry registry = LocateRegistry.getRegistry("localhost", 1099);
        //利用注册中心的代理去查询远程注册表中名为sayhello的对象
        SayHello sayhello = (SayHello) registry.lookup("sayhello");
        //调用远程方法
        System.out.println(sayhello.sayhello("POWER7089"));
    }
}

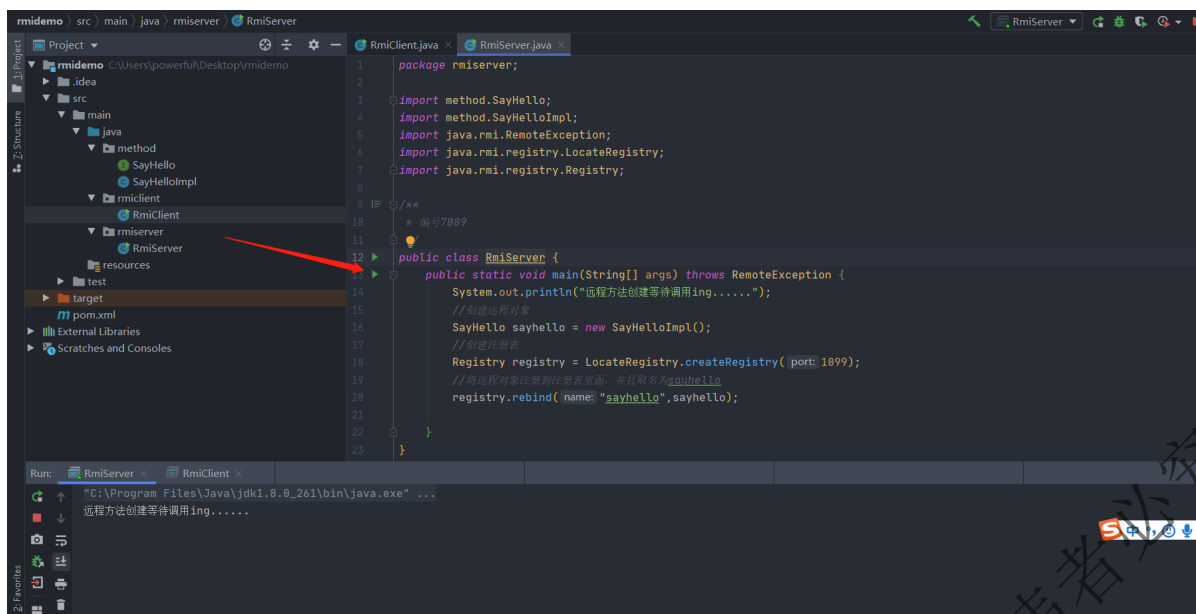
```



⑤、执行程序

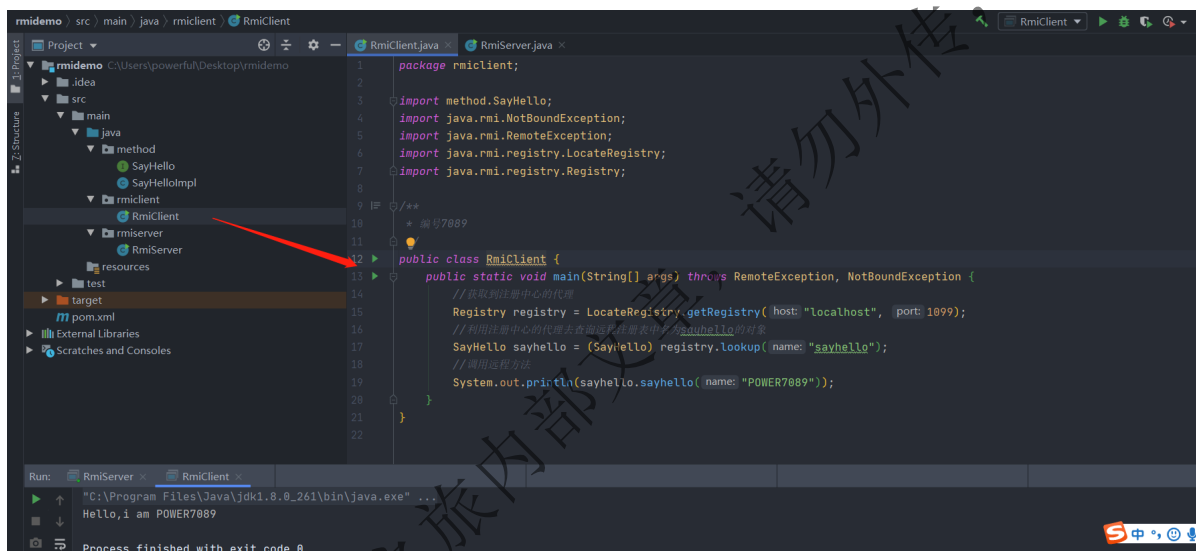
这个步骤我们运行程序观察结果。

首先，启动服务端RmiServer，如下图所示：



```
1 package rmiserver;
2
3 import method.SayHello;
4 import method.SayHelloImpl;
5 import java.rmi.RemoteException;
6 import java.rmi.registry.LocateRegistry;
7 import java.rmi.registry.Registry;
8
9 /**
10  * 编号7089
11  */
12 public class RmiServer {
13     public static void main(String[] args) throws RemoteException {
14         System.out.println("远程方法创建等待调用ing.....");
15         //创建远程对象
16         SayHello sayhello = new SayHelloImpl();
17         //创建注册表
18         Registry registry = LocateRegistry.createRegistry( port: 1099);
19         //将远程对象注册到注册表里面, 并且取名为sayhello
20         registry.rebind( name: "sayhello", sayhello);
21     }
22 }
23
```

最后，启动客户端RmiClient，如下图所示：



```
1 package rmiclient;
2
3 import method.SayHello;
4 import java.rmi.NotBoundException;
5 import java.rmi.RemoteException;
6 import java.rmi.registry.LocateRegistry;
7 import java.rmi.registry.Registry;
8
9 /**
10  * 编号7089
11  */
12 public class RmiClient {
13     public static void main(String[] args) throws RemoteException, NotBoundException {
14         //获取到注册中心的代理
15         Registry registry = LocateRegistry.getRegistry( host: "localhost", port: 1099);
16         //利用注册中心的代理去查询远程注册表中名为sayhello的对象
17         SayHello sayhello = (SayHello) registry.lookup( name: "sayhello");
18         //调用远程方法
19         System.out.println(sayhello.sayhello( name: "POWER7089"));
20     }
21 }
22
```

大家动手调试下吧。

至此，RMI基础到这里就结束了。在了解基础概念后，我们下面会继续学习如何攻击RMI服务。