

Тема    **RMI**

Часть    **Введение в технологию**

Автор    **ASKIL (omendba@gmail.com)**

22.03.2007

Чтобы создать RMI-приложение, необходимо выполнить следующие шаги:

- Создать интерфейс, который расширяет *java.rmi.Remote* интерфейс. Этот интерфейс определяет экспортируемые методы, которые удаленные объекты реализуют. Каждый метод в этом интерфейсе должен быть объявлен, чтобы при необходимости вызвать *java.rmi.RemoteException*, который является суперклассом определенных для RMI исключений.

- Определить подкласс *java.rmi.server.UnicastRemoteObject*, который реализует удаленный интерфейс. Этот класс представляет удаленный объект.

- Написать программу (server) которая создает экземпляр удаленного объекта. Экспортируйте объект, делая его доступным для использования клиентами, регистрируя объект по имени в сервисе реестра. Это обычно делается с помощью *java.rmi.Naming* класса и программы *rmiregistry*. Программа сервер может также действовать как собственный сервер реестра, используя *LocateRegistry* класс и интерфейс *Registry* пакета *java.rmi.registry*.

- Если server использует текущую службу реестра, предоставляемую классом *Naming*, вы должны запустить сервер реестра, если он еще не работает. Для этого необходимо использовать программу *rmiregistry*.

- Написать программу-клиент для использования удаленного объекта, экспортируемого сервером. Клиент должен сначала получить ссылку на удаленный объект, используя класс *Naming*, чтобы найти объект по имени, например *rmi:URL*. Удаленная ссылка, которая возвращается – экземпляр удаленного интерфейса для объекта (или более определенно – объект заглушка для удаленного объекта). Как только клиент имеет этот удаленный объект, он может вызывать его методы точно также как будто это локальный объект. Единственно, что необходимо помнить – все удаленные методы могут вызывать исключения *RemoteException*.

- Наконец, запустить программу server и запустить на работу клиента.

Создадим программу, которая возвращает содержимое директории на сервере.

## 1. Интерфейс (InterfaceClient.class)

```
import java.rmi.*;
public interface InterfaceClient extends Remote
{
    public String GetListFoldersInDirectoryOnServer(String dir)
        throws RemoteException, java.rmi.server.ServerNotActiveException;
}
```

## 2. Реализация интерфейса (IpmlInterfaceClient.class)

```
import java.io.*;
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;
public class IpmlInterfaceClient
    extends UnicastRemoteObject
    implements InterfaceClient
{
    //Конструктор
    public IpmlInterfaceClient() throws RemoteException
    {
```

```

        System.out.println("The object IpmlInterfaceClient is create!");
    }
    //Реализация класса
    public String GetListFoldersInDirectoryOnServer(String dir)
        throws RemoteException, java.rmi.server.ServerNotActiveException
    {
        System.out.println("Client "+this.getClientHost()+
            " call method GetListFoldersInDirectoryOnServer!");
        String filedir="";
        File f = new File(dir);
        File [] indir = f.listFiles();
        for (int i=0;i<indir.length;i++)
            filedir+=indir[i].getName()+"\n";
        return filedir;
    }
}

```

### 3. Сервер (Server.class).

```

import javax.naming.*;
public class Server
{
    public static void main (String args[]) throws Exception
    {
        System.out.println("The server is start!");
        IpmlInterfaceClient c = new IpmlInterfaceClient();
        Context nc = new InitialContext();
        nc.bind("rmi:RMIObject",c);
    }
}

```

### 4. Клиент (Client.class).

```

import java.rmi.*;
import javax.naming.*;
public class Client
{
    public Client(String server, String dir, String policy) throws Exception
    {
        System.setProperty("java.security.policy",policy);
        System.setSecurityManager(new RMISecurityManager());
        String url = "rmi://" + server + "/";
        Context nc = new InitialContext();
        InterfaceClient c = (InterfaceClient) nc.lookup(url+"RMIObject");
        System.out.println(c.GetListFoldersInDirectoryOnServer(dir));
    }
    public static void main(String [] args) throws Exception
    {
        new Client(args[0],args[1],args[2]);
    }
}

```

Клиент должен иметь право на соединение с сервером, поэтому создадим файл политики client.policy:

```

grant
{
    permission java.net.SocketPermission " *:1024-65000", "connect";
};

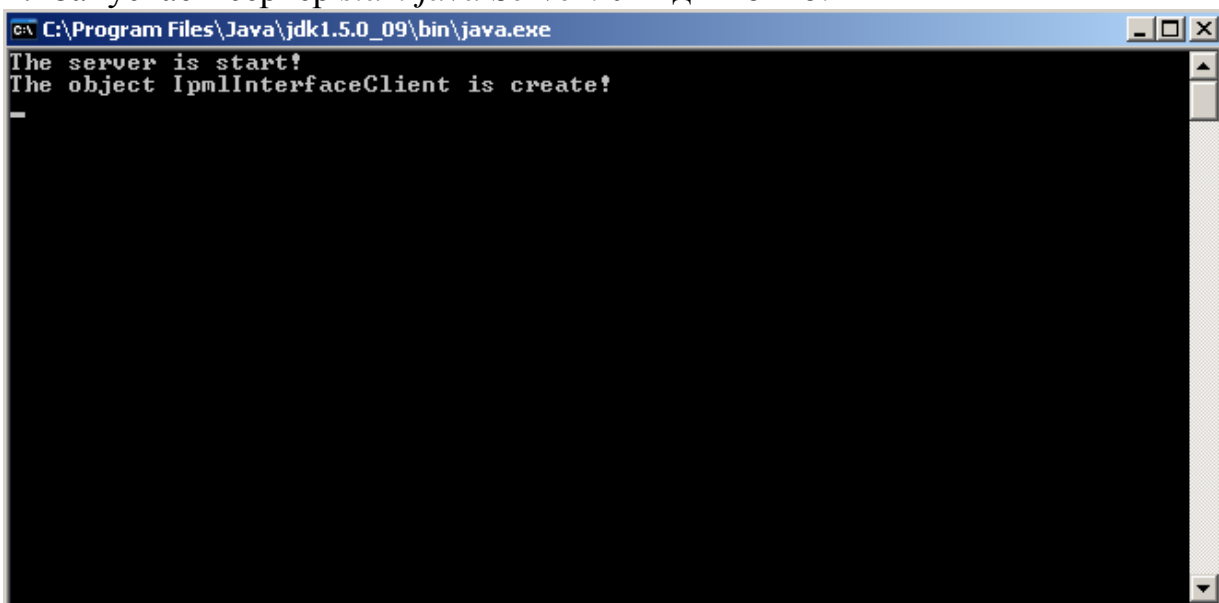
```

Запуск:

1. На сервере 192.168.181.1 запускаем *rmiregistry.exe*.
2. На сервер в папку *jdk1.5.0\_09\bin* копируем файлы *InterfaceClient.class*, *IpmlInterfaceClient.class*, *Server.class*.

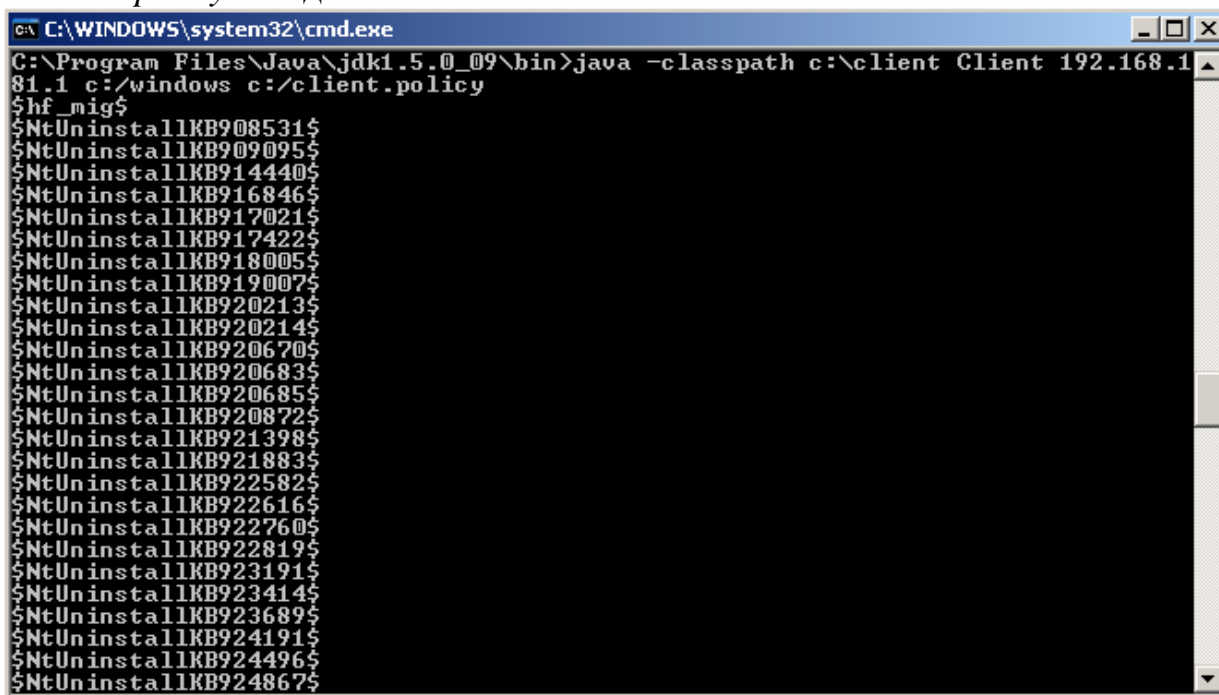
3. Клиенту 192.168.181.1 отправляем файлы *client.policy*, *Client.class*, *InterfaceClient.class*.

4. Запускаем сервер *start java Server*. Увидим окно:



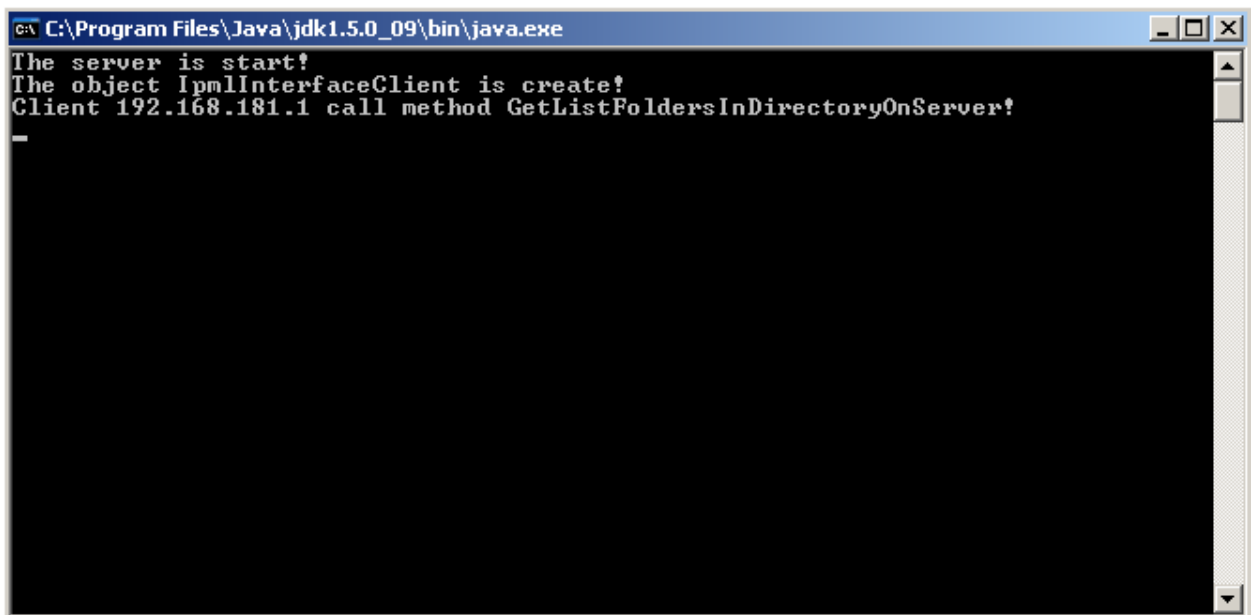
```
C:\Program Files\Java\jdk1.5.0_09\bin\java.exe
The server is start!
The object IpmlInterfaceClient is create!
-
```

5. Запускаем клиента *java -classpath c:/client Client 192.168.181.1 c:/windows c:/client.policy* и видим окно



```
C:\WINDOWS\system32\cmd.exe
C:\Program Files\Java\jdk1.5.0_09\bin>java -classpath c:\client Client 192.168.181.1 c:/windows c:/client.policy
$hf_mig$
$NtUninstallKB908531$
$NtUninstallKB909095$
$NtUninstallKB914440$
$NtUninstallKB916846$
$NtUninstallKB917021$
$NtUninstallKB917422$
$NtUninstallKB918005$
$NtUninstallKB919007$
$NtUninstallKB920213$
$NtUninstallKB920214$
$NtUninstallKB920670$
$NtUninstallKB920683$
$NtUninstallKB920685$
$NtUninstallKB920872$
$NtUninstallKB921398$
$NtUninstallKB921883$
$NtUninstallKB922582$
$NtUninstallKB922616$
$NtUninstallKB922760$
$NtUninstallKB922819$
$NtUninstallKB923191$
$NtUninstallKB923414$
$NtUninstallKB923689$
$NtUninstallKB924191$
$NtUninstallKB924496$
$NtUninstallKB924867$
```

На сервере в главном окне увидим

A screenshot of a Windows command window titled "C:\Program Files\Java\jdk1.5.0\_09\bin\java.exe". The window has a blue title bar and standard Windows window controls (minimize, maximize, close) in the top right corner. The main area is black with white text. The text displayed is: "The server is start!", "The object IpmlInterfaceClient is create!", and "Client 192.168.181.1 call method GetListFoldersInDirectoryOnServer!". There is a small white cursor on the line following the last message. A vertical scrollbar is visible on the right side of the window.

```
C:\Program Files\Java\jdk1.5.0_09\bin\java.exe
The server is start!
The object IpmlInterfaceClient is create!
Client 192.168.181.1 call method GetListFoldersInDirectoryOnServer!
_
```