**Aplicação Java RMI usando NetBeans**  
  
Nesta aula mostrarei passo-a-passo como escrever um aplicação RMI bem simples usando a IDE NetBeans (aqui usei a versão 6.7).

Note, contudo, que isso não o(a) liberará da tarefa de executar alguns comandos na janela do prompt de comando. Desta forma, se você ainda não sabe como fazê-lo, as variáveis de ambiente PATH e CLASSPATH precisrão ser configuradas.

**Sobre a aplicação Calculadora**  
  
A aplicação RMI que escreveremos é realmente muito simples.   
  
Trata-se de uma calculadora (chamada aqui de Calculator) que recebe dois inteiros e devolve os resultados das quatro operações: soma, subtração, multiplicação e divisão.

A aplicação-cliente (chamada aqui CalculatorClient) se conectará à aplicação- servidor (chamada aqui CalculatorServer).   
  
CalculatorClient enviará dois valores inteiros e o servidor devolverá a o resultado das operações soma, subtração, multiplicação e divisão, dos valores fornecidos.

Lembre-se de que usaremos apenas Java RMI, nada de sockets envolvidos. Sockets, agora, somente nos bastidores, ou seja, na implementação de mais baixo nível que é transparente para você. 

---------------------------------------------------------------  
 **1. Escrevendo a aplicação Calculadora em Java RMI**   
  
O primeiro passo para escrever a aplicação cliente é abrir o NetBeans e criar um novo projeto.

Em seguida vá até a opção Arquivo -> Novo Projeto.

Na janela de novo projeto escolha Java de um lado e Aplicativo Java do outro.   
  
Clique o botão Próximo.

Dê o nome Calculadora ao projeto e salve-o no diretório

"C:\...\NetbeansProjects\RMI\"

de forma que a pasta do projeto seja   
  
"C:\...\NetbeansProjects\RMI\Calculadora".   
  
Clique o botão Finalizar.  
--------------------------------------------------------------

**2. Criando a interface remota**  
Vamos agora criar a interface remota, que será compartilhada tanto pela aplicação-servidor quanto pela aplicação-cliente.

Assim, vá na opção Arquivo -> Novo Arquivo.

Na janela Novo arquivo escolha Java de um lado e Interface Java do outro.   
  
Clique o botão Próximo.

Dê o nome "Calculator" à interface e selecione o pacote "calculadora".   
  
Clique o botão Finalizar.  
  
Altere o código do arquivo Calculator.java para a versão mostrada abaixo:

Na interface RMI abaixo são definidos os métodos add(), sub(), mul() e div() que retornam os resultados das operações artiméticas quando são chamados.

Descreve a interface Calculator

public interface Calculator

extends java.rmi.Remote {

public long add(long a, long b)

throws java.rmi.RemoteException;

public long sub(long a, long b)

throws java.rmi.RemoteException;

public long mul(long a, long b)

throws java.rmi.RemoteException;

public long div(long a, long b)

throws java.rmi.RemoteException;

}

A exceção java.rmi.RemoteException indica erros na chamada remota, e deve ser prevista pelos métodos de interfaces RMI.

--------------------------------------------------------------------------------------

**3. Criando a implementação da interface Calculator, chamada CalculatorImpl**

Vamos agora escrever uma classe que implementa esta interface Calculator.   
  
Vá na opção Arquivo -> Novo Arquivo.

Na janela Novo arquivo escolha Java de um lado e Classe Java do outro.

Clique o botão Próximo.

Dê o nome "CalculatorImpl" à classe e selecione o pacote "calculator".   
  
Clique o botão Finalizar.  
  
Altere o código do arquivo CalculatorImpl.java para a versão mostrada abaixo:

Implementando a interface Calculator

public class CalculatorImpl extends   
 java.rmi.server.UnicastRemoteObject implements Calculator {

public CalculatorImpl()

throws java.rmi.RemoteException {

super();

}

public long add(long a, long b)

throws java.rmi.RemoteException {

return a + b;

}

public long sub(long a, long b)

throws java.rmi.RemoteException {

return a - b;

}

public long mul(long a, long b)

throws java.rmi.RemoteException {

return a \* b;

}

public long div(long a, long b)

throws java.rmi.RemoteException {

return a / b;

}

}

----------------------------------------------------------------------------------

**4. Criando o servidor CalculatorServer**

Agora já podemos implementar o servidor CalculatorServer e registrá-lo com o RMI server.

Altere o código do arquivo Main.java para a seguinte versão:

Na função main() do servidor iremos criar um objeto que implementa a interface Calculator e registrá-lo como um servidor no registro do RMI, com o nome "CalculatorService", para que ele possa ser localizado pelo cliente CalculatorClient.

import java.rmi.Naming;

public class CalculatorServer {

public CalculatorServer() {

try {

Calculator c = new CalculatorImpl();

Naming.rebind("//localhost/CalculatorService", c);

} catch (Exception e) {

System.out.println("Trouble: " + e);

}

}

public static void main(String args[]) {

new CalculatorServer();

}

}

Pronto! A aplicação-servidor já está concluída.

Pressione F6 para executá-la. Teremos o seguinte erro de tempo de execução:

Trouble: java.rmi.ConnectException: Connection refused to host:   
 localhost; nested exception is:   
 java.net.ConnectException: Connection refused: connect

Abra uma janela de comando e inicie o RMI Registry.

> cd   
  
C:\Users\bosco\Documents\NetbeansProjects\RMI\Calculadora\src\calculadora\start rmiregistry

Isso fará com que uma nova janela de comando seja aberta. Não a feche. O RMI Registry está sendo executado nela.

-------------------------------------------------------------------------------------

Volte à aplicação e pressione F6 novamente. Agora o erro de tempo de execução é:

Trouble: java.rmi.ServerException: RemoteException occurred in server thread; nested exception is:   
java.rmi.UnmarshalException: error unmarshalling arguments; nested exception is:   
java.lang.ClassNotFoundException: calculadora.Calculator

Este erro ocorreu porque o RMI Registry não conseguiu encontrar nenhuma informação a respeito da interface Calculator.

Para contornarmos este problema, vamos apenas construir a aplicação: (pressionando F11 ou acessando Executar -> Construir projeto principal).  
  
Isto resultou em:

-----------------------------------------------------------------------------------------

Created dir: C:\Users\bosco\Documents\NetBeansProjects\RMI\Calculadora\build\empty

Compiling 5 source files to C:\Users\bosco\Documents\NetBeansProjects\RMI\Calculadora\build\classes

compile:

Created dir: C:\Users\bosco\Documents\NetBeansProjects\RMI\Calculadora\dist

Building jar: C:\Users\bosco\Documents\NetBeansProjects\RMI\Calculadora\dist\Calculadora.jar

Not copying the libraries.

To run this application from the command line without Ant, try:

java –jar

"C:\Users\bosco\Documents\NetBeansProjects\RMI\Calculadora\dist\Calculadora.jar"

jar:

CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)

-------------------------------------------------------------------

Verifique que o arquivo Calculadora.jar está mesmo em

C:\Users\bosco\Documents\NetBeansProjects\RMI\Calculadora\dist

Agora, vamos executar a partir do arquivo .jar gerado (Calculadora.jar).

Assim, abra uma janela de terminal e vá até o diretório de sua localização:  
  
> cd   
  
C:\Users\bosco\Documents\NetbeansProjects\RMI\Calculadora\dist>  
java -jar Calculadora.jar

Mesmo erro ?   
  
Pare o RMI (fazendo Ctrl + C),   
  
Feche todas as janelas de comando e coloque a sua variável de ambiente CLASSPATH como:

C:\Users\bosco\Documents\NetbeansProjects\RMI\Calculadora\dist

Execute

C:\Users\bosco\Documents\NetbeansProjects\RMI\Calculadora\dist>

Calculadora.jar  
  
Abra uma nova janela de terminal e inicie o RMI novamente,

>start registry

Vá até o diretório abaixo e execute o JAR novamente.   
  
>C:\Users\bosco\Documents\NetbeansProjects\RMI\Calculadora\dist>  
Calculadora.jar   
  
Se tudo correu bem você terá o resultado esperado.

Esta mensagem indica que a aplicação-servidor está 100% funcionando e pronto para atender às requisições da aplicação-cliente.   
 **Nota importante:**   
  
Neste momento você já consegue executar a aplicação-servidor a partir do NetBeans. Mas, é recomendável executá-la a partir do prompt de comando, uma vez que a aplicação-servidor deverá estar rodando quando a aplicação cliente for executada.

Hora de passarmos ao desenvolvimento do cliente.

**5. Criando a aplicação-cliente CalculatorClient**  
  
Cria um cliente que obtenha uma referência para o servidor no registro RMI e chame os métodos add(), sub(), mul() e div().

Vá na opção Arquivo -> Novo Arquivo.

Na janela Novo arquivo escolha Java de um lado e Classe Java do outro.

Clique o botão Próximo.

Dê o nome "CalculatorClient" à classe e selecione o pacote "calculadora".   
  
Clique o botão Finalizar.  
  
Altere o código do arquivo CalculatorClient.java para a versão mostrada abaixo:

import java.rmi.Naming;

public class CalculatorClient {

public static void main(String[] args) {

try {

Calculator c =

(Calculator) Naming.lookup( "rmi://jaca.ime.usp.br:1099/CalculatorService");

System.out.println( c.sub(4, 3) );

System.out.println( c.add(4, 5) );

System.out.println( c.mul(3, 6) );

System.out.println( c.div(9, 3) );

}

catch (Exception e) {

System.out.println(e);

}

}

}

6. Compilando a Aplicação RMI

Edite e compile os arquivos acima usando o compilador Java ou um IDE da sua escolha.

> javac \*.java

> rmic CalculatorImpl (usando compilador RMI “rmic” )

Como resultado são criados os stubs e os skeletons para cada objeto da lógica da aplicação ( os orieundos das classes <...>Impl )

Stubs e Skeletons são os códigos para a comunicação via RMI) para a nossa aplicação

Executando a Aplicação RMI

Primeiramente, inicie o registro RMI com o comando:

> start rmiregistry

Em seguida, inicie o servidor:

> start java CalculatorServer

Finalmente, inicie o cliente com o comando:

> java CalculatorClient localhost

Veja os resultados !!!

----------------------------------------------------------------------------------------

Tente também, com a ajuda de um colega, fazer a comunicação entre máquinas diferentes.

Para isto, substitua 'localhost' pelo nome da máquina do outro grupo ao chamar o cliente.

Um exemplo, numa máquina remota jaca.ime.usp:1099  
  
"rmi://jaca.ime.usp.br:1099/CalculatorService")

-----------------------------------------------------------------------------------------

Vejam as variáveis de ambientes:  JAVA\_HOME,  CLASSPATH  e  PATH  
No Painel de Controle do Windows, procurem Sistema, depois Anvançado, depois Variáveis de Ambiente  
  
Compilando pelo prompt da linha de comando e estabelecendo variáveis de ambiente no Windows:  
  
Façam como no link:  <http://richardikeda.com/blog/java-compilando-pelo-prompt.html>    
  
Estabelecer as variáveis de ambiente:     
  
 JAVA\_HOME  =  C:\Arquivos de programas\Java\jdk1.6.0\_xx  
 CLASSPATH  =  .;C:\Arquivos de programas\Java\jdk1.6.0\_xx\lib

(lembre do ponto,  seguido de ponto e vírgula, no antes de C:\, no estabelecimento da variável de ambiente CLASSPATH.                         

PATH = C:\Arquivos de programas\Java\jdk1.6.0\_xx\bin  
  
Inicializar o computador