Tutorial RMI
(Remote Method Invocation)
por Alabê Duarte

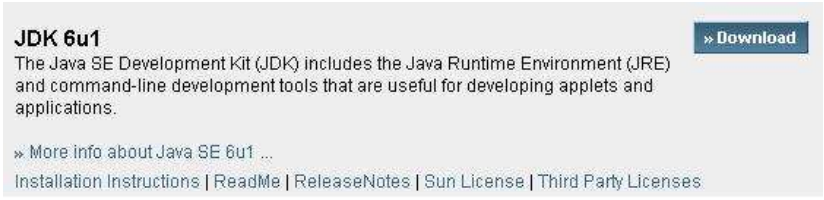
Este tutorial explica basicamente como se implementa a API chamada **RMI (Remote Method Invocation)**. O RMI nada mais é que a Invocação de Métodos Remotamente, ou seja, permite a execução de chamadas remotas em aplicações desenvolvidas na linguagem **Java**. A maior parte deste tutorial é dedicada a **Thiago Rodrigues** e a **Victor Reyes**, ambos estudantes da Faculdade Ruy Barbosa – Salvador BA, pois foram de grande ajuda no estudo desta API. Para que isso seja possível, é necessária a instalação de algumas ferramentas e é preciso também que alguns procedimentos sejam seguidos.

As ferramentas necessárias pra começar a desenvolver são: **Eclipse 3.2** e **JDK 6.0**.
É necessário o **Eclipse** com o plugin **Web Tools Platform(WTP)**. Existe uma versão do eclipse que já vem com o plugin incluso (**1.5.3**) e esta descrito como **WebTools Platform All-in-one**. Esta pode ser baixada neste endereço: http://download.eclipse.org/webtools/downloads

Após o download, pra instalar basta descompactar no diretório de preferência.

O **JDK 6.0** pode ser baixado no endereço:
http://java.sun.com/javase/downloads/index.jsp

É recomendável que desinstale as versões anteriores antes de executar o eclipse ou mudar as variáveis de ambiente para tornar esta versão padrão.



Após as configurações necessarias, vamos à implementação.

O desenvolvimento foi feito em duas máquinas, mas nada impede que seja feita em apenas uma máquina para testar o seu funcionamento. Porém, simularemos uma aplicação “Hello World” feita em duas máquinas.

Inicialmente, criaremos um projeto Java comum (J2SE) compostos por um pacote chamado **servidor**. Dentro deste pacote haverá:

- ServidorRMI.java (código java que é a chave da aplicação distribuída.)
- Mensagem.java (interface que será implementada pela classe onde haverá o metodo a ser chamado remotamente.)
- MensagemImpl.java (implementação da interface Mensagem.java)

Na outra máquina criaremos um projeto Java comum (J2SE) composto por um pacote chamado **cliente**. Dentro deste pacote haverá:

- Cliente.java (código que acessará o servidor remotamente que será mostrado mais tarde.)

Segue abaixo o código: **Mensagem.java**

```
package servidor;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface Mensagem extends Remote {
	public String mostraMensagem() throws RemoteException;
	}
```

Segue abaixo o código: **MensagemImpl.java**

```
package servidor;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class MensagemImpl extends UnicastRemoteObject implements Mensagem {

	// precisamos definir um construtor pois o default não
	// gera RemoteException
		protected MensagemImpl() throws RemoteException {

			// opcional, vai ser chamado mesmo que não
			// o coloquemos aqui
			super();
		}

		public String mostraMensagem() throws RemoteException {
			return "RMI, EU CONSEGUI!";
		}

	}
```

Segue abaixo o código: **ServidorRMI.java** (classe que será acessada remotamente por **Client.java**)

```
package servidor;


import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;



public class ServidorRMI {


      public static void main(String[] args) throws RemoteException {

            int port = 1099; //número-padrão da porta para o rmiregistry.

            try {

                  //registra a porta
                  LocateRegistry.createRegistry( port );

            } catch ( Exception e1 ) {
              }

            //Criação do gerenciador de segurança (Security Manager)
            if ( System.getSecurityManager() == null )
                    System.setSecurityManager(new SecurityManager());

            try {

                   String name = "MensagemServer";

                        // aqui criamos o objeto que sera acessado
                        // remotamente
                        MensagemImpl m = new MensagemImpl();

                    Registry registry = LocateRegistry.getRegistry( port );

                    // aqui registramos o objeto 'm' como 'name'
                    // no servidor
                    registry.rebind(name, m);

                   System.out.println( " Servidor RMI Rodando... ");

                   } catch (Exception e) {

                     System.err.println("exceção RMI:");
                     e.printStackTrace();

                  }
      }
```

Gerando classes Stub e Skeleton:

Compilaremos nossas classes normalmente e após isso compilaremos a classe de implementação utilizando o rmic para gerar os Stubs* e Skeletons.

* Stub é um espécie de proxy que ficará junto com a aplicação cliente e permitirá o acesso aos objetos que estão no servidor RMI.

Primeiramente, dirija-se ao pacote onde está localizado o servidor e a classe que contém o(s) método(s) será/serão acessada(s) remotamente. Crie um documento de texto e escreva o seguinte:

```
rem @echo off
Echo Compilando
rmic -v1.1 -d C:\workspace\ExemploRMI\ -classpath C:\workspace\ExemploRMI
servidor.MensagemImpl

pause
```

Agora clique em Salvar como... e mude a extensão do arquivo de .txt para .bat.

Note que o diretório **C:\workspace\ExemploRMI** foi o diretório escolhido para a criação do projeto. É preciso muita atenção na edição deste arquivo .bat. Coloque o diretório correspondente ao projeto que você, litor, está desenvolvendo.

Pronto, agora é só clicar no arquivo.bat criado para gerar automaticamente os arquivos:

- MensagemImpl_Stub.class
- MensagemImpl_Skel.class

Iniciando o Servidor:

Antes de iniciarmos o servidor, precisaremos também de um arquivo de policy para especificar os privilérios de acesso ao servidor.
Crie um arquivo com o código abaixo e salve-o com o nome de "policy" no diretório do projeto:

```
grant{
          permission java.security.AllPermission;
        };
```

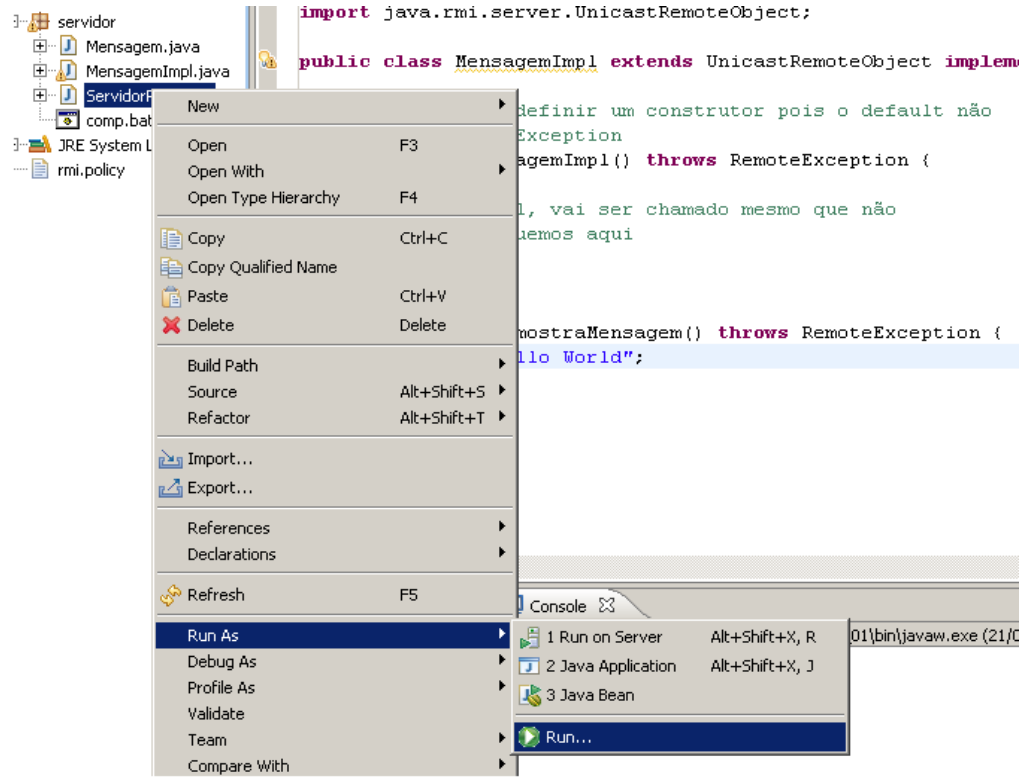
No nosso caso, chamaremos este arquivo “policy” de rmi.policy.

Agora poderemos iniciar nosso servidor. Para tanto executaremos a seguinte instrução, a partir do diretório onde estão os fontes, na linha de comando:

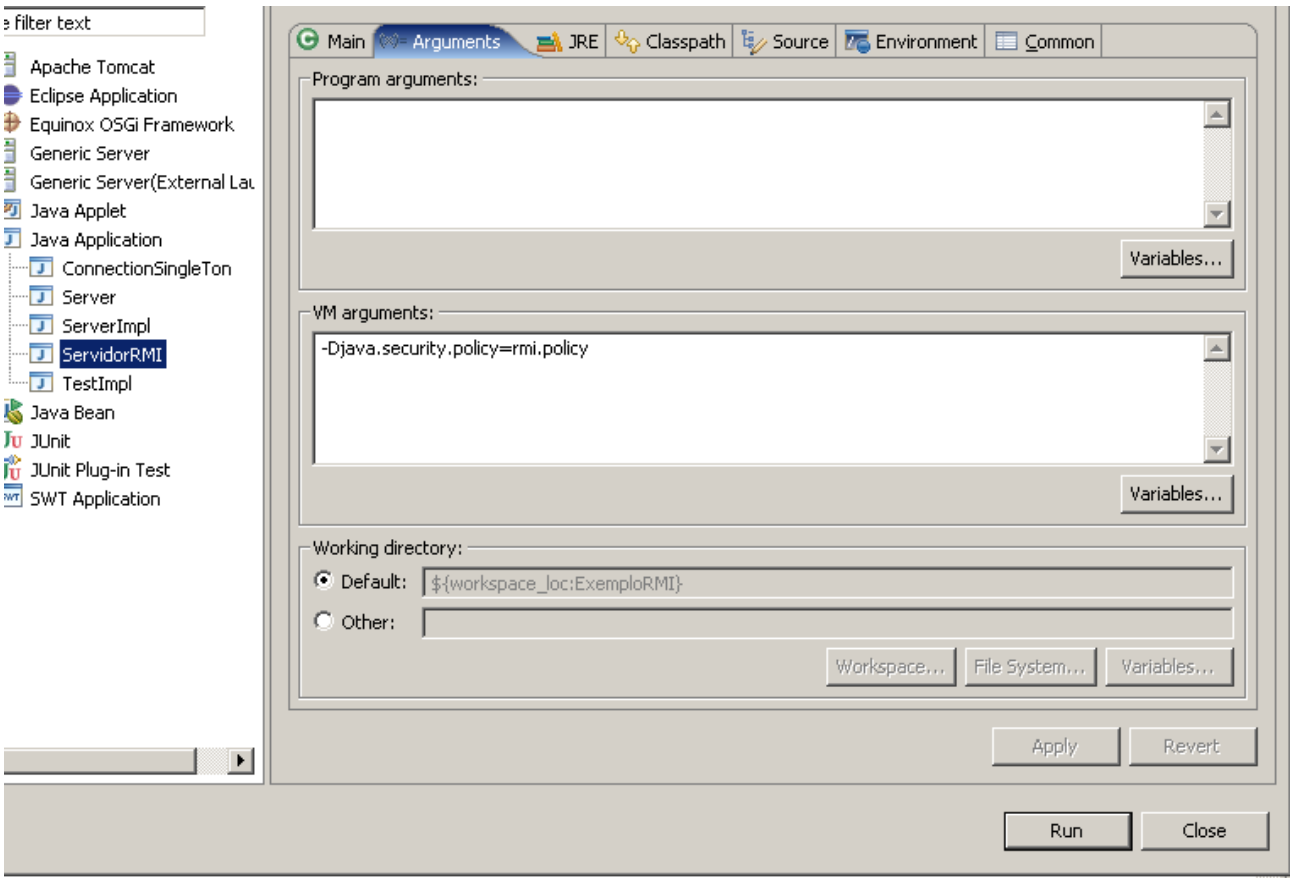
```
-Djava.security.policy=rmi.policy
```

Essa instrução será dada da seguinte forma:

Dentro do eclipse, dirija-se à classe ServidorRMI, clique com o botão direito do mouse e clique na opção **Run As/Run...** como mostra a figura abaixo:



Após o clique, selecione a classe ServidorRMI (se não aparecer clique com o botão direito, aperte **new** e selecione a sua classe.) clique na aba **Arfuments** e digite a instrução de segurança.



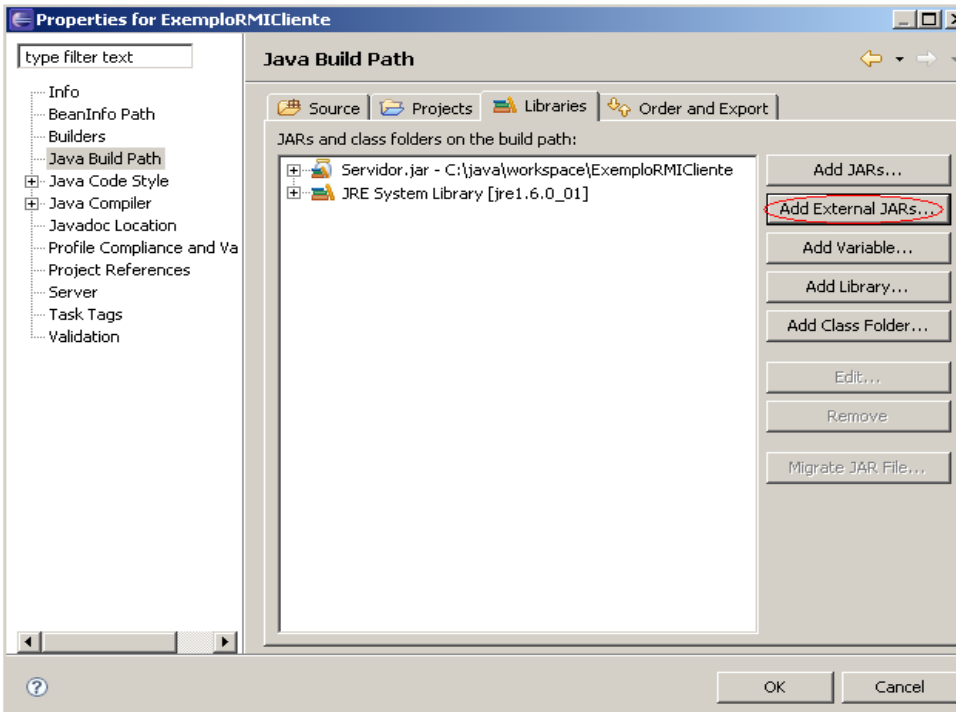
Feito isto, é só clicar em Run e ficar atento ao console do eclipse.
A mensagem esperada é: “Servidor RMI Rodando...”

Vamos agora criar um arquivo .bat para o servidor. Crie um documento de texto e escreva o seguinte comando:

```
rem @echo off
Echo servidor
java -classpath C:\workspace\ExemploRMI\ -Djava.security.policy=rmi.policy
servidor.ServidorRMI
pause
```

Agora salve o documento de texto como um arquivo .bat no diretório correspondente a **ExemploRMI**. Note que o arquivo rmi.policy também deve estar neste diretório.
Pronto, criamos um servidor com sucesso. Próximo passo agora é desenvolver o cliente. Para isso, falta apenas um pequeno detalhe.

Exemplo.jar:
Para começarmos a construir o cliente, teremos que compactar o projeto ExemploRMI na extensão **.jar**. Uma maneira bastante fácil de fazer isto é dirigir-se ao diretório ExemploRMI, selecione todos os arquivos e pastas, e a partir deles, crie uma compactação da extensão **.zip** e renomeie para **.jar**.
Após a compactação, dirija-se ao **projeto do cliente** e diretamente do eclipse faça a importação do .jar (no nosso caso o arquivo foi nomeado de **Servidor.jar**) criado clicando no projeto ExemploRMICliente em propriedades/Java Build Path/Add External JARs... conforme a figura abaixo:



Após o processo, escreva o código do cliente.

Segue abaixo o código: **Cliente.java**

```java
package cliente;

import java.rmi.ConnectException;
import java.rmi.Naming;

import servidor.*;

public class Cliente {

	private static String port = "1099";

	public static void main(String[] args) {

		/* objeto "mensagem" é o identificador que iremos
		 * utilizar para fazer
		 * referencia ao objeto remoto que será implementado
		 */
		Mensagem mensagem = null;

		// ip da máquina do servidor: 10.0.5.124

		String ipConfig = "rmi://10.0.5.124:" + port + "/";
		String retorno = null;

		try {
			/* - o lookup carrega o MensagemImpl_Stub do
			 *   CLASSPATH 10.0.5.124 -
			 * é o IP da máquina de onde o servidor está
			 * rodando.
			 * "MensagemServer" é o nome que utilizamos para
			 *  fazer referencia ao
			 * objeto no servidor.
			 */

			// aqui instanciamos o objeto remoto
			mensagem = (Mensagem) Naming.lookup(ipConfig +
			"MensagemServer");

			// agora executamos o metodo "mostraMensagem" no
			// objeto remoto
			retorno = mensagem.mostraMensagem();
			System.out.println(retorno);
		} catch (ConnectException e) {

			System.out.println("ERRO: Servidor Desconectado");
			//e.printStackTrace();

		} catch (Exception e) {
			System.out.println("Erro Interno");
			//e.printStackTrace();
		}
	}
}
```

Pronto, agora é só rodar a aplicação Java e verá a seguinte mensagem: **“Hello World”**

Agora certamente o leitor deve ser capaz de criar a sua própria aplicação RMI com sistemas distribuídos.

Este tutorial foi desenvolvido por **Alabê Duarte**, estudante da **Faculdade Ruy Barbosa**.