UNIVERSIDADE ESTADUAL DE MARINGÁ

DEPARTAMENTO DE INFORMÁTICA

CIÊNCIA DA COMPUTAÇÃO

Aplicação RMI Java

Autor:
Adriano Ferrari CARDOSO
Gabriel H. Tasso SCHIAVON

Professor(a):
Raqueline PENTEADO

ra77274@uem.br
ra80822@uem.br

10 de maio de 2017

1. Descrição da Aplicação

A aplicação desenvolvida para este trabalho tem como objetivo demonstrar a utilização da funcionalidade RMI[1] nas operações de criação, listagem, edição e remoção de objetos de um banco de dados. Essas operações são comumente conhecidas como CRUD (acrônimo de *Create*, *Read*, *Update* e *Delete* na língua Inglesa).

Uma lista foi utilizada para representação do banco de dados, visto que o foco do trabalho é explorar as funcionalidades do RMI. Nesta lista pode ser inserido um usuário com os seguintes atributos:

- Nome
- Idade
- CPF

2. Desenvolvimento

O RMI é uma interface de programação que permite a execução de chamadas remotas em aplicações desenvolvidas em Java. O funcionamento de RMI consiste basicamente em dois programas, segundo a arquitetura cliente-servidor, onde um seria o cliente e outro o servidor. O servidor instancia objetos remotos, o referencia com um nome e faz um ”vinculo”dele numa porta, onde este objeto espera por clientes que invoquem seus métodos. Já o cliente referencia remotamente um ou mais métodos de um objeto remoto.

2.1. Definição da interface do objeto remoto

Para se tornar um objeto remoto, a classe deve implementar uma interface remota que estende **java.rmi.Remote** e cada método declarado tem de indicar o envio excepções do tipo **RemoteException**:

```
public interface CrudUsuario extends Remote{
  public Retorno add(Usuario u, List<Usuario> lista) throws RemoteException;
  public Retorno remove(String cpf, List<Usuario> lista) throws RemoteException;
  public Retorno edita(Usuario u, List<Usuario> lista) throws RemoteException;
  public String listar(List<Usuario> lista) throws RemoteException;
}
```

2.2. Implementação da classe do objeto remoto

A classe **Gestor** vai implementar os métodos definidos pela interface para o usuário poder utilizá-lo. Segue um trecho do código onde ocorre a implementação do método para adicionar um novo usuário ao banco de dados:

```
public class Gestor extends UnicastRemoteObject implements CrudUsuario{
  protected Gestor() throws RemoteException {
    super();
}
  private static final long serialVersionUID = 1L;

  @Override
  public Retorno add(Usuario u, List<Usuario> lista) throws RemoteException {
    String mensagem = "usuario Inserido com sucesso ! \nDados:\n\tNome: "+
      u.getNome()+"\n\tIdade: "+u.getIdade()+"\n\tCPF: "+u.getCpf();
```

[1] *Remote Method Invocation*

```
    lista.add(u);

    Retorno ret = new Retorno(lista, mensagem);
    return ret;
  }
```

2.3. Criação do *Stub*

A camada mais próxima do programador, ou seja da aplicação cliente e do objeto remoto, é a camada *Stubs*. Os *Stubs* recebem os parâmetros dos métodos exportados pelo objeto remoto (definidas pela interface da classe remota) e reencaminham-nos para o lado do servidor.

Este passo é exclusivo do RMI e consiste na geração automática das classes de *Stub* usando um compilador dedicado denominado **rmic**. O **rmic** gera *Stubs* a partir da classe remota compilada.

2.4. Criação de Aplicação Servidora

Para que um determinado objeto remoto fique disponível aos clientes é necessário que este seja criado e registado no sistema RMI. Assim, vamos criar uma pequena aplicação servidora com o único objectivo de construir um objeto remoto e registá-lo no Serviço de Registos do sistema RMI sob um determinado nome. O registo é feito recorrendo ao método estático RMI **Naming.rebind()** que recebe dois parâmetros: (i) o nome pelo qual o objeto remoto deverá ficar conhecido e (ii) a referência do próprio objeto remoto. A partir do momento em que o registo é executado com sucesso, o objeto encontra-se disponível para acesso remoto. O código fonte é o seguinte:

```
public class UsuarioServidor {
  public static void main(String[] args)
  throws RemoteException, MalformedURLException {
    Gestor gestor = new Gestor();
    Naming.rebind("//localhost/CRUD", gestor);
  }
}
```

2.5. Criação da Aplicação Cliente

A aplicação cliente que vamos criar realiza as operações CRUD mencionadas anteriormente. Para poder invocar os métodos remotos, o cliente deverá primeiro consultar o Serviço de Registo do RMI de modo a obter o *stub* para o objeto remoto. Isso é feito recorrendo ao método estático **Naming.lookup()**, que executa uma pesquisa num Serviço de Registos.

```
public class UsuarioCliente {
  public static void main(String[] args)
  throws NotBoundException, MalformedURLException, RemoteException {
    CrudUsuario crud = (CrudUsuario) Naming.lookup("//localhost/CRUD");
    Scanner sc = new Scanner(System.in);
    List<Usuario> listaUsuario = new LinkedList<>();

    int resp = 1;
    while (resp >= 1 && resp <= 4) {
      mostraMenu();
```

```
      resp = sc.nextInt();
      switch (resp) {
        case 1:
          Retorno ret1 = adicionarUsuario(crud, listaUsuario);
          listaUsuario = ret1.getLista();
          System.out.println(ret1.getMensagem());
          break;
        case 2:
          Retorno ret2 = editarUsuario(crud, listaUsuario);
          listaUsuario = ret2.getLista();
          System.out.println(ret2.getMensagem());
          break;
        case 3:
          Retorno ret3 = removerUsuario(crud, listaUsuario);
          listaUsuario = ret3.getLista();
          System.out.println(ret3.getMensagem());
          break;
        case 4:
          System.out.println(crud.listar(listaUsuario));
          break;
      }
    }
  }
...
```

Referências