Universidade Federal do Ceará

Sistemas Distribuídos - 2019.2

Trabalho 3 – Prática com o Middleware RMI

Objetivo

O objetivo do trabalho é praticar Invocação remota a métodos utilizando o RMI. Ao final do trabalho, o aluno deverá ser capaz de desenvolver aplicações distribuídas utilizando a linguagem Java.

Atividades a serem desenvolvidas:

Os alunos, individualmente ou em dupla, devem fazer e documentar as seguintes questões práticas:

Q1 – Baixe e execute a aplicação RMI disponível em http://www.cdk5.net/wp/extra-material/supplementary-material-for-chapter-5

Q2 – Faça um serviço de sua escolha que utilize RMI. Para isso, siga os passos abaixo:

1. Crie uma interface remota que proporcionará a comunicação entre cliente e servidor (serviço):

```
import java.rmi.*;
public interface InterfaceRemota extends Remote {
metodoRemoto1 throws RemoteException;
...
metodoRemotoN throws RemoteException;
}
```

2. Crie uma classe Servente que o lado servidor implementará de acordo com a interface remota. As classes Serventes dão “corpo” ao serviço fornecido pelo servidor:

```
import java.rmi.*;
public class Servente extends UnicastRemoteObject implements
InterfaceRemota {
public metodoRemoto1()throws RemoteException{
//implementacao
}
}
//...
public metodoRemotoN()throws RemoteException{
//implementacao
}
```

3. Crie uma classe Servidora (que possui um método main()) e publique o serviço:

```
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;
public class Servidor{
   public static void main(String args[]){
      System.setSecurityManager(new SecurityManager());
      try{
         InterfaceRemota refServente = new Servente();
         InterfaceRemota stub = (InterfaceRemota)
         UnicastRemoteObject.exportObject(refServente, 0);
         Naming.rebind("Apelido_do_Servico", refServente);
         System.out.println("Servidor em execucao");
      }catch(Exception e) {
         System.out.println("ShapeList server main " + e.getMessage());
      }
   }
}
```

4. Criação da classe Cliente que obterá a referência remota para o objeto que implementa o serviço:

```
public class ShapeListClient{
   public static void main(String args[]){
      if(System.getSecurityManager() == null){
      System.setSecurityManager(new SecurityManager());
      } else System.out.println("Já há um gerenciador de Seg");
      InterfaceRemota refRemota = null;
      try{
         refRemota =(InterfaceRemota)
Naming.lookup("end/apelido");
         System.out.println("Found server");
         refRemota.metodoRemoto1();
         //...
        refRemota.metodoRemotoN();
}
```

5. Definição da política de segurança. Java é muito restrito no que diz respeito a comunicação, por questões de segurança, e para conectar uma classe a outra remota é necessário o uso de um arquivo policy, que diga à JVM quais os serviços disponíveis e permitidos àquela classe, como apenas conexão ou fazer download de algum arquivo/classe.

```
Ex: sem nenhuma restrição

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

6. Execute o servidor de Nomes (***rmiregistry***)

Considerando que foi criado o pacote: rmi

Compile os fontes com Javac:
Execute dentro do diretorio que possui os fontes (/src/rmi):
javac -d ../../bin/ *.java

Coloque a politica no diretorio dos .class (/bin)

Rode o serviço de nomes
Execute dentro do diretório que possui os .class (/bin):
rmiregistry

7. Execute o servidor.

Execute dentro do diretorio que possui os .class (/bin):
java -Djava.server.rmi.codebaseile:///rmi/ -Djava.security.policy=policy Servidor

8. Rode o cliente

Execute dentro do diretorio que possui os .class (/bin):
java -Djava.security.policy=policy Cliente

Pontuação

Resposta às questões: 7.5 pontos; Apresentação explicando as respostas: 2 pontos.

Data da entrega e apresentação: 15 de Outubro de 2019