Java RMI

Programação Objetos Distribuidos

Objetivos

- Nesta aula iremos apresentar o Java RMI que estende o modelo de objetos Java para dar suporte para objetos distribuídos na linguagem Java. O entendimento da arquitetura e operação do Java RMI são fundamentais para o desenvolvimento de aplicações em ambientes distribuídos.

Plano de Aula

- Java RMI
  - Introdução
  - O que é Java RMI?
  - Arquitetura Java RMI
  - A Operação de Java RMI
  - O pacote java.rmi
  - Um exemplo básico

Introdução

- Em aplicações distribuídas, é comum organizar os processos em: **processo servidor** e **processo cliente**;
  - Estes processos podem estar localizados em **plataformas diferentes**, em **computadores diferentes**;
  - Padrão **CORBA** (*Common Object Request Architecture*):
    - implementação de aplicações distribuídas entre diversas plataformas de sistemas operacionais ou hardware;
  - Java RMI (*Java Remote Method Invocation*)
    - padrão criado especificamente para a linguagem Java.

O que é Java RMI?

- **Java RMI** é um mecanismo para permitir a invocação de métodos que residem em diferentes máquinas virtuais Java (**JVM**);
- O **JVM** pode estar em diferentes máquinas ou podem estar na mesma máquina;
- Em ambos os casos, o método pode ser executado em um endereço diferente do processo de chamada;
- **Java RMI** é um mecanismo de chamada de procedimento remoto orientada a objetos.

• Arquitetura Java RMI

- Em aplicações Java RMI, o **servidor** é usado para **criar objetos remotos** cujos **métodos** são **invocados pelos clientes** da mesma forma como se fossem objetos locais;
- Dessa forma, o **Java RMI** esconde a complexidade inerente à transmissão de dados por *streams* pela internet;
- Um registro de objetos remotos é utilizado é usado para que clientes localizem objetos (***lookup***) e vincular objetos a nomes no lado do servidor (***bind***);

Java RMI

- **A Operação de Java RMI**
  - A operação de RMI está baseada num ambiente de comunicação específico formado por três elementos:
    - O **Servidor**, o **Cliente** e **Servidor de Registro**
  - **Servidor**
    - Cria objetos remotos (que serão invocados através do sistema RMI);
    - Deve especificar uma interface contendo os métodos a serem disponibilizados para acesso remoto e ainda estabelece uma "associação" entre o objeto da classe que implementa esses métodos e um endereço IP e porta TCP locais;
    - Espera por chamadas dos clientes aos métodos dos objetos criados.

Java RMI

- **A Operação de Java RMI**
  - **Cliente**

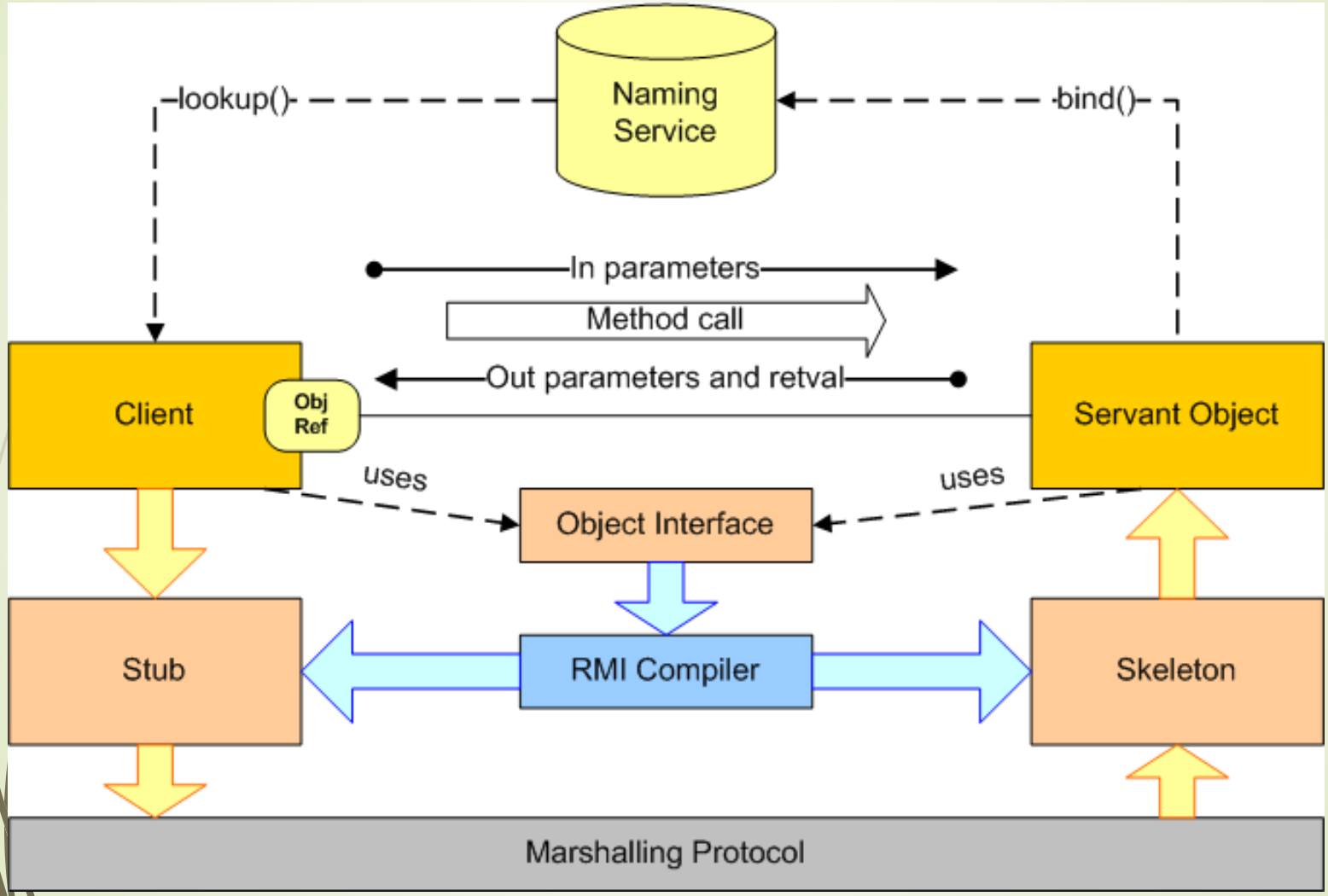
Obtém referências para os objetos remotos disponibilizados pelo servidor;

- Invoca os métodos dos objetos como se estes fossem locais;

- **Servidor de Registro**
  - Funciona de forma parecida com um servidor DNS;
  - Armazena as informações de localização dos servidores RMI que estejam registrados;
  - Os servidores RMI precisam se associar ao servidor de registro;
  - Os clientes RMI precisam consultar esses servidores;
  - Ambos devem saber o endereço do servidor de registros;

Java RMI

- **Arquitetura e Operação de Java RMI**



Java RMI

- **O Pacote java.rmi**
  - O pacote **java.rmi** contém a implementação do Java RMI pertencente ao JDK;
  - Ele disponibiliza **classes**, **interfaces** e **subpacotes** para serem utilizados no desenvolvimento de aplicações com objetos distribuídos; Dentre estes recursos, os principais são:

- ***Remote***
  - É uma interface que indica se um objeto possui métodos que podem ser invocados remotamente por outras JVMs.
- ***MarshalledObject***
  - É uma classe, e o seu construtor recebe um objeto e o converte para um vetor de bytes (*marshalling*). Suporta também a reconstrução do objeto (*unmarshalling*);

Java RMI

- ***Naming***
  - É uma classe que oferece métodos para armazenar e consultar referências a objetos remotos no registro de objetos remotos do Java RMI.
- ***RMISecuretyManager***
  - É uma classe utilizada por aplicações para verificar os requisitos de segurança para acessar classes descarregadas pela internet para serem executadas na máquina local;
- ***java.rmi.registry***
  - É um pacote que oferece suporte ao registro de objetos remotos do Java RMI. Suas principais classes são ***LocateRegistry*** e ***Registry***;
- ***java.rmi.server***
  - É um pacote que oferece classes e interfaces para a implementação do lado servidor do Java RMI. Suas principais classes são ***UnicastRemoteObject***, ***RemoteServer***, ***RemoteObject***, entre outros;

Java RMI

- **A classe *Naming* (**pacote java.rmi)
  - A classe ***Naming*** tem função primordial no controle do registro remoto RMI. Ela é usada para armazenar e consultar referências a objetos remotos em um dado servidor. Seus principais métodos são:
  - static void bind(String name, Object obj): Liga o objeto informado (obj) ao nome informado (name) no registro remoto RMI;
  - static String[] list(String name): Retorna um vetor com os nomes encontrados no registro;
  - static Remote lookup(String name): Retorna uma referência ao objeto remoto que está ligado ao nome informado como parâmetro;
  - static void rebind(String name, Object obj): Liga o objeto informado (obj) ao nome informado (name) no registro remoto RMI, sobrescrevendo a ligação anterior que houver com o mesmo nome;
  - static void unbind(String name): Elimina a ligação que existia do nome informado ao objeto que ele referenciava.

Java RMI

- **A classe *LocateRegistry* (**pacote java.rmi.registry)
  - É responsável por criar o registro RMI ou encontrá-lo em uma máquina virtual remota;
  - Seus principais métodos são:
  - static Registry createRegistry(int port): Cria um registro remoto RMI na máquina local usando a porta indicada;
  - static Registry getRegistry(): Retorna o registro remoto RMI presente na máquina local. Procura na porta padrão 1099;
  - static Registry getRegistry(int port): Retorna o registro remoto RMI presente na máquina local na porta indicada;
  - static Registry getRegistry(String host): Retorna o registro remoto RMI presente na máquina host. Procura na porta padrão 1099;
  - static Registry getRegistry(String host, int port): Retorna o registro remoto RMI presente na máquina host. Procura na porta indicada.

Java RMI

- **A classe *Registry* (**pacote java.rmi.registry**)**
  - Tem a função de manipular o registro remoto RMI, ligando ou desligando objetos remotos a nomes;
  - Objetos da classe *Registry* podem ser obtidos quando realizadas consultas ao registro pela classe *LocateRegistry*;
  - Os principais métodos de *Registry* são:
  - void bind(String nome, Remote object): Liga o nome indicado (name) ao objeto remoto informado (object);
  - String[] list(): Retorna um vetor com todos os nomes presentes nesse registro;
  - Remote lookup(String name): Consulta pelo objeto que esteja ligado ao nome indicado no registro;
  - void rebind(String nome, Remote object): Sobrescreve a ligação do nome indicado, vinculando-o ao objeto remoto informado;
  - void unbind(String nome): Elimina a ligação do nome indicado ao objeto que ele estava ligado no registro.

Java RMI

- **A interface *Remote***
  - Disponibilizada no pacote **java.rmi**;
  - É usada para indicar objetos que contém métodos que são acessíveis **remotamente** por JVMs em clientes remotos;
  - Para definir os métodos remotos é necessário que seja criada uma interface que contém a assinatura dos mesmos. Tal interface deve ser filha de ***Remote***;
  - Para se criar um **objeto no servidor** com métodos acessíveis remotamente, a classe desse objeto deve implementar uma **interface filha de *Remote***;
  - Se um método de um objeto remoto retornar um objeto, este objeto deve estender a interface ***java.io.Serializable***.

Java RMI

- **A classe *UnicastRemoteObject***
  - Para criar uma classe que seja acessível remotamente, pode-se fazer com que ela estenda ***UnicastRemoteObject*** (presente no pacote java.rmi.server);
  - A classe remota deve implementar uma **interface filha de *Remote***;
  - Dessa forma, definindo precisamente as assinaturas dos métodos remotos que são utilizadas tanto no lado do cliente como no lado do servidor.

Java RMI

- **Um exemplo básico**
  - Vamos apresentar uma aplicação exemplo que demonstra a utilização dos principais recursos do **pacote java.rmi**. O código fonte se encontra no Moddle.

Java RMI

- **Considerações sobre a Implementação**
  - Há um grande número de exceções a serem gerenciadas. Algumas:
  - ***RemoteException***: É a superclasse comum a um grande número de exceções que podem ocorrer durante a invocação de um método remoto;
  - ***AlreadyBoundException***: É uma exceção que ocorre quando se tenta registrar um nome no registro de objetos remotos, porém esse nome já se encontra registrado;
  - ***MarshallException***: É uma exceção que é lançada quando ocorre erro na operação de *marshalling*;
  - ***UnmarshallException***: É uma exceção que é lançada quando ocorre erro na operação de *unmarshalling*;
  - Em versões mais antigas do Java era necessário usar o compilador **rmic** para gerar as classes ***stubs*** e ***skeletons***. Após a versão 5 do Java, não é mais necessário usar o comando **rmic**.

Referências

- Sistemas Distribuídos - Conceitos e Projeto, George Coulouris, 4ª Edição - Editora Bookman, 784 páginas.
- http://www.devmedia.com.br/remote-method-invocation-rmi-na-pratica/31180.