Remote Method Invocation

Introdução

RMI é uma das abordagens da tecnologia Java para prover as funcionalidades de uma plataforma de objetos distribuídos. Esse sistema de objetos distribuídos faz parte do núcleo básico de Java desde a versão JDK 1.1, com sua API sendo especificada através do pacote java.rmi.

Processos Distribuídos

“Um sistema distribuído é uma coleção de computadores independentes que aparecem para os usuários do sistema como um único computador.” (Tanenbaum)

“Consiste de uma coleção de computadores autônomos ligados através de uma rede e equipados com software de sistemas distribuídos” (Coulouris)

História

Com a evolução das redes de computadores surgiram as aplicações distribuídas. No início, o processamento era realizado somente no servidor e as aplicações eram inicializadas por um cliente que estava interligado a ele por meio da rede. Mais tarde com o surgimento da programação orientada a objetos surgiu os processos distribuídos derivados das linguagens C e JAVA.

Introdução

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. Através do RMI é possível realizar a comunicação entre objetos que estão em diferentes Máquinas Virtuais Java(JVM). Sendo possível inclusive transferir objetos serializados através da rede.

Uma aplicação RMI é frequentemente composta por dois programas diferentes, um servidor e um cliente.

Cliente / Servidor

O servidor cria objetos remotos e faz referências a esses objetos disponíveis. Em seguida, ele é válido para clientes invocarem seus métodos sobre os objetos. O RMI é uma evolução da arquitetura cliente/servidor.

Um cliente é um computador que efetua requisições por serviços, e um servidor é um componente que "entrega" os serviços solicitados.

O RMI mantém a noção, mas a abordagem é mais flexível, ou seja, um componente RMI pode agir tanto como um cliente quanto como um servidor.

O funcionamento do RMI se baseia em objetos remotos, ou seja, objetos que são acessíveis a partir de um servidor remoto. Para que um objeto possa ser invocado remotamente, ele deve ser definido em uma interface Java acessível tanto ao servidor quanto ao cliente.

A aplicação de objetos distribuídos tem de prover as seguintes propriedades:

- **Localização de objetos remotos:** O sistema tem de obter referências a objetos remotos. Isto pode ser feito de duas maneiras. Ou, usando as instalações de nomeação do RMI, o registro RMI, ou passando e retornando objetos remotos.

- **Comunicação com objetos remotos:** O desenvolvedor não tem de lidar com a comunicação entre os objetos remotos desde que este é tratado pelo sistema RMI.

Funcionamento

Para que essa comunicação se realize o RMI fornece objetos “auxiliares” que facilitam a comunicação entre cliente e servidor, liberando o desenvolvedor de se preocupar com todo o código envolvido nessa troca de informações.

Do lado do cliente existe um auxiliar que captura a chamada realizada pelo cliente e envia essa chamada para o servidor, esse auxiliar é chamado de Stub. O servidor por sua vez, também possui um auxiliar, chamado Skeleton, que captura a chamada passada por um Stub e chama o método no servidor.

A principal função de RMI é permitir que métodos de objetos remotos fossem chamados de forma tão transparente como se os objetos estivessem localmente. Toda classe que deverá ter métodos remotamente chamados por RMI, deverá ter um Stub

Com esse esquema de objetos “auxiliares” o RMI realiza a chama a objetos remotos simulando-os na máquina do cliente. Outro aspecto importante no RMI é a interface remota.

Existe entre cliente e servidor uma interface remota de métodos. Essa interface permite que o cliente conheça as funções pré-estabelecidas que ele tenha acesso no servidor.

Essa interface define essas funções junto com seus parâmetros e retornos. Caso ocorra algo de errado na comunicação é lançada uma exceção.

Tanto o Strub, do lado do cliente, quanto o Skeleton, do lado do servidor, é criado a partir dessa interface remota.

O registro RMI é similar a uma página de cadastro telefônico, ela possui um nome para o serviço e o local de sua implementação no servidor. Esse registro RMI deve ser sempre iniciado pelo servidor e chamado pelo cliente quando deseja utilizar um dos métodos da interface remota.

Configurações

Principais configurações.

Interface de objeto de servidor: Uma sub-interface de java.rmi.Remote que define os métodos para o objeto de servidor.

Implementação de servidor: Uma classe que implementa a interface de objeto remoto.

Objeto de servidor: Uma instância da implementação de servidor.

Programa cliente: Um programa que invoca os métodos no objeto de servidor remoto.

Stub de servidor: Um objeto que reside na máquina cliente e serve como um substituto para o objeto de servidor remoto.

Esqueleto de servidor: Um objeto que reside na máquina servidor e comunica com o stub e o objeto de servidor real.

O funcionamento do RMI se dá da seguinte forma:

- Um objeto de servidor é registrado com o RMI registry;
- Um cliente pesquisa através do RMI registry um objeto remoto;
- Se o objeto remoto for localizado, seu stub é retornado para o cliente.

O objeto remoto pode ser usado da mesma forma que um objeto local. A comunicação entre o cliente e o servidor é gerenciada por meio do stub e do esqueleto.

Segurança

O RMI fornece o "RMISecurityManager", sem o qual objetos locais se tornam incapazes de baixar código ou utilizar métodos remotos.

No modelo de segurança do JDK 1.6, o código, independentemente se é local ou remoto, é sempre executado sob uma política de segurança, que define um conjunto de permissões disponíveis para o código.

Cada permissão define um acesso a um recurso em particular, como por exemplo, permissão de leitura ou escrita a um diretório específico ou permissão de conexão para um host e porta.

O sistema de execução organiza o código em domínios individuais, cada domínio possui seu conjunto de classes, sendo que as instâncias dessas classes possuem o mesmo conjunto de permissões.

Um domínio pode ser configurado de forma que applets rodem em um ambiente restrito, de acordo com o administrador da rede.

As aplicações "stand alone" rodam sem nenhuma restrição, porém elas podem ser submetidas a uma política de segurança.

Vantagens

Com RMI é possível trabalhar com interfaces e passar objetos que implementam essas interfaces em tempo de execução. Dessa forma é possível “mover o comportamento” para o outro lado da comunicação.

Padrões de Projeto: O fato de ser possível passar objetos serializados do cliente para o servidor garante o uso de todo o poder do paradigma de orientação a objetos na computação distribuída. Como é possível “mover o comportamento” com o uso de interfaces Java é possível usar padrões de projeto orientado a objetos facilmente.

Facilidade para Usar: RMI possui uma estrutura simples .

Portabilidade: Essa característica é herdada pelo fato de RMI ser 100% Java.

Se utilizado com JNI(Java Native Interface) é possível construir aplicativos Java que se comuniquem com outro sistemas já existentes.

Programação Paralela: RMI é multi-thread, garantindo que as requisições dos clientes seja atendidas sem necessidade código extra.