Remote Method Invocation (RMI)

Prof. Marcelo Veiga Neves
marcelo.neves@pucrs.br

Middlewares para Comunicação

- Sun RPC
- Objetos Distribuídos
  - JAVA RMI
  - CORBA (a language-independent standard)
  - JavaSpaces
  - PyRO (python)
- Webservices
- Etc.

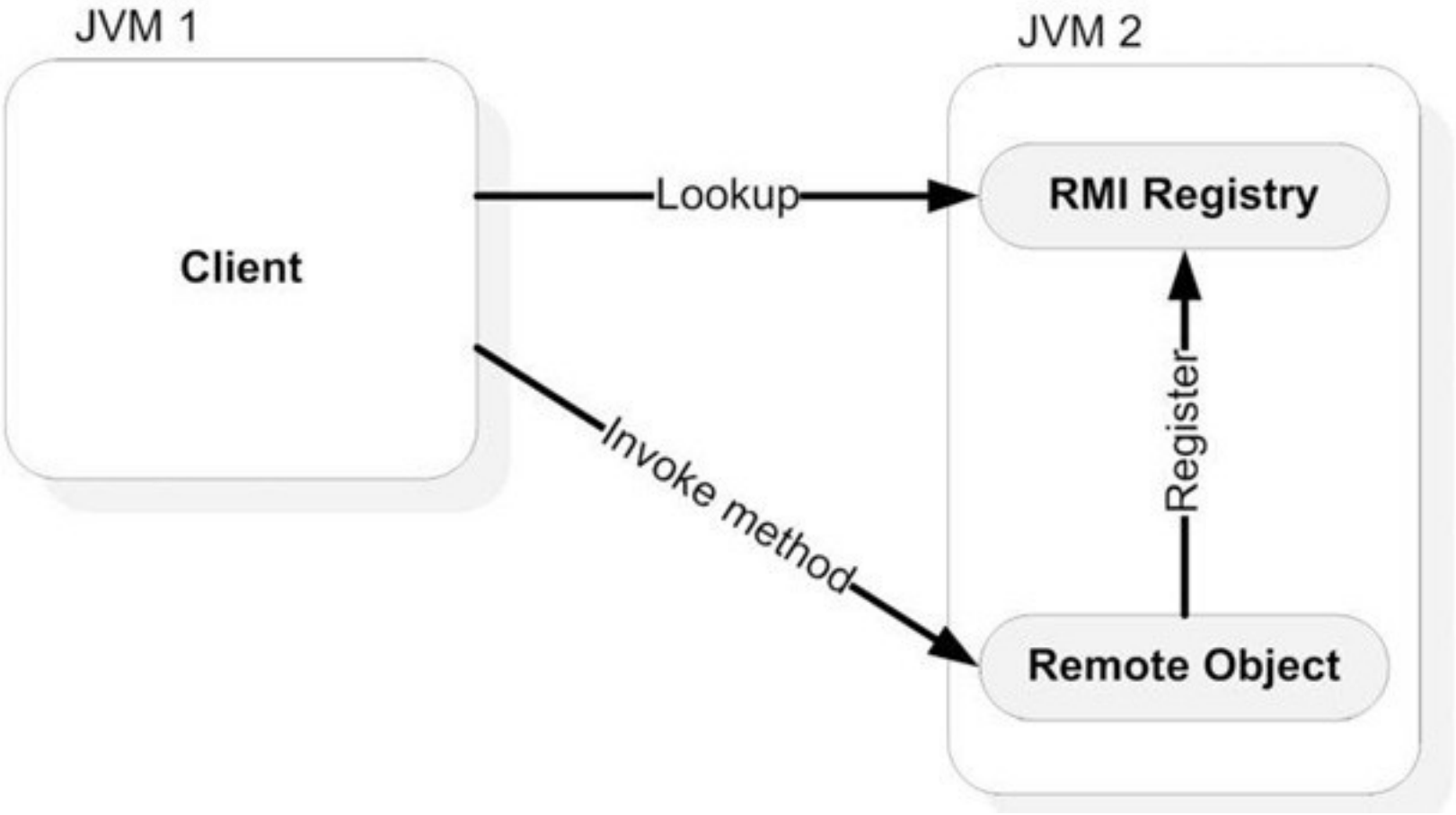
Remote Method Invocation (RMI)

- Permite que se invoque um método em um objeto remoto
- Estende o modelo de objetos Java para prover suporte a **objetos distribuídos**
- Invoca-se um método remoto usando a mesma sintaxe de uma invocação local

Remote Method Invocation (RMI)

- RMI = RPC orientado a objetos
- Por ser um sistema de uma única linguagem, é simples
  - Não precisa de uma linguagem de intermediária como no caso o RPC
- Modelo de sincronismo:
  - o programador de um objeto remoto deve considerar o seu comportamento em um ambiente concorrente
- Processos envolvidos:
  - Cliente
  - Servidor (remote object)
  - Registrador de objetos (rmiregistry)

Remote Method Invocation (RMI)



Fonte: https://goo.gl/Qnd3cF

Remote Method Invocation (RMI)

- Há dois tipos de classes que podem ser usados em Java RMI:
  - Classes que estendem **Remote** podem ter seus métodos invocados remotamente
  - Classes que implementam a interface **Serializable** podem ser passadas como parâmetros e recebidas em invocações remotas

Classes Remotas e Interfaces

- Uma classe remota tem duas partes: a interface e a classe propriamente dita (implementação da interface)
- A interface:
  - Deve ser pública
  - Deve estender java.rmi.Remote
  - Métodos da interface devem lançar java.rmi.RemoteException
- A classe remota:
  - Implementa a interface remota
  - Estende a classe java.rmi.server.UnicastRemoteObject
  - Pode ter métodos que não fazem parte da interface

Classes Remotas e Interfaces

- O cliente precisa apenas da definição da interface
- O servidor precisa tanto da interface quando da classe remota
- Uma aplicação típica, pode ser, portanto, formada pelos seguintes componentes:
  - Interface remota
  - Classe remota (implementação da interface)
  - Classe servidora (instancia um objeto remoto e registra este objeto remoto via Naming.rebind())
  - Classe cliente (obtém uma referência ao objeto remoto via Naming.lookup() e chama os métodos do objeto remoto)

Exemplo

- HelloInterface.java
- HelloImpl.java
- HelloServer.java
- HelloClient.java

HelloInterface.java

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

public interface HelloInterface extends Remote {
   public String say() throws RemoteException;
}
```

HelloImpl.java

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

public class HelloImpl extends UnicastRemoteObject implements HelloInterface {
   private static final long serialVersionUID = 789679589892878284 6L;
   private String message;

   public HelloImpl (String msg) throws RemoteException {
      message = msg;
   }

   public String say() throws RemoteException {
      return message;
   }
}
```

HelloServer.java

```
import java.rmi.Naming;
import java.rmi.RemoteException;

class HelloServer {
   public static void main (String[] argv) {
      try {
         java.rmi.registry.LocateRegistry.createRegistry(1099);
         System.out.println("RMI registry ready.");
      } catch (RemoteException e) {
         System.out.println("RMI registry already running.");
      }
      try {
         Naming.rebind ("Hello", new HelloImpl ("Hello, world!"));
         System.out.println ("HelloServer is ready.");
      } catch (Exception e) {
         System.out.println ("HelloServer failed:");
         e.printStackTrace();
      }
   }
}
```

HelloClient.java

```
import java.rmi.Naming;

class HelloClient {
   public static void main (String[] argv) {
      try {
         HelloInterface hello = (HelloInterface) Naming.lookup ("//localhost/Hello");
         System.out.println (hello.say());
      } catch (Exception e) {
         System.out.println ("HelloClient failed:");
         e.printStackTrace();
      }
   }
}
```

Referências

- Material baseado em slides dos Profs. Roland Teodorowitsch, Avelino Zorzo, Celso Costa, Fernando Dotti e Luiz Gustavo Fernandes
- COULOURIS, George F.; DOLLIMORE, Jean; KINDBERG, Tim. Distributed systems: concepts and design. 4. ed. Harlow (England): Addison Wesley, 2005. 927 p.
- BACLAWSKI, Kenneth. Java RMI Tutorial. Boston: [s.n.], 1998. Disponível em: http://www.airchennai.org/data/1/circular/www-eg-bucknell-edu.pdf