RMI: Remote Method Invocation

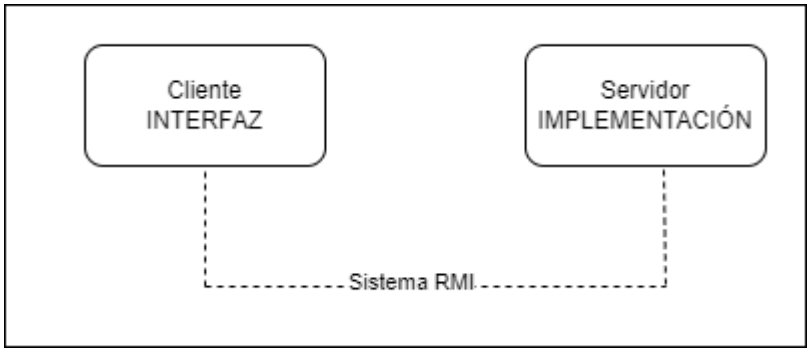
- Implementación de java del Remote Procedure Call
- Invocación aun método que puede estar en otra máquina
- Los datos se pasan como argumentos del método.7

Diferencias con la programación local

- Definicion de objetos
  - Locales: definidos por una clase
  - Remotos: Comportamiento definido por una interface que extiende java.rmi.remote
- Implementación de ojetos
  - Locales: implementados por su clase
  - Remotos: comportamiento ejecutado por una clase que implementa la interfaz
- Creación de objetos
  - Locales: Instancias se crean con new
  - Remotos: instancias creadas en el servidor con un new, el cliente no puede crear objetos
- Acceso a objetos:
  - Locales: a través de referencia local
  - Remoto: referencia a un stub que implementa la interfaz remota y que hace proxy
- Referencias:
  - Locales: una referencia a un objeto apunta directamente al objeto en memoria
  - Remotos: referencia al proxy que sabe como conectar con el objeto remoto
- Referencias activas:
  - Locales: si es apuntada por al menos un objeto
  - Remoto: activo mientras sus servicios sean solicitados durante un período de tiempo
- Recolector de basura:
  - Locales: Recogido si no está activo
  - Remotos: Recogido si no hay referencias locales ni está siendo utilizado por clientes externos
- Excepciones:
  - Locales: RunTimeExceptions Y Exceptions
  - Remotos: RemoteException

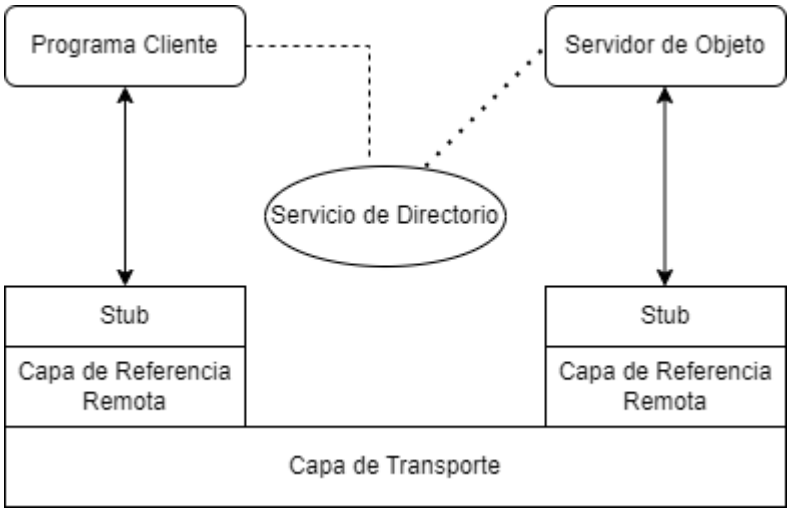
Arquitectura

Tenemos dos conceptos principales: Definición de comportamiento e implementación de comportamiento



Tres capas

- Stub: captura las invocaciones del cliente y las redirige al objeto completo
- capa Referencias Remotas: Sabe como gestionar las referencias de los clientes a objetos remotos
- Capa de transporte: Permite la conectividad entre objetos basada en TCP/IP.



Servicio de Nombres

- JNDI (Java Naming Directory Interface)
  - Busca los objetos de java remotos por su nombre
  - No se limita solo a RMI
- RMIRegistry
  - El servidor registra un objeto y lo hace accesible a los clientes
  - Corre en máquinas donde estén los servidores
  - Puerto por defecto: 1099

Como Buscar

- El cliente debe buscar el servicio:
  - java.rmi.Naming.lookup(RMIUrl)
- Una URL en versión RMI sería:

```
rmi://<host_name>[:puerto del servicio de directorio]/<nombre del servicio>
```

Distribución de clases

- Servidor:
  - Interface
  - Implementación de la Interface
  - Stubs
- Cliente:
  - Interface
  - Stubs

Implementación Servidor

En este ejemplo implementaremos una calculadora en un servidor que será utilizada por un cliente.

Interfaz Servidor

La interfaz debe heredar de java.rmi.Remote

InterfazSertvidor.jar

```
public interface InterfazServidor extends java.rmi.Remote{
public long sumar(long a, long b)
  throws java.rm.RemoteException;

public long restar (long a, long b)
  throws java.rm.RemoteException;

public long multiplicar ()
  throws java.rmi.RemoteException;

public long dividir()
  throws java.rmi.RemoteException;

}
```

Implementación Servidor

ImplementacionServidor.jar

```
public class ImplementacionServidor extends UnicastRemoteObject
implements Calculator{
  private static final long serialVersionUID = 1L;

  public implementacion() throws RemoteException{
```

```
    super();
  }

  public long sumar(long a, long b) throws RemoteException{
    return a+b;
  }

  public long restar(long a, long b) throws RemoteException{
    return a-b;
  }

  public long multiplicar (long a, long b) throws RemoteException{
    return a*b;
  }

  public long dividir (long a, long b) throws RemoteExcepcion{
    return a/b;
  }

}
```

Implementación Cliente

From:
https://knoppia.net/ - **Knoppia**

Permanent link:
https://knoppia.net/doku.php?id=dad:rmi&rev=1701338337

Last update: **2023/11/30 09:58**