

# RMI

## *Lenguaje de Programación 2*



PONTIFICIA  
**UNIVERSIDAD**  
**CATÓLICA**  
DEL PERÚ

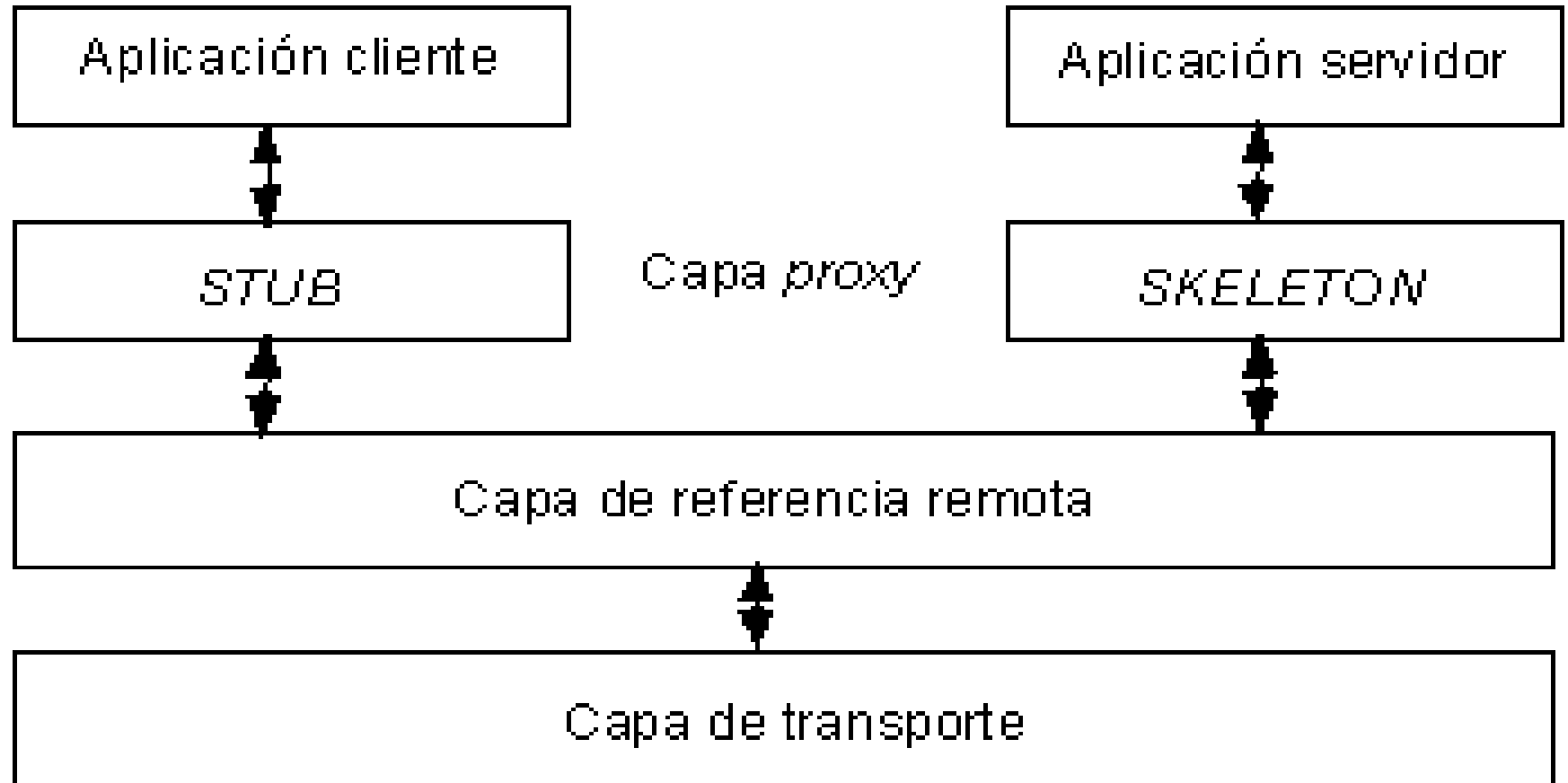
*Dr. Freddy Paz*

# Concepto de RMI

---

- ① RMI (**Remote Method Invocation**) es un mecanismo que permite realizar llamadas a métodos de objetos remotos situados en distintas (o la misma) máquinas virtuales de Java, compartiendo así recursos y carga de procesamiento a través de varios sistemas.

# Arquitectura de RMI



# Capa de Aplicación

---



- **Implementación real de las aplicaciones cliente y servidor.**
- **Llamadas a alto nivel para acceder y exportar objetos remotos.**
- **Se declaran métodos en una interfaz que herede de `java.rmi.Remote`.**
- **Una vez que los métodos han sido implementados, el objeto debe ser exportado.**
  - **De forma implícita: si el objeto hereda de la clase `UnicastRemoteObject` (paquete `java.rmi.server`)**
  - **De forma explícita: con una llamada al método `exportObject()` del mismo paquete.**

# Capa Proxy o Capa Stub-Skeleton

---



- **Capa que interactúa directamente con la capa de aplicación.**
- **En esta capa se realizan las llamadas a objetos remotos y acciones junto con sus parámetros y retorno de objetos.**

# Capa de Referencia Remota

---



- **Responsable del manejo de la parte semántica de las invocaciones remotas.**
- **Asimismo es responsable de la gestión de la replicación de objetos y realización de tareas específicas de la implementación con los objetos remotos**
- **Se establecen las persistencias semánticas y estrategias adecuadas para la recuperación de conexiones perdidas.**

# Capa de Transporte

---



- Es la responsable de realizar las conexiones necesarias y manejo del transporte de los datos de una máquina a otra.
- El protocolo de transporte es JRMP (Java Remote Method Protocol), que solamente es interpretado por programas escritos en Java.

# Crear una aplicación con RMI

---



**Toda aplicación RMI normalmente se descompone en 2 partes:**

- **Un servidor, que crea algunos objetos remotos, crea referencias para hacerlos accesibles, y espera a que el cliente los invoque.**
- **Un cliente, que obtiene una referencia a objetos remotos en el servidor, y los invoca.**




# Crear una aplicación con RMI

---

 Se deben seguir los siguientes pasos:

1. Diseñar e implementar los componentes de la aplicación distribuida.
2. Compilar los fuentes y generar Stubs.
3. Hacer las clases accesibles a la red.
4. Iniciar la aplicación.

# Diseñar e Implementar los Componentes de la App. Distribuida

- 
- Se decide la arquitectura de la aplicación y se determina qué componentes son objetos locales y cuales deberían ser accesibles remotamente.
- Este paso incluye:
    - Definir las interfaces remotas.
    - Implementar los objetos remotos.
    - Implementar los clientes.

# Compilar los Fuentes

---



- Se utiliza el compilador `javac` para compilar los ficheros fuentes de Java que contienen las implementaciones de las interfaces remotas, las clases del servidor y del cliente.
- Se utiliza el compilador `rmic` para crear los Stubs de los objetos remotos. RMI utiliza una clase `Stub` del objeto remoto como un proxy en el cliente para que los clientes puedan comunicarse con un objeto remoto particular.

# Hacer accesibles las clases en la red

---

- En este paso, se tiene que hacer que todo (los ficheros de clases Java asociados con las interfaces remotas, los Stubs y otras clases que se necesiten descargar en los clientes) sean accesibles a través de un servidor web.

# Iniciar la aplicación

---

- **Iniciar la aplicación incluye ejecutar el registro de objetos remotos de RMI, el servidor y el cliente.**

# Definir la interfaz remota

---

- La interfaz debe ser pública.
- Debe heredar de la interfaz **java.rmi.Remote**, para indicar que puede llamarse desde cualquier Máquina Virtual Java.
- Cada método remoto debe lanzar la excepción **java.rmi.RemoteException**, además de las excepciones que pueda manejar.

# Definir la interfaz remota

```
public interface InterfazRemota
extends java.rmi.Remote {

    public void metodo1()
    throws java.rmi.RemoteException;

    public String metodo2(String nombre)
    throws java.rmi.RemoteException;
}
```

# Definir la clase que implementa la interfaz

```
public class ClaseRemota
    extends java.rmi.server.UnicastRemoteObject implements InterfazRemota{
    public ClaseRemota() throws java.rmi.RemoteException {
    }

    public void metodo1() throws java.rmi.RemoteException{
        System.out.println("Método 1");
    }

    public String metodo2(String nombre) throws java.rmi.RemoteException{
        System.out.println("Método 2");
        return "Hola " + nombre;
    }
    public void metodo3(){
        System.out.println("Este metodo no puede llamarse remotamente");
    }
}
```



# Definir la clase del Servidor

```
public class Principal{  
    public static void main(String[] args){  
        try{  
System.setProperty("java.rmi.server.hostname","127.0.0.1");  
            String puerto = "1234";  
            LocateRegistry.createRegistry(1234);  
            InterfazRemota ir = new ClaseRemota();  
            java.rmi.Naming.rebind("//"+java.net.InetAddress.getLocalH  
ost().getHostAddress()+ ":" + puerto + "/rmi", ir);  
        } catch (Exception e){  
            System.out.println(e.getMessage());  
        }  
    }  
}
```

# Definir la clase del Cliente

```
public class ClienteRMI {  
    public static void    main(String[] args) {  
        try {  
System.setProperty("java.rmi.server.hostname","127.0.0.1");  
        InterfazRemota ir = (InterfazRemota)  
java.rmi.Naming.lookup("//" + args[0] + ":" + args[1] + "/rmi");  
        //Se invoca al método 1 y al método 2  
        ir.metodo1();  
        System.out.println(ir.metodo2("Karla Perez"));  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
    }  
}
```

# Referencias

---

- Burns, B. (2018). Designing Distributed Systems: Patterns and Paradigms for Scalable, Reliable Services. Estados Unidos: O'Reilly Media.
- Downing, T. (1998). Java RMI. Estados Unidos: Hungry Minds Inc.
- Grosso, W. (2001). Java RMI. Estados Unidos: O'Reilly Media
- <https://docs.oracle.com/javase/tutorial/rmi/index.html>
- Universidad de Alicante – España :Arquitectura RMI.  
<http://www.jtech.ua.es/j2ee/2002-2003/modulos/rmi/apuntes/>