Aplicaciones Distribuidas con Java Remote Method Invocation

Henry Tenorio Guzmán

A76456

Agenda

- Introducción
- Conceptos Importantes
- ¿Qué es Java RMI?
- Objetivos de Java RMI
- Características
- ¿Cómo se utiliza?
- Conclusiones

Introducción

- La computación distribuida se ha convertido en un aspecto fundamental en los negocios y la industria a nivel mundial.

Introducción (2)

- La Programación Orientada a Objetos (POO) es utilizada ampliamente en la computación distribuida.

Introducción (3)

- *Remote Method Invocation* (RMI) de Java es un modelo de objetos distribuidos para desarrollar aplicaciones complejas y robustas.

Conceptos Importantes

- **Aplicación distribuida**
  - Las funcionalidades de la aplicación han sido separadas.
  - Distribución de las unidades funcionales.
  - Comunicación.

Conceptos Importantes (2)

- **Objetivos de una aplicación distribuida**
  - Rendimiento y bajo costo.
  - Tolerancia a fallos.
  - Funcionalidad.

Conceptos Importantes (3)

- **Programación orientada a objetos**
  - La programación orientada a objetos se basa en el concepto de objeto para establecer la estructura modular de los programas.

Conceptos Importantes (4)

- **Características de la POO**
  - Modularización.
  - Ocultamiento.
  - Abstracción
  - Herencia y polimorfismo.

Conceptos Importantes (5)

- **Serialización**
  - Consiste en convertir un objeto en un *stream* de bytes para ser trasmitido por una red.
  - En **Java** los tipos primitivos son serializables por defecto.

Conceptos Importantes (6)

- **Si se trata de un objeto**
  1. La clase debe implementar la interfaz *Serializable*.
  2. Se debe generar un *serialVersionUID*.
  3. Los atributos del objeto que se desea serializar deben ser de tipo primitivo o de tipo Serializable.
  4. Asegurarse que la superclase del objeto a serializar es una clase serializada.
  5. Redefinir los métodos *equals()* y *hashcode()*.

¿Qué es Java RMI?

- Java RMI es una extensión al modelo de objetos de Java para soportar objetos distribuidos.

Objetivos de Java RMI

- Transparencia.
- Orientado a objetos.
- Facilidad.
- Mantenimiento.
- Seguridad.
- Portabilidad.
- Robustez.
- Versatilidad.

Características

- **Concurrencia**
  - Para cada cliente que trate de acceder a un objeto remoto, el servidor creará un nuevo hilo que se encargará de darle servicio.

Características (2)

- **Nombrado de objetos**
  - Utiliza la notación URL.

    Por Ejemplo: rmi://localhost:8080/miObjeto.

  - Adicionalmente se cuenta con el servidor de nombres *rmiRegistry*.

Características (3)

- **Paso de parámetros**
  - La Serialización se encarga de informar al compilador y al entorno de ejecución de Java que deberá pasar por valor copias de los objetos de este tipo desde la JVM local a la JVM remota.

Características (4)

- **En una invocación a un método de un objeto remoto puede contar con los siguientes parámetros**
  - Primitivos.
  - Serializados.
  - Objetos remotos.

Características (5)

- **Recolector de basura**
  - En los sistemas distribuidos, las referencias a los objetos son más complejas y de mayor tamaño que en un entorno local.
  - Una referencia a un objeto remoto indica la localización del objeto, datos sobre el tipo del objeto e información de seguridad.

¿Cómo se utiliza?

- **Para el *Servidor***
  - Definir la interfaz remota.

```
public interface MiInterfazRemota extends Remote{
    public void miMetodo1() throws RemoteException;
    public String miMetodo2() throws RemoteException;
}
```

¿Cómo se utiliza? (2)

- Implementar la interfaz remota.

```
public class MiObjetoRemoto extends UnicastRemoteObject
                            implements MiInterfazRemota{

    public MiObjetoRemoto() RemoteException{
        // Código del constructor.
    }

    public void miMetodo1() RemoteException{
        //Implementación del método1.
        System.out.println("Hola Mundo");
    }

    public String miMetodo2() RemoteException{
        //Implementación del método2.
        return "Con Java RMI";
    }

    public int miOtroMetodo(){
        return 0;
    }
}
```

¿Cómo se utiliza? (3)

- Crear el *main* del servidor.

```
public class main{
    public static voin main(String arg[]){
        try{
            MiInterfazRemota miInterfaz= new MiObjetoRemoto();
            java.rmi.Naming.rebind("//"+java.net.InetAddress.getLocalHost().getHostAddress()+
                                   ":" +args[0]+"/Expo",mir);
        }catch(Exception e){
            //Se captura la excepción.
        }

    }
}
```

¿Cómo se utiliza? (4)

- Posteriormente
  - Compilamos la interfaz remota.
  - Compilamos las clases que implementan las interfaces para generar los *Stub* y *Skeleton* para mantener la referencia con el objeto remoto.
  - Arrancar el *rmiRegistry*.
  - Lanzar el servidor.

¿Cómo se utiliza? (5)

- **Para el cliente**
  - Creamos, compilamos y ejecutamos la clase Cliente.

```
public class Cliente{
    public static void main(String args[]){
        try{
            MiInterfazRemota miInterfaz = (MiInterfazRemota)java.rmi.Naming.lookup("//" +
                                          args[0] + ":" + args[1] + "/Expo");

            miInterfaz.miMetodo1();
            System.out.println();
            miInterfaz.miMetodo2();
           catch (Exception e)
           {
             e.printStackTrace();
           }
        }
    }
}
```

Conclusiones

- Las aplicaciones distribuidas son muy importantes hoy en día.
- La programación orientada a objetos provee facilidades para la solución de problemas.
- El modelo RMI de Java resulta una buena opción para la creación de aplicaciones distribuidas.

Preguntas

Referencias

[1] Santi Caballé y Fatos Xhafa, 2008. *Aplicaciones distribuidas en java con tecnología RMI*. Delta Publicaciones.

[2] N. Narasimhan, L. E. Moser y P.M. Melliar-Smith, 2000. *Transparent consistent replication of Java RMI objects*. IEEE Computer Society Washington DC, USA.

[3] Keshk, A.E, 2008. *Implementation of Distributed Application using RMI Java threads*. IEEE Computer Society Washington DC, USA.

[4] Mariana Sharp y Atanas Rountev, 2006. *Static analysis of object references in RMI-based Java software*. IEEE Computer Society San Francisco, CA, USA.

Referencias (2)

[5] Rolando Menchada Méndez y Félix García Carballeira, 2001. *Java RMI*. Revista Digital Universitaria. Tomado de: http://www.revista.unam.mx/vol.2/num1/art3/, el día 20 de enero del 2012.

[6] Sun Microsystems. RMI Specification. 2002.

[7] Sun Microsystems. Serialization Specification. 2003.

[8] David Basanta Gutiérrez y Lourdes Tajes Martínez, 1999. *Tecnologías para el desarrollo de Sistemas Distribuidos: Java versus Corba*. X Jornadas de Paralelismo, La Manga del Mar Menor – Murcia.