Práctica RMI

Asignatura:`Arquitectura Software`
Dpto. de Informática e Ingeniería de Sistemas
Universidad de Zaragoza
`jmerse@unizar.es`

1. Objetivo

Desarrollar un programa C/S sencillo haciendo uso de un middleware. El middleware RMI (Remote Method Invocation) [1] es una implementación de un broker de objetos. Por tanto, vamos a conocer un mecanismo de invocación remota de métodos dentro del paradigma de la orientación a objetos.

2. Fecha límite de entrega de la práctica

Semana del 20 al 24 de abril de 2020 (pero siempre antes de tu sesión de revisión del trabajo con el profesor, para así poder comentar la práctica en esa sesión con el profesor).

3. Introducción a RMI

Una aplicación básica RMI (Invocación Remota de Métodos) está formada por dos programas, el cliente y el servidor. El programa servidor crea objetos remotos, construye referencias para acceder a estos objetos, y espera que los programas cliente invoquen métodos de estos objetos remotos. El programa cliente, por su parte, debe obtener la referencia al objeto remoto e invocar del mismo los métodos que desee. RMI proporciona el mecanismo por el cual el programa cliente y el programa servidor se comunican. Este tipo de aplicaciones se llaman aplicaciones distribuidas basadas en objetos y necesitan lo siguiente:

- **Localizar objetos remotos**. Existen varios mecanismos para que el programa cliente obtenga referencias a objetos remotos. Por ejemplo, registrar los objetos remotos en el RMI "registry"(registro de RMI). De manera alternativa se pueden pasar las referencias a los objetos remotos como parte de otra invocación remota.
- **Comunicación con objetos remotos**. RMI gestiona los detalles de la comunicación con los objetos remotos. Para el programador la comunicación es parecida a la invocación de métodos en Java.

4. Creación de una aplicación RMI

A continuación ilustramos los pasos a seguir para crear una aplicación RMI con la descripción del código que debe implementarse para esta práctica.

Pasos:

1. Escribir los programas en Java.
2. Poner en marcha los programas en el host servidor.
3. Poner en marcha los programas en el host cliente.

4.1. Escribir los programas en Java

En el ejemplo hay que escribir cuatro ficheros fuente. La mayor parte del código aparece ya escrito, tenéis que añadir código allí donde pone "TODO:":

1. La interface remota Java (Collection.java).
2. Clase que implementa la interface remota (CollectionImpl.java).
3. El programa cliente que invoca de manera remota los métodos del servidor (Cliente.java).
4. El fichero de seguridad (java.policy).

4.1.1. Interfaz remota

La invocación de métodos remotos puede dar lugar a errores que no se presentan en el caso de invocaciones locales debido a problemas de comunicación relacionados con la red y problemas en el servidor. Para indicar que un objeto es remoto, este implementará una interface remota, que tiene las siguientes características:

- La interface remota debe ser pública. De otro modo se produciría un error cuando el cliente tratara de cargar un objeto remoto que implementa la interface remota.
- La interface remota debe descender de java.rmi.Remote
- Cada método debe declarar una excepción java.rmi.RemoteException en su cláusula throws, además de excepciones específicas del método si las tuviera.
- Un objeto remoto pasado como argumento o valor de retorno debe ser declarado como interface remota, no como clase de implementación (objeto remoto).

A continuación se muestra la definición del interface del ejemplo, esta interface define métodos para gestionar una colección de libros.

```
import java.rmi.Remote;
import java.rmi.RemoteException;
//TODO: otros imports
```

```
public interface Collection extends Remote
{
//metodos de la interface
 int number_of_books() throws RemoteException;
 String name_of_collection() throws RemoteException;
 void name_of_collection(String  _new_value) throws RemoteException;
}
```

4.1.2. Clase que implementa la interface remota

Para construir un objeto remoto, se escribirá una clase que implemente una o más interfaces remotas. La clase de implementación cubre los siguientes aspectos:

- Especifica la(s) interface(s) remota(s) que implementa.
- Define el constructor para el objeto remoto.
- Implementa los métodos que pueden ser invocados de forma remota.
- Crea e instala un gestor de seguridad (Security Manager).
- Creación del objeto remoto.
- Registra al menos uno de los objetos remotos mediante el registro de RMI.

```
//TODO: imports necesarios

public class CollectionImpl extends UnicastRemoteObject
implements Collection
{
  //Private member variables
  private int                m_number_of_books;
  private String        m_name_of_collection;


  //Constructor
  public CollectionImpl() throws RemoteException
  {
          super();//Llama al constructor de UnicastRemoteObject
          //TODO: inicializar las variables privadas
  }

  //TODO: Implementar todos los metodos de la interface remota
 public int number_of_books() throws RemoteException
 {
        ...
 }

 public static void main(String args[])
 {
        //Fijar el directorio donde se encuentra el java.policy
```

```
   //El segundo argumento es la ruta al java.policy
         System.setProperty("java.security.policy", "./java.policy");

         //Crear administrador de seguridad
        System.setSecurityManager(new SecurityManager());

        //nombre o IP del host donde reside el objeto servidor
        String hostName = "IPhostremoto"; //se puede usar "IPhostremoto:puerto"
  //Por defecto RMI usa el puerto 1099

        try
        {
                // Crear objeto remoto
                CollectionImpl obj = new CollectionImpl();
                System.out.println("Creado!");
                //Registrar el objeto remoto
                Naming.rebind("//" + hostName + "/MyCollection", obj);
                System.out.println("Estoy registrado!");
        }
        catch (Exception ex)
        {
                System.out.println(ex);
        }
 }
}
```

4.1.3. El programa cliente

Debe obtener una referencia al objeto remoto e invocar sus métodos.

```
//TODO: imports necesarios
public class Cliente
{
 public static void main(String[] args){
         //TODO: Fijar el directorio donde se encuentra el java.policy

         if (System.getSecurityManager() == null) {
        //TODO: Crear administrador de seguridad
        }
          try
          {
          //Paso 1 - Obtener una referencia al objeto servidor creado anteriormente
             //Nombre del host servidor o su IP. Es dónde se buscará al objeto remoto
             String hostname = "IPremotehost"; //se puede usar "IP:puerto"
             Collection server =
                   (Collection) Naming.lookup("//"+ hostname + "/MyCollection");

          //Paso 2 - Invocar remotamente los metodos del objeto servidor:
                       //TODO: obtener el nombre de la colección y el número de libros
                       //TODO: cambiar el nombre de la coleccion y ver que ha funcionado
```

```
         }
         catch (Exception ex)
         {
                System.out.println(ex);
        }
 }
}
```

4.1.4. El fichero de seguridad (java.policy)

```
grant {
        //Allow everything for now
        permission java.security.AllPermission;
        };
```

4.2. Poner en marcha los programas en el host servidor

1. Compilar en el host servidor la interface remota y la clase que la implementa.
2. Iniciar en el host servidor el registro remoto.
   Windows: start rmiregistry [puerto] – se usará "puerto"sólo si se lanzó el servidor en un puerto determinado
   Unix: rmiregistry [puerto]& – si haces la práctica en el Lab0.02, las máquinas únicamente tienen habilitados los puertos 32000 al 32009
3. Ejecutar en el host servidor el programa servidor para que quede a la escucha de peticiones por parte del cliente.
   java CollectionImpl

4.3. Poner en marcha los programas en el host cliente

1. Crear el fichero java.policy.
2. Compilar en el host cliente el programa cliente
3. Ejecutar en el host cliente el programa cliente

5. Ejecución y entrega de la práctica

Primero ejecutarás los programas servidor y cliente en tu equipo probando que todos los métodos implementados funcionan correctamente. Si tienes dos equipos ejecuta en uno el servidor y en otro el cliente.

Entrega de la práctica en hendrix:

1. Copia tus ficheros "Collection.java", "Cliente.java" y "java.policy" en tu cuenta de hendrix.
2. Añade a "Collection.java" el método:

   ```
   String practica_hecha(String  _mi_NIA, String _mi_directorio) throws RemoteException;
   ```

3. Añade a "Cliente.java" estas líneas:

```
 //Quita tu hostname actual y cambialo por el del profesor
String hostname = "155.210.152.177"; //IP del profesor

//Entrega de la practica
//Las Xs son tu NIA
String resultado = server.practica_hecha("AXXXXXX", "/home/AXXXXXX/RMI/");
System.out.println("El profesor me responde: " + resultado);
```

4. Compila en hendrix "Collection.javaz "Cliente.java"

5. Ejecuta en hendrix "Cliente.java" y obtendrás la respuesta del profesor.
   (No tienes que lanzar el "rmiregistry" en hendrix)

Referencias

[1] **Remote Method Invocation**:
`http://docs.oracle.com/javase/tutorial/rmi/`