

Programación Orientada a Objetos

RMI (Remote Method Invocation)



Alumno: Vargas Romero Erick Efraín

Profesor: Tecla Parra Roberto

Fecha: 07-06-2017

Grupo: 2CM4

**Introducción**

RMI (Remote Method Ivocation) Es un mecanismo que Java nos ofrece para invocar un método de manera remota. Este forma parte del entorno estándar de ejecución de Java y proporciona un mecanismo de comunicación de servidores en aplicaciones distribuidas basadas exclusivamente en Java. Si se requiere comunicación entre otras tecnologías deben utilizarse otros métodos.

**Desarrollo**

Para esta práctica aplicando RMI primeramente fue necesario crear una interfaz remota la cual tiene como nombre Imágenes, y al ser remota debe implementar la interfaz Remote que se encuentra dentro del paquete java.rmi.\* además dentro tenemos nuestros métodos remotos los cuales pueden arrojar una excepción remota (Remote Exception) los métodos remotos que tenemos dentro de nuestra clase remota son: getImage la cual nos devolverá un ImageIcon y recibe un argumento el cual es imageID, también está el método remoto getNombres el cual retorna un objeto tipo Vector y finalmente un método remoto setIma el cual no retorna nada y recibe dos argumentos uno de tipo string y otro tipo ImageIcon

**import** javax**.**swing**.\*;**

**import** java**.**util**.\*;**

**import** java**.**rmi**.\*;**

public interface Imagenes **extends** Remote **{**

ImageIcon getImage**(**String imageID**)** **throws** RemoteException**,** Exception**;**

Vector getNombres**()** **throws** java**.**rmi**.**RemoteException**;**

void setIma**(**String nombre **,** ImageIcon ima**)** **throws** RemoteException**,** Exception**;**

**}**

El siguiente paso es crear nuestro “servidor” tendremos una clase llamada ImagenesImpl la cual extiende de la clase UnicastRemoteObject e implementa la interfaz Imágenes, la cual es la interfaz remota que ha sido creada, primeramente, dentro de esta clase tenemos dos variables de instancia de tipo Vector el primero de tipo ImageIcon y el segundo de tipo String. Nos encontramos con el constructor, este constructor puede arrojar une Excepción remota posteriormente dentro del constructor se invoca al método super() de nuestra superclase, y añadimos el primer elemento a nuestro vector tipo String, y posteriormente hacemos lo mismo con nuestro vector tipo ImageIcon

public class ImagenesImpl **extends** UnicastRemoteObject **implements** Imagenes **{**

Vector **<**ImageIcon**>** ima**=new** Vector**<**ImageIcon**>();**

Vector **<**String**>** nombre **=** **new** Vector**<**String**>();**

public ImagenesImpl**()** **throws** RemoteException **{**

**super();**

nombre**.**addElement**(**"t.JPG"**);**

ima**.**addElement**(new** ImageIcon**(**"t.JPG"**));**

**}**

El siguiente método llamado getNombres retornara un vector y arroja una excepción remota, dentro del método solo retornamos nuestro vector de tipo String

public Vector getNombres**()** **throws** java**.**rmi**.**RemoteException **{**

**return** nombre**;**

**}**

Nuestro siguiente método se llama getImage y retorna un ImageIcon además este método tiene un argumento de tipo String, este método puede arrojar excepciones de tipo RemoteException y de tipo Exception. Dentro de este método tenemos una variable tipo entero llamada j iniciada en cero, posteriormente tenemos un bloque try, y dentro un ciclo for, el cual va de cero hasta que el contador sea menor al tamaño de nuestro vector de tipo String y se autoincrementa uno a uno, dentro de este for tenemos un if, el cual evalua los elementos dentro de nuestro vector de Strings y compara lo que recibió el método con el elemento obtenido de nuestro vector, si esto es verdadero entonces el contador de nuestro for es igual a nuestra variable j y termina el ciclo, finalmente retornamos el elemento de nuestro vector de ImageIcon que está en el índice de nuestra variable j

public ImageIcon getImage**(**String imageID**)** **throws** RemoteException **,** Exception **{**

int j**=**0**;**

**try {**

**for(**int i**=**0**;**i**<**nombre**.**size**();**i**++){**

**if((**nombre**.**elementAt**(**i**)).**equals**(**imageID**))**

j**=**i**;**

**}**

**return** ima**.**elementAt**(**j**);**

**}** **catch** **(**Exception e**){**

System**.**out**.**println**(**"Fehler in GetImageIcon:\n "**+** e**.**toString**()+**"\n"**);**

**throw** e**;**

**}**

**}**

Tenemos un método más, llamado setIma que tiene dos argumentos el primero de tipo String y el otro de tipo ImageIcon, lo que hacemos en este método es añadir un elemento en cada uno de nuestros vectores.

public void setIma**(**String nom **,** ImageIcon iman**)** **throws** RemoteException **,** Exception**{**

nombre**.**addElement**(**nom**);**

ima**.**addElement**(**iman**);**

**}**

**Conclusión:**

Como fue posible comprobar, usar RMI es realmente sencillo, pero debemos recordar cosas que son básicas para su uso, como la creación de una clase remota que tenga métodos remotos, los cuales deben arrojarnos excepciones remotas, además de que nuestra clase principal, por así decirla debe de extender de la clase UnicastRemoteObject la cual vuelve “Servidor” a nuestra clase.