Java RMI: Callback de cliente

Grupo ARCOS

Desarrollo de Aplicaciones Distribuidas Ingeniería Informática Universidad Carlos III de Madrid

Contenidos

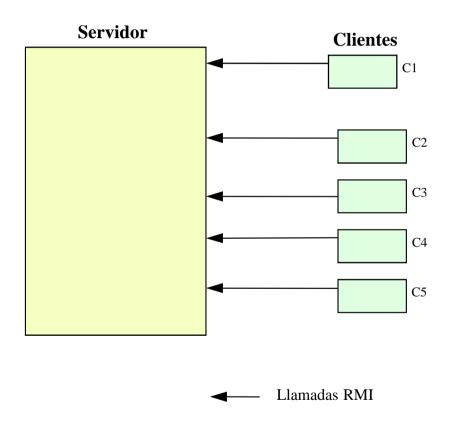
- 1. Introducción:
 - 1. Callback de cliente
- 2. Callback de cliente en Java RMI
 - 1. Arquitectura del sistema
 - 2. Elementos a desarrollar
- 3. Ajustes en desarrollo y despliegue
 - 1. Descarga del resguardo
 - 2. Políticas de seguridad

Contenidos

1. Introducción:

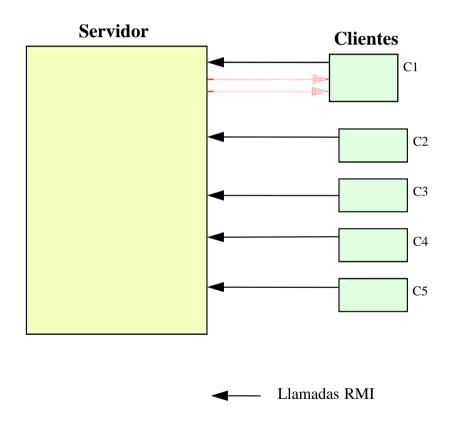
- 1. Callback de cliente
- 2. Callback de cliente en Java RMI
 - 1. Arquitectura del sistema
 - 2. Elementos a desarrollar
- 3. Ajustes en desarrollo y despliegue
 - 1. Descarga del resguardo
 - 2. Políticas de seguridad

Callback de cliente



- En el modelo cliente-servidor el servidor es pasivo
- El cliente pide un servicio y el servidor devuelve unos resultados como respuesta en un plazo de tiempo

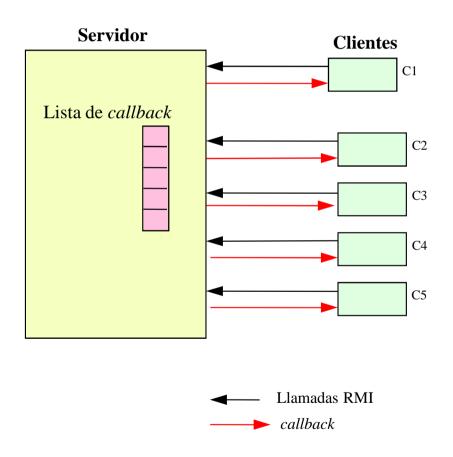
Callback de cliente



5

- ¿Qué pasa cuando el cliente quiere una o varias respuestas de forma asíncrona?
- Ejemplo: apuntarse para ser informados de los cambios en la bolsa

Callback de cliente



6

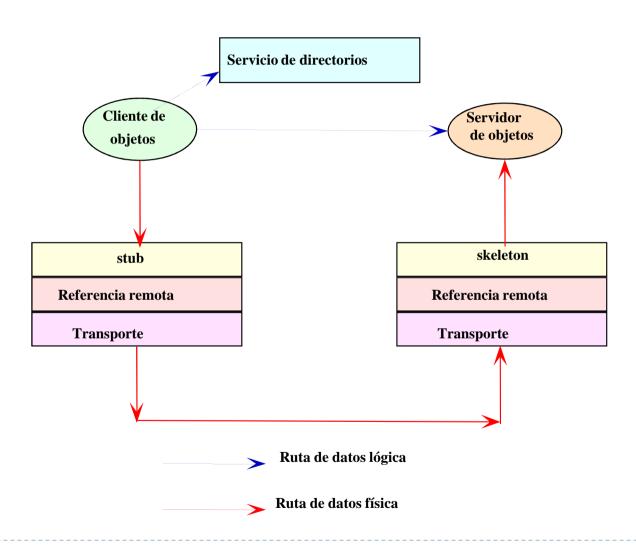
Alternativas:

- Polling
 - Preguntar cada cierto tiempo
 - Peticiones sin cambios causan sobrecarga
- Callback
 - Registrarse en el servidor para que nos avise ante los eventos

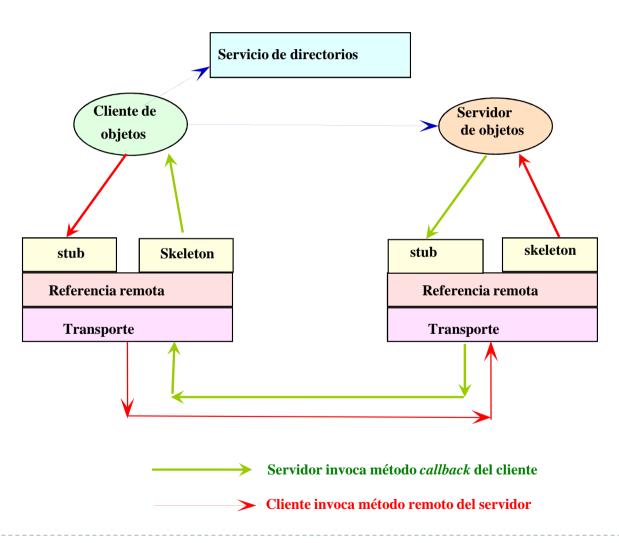
Contenidos

- 1. Introducción:
 - Callback de cliente
- 2. Callback de cliente en Java RMI
 - 1. Arquitectura del sistema
 - 2. Elementos a desarrollar
- 3. Ajustes en desarrollo y despliegue
 - 1. Descarga del resguardo
 - 2. Políticas de seguridad

Arquitectura de RMI (no callback)



Arquitectura de RMI (callback)

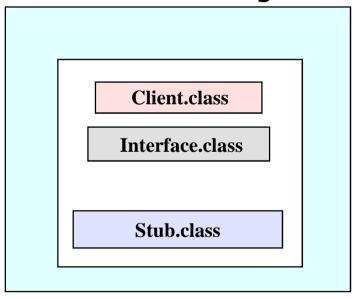


Contenidos

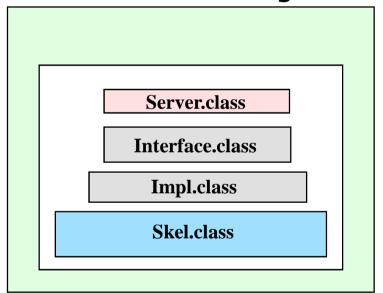
- 1. Introducción:
 - Callback de cliente
- 2. Callback de cliente en Java RMI
 - 1. Arquitectura del sistema
 - 2. Elementos a desarrollar
- 3. Ajustes en desarrollo y despliegue
 - 1. Descarga del resguardo
 - 2. Políticas de seguridad

Invocación remota (no callback)

Cliente de objetos



Servidor de objetos



Invocación remota (callback)

Cliente de objetos

Client.class

ClientInterface.class

ServerInterface.class

ClientImpl.class

ServerImpl_Stub.class

ClientImpl_skel.class

Servidor de objetos

Server.class

ServerInterface.class

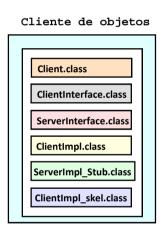
ClientInterface.class

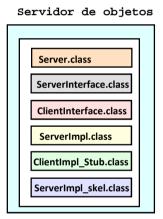
ServerImpl.class

ClientImpl_Stub.class

ServerImpl skel.class

Invocación remota (callback)





- Los clientes se registran como servidor de objeto remoto para *callbacks*.
- Dos conjuntos de proxies.
- Piezas clave:
 - Interface remota de cliente.
 - Servicio de registro de interfaces de cliente en servidor

ClientInterface.java

Client.class

ClientInterface.class

ServerInterface.class

ClientImpl.class

ServerImpl Stub.class

ClientImpl skel.class

Cliente de objetos

Servidor de objetos

Server.class

ServerInterface.class

ClientInterface.class

ServerImpl.class

ClientImpl_Stub.class

ServerImpl_skel.class

- o Interfaz remota del cliente
- Debe contener, al menos, un método a ser invocado por el servidor

Servidor de objetos

ServerInterface.class



ClientImpl.class

ServerImpl Stub.class

ClientImpl skel.class

Cliente de objetos

ClientInterface.class

Server.class

ServerImpl.class

ClientImpl Stub.class

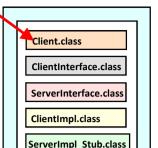
ServerImpl_skel.class

```
ClientImpl.java
```

- o Implementación de la Interfaz remota del cliente
- o Generación the proxies con rmic ClientImpl

Cliente de objetos

Servidor de objetos



ClientImpl skel.class

```
Server.class

ServerInterface.class

ClientInterface.class

ServerImpl.class

ClientImpl_Stub.class

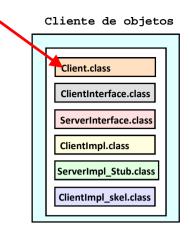
ServerImpl_skel.class
```

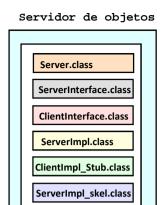
```
import java.io.*;
```

Client.java (1/3)

```
import java.rmi.*;
public class Client
 public static void main(String args[])
    try
      InputStreamReader is = new InputStreamReader(System.in);
      BufferedReader br = new BufferedReader(is);
      System.out.println("Enter the RMIRegistry host namer:");
      String hostName = br.readLine();
      System.out.println("Enter the RMIregistry port number:");
      String portNum = br.readLine();
      System.out.println("Enter how many seconds to stay registered:");
      String timeDuration = br.readLine();
      int time = Integer.parseInt(timeDuration);
```

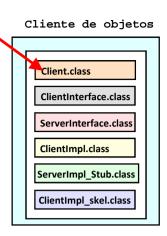
Client.java (2/3)



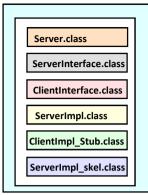


```
String regURL = "rmi://" + hostName + ":" + portNum + "/callback";
ServerInterface h = (ServerInterface)Naming.lookup(regURL);
System.out.println("Lookup completed ");
System.out.println("Server said " + h.sayHello());
ClientInterface callbackObj = new ClientImpl();
h.registerForCallback(callbackObj);
System.out.println("Registered for callback.");
```

Client.java (3/3)



Servidor de objetos



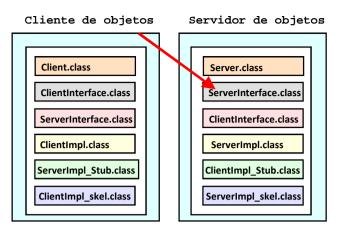
```
try {
        Thread.sleep(time * 1000);
        h.unregisterForCallback(callbackObj);
        System.out.println("Unregistered for callback.");
    } catch (InterruptedException ex){ /* sleep over */ }

} // try
    catch (Exception e)
    {
        System.out.println("Exception in CallbackClient: " + e);
    }

} // main
} // class
```

ServerInterface.java

- o Extensión en la parte servidora
- o Método remoto de registro del cliente para callback



ServerImpl.java (1/4)

return("hello");

public String sayHello() throws RemoteException {

Cliente de objetos

Client.class

ClientInterface.class

ServerInterface.class

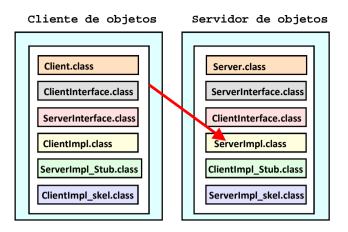
ClientImpl.class

ClientImpl_stub.class

ClientImpl_skel.class

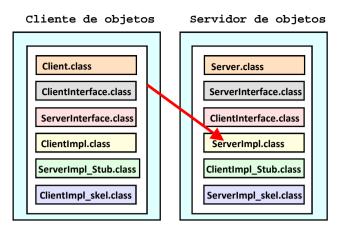
ClientImpl_skel.class

ServerImpl.java (2/4)



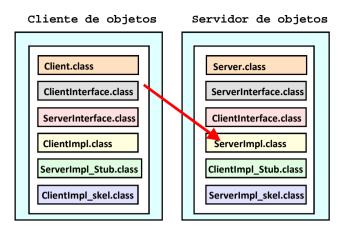
```
public synchronized void unregisterForCallback (
   ClientInterface callbackClientObject
) throws RemoteException
{
   if (clientList.removeElement(callbackClientObject)) {
      System.out.println("Unregistered client. ");
   } else {
      System.out.println("unregister: client wasn't registered.");
   }
}
```

ServerImpl.java (3/4)



```
public synchronized void registerForCallback(
    ClientInterface callbackClientObject
) throws RemoteException
{
    if (!(clientList.contains(callbackClientObject)))
    {
        clientList.addElement(callbackClientObject);
        doCallbacks();
    }
}
```

ServerImpl.java (4/4)



Server.java (1/3)

```
import java.rmi.*;
import java.rmi.server.*;
import java.rmi.registry.Registry;
import java.rmi.registry.LocateRegistry;
import java.net.*;
import java.io.*;

public class Server
{
    public static void main(String args[])
    {
        InputStreamReader is = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(is);
```

Cliente de objetos

Client.class

ClientInterface.class

ServerInterface.class

ClientImpl.class

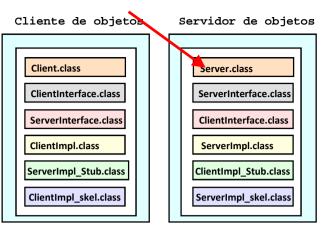
ClientImpl_Stub.class

ClientImpl_Stub.class

ClientImpl_skel.class

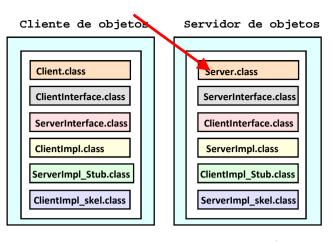
ClientImpl_skel.class

Server.java (2/3)



```
String portNum, registryURL;
try
{
    System.out.println("Enter the RMIregistry port number:");
    portNum = (br.readLine()).trim();
    int RMIPortNum = Integer.parseInt(portNum);
    startRegistry(RMIPortNum);
    ServerImpl exportedObj = new ServerImpl();
    registryURL = "rmi://localhost:" + portNum + "/callback";
    Naming.rebind(registryURL, exportedObj);
    System.out.println("Callback Server ready.");
} catch (Exception re) {
    System.out.println("Exception in HelloServer.main: " + re);
}
} // end main
```

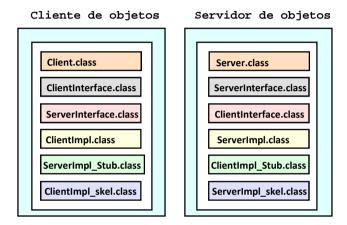
Server.java (3/3)



```
private static void startRegistry (int RMIPortNum)
throws RemoteException
{
   try {
     Registry registry = LocateRegistry.getRegistry(RMIPortNum);
     registry.list();
   } catch (RemoteException e) {
        // No valid registry at that port.
        Registry registry = LocateRegistry.createRegistry(RMIPortNum);
   }
} // end startRegistry
} // end class
```

Compilación del ejemplo

guernika.lab.inf.uc3m.es



javac -cp /usr/lib/jvm/java-1.4.2-gcj-4.1-1.4.2.0/jre/lib/rt.jar \
-g ClientInterface.java ClientImpl.java

rmic ClientImpl

javac -cp /usr/lib/jvm/java-1.4.2-gcj-4.1-1.4.2.0/jre/lib/rt.jar -g *.java

Ejecución del ejemplo

guernika.lab.inf.uc3m.es

```
# rmiregistry 9090 &
# java -Djava.security.policy=policy.all Server
Enter the RMIregistry port number:
9090
Callback Server ready.
^Z
# bg
# java -Djava.security.policy=policy.all Client
Enter the RMIRegistry host namer:
localhost
Enter the RMIregistry port number:
9090
Enter how many seconds to stay registered:
Lookup completed
Server said hello
doing 0-th callback
Call back received: Number of registered clients=1
Registered for callback.
```

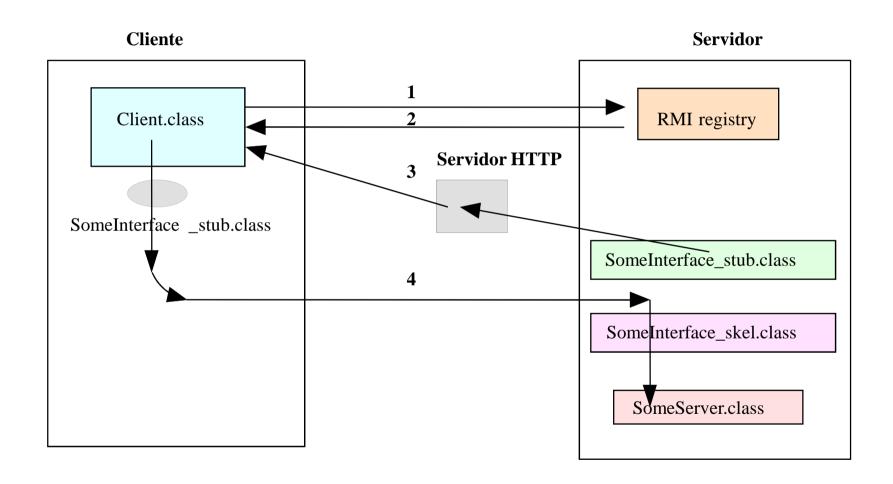
Contenidos

- 1. Introducción:
 - 1. Callback de cliente
- 2. Callback de cliente en Java RMI
 - 1. Arquitectura del sistema
 - 2. Elementos a desarrollar
- 3. Ajustes en desarrollo y despliegue
 - 1. Descarga del resguardo
 - 2. Políticas de seguridad

Descarga del resguardo

- Objetivo: eliminar la necesidad de que el cliente deba tener el stub.
- El stub es descargado dinámicamente desde un servidor web.
- Es necesario conocer la URL del servidor.
- La clase resguardo no es persistente.

Descarga del resguardo



Gestor de seguridad (por defecto)

- Security Manager
- Supervisión del programa:
 - Acceso a ficheros
 - Conexiones de red
- Nivel de seguridad inicial:
 - Únicamente conexiones locales.
 - No es posible la transferencia del resguardo.

- Es posible usar una gestor de seguridad que configurado adecuadamente permita la descarga del resguardo.
 - Clase RMISecurityManager
 - Fichero de política de seguridad

Inicio del servicio de seguridad:

```
try {
    System.setSecurityManager(
        new RMISecurityManager());
} catch {
    //...
}
```

- Añadir el anterior fragmento de código tanto al programa principal (main) del cliente como del servidor.
- Usar antes de utilizar los servicios del registro RMI.

Ejemplo de fichero de políticas de seguridad:

Ejemplo de despliegue de políticas de seguridad:

Resguardos y gestor de seguridad RMI

- Crear directorio
- Definir interfaz remoto.
- Realizar su implementación.
- 4. Generar ficheros resguardo y esqueleto.
- Diseñar programa servidor.
- 6. Copiar fichero resguardo al servidor HTTP.
- Activar registro RMI.
- 8. Construir fichero de políticas de seguridad.
- Activar servidor.

Resguardos y gestor de seguridad RMI

Cliente

SomeClient.class
SomeInterface.class
java.policy

Servidor

SomeServer.class

SomeInterface.class

SomeImplementation.Skeleton.class

SomeImplementation.class

java.policy

Servidor HTTP

SomeImplementation_stub

Java RMI: Callback de cliente

Grupo ARCOS

Desarrollo de Aplicaciones Distribuidas Ingeniería Informática Universidad Carlos III de Madrid