Egyszerű Java-RMI példa

Először is, készítsünk egy interakciós diagramot, ami bemutatja, hogyan kommunikál egymással a **Client** és a **RObject** a **RegisterService**-en keresztül:

sequenceDiagram participant C as Client participant RS as RegisterService participant RO as RObject participant RMI as RMI Registry RS->>RMI: Register RObject RMI-->>RS: RObject Registered C->>RMI: Lookup RObjectServer RMI-->>C: Return RObject C->>RO: primitiveArg(2012) RO-->>C: Acknowledge C->>RO: argumentByValue(2012) RO-->>C: Acknowledge

Ez a diagram szemlélteti a folyamat lépéseit: a RegisterService regisztrálja az RObject-et az RMI Registry-ben, a Client lekérdezi a távoli objektumot, majd használja annak funkcióit.

Most pedig nézzük meg, hogyan néz ki ez a rendszer egy struktúra diagramon, ami bemutatja a komponensek és azok kapcsolatait:

graph TD subgraph "RMI Registry" RO[RObject Interface] end subgraph "Server Side" ROS[RObjectImpl] -- Implements --> RO end subgraph "Client Side" C[Client] -- Uses --> RO end RegisterService -- Registers --> RO C -- Looks up --> RO C -- Calls methods on --> ROS

A struktúra diagram világosan ábrázolja a rendszer különböző részeit és azok közötti kapcsolatokat:

A Server Side rész tartalmazza az **RObjectImpl** implementációt, amely megvalósítja az **RObject** interfészt. Az **RMI Registry** regisztrálja az interfészt, lehetővé téve, hogy a **Client Side**-on lévő **Client** megtalálja és használja azt. A **Client** a távoli objektum (ROS) metódusait hívja meg, ami ábrázolja a két fő műveletet: egy primitív és egy érték szerinti paraméter átadást.

Megvalósítás:

Telepítsük a gradle-t innen: https://gradle.org/releases/ A *bin* könyvtárat adjuk hozzá a *PATH*-hoz.

A teljes forráskód elérhető itt:

```
git clone https://github.com/knehez/isi.git
cd java_rmi
```

Nyissunk két terminált, az egyikben:

```
gradle runRegisterService
```

A másikban:

```
gradle run
```

A program részletesen:

1.) Definiáljuk az RObject interfészeket, mindkettő implementálja a Remote interfészt

```
package org.ait;
import java.rmi.*;

public interface RObject extends Remote {
  // egyszerű paraméterátadás
  void primitiveArg(int num) throws RemoteException;
  // érték szerinti paraméterátadás
  void argumentByValue(Integer num) throws RemoteException;
}
```

2.) Implementáljuk a távoli objektumok kódját

```
package org.ait;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class RObjectImpl extends UnicastRemoteObject implements RObject {
      private static final long serialVersionUID = 6350331764929058681L;
      public RObjectImpl() throws RemoteException {
      }
      @Override
      public void primitiveArg(int num) throws RemoteException {
         System.out.println(num);
      }

      @Override
      public void argumentByValue(Integer num) throws RemoteException {
         System.out.println(num);
      }
}
```

3.) Indítsuk el az RMI registry-t. Ez egy Java JDK komponens, elindítani a létrehozott projekt **/bin** könyvtárából kell.

```
> rmiregistry.exe
```

4.) Hozzuk létre egy példányát a távoli objektumnak és kössük hozzá a registry-hez

```
package org.ait;

import java.rmi.Naming;

public class RegisterService {
    /**
     * @param args
```

```
     */
    public static void main(String[] args) {
        try {
            RObject robj = new RObjectImpl();
            Naming.rebind("rmi://localhost:1099/RObjectServer", robj);
            System.out.println("Registered...");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Indítsuk el a RegisterService-t, ha fut a rmiregistry, csak akkor fog tudni regisztrálni.

5.) Használjuk a távoli objektumot

Indítsuk el az alábbi kódot.

```
package org.ait;

import java.rmi.Naming;

public class Client {
    public static void main(String[] args) {
        try {
            // Távoli objektum lekérése a registry-ből
            RObject robj = (RObject)
Naming.lookup("rmi://localhost:1099/RObjectServer");

            // Egyszerű argumentum
            robj.primitiveArg(2012);

            // Serilaizált argumentum
            robj.argumentByValue(new Integer(2012));

        } catch (Exception e) {
            e.printStackTrace();
            ;
        }
    }
}
```

From:
https://edu.iit.uni-miskolc.hu/ - **Institute of Information Science - University of Miskolc**

Permanent link:
https://edu.iit.uni-miskolc.hu/tanszek:oktatas:informacios_rendszerek_integralasa:java_rmi

Last update: **2024/03/26 18:26**