Použití jazyka Java pro aplikace měření a řízení

Roman Gužík

Katedra měřicí a řídicí techniky, VŠB - Technická univerzita v Ostravě, tř. 17. listopadu, 708 33 Ostrava-Poruba, Česká republika

Abstrakt

Příspěvek pojednává o možnostech moderního jazyka Java v distribuovaných měřicích a řídicích aplikacích. Nejprve jsou popsány možnosti, které jazyk Java pro tyto aplikace nabízí. Hlavního záměrem je však nastínit a zhodnotit možnosti použití technologie RMI (remote method invocation) ve verzi Java 1.2.

1. Java v distribuovaných systémech

1.1 Distribuované systémy

Žádný distribuovaný systém dosud nebyl vytvořen snadno a bez problémů. Toto je způsobeno zejména tím, že distribuovaný systém se skládá z mnoha samostatných programů, které běží na mnoha různých platformách a mají za úkol spolu spolehlivě komunikovat. Tímto se náročnost návrhu jednotlivých částí distribuovaného systému komplikuje o zajištění této komunikace a správné synchronizace mezi jednotlivými částmi. V jazyce Java je řešení této komunikace dvojí, pomocí soketů a pomocí RMI (remote method invocation – volání vzdálených metod).

1.2 Sokety

Pro základní komunikaci se v Javě používají sokety, které jsou dostačující pro běžnou komunikaci. Nevýhodou soketové komunikace je však nutnost definice komunikačního protokolu a realizace nového soketového spojení pro každé dvě komunikující části distribuovaného systému. Při realizaci složitějšího systému je tato technologie z hlediska návrhu těžkopádná, a tímto také náchylná k chybám.

1.3 RMI

RMI aplikace obsahuje dva oddělené programy: server a klient. Server vytvoří určitý počet vzdálených objektů, zpřístupní reference na tyto vzdálené objekty a čeká na klienty, kteří tyto objekty využijí vzdáleným voláním jejich metod. Klient přistupuje

k objektu implementovaném na serveru pomocí svého lokálního zástupce stejným způsobem, jako přistupuje ke svým lokálním objektům.

Při návrhu takovéto distribuované aplikace je pak možno použít návrhových metod stejných jako při tvorbě nedistribuovaných aplikací. Toto je umožněno tím, že veškerá komunikace mezi jednotlivými částmi distribuovaného systému se řeší na úrovni JVM (Java virtual machine – virtuálního stroje Javy).

2. Technologie RMI

2.1 Definice terminů

Remote object (vzdálený objekt) - objekt, jehož metody mohou být volány z dalších JVM (virtuální stroj Javy).
Remote interface (vzdálené rozhraní) - deklarace metod vzdáleného objektu pomocí Java rozhraní.
Remote method invocation RMI (volání vzdálené metody) - volání metody vzdáleného objektu přes vzdálené rozhraní. Toto volání má stejnou syntaxi jako volání metody lokálního objektu.

2.2 Architektura RMI

Architektura RMI implementovaná v JVM používá standardního mechanismu pro komunikaci se vzdálenými objekty a to **stub** (zástupce vzdáleného objektu na straně klienta) a **skeleton** (kostra vzdáleného objektu na straně serveru). Příslušní zástupci vzdáleného objektu (stub) a kostry vzdáleného objektu (skeleton) jsou generovány **rmic** kompilátorem.
Zástupce vzdáleného objektu je odpovědný za :

- inicializaci spojení se vzdálenou VM obsahující vzdálený objekt
- volá metody vzdáleného objektu
- prostřednictvím serializace (možnosti uložení objektu do streamů, ze kterého je tento objekt možno plně obnovit) umožňuje přemisťovat jako argumenty vzdálených metod i objekty
- čeká na výsledek vzdáleného volání
- načítá návratovou hodnotu nebo nastanuvší výjimku

Jakmile kostra vzdáleného objektu obdrží volání vzdálené metody provede následující:

- vyvolá implementaci příslušné volané metody
- zašle volající VM návratovou hodnotu nebo nastanuvší výjimku

2.3 Garbage Collection vzdálených objektů

V distribuovaném systému stejně jako v lokálním systému JVM automaticky ruší vzdálené objekty, na které se neodkazuje už žádný klient. Tento rys Javy je nejvýznačnější v porovnání s ostatními technologiemi realizací vzdálených objektů. JVM používá algoritmus počítání referencí podobný Síťovým Objektům v jazyce Modula-3.

2.4 Dynamické nahrávání tříd

RMI dovoluje zasílat jako parametry, nebo návratové hodnoty objekty, které je možno serializovat. RMI využívá serializačního mechanismu pro zasílání objektů z jednoho JVM do druhého. Aby se serializovaný objekt mohl použít v cílové JVM, je nutno tento objekt deserializovat a je také potřeba definice třídy pro tento objekt. RMI proto poskytuje prostředek pro dynamické nahrávání těchto tříd, jestliže tyto třídy nejsou nahrány lokálně.

2.5 Dostupnost RMI přes firewally

Transportní vrstva RMI normálně komunikuje prostřednictvím soketů. Tento způsob komunikace firewally nedovolují. RMI proto poskytuje dvě alternativy založené na HTTP mechanismu, které dovolují klientovi volat vzdálenou metodu přes firewall.

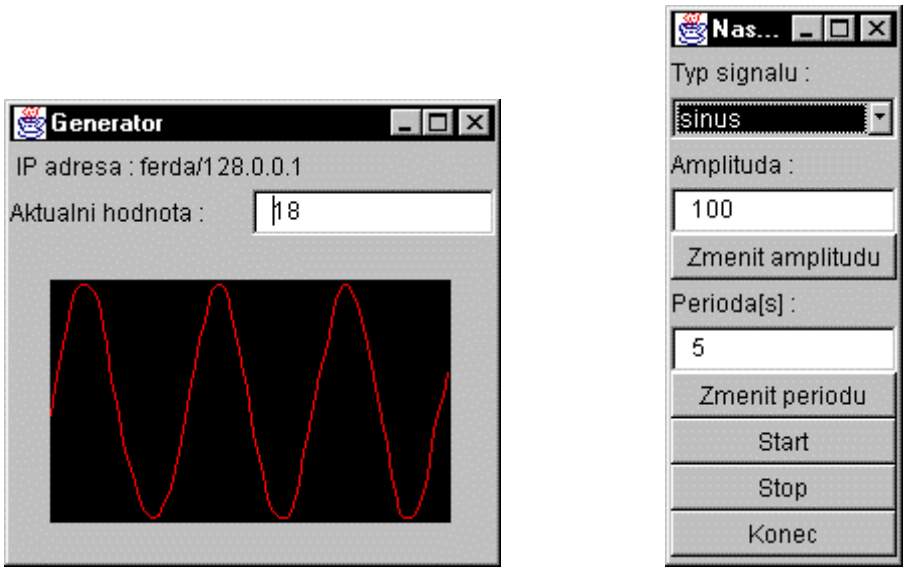
3. Realizace distribuovaného systému prostřednictvím RMI

3.1 Vzdálené rozhraní

Pokud je vzdálený objekt realizací nějakého měřicího přístroje, pak jsou v tomto rozhraní metody, které umožňují jeho nastavování a načítání změřených hodnot.

Pokud k tomuto měřícímu přístroji přistupuje více uživatelů s různými právy, vzdálený objekt může implementovat více rozhraní a klienti mohou podle svých práv používat adekvátní rozhraní. Pro zajištění robustnosti aplikací používající RMI technologii, každá metoda deklarovaná ve vzdáleném rozhraní musí deklarovat v klauzuli **throws** kromě svých vlastních výjimek navíc i výjimku java.rmi.RemoteException. Tato výjimka je nadtřídou všech výjimek, které mohou nastat během běhu RMI, například při komunikační chybě, chybě protokolu a pod.

V příkladu je realizován jako vzdálený objekt generátor.



Obr. 1: Vzdálený objekt generátor

U tohoto generátoru je možno nastavit typ signálu (obdélník, pilu nebo sinus), amplitudu signálu a periodu signálu. Tyto parametry je možno nastavit jednak pomocí ovládacího panelu na straně serveru, tak i vzdáleně pomocí ovládacího panelu na straně klienta. Ne všichni klienti však mohou nastavovat parametry tohoto generátoru, někteří z nich mohou generovaná data pouze načítat. Z tohoto důvodu je vzdálený generátor reprezentován pomocí dvou rozhraní:

První umožňující pouze načítání:

```
public interface Generator extends java.rmi.Remote {
        int nacti() throws java.rmi.RemoteException;
}
```

Druhé umožňující jak načítání, tak i nastavení parametrů:

```
public interface NastavitelnyGenerator extends java.rmi.Remote {
        ZmenaTypuSignalu(int TypSignalu) throws java.rmi.RemoteException;
        ZmenaAmplitudy(int Amplituda) throws java.rmi.RemoteException;
        ZmenaPeriody(int Perioda) throws java.rmi.RemoteException;
        Start() throws java.rmi.RemoteException;
        Stop() throws java.rmi.RemoteException;
        int nacti() throws java.rmi.RemoteException;
}
```

3.2 Implementace vzdáleného objektu

Třída je obvykle dědicem třídy **java.rmi.server** jejichž nadtřídy jsou **java.rmi.server.RemoteObject** a **java.rmi.server.RemoteServer**. Může však být dědicem i jiné implementační třídy rozhraní. Třída dále musí implementovat jedno, nebo i více vzdálených rozhraní.

```
public class ImplGenerator extends UnicastRemoteObject implements
NastavitelnyGenerator
{
        // ...
        // zde jsou implementovány jak metody rozhraní, tak i lokální metody
        // ...

        // metoda spouštějící program
        public static void main(String args[])
        {
                // Vytvoření a instalace security manager
                System.setSecurityManager(new RMISecurityManager());

                try {
                        ImplGenerator obj = new ImplGenerator();
                        Naming.bind("Generator", obj);
```

```
                    System.out.println("Generátor byl vytvořen a registrován
                    pod jménem Generator ");
                } catch (Exception e) {
                    System.out.println("Při vytvoření serveru Generátor
                    nastala chyba.");
                    e.printStackTrace();
                }
            }
        }
```

3.3 Registrace vzdálených objektů

Klient, který volá metodu vzdáleného objektu musí nejdřív obdržet reference na tyto vzdálené objekty. Třída java.rmi.Naming poskytuje metody založené na URL pro registraci, vyhledání, odregistraci a obdržení seznamu objektů.

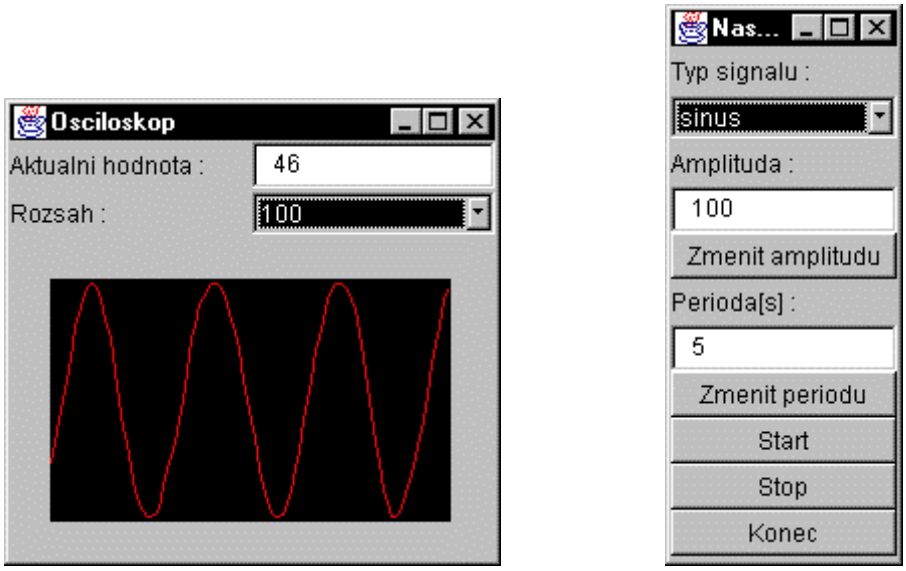
V příkladu výše se registruje vzdálený objekt pomocí metody Naming.bind(String url, object obj). Na takto registrovaný objekt může klient získat reference a vzdáleně volat jeho metody. K registraci je možno použít také metodu Naming.rebind(String url, object obj). Jestliže jméno vzdáleného objektu registrovaného touto metodou už existuje je předefinováno objektem novým a dříve registrovaný objekt je ztracen. Registrovaný objekt lze rovněž odregistrovat a to metodou Naming.unbind(String name) s parametrem jména objektu, který chceme odregistrovat. Pokud objekt tohoto jména není registrován pak tato metoda vyhodí vyjímku NotBoundException.

3.4 Security Manager

Pro zajištění spolehlivosti, RMI aplikace musí nastavit objekt secutity manager a to buď **RMISecurityManager** nebo nově nadefinovaný. Třida RMISecurityManager definuje metody, nezbytné pro implementaci bezpečnostních pravidel. Před provedením potenciálně citlivých operací tak Java zavolá metody objektu RMISecurityManager, který je momentálně v platnosti a tak zjistí, jestli je operace povolena. Pokud operace povolena není, vyhodí výjimku RMISecurityException.

3.5 Implementace klientské aplikace

Jakmile máme takto vytvořen a registrován vzdálený objekt můžeme implementovat klientskou aplikaci, která bude tento vzdálený objekt používat. V demonstračním příkladu je klientskou aplikací osciloskop, který snímá signály generované generátorem a rovněž nastavuje jeho parametry.



Obr. 2: Klientská aplikace osciloskop.

```
public class VzdaleneNastaveniParametruOsciloskopu {

// ...
// zde jsou implementovány ostatní metody třídy
// ...

public static void main(String args[]) {

        try {
                NastavitelnyGenerator obj =
                (NastavitelnyGenerator)Naming.lookup("Generator");

                // ...
                // zde je možno volat metody vzdáleného objektu
                // ...

        }
        catch (Exception e) {
                System.out.println("Ve vzdáleném objektu Generátor nastala
                chyba.");
        }
}
```

3.6 Lokalizace vzdálených objektů

Lokalizace vzdáleného objektu se provádí opět pomocí třídy java.rmi.Naming a to pomocí její metody Naming.lookup(String url), která vrátí vzdálený objekt jestliže existuje. V opačném případě vyhodí výjimku NotBoundException.
Další metoda třídy Naming.list(String url) vrací seznam vzdálených objektů.

Závěr

Výhodnost technologie RMI oproti konkurenčním technologiím je především v použití přirozené syntaxe jazyka. V případě technologie CORBA, kterou rovněž Java podporuje je nutné mapování standardu do jazyka, což v RMI odpadá. Další výhodou, kterou sebou přináší Java je platformová nezávislost. Nedostatkem naopak je komunikace mezi vzdálenými objekty implementovanými jinými jazyky než je Java. Toto je však možno řešit pomocí mostu RMI – CORBA.

V budoucnu bude zřejmě zajímavé sledovat, která z technologií vzdálených objektů se v Javě masivněji prosadí, zdali RMI nebo CORBA.

Literatura

[SUN99] Java Developer Kit 1.2 Documentation, Sun Microsystems, Inc., California, 1999
[FLA96] Flanagan, D.: Java in a Nutshell, O'Reilly & Associates, Inc., 1996
[SZY98] Szyperski C.: Component Software – Beyond Object-Oriented Programming, Addison-Wesley, Essex, 1998

URL

[URL1] http://java.sun.com