RMI V SKRATKE

Ondrej Jaura

10. jún 2002

1 RMI

RMI = Remote Method Invocation

RMI je postavené na princípe: server poskytuje službu (zloženú z rozhrania a triedy implemetnujúcej rozhranie), na ktorú sa klient prihlási a volá metódy služby.

2 Server

2.1 Definícia rozhrania

Definuje rozhranie služby.

Rozhranie musí byť potomkom `java.rmi.Remote`.

Metódy musia mať deklarované hádzanie výnimky `java.rmi.RemoteException`.

2.2 Implementácia vzdialeného objektu

Vzdialený objekt je inštancia triedy, ktorá implementuje vyššie spomínané rozhranie. Definuje teda funkčnosť metód služby.

Môže byť potomkom `java.rmi.UnicastRemoteObject` (okrem iného zabezpečuje automatické exportovanie).

Konštruktor môže vyhodiť výnimku, t.j. musí deklarovať hádzanie výnimky `java.rmi.RemoteException`.

2.3 Vygenerovanie stub a skeleton

Príkazový riadok: rmic MyImplementation, kde parametrom je už skompilovaný *`.class`* súbor. Napr.:

```
rmic -classpath classes -d classes mypackage.MyImplementation
```

Vygenerujú sa dva súbory, stub a skeleton: *`MyImplementation_stub.class`* a *`MyImplementation_skel.class`*.

2.4 Implementácia serveru vytvárajúceho inštanciu vzdialeného objektu

Kedže implementácia vzdialeného objektu je potomkom `java.rmi.UnicastRemoteObject`, tak nie je potrebné exportovať vzd. objekt – stane sa tak automaticky.

Ondrej Jaura, ondrej@realtime.sk

Nastaviť security:

```
System.setSecurityManager(new RMISecurityManager());
```

Vytvoriť inštancie implementácie vzdialeného objektu

```
MyImplementation myImplementation = new MyImplementation();
```

registrovať inštancie:

```
Naming.bind("Name", myImplementation);
```

2.5 Umiestnenie súborov

Stub, skeleton, implementácia, interface, súbor bezpečnostnej politiky.

2.6 Program rmiregistry

Slúži na priradenie mena vzdialeného objektu k príslušnému stub-u, teda uchováva dvojice [meno, stub] a zabezpečuje prístup k nim pre klientov. Implicitný port je 1099. *rmiregistry* je implementovaný a dodávaný s JDK (`java.rmi.registry.RegistryImpl`), takže je spustitelný i priamo z aplikácie.

2.7 Bezpečnostná politika

Je definovaná súborom *MyRMI.policy*:

```
grant
{
        permission java.security.AllPermission;
};
```

2.8 Parametre pre JVM

```
-Djava.rmi.server.codebase=file:c:\cesta_k_projektu\classes
-Djava.security.policy=file:c:\cesta_k_projektu\MyRMI.policy
```

resp. nastavenie priamo v aplikácii:

```
System.getSystemProperties.set(názov_property, hodnota);
```

3 Klient

3.1 Implementácia klienta (využívajúceho vzdialený objekt)

Nastaviť security:

```
System.setSecurityManager(new RMISecurityManager());
```

Deklarácia premennej typu rozhrania vzdialeného objektu a vyhľadanie zaregistrovaného objektu (inštancie):

```
MyInterface myInterface = (MyInterface) Naming.lookup("//host:port/Name");
```

Používanie je podobné ako pri práci s bežným lokálnym objektom

```
int i = myInterface.getNumber(4, "text", true);
```

3.2 Umiestnenie súborov

Stub, interface, súbor bezpečnostnej politiky.

3.3 Parametre pre JVM

```
-Djava.security.policy=file:c:\cesta_k_projektu\MyRMI.policy
```

4 Ako to celé spustiť?

1. Skopírujeme súbory ako je potrebné.

2. Spustíme na serveri *rmiregistry* (resp. *start rmiregistry*), príp. v aplikácii (v bode 3).

3. Spustíme na serveri server vytvárajúci inštanciu vzdialeného objektu (viď IV.):

```
java -classpath classes
-Djava.rmi.server.codebase=file:c:\cesta_k_projektu\classes\
-Djava.security.policy=file:c:\cesta_k_projektu\MyRMI.policy
skoleniej2.rmi.MyImplementation
```

4. Spustíme na klientovi klientskú aplikáciu (viď V.):

```
java -classpath classes
-Djava.security.policy=file:c:\cesta_k_projektu\MyRMI.policy
skoleniej2.rmi.MyClient
```