Java tutorial

RMI = Remote Method Invocation

- Meg tudjuk hívni olyan objektumok metódusait is, amelyek más virtuális gépben futnak
- Kommunikáció különböző gépeken futó Java programok között
  - Kliens-szerver alkalmazások
  - Elosztott alkalmazások
- Alternatíva: CORBA
  - Nem csak Java programok/objektumok között

Kliens-szerver forgatókönyv

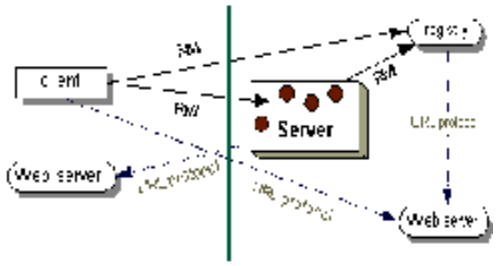
- A szerver program elérhetővé tesz objektumokat
  - bejegyzi őket egy registry-be
  - a távoli objektumok várják, hogy a kliensek meghívják a metódusaikat
- A kliensek szereznek egy referenciát egy ilyen objektumra
  - aztán hívogatják a távoli objektumok metódusait
- A részleteket az RMI rendszer elintézi...

Elosztott objektumok rendszere forgatókönyv

- Különböző számítógépeken futó különböző virtuális gépekben elhelyezkedő objektumokat bejegyzünk
- Megkeresik egymást és kommunikálnak egymással
  - hívogatják egymás metódusait
- Egymás megkeresése: pl. registry-n keresztül
- A részleteket az RMI rendszer elintézi...

Tevékenységek

- Távoli objektumok megkeresése rájuk referencia beszerzése
  - rmiregistry-n keresztül
  - metódushívások paramétereként/eredményeként
- Kommunikáció a távoli objektummal
  - sima metódushívások a programozó számára
  - a részleteket az RMI elfedi
- Osztálydefiníciós fájlok továbbítása
  - egymásnak átadott objektumok osztályának betöltése egy másik, távoli virtuális gépből



Dinamikus osztálybetöltés

- Amikor a távoli objektumok egymás metódusait hívják...
- … paramétereket adnak át és visszatérési értékeket kapnak
- Nem biztos, hogy egy virtuális gép ismeri annak az objektumnak az osztályát, amit így kap
- Az RMI letölti az osztálydefiníciót is

Távoli objektumok

- Amelyeknek meg lehet hívni metódusait az RMI segítségével más virtuális gépekből
- A **java.rmi.Remote** interfészt megvalósítják
- Megvalósítanak egy távoli interfészt (a Remote egy kiterjesztését), amiben fel vannak sorolva a távolról elérhető metódusaik

Távoli metódus

- Kiterjesztjük a java.rmi.Remote interfészt
- Ez lesz a „távoli interfész”
- Ebben vannak definiálva a távolról meghívható metódusok
  - csak ezek hívhatók távolról
- Ezek a metódusok specifikálják, hogy kiválthatják a **java.rmi.RemoteException** kivételt

Mi is történik?

- A kliens egy „távoli referenciával” rendelkezik a távoli objektumra
- A kliens virtuális gépben egy proxy objektum van, ami úgy csinál, mintha ő lenne a távoli objektum
  - továbbítja a metódushívásokat a távoli objektumnak
- Ezt a proxy objektumot **stub**-nak nevezik

Stub - csonk: klienscsonk

- A távoli objektum reprezentánsa a klien virtuális gépekben
- Amikor távoli objektumra referenciát szerzünk a registry-ből, akkor egy ilyen jön létre
- Amikor távoli objektumokat paraméterként vagy visszatérési értékként átadunk, akkor ilyen adódik át helyette
- Ugyanazokat a távoli interfészeket valósítja meg, mint amit a távoli objektum
  - Még cast-olni is lehet

Java tutorial

Mit is kell akkor tennünk?

- Az elosztott objektumrendszer komponenseinek megtervezése és megvalósítása
- Fordítás, és utána a csonkok legenerálása
- A fájlok kitevése a hálóra
- Futtatás

Java tutorial

Az elosztott objektumrendszer komponenseinek megtervezése és megvalósítása (4/1)

- El kell dönteni, hogy a rendszerben mely komponensek lesznek távolról elérhetők, és melyek lesznek lokálisak

Az elosztott objektumrendszer komponenseinek megtervezése és megvalósítása (4/2)

- A távoli interfészek definiálása
  - a távolról elérhető szolgáltatások definiálása
- A távoli metódusok paramétereiben és visszatérési értékeiben használt típusok (osztályok és interfészek) megírása

Az elosztott objektumrendszer komponenseinek megtervezése és megvalósítása (4/3)

- A távoli osztályok elkészítése
  - implementálják a távoli interfészeket
  - megvalósítják a távolról igénybevehető szolgáltatásokat
- A metódusok paramétereiben és visszatérési értékeiben használt típusok (osztályok és interfészek) megírása

Az elosztott objektumrendszer komponenseinek megtervezése és megvalósítása (4/4)

- Kliensek megvalósítása
  - ráér akkor is, ha a távoli objektumok már üzembe vannak helyezve...

Fordítás, és utána a csonkok legenerálása

- Forrásfájlok lefordítása
  - **`javac`**
- A .class fájlokból a csonkok legenerálása
  - **`rmic`**
  - – többek között ez is része a Development Kit-nek

A fájlok kitevése a hálóra

- Minden .class fájlt, amit a kliensek használni szeretnének majd, elérhetővé tesszük egy WEB-szerveren keresztül
  - – a távoli interfészek
  - – a klienscsonkok
  - – a segédosztályok és segédinterfészek (paraméterek, visszatérési értékek típusa...)

Futtatás

- Az **`rmiregistry`** program
  - – ez is része a Development Kit-nek
- A távoli objektumokat bejegyző programok
  - – pl. a szerverek
- A használó programok
  - – kliensek

Java tutorial

Hogyan is kell akkor ilyet írni?

- Készítsünk el egy távoli objektumot, ami tud összegezni egész számokat!
- Definiáljuk a szolgáltatást: Adder interfész
- Megvalósítjuk a távoli objektumot: RemoteAdder osztály
- Írunk egy kliens programot, ami használja

Adder interfész

- Terjesszük ki a java.rmi.Remote interfészt
- Írjuk bele a szolgáltatást, ami most két metódus
  - – add: hozzád egy számot az eddigi összeghez
  - – get: lekérdezi az eddigi összeget
- Gondoskodjunk arról, hogy a metódusok deklarálják a throws klózukban a java.rmi.RemoteException kivételt
  - – mellette egyebeket is deklarálhatnának...

Java tutorial

RemoteAdder osztály

- Meg kell valósítson egy távoli interfészt, most pl. az Adder interfészt
- Kiterjesztjük a java.rmi.server.UnicastRemoteObject osztályt
- Írnunk kell konstruktort is
- Gondoskodni kell arról, hogy beregisztráljuk az osztály egy példányát egy registry-be
  - pl. írhatunk egy main metódust is benne, ami ezt megteszi...

Távoli interfész megvalósítása

- Implementáljuk az interfészben specifikált metódusokat
- A throws klózban már nem kell feltüntetni a java.rmi.RemoteException kivételt
  - persze, ha akarjuk, kiválthatjuk, és akkor fel kell tüntetni
- Konstruktort is kell írni, ami viszont kiválthatja ezt a kivételt, tehát nem lehet a default konstruktorra támaszkodni.

UnicastRemoteObject

- Érdemes ebből származtatni a távoli objektumok osztályát
- Kényelmi osztály
  - toString, equals, hashCode
  - "exportálás", leszármazásnál automatikus
- Pont-pont (unicast, és nem broadcast) kommunikáció
- Alapértelmezett socket-os megvalósítása az RMI-nek

Ha nem leszármazott...

- Meg kell valósítani az RMI specifikációban adott szemantikával az Object-beli műveleteket
- Explicit exportálni kell a távoli objektumot
  - hogy az RMI rendszer tudomást szerezzen róla

```
UnicastRemoteObject.exportObject(…)
```

A beregisztrálás módja

- Megfelelő biztonsági felügyelő (security manager) installálása
- A távoli objektum osztályának példányosítása
- Bejegyzés
  - pl. rmiregistry-be
  - vagy más szolgáltatóba, pl. JNDI

Biztonsági felügyelő

- Védi a virtuális gépet és az azt futtató rendszert a letöltött kódtól (pl. lokális fájlrendszer elérése)
- Kötelező definiálni, ha azt akarjuk, hogy dinamikus osztályletöltés menjen
  paraméterek, visszatérési értékek, kivételek típusára
- java.rmi.RMISecurityManager
- Olyasmi, mint ami az appletek esetén is van
- Lehet mást is beállítani, vagy jogokat adni egy policy fájllal

`System.get/setSecurityManager()`

Bejegyzés

- Ahhoz, hogy egy kliens szerezni tudjon egy referenciát élete első távoli objektumára
- A többire már tud referenciát szerezni ettől
- Predefinit távoli objektum: RMI registry
- Név alapján lehet tőle távoli objektumokra referenciát kérni
- Szolgáltatás elindítása: rmiregistry
- API hozzá: java.rmi.Naming

registry

- **`rmi://`***számítógép*`:`*port*`/`*név*
- Ha nem adunk meg számítógépet, akkor az aktuális számítógép
- Ha nem adunk meg portot, akkor 1099

- Ha bejegyeztünk egy távoli objektumot, akkor már van egy referencia rá, ezért nem lehet őt felszabadítani szemétgyűjtéssel, ezért a main végetérése után nem áll le a prg.

Java tutorial

Példa

- Olyan távoli objektumot fogunk írni, ami valamilyen feladatot old meg. Az, hogy mi a feladat, ismeretlen akkor, amikor a távoli objektumot elkészítjük. Annyit tudunk csak, hogy leírható egy bizonyos interfésszel. A dinamikus osztálybetöltés gondoskodik a többiről…