Rendszerintegráció és -felügyelet laboratórium (VIMIM309)

Kommunikáció JMS és JMX technológia segítségével

Mérési segédlet

Készítette: Hegedüs Ábel

Utolsó módosítás: 2014. március 5.

Verzió: 1.3

Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

1 Bevezető

A labor során a méréseket végző hallgató a gyakorlatban is megismerkedik a rendszerintegráció és rendszerfelügyelet során használatos módszerekkel és eszközökkel. Végigköveti egy elosztott alkalmazás megvalósításának és felügyeletének legfontosabb lépéseit, ipari környezetben használt integrációs köztes réteg (middleware) technológiák és felügyeleti eszközök használatával. A mérések a következő témakörökhöz kapcsolódnak:

1. Munkafolyamatok megvalósítása Java nyelven
2. Megbízható üzenetküldés IBM WebSphere MQ alapon
3. Kommunikáció JMS és JMX technológia segítségével
4. OSGi szolgáltatások fejlesztése
5. Modell alapú eszközintegráció elosztott környezetben (SDE)
6. Felügyeleti adatok vizuális elemzése
7. Rendszerfelügyelet támogatás komplexesemény-feldolgozással

A jelen mérés során a hallgatók megismerkednek az üzenetsorok kezelésének szabványos módjával JMS segítségével. Ehhez kapcsolódóan betekintést nyernek a JMX keretrendszerbe, amely segítségével konfigurálhatóak az üzenetsorok és a JNDI címfeloldó technológiával, amellyel egyedi azonosítók alapján lehet referenciát kérni a megfelelő objektumokra (pl. üzenetsorok).

2 Java Message Service bevezető

Az előző mérésen használt Websphere MQ alapú üzenetsorok hátránya, hogy az alkalmazások kizárólag az IBM adott keretrendszerén keresztül tudnak kommunikálni. Ehelyett jó lenne, ha az elkészített alkalmazások tetszőleges üzenetsor kezelő keretrendszer felett futtathatóak lennének. Erre ad lehetőséget a Java Message Service (JMS) szabványos alkalmazásprogramozási felület (API).

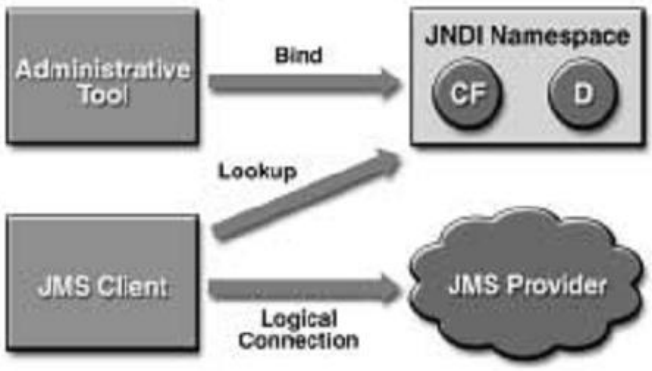
A JMS segítségével az alkalmazások képesek üzentek létrehozására, küldésére, fogadására és feldolgozására. A JMS igyekszik maximalizálni az alkalmazások hordozhatóságát különböző JMS szolgáltatók között egy adott üzenetküldő megoldásban. Segítségével megvalósítható az aszinkron és megbízható kommunikáció.

2.1 Mikor érdemes JMS API-t használni?

Egy vállalati alkalmazás szolgáltató a következő esetekben szokta az üzenetküldő APIt választani egy szorosan csatolt API helyett, ha:

- A szolgáltató azt szeretné, ha a komponensek nem függnek más komponensek interfészeinek információtól, hogy könnyen kicserélhetők legyenek.
- A szolgáltató szeretné. hogy az alkalmazás futni tudjon akkor is, ha nem minden komponense elérhető.
- Az alkalmazás üzleti modellje lehetővé teszi, hogy egy komponens információkat küldjön egy másiknak és folytassa működését anélkül, hogy azonnali választ kapna.

2.2 JMS API architektúra



1. ábra JMS API architektúra

Egy JMS alkalmazás következő részekből áll, ahogy az ábra is mutatja:

- A JMS szolgáltató (provider) implementálja a JMS interfészt, továbbá adminisztratív és vezérlési funkciókat tartalmaz.
- A JMS kliensek olyan Java nyelven írt programok vagy komponensek, amelyek üzeneteket termelnek és dolgoznak fel.
- Az üzenet olyan objektumok, amelyek a kliensek között információt visznek át.
- Az adminisztrált objektumok olyan előre konfigurált JMS objektumok, amelyeket egy adminisztrátor készített és a kliensek használják. Ilyenek a célállomások (destination) és a kapcsolatkezelők (connection factory).
- A natív kliensek olyan programok, amelyek az üzenetküldő rendszer natív API-ját használják a JMS API helyett.

2.3 Üzenetküldő megoldások

A leggyakrabban használt megoldások a pont-pont összeköttetés és a hirdető/feliratkozó módszer.

- Pont-pont összeköttetés (Point-to-point) esetén az alkalmazások az üzenetsorok, küldők és fogadók koncepcióira épülnek. Minden üzenetnek egy fogyasztója van. A küldőnek és a fogadónak nincsenek időzítési függései. A fogadó attól függetlenül megkaphatja az üzenetet, hogy a küldés időpontjában futott-e. A fogadó visszajelez az üzenet sikeres fogadása esetén.
- A hirdetés/feliratkozás (publish/subscribe) esetén a kliensek egy témára küldik az üzeneteket. A rendszer biztosítja, hogy az üzenetek eljutnak az összes feliratkozott klienshez. Ezáltal minden üzenetnek több fogyasztója lehet. A küldők és a feliratkozók között időzítési függőség van. A kliens, amely feliratkozik egy témára, csak olyan üzeneteket kap meg, amelyek az után érkeztek, hogy a kliens feliratkozott. Végül a kliensnek aktívnak kell maradnia ahhoz hogy üzeneteket fogadjon.

2.4 Üzenet feldolgozás

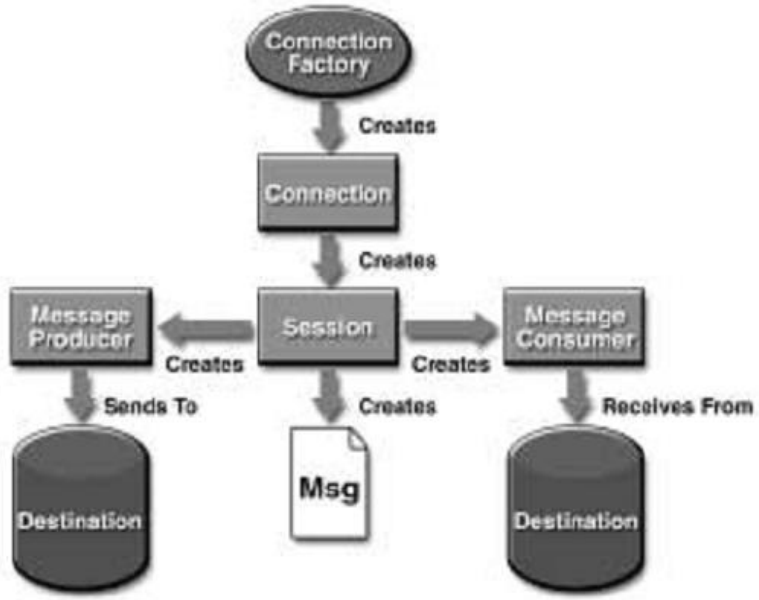
Lehetőség van az üzeneteket szinkron és aszinkron módon fogadni. Szinkron esetben a fogadó explicit módon lekéri az üzenetet egy „receive” metódus segítségével. Aszinkron esetben a kliens egy „message listener”-t regisztrál, amely akkor fut le, ha érkezik egy üzenet. Ilyenkor a JMS szolgáltató meghívja az „onMessage” metódust a regisztrált üzenetkezelőben.

3 Java Naming and Directory Interface

A Java Naming and Directory Interface (JNDI) API egy könyvtár és elnevezési funkcionalitást szolgáltat Java alkalmazásokhoz.

4 JMS API programozási modell

A JMS alkalmazások alapvető építőelemei a következők: adminisztrált objektumok, kapcsolatok, munkamenetek (session), üzenettermelők (producers), üzenetfogyasztók (consumers), üzenetek. Az elemek közötti kapcsolatokat ábrázolja az alábbi ábra:



2. ábra JMS API programozási modell

4.1 Kapcsolatkezelő

A kapcsolatkezelő (connection factory) olyan adminisztrált objektum, amelynek segítségével a kliensek kapcsolatot teremtenek a szolgáltatóval. A kapcsolatkezelő tartalmazza az adminisztrátor által definiált kapcsolat konfigurációs paraméterek halmazát.

```
Context ctx = new InitialContext();
QueueConnectionFactory queueConnectionFactory =
  (QueueConnectionFactory) ctx.lookup("QueueConnectionFactory");
```

5 A mérés elvégzése

A mérés során a hallgatóknak át kell alakítaniuk az második mérésen készült programjukat olyan formába, hogy a folyamat egyes lépései közötti adatáramlás üzenetsorok segítségével legyen megoldva. Az összes csomópont külön Java folyamat legyen (nem csak thread), amelyek kizárólag a JMS API-n kommunikálnak, nincs központi szinkronizációs komponens. A csomópontok között tényleges adatáramlás legyen, ne csak primitív token átadás.

A következő feladatokat kell mindenképpen elvégezni a mérés teljesítéséhez:

- JMX-en keresztül a folyamat csomópontjai közötti kommunikációhoz szükséges egyes üzenetsorok létrehozása.

- Java osztályok átalakítása, lehetőleg a kommunikáció elkülönítése (ahogy előző mérésen is). Általában egy get és egy put metódus megvalósítása szükséges, amely magába foglalja az JMS-el való kommunikációt.
- Folyamat lépéseinek összekötése a megfelelő ki és bemeneti sorok megadásával.
- Egyszerű grafikus megjelenítés, ahol látható az adott csomóponton elérhető adat és léptethető a folyamat, valamint elágazás esetén döntés kiválasztható.
- A folyamat többször is futtatható legyen az JMX kezelő program használata nélkül (ne maradjon futást zavaró üzenet a sorokban).

További feladatok, amelyek közül legalább egy megoldása szükséges a maximális pontszám eléréséhez:

- Pont-pont összeköttetés helyett alakítsátok át a programot hirdetés/feliratkozás alapú működésre.
- Alakítsa át a kezdőcsomópontot úgy, hogy egy újonnan felvett gomb megnyomására kérjen le adatokat a JMX-től a ManagemantFactory osztály segítségével és írja ki őket.
- Valósítson meg egy saját MBean-t, legyen olyan operációja, amellyel egy új folyamatpéldány indítható az attribútumokban beállított adatokkal.

5.1 A mérés kiértékelése

A mérés után leadott jegyzőkönyvben szerepeljen a mérés elvégzéséhez szükséges lépések leírása megismételhető módon. Legyen benne a megvalósított folyamat rövid leírása, azok a műveletek, amelyeket az JMX felületén el kellet végezni, a létrehozott üzenetsorok és paramétereik. Szerepeljen benne az, hogy hogyan és mennyire kellett az eredeti programot megváltoztatni, hogyan valósították meg az egyes csomópont típusokat. Tartalmazzon megfelelő mennyiségű képernyőképet és útmutatót ahhoz, hogy hogyan kell lefuttatni az elkészült programot.

6 Ellenőrző kérdések

A beugró kérdések kizárólag ebben a dokumentumban leírtakat fogják visszakérdezni, a négy linkelt leírás a mérés elvégzéséhez szükséges pluszinformációkat tartalmazzák.

1. Mik a JMS API használatának előnyei az IBM MQSeries használatával szemben?
2. Mi a JNDI rövidítés feloldása, mire használható?
3. Rajzolja fel a JMS API architektúráját?
4. Mutassa be a JMS API programozási modelljét!

7 Segédanyagok

Az alábbi segédanyagok egyrészt bővebb elméleti ismereteket tartalmaznak, másrészt hasznosak lehetnek a mérés elvégzése során.

The JBoss 4 Application Server Guide 2.1 fejezet: An introduction to JMX

Java™ Message Service API Tutorial 1, 2, 3 fejezetek: Overview, Basic Programming Concepts, The JMS API Programming Model

Java Naming and Directory Interface™ Application Programming Interface 3, 4, 5 fejezetek: Overview of the Architecture, Fundamentals, Overview of the Interface

Java Message Service API Architecture, JMS Message Model, JMS Common Facilities, JMS Point-to-Point Model, JMS Example Code