WebLogic JMS -
Messaging für Fortgeschrittene

Marcel Amende
ORACLE Deutschland B.V. & Co. KG
Düsseldorf

Schlüsselworte:

WebLogic, JMS, Queues, Topics, Unit-of-Order, Unit-of-Work, Store-and-Forward, Clustering

Einleitung

Der JMS-Standard wurde von SUN ins Leben gerufen, um Java Applikationen den Zugriff auf Enterprise Messaging Systeme zu ermöglichen und somit asynchron zu kommunizieren. Mittlerweile ist JMS unter JSR-914 ein fester Bestandteil des Java Community Prozesses und der JEE Plattform.

WebLogic JMS bietet zahlreiche Zusatzfunktionen, die über den Umfang der JMS 1.1 Spezifikation hinausgehen. Neben den Grundkonzepten der Messaging Plattform werden hier insbesondere Themen besprochen, die WebLogic JMS zur ersten Wahl für den anspruchsvollen Einsatz auf Unternehmensebene machen: "JMS Clustering", "Distributed Destinations", "Store-and-Forward", "Unit-of-Order" und „Unit-of-Work“.

JMS Clustering

Messagingsysteme in kritischen Einsatzszenarien, wie sie häufig z.B. in der Finanzindustrie anzutreffen sind, müssen hochverfügbar und ausfallsicher sein, um strenge Qualitätssicherungsvereinbarungen einhalten zu können. Da es sich beim JMS Standard um eine reine API handelt, sind die hierfür nötigen Mechanismen nicht spezifiziert, sondern den Anbietern der Implementierung überlassen. Somit hat WebLogic JMS als ausfallsichere JMS-Implementierung sicherzustellen, dass alle JMS Resourcen (Queues, Topics und Connection Factories) im Cluster betreibbar sind und die Nachrichtenlast auf die einzelnen JMS Entpunkte verteilt werden kann.

In einem WebLogic Cluster können herkömmliche Queues und Topics als verteilte Endpunkte konfiguriert und deren physikalische Endpunkte auf mehrere unterschiedliche Server verteilt werden. Diese bilden dann in Summe eine hochverfügbare Messaging-Schicht. Da auch die Connection Factories auf diesen Wege verteilt werden, können Verbindungsanfragen von JMS Clients und deren Nachrichtenverkehr unter Anwendung von bestimmten Lastkriterien auf die zugehörigen Endpunkte geleitet und bei Ausfall einzelner Endpunkte sogar umgeleitet werden. Aus Sicht des Clients erscheinen verteilte Endpunkte genauso, wie gewöhnliche, nicht verteilte Endpunkte. Daher sind für einen Wechsel von einem einzelnen auf verteilte Endpunkte keine Änderungen am Code des Clients nötig.

Bei Verwendung von Queues nutzen Clients den JNDI-Namen der verteilten Queues, um über den Cluster-weiten JNDI-Baum des WebLogic den Sendepunkt zu identifizieren. Für jede einzelne zu sendende Nachricht wird nun über einen Lastverteilungsalgorithmus (Round-Robin oder Random) eine Queue des Clusters als Ziel identifiziert. Auf Empfangsseite bleiben die einzelnen Queues des Clusters an einen bestimmten Endpunkt gebunden. Um zu vermeiden, dass Nachrichten in Queues ohne aktiven oder mit nicht erreichbarem Empfänger verbleiben, kann eine Weiterleitungsverzögerung

konfiguriert weden, nach der Nachrichten automatisch an Queues mit aktiven Empfängern weitergeleitet werden.

Im Gegensatz dazu werden bei Verwendung von Topics die von JMS Clients gesendeten Nachrichten immer an alle Mitglieder eines verteilten Topics geleitet, unabhängig davon, ob der JNDI-Name des verteilten Topics oder der eines physikalischen Topics genutzt wird. Die Subscriber bleiben an ein physikalisches Topic gebunden und halten ihre Verbindung aufrecht. Wenn das Topic nicht mehr verfügbar sein sollte, erhält der Subscriber-Client eine JMSException. Bei einer Kombination von persistenten und nicht-persistenten Topics in einem Cluster werden die persistenten Topics bevorzugt behandelt, um den Verlust von Nachrichten zu vermeiden. Sollten einzelne Topics im Moment des Eintreffens neuer Nachrichten nicht verfügbar sein, werden diese Nachrichten auf den persistenten Knoten vorgehalten und an die restlichen Topics weitergeleitet, sobald diese wieder verfügbar sind.

JMS Unit-of-Order

Die JMS Spezifikation behandelt die geordnete Zustellung von Nachrichten nur in sehr grundlegender Form: Sie ermöglicht nur die Einhaltung der Nachrichtenreihenfolge zwischen genau einem Produzenten und einem Konsumenten. Weiterführende Themen, wie der Umgang mit mehreren Konsumenten, mehreren Produzenten, die Nachrichtenwiederherstellung, das Zurückrollen von Transaktionen und die Verwendung eigener Sortierschlüssel werden nicht behandelt.

Dies kann in vielen Szenarien zum Problem werden: Nehmen wir an, wir bestellen ein Buch auf elektronischem Wege. Unsere Bestellung wird in eine Queue eingestellt. Wir stornieren die Bestellung. Das Storno wird in dieselbe Queue eingestellt. Ein Message Driven Bean entnimmt die Bestellung aus der Queue, ein anderes Bean entnimmt das Storno. Das Storno-Bean wird schneller ausgeführt und schickt das Storno an das Datenbanksystem, in dem die zugehörige Bestellung noch nicht vorliegt. Das andere Bean schreibt danach die Bestellung in die Datenbank. Als Folge werden wir unerwünschterweise ein Buch bekommen!

WebLogic Message Unit-Of-Order vermeidet Probleme dieser Art, indem es Nachrichten von einem oder mehreren Produzenten in eine Einheit gruppieren kann, in der die Nachrichten streng sequentiell in der Reihenfolge ihrer Erstellung abgearbeitet werden. Die Verarbeitung weiterer Nachrichten der Einheit wird dafür blockiert, bis die Abarbeitung der vorhergehenden Nachricht durch „Acknowledged“, „Commited“, „Rolled Back“ oder „Recovered“ bestätigt ist.

Im Falle unserer Buchbestellung wird das Storno-Bean blockiert, bis das Order-Bean die erfogreiche Verarbeitung in die Datenbank durch ein „Acknowledge“ bestätigt.

In verteilten Szenarien stellt Unit-of-Order sicher, dass alle Nachrichten einer Einheit zu demselben Endpunkt geliefert werden.

JMS Unit-of-Work

Viele Applikationen benötigen eine noch stiktere Sicht auf Nachrichtengruppen, als sie die “Unit-of-Order” Funktionalität mit ihrer geordeten Nachrichtenreihenfolge bietet. Bei „Unit-of-Work“ werden JMS Nachrichten als Gruppe betrachtet, die auch als solche von einem JMS Konsumenten verarbeitet werden kann. Ein Produzent kann zum Beispiel einen Satz von Nachrichten definieren, die unterbrechungsfrei und als eine Einheit an einen Abnehmer geliefert werden sollen. Dabei werden Abnehmer nicht blockiert, d.h. sie müssen nicht auf den vollständigen Abschluss einer Nachrichtengruppe warten, wenn andere, vollständige Gruppen bereits vorliegen.

Hier sehen wir ein Codebeispiel für einen JMS Produzenten, der 100 JMS Nachrichten als „unit-of-Work“ sendet:

```
for(int i=1; i<=100; i++){
      sendMsg.setStringProperty("JMS_BEA_UnitOfWork", "My Unit");
      sendMsg.setIntProperty("JMS_BEA_UnitOfWorkSequenceNumber", i);
      if(i == 100){
            System.out.println("Setzen des Endkennzeichens: " + i);
            sendMsg.setBooleanProperty("JMS_BEA_IsUnitOfWorkEnd", true);
      }
      qSender.send(sendMsg, DeliveryMode.PERSISTENT, 7, 0);
}
```

Der Konsument entnimmt alle der Nachrichten der Gruppe in einem Vorgang als Array aus der Queue:

```
msg = qReceiver.receive();
System.out.println("Empfangene Nachrichtengruppe: " + msg);
ArrayList msgList = (ArrayList)(((ObjectMessage)msg).getObject());
numMsgs = msgList.size();
System.out.println("Anzahl der Nachrichten in der Gruppe: " + numMsgs);
```

JMS Store-and-Forward

JMS „Store-and-Forward“ ist eine sehr nützliche Funktionalität, wenn Nachrichten von einem WebLogic Server oder von einer Java Applikation in eine entfernte JMS Queue eingestellt und dabei Ausfallzeiten des entfernten Systems, z.B. auf Grund von Netzwerkproblemen, toleriert werden sollen. In diesem Falle würden die Nachrichten auf dem lokalen System persistiert und weitergeleitet, sobald der entfernte Endpunkt wieder erreichbar ist. Für den Client erscheint „Store-and-Forward“ in jedem Fall wie ein gewöhnlicher JMS Endpunkt.

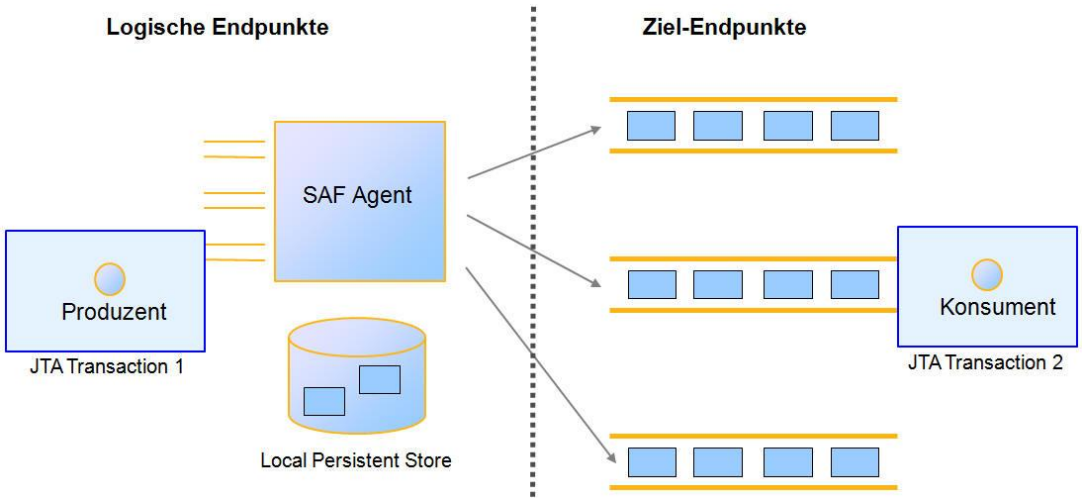

Abb. 1: JMS Store-and-Forward

Ein gutes Beispiel für die Verwendung dieser Funktionalität sind Kassensysteme im Handel: Kassenbons sind möglichst zeitnah von den verschiedenen, räumlich weit verteilen Geschäften in das

zentrale Warenwirtschaftssystem zu übertragen. Die Netzwerkanbindung erfolgt oft über gewöhnliche, nicht ausfallsichere DSL-Verbindungen. Für das Kassensystem, eine Java („Thick“) Client Implementierung, ist „Store-and-Forward“ wie ein lokaler JMS Endpunkt verwendbar. Tatsächlich werden JMS Nachrichten aber an eine zentrale Nachrichten-Queue weitergeleitet. Ist der Serverseitige Endpunkt nicht erreichbar, werden die Nachrichten lokal im Filesystem zwischengespeichert und weitergeleitet, sobald der Client sich wider mit dem Server verbinden kann.

Kontaktadresse:

Marcel Amende
ORACLE Deutschland B.V. & Co. KG
Hamborner Str. 51
D-40472 Düsseldorf

Telefon: +49 (0) 211-74839 539
E-Mail Marcel. Amende@oracle.com
Internet: www.oracle.de