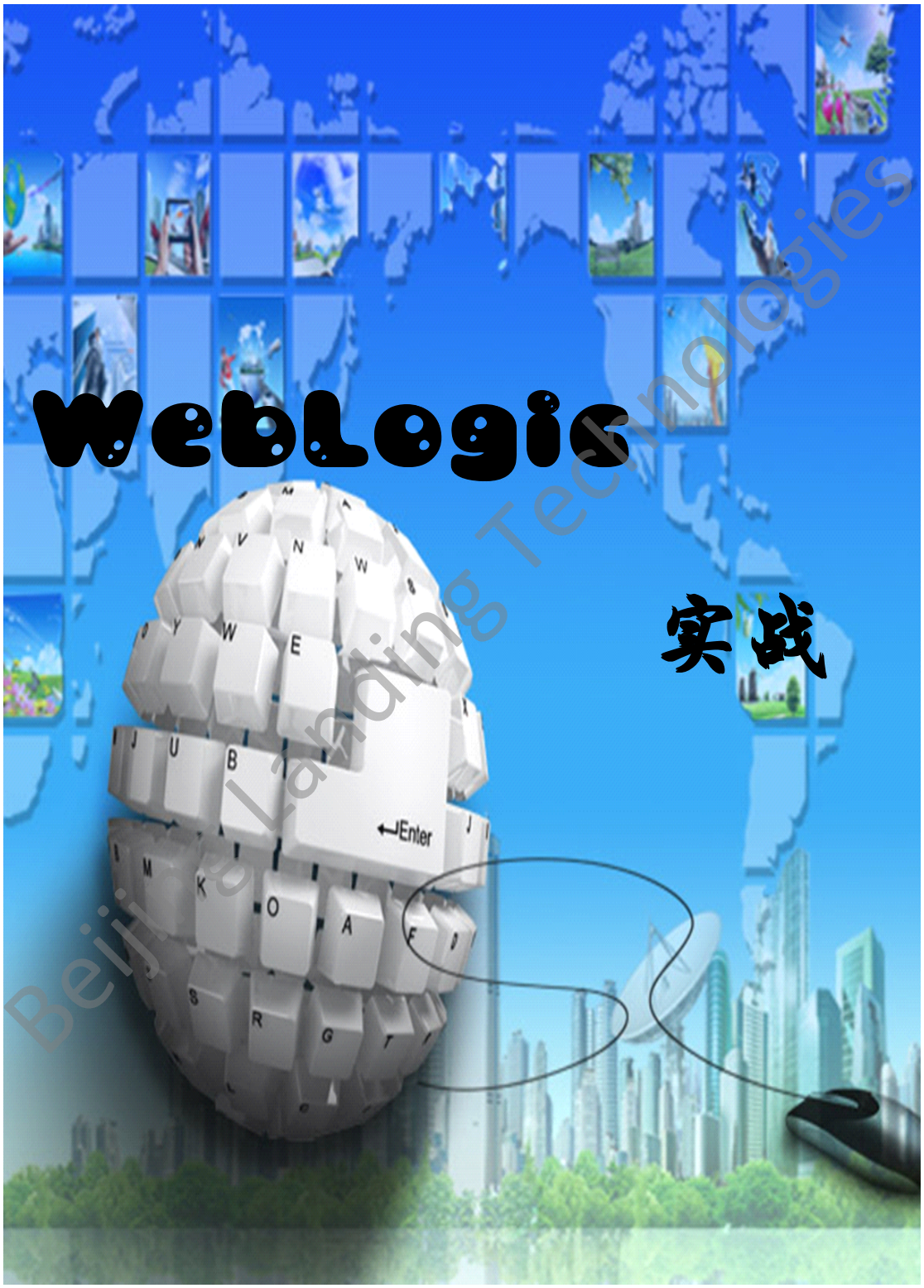
联动北方
LANDING

企业级 IT 运维宝典系列



联动北方 出品

目　录

第 21 章 JMS 消息重发故障

21.1 问题描述

JMS 消息被多次重新发送给接收器/订阅用户。一旦有消息到达 JMS 目标，JMS 服务器试图将其发送给有效用户然后等待用户确认。从 JMS 服务器的角度来看，在 JMS 消息被确认收到以前，该消息就不被视为“已发送”。无论什么原因，只要 JMS 服务器没有收到消息确认，它就会重新发送消息。

JMS 重新发送故障症状包括：

1、 JMS 接收器的 onMessage() 方法被多次执行。

2、 消息接受了多次处理。

3、 服务器的运行速度可能会因多次尝试发送一条或多条“恶性”消息而下降，此类消息：

- ☹ 是接收器拒绝接收的消息；
- ☹ 通常因实际消息存在问题而遭到拒绝。

4、 消息可能因暂时性的资源短缺或资源无法使用而遭到拒绝

21.2 问题定位

21.2.1 Java Message Service(JMS)简介

JMS 即 Java 消息服务（Java Message Service）应用程序接口，是一个 Java 平台中关于面向消息中间件（MOM）的 API，用于在两个应用程序之间或分布式系统中发送消息，进行异步通信。Java 消息服务是一个与具体平台无关的 API，绝大多数 MOM 提供商都对 JMS 提供支持。

21.2.2 为什么 JMS 消息会被重新发送

如果 JMS 服务器认为消息未能成功发送，就会发生 JMS 重新发送，发送的 JMS 消息只有得到确认后，才会将该消息视为已发送。如图消息 3 已发送，对方收到并返回确认信息，则此时 JMS 服务器就认为 3 已发送，而 1、2、5、6 均未发送。

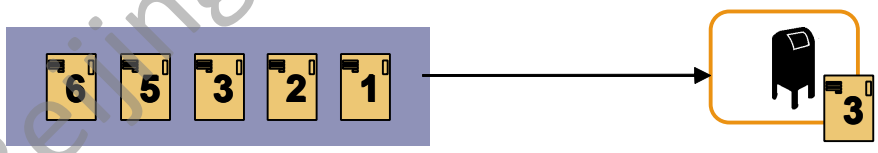


图 21-1

JMS Server 可能会因下列任一原因而重新发送消息：

5、 接收器或 MDB 的 onMessage()方法抛出 Java 异常。

例如：

```
java.lang.Error
java.lang.RuntimeException
```

示例 21-1

6、 出现‘恶性’消息，即这样一些消息：

- ☹ 格式不正确并因此遭到拒绝的消息；
- ☹ 或因暂时性资源中断而遭到拒绝的消息。

7、 客户端接收器（显式 Acknowledge）未调用 session.acknowledgement()

8、 因出现下列情况而使事务被隐式或显式回滚：

- ☹ 参与事务的一个 MDB 因某种原因（例如，下游数据库错误）而失败；
- ☹ MDB 是事务性的，但因处理消息时花费的时间过长而引发了超时；
- ☹ 调用了 MDB 的 onMessage()方法中的 ejbcontext.setRollbackOnly()（仅限 CMT）；
- ☹ 独立接收器调用了 session.recover()。

注：JMS 消息重新发送问题大多是由应用程序编码错误引起的！

21.2.3 JMS 重新发送故障的两种类型

- 在不应重新发送消息时重新发送消息；
- 在应该重新发送消息时未重新发送消息。

21.2.4 JMS 重新发送模式问题

9、 “JMS 重新发送模式”与下列情况下出现的问题有关 ：

- JMS 服务器认为消息的第一次发送不成功，JMS 服务器随后再次发送该消息；
- JMS 服务器认为第一次发送成功，于是不再重新发送。

10、发送的 JMS 消息在接受以下操作后视为已发送：

- 确认；
- 在提交了周围事务的情况下，消息可能会得到确认。

21.3 JMS 确认

JMS 消息传送给用户后，JMS 服务器会等待用户的确认，如果 JMS 服务器未收到确认，会尝试重新发送消息。

确认的途径如下面所述：

21.3.1 事务会话

21.3.1.1 使用 JMS 事务会话的操作

- 将在内部启动一个本地事务，范围限制在会话内的发送/接收操作；
- 如果接收器对会话进行回滚调用，JMS 服务器将重新发送消息。

21.3.1.2 JMS 事务会话的适用范围限制

- 不能参与外部事务，如 Bean 管理的或容器管理的（JTA）事务；
- 在会话范围外不起作用。

21.3.2 容器管理的事务

CMT 是 Container-Managed Transaction 的缩写，即容器管理事务。

CMT 的工作方式如下：

- 容器开始事务；
- 调用 onMessage()方法，将消息传递给 MDB；
- 如果 onMessage()方法成功返回，容器将提交该事务；
- 如果 onMessage()方法返回异常，容器将回滚该事务。

如果事务被回滚，JMS 服务器会立即将消息重新发送给下一位有效用户。

21.3.3 Bean 管理的事务

BMT 是 Bean—Managed Transaction 的缩写，即 Bean 管理事务，另外，MDB 是 Message Driven Bean 的缩写，即消息驱动 EJB。

BMT 的一些说明如下：

- 依赖 MDB 应用程序代码来启动、提交或回滚与消息处理有关的事务；
- 也称作用户事务；
- 适用于 MDB 用户和独立用户。

另外，在 MDB 内部启动事务时：

- 在目标（队列或主题）获取消息不是事务的组成部分；
- 事务结果对重新发送没有影响，即使事务被回滚也是如此。

21.3.4 设置确认模式

一般在创建 JMS 会话时设置，例如：

```
QueueSession qs=qconn.CreatQueueSession(false,Session.Auto_ACKNOWLWDGE)
```

示例 21-2

确认模式的选项一般有：

- AUTO_ACKNOWLEDGE

当客户成功的从 receive 方法返回的时候，或者从 MessageListener.onMessage()方法成功返回的时候，会话自动确认客户收到的消息。

- DUPS_OK_ACKNOWLEDGE

该选择只是会话迟钝确认消息的提交。如果 JMS provider 失败，那么可能会导致一些重复的消息。

如果是重复的消息，那么 JMS provider 必须把消息头的 JMSRedelivered 字段设置为 true。

- CLIENT_ACKNOWLEDGE

接收器必须显式调用 message.acknowledge() ，才能确认消息。在这种模式中，确认是在会话层上进行：确认一个被消费的消息将自动确认所有已被会话消费的消息。

比如，如果一个消息消费者消费了 10 个消息，然后确认第 5 个消息，那么所有 10 个消息都被确认。

- NO_ACKNOWLEDGE

不确认消息。

对于独立的异步监听器，如果消息未得到确认，则只会将消息重新发送一次；如果重新发送失败，则消息将从 JMS Server 中隐式删除。

21.4 诊断 JMS 重新发送问题

21.4.1 应用程序设计

使用程序设计文档或其他工件确定应用程序应如何处理消息，例如：

11、接收器是 MDB 还是独立的同步或异步接收器；

12、使用何种类型的事务处理，比如 CMT、BMT，或者事务会话；

13、应进行何种类型的确认；

14、在什么条件下执行提交、回滚或确认；

15、确定设置了哪些处理超时，以及是否在任何情况下均可实现。

21.4.2 应用程序代码诊断

1. 记录接收器 onMessage()方法收到的每条消息：

   - 消息 ID
   - 该消息是否被重新发送
   - 如果消息数量很大，则只记录重新发送的消息

   比如，在 onMessage()方法中实现实现发送记录的示例代码如下：

```
public void onMessage(javax.jms.Messsage msg)
{
  try{
    System.out.println(" Msg ID: " + msg.getJMSMessageID());
    System.out.println("Is Message redelivered:"
                       + msg.getJMSRedelivered());
    ...
  }
}
```

示例 21-3

2. 捕捉、堆栈跟踪和再次抛出导致 onMessage() 方法不能成功返回的异常

   比如，捕捉异常的示例代码如下：

```
public void onMessage(Message msg) {
    try {
      TextMessage tm = (TextMessage) msg;
      String text = tm.getText();
      System.out.println("Msg: "+text+" Msg ID:"+msg.getJMSMessageID());
```

```
        . . .
      }
      catch(JMSException ex) {
        ex.printStackTrace();
      }
      catch(java.lang.RuntimeException ey) {
        ey.printStackTrace();
        throw ey;
      }
      catch(java.lang.Error ez) {
        ez.printStackTrace();
        throw ez;
      }
  }
```

示例 21-4

3. 记录下列各项的其他重要事件：

- BMT 和事务会话，事务在事务会话中开始、提交和回滚；
- 确认模式，确认和 NACK 将在该模式下被调用；
- CMT，退出 onMessage() 方法时，记录事务的状态；
- 对事物有影响的调用，如对 ejbcontext.setRollbackOnly() 或 session.recover()的调用。

21.4.3 JMS 调试

使用下列 JMS 调试标志,可以收集有关所执行的 JMS 操作的更多信息：

- DebugMessagePath 可帮助确定消息是否被重新发送

同一消息被两次发送给接收器示例：

```
<Feb 3, 2004 5:01:25 PM PST> <Debug> <JMS> <BEA-040002>
<JMS Debugging MSG_PATH!
BACKEND/BEQueue: Assigning to the backend consumer,
message ID:P<802808.1075856485894.0>>
<Feb 3, 2004 5:01:25 PM PST> <Debug> <JMS> <BEA-040002>
<JMS Debugging MSG_PATH!
BACKEND/BEQueue: Adding backend session's unacked message list,
message ID:P<802808.1075856485894.0>>
<Feb 3, 2004 5:01:25 PM PST> <Debug> <JMS> <BEA-040002>
<JMS Debugging MSG_PATH!
BACKEND/BEQueue: Dispatching to the frontend,
message ID:P<802808.1075856485894.0>>
<Feb 3, 2004 5:01:25 PM PST> <Debug> <JMS> <BEA-040002>
<JMS Debugging MSG_PATH!
FRONTEND/FESession (id: <576180353183090550.27>) :
Pushing to the client, message ID:P<802808.1075856485894.0>>
>>Print From the onMessage method: I am the MESSAGE,
MsgID: ID:P<802808.1075856485894.0>
```

```
...
<Feb 3, 2004 5:02:05 PM PST> <Debug> <JMS> <BEA-040002>
<JMS Debugging MSG_PATH!
BACKEND/BEQueue: Assigning to the backend consumer,
message ID:P<802808.1075856485894.0>>
<Feb 3, 2004 5:02:05 PM PST> <Debug> <JMS> <BEA-040002>
<JMS Debugging MSG_PATH!
BACKEND/BEQueue: Adding backend session's unacked message list,
message ID:P<802808.1075856485894.0>>
<Feb 3, 2004 5:02:05 PM PST> <Debug> <JMS> <BEA-040002>
<JMS Debugging MSG_PATH!
BACKEND/BEQueue: Dispatching to the frontend,
message ID:P<802808.1075856485894.0>>
<Feb 3, 2004 5:02:05 PM PST> <Debug> <JMS> <BEA-040002>
<JMS Debugging MSG_PATH!
FRONTEND/FESession (id: <576180353183090550.34>) :
Pushing to the client, message ID:P<802808.1075856485894.0>>
>>Print From the onMessage method: I am the MESSAGE,
MsgID: ID:P<802808.1075856485894.0>
```

示例 21-5

- DebugJMSXA 将有助于确定消息是都因事务相关问题而被重新发送

通过 DebugJMSXA 输出示例：

```
    <Feb 3, 2004 5:16:10 PM PST> <Debug> <JMS> <BEA-040002> <JMS Debugging XA !
XA(25421973,1007511,0000013BC2697F28EDB9) >RM-rollback() >
    <Feb 3, 2004 5:16:10 PM PST> <Debug> <JMS> <BEA-040002> <JMS Debugging XA !
XA(25421973,1007511,0000013BC2697F28EDB9) >TE-recv-startRollback() (TE-recv
hash=31838215 xid=0000013BC2697F28EDB9 mId=<712671.1075857330638.0>
queue=TestDest)>
    <Feb 3, 2004 5:16:10 PM PST> <Debug> <JMS> <BEA-040002> <JMS Debugging XA !
XA(25421973,1007511,0000013BC2697F28EDB9) <TE-recv-startRollback() (TE-recv
hash=31838215 xid=0000013BC2697F28EDB9 mId=<712671.1075857330638.0>
queue=TestDest)OK>
    <Feb 3, 2004 5:16:10 PM PST> <Debug> <JMS> <BEA-040002> <JMS Debugging XA !
XA(25421973,27221567,0000013BC2697F28EDB9) >RM-rollback() >
```

示例 21-6

应用上述标志的方法：

- 将它们添加到 config.xml 文件的 ServerDebug 部分；
- 在服务器启动脚本中使用-D 标志。

21.5 检查“恶性”消息

1. 检查用于避免‘恶性’消息的 JMS 目标参数如下：

- Redelivery Delay Time

— 指在尝试重新发送消息前消息被搁置的时间。

- Redelivery Limit

— 指最多可以尝试重新发送消息的次数，尝试次数用尽后，消息将被移至错误目标。

- Error Destination

— 指已达到其重新发送极限的消息的目标目的地（队列或主题）。

2. 检查可能会引发“恶性”消息的条件，如：
   - 数据库或其他后端资源变为不可用时发生的情况；
   - 是否在任何情况下都会出现格式不正确的消息。

21.6 故障排除检查清单

1. 与应用程序开发团队协作，以确保：
   - 正确处理异常；
   - 正确处理确认和、或事物；
   - 安排了足够的记录操作来捕捉不正确的处理。
2. 由于重新发送问题大多由应用程序编码错误导致，所以注意以下两点：
   - 请先确保应用程序工作正常；
   - 如果确信应用程序工作正常，次要可疑点可能就是 JMS 实现本身。