用 HornetQ 实现异步消息传输

关 辉[1],李四海[2]

(1.苏州市职业大学,江苏 苏州 215104;2.湖北省轻工业技工学校,湖北 武汉 430070)

摘 要:在电子商务/电子政务的应用中,很多处理都需要以异步方式进行,JMS 规范可以支持应用程序间消息的异步传递。详细论述了利用实现 JMS 规范的组件 HornetQ 来完成异步消息传输的过程。

关键词:JMS;异步;消息;JBoss;HornetQ

中图分类号:TP301 文献标识码:A 文章编号:1672-7800(2010)12-0033-02

0 引言

在一些应用程序中许多处理并不是都能够在很短的时间内执行完毕,因此从使用者的角度来看,对于运行时间相对较长或运行时间不确定的处理,就应当断开与应用程序的连接而以异步方式运行。这就意味着在请求发出之后,调用就立即返回,发出此调用的系统并不需要等待该请求执行完毕。采用这种异步处理方式有许多优点,其最主要的特点就是能够实现系统之间的松散耦合。它可以切断系统中不同处理之间的连接,让它们以不同的速度运行,因而有助于在分布性及伸缩性方面获得最大的灵活性。J2EE 中的 JMS 技术能够非常容易地实现企业系统之间的松散耦合,并且能够通过消息服务提供商实现可靠、健壮和异步的消息传递,它具有 Java 语言特有的平台无关性,能够满足分布式环境下异构平台的交互行为的需要,是最好的选择。

1 用HornetQ实现异步消息传输

JMS 只是一个技术规范,不是直接可以使用的软件。要想进行 JMS 应用程序开发,必须选择合适的 JMS 提供者(Provider)。近年来,开放源代码的 JMS 提供者应用越来越广泛,如:mom4j、Openjms、ActiveMQ 和 HornetQ 等。其中,HornetQ 是 JBoss 的一个子项目,JBoss 被 RedHat 公司收购后,于去年宣布将它用来实现 JMS 规范的一个组件——JBoss Messaging 2.0 更名为 HornetQ,并把它剥离出来成为一个独立项目。HornetQ 可以用于构建多协议、嵌入式、高性能、集群及异步的消息系统,既能用于小型应用,也能用于大规模的企业消息系统中。HornetQ 不依赖于任何 JBoss 应用服务器组件,事实上它的核心只依赖于 JDK。既可以将 HornetQ 集成到 JBoss 应用服务器中作为 JMS 提供者,也可以在 JBoss 应用服务器之外以独立的方式运行,功能上不会受到任何影响,甚至还可以通过依赖注入框架如:Spring 或者 Google Guice 对其进行实例化。除此以外,还可以将 HornetQ 直接嵌入到自己的应用中。HornetQ 的消息系统性能指标打破了 SPEC jms2007 工业标准基准记录,在同样的基准测试下,它比其它开源 JMS 提供者要快很多,性能也更好。因此,用 HornetQ 来实现消息的异步传输是一种很好的选择。

下面就以点对点模式为例说明用 HornetQ 实现消息的异步传输的过程。

JMS 使用 JNDI 来存储连接工厂和队列,因此要使用任何一种消息模式,首先必须创建 JNDI context。代码如下所示:

```
private static Context getInitialContext() throws NamingException
{
  Hashtable props=new Hashtable();
  props.put(Context.INITIAL_CONTEXT_FACTORY, "org.jnp.interfaces.NamingContextFactory");
  props.put (Context.URL_PKG_PREFIXES, "org.jboss.naming:org.jnp.interfaces");
  props.put(Context.PROVIDER_URL, "jnp://localhost:1099");
  //localhost 在实际应用时应改为服务器端 IP 地址
  Context ctx=new InitialContext(props);
  return ctx;
}
```

1.1 消息的发送

JMS 客户端负责发送消息,具体步骤如下:

(1)创建用于 JNDI 查询的 Context。InitialContext ctx=ge-

作者简介:关辉(1973-),男,湖北襄樊人,硕士,苏州市职业大学讲师,研究方向为 Web 应用技术、网络数据集成;李四海(1969-),男,湖北老河口人,湖北省轻工业技工学校讲师,研究方向为 Web 应用技术。

tInitialContext()。

(2)得到 QueueConnectionFactory。QueueConnectionFactory qconFactory =(QueueConnectionFactory)ctx.lookup ("ConnectionFactory")。

(3) 得到 Queue。Queue queue=(Queue)ctx.lookup("queue/testQueue")。

(4)从 QueueConnectionFactory 中创建 QueueConnection。QueueConnection qcon=qconFactory.createQueueConnection()。

(5)从 QueueConnection 中创建 QueueSession。QueueSession qsession =qcon.createQueueSession (false,Session.AUTO_ACKNOWLEDGE)。

(6)从 QueueSession 中创建 QueueSender。QueueSender qsender=qsession.createSender(queue)。

(7)从 QueueSession 中创建 Message。TextMessage msg=qsession.createTextMessage()。

(8)将消息写入到 Message。msg.setText(message)。

(9)发送消息。qsender.send(msg);System.out.println("消息发送成功!")。

(10)关闭消息。qsender.close();qsession.close();qcon.close()。

其中,第 2、3 步的 ConnectionFactory 和 queue/testQueue 是 JNDI 名字,可以在 hornetq-jms.xml 文件中进行配置。第 5 步在创建 QueueSession 时,createQueueSession()方法的第一个参数设置为 false 表示 Session 是 non-transactional(非事务性)的,第二个参数设置为 AUTO_ACKNOWLEDGE 表示 Session 将自动地确认收到一则消息。在第 9 步发送消息时,send()方法中还可以设置发送模式 (PERSISTENT 或 NON_ PERSISTENT)、优先级(0-9)和生存时间参数。

1.2 消息的接收

JMS 服务器端随时监听客户端发来的消息队列,若有消息则接收下来。在 HornetQ 中,可以通过实现 MessageListener 接口的对象来完成消息监听器的功能。每当消息到达时,自动调用 MessageListener 接口的 onMessage()方法。

JMS 服务器端监听并接收消息的具体实现步骤如下:

```
//创建实现 MessageListener 接口的类 MessageReceiver,完成消息监听器的功能。
public class MessageReceiver implements MessageListener{
//实现 onMessage()方法
public void onMessage(Message m){
  try{
……    //当消息到来时可以通过这部分代码对消息进行一些处理
System.out.println("消息成功接收!");
   } catch(Exception ex) {
System.out.println("消息接收失败!");
ex.printStackTrace();
}
}
public static void main(String[] args) {
try {
…… //此部分代码与消息发送部分的 1-5 步相同
QueueReceiver qreceiver=qsession.createReceiver (queue);  //从 QueueSession 中创建 QueueReceiver
qreceiver.setMessageListener(this);   //设置监听器
qcon.start();    //准备接收消息
} catch(Exception ex) {
ex.printStackTrace();
}
}
}
```

2 结束语

目前在电子政务/电子商务应用中很多处理都需要以异步方式进行,利用 HornetQ 来实现异步消息传输可以使系统轻松拥有松耦合性、异步性和可靠性,并具有良好的可移植性。

参考文献:

[1] 安勇,侯晓霞.基于 JMS 构建异步应用系统[J].计算机应用与软件,2007(10).

[2] [美]MARK HAPNER,RICH BURRIDGE.Java 消息服务 API 参考指南[M].康博,译.北京:清华大学出版社,2002.

[3] 李华飚,郭英奎.Java 中间件开发技术[M].北京:中国水利水电出版社,2005.

(责任编辑:周晓辉)

Asynchronous Message Transfer with HornetQ

Abstract: In E-commerce/E-government applications,many processing needs in an asynchronous mode.Java Message Service (JMS) specification can support asynchronous message transfer between applications. This paper introduces the architecture, message transfer mode and message interface of JMS,and on this basis discuss in detail the process of asynchronous message transfer using HornetQ-a component of achieving JMS specification.

Key Words: Java Message Service(JMS); Asynchronous; Message; JBoss; HornetQ