Weblogic JMS 编程

限于译者的水平和能力，错误和不当之处在所难免，希望广大读者给予批评指正。

Java 研究组织
www.javaresearch.org
疾风摩郎
dengke@javaresearch.org
2002 年 6 月 28 日

第二篇 WebLogic JMS 基础

下列章节描述了 WebLogic JMS 的组件和特性：

- 消息通信模型
- WebLogic JMS 类
- 连接工厂
- 连接
- 会话
- 目的
- 分布式目的
- 消息生产者和消息消费者
- 消息
- 服务器会话池工厂
- 服务器会话池
- 服务会话
- 连接消费者

一. 消息通信模型

JMS 支持两种消息通信模型：点到点（point-to-point）（PTP）模型和发布/订阅（Pub/Sub）模型。除了下列不同之外，这两种消息通信模型非常地相似：

- PTP 模型规定了一个消息只能有一个接收者。
- Pub/Sub 模型允许一个消息可以有多个接收者。

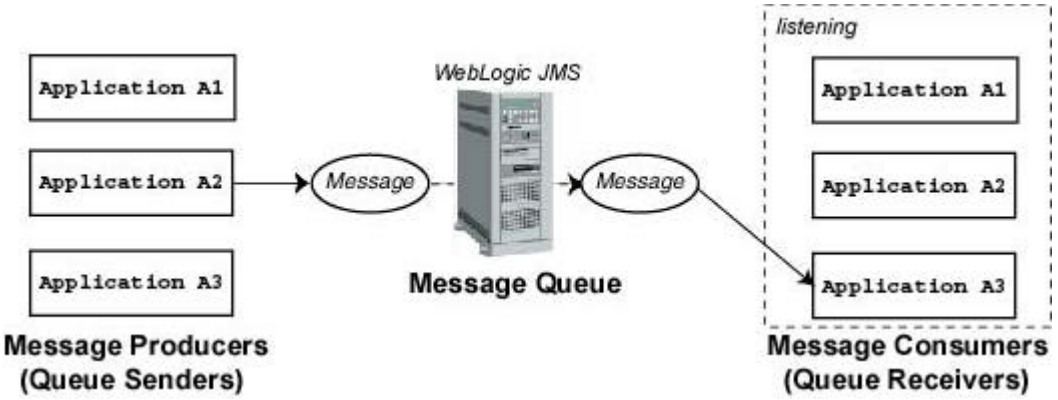
每个模型对应的类继承了相同的父类。举个例子，PTP 类 javax.jms.Queue 和 Pub/Sub 类 javax.jms.Topic 都继承了 javax.jms.Destination 类。

每个消息模型会在随后详细说明。

注意：在每个消息通信模型中，术语生产者和消费者被分别用来作为发送和接收消息的应用程序的一般性说明。然而，对于每个明确的消息通信模型，当提及生产者和消费者概念时，会有明确的唯一性的术语。

1. 点到点（point-to-point）模型

点到点模型授权应用程序发送消息给其他应用程序。点到点应用程序使用指定的队列发送和接收消息。一个队列发送者（生产者）发送消息到一个指定的队列。一个队列接收者（消费者）从这个指定的队列中接收消息。下面是 PTP 模型的图解：

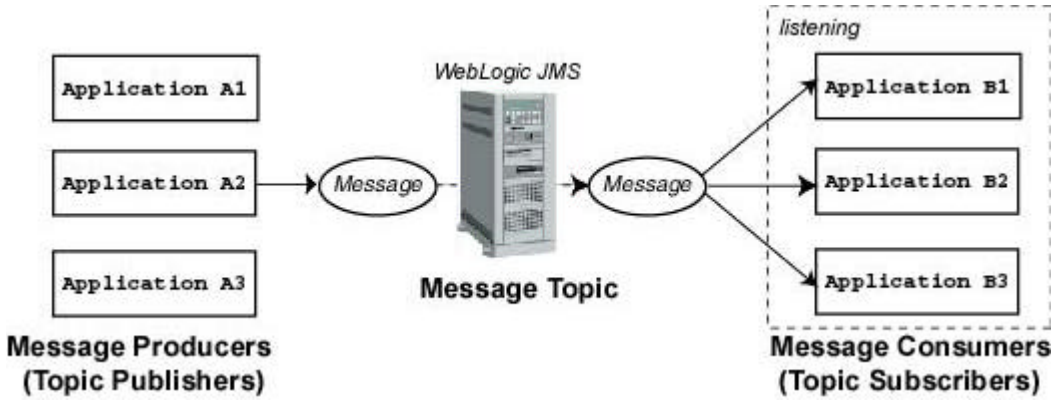


一个消息队列可以有多个队列发送者和队列接收者，但是一条消息只能被递送给唯一的一个队列接收者。

如果多个队列接收者同时在监听一个队列，基于先到先服务的原则，WebLogic JMS 哪个队列接收者能接收下一条消息。如果没有队列接收者在监听这个队列，那么消息会保存在这个队列中，直到有一个队列接收者来检测这个队列。

2. 发布/订阅模型

发布/订阅模型授权一个应用程序发送消息给多个应用程序。发布/订阅消息应用程序通过订阅一个主题来发送和接收消息。一个主题发布者（生产者）发送消息到一个指定的主题。一个主题订阅者（消费者）从这个指定的主题中接收消息。下面是发布/订阅模型的图解：



不象点到点模型，发布/订阅模型允许多个主题订阅者接收相同的一个消息。JMS 会保存这个消息，直到所有的主体订阅者都接收到它为止。

发布/订阅模型支持持久订阅者，允许你为主题订阅者指派一个名字，并将这个主题订阅者分配给一个用户或者一个应用程序。

3. 消息的持久性

消息可以被指定为持久的或非持久的。

一个持久的消息被保证至少能发送一次——在它被安全地写入文件或者数据库之前，不会被发送。WebLogic JMS 将持久的消息保存在一个持久性的存储备份中（文件或 JDBC 数据库）。这个备份存储是通过配置，指派给每个 JMS 服务器的。

非持久的消息不会被保存。除非系统故障导致消息丢失，否则它们被保证至少发送一次。如果一个连接被关闭或者复原，那么所有的还没有被确认的非持久的消息将会被重新发送。一旦一个非持久的消息被确认了，那么它将不会被再次发送。

二．WebLogic JMS 类

使用 javax.jms API 来创建 JMS 应用程序。它能帮助你创建连接 JMS 所需的类的对象，并发送和接收消息。JMS 类的接口被创建为提供公共父类的明确的队列化和主体化版本的子类。

下面的表格列出了会在后续章节中详细说明的 JMS 类。关于所有 JMS 类的完整说明，请看 javax.jms 包，weblogic.jms.ServerSessionPoolFactory 类，或者 weblogic.jms.extensions 类的文档。

JMS 类	说明
ConnectionFactory	封装连接的配置信息。连接工厂用来创建连接。使用 JNDI 来查找连接工厂。
Connection	描述消息通信系统的开放通信频道。连接用来创建会话。
Session	定义消息生产和消费的一系列命令
Destination	确定一个队列或主题。封装明确的提供者的地址。队列型和主题型目的分别管理从点到点和发布/订阅模型中发送来的消息。
MessageProducer and MessageConsumer	提供发送和接收消息的接口。消息生产者发送消息到队列或者主题。消息消费者从队列和主题中接收消息。
Message	封装被发送和接收的信息
SeverSessionPoolFactory *	封装消息消费者的一个服务－管理池的配置信息。服务器会话池工厂用来创建服务器会话池。
ServerSessionPool *	提供一个服务器会话的池，能够处理连接消费者的并发消息。
ServerSession *	联合 JMS 会话和一个线程。
ConnectionConsumer *	指定接收那些处理并发消息的服务器会话的消费者。

其中带*号的类表示支持处理多个并发消息的一个可选的 JMS 接口。

三．连接工厂

连接工厂对象封装了连接的配置信息，并授权 JMS 应用程序创建一个连接。系统管理员配置好连接工厂，以创建带有预定义属性的连接。

系统管理员定义并配置一个或更多的连接工厂，然后通过启动 WebLogic 服务器将它们加入 JNDI 空间。应用程序就可以使用 WebLogic JNDI 查找到一个连接工厂。

系统管理员也可以建立集群范围，通过配置多个连接工厂并用控制台中的目标选项将它们分配给若干 WebLogic 服务器，以便能够透明的访问集群中任一个 JMS 服务器的目的。每个连接工厂可以在多台 WebLogic 服务器上配置。

WebLogic JMS 定义了一个默认的连接工厂。可以用 JNDI 名来找到它：weblogic.jms.ConnectionFactory。你只需要在这个默认的连接工厂无法满足你的应用程序需求时，再去定义一个新的连接工厂。

注意：为了向后兼容，WebLogic JMS 仍旧支持两个已经不赞成使用的默认连接工厂。这些工厂的 JNDI 名字是：javax.jms.QueueConnectionFactory 和 javax.jms.TopicConnectionFactory。

连接工厂类并未定义方法。然而，它的子类针对各自的消息通信模型，定义了相应的方法。连接工厂支持并发使用，允许多个线程同时访问它。

下面的表格描述了连接工厂类的子类：

子类	消息通信模型	用来创建…
QueueConnectionFactory	点到点	指向 JMS 点到点提供者的队列连接
TopicConnectionFactory	发布/订阅	指向 JMS 发布/订阅提供者的主题连接

四．连接

连接对象描述了应用程序和消息通信系统之间的开放通信频道。它用来创建生产和消费消息的会话。连接创建了在应用程序和消息通信系统之间的，管理消息通信行为的服务器端和客户端对象。连接也提供用户认证。

连接通过连接工厂创建。连接工厂则通过 JNDI 查找获得。

由于认证用户和安装通信的资源开销，大多数应用程序为所有的消息通信建立一个单一的连接。在 WebLogic 服务器中，JMS 与其它通过客户端连接到服务器的 WebLogic 服务复用通信线路。JMS 不会创建额外的 TCP/IP 连接。Servlets 和其他服务器端对象也可以获得 JMS 连接。

缺省情况下，连接是在“停止模式”下创建的，即连接是不活动的。

连接支持并发使用，允许多个线程同时访问它。

下表描述了连接的子类。

子类	消息通信模型	用来创建…
QueueConnection	点到点	用来创建队列会话，由 QueueConnectionFactory 创建的指向 JMS 点到点提供者的连接构成。
TopicConnection	发布/订阅	用来创建主题会话，由 TopicConnectionFactory 创建的指向 JMS 发布/订阅提供者的连接构成。

五．会话

会话对象定义了消息生产和消费的一系列命令，并且能够创建多个消息生产者和消费者。同一个线程可以用来生产并消费消息。如果应用程序希望另开一个线程来生产和消费消息，那么这个应用程序应该为各个功能都创建一个会话。

会话由连接创建。

注意：会话以及它的消息生产者和消费者，在一个时间段，只能被一个线程访问。如果多个线程同时访问它们，将无法明确它们的行为。

下表描述了会话类的子类：

子类	消息通信模型	为…提供一个语境
QueueSession	点到点	由 QueueConnection 创建,为 JMS 点到点提供者生产和消费信息。
TopicSession	发布/订阅	由 TopicConnection 创建，为 JMS 发布/订阅提供者生产和消费信息。

1．非事务性会话

在非事务性会话中，应用程序创建的会话会选择下表中的 5 种确认模式之一：

确认模式	说明
AUTO_ACKNOWLEDGE	自动确认模式。一旦接收方应用程序的方法调用从处理消息处返回，会话对象就会确认消息的接收。
CLIENT_ACKNOWLEDGE	客户端确认模式。会话对象依赖于应用程序对被接收的消息调用一个 acknowledge()方法。一旦这个方法被调用，会话会确认最后一次确认之后所有接收到的消息。 这种模式允许应用程序以一个调用来接收，处理并确认一批消息。 注意：在管理控制台中，如果连接工厂的 Acknowledge Policy（确认方针）属性被设置为 ”Previous ”（提前），但是你希望为一个给定的会话确认所有接收到的消息，那么就用最后一条消息来调用 acknowledge()方法。
DUPS_OK_ACKNOWLEDGE	允许副本的确认模式。一旦接收方应用程序的方法调用从处理消息处返回，会话对象就会确认消息的接收；而且允许重复确认。 在需要考虑资源使用时，这种模式非常有效。 注意：如果你的应用程序无法处理重复的消息的话，你应该避免使用这种模式。如果发送消息的初始化尝试失败，那么重复的消息可以被重新发送。
NO_ACKNOWLEDGE	不确认模式。不确认收到的消息是需要的。消息发送给一个 NO_ACKNOWLEDGE 会话后，它们会被 WebLogic 服务器立即删除。在这种模式下，将无法重新获得已接收的消息，而且可能导致下面的结果：1．消息可能丢失；

	和（或者）另一种情况：2．如果发送消息的初始化尝试失败，会出现重复消息被发送的情况。
MULTICAST_NO_ACKNOWLEDGE	IP 组播下的不确认模式,同样无需确认。 发送给一个 MULTICAST_NO_ACKNOWLEDGE 会话的消息，会共享之前所述的 NO_ACKNOWLEDGE 确认模式一样的特征。 这种模式支持希望通过 IP 组播方式进行消息通信的应用程序，而且无需依赖会话确认提供的服务质量。 注意：如果你的应用程序无法处理消息的丢失或者重复，那么你应该避免使用这种模式。如果发送消息的初始化尝试失败的话，重复的消息可能会被再次发送。

译者注：在上表的 5 种确认模式中，AUTO_ACKNOWLEDGE，DUPS_OK_ACKNOWLEDGE 和 CLIENT_ACKNOWLEDGE 是 JMS 规范定义的，NO_ACKNOWLEDGE 和 MULTICAST_NO_ACKNOWLEDGE 是 WebLogic JMS 提供的。

2．事务性会话

事务性会话中，在任意给定的时间内，只有一个事务处于活动状态。任何通过事务发送或者接收的消息都被看作是一个原子单元。

当你创建了事务性会话，确认模式就被忽略了。当应用程序提交了事务，应用程序通过这个事务接收的消息都会被消息通信系统确认接收，通过这个事务发送的消息也都会被确认发送。如果应用程序回滚了事务，那么上面提到的消息都会被不确认和取消。

JMS 可以和其他使用 Java 事务 API 的 Java 服务一起参与分布式事务（distributed transaction），例如 EJB。由于事务性会话被限制，只能访问与此会话相关的消息，所以它不支持这个功能。（译者注：推荐使用 JTA 代替事务性会话。）

六．目的

一个目的对象可以是一个队列,也可以是一个主题。它封装了指定的接收者的地址语法。由于提供者之间语法的不同，JMS 规范并未定义一个标准的地址语法。

类似于连接工厂,系统管理员定义并配置目的,并启动 WebLogic 服务器,将它加入 JNDI 空间。应用程序也能够创建临时目的。临时目的仅在 JMS 连接的持续期间存在。

注意 系统管理员也能够配置多个物理目的,作为服务集群中单一的分布式目的的成员。更多信息参见下面的分布式目的小节。

在客户端，队列和主题对象其实是服务器端的对象的句柄。它们的方法只返回它们的名字。要进行消息通信而访问它们，你需要创建消息生产者和消费者来触发它们。

目的支持并发使用，允许多个线程同时访问它。JMS 队列和主题都继承于 javax.jms.Destination 类。下表描述了目的类的子类：

子类	消息通信模型	为…管理消息
Queue	点到点	JMS 点到点提供者
TemporaryQueue	点到点	JMS 点到点提供者，并在 JMS 连接的持续期间存在。临

		时队列只能通过创建它的队列连接而被消费。
Topic	发布/订阅	JMS 发布/订阅提供者
TemporaryTopic	发布/订阅	JMS 发布/订阅提供者，并在 JMS 连接的持续期间存在。临时主题只能通过创建它的主题连接而被消费。

注意：应用程序也可以选择通过创建在它的队列会话中的队列浏览者对象，以便浏览队列。浏览者对象在被创建时，生产一个当时队列中消息的快照。应用程序能够查看队列中的消息，当时消息不能被标记为“已读”或者从队列中删除。

七．分布式目的

系统管理员可以配置多个物理目的，作为 WebLogic 服务器集群中的单一的分布式目的的成员。一旦适当地配置后，你的生产者和消费者能够发送和接收分布式目的。WebLogic JMS 可以在分布式目的中，跨越所有有效的目的成员，分布地加载消息。

八．消息生产者和消息消费者

消息生产者对象发送消息到队列或者主题。消息消费者对象从队列或主题中接收消息。消息生产者和消费者彼此独立操作。消息生产者产生并发送消息，与消息消费者是否创建并等待消息无关，反之亦然。

会话创建匹配队列和主题的消息生产者和消息消费者。

消息发送者和接收者对象被创建为消息生产者和消息消费者类的子类。下表描述了消息生产者和消息消费者类的子类。

子类	消息通信模型	完成的功能
QueueSender	点到点	发送消息给 JMS 点到点提供者
QueueReceiver	点到点	从 JMS 点到点提供者那里接收消息,并且在 JMS 连接关闭前一直存在。
TopicPublisher	发布/订阅	发送消息给 JMS 发布/订阅提供者
TopicSubscriber	发布/订阅	从 JMS 发布/订阅提供者那里接收消息，并且在 JMS 连接持续期间一直存在。消息目的必须用合适的 JNDI 接口显性地绑定。

点到点模式，允许多个会话从同一个队列中接收消息。然而，一条消息只能被发送给一个消息接收者。当有多个消息接收者时，WebLogic JMS 以先到先服务的原则，指定下一个消息的接收者。

发布/订阅模式，允许消息被发送给多个主题订阅者。主题订阅者可以是持久的或非持久的。

应用程序可以使用同一个 JMS 连接，对同一个主题进行发布和订阅。因为主题消息会

被发送给所有的订阅者，应用程序可以接收到它自己发布的消息。为了预防客户端接收到它们自己发布的消息，JMS 应用程序可以在主题订阅者中设置一个非本地属性。

九．消息

消息对象封装了应用程序之间的交流信息。这些信息包括了三个组成部分：一组标准的头部域，一组应用程序定义的属性，还有一个消息体。下面的小节描述了这些组件。

1．消息头部域

每条 JMS 消息包含了一组可以被消息消费者利用的标准的头部域。消息生产者可以设置其中一些域。

下表描述了消息头部域，以及每个域的值是如何被定义的。

域	说明	用…定义
JMSCorrelationID	指定下列之一：一个 WebLogic JMSMessageID（稍后会在表格中说明），一个应用程序指定的字符串，或者是一个 byte[]数组。JMSCorrelationID 用来使消息相关联。 两种普通的应用程序适合用这个域。第一种应用程序通过设置请求/响应计划来链接消息，如下： 1．当一个应用程序发送了一条消息，它 会保存分配给它的 JMSMessageID。 2．当另一个应用程序接收了这条消息，它会回复一条响应消息给发送者。在响应消息中，它会将这个 JMSMessageID 拷贝进响应消息的 JMSCorrelationID 域中。 第二种应用程序可以使用 JMSCorrelationID 域来携带你选择的任意字符串，允许以应用程序决定的值来将一系列消息链接起来。	应用程序
JMSDeliveryMode	指定持久性或非持久性的消息通信。 当一条持久性的消息被发送后，WebLogic JMS 将它保存在 JMS 文件中或者是 JDBC 数据库中。在消息的发送被确认之前，send()操作不会被认为成功。持久性消息保证被至少发送一次。 WebLogic JMS 不会在 JMS 数据库中存储非持久性消息。这种操作模式提供了最低的开销。除非系统故障导致消息丢失，否则消息会至少被发送一次。如果连接被关闭或复原，所有仍未被确认的消息都会被重发。一旦非持久性的消息被确认，它就不会再被重发。 消息被发送时，这个域的值忽略不计。当消息被接收时，消息中会包含发送方法指定的发送模式。	send()方法
JMSDeliveryTime	定义消息被发送给消费者的最早的绝对时	send()方法

	间。这个域可以用来对目的中的消息进行排序，并检索消息。出于数据类型转换的考虑，JMSDeliveryTime 是一个长整型数据。	
JMSDestination	指定消息被发送的目的（队列或者主题）。当消息被发送时，应用程序的消息生产者设置这个域的值。	send()方法
JMSExpiration	指定消息的期限或生存时间值。WebLogic JMS 用当前格林尼治标准时间（GMT）与应用程序的生存时间之和，计算出 JMSExpiration 的值。如果应用程序指定了生存时间为 0，那么 JMSExpiration 也被设置为 0，也就意味着消息永不过期。 为防止过期的消息被发送，WebLogic JMS 会删除它们。	send()方法
JMSMessageID	包含一个字符串值，它唯一地标识了经由一个 JMS 提供者发送的每一条消息。所有的 JMSMessageID 都以前缀 ID:开始。当消息被发送时，这个域的值被忽略。当消息被接收时，这个域包含了提供者分配的值。	send()方法
JMSPriority	指定优先权级别。这个域在消息发送之前被设置。JMS 定义了 10 种优先级，0 到 9，0 是最低的优先级。0 到 4 属于普通优先级的级别，5 到 9 属于畅通优先级。 当消息被接收时，它包含了发送这条消息的方法所指定的这个域的值。 通过配置目的键，可以按优先级来对目的进行排序。	消息消费者
JMSRedelivered	当消息由于没有接收到确认而被重发时，指定一个标记位。这个标记仅仅针对于接收消息的应用程序。 如果设置了这个域，那么这个标记说明了在这之前，JMS 可能已经发送了这条消息，原因如下： ● 应用程序已经接收到了这条消息，但是还没有确认它。 ● 在最后一条被确认的消息之后，为了重起会话，这个会话的 recover()方法被调用。	WebLogic JMS
JMSReplyTo	指定一个队列或主题，以便回复那些应该被发送的消息。在消息被发送之前，这个域被发送方应用程序设置。 这个特性可以和头部域 JMSCorrelationID 一起被用来协调请求/响应消息。 简单地设置 JMSReplyTo 域，不能保证会有响应。如果这个域如此设置的话，它仅仅授权接收方应用程序响应。	应用程序

	你可以把 JMSReplyTo 设置为空，它也许对接收方应用程序是有意义的，比如可能表示是一个通知事件。	
JMSTimeStamp	包含了消息被发送的时间。并非在应用程序发送消息时，而是当 WebLogic JMS 接受了一条消息以便发送时，它会将时间标记写入消息中。 当消息被发送时，它包含了这个时间标记。 这个域中的值是一个以微秒为单位的 Java 的时间值。	消息消费者
JMSType	指定发送方应用程序所设置的消息类型标识符（字符串型）。 为了兼容不同的 JMS 提供者，JMS 规范允许这个域具有一些适应性。一些消息通信系统允许使用应用程序规定的消息类型。对于这些系统，JMSType 域可以被用来保存一个消息类型 ID。这个消息类型 ID 提供了访问已存储型目的的通路。 WebLogic JMS 并不约束这个域的使用。	应用程序

2. 消息属性域

消息的属性域包含了发送方应用程序额外增加的头部域。这些属性都是标准的 Java 名/值对。属性名必须遵守消息选择器的语法规范。这些语法规范定义在 javax.jms.Message 文档中。下列类型的值是有效的：boolean，byte，double，float，int，long，short 和 String。

尽管消息属性域可以被用来实现应用程序指定的意图，但是 JMS 规定它们主要在消息选择器中使用。

3. 消息体

消息体包含了从生产者发送到消息者的内容。下表说明了 JMS 定义的消息类型。所有的消息类型都继承于 javax.jms.Message 类。Message 类由消息头部和属性组成，不包含消息体。

类型	说明
javax.jms.BytesMessage	未解释的字节流，它必须被发送者和接收者所理解。这种消息类型的访问方法是基于 java.io.DataInputStream 类和 java.io.DataOutputStream 类的面向流的 readers 和 writers 类。
javax.jms.MapMessage	一套名/值对，名是字符串型，值是 Java 基本类型。通过指定一个名，名值对可以顺序地或随机地读取。
javax.jms.ObjectMessage	单个的串行化的 Java 对象。
javax.jms.StreamMessage	除了只能读取和写入 Java 基本类型之外，它类似于 BytesMessage。
javax.jms.TextMessage	单个的字符串。它也能包含 XML 内容。
javax.jms.extendsions.XMLMessage	XML 内容。XMLMessage 类型的使用，使

	得消息过滤非常便利。而消息过滤在操作 TextMessage 中的 XML 内容时是相当复杂的。

十．服务器会话池工厂

服务器会话池是 WebLogic JMS 定义的 JMS 特性。它允许你同步地处理消息。服务器会话池工厂用来创建服务器范围的服务器会话池。

WebLogic JMS 默认地定义了一个服务器会话池工厂对象：weblogic.jms.ServerSessionPoolFactory:<name>。在<name>中指定创建服务器会话池的 JMS 服务器。WebLogic 服务器在启动时将这个默认的服务器会话池工厂加入 JNDI 空间。随后，应用程序根据 WebLogic JNDI 获得这个服务器会话池工厂。

十一． 服务器会话池

服务器会话池对象,提供了一个服务器会话的池,以便连接消费者能够同步地处理消息。

服务器会话池由服务器会话池工厂对象创建。后者通过 JNDI 查找获得。

十二． 服务器会话

服务器会话对象允许你将一个线程和一个通过提供语境创建的 JMS 会话联合起来，发送并接受消息。

服务器会话由服务器会话池对象创建。

十三． 连接消费者

连接消费者对象使用服务器会话来处理接收到的消息。如果消息通信量繁重，连接消费者可以通过每个服务器会话加载多个消息，最小化线程语境的切换。

附：本篇术语表（按英文字母表顺序排列）

中文	英文
地址语法	address syntax
管理控制台	administration console
兼容	accommodate
原子单元	atomic unit
存储备份	backing store
向后兼容	backwards compatibility
父类	base class
绑定	bind
调用	call
特征	characteristic
提交	commit
公共父类	common parent class
并发	concurrent

连接消费者	ConnectionConsumer
消费者	consumer
语境	context
协调	coordinate
数据类型转换	data type conversion
不赞成	deprecate
目的键	destination key
分布式目的	Distrubuted Destination
分布式事务	distrubuted transaction
重复确认	duplicate acknowledge
持久订阅者	durable subscriber
持续期间	duration
封装	encapsulate
畅通的	expedited
显性地	explicitly
继承	extend
先到先服务	first come first server
标记位	flag set
适应性	flexibility
句柄	handle
头部域	header field
标示符	identifier
初始化尝试	initial attempt
Java 事务 API	JTA
监听	listen
长整型	long integer
消息通信行为	messaging activity
消息通信模型	Messaging Model
消息体	message body
消息过滤	messsage filtering
消息选择器	message selector
消息通信量	message traffic
名/值对	name/value pair
非本地	nolocal
非持久的	non-persistent
非事务性会话	non-transacted session
通知事件	notification event
持久性	persistence
持久的	persistent
物理目的	physical destination
开放通信频道	open communication chanel
可选的	optional
预定义	predefined

基本类型	primitive type
优先权级别	priority level
生产者	producer
属性域	property field
提供者	provider
点到点模型	PTP Model
点到点提供者	PTP Provider
发布/订阅模型	Pub/Sub Model
发布/订阅提供者	Pub/Sub Provider
服务质量	quality of service
队列浏览者	QueueBrower
队列连接	QueueConnection
队列会话	QueueSession
队列发送者	queue sender
队列接收者	queue receiver
接收者	receiver
复原	recover
回复	reply
资源开销	resource overhead
资源使用	resource usage
重起	restart
约束	restrict
回滚	roll back
串行化	serializable
服务器会话	ServerSession
服务器会话池	ServerSessionPool
服务器会话池工厂	ServerSessionPoolFactory
服务集群	server cluster
服务－管理池	server-managed pool
服务器范围	server-wide
会话确认	session acknowledge
快照	snapshot
启动	startup
已存储型目的	stored type destination
切换	switch
语法规范	syntax specification
临时目的	Temporary Distination
时间标记	timestamp
生存时间	time-to-live
主题连接	TopicConnection
主题会话	TopicSession
主题发布者	topic publisher
主题订阅者	topic subscriber

事务性会话	transacted session
用户认证	user authentication
唯一	unique
未解释的	uninterpreted