Fiorano
Enabling change at the speed of thought

www.fiorano.jp

フィオラノ ソフトウェア 株式会社
東京都港区北青山 3-6-7
パラシオ タワー 11 階
Tel. (03) 5464 8743
Fax. (03) 5778 7676
e-mail: info_jp@fiorano.com

Fiorano
Enabling change at the speed of thought™

ホワイト ペーパー

JMS2.0 における主要な改善点

JMS 2.0 に対応した FioranoMQ 10.0

はじめに

FioranoMQ 10 は、2002 年の JMS 1.1 から 10 年以上を経て改定となった JMS 2.0 に業界で初めて 100% 準拠した JMS サーバー製品となりました。

このホワイトペーパーでは、JMS 2.0 における代表的な改善点について説明しています。

1. JMS 2.0 における主要な改善点
    1.1 Simplified API (簡易 API)
    1.2 遅延メッセージ配信
    1.3 共有サブスクリプション (Shared Subscription)
    1.4 メッセージの非同期送信
    1.5 セッション生成の新規メソッド
    1.6 新規 createDurableConsumer メソッド
    1.7 AutoCloseable のサポート
    1.8 起動プロパティ Activation properties

2. 共有サブスクリプション (Shared Subscription)

1 JMS 2.0 における主要な改善点

JMS 2.0 における代表的な改善点について、以下の各節で説明します。

1.1 Simplified API (簡易 API)

JMS 2.0 で新規に簡易 API が設けられました。これによって、スリムなコーディングとなり必要なコード量が劇的に削減されます。

Simplified API として、新規に 3 つのオブジェクトが導入されました。

JMSContext　　JMSProducer　　JMSConsumer

これらのオブジェクトは、JMS 1.1 の Connection、Session、MessageProducer、MessageConsumer オブジェクトの機能を有しています。

Simplified API は、完全に後方互換性があり、旧コードを使い続けたまま、新規コードに Simplified API を使うことができます。

その他、以下のメッセージ処理および例外処理の新規クラスが追加されました。

javax.jms.CompletionListener

javax.jms.IllegalStateRuntimeException

javax.jms.InvalidClientIDRuntimeException

javax.jms.InvalidDestinationRuntimeException

javax.jms.InvalidSelectorRuntimeException

javax.jms.JMSRuntimeException

その他

1.2 遅延メッセージ配信

JMS 2.0 では、メッセージのセンダー側 (プロデューサ) でメッセージ配信をスケジュールすることができるようになりました。

これにより、一日の終わりに配信するといった遅延処理をサポートすることができるようになります。

1.3 共有サブスクリプション (Shared Subscription)

JMS 2.0 から共有サブスクリプションが導入されました。これは、トピックの個々のサブスクライバーに複数のコンシューマを指定できるようにするものです。

ただし、これらの複数のコンシューマの間でメッセージをサブスクライブできるコンシューマは 1 つだけです。

こうすることで、トピックのメッセージ受信による処理を複数のコンシューマ間で分散させることができるようになります。ここでいう分散処理とは、メッセージ 1 の処理をコンシューマ A に、メッセージ 2 の処理をコンシューマ B に振り分けて処理することを指します。

共有サブスクリプションは、継続サブスクリプションおよび非継続サブスクリプション毎に設けられています。

共有サブスクリプションの詳細については、2 章を参照ください。

1.4 メッセージの非同期送信

JMS 1.1 では、メッセージのセンダー側 (プロデューサ) はサーバーから ACK が返ってくるまでブロックする同期モードで動作していましたが、JMS 2.0 ではメッセージを非同期に送信するメソッドが追加されました。

このメソッドでは、サーバーからの ACK を待たずに直ちに次のオペレーションに移行できます。ACK を受信すると、非同期コールバックがインボークされ、送信完了の処理を非同期に行うことができます。

1.5 セッション生成の新規メソッド

セッションを生成するためのメソッドが javax.jms.Connection に追加されました。

createSession(int sessionMode)

このメソッドは、従来の createSession(boolean transacted, int acknowledgeMode) と同じ機能を果たしますが、コードを簡素化するために、アーギュメントが 1 つになりました。

createSession( )

このメソッドは、JTA トランザクションが稼働している Java EE web または EJB コンテナで使用することを目的としたものです。

このような環境では、createSession(int sessionMode) メソッドで指定したセッション モードが無視されるためです。

1.6 新規 createDurableConsumer メソッド

javax.jms.Session インタフェースに MessageConsumer を返す createDurableConsumer メソッドが追加されました。

従来の createDurableSubscription メソッドがドメイン スペシフィックな TopicSubscriber が返されるため、これを避けるために使用します。

1.7 AutoCloseable のサポート

下記の各インタフェースが、java.lang.Autocloseable インタフェースをサポートするよう変更されました。

Connection **Session** **QueueBrowser**

MessageProducer **MessageConsumer**

これによって、アプリケーションが Java SE 7 try-with-resources ステートメントを使用してこれらのオブジェクトを生成できるようになり、明示的に cloase ( ) を呼び出す必要がなくなりました。

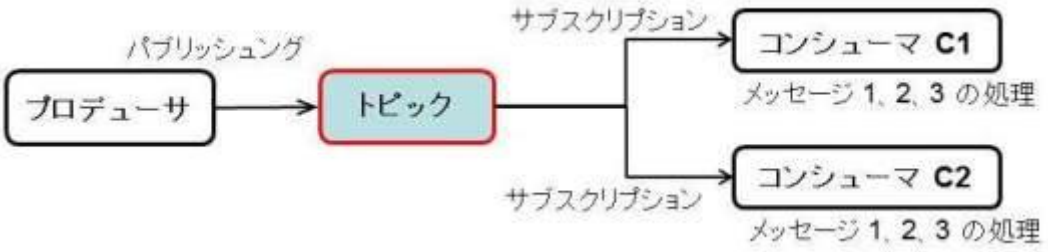
1.8 起動プロパティ Activation properties

JMS メッセージ ドリブン ビーン (MDB: Message-Driven Bean) を使用する場合の起動プロパティ (MDB activation properties) に追加およびオプション化が実施されました。(Java EE Connector Architecture specification, version 1.6).

2. 共有サブスクリプション (Shared Subscription)

共有サブスクリプション (Shared Subscription) は、JMS 2.0 で採用された最も重要な機能の 1 つです。

従来の JMS 1.1 における パブリッシュ - サブスクライブ モデルでは、トピックをサブスクライブ (購読) するすべてのサブスクライバー (コンシューマ) に同一のメッセージが配信されます。(下図参照)

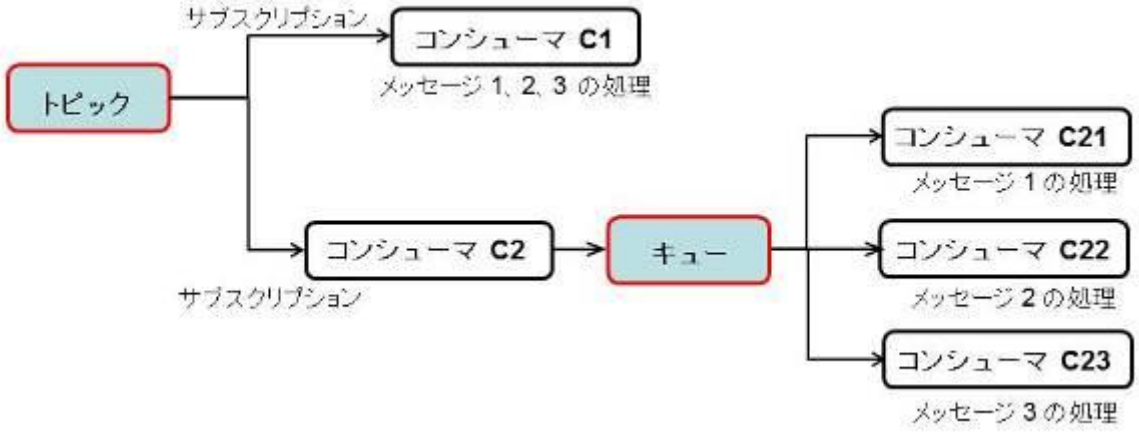


通常の パブリッシュ - サブスクライブ モデル

ここで、受け取ったメッセージに対する処理がコンシューマ C1 とコンシューマ C2 では異なるものとします。

コンシューマ C2 におけるメッセージ処理にたいへん時間がかかるものと仮定します。次々とパブリッシュされるメッセージを滞りなく処理するためには、メッセージを並行して (あるいは複数で分散して) 処理したくなります。

この課題に対する JMS 1,1 のソリューションの 1 つの例として下図のように間にキューを配置することが考えられます。



JMS 1.1 によるソリューション例

しかしながら、上図に示すソリューションでは、不都合が生じてしまいます。例えば、

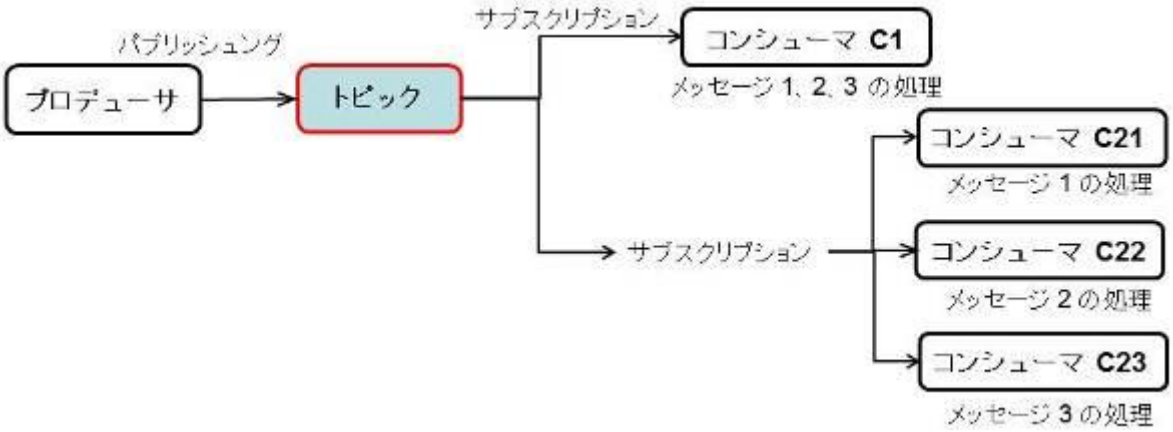
キューを配置することによるトポロジーの複雑化とそれに伴うシステム管理の負荷増大

トピックから直接メッセージを受け取るのではなくキューを経由することによる配信パフォーマンスの低下

実際にトピックをサブスクライブするコンシューマの実数が把握できない

などが挙げられます。

JMS 2.0 の共有サブスクリプションによって、1 つのサブスクリプションに複数のコンシューマを指定できるようになり、上述の不都合を解消するかたちで、メッセージ処理のスケールアウトが可能になりました。(下図を参照)



JMS 2.0 共有サブスクリプション

共有サブスクリプションによるソリューションでは以下の利点を得ることができます。

- すべてのコンシューマがトピックから直接メッセージを受け取れる -- パフォーマンスの低下を招かない
- キューの追加など新たなリソースを必要としない -- システム管理の負荷増大を招かない